25/6/2012 N/M.

Oracle - 11g

RS = 130/=

Oracle 11g released 2007.

ORACLE (Objected Relational DBMS)

→ to store data in 2ndary storage devices permanently.

→ It is a product from Oracle Corporation.

Table with primary key is called master table & table with foreign key called child table.

It is not possible to delete any data from master table.

In order to delete the data first we need to delete from the data from child table then we can delete master table.

on delete cascade clause can be used to delete the data from all the child tables.

In Oracle Nvl replaces null value with user defined value

Oracle is an object relational DBMS product which is used to stored data permanently in hard disk

All business activities deals with a lot of data

Data : It is a collection of raw-facts.

Ex: student marks
customer names

Information : when we are processing data we achieve meaningful results called information

Ex: mark sheet
invoice of a customer.

Data Store : It is a place where we can store data

Ex: books & papers
flat files
databases

* Flat files : traditional mechanism used to store data or information

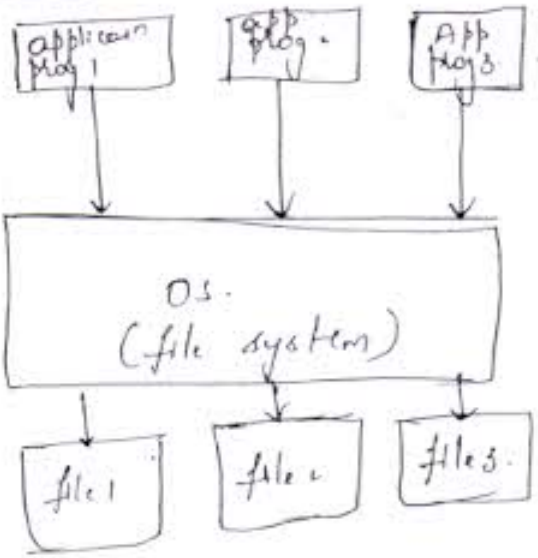
disadvantages

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

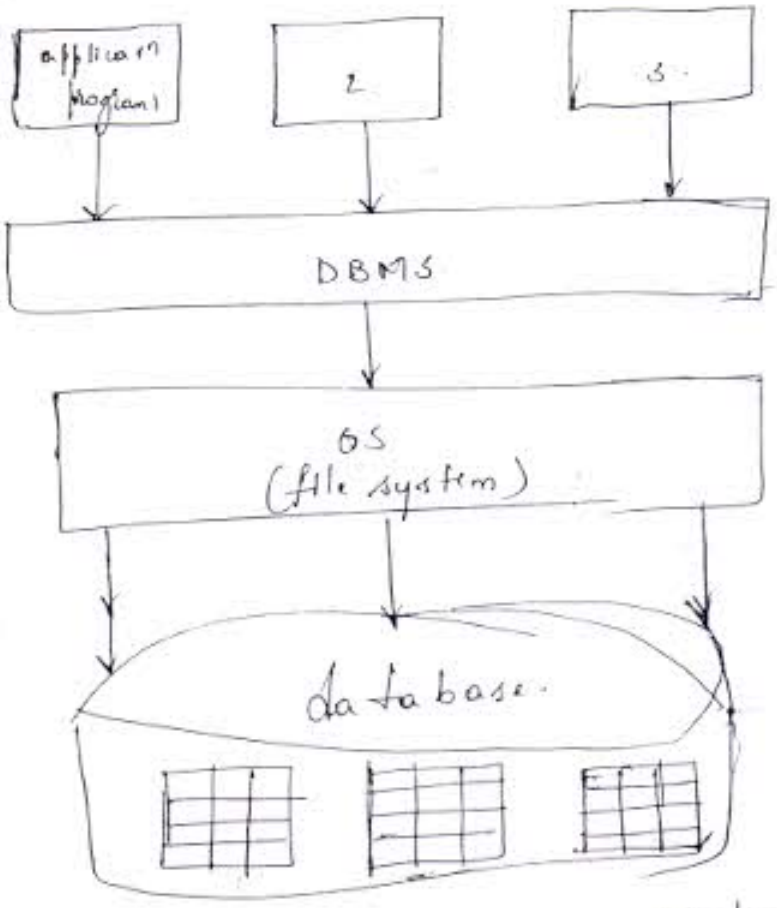
26/6/2017

1) Data retrieval : To retrieve data from database, we are using Sequel language whereas if we want to retrieve data from flat files we must develop application programs using high level languages.

Flat file mechanism :



Database :



Data redundancy : Sometimes we maintain multiple copies of the same data in different locations. This type of duplicate data is also called as redundant data

In flat files whenever a data is modified in one file, it is not affected in another file thats why flat files dosnot maintain Constant state

Whereas in database if the data is modified in one file automatically it is affected in another file & also database automatically maintain Constant state using transaction

Rollback : undo from mm area.

27/6/2012

Data integrity : if we want to insert proper data into database we are defining set of rules according to client requirement. These set of rules are also called as business rules. In database we are defining business rules using two methods
1.) Constraints 2.) Triggers.

Data Security : flat files data cannot be secured because flat files doesnot provide security mechanism whereas data bases provides role based security.

Data Indexing : if we want to retrieve data very fastly databases uses indexing mechanism

Data base : It is an organised collection of interelated data.

DBMS : It is a collection of programs (s/w) written to manage database

DBMS architecture : most commercial databases uses 3 level architecture. This architecture contains
1) Conceptual view
2) Physical view
3) External view

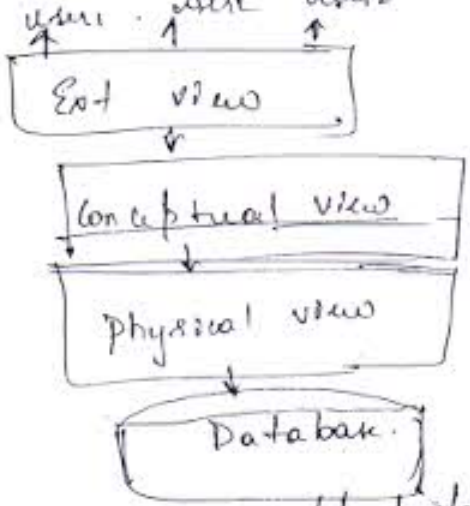
This architecture is also called as ANSI spark architecture (standard planning & requirement committee)

Conceptual view : This view describes enterprise view of the database.

Ex : CREATE TABLE emp(empno (10), ename varchar2(10), sal number (10), deptno number (10));

External view : If we want to access portion of the data within database, database administrator creating views & giving to the number of users. Number of users only perform operations on the view automatically base tables affected.

Physical view : it describes physical files in a database, in this files only actual data is stored.



Data model : it is an conceptual tool which describes data & data relationships

1) hierarichal data model
2) n/w data model
3) Relational data model

Hierarical data model : In this model we are storing data in the form of records & also record type is also same as tables in relational model. This model implemented based on 1 to many relationships. i.e every child has only one parent thats why in this model child segments are referred This model has lot of redundancy and also if we want to retrieve data from database we must use root node thats why this model retrieve data very slowly

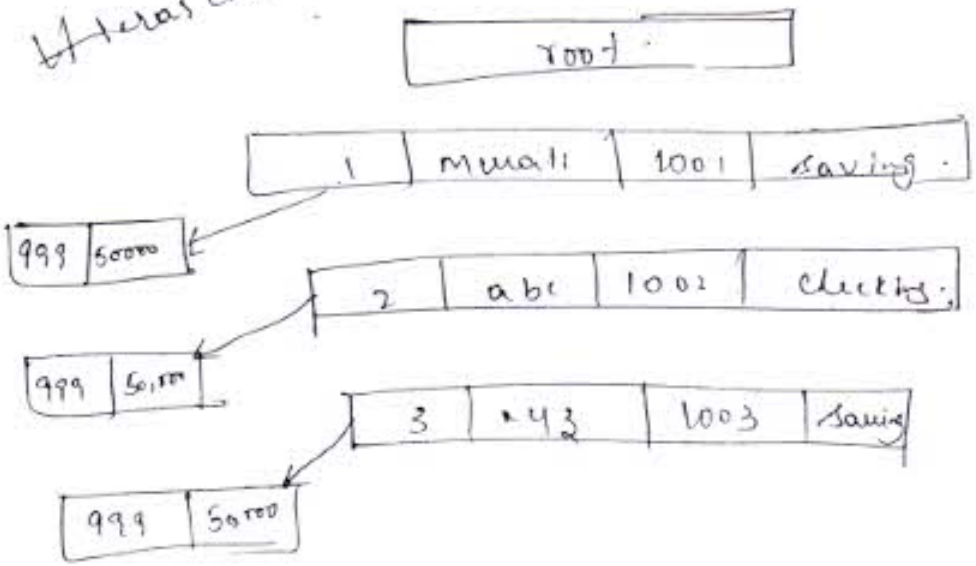
Eg

In 1960 IBM introduced IMS product based on hierarchical data model

Cust no.	Cust name	Acc no.	Acc type
1	Murali	1001	saving
2	abc	1002	checking
3	xyz	1003	saving

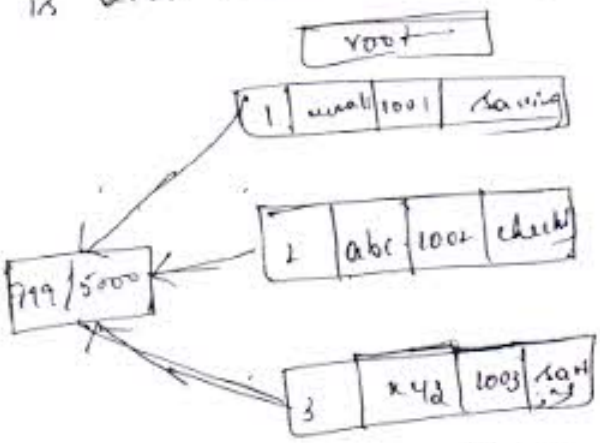
loan id	amount
999	50000
888	60000
777	7000

Hierarchical model



N/w data model : In 1969 first specification of n/w data model introduced by CODASYL (Conference on data syst languages) Committee

This model implemented based on many to many relationsh
-ips
when a database contains many to many relationships automatically reduces redundancy because in this model child segments are not repeated. In this modell also data stored in the format of records and also record type is also same as table in relational model.



In 1969 IBM introduced IDMS (Integrated data management system) product based on n/w data model.

28/6/2017

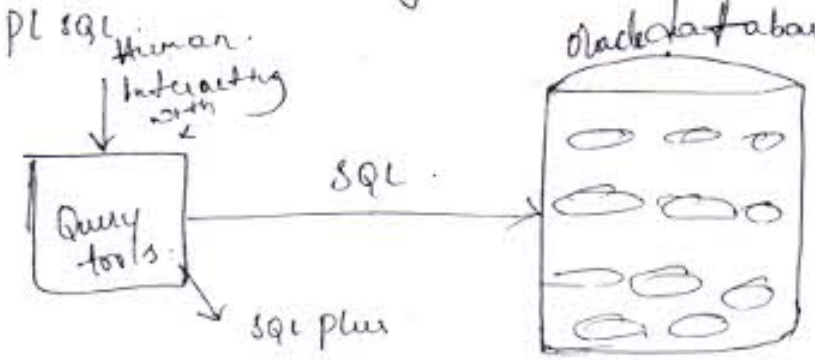
Relational data model:-

→ In this data model we are storing data in two dimensional tables (rows & columns)

→ In 1970 E.F. CODD wrote a paper "Relational model of data for large shared data banks" In this paper E.F. CODD designed first three normal forms and also introduced dsl/alpha language which is used to communicate relational databases

Based on this paper IBM introduced first relational database produ -ct system/8.

→ we cannot communicate directly with the database, it is possible only using some query tools like (sql plus, toad,



toad
(Enterprise edition)

→ from oracle log for the station e

username [Scott]

password [Tiger]

Error - account locked.

to overcome the above error, to unlock the a/c.

username [1 sys as sysdba]

Password [sys]

SQL> Alter user Scott a/c unlock;

SQL> Conn Scott/tiger;

Pswrd: tiger

Conpswrd: tiger

start → programs - oracle 11g - application development - SQL plus

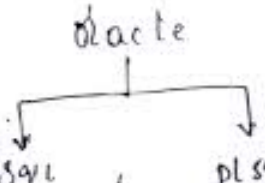
→ to clear the screen cl scr.

(or)

shift + delete

To view all table.

sql > select * from table;



Sql (Structured query language):-

→ DDL (data definition language):
  - → create
  - → alter
  - → drop
  - → truncate
  - → Rename (oracle 9i)

→ DML (data manipulation language):
  - → insert
  - → update
  - → delete
  - → Merge (oracle 9i)

→ Data retrieval language (or) data query language (DQL)
  - Select

→ Transactional Control language (TCL)
  - → Commit
  - → Roll back

→ Data Control language (DCL)
  - → Grant
  - → Revoke

Data types

data type identifies
type of data in a column

- Number(p,s) — p - Precision (total number of digits); s - scale
- char → varchar2 (max size)
- Date

Number : It is used to store fixed, floating point numbers

Syntax : Columnname data type (size)

Max limit of precision is upto 38 digits

Char : it is used to store fixed length alphanumeric data in bytes

Max limit is upto 2000 bytes

Character data type by default has 1 byte.

Syntax : columnname char (size)

VARchar2 (max size) : used to store variable length alphanumeric data in bytes max limit is 4000 bytes.

This is introduced in oracle (7.0) Before oracle (7.0) used varchar ~~this~~ ~~date~~ This varchar supports upto 2000 bytes.

Date : it is used store dates in oracle date format

Syntax : Columnname date.

By default, oracle date format is

DD-MON-YY

DDL : these commands are used to define structure of table.

CREATE : It is used to create database objects like tables, views, indexes, etc.

Creating table :

Creating table :

Syntax : Create table tablename (col1 datatype (size), col2 datatype (size) ...);

Max limit of columns are 1000.

Create table j1 (sno. number (10), name varchar2 (10));

to view the structure of the table

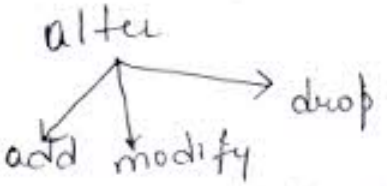
desc tablename ;

Ex : desc j1 ;

to add a drop or modify the data type we can use Alter

27/6/2012

Alter : It is used to change existing table structure.



ADD : It is used to add no of columns into the existing table.

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

Eg.

SQL > alter table j1 add sal number (10);

modify : It is used to change column data type (or) Column datatype size

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

Sql > drop table j1 ;

flashback table j1 to before drop ;

desc j1 ;

drop permanently

sql > drop table j1 purge ;

flashback table j1 to before drop ;

error : object not in recycle bin

4.) Truncate (permanently removed)

It is used to remove tabbel data from a table and Columns remain same

Syntax : truncate table tablename ;

Ex : sql > create table j1 as select * from emp ;

sql > truncate table j1 ;

sql > select * from j1 ;

no rows selected

sql > desc j1 ;

Renaming a table

syntax : rename old-tablename to new tablename

sql > rename j1 to j2 ;

Renaming a Column : only oracle 9.2 onwards it will works.

syntax : alter table tablename rename Column oldcolumn to newcolumn name ;

Eg : sql > alter table emp rename ~~ename~~ Column empno to sno ;

Note :

By default all DDL Commands are automatically Commited

Enter value for name : efg

Method3 : Skipping Columns.

To insert particular Columns.

Syntax :

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

To insert only name Column.

SQL> insert into tablename (name) values (Sahasra);

o/p

Sno	name
1	Abbaji
2	xyz
3	abc
4	efg.
	Sahasra.

Update : It is used to change data in a table.

data - update
schema - alter.

Syntax : update tablename
set Columnname : new value where
where Columnname : existing value;

Ex: SQL> update emp set ENAME : 'SUNDAY' where ENAME : 'SMITH';

To insert a particular cell value

SQL > update J1 set Address : 'Mumbai' where
Name : "Sand 'xyz';

To suppress the cell a particular cell

update J1

To add a Column :

sql > alter table J1 add address varchar2 (10);

```
Sql > update j1 set address: 'mumbai'
      where name : 'x43';

Sql > update j1 set address: null
      where address: 'mumbai';
```

Delete:

It is used to delete all rows of particular rows from a table.

Syntax:

delete from tablename.; (to delete all rows)

delete from tablename where Condition;
(to delete particular rows)

to delete all the rows from a table

```
delete from j1;
```

Note: For deleting all the rows but columns exist we use truncate & delete but in delete roll back is possible, while not in truncate as truncate is a ddl command as after completing the task it is saved in ddl commands.

```
Sql > delete from j1;
Sql > rollback;
Sql > select * from j1;
```

Whenever we are using delete from tablename, truncate table tablename all rows are deleted and columns remains same, but when we are using delete from table name deleted data automatically stored in a buffer, that's why we are getting back the data using roll back

where as in truncate table tablename we cannot get back the data

2/2/2012

Data retrieval language (or) Data query language:

→ Select:

Syntax: Select col1, col2 ....
from table name
where condition
group by column name
having condition
order by column name [asc / desc];

→ Creating a table from another existing table

Syntax: Create table tablename as Select * from existing tablename;

Ex:
Sql> Create table j1 as Select * from emp;
Sql> Select * from j1;

To copy only columns creating new table without copying data

Syntax:
Create table tablename as Select * from another tablename where false condition;

Ex: Create table j2 as Select * from emp
Where 1=2;

```
Sql> Select * from jo ;
     no rows selected
Sql> desc jo ;
```

1.) Select all columns & all columns.

2.) Select all col. & particular rows. (where clause : true)

3.) Select particular cols & all rows. (where)

4.) Select particular rows & particular cols.

Operators used in Select Statement

1.) Arithmatic operators (*, /, +, -) ← used in select col1, col2 ....

2.) Relational operators (or) Comparison (=, <, <=, >, >= <> or !=)' ← used in where clause

3.) Logical operators (AND, OR, NOT) ← used in where clause

4.) Special operators. ← used in where clause

→ Arithmatic operators are used in number, Date, datatype Columns.

Ex: Sql> SELECT ENAME, SAL, SAL * 12 "Annual salary"
FROM Emp ;

(to display Hole string values we declare in " ")

Query: Display the employees except clerks using job column from emp table

SELECT * from emp where job != 'CLERK' ;

```
Select * from emp
where JOB <> 'CLERK';
```

Q) Display the employees who are getting more than 2000 salary

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

for checking both sal > 2000 & job: clerk

```
Select * from emp
  where JOB = 'CLERK'
  AND  SAL > 2000;
```

to check each individual we use OR operator.
When we are using OR operator it displays the sal irrespective of the job.

```
Select * from emp
  where JOB: 'CLERK'
  OR  SAL > 2000;
```

Q.) Display the employees who belongs to 20, 30, 50, 60, 80, 90 dept numbers from emp table.

```
Select * from emp
where DEPT No = 20 OR
 DEPT No = 30 OR
 DEPT No = 50 OR
 DEPT No = 60 OR
 DEPT No = 80 OR DEPT No = 90.
```

Special operators :

1.) IN	NOT IN
2.) between	not between
3.) Is Null	Is not null.
4.) Like	not like

→ IN : It is used to pick the values one by one from list of values, generally, in place of OR operator we can use IN operator because IN operator performance is very high compared to OR operator

Syntax :

```
Select * From tablname
  where Columnname in (list of values);
```

Ex : Select * from emp (Any data type)
  where deptNo in (10, 30, 50, 60, 80, 90);

```
Select * from emp
  where ENAME in ('SMITH', 'KING');
```

Note: NOT IN operator does not work with null values

Ex : Select * from emp
  where deptNo NOT IN (10, 10, Null)

Result : No rows selected

8/7/2012

Between :

This is operator returns range of values in a column this operator is also called as between and operator.

Syntax :

Select * from tablename where columnname between low value AND high value.

Eg : Select * from emp where sal between 2000 & 5000

Null :

Null is an unknown, unavailable, undefined value it is not same as zero.

Any arithmetic operations performed on Null values again it will become null

Eg : null + 10 → null

Query :

1) Display ename, sal, commition sal + comm of employee SMITH from emp table

Select ename, sal, comm, sal+comm from emp

where ename : 'SMITH'

O/P

ename	sal	Comm	sal + Comm
SMITH	2000	–	–

To overcome this problem oracle introduced NVL() function.

NVL() :

This is a predefined function which is used to replace a substituted user defined value in place of Null

Syntax:

Nvl (exp1, exp2);

Here these two expressions must belong to same data type, always oracle will check first exp. If first exp is null then it returns second exp; otherwise it returns first exp

1) nvl (null, 30)
   → 30
2) nvl (60, 20)
   → 60

Solution:

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

O/P:

ename	sal	comm	sal + nvl (comm,0)
SMITH	2000	-	2000

Sal + NVL (comm, 0)

=> 2000 + Nvl (null, 0)

=> 2000 + 0

= 2000

is Null, Is not null:

These operators are used in where clause only, these operators are used to test whether a column having null values & not

Note: Generally we are not allowed to use =null in where clause in place of this one we are using "is null"

syntax:

Select * from tablename where Columnname is null;

syntax:

select * from tablename where Columnname is not null;

Query:

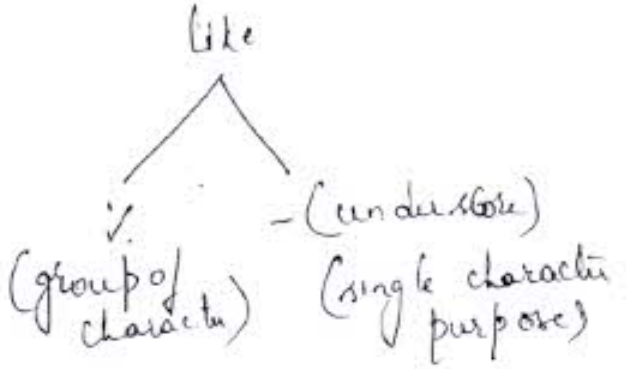
1) display the employees who are not getting Commition from emp table

Select * from emp where Comm is null

2) display the employees who are getting Comm from emp table

Select * from emp where Comm is not null.

Like: This operator is used to search data based on character data along with like operator we are using 2 special characters



Syntax:

Select * from tablename where Columnname like 'character pattern';

Q) display the employees whose ename start with 'M'

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

o/p

ename
MARTIN
MILLER

Q) Display the employees whose ename second letter would be 'L'

Select * from emp where ename like '_L%';

o/p

ename
ALLEN
BLAKE
CLARK.

Concatination operator (||)

Java → +

C# → +

VB.net → -

This operator is used to Concatinate different datatype columns it is also used to Concatinate column value with strings

is Null, is not null:

These operators are used in where clause only, these operators are used to test whether a column having null values & not.

Note: Generally we are not allowed to use = null in where clause in place of this one we are using "is null"

syntax:

Select * from tablename where Columnname is null;

syntax:

select * from tablename where Columnname is not null;

Query:

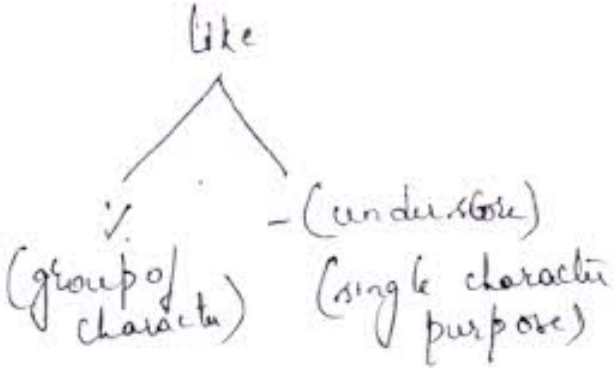
1) display the employees who are not getting Commition from emp table

Select * from emp where Comm is null

2) display the employees who are getting Comm from emp table

Select * from emp where Comm is not null.

Like: This operator is used to search data based on character data along with like operator we are using 2 special characters



Syntax

Select * from tablename where Columnname like 'character pattern';

Q) display the employees whose ename start with 'M'

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

o/p ename
MARTIN
MILLER.

Q) Display the employees whose ename second letter would be 'L'

Select * from emp where ename like '_L%';

o/p ename
ALLEN
BLAKE
CLARK.

Concatination operator (||)

Java → +
C# → +
VB.net → -

This operator is used to Concatinate different datatype columns it is also used to Concatinate Column value with strings.

is Null, is not null:

These operators are used in where clause only, these operators are used to list whether a column having null values & not.

Note: Generally we are not allowed to use =null in where clause in place of this one we are using "is null".

syntax:

Select * from tablename where Columnname is null;

syntax:

select * from tablename where Columnname is not null;

Query:

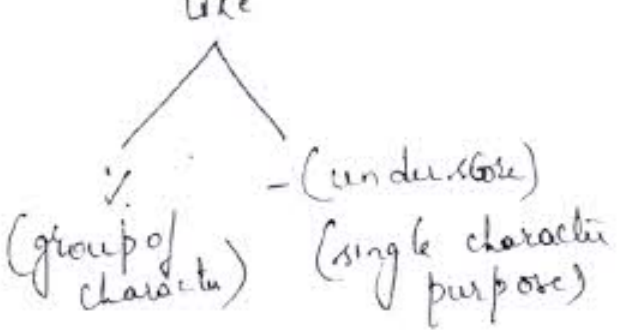
1) display the employees who are not getting Commition from emp table

Select * from emp where Comm is null

2) display the employees who are getting Comm from emp table

Select * from emp where Comm is not null.

Like: This operator is used to search data based on character data along with like operator we are using a special characters



Syntax

Select * from tablename where Columnname like 'character pattern';

Q) display the employees whose ename start with 'M'

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

o/p

ename
MARTIN
MILLER

Q) Display the employees whose ename second letter would be 'L'

Select * from emp where ename like '_L%';

o/p

ename
ALLEN
BLAKE
CLARK.

Concatination operator (||)

Java → +

C# → +

VB.net → &

This operator is used to Concatinate different datatype columns it is also used to Concatinate column value with strings.

Ex: Select ename || ' ' || sal from emp;

Functions

There are used to solve some particular task.
There are 2 types of functions supported by oracle

1) Predefined functions
2) Userdefined functions.

4/2/2012

Predefined functions:

1) Number functions
2) character functions
3) date functions.
4) group functions (or) aggrigate functions.

Number functions: These funct^n operate over number data

1) abs ( ) : this function returns +ve values.

Ex: Select abs(-50) from dual

-> 50

-ve value gets Converted to +ve value

dual → is a predefined virtual table

If instead of dual, emp or dept is used it returns the values as so that number of rows times.

dual: it is a predefined virtual table, which Contains only one row and one Column, generally this table is used to test predefined functions functionality

By default dual table Column data type is varchar2

Ex: Sql > Select * from dual;

Sql > desc dual;

sql> Select abs(-20) from dual

sql> Select ename, comm, sal, abs(comm - sal) from emp
where comm is not null

* mod(m,n) : it will give remainder after m is divided by n

Select mod(10, 5) from dual

→ '0'

round(m,n) : it rounds given floated value number m based on n

Select round(1.7) from dual

→ 2

Select round(1.23456, 3) from dual

1.234(56 → as the remaining is above 50% therefore we need to add 1 to the 3 decimal

1.235

Note: round always checks remaining number, if remaining numb above 50% 1 added to the rounded number

Ex: Select ename, sal, sal/22, round(sal/22) from emp;

Trunc(m,n) : it truncates given floated value number m, based on n

Ex: Select trunc(1.8) from dual

→ 1

Select trunc (1.23456, 3) from dual;

→ 1.234

* greatest (exp1, exp2 ....), least (exp1, exp2 ....)

greatest returns maximum value among given expressions where as least returns minimum value among the given exp

Select greatest (3, 6, 9) from dual

→ 9

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

Ceil(), floor()

Ceil returns nearest greatest integer where as floor returns nearest lowest integer

Ex: Select ciel (1.2) from dual

→ 2.

Select floor (1.8) from dual

→ 1

Character functions

these functions operate over character data

1.) upper() : It is used to convert column values or string into uppercase.

⇒ Select upper (ename) from emp

Select upper ('murali') from dual

MURALI

2.) lower();

Select lower (ename) from emp;

to modify the data in the table

update emp set ename = lower (ename);

3) init cap: It returns initial letter capital all remaining letters small.

Sql > Select init cap (ename) from emp;

Select init Cap ('ab cd ef') from dual.

Ab Cd Ef

4.) Length: It returns total length of the string including space

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

-> 5

* Substr: It returns portion of the string within given string based on 2nd, 3rd parameters

Syntax: Substr (Column (or) 'stringname', searching position, number of characters from position)

Select Substr ('ABCDEFG', 2, 3) from dual

-> BCD

Select Substr ('ABCDEFG', -2, 3) from dual

FG

Select Substr ('ABCDEFG', -5)

CDEFG

Q) Display the employees whose ename 2nd letter would be LA using substr.

~~Select * substr (ENAME, 2, 2) from emp;~~

Select * from emp where substr (ename, 2, 2) = 'LA'

BLAKE
CLARK

5/7/2012

INSTR() : this function returns position of the character or position of string or posit^n of delimiter (, ?, #, ...) within a given string.
This returns number data type

Select instr ('ABCDEFGDGHCDIJK', 'CD')
from dual

o/p : 3

syntax : instr (columnname of stringname, 'str', searching position, number of occurence from position)

Instr function always returns position of a string based on the 3rd, 4th parameters numbers but oracle server Counting character from left side first position onwards.

Select instr ('ABCDEFGDGHCDIJK', 'CD', -5, 2)
1 2 3 4 5 6 7 -5 -4 -3 -2 -1
occurance
from dual

o/p : 7

LPAD() : It fills the remaining spaces with specified character on left side. Here always 2nd parameter returns total length of the string.

Ex: Select LPAD ('ABCD', 10, '#') from dual

#####ABCD

RPAD :

Select RPAD ('ABCD', 10, '#') from dual

ABCD######

Select RPAD (ename, 40, '_') || SAL from emp.

Ename	Sal
Smith	2000
Kenex	3000

Date functions

1.) Sysdate
2.) add_months()
3.) last_day()
4.) next_day()
5.) months_between(date1, date2)

1.) Sysdate : It returns Current system date in oracle date format

Select Sysdate from dual

o/p : 05-JUL-12.

2.

2.) Add_months( ) : this function is used to add of subtrac
-t months based on 2nd parameter.

Select add_months (sysdate, -1) from dual

o/p. 05-june-12

3.) Last_day ():it returns last day of specified month

select last_day (sysdate) from dual

31-jul-12.

4) Next_day ( ) : it returns next occurance day
based on 2nd parameter.

select Next_day (sysdate, 'sun') from dual

o/p 08-jul-12.

5.) Months_between (date1, date2) : returns number of
months between two dates. & also this function returns
number data type

Select Months_between (sysdate, Hiredate)
from emp.

Opn

Date arithmatic :

1.) Date + number
2.) Date - number
3.) Date1 - Date2

Seldct Date + (sysdate + 1) from dual
Select (sysdate - 1) from dual
Select (sysdate - sysdate) from dual.

Date Conversion functions:

1) To_char()
2) To_date()

1) To_char(): this function is used to Convert date type into character type. i.e it Converts date type into date string.

Ex: Select To_char (Sysdate, 'DD/MM/YY')
from dual

05/07/12.

D DD MM YY — numbers

DAY DY MONTH MON YEAR — characters

Note: Basically To_char is a Case sensitive function

Ex: Select To_char (sysdate, 'DAY')
from dual

THURSDAY.

Select To_char (sysdate, 'day')
from dual

thursday.

Select To_char (sysdate, 'DY') from dual

o/p. THU

Select To_char (sysdate 'D') from dual

o/p 5

Sunday: 1
Monday: 2
Tuesday: 3
Wed: 4
Thurs: 5

DD → Day of the month.

DDD → Day of the year.

Select To_char (sysdate, 'DDSPTH') from dual

fifth.

Note always To_char first parameter must belong to date type

To_Date; it is used to Convert date string into date type (Oracle date format)

Ex: Select To_date ('10/FEB/04') from dual

o/p: 10-FEB-04

Select To_date ('15/JUNE/04') from dual

15-JUN-04.

Select To_date ('15/06/04') from dual

Error: not a valid month.

Select To_date ('15/06/04', 'DD/MM/YY') from dual

o/p 15/06/04. 15-JUN-04.

Select To_char (To_Date ('14-JUN-03'), 'DAY-/MONTH/YY')

from dual.

Saturday / June / 03.

6/7/2012

Note: whenever we are using To_date() first parameter must be follows format mask. of the oracle default date format i.e To_date() parameter return values match with default date format return value.
otherwise, use a second parameter as same as 1st parame -ter format then only oracle server returns date type

Q) Display the employees who are joining in the month dec from emp table using To_char()

Select To_char (emp Hiredate, 'MON') from emp;

Select * from emp where (To_char Hiredate, 'MON'): DEC

Select * from emp where To_char (Hiredate, 'MON'): DEC

Select '10-FEB-03' +4 from dual;

Error: invalid number.

Sol: Select To_date ('10-FEB-03')+4 from dual

Group functions: These functions operate over group of data and returns a single value, these are

1) MAX()
2) MIN()
3) AVG()
4) SUM()
5) COUNT(*)
6) COUNT (COLUMNNAME)

-> Select MAX(SAL) from emp;

Select MAX (Hiredate) from emp;

23-MAY-87.

Note: We are not allowed to use group functions in where clause

MIN( ) :

Select MIN (SAL) from emp;

1350

Select MIN (Hiredate) from emp;

17-Dec-80

AVG ( ) :

Select AVG (SAL) from emp;

8968.63714.

Note: By default all group functions ignores null values except Count(*)

Select AVG (comm) from emp;

550

Select AVG (NVL (comm, 0)) from emp;

0.157.142857.

-to Count the null values also we use NVL funct?

SUM ( ) : It returns total

Select Sum (sal) from emp;

Count(*) : It Counts number of rows in a table.

```
Select Count (*) from emp;
```

→ (14)

```
Select Count (*) from dual;
   (1)
Select Count (*) from dept;
   (4)
```

COUNT COLUMNNAME(): It Counts number of not null values in a Column

```
Select COUNT (COMM) from emp;
   (4)
```

Group by (): this clause is used to arrange similar data items into set of logical rows

Syntax: Select Columnname ....
from tablename
group by (Columnname);

Q) Display number of employees dept wise

```
Select deptno. Count (*) from emp
GROUP BY deptno;
```

O/P:

Dept No	Count(*)
10	3
20	5
30	6

Q) Display number of employees job wise.

```
Select job Count(*) from emp
Group By job ;
```

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

```
Select deptno., MAX(SAL)., MIN(SAL) from emp
Group By deptno. ;
```

dept No.	Max (sal)	MIN (SAL)
10	8000	6000
20	20000	13500
30	600	300.

Group By clause work without the use of group functions but of no use

```
Select deptno from emp Group By DeptNo ;
```

10
20
30

Rule: Other than group function columns specified after select those columns must use after group by otherwise oracle server returns an error if not a group by expression.

Ex: Select deptno, MAX(SAL), ENAME, from emp
Group By DEPTno, EName.

Select deptno, JOB from emp Group By
deptno, JOB.

7/7/2017

Display those depts having more than 4 employees in them

Sql> Select deptno, Count(*) from emp where Count(*) > 4
group by deptno;

where function is not used with group by funct"

Sol: select deptno, Count(*) from emp group by deptno
having Count(*) > 4;

Result:

DeptNo	Count(*)
20	5
30	6

Having: Generally we are not allowed to use where clause after group by

→ In place of this, ANSI/ISO sql standard introduced having clause

→ Generally where clause doesn't work with group functns where as having clause works with group functns.

→ Generally if we want to restrict rows in a table we are using where clause, where as if we want

to group after group by then we are using having clause.

Sql> Select deptno, Count(*) from emp where sal > 4000
group by deptno having Count(*) > 4;

Order by: This clause is used to arrange data in sorting order, along with order by clause we are using 2 keywords 1.) asc
2.) desc.

By default order by clause having ascending order

Syntax: Select * from tablename order by Columnname [asc/desc]

Sql> Select sal from emp order by sal desc;
sql> Select * from emp order by ename desc;
Sql> Select * from emp order by 6 desc
→ Columnno.

Syntax of Select

Select Col1, Col2, ... from tablename
where Condition
group by colname
having Col
order by Col;

Sql> Select deptno, Count(*) from emp
where sal > 1500
group by deptno
having Count(*) > 4
order by deptno desc;

Result:

deptno	Count(*)
30	6
20	5

→ Along with group by clause we can use Rollup & Cube

Rollup, Cube:

→ These clauses are used along with group by clause only.

→ These clauses are used to calculate subtotal, grandtotal automatically.

→ These clauses are introduced in Oracle 8i

Rollup: is used to calculate subtotal values based on single column where as

Cube: is used to calculate subtotal values based on all columns which are specified

Syntax for Rollup:

```
Select col1, col2 .....
  from tablename
  group by rollup (col1, col2 ... );
```

Syntax for Cube:

→
```
Select col1, col2 .....
  from tablename
  group by cube (col1, col2, ... );
```

Rollup Ex:

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

Cube ex:

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

Ex:

```
SQL> Select ename, sum(sal) from emp group by
     rollup (ename);
SQL> Select deptno, sum(sal) from emp group by deptno
     having sum(sal) > 2000;
SQL> Select deptno, sum(sal) from emp group by
     rollup (deptno);
```

9/7/2012

Joins: Joins are used to retrieve data from multiple tables

→ If we are joining 'n' tables we are using 'n-1' join conditions.

→ Oracle server supports following types of joins.

1.) Equi join (or) Inner join
2.) non equi join
3.) Self join
4.) Outer join.

Now a days all database systems are using "9i joins (or) ansi joins"

1.) Inner join
2.) Left outer join.
3.) right outer join

Result:

deptno	Count(*)
30	6
20	5

→ Along with group by clause we can use Rollup & Cube

Rollup, Cube:

→ These clauses are used along with group by clause only.

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

→ These clauses are introduced in Oracle 8i

Rollup: is used to calculate subtotal values based on single column where as

Cube: is used to calculate subtotal values based on all columns which are specified

Syntax for Rollup:

```
Select col1, col2 ....
  from tablename
  group by rollup (col1, col2 ....);
```

Syntax for Cube:

→
```
Select col1, col2 ....
  from tablename
  group by cube (col1, col2, ....);
```

Rollup Ex:

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

Cube ex:

```
SQL> Select deptno, job, sum(sal) Count(*) from emp
     group by cube (deptno, job);
```

Ex:

```
SQL> Select ename, sum(sal) from emp group by
     rollup (ename);
SQL> Select deptno, sum(sal) from emp group by deptno
     having sum(sal) > 2000;
SQL> Select deptno, sum(sal) from emp group by
     rollup (deptno);
```

9/7/2012

Joins: Joins are used to retrieve data from multiple tables

→ If we are joining 'n' tables we are using n-1 join conditions.

→ Oracle server supports following types of joins.

1. Equi join (or) Inner join
2. non equi join
3. Self join
4. Outer join.

Now a days all database systems are using "ansi joins" (or)

"ansi joins"

1. Inner join
2. Left outer join.
3. right outer join

4) full outer join
5) Natural join

→ We have default join that is "cross join" it depends on Cartesian product.

Note: We can also retrieve data by multiple tables without using join.

→ In this case data base server internally uses Cross join

→ Cross join internally implemented based on Cartesian product

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

Equi join: Based on equality condition we are retrieving data from multiple tables.

→ Here joining Conditional columns, must belongs to same datatypes.

→ whenever tables having Common Columns then only we are using equi join.

Syntax: Select col1, col2 from emp, dept where table1. Common column = table2. Common column

(join Condition)

Syl> Select ename, sal, deptno, dname, loc from emp, dept where emp. deptno = dept. deptno;

Error: Column ambiguously defined

Syl> Select ename, sal, dept. deptno, dname, loc from emp, dept where emp. deptno = dept. deptno;

Note: To avoid ambiguity in future generally we must specify every column name along with table name

using alias names:

```
SQL> 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;
```

(this join doesnot retrieve deptno if we are using d.deptno also)

Display the employees who are working in the location CHICAGO using equi join from emp, dept table

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

ENAME	LOC
ALLEN	CHICAGO
WARD	"
MARTIN	"
BLAKE	"

Note: If we want to use Conditional clause after join Condition we are using "AND" operation in "8i join"

→ where as in 9i join we are using it here AND & where clause.

Display dname, sum(sal) from emp, dept table using equi join.

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

Result:

DNAME	SUM(SAL)
Accounting	8750
Research	10,875
Sales	9400

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

Deptno	dname	Sum(sal)
10	Accounting	8750
20	Research	10,875
30	SALES	9400

Select dname, sum(sal) from emp e, dept d where e.deptno = d.deptno group by dname having sum(sal) > 9000

DName	Sum(sal)
Research	10,875
Sales	9400

Relation

Non Equijoins :-

→ Any type of relation other than '=' is known as non equi join.

→ Based on other than equality condition.

(<, >, <=, >=, between, in) we are retrieving data from tables

→ This join is also called between AND join.

Select ename, sal, losal, hisal from emp, salgrade.
where sal >= losal and sal <= hisal

→ Select ename, sal, losal, hisal from emp, salgrade where sal between losal and hisal

3.) Self join :-

Joining a table to itself is called self join

→ Here joining conditional columns must belongs to same data type

→ Generally self join is used to compare diff Column values but these columns must belongs to same datatype & also used to compare with in the column no. of values

→ Whenever we are working with self joins in database we should write alias names to tables.

→ In self join we must create alias name in from class

Syntax:

table name alias name1, table name alias name2 ...

Display Employee names and their mgr name using self join from emp table

Sql> Select e1.ename, e2.ename from emp e1, emp e2
where e1.mgr = e2.empno;

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

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

1) Display the employees who are joining in same month using self join

Sql> Select e1.ename, e1.hiredate, e2.hiredate, e2.ename
from emp e1, emp e2
where to_char(e1.hiredate, 'MON') =
to_char(e2.hiredate, 'MON');

Sql> Select e1.name, e1.hiredate, e2.ename, e2.hiredate
from emp e1, emp e2
where to_char(e1.hiredate, 'MON') = to_char(e2.hiredate,
'MON')
and e1.empno <> e2.empno;

3) Display the emps who are getting same salary in diff department.

```
sql> select e1.deptno, e1.name, e1.sal, e2.deptno,
     e2.ename, e2.sal
     from emp e1, emp e2
     where e1.sal = e2.sal
     and e1.deptno <> e2.deptno
     and e1.empno <> e2.empno;
```

4) Outer join : This join is used to retrieve all rows from one table, matching rows from another table

Generally when we are using equi join we are retrieving matching rows only, if we want to retrieve non matching rows also, we are using equi join along with join operator (+)

→ This operator is used only one side at a time in joining condition

→ This join is called 'outer join'

This is an extension to 'equi join'

Which tables data & content we want to display that side we should not use (+), this is reverse operation in (3) But accurate in 9i, that is which tables record we want to display.

Select ename, sal, d.deptno, dname, loc from
emp e, dept d
where e.deptno (+) = d.deptno
↓ → all rows
matching rows.

Both the sides we are not allowed to use join operator at a time.

(+) = (+) // not allowed.

11/7/2012

→ Equi join is used only when there are Common Columns
It discards the data i.e not having Common data.

→ 9i version is introduced in 2001

Before using outer join to retrieving data they used union operator.

Note: If we want to retrieve matching and non matching rows from both the tables we are using full outer join. But full outer join is 9i join

→ But before Oracle 9i we retrieved this type of data using union operator.

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

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

/

9i joins (or) ANSI joins.

Always 9i joins increase the performance

1.) inner join
2.) Left outer join
3.) right outer join
4.) full outer join
5.) natural join

1) inner join: This join also returns matching rows only when tables contains common column, then only this join is used. And also this join performance is very high compared to 8i equi join

for converting 8i joins to 9i. in place of where clause on clause is used in 9i
in place of common we use join keyword

```
Sql> Select ename, sal, d.deptno, dname, loc
     from emp e join dept d      - for join both emp & dept table
     on e.deptno : d.deptno
     /
```

Display the employees who are working in the location chicago using 9i inner join:
(in 9i after join we can also use where instead of and)

```
Sql> Select ename, loc
     from emp e join dept d
     on e.deptno: d.deptno
     where loc : 'CHICAGO'
     /
```

in place of on clause we can also use using clause to increase the performance

Note: we can also use using clause in place of on clause, this clause returns common columns only.

Z_1 table

A	B	C
x	y	z
p	q	r

Z_2 table

A	B
x	y
s	t

Select * from Z_1 left outer join Z_2
on $Z_1.a = Z_2.a$ and $Z_1.b = Z_2.b$

/

O/p:

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

Right outer join :- this join returns all rows from right side table and also returns null value in place of non-matching rows in the other table.

Select * from Z_1 right outer join Z_2 on $Z_1.a = Z_2.a$
and $Z_1.b = Z_2.b$;

full outer join : this join returns matching, non matching rows from both the tables and also returns null values in place of non matching rows.

Select * from Z_1 full outer join Z_2 on $Z_1.a = Z_2.a$
and $Z_1.b = Z_2.b$,

O/P:

A	B	C	A	B
x	y	z	x	y
p	q	r	-	-
-	-	-	s	t

Joining 3 tables (or) more than 2 tables.

Q in 8:

Syntax: Select Col 1, Col 2 .....
from table 1, table 2, table 3
where table 1 . Common Col = table 2 . Common Col
And table 2 . Common Col = table 3 . Common Col;

Q in 9:

Syntax: Select Col 1, Col 2 .....
from table 1 on table 2 .
from table 1 join table 2
on table 1 . Common Col = table 2 . Common Col
join table 3
on table 2 . Common Col = table 3 . Common Col;

12/7/2017

Constraints

→ Constraints are used to prevent (restrict) invalid data entry into our tables

→ Generally Constraints are created on table columns

→ Oracle server supports following types of Constraints

1) Not Null
2) unique
3) primary key
4) foreign key
5) check

Z_1 table

A	B	C
x	y	z
p	q	r

Z_2 table

A	B
x	y
s	t

Select * from Z_1 left outer join Z_2
on $Z_1.a = Z_2.a$ and $Z_1.b = Z_2.b$

/

O/P:

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

Right outer join :- this join returns all rows from right side table and also returns null values in place of non-matching rows in the other table.

Select * from Z_1 right outer join Z_2 on $Z_1.a = Z_2.a$
and $Z_1.b = Z_2.b$;

full outer join :- this join returns matching, non matching rows from both the tables and also returns null values in place of non matching rows.

Select * from Z_1 full outer join Z_2 on $Z_1.a = Z_2.a$
and $Z_1.b = Z_2.b$;

O/P:

A	B	C	A	B
x	y	z	x	y
p	q	r	-	-
-	-	-	s	t

Joining 3 tables (or) more than 2 tables.

In 8i

Syntax : Select Col 1, Col 2 .....
from table 1, table 2, table 3
where table 1 . CommonCol : table 2 . Common Col
And table 2 . Common Col : table 2 . Common Col;

In 9i

Syntax : Select Col 1, Col 2 .....
from table 1 on table 2 .
from table 1 join table 2
on table 1 . Common Col = table 2 . Common Col
join table 3
on table 2 . Common Col = table 3 . Common Col;

12/7/2011

Constraints

→ Constraints are used to restrict / prevent invalid data entry into our tables

→ Generally Constraints are created on table columns

→ Oracle server supports following types of Constraints

1) Not Null
2) unique
3) primary key
4) foreign key
5) check

→ All the above Constraints are created in two levels

1) Column level.
2) Table level

Column level :- In this method we are defining Constraints for individual Columns i.e whenever we are creating a column then only we specify the Constraint type

Syntax : Create table tablename (col1 datatype (size) Constraint type, col2 datatype (size2), ...);

Table level :- In this method we are defining Constraints on group of Columns i.e first we should define all Columns then at the last we must specify Constraint type along with group of Columns.

Syntax : Create Table Tablename (col1 datatype (size), col2 datatype (size2) .....
Constraint type (group of cols like col1, col2 .....));

→ Not Null :- It doesnot support table level It doesnot accept null values, but it accepts duplicate values.

Column level :- Sql> Create table t1 (sno number (10) not null, name varchar2 (10));

Sql> insert into t1 values (null, 'murali');

error:
→ In the above case it does not insert the values as it doesnot accept null values

Unique : It doesnot accept duplicate values but it will accept null values

And also oracle server internally creates a btree index on those columns.

Column Level : Create table w2 (sno number(10) unique, name varchar2(10));

Table level : Create table w3 (sno number(10), name varchar2(10), unique (sno, name));

```
Sql> Select * from w3

     Sno    name
      1     rego
      1     murali
```

Primary key : primary key eg uniquely identifies record in a table, and also there can be only one primary key in a table, it

→ It does not accept duplicate, null values.

→ oracle server internally creates btree indexes on primary key columns.

Column level :

```
Sql> Create table w4 (sno number(10) primary key,
                      name varchar2(10));
```

Table Level:

Sql > Create table ws (sno number(10), name nvarchar2(10),
primary key (sno, name));

Cons: This is also called as Composite primary key
i.e this is a Combination of Columns as a single primary key.

Foreign Key: If we want to establish Relationship b/w tables we are using Referential Integrity Constraint, foreign key.

→ One table foreign key must belong to another table primary key and also these two columns must belong to same data type.

→ Generally foreign key values based on primary key values only and also foreign key accepts duplicate, null values

foreign key is not used in column level applicable only for table level

To establish a relation b/w diff tables we uses need of to use References

Column level: In Column level (references)

Syntax: Create table table name (col1 datatype (size)
References master table name (Primary key col1),
col2 datatype (size) - -);

SQL> Create table Z4 (sno number(10) references W4)

SQL> Create table Z5 (x number(10) references W4(sno));

13/7/2012

Table level:-

Syntax:

Create table table name (col1 datatype size), col2 datatype size
..... foreign key (col1, col2)
references master table name (primary key columns));

SQL> Create table d5 (sno number(10), name varchar(10),
foreign key (sno, name) references W5);

We cannot insert values in child table other than P key in master table

Whenever we are establishing a relationship b/w tables oracle server support 2 Rules.

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

1.) Deletion in Master table:-

whenever we try to delete a master table record oracle server returns an error.

ora - 2292

→ To overcome this problem first we are deleting related records in child table then only we can delete master table record

Otherwise use on delete cascade clause.

On delete Cascade:

whenever we are using this clause in child table, if you are deleting a master table record, automatically related master, child table records are deleted.

Syntax:

Create table tablename (col1, datatype (size)
references mastertable (1° key column)
on delete cascade);

→ Oracle server also support another clause in child table

On delete set null:

→ When we are using this clause in child table, if we are deleting a master table record automatically related foreign key values set to null

Syntax:

Create table child tablename (col1 datatype (size)
references master table name (primary key column
on delete set null);

Insertion in child table:

If we are trying to insert other than 1° values to foreign key column oracle server returns an

an error

Ora - 2291

Because always foreign key values based on P key values only.

Check:

This Constraint is used to define logical Conditions according to our business rules.

Syntax:

Create table tablename (col1 datatype (size) check logical condition, col2 datatype (size) ....);

Ex:

sql > Create table b1 (name varchar2 (10) check (name = upper (name)));

sql > to insert into b1 values ('murali');

error: Check Constraint (SCOTT . SYS - C005993) violated.

Assign userdefined names to Constraints:

whenever we are creating a Constraints automatically each server generates an unique identification number in the format of SYS-CN This is called userdefined Constraint name.

Syntax:

Constraint userdefined name Constraint type

↓ Constraint name

↓ primary, foreign, check, unique

Otherwise use on delete cascade clause.

On delete Cascade:

whenever we are using this clause in child table, if you are deleting a master table record, automatically related master, child table record are deleted.

Syntax:

```
Create table tablename (col1, datatype (size)
    references mastertable (1° key column)
        On delete cascade);
```

→ Oracle server also support another clause in child table

On delete set null:

→ when we are using this clause in child table, if we are deleting a master table record automatically related foreign key values set to null

Syntax:

```
Create table childtablename (col1 datatype (size)
    References mastertable name (primary key column
        on delete set null);
```

Insertion in child table:

if we are trying to insert other than 1° values to foreign key column oracle server returns an

an error

ora - 02291

Because always foreign key value based on P key value only.

Check:

This Constraint is used to define logical Conditions according to our business rules.

Syntax:

Create table tablename (col1 datatype (size)
check logical condition, col2 datatype (size)
....);

Ex:

Sql > create table b1 (name varchar2 (10) check (name = upper (name));

sql > insert into b1 values ('murali');

error: Check Constraint.

(SCOTT.SYS-C005293) violated.

Assign user defined names to Constraints:

whenever we are creating a Constraints automatically each server generates an unique identification number in the format of SYS-CN This is called user defined Constraint name

Syntax:

Constraint user defined name Constraint type

↓ Constraint name

↓ primary, foreign, check, unique

Ex:

```
SQL> create table d8 (sno number(10) constraint
     p_eega primary key);
SQL> insert into d8 values(1);
SQL> insert into d8 values(1);
```

Error: unique constraint (SCOTT.P_EEGA) violated

Note: All constraints information stored in user_data dictionary

```
SQL> DESC USER_CONSTRAINTS;
     select constraint_name, constraint_type from
     user_constraints where Table_Name = 'EMP';
```

(after equal to it is mandatory to write in capital letters)

O/P:

Constraint Name	Constraint type
PK_EMP	P
FK_DEPTNO	R → Referential Integrity.

Note: If we want to view columnnames along with constraint names we are using user_cons_columns.

Ex: SQL> DESC user_cons_columns;

```
Sql> Select
  Constraint_name , Column_name from user_Cons_Columns
    where Table_Name = 'EMP';
    /
```

O/p :

Constraint_Name	Column_name
Fk_deptno	deptno
Pk_Emp	Empno

Note: If we want to view logical condition of the check Constraint, we use search_Condition property.
from user_Constraint data dictionary.

```
Sql> Select Search_Condition from user_Constraint
    where Table_name = 'B';
    /
```

O/p : name : upper(name)

All Columns information stored in user_tab_Columns data dictionary

```
Sql> DESC user_tab_Columns.
```

Q) write query to return number of Columns in a table

```
Select Count(*) from user_tab_Columns
    where table_name = 'EMP';
```

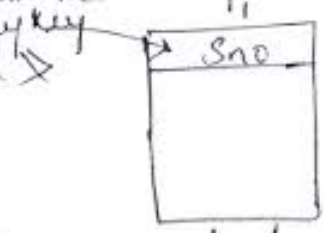
O/p : 8

Adding, Dropping Constraints on existing table:

→ using alter we can also add or drop constraints on existing table

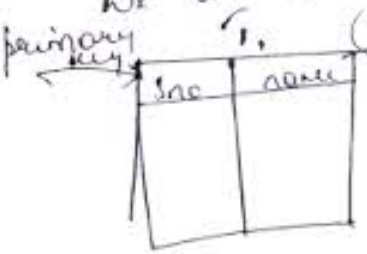
Syt → ~~Create table T1 (sno~~

Note: If we want to add constraints on existing table, existing column we are using table level constraint syntax method



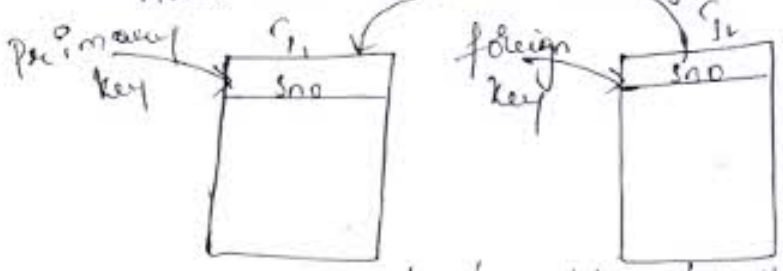
Alter Table T1 add primary key (sno);

→ If we want to add new column along with constraints we are using column level syntax method.



Alter table T1 add name varchar(10) unique;

→ Alter table T2 add foreign key (sno) references T1;



→ If we want to add not null constraint to the existing table existing column we alter with modify

Syntax: alter table T3 modify Sno not null;

alter table tablename modify Columnname not null;

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

Sql > Alter table t3 modify sno. not null; /

to add new Column

Alter table t3 add name varchar2(10) not null; /

16/7/2012

Dropping Constraints:

Method 1 :- Alter table tablename drop Constraint Constraintname;

(8)

Method 2 : Alter table tablename drop primary key;

(8)

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

Sql > Create table c1 (sno number(10) primary key);

sql > alter table c1 drop primary key;

Note: If we want to drop primary key along with foreign keys we use cascade clause.

Syntax : alter table tablename drop primary key

Cascade;

alter table t1 drop primary key Cascade;

To drop Not Null Constraint

Alter table t3 drop constraint

Sys_c005319;

Sql > Create Table C2 (sno number(10) Primary key);

Sql > Create table D2 (sno number(10));

Sql > Alter table D2 add foreign key (sno) References C2 on
ON delete Cascade

Sub Queries :-

→ Query within another query is called nested query (or) sub query, generally child query is also called as sub query.

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

There are two types of subqueries supported by all database systems

→ Non Correlated subqueries.

→ Correlated subqueries.

In non Correlated subqueries child query is executed first then only parent query is executed whereas in Correlated subqueries, parent query is executed first then the child query is executed

Non Correlated subqueries :-

1. Single row subqueries.
2. Multiple row
3. Multiple Column subqueries
4. Inline views(2) subqueries in from clause

Q) Display the employees who are getting more than the avg salary from emp table

~~Select * Employ~~

Select * from emp where sal > avg(sal); X

Select * from emp where
sal > (Select Avg(SAL) from emp); ✓

This is a single row sub query because here child query returns single value. In single row sub query we are using equal to, <, <=, >, >= operators

Display the emp who are working in sales department using emp, dept tables

Select * from emp where
~~department~~ deptno = (select * deptno = (Select deptno from dept where dname = 'sales');

90% of queries contain the same Columnname i.e after where clause inside the child query after 5% where clause columnname after is used as groupname after select in child query

ex: sal > (select Avg(sal) from emp)

3% sal = (select sal+100 from emp) expression

Select * from emp where deptno = (Select deptno from dept where dname = 'Sales');

Sql > Create Table C2 (sno number(10) Primary key);

Sql > Create table D2 (sno number(10));

Sql > Alter table D2 add foreign key (sno) References C2 on
ON delete Cascade

Sub Queries :-

→ Query within another query is called nested query (or) sub query, generally child query is also called as sub query.

→ Sub queries are used to retrive data from single or multiple tables based on more than one step process

There are two types of subqueries supported by all database systems.

→ Non Correlated subqueries.

→ Correlated subqueries.

In non Correlated subquery child query is executed first then only parent query is executed whereas in Correlated subquery, parent query is executed first then the child query is executed

Non Correlated Subqueries :-

① Single row subqueries.

② Multiple row

③ Multiple Column subqueries

④ Inline views(2) subqueries in from clause

Q) Display the employees who are getting more than the avg salary from emp table.

~~Select *~~ Employ

Select * from emp where sal > avg(sal); X

Select * from emp where
sal > (Select Avg(SAL) from emp); ✓

This is a single row sub query because here child query returns single value. In single row sub query we are using equal to, <, <=, >, >= operators

Display the emp who are working in sales department using emp, dept tables

Select * from emp where
~~department~~ deptno = (~~select * dname = sales~~);
deptno = (Select deptno
from dept where dname = 'sales');

90% of queries contain the same Columnname i.e After where clause Columnname after is used as groupname after select in where clause

5% child query ex: sal > (select Avg(sal) from emp)

3% sal = (select sal+100 (expression) from emp)

Select * from emp where deptno = (Select deptno from dept where dname = 'Sales');

Select Ename, Dname from emp E, dept D
where E.Deptno = D.deptno AND D.deptno
: (Select Deptno from dept where Dname = 'SALES');

Display the most employee details from emp ~~details~~ table

Select * from emp where ~~higher date~~ hiredate : (~~select highest date~~ = Max
(select Max(hiredate) from emp);

17/7/2012

Q) Display the employees who are working under Zones using empno, Mgr

SQL > Select * from emp where mgr : (select empno from emp where ename = 'JONES')

Group by clause and in parent query:

Q) Display the highest AVG salary job from emp table.

SQL > Select job, AVG(SAL) from emp Group by Job having AVG(SAL) = (Select MAX (AVG(SAL)) from emp)

Error: Nested group funct^n without group by

SQL> Select job, AVG(SAL) from emp group by Job having AVG(SAL) : (Select MAX (AVG(SAL)) from emp group by job)

O/P : JOB AVG(SA)
PRESIDENT 5600.

Note: Whenever we are using nested group functions we must use group by clause.

```
SQL> Select deptno, MAX(SAL) from emp Group by
     Deptno Having MAX(SAL) > (select MAX(SAL) from
                emp where deptno = 30)
```

Q) Display the employee details who are getting MAX (sal) in each dept.

```
SQL> select * from emp where SAL = (select MAX(sal)
                from emp group by deptno)
```

Error: Single row sub query returns more than one row.

```
SQL> select * from emp where SAL IN (select MAX(sal)
                from emp group by deptno)
```

left side one value, right side more values means we use IN.

```
SQL> Select deptno, SAL, ename from emp where SAL
     IN (select MAX(SAL) from emp group by deptno)
```

deptno	sal	eName
10	5600	KING
20	3479.94	JONES
30	3342.13	BLAKE

2) This is a multiple row subquery because here child query returns more than one values in multiple row subquery's we are using IN, ALL, ANY operators.

Note: We can also used IN operator in single row sub queries..

Q.) Display the emp who are working in the either sales or RESEARCH dept no

Sql> Select * from emp where deptNo IN (Select deptno from dept where DNAME = 'SALES' OR DNAME = RESEARCH);

18/7/2011

Top N Analysis:

~~In line Views~~

1.) INLINE Views

2.) ROWNUM

In child query order by clause shouldn't be used inorder to use order by

INLINE Views: Oracle 7.2 introduced inline views, generally we are not allowed to use order by clause in child query to overcome this problem Oracle introduced sub queries in from clause, these type of queries are also called as inline views.

i.e In this query we are using sub query in place of table name

syntax: Select * from (sub query);

ROWNUM: It is a pseudo column, it behaves like a table column this column is used in any type of tables

ROWNUM assigns numbers to each row in a table at the time of selection

Sql> Select rownum, enam from emp;

RowNum has temporary values i.e based on the select stat it assigns values

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

Q) display first row from emp table using Rownum

Sql> Select * from emp where Rownum=1;

Note: Rownum doesnot work with more than one positive integer i.e this pseudo column always work with <, <= operators

Q) display first five rows from emp table using Rownum

Sql> Select * from emp where rownum <=5;

Q) display first five highest salary employees from emp table using rownum.

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

Q) display fifth highest sal employee using rownum.

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

minus

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

Note: We can also used IN operator in single row sub query..

Q.) Display the emp who are working in the either sales or RESEARCH dept no.

Sql> Select * from emp where deptNo IN (Select deptno from dept where DName = 'SALES' OR DNAME = RESEARCH');

18/7/2011

Top N Analysis:

~~Inline Views~~

1.) INLINE Views
2.) ROWNUM

In child query order by clause shouldn't be used inorder to use order by

INLINE Views: Oracle 7.2 introduced inline views, generally we are not allowed to use order by clause in child query to overcome this problem Oracle introduced sub queries in from Clause, these type of queries are also called as inline views.

i.e In this query we are using sub query in place of table name

Syntax: Select * from (sub query);

ROWNUM :- It is a pseudo column, it behaves like a table column this column is used in any type of tables

ROWNUM assigns numbers to each row in a table at the time of selection

Sql> Select rownum, enam from emp;

RowNum has temporary values i.e based on the select stat it assigns values.

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

Q) display first row from emp table using Rownum

Sql> Select * from emp where RowNum = 1;

Note: Rownum doesnot work with more than one positive integer i.e this pseudo column always work with <, <= operators

Q) display first five rows from emp table using Rownum

Sql> Select * from emp where rownum <= 5;

Q) display first five highest salary employees from emp table using rownum.

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

Q) display fifth highest sal employee using rownum.

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

minus

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

* Select * from emp
where rownum between 1 and 5 ;

Display rows blw 3 & 8 from emp table using rownum.

Select * from emp
where rownum between 3 & 8

O/p : no rows selected

Sql > Select * from emp
where rownum <= 8
minus
select * from emp
where rownum <= 3 ;

/

Sql > Select * from emp where rownum >= 1

/ o/p rows are selected.

Sql > Select * from emp where rownum > 1

no rows selected

Q) display last two rows from emp table using rownum.

Sql > Select * from emp
minus ~~rownum <= 12~~
Select * from emp where rownum <= 13.

Sql > Select * from emp
minus
select * from emp where
rownum <= (Select Count(*) - 2 from emp)

Note: Whenever we are creating an alias name for rownum in line view that alias name works with all sql operators

display 5th row from emp table using rownum alias name

Sql> Select * from (select rownum r, ename, sal from emp) where r = 5

Sql> Select * from (select rownum r, ename, sal from emp) where r between 3 and 8.

to display all the columns

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

display even number of rows from emp table using rownum alias name

Select * from (select rownum r, ename, sal from emp) where ~~r=0~~ mod(r, 2) = 0

19th Row

Display first row, last row from emp table using rownum alias name

Select * from (Select rownum r, ename, sal from emp) where r=1 or r= (select Count(*) from emp)

Display fifth highest sal employee from emp table using rownum alias name.

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

Analytical functions used in Inline views

→ Analytical functions are introduced in oracle 8i. there are
i) row_number() ii) rank(); iii) dense_rank()
These 3 analytical functions assigns ranks either group wise
or rows wise

Syntax: Analytical functionname() over
(Partition by Columnname
order by Columnname [asc/desc])

When the values are same if u row_number() should not be
used instead we can use rank or dense_rank()
which produce same rank instead of different rank.
but rank skips the next Consecutive number
while dense_rank doesnot skip the numbers.

→ Row_number() analytical function assigns different rank numbers
when values are same where as rank, dense_rank analytical
functions assigns same ranks when values are same and
also rank skips next Consecutive rank numbers whereas
dense_rank() doesnot skip next Consecutive rank number

row number()

20	SCOTT	3400	2
10	FORD	3400	3
20	ADAMS	1900	4

rank ( )

20	SCOTT	3400	2
20	FORD	3400	2
20	ADAMS	1900	4

dense_rank ( )

20	SCOTT	3400	2
20	FORD	3400	2
20	ADAMS	1900	3.

Ex 1 Select * from (Select deptno, ename, sal, row_number() over (partition by deptno order by sal desc) r from emp) where r <= 10

***Q) Display 2nd highest salary employees dept wise using analytical functions from emp table

Select * from (Select deptno, ename, sal, dense_rank() over (partition by deptno order by sal desc) r from emp) where r = 2

Q) Display fifth highest salary employee from emp table using analytical function

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

o/p: BLAKE 80000 5

partition by is optional
for displaying the rank group wise we use partition by Clause.

Multiple Column subquery :-

→ We can also compare multiple columns of the child query with the multiple columns of the parent query.
In this case we must specify parent query where conditional columns within paranthesis

Syntax: Select * from tablename where
(Col1, Col2 .....) in (select col1, col2 ·
from tablename where Condition)

Display the employees whose job, Mgr match with job, Mgr of the employee scott

Select * from emp where (job, mgr) in (select job, mgr
from emp where ename = 'scott')

Display ename, dname, sal of the employee whose salary, comm ition match with sal, comm of the employee located in Dellas.

Select ename, dname from, sal from empe join dept d
where ~~e.deptno = d.deptno~~ on e.deptno = d.deptno.

```
Select ENAME, DNAME, SAL from emp e, dept d
where emp e.deptno = d.deptno
AND (SAL, COMM) IN (select SAL, comm from emp e,
dept d where
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 D.LOC = 'DALLAS')
```

/

20/1/2012

Q) Display the employees who are getting more than the highest paid employee in 20th dept.

```
Select * from emp where Sal > (Select max (sal) from
    emp where deptno = 20);
```

/

Q) Display the employees who are getting more than the lowest paid employee in 10th dept.

```
Select * from emp where sal > (Select min (sal) from
emp where deptno = 10);
```

→ whenever we use max, min functions in child queries and also tables having large amount of data & also comparing number of values those queries performance is very low to overcome this problem all database systems introduced

sub query special operators they are (all, any)

Select * from emp where sal > all
(Select * from emp where deptno = 20)

Select * from emp where sal > any (select sal from emp
where deptno = 10)

all, any operators used in multiple row sub queries.

IN: returns same values in child query

ALL: if satisfies all values in the child query.

ANY: if satisfies any value in child query.

Display the employees who are getting more than all salaries of clerk using sub query special operator.

Select * from emp where sal > all (select sal from
emp where job = 'CLERK');

Correlated Sub queries

→ Generally in non correlated sub query child query is executed first whereas in correlated sub query parent queries are executed first.

In correlated sub query we must create an alias name in parent query table and pass that alias name into child query where clause

Generally these queries are used in denormalization process i.e in denormalization we are combining normalized tables

into single table i.e in this case we are using Correlated update, Correlated delete.
If we want to modify one table column values based on another table we use Correlated update.

Ex:

Sql> alter table emp add dname varchar(10);

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

Select * from emp;

In Correlated query child query is executed for each row of parent query.
In non Correlated subquery child query is executed only once.

* Generally in non Correlated subqueries, child query is executed only once for parent table whereas in Correlated subqueries child query is executed for each row of parent table i.e whenever parent table contains large amount of data generally we donot use Correlated subqueries.

Process:

Step 1: get a candidate row (first row) from parent table

Step 2: Then control goes transferd into child query where condition

Step 3 : Based on evaluation value it compares with parent query.

Q.) Display the employees who are getting more than the avg sal of their jobs. using Correlated subqueries

Sql > Select * from emp e where sal > (Select avg(sal) from emp where job : e.job);

→ In Correlated subqueries Oracle server retrieves different values in each phase.

21/7/2012

n-1 method.

Display first highest salary employee from emp table using correlated sub query.

Sql > Select * from emp e_1 where (1-1) : (Select Count (*) from emp e_2 where $e_2.sal > e_1.sal$) /

O/p : king 5500.

Q.) Display fourth highest salary employee from following table using Correlated sub query.

Select *(from emp e_1;

SAL
3000
2000
4000
5000
7000
1500

Sql> Select * from emp e_1 where (4-1) : (select Count (*) from emp e_2 where $e_2.sal > e_1.sal$); /

step 1 : get a candidate row (first row) (3000)

step 2 : Control goes to child query

select Count (*) from emp e_2 where $e_2.sal > e_1.sal \Rightarrow e_1.sal > 3000$

2

step 3 : select * from emp1 e1 where (4-1) ⇒ 3

Result 2 > 3 (false) Hence it won't print 3000 in the o/p

Phase 2 :

step 1 : get a candidate row (first row)

2000.

step 2 : Control goes to child query

select Count (*) from emp1 e2 where e2.sal > e1.sal

e1.sal > 2000 ⇒ 3

step 3 : select * from emp1 e1 where (4-1) = 3.

3 = 3. true hence it prints 2000 in the o/p.

o/p : 2000
2000

nth method

→ for duplicate values in nth technique we need to use distinct keyword.

Select (*) Select * from emp1 e1 where 5

4 = (Select Count (distinct (sal)) from emp1 e2

where e2.sal >= e1.sal)

/

Result : 2000
2000

If distinct is not used

the o/p is no rows selected.

Exists operator:

exists operator used in Correlated sub query, this operator always returns boolean value either true or false.

→ This operator performance is very high compared to IN operator

→ This operator is used in where clause.

→ In this case we are not allowed to use column name.

→ Always exist operator start searching mechanism when child query returns true.

Syntax: Select * from tablename aliasname
where exists (Select * from tablename
where condition using aliasname)

Q.) Display those departments having employees in emp table using Correlated sub query from emp, dept tables

sql> Select * from dept d where exists
(select * from emp e where deptno = d.deptno)

it checks whether the value is present in the table or not.

Q.) Display those departments doesnot have any employees in them using Correlated subquery

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

Q.) Display those departments doesnot have any employees in them using non Correlated sub query.

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

→ Not In operator does not work with null values, so inorder to resolve this problem we use not exist operator in correlated subquery

Ex: Sql > insert into tablename emp (empno deptno)
values (', null);
Sql > Select * from dept where deptno not in
(select deptno from emp);
no rows selected

Views:

→ View is an a database object, which is used to provide authority level of security
Generally views are created by database administrator & then these views are given to number of users.
Views doesnot store data and also views are created from base tables
Based on the base tables views are categorised into two
Views
1) Simple view
2) Complex view (or) join view.

→ Simple view is a view which is created from only one base table whereas Complex view is view which is created from number of base tables.
Simple view : it is a view which is created from only one base table

23/7/2012

Syntax: Create or replace view viewname as
select statement;

Sql> Create or replace view v1
as
select * from emp where deptno = 10;

→ DML operation on View:

→ We can also perform DML operations through simple view to base table based on following restrictions

→ we must include base table NOT NULL Column into the view, then only we can perform insertion operation through view to base table.

→ If a simple view contains group functions, groupby clause, rownum, set operators, joins then we cannot perform DML operations through view to base table.

Sql > Create or replace view v1
as
select * from emp where deptno = 10;

Sql > Select * from v1;

Sql > insert into v1(empno, deptno) values (', 30);

Sql > Select * from emp;

Case 2:

Sql > Create or replace view v2
as
select ename, deptno from emp where deptno=10;

Sql> Select * from v1;

Sql > Insert into v1 (ename, deptno) values ('abc', 10);

Error : Cannot insert null value into empno.

→ View does not store data instead it stores the view definition ie select stat in user_data dictionary.

→ whenever we are creating a view, automatically view definitions are stored in database

In oracle automatically, view definitions are stored in user_views data dictionary.

Sql> Select TEXT from user_views where VIEW_NAME = 'V1';

With check option :-

→ If we want to create constraints on views, we use with check option clause.

Syntax: Create or replace view viewname
as
(select * from tablename where condition
with check option;

Sql > Create or replace view v1
as
select * from emp where deptno = 10
with check option;

Sql > Insert into v1 (empno, deptno) values (2, 30);

error : with check option where clause violation.

→ Read only : When read only option is used in a view, we cannot perform operation through view to base table, then type of view are called read only views

~~Select * from emp where~~

Sql > create or replace view v1

as

select * from emp where deptno = 10

with read only;

Force or forced views : We can also create views without base tables. these type of views are also called force or forced views.

syntax : create or replace force view viewname

as

select * from ~~no. any tablename~~ name;

force

Ex: Create or replace view v1

as

select * from name;

warning: view created with compilation errors

Sql > Create table name (sno number(10));

Sql > alter view v1 compile;

Sql> desc v1;

24/7/2012

Complex views (or) join views:

Complex view is a view which is created from number of base tables.

Create or replace view v_1

as

select ename, sal, dname, loc

from emp, dept

where emp.deptno = dept.deptno;

Whenever we try to perform dml operation through complex view to base table, some table columns are affected & some other table columns are not affected. Generally non key-preserved tables are not affected. If we want to view affected, non effected views we use user_updatable - columns data dictionary.

Sql > Select * from v_1;

Sql > update v_1 set ename = 'abc' where eno

ename: 'SMITH';

Sql > update v_1 set dname: 'xyz' where

dname = 'SALES';

Error: cannot modify a column which maps to a non-key preserved table

Sql > desc user_updatable - columns;

Sql> Select Column_name, updatable from
user_updatable_Columns
where table_name = 'V1';

O/P:

COLUMN_NAME	UPDATABLE
ENAME	YES
SAL	YES
DNAME	NO
LOC	NO

To overcome this problem oracle 8.0 introduced instead of triggers in plsql

By default instead of triggers are row level triggers and also these triggers are created on views

Triggers (plsql)

Triggers are also same as stored procedures & also it will automatically invoke when dml operations performed on table

There are two types of triggers supported by oracle

1.) Statement level triggers
2.) Row level triggers

In statement level trigger trigger body is executed only once for dml statements whereas in row level triggers, trigger body is executed for each row for DML statements.

Trigger Syntax: create or replace Trigger Triggername
Specification for Before | after insert | update | delete on
(row level trigger) tablename [for each row] where restriction
(for statement level trigger)

Trigger body { Begin
=
=
end;

→ difference blw statement level, row level.

Ex+

Sql > Create table tst (tod date);

Statement level trigger:

Sql > Create or replace trigger tv7.
after update on emp
begin
insert into test values (sysdate);
end;
/.

Sql > update emp set sal: sal + 100 where.
deptno : 10;
3 rows updated.

sql > select * from test;

tod
24-jul-12.

Sql> drop trigger tv7;

Row level trigger.

Sql > Create or replace trigger tv8
after update on emp
for each row

```
begin
insert into test values(sysdate);
end;
/
```

```
SQL> update emp set sal: sal+ 100 where
     deptno :10;
3 rows updated
SQL > select * from test;

Tod
24-jul-12 -> previous query result
24-jul-12
24-jul-12
24-jul-12
```

25/7/2012

Row level triggers: In row level triggers, trigger body is executed for each row for dml statements. That's why we are using for each row clause in trigger specification and also data is internally stored in two roll back segments qualifiers. They are 1) old 2) New
These qualifiers are used in either in trigger specification & in trigger body, but when we are using these qualifiers in trigger body we must use ':' colan in front of the qualifier name

Syntax: :old.Columnname
        :new.Columnname

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

Write pl/sql row level trigger on emp table whenever user delete data in emp table those records are automatically stored in another table

```
Sql > Create table backup as select * from emp where 1=2;
Sql > Create or replace trigger tk1
      after delete on emp          → from the above table for delete is possible in old qualifier.
      for each row
      begin
      insert into backup values (:old.empno, :old.ename,
                  :old.job, :old.mgr, :old.hiredate, :old.sal,
                  :old.comm, old.deptno);
      end;
Sql > delete from emp where sal > 2000;
Sql > select * from backup;
```

(Or this is a row level query we used for each row clause.)

Instead of triggers:

→ These triggers are created on views. By default these triggers are row level triggers

Syntax: Create or replace Trigger Triggername
Instead of insert / update / delete on
viewname
For each row

(If is mandatory to use row level because for each row is used)

```
Begin
==
End;

Sql > Create or Replace Trigger TK2
      instead of update on V1
      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;
        /
Trigger created
Sql > update V1 set Dname : 'xyz' where dname : 'SALES';
rows updated
```

Materialized views:

To store data in views we have materialized views.
There are introduced in oracle 8i, generally views doesnot store data, but materialized views store data.
Generally materialized views are used to improve performance of the joint or aggregatable queries thats why materialized views are maintained by database Admins, i.e materialized views store result of the query.
Then store replication of the remote database info local nodes (servers, clients)

→ Store data same like tables but when we are refreshing materialized view it synchronizes data based on base tables

Syntax: Create materialized view viewname

→ as
select statement;

Before creating materialized view database administrator must give create any materialized view privilege.

Syntax: grant Create any materialized view
to username;

Sql> Conn sys as sysdba;
Enter password: sys

Sql> grant Create any materialized view to Scott;
Grant succeeded.

Sql> Conn Scott/tiger;
Connected.

26/7/2012
Sql> Select * from base;

sno	Name
1	a
2	b
3	c
4	d

Sql> Create or replace view v1
as
select * from base;

```
Sql> Create materialized view mv1
     as
     select * from bank;
materialized view created
```

Here materialized view is also same as view i.e when we are creating materialized view automatically materialized views definitions are stored in database.

```
Sql> desc user_views;
Sql> select query from user_views where
     MVIEW_NAME : 'MV1';
```

```
Sql> select rowid, sno, name from bank;
Sql> select rowid, sno, name from v1;
```

Here view rowid is same as base table rowid, thats why view does not store data ie through the view we are viewing base table data

```
Sql> select rowid, sno, name from mv1;
```

Here materialized view rowids are different from base table rowids, thats why materialized view store same data

```
Sql> update bank set name = upper(name);
Sql> select * from bank;    Sql> select * from v1;
```

Sno	Name
1	A
2	B
3	C
4	D

By refreshing materialized view frequently will recreate the rowid

There are two types of materialized views supported by oracle

1.) Complete refresh.
2.) Fast refresh.

Complete refresh: By default materialized view uses complete refresh method, in this method rowids are recreated when we are refreshing materialized views

That's why these type of materialized views degrades the performance of the application.

Syntax: Create materialized view viewname
refresh Complete
as
Select statement;

Sql> Select rowid, sno, name from mni;
Sql> exec dbms_mview.refresh('mni');
Sql> Select rowid, sno, name from mni

In this case rowids are different

To overcome this problem oracle introduced fast refresh method

Fast refresh :- In this method rowids are not recreated when we are refreshing materialized views number of times

Syntax: Create materialized view viewname
refresh fast
as
Select statement;

Before creating fast refresh methods, we must create materialized view logs on base tables

Syntax: Create materialized view log on b.
base table name;

Sql > Create materialized view log on
base;
/

Sql > Create materialized view mv1
refresh fast
as
select * from base;

Sql > update base set name = 'xy' where
sno : 1 ;

Sql > Select rowid, sno, name from mv1;

Sql > exec dbms_mview.refresh('mv1');

Sql > select rowid, sno, name from mv1;

(here rowids are not recreated & also data is synchronized)

On demand, on Commit

Generally we are refreshing materialized views either manually & automatically, in manual method we are using dbms_mview package

This method is also called as ondemand method

By default method is on demand

automatically : we can also refresh materialized views automatically without using dbms_mview package
This method is called on commit method

Syntax : Create materialized view viewname
refresh complete/refresh fast on demand/on commit
as
select statement.

```
Sql > Create materialized view mv3
      refresh fast on commit
      as
      select * from loan ;
Sql > update loan set name = 'ZZ' where
      sno = 2;
Sql > select * from loan;
```

SNO	NAME
1	XY
2	ZZ
3	C
4	d

```
Sql > select * from mv3;
```

SNO	NAME
1	XY
2	b
3	C
4	d

Even though on commit clause is used the data is not effected here we should explicitly use commit instead of dbms_mview refresh

```
Sql > Commit
```

SNO	NAME
1	XY
2	ZZ
3	C
4	d

27/12/2012

Normalization

→ It is a scientific process.
→ It is developed in 1970.
→ Process of decomposing single
→ Normalization is a scientific process which is used to decompose a table into no of tables
→ This process automatically reduces redundancy & also automatically eliminates insertion, u/dat^n, delet^n problems
→ In the design phase of SDLC database designers design logical model of the database.
→ In this logical model only they are using normalization process.
→ In 1970 E.F. CODD written a paper article
→ Relational model of data for large shared data banks.
→ E.F CODD is father of SQL
→ In this paper only E.F CODD defined first three Normal forms.

1) First
2) Second
3) Third
4) bcnf
5) Fourth
6) fifth

First normal form

→ A table is a first normal form every cell must contain single value and also identifying a record uniquely using 1° key.

Item table.

Item name	Color	Price	Tax
Marker	red, blue	30	0.3
Pen	blue black	20	0.2

1NF →

Candidate key

Item name	Color	Price	Tax
Marker	red	30	0.3
Marker	blue	30	0.3
Pen	blue	20	0.2
pen	black	20	0.2

Procedure :

→ Identifying repeated groups & put it into seperate table in order automatic form -

Electronic shop

order no
order date
customer name
address
Phno
Item name
Amount
Submit

order no	order date	last name	address	phno
1	27	—	—	—

order no	Item name	Amount
1	Item	

Second Normal form :

→ Candidate key Columns are key attributes.

→ A table is 1st NF & also all nonkey attributes fully functionally dependent on all key attributes of total Candidate key.

→ Generally first Normal form deals with automatically where as 2nd NF deals with relationship b/w key, Nonkey attributes

→ If nonkey attributes partially dependant on key attributes then that table not in second NF.

Procedure :

→ Identify partial nonkey attributes which is dependant on partial key attribute, put it into separate table

→ This table is called 2NF. This table having unique values.

candidate key → Item name, Color
key attributes → Price, Tax

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

ItemName	Colour
Marker	Red
Marker	blue
Pen	blue
Pen	black

Master table

Item name	Price	Tax
Marker	30	0.3
Pen	20	0.2

2NF

Then price & Tax will depend only on itemname, there is no need of depending on colour.

28/7/2017

Sno	stud name	Activity 1	Cost1	Activity 2	Cost 2
101	abc	Cricket	$10	swimming	$20
102	xyz	swimming	$20	Golf	$50
103	bbb	Cricket	$10	Golf	$30
104	aaa	swimming	$20		

Stno	Sname
101	abc
102	xyz
103	bbb
104	aaa

Stno	activity 1	Cost 1	activity 2	Cost 2
101	Cricket	$10	swimming	$20
102	swimming	$20	Golf	$50
103	Cricket	$10	Golf	$30
104	swimming	$20		

1NF →

Candidate key

Sno	Activity	Cost
101	Cricket	$10
101	swimming	$20
102	swimming	$20
102	Golf	$30
103	Cricket	$10
103	Golf	$30

PK

Sno	Sname
101	abc
102	xyz
103	bbb
104	aaa

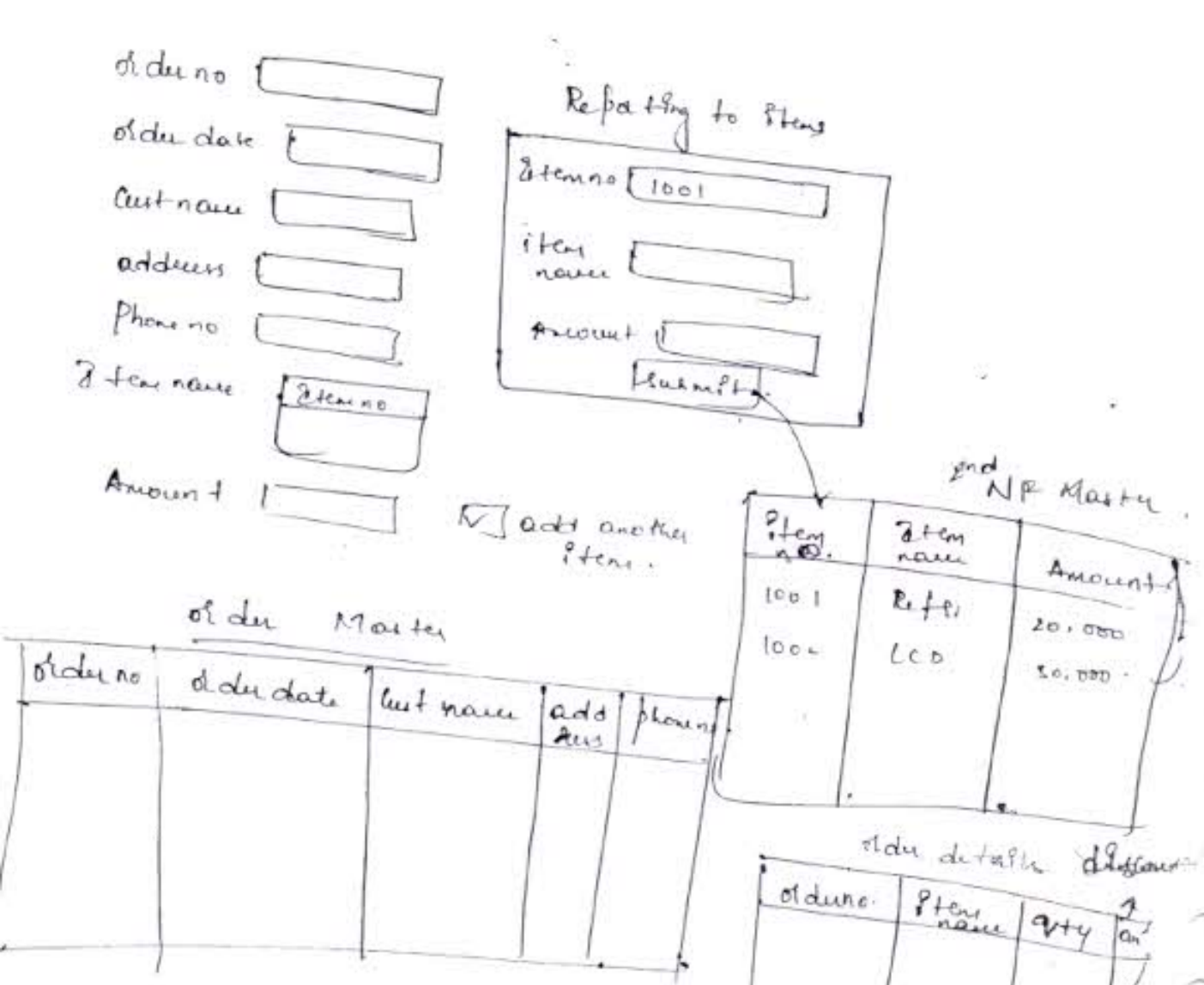
PK

Activity	Cost
Cricket	$10
swimming	$20
Golf	$30

PK FK

Stno	activity
101	Cricket
101	swimming
102	swimming
102	Golf
103	Cricket
103	Golf
104	swimming

~~Key attribute~~



ECode	ProjCode	Dept	Dephead	Hours
E101	P_1	system	D_1	5
E102	P_1	sales	D_2	2
E101	P_2	system	D_1	3
E101	P_3	system	D_1	4
E102	P_3	sales	D_2	8

ECode → Hours (x)

Hours not functionally dependant on Code.

E101 → 5, 3, 4.

as E Code contains duplicate values.

Proj code ⟶ (Hours (x)

P1 ⟶ 5
   ⟶ 2.

(3) ECode + Proj code → Hours (v)

Phase 2

(1) ECode → dept (v)

E101 → System

(2) Proj Code → dept (x)

3) ECode → dept & Head (v)

Proj Code → dept head (x)

PK. 2NF Master

ECode	dept	dept head
E101	System	D1
E102	Sales	D2

FK. child

ECode	Proj code	Hours
E101	P1	5
E101	P2	3
E101	P3.	4.
E102	P1	
E102	P3	

30/7/2012

Candidate key, fully depend, partially dependant

stid	Courseid	Grade	address
101	CS111	A	101 Ameerpet
102	CS111	B	102 SRNagar
103	CS222	C	103 kukatpally
104	CS222	A	108 Banjarahills

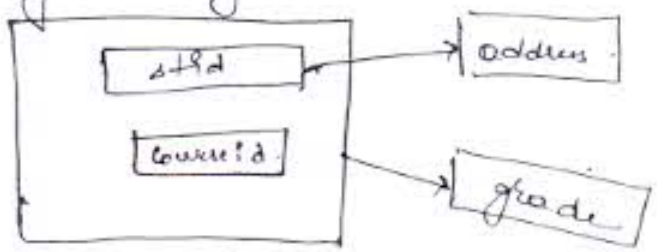
2NF table

stid	~~Candidate~~ Courseid	grade

2nf Tables

stid	address

Logical diagrams



Functional Dependancy

→ Def: If any given two tuple in a relation 'r', if x (an attribute set) agrees then y (another attribute set) also agree then y agrees then y (another attribute set) also agree then

$x \longrightarrow y$ is called functional dependancy.

$x \to y$ y functionally dependant on x

(or)

x functionally determines y

attributes.

stid	stname	Hall
101	aaa	H1
102	abc	H1
103	245	H2
104	xxx	H

tuples.

3rd Normal form:-

A table in 2nd Normal form and also non key attributes, only dependant on 1° key

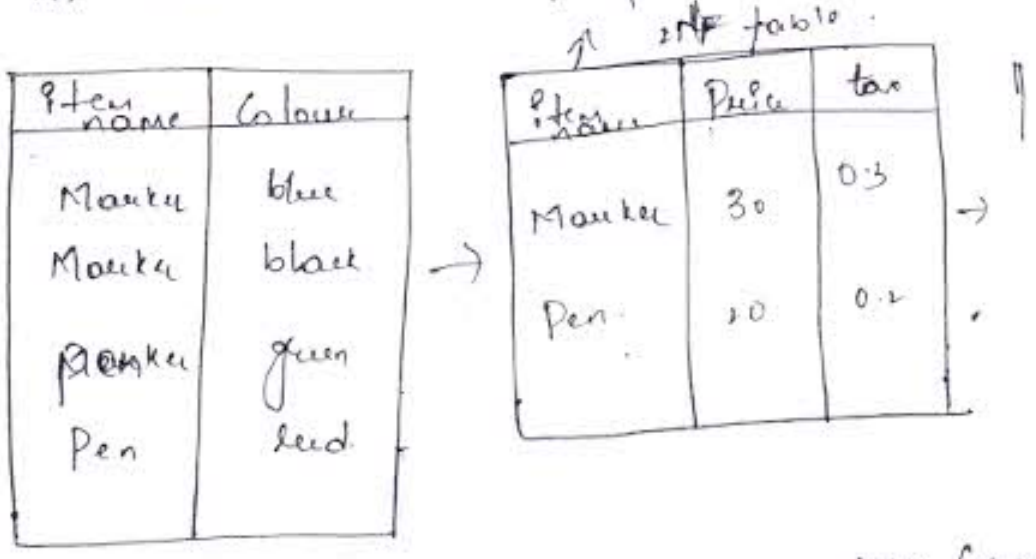
If non key attributes depends on another non key attributes then that table is not in 3rd NF.

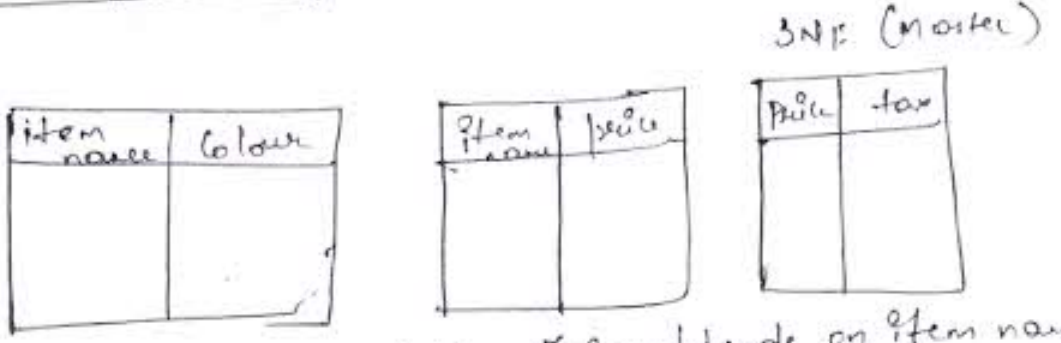
Process:

→ Identify non key attributes, which depends on other non key attributes. put them into seperate table.

→ Then that table is in 3rd NF

→ This table has unique values.





Here in 3NF from above table Price depends on item name, but the non key attribute tax depends on another non key attribute price, hence it is using those two non key attributes an other table is created which is in 3NF.

Order no [ ]

Order date [ ]

Cust name [ ]

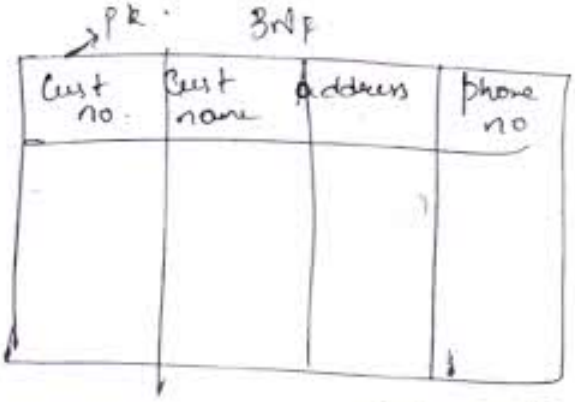
address [ ]

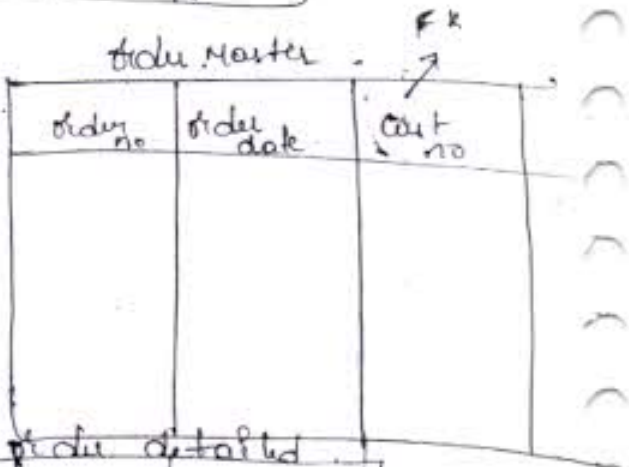
phone no [ ]

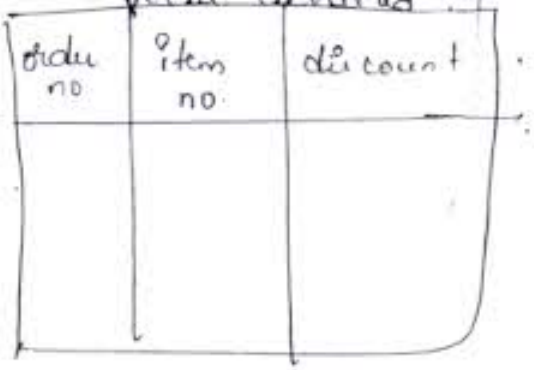
Item name [Item no]

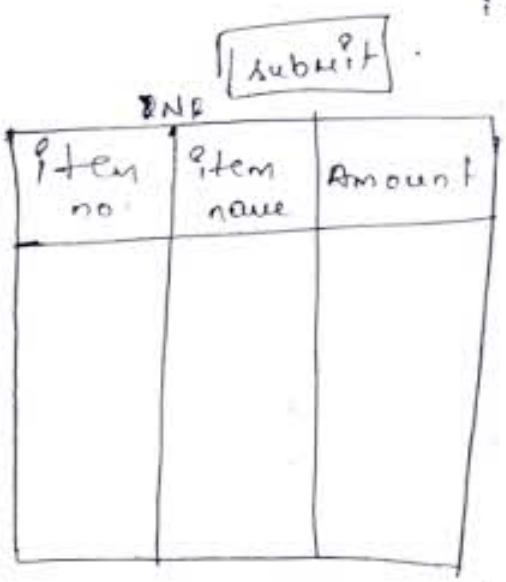
Amount [ ] [ ] add another item.

[Submit]









Customer enquiry form:

Cust no [ ]

Cust name [ ]

address [ ]

phone no [ ]

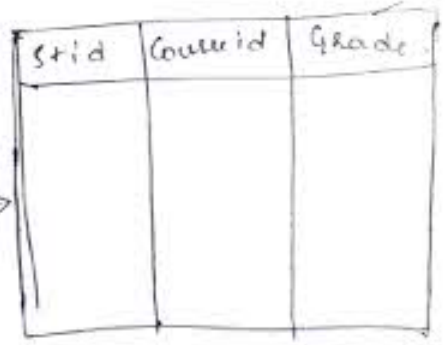
[Submit]

1NF → Remove repeating groups.

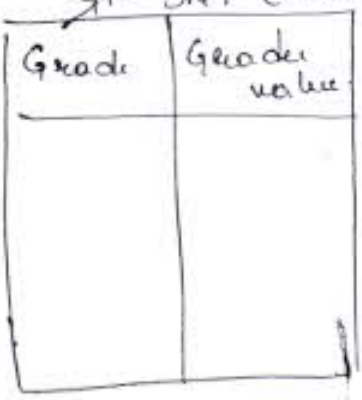
2NF → Remove partial attributes

3NF → Remove attributes that are not dependant on P key.

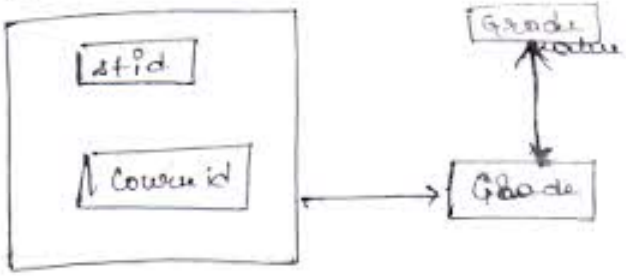
Stid	Courseid	Grade	Grade value
101	CS 111	A	4.00
102	CS 111	B	3.00
103	CS 222	C	2.00
104	CS 222	A	4.00



PK 3NF (Master table



Logical diagram



31/7/2012

Courseid ☐ CourseName ☐

Tutorid ☐ tutorname ☐

stid	sname	D.O.B	gender	Cost attendance

Unnormalized form

UNF

Courseid
Course name
Tutor name
stid
sname
DOB
Gender
Last attendance

1st 1NF

Courseid
Course name
Tutorid
Tutor name
✓ Courseid
✓ stid
sname
DOB
gender
Lastattendance

1NF

Courseid
Course name
Tutorial
tutorname

3NF → pk | Tutorial | Tutorname

2NF → pk | stid | sname | DOB | gender

BCNF:

if a table is in BCNF every determinent is a Candidate key. In their contains only one Composite Candidate BCNF is a 3NF

In 2nd 3rd NF is specifies relationship blw key, non key attributes whereas BCNF specifies relationship blw within Candidate keys. If a table contains more than Composite Candida te keys and also their Candidate key is overlaped and also one Candidate non key attributes functionally dependant on another Candidate key non key attributes.

Process:

Identify one Candidate key nonkey attribute which is functionally dependant on another Candidate key non key attribute put it into seperate table

This table is also called as BCNF table. In this table every determinent is a candidate key and also this table does not contain nonkey attributes.

ECode	Name	Projcode	hours
E_1	Murali	P_1	5
E_2	abc	P_1	7
E_1	Murali	P_2	9
E_3	x43	P_2	12

BCNF

ECode	Name
E_1	Murali
E_2	abc
E_3	x43

determinent.

↑ ECode → Name.

↓

ECode – behaves as a Candidate key

determinent.

↑

Name → ECode.

↓

behaves as a Candidate key.

Prof. Code	dept	HOD	Hours
P_1	Physics	H_1	6
P_1	Computer	H_2	3
P_2	Zoology	H_3	7
P_2	chemistry	H_4	8

dept → HOD

HOD → dept

Prof Code + dept → Candidate key

Prof Code + HOD → Candidate key

Prof Code	Dept	hours
P_1	Physics	6
P_2	Computer	3
P_3	Zoology	7
P_2	chemistry	8

dept	HOD
Phys	H_1
Com	H_2
Zoolog	H_3
chemis	H_4

Fourth Normal form :-

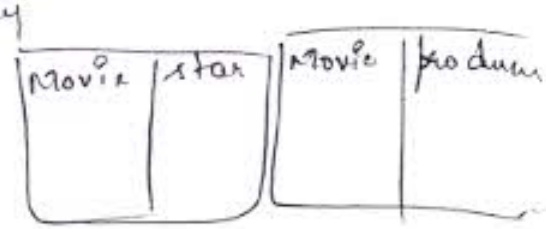
A table is in 4^{th} NF, that table does not contain more than one independent multivalued attribute.

If you want to identify a record using all columns and also one column set of values depends on another column set of values and also one column set of values depends on another third column set of values and also one of the two columns not logically related then we use 4^{th} NF concept process.

Process: identify Columns which are not related into another columns, put it into seperate table. Multi valued attribute represented using $\rightarrow\rightarrow$ candidate key

movie	star	producer
M_1	S_1	P_1
M_1	S_2	P_1
M_1	S_3	P_1
M_1	S_4	P_2
M_2	S_1	P_3
M_2	S_5	P_3

4NF →



1/8/2012

5NF :

→ A table is in 5NF, then that table doesnot contain cycle dependencies.

→ If a table having more than 2 columns the only we apply 5NF

→ This NF is also called as projection join NF

→ Generally in 4NF atleast 2 cols are not logically related

→ But when the table columns are related to each other then we use 5NF

- In 5NF split the table into no of tables & lastly testing Condition.

whenever we are joining these tables resultant record must be available in original table.

Buyer	product	Company
B_1	shirt	Levis
B_1	Jeans	Peppi
B_1	shirt	pepe
B_2	Jeans	Levis

→ 5NF.

Buyer | product product | company Buyer | company

DCL (data Control language)

1.) grant.
2.) Revoke

SQL> Conn sys as sysdba;
password : sys

Syn : Create user username identified by password;
Grant Connect, Resource to username
(or)
Conn username/password

Creating a user.

SQL> Conn sys as sysdba;
password : sys.

SQL> Create user U_1 identified by U_1;
SQL> grant Connect, Resource to U_1;
SQL> Conn U_1/U_1;

SQL > select * from emp;

error.

SQL > Conn scott/tiger;

grant all on emp to u1;

SQL > Conn u1/u1

sql > select * from emp;

error: table or view does not exist.

sql> Conn scott/tiger;

sql > grant all on emp to

sql > select * from scott.emp;

sql > create synonym emp1 for scott.emp;

sql > select * from emp1;

Connect, resource, DBA :-

→ These are predefined roles.

→ There are two types of privileges supported by all database systems for security purpose, they are

1.) System privileges

2.) object privileges.

System privileges: These are given by dba. There are more than 80 sys privileges supported by Oracle

→ Before creating object, user session dba gives privileges to user

→ They are create table, create procedure, create any material views, create session, any index etc.

Syn : grant system privileges to user1, user2 ---;

SQL> Conn sys as sysdba;

password: sys

SQL> grant create table, create any materialized view,
create trigger, create any index to u1, scott, murali;

SQL> grant create table,
create any materialized view,
create trigger,
create any index to
u1, scott, murali;

Role: It is a group of privileges, these are created by database administrator only (dba)

→ Generally in multiuser environment, so many users are working on same project. In this case DBA's giving privileges to individual user is a difficult task

→ To overcome this problem all the dba's introduced role concept.

→ In this case dba identify common set of privileges & assigns it to a role.

and then that role is given to the users

Step 1: Creating a role

Syntax: Create role rolename;

Step 2: Assign privileges to a role

sql> grant system privileges to rolename;

Step 3: Assign rolename to no. of users.

Syn: grant rolename to user1, user2 ....;

Procedure:

Sql> Conn sys as sysdba;

password: sys

Sql> Create role r5;

Sql> grant create procedure, create any view,
create any index, create trigger to r5;

Sql> grant r5 to scott, u1, u2, manali;

* All system privileges related role stored under
role_sys_privs

Sql> desc role_sys_privs

Sql> Select ROLE, PRIVILAGE from role_sys_privs
where role = R5;

2/8/2012

Object Privilages:

These privilages are used to perform some operations on the object. These privilages are given by either developer (or) database administrator.

They are insert, update, delete, select, execute.

Syntax:- Grant object privilages on objectname to username / rolename / public

→ On place of insert update, delete select we all as keyword.

```
sql > Conn scott / tiger
sql > grant all emp to u1;
sql > grant all emp to u5;
```

→ All object privilages related to user stored under user_tab_privs data dictionary.

```
sql> desc user_tab_privs;
```

With grant option: who receives with grant option clause then usernames / public with grant option.

```
sql > grant all on emp to s with grant option;
error :- Cannot grant to a role with grant option
```

Revoke: If we want to cancel system(or) object privilages then we use revoke

Syntax: Revoke system privilages from users, users ...;

Syntax for object privilages:-

Revoke object privilages
on objectname from users / rolename / public;

Sequence: It is an independant database object which is used to generate sequence values automatically.

Generally sequences are used to generate primary key values automatically. Generally sequences are created by DBA. Once if sequence is created number of users simultaneously access that sequences.

syntax: create sequence sequencename
start with n, increment it by n, min value n
cycle/no cycle cache/no cache;

Sql> Create sequence s1
start with 5
increment by 2
max value 100.

If we want to generate sequence values we can use two pseudo columns they are:

(i) currval
(ii) nextval

syntax: Sequence name.currval
(or)
Sequence name.nextval.

Note: If we want to generate sequences values we can use dual table along with the pseudo columns.

syntax: Select sequence name.currval from dual;
Select sequence name.nextval from dual;

Sql> select s1.currval from dual

Error: Sequence s1.currval is not yet defined in this session.

Generally curr value returns Current sequence value & next value returns next sequence value but if we want to generate first sequence no, we are not allowed to use pseudo Column Currval because Currval returns current sequence value of session having a sequence value for this reason if we want to generate first sequence no we use nextval pseudo Column.

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

we can also change properties of the sequences through alter

```
alter sequence sequencename
Propertyname new value;
sql > alter sequence sequencename s1 increment by 3;
sql > select s1.nextval from dual;
```

Note: we are not allowed to change starting sequence number.

```
sql > alter sequence s1 start with 3.
error: Cannot alter starting sequence number
```

Note: start with can't be less than min value

```
sql > create sequence s1
      start with 5
      increment by 2
      min value 10
      max value 100
error; start cannot be less than min value
```

```
sql > Create sequence s1
      start with 10
      increment by -1
        minvalue 10
        max value 100
      /
sql > select s1.nextval from dual;

      10

      10
```

16/8/2012

Oracle Server.

→ Oracle server mainly consists of two parts.

1) storage area.
2) Memory area.

It is also called as database

storage area:

It mainly contains 3 files

→ data files (extension as .dbf)
→ Control files (.ctl)
→ Redo log files (.log)

data files store all database objects information physically

i.e these files store tables, indexes, sequences, procedures, functions, triggers, packages

To view all data files information

dba_data_files

All data files information stored under dba_data_files data dictionary

```
Sql > Conn sys as sysdba ;
Enter password : sys
Sql > desc dba_data_files ;
Sql > Select File Name from
      dba_data_files ;
```

→ Control files Controls all other files in storage area
All Control files information is stored under v$Controlfile data dictionary.
Control files stores database information
Control files are used by database administrator in recovery process

```
Sql > desc v$Controlfile ;
Sql > Select Name from v$controlfile ;
```

→ Redo log files stores Commited information for the transaction
these files are also used by database administrator in recovery process.
All log files information is stored under v$logfile data dictionary

```
Sql > desc v$log file ;
Sql > Select number from v$log file ;
```

Memory area.

whenever we are connecting to the database server using a tool, automatically an instance is created in Ram memory area.

This instance consists of two parts.

1.) Sga (system global area)

2.) Processes

System global area consists of set of buffers, they are

1.) database buffer cache

2.) shared pool

3.) java pool

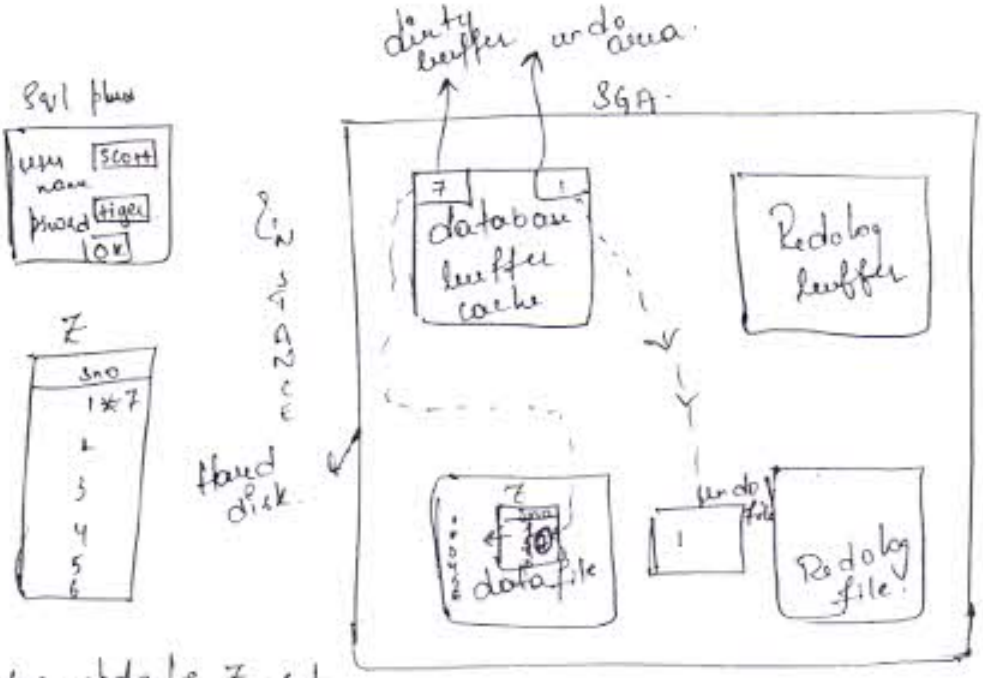
4.) large pool

5.) Redo log buffer

whenever we are submitting sql, plsql code to the database server that code is stored under library cache in shared pool. Always library cache reduces parsing

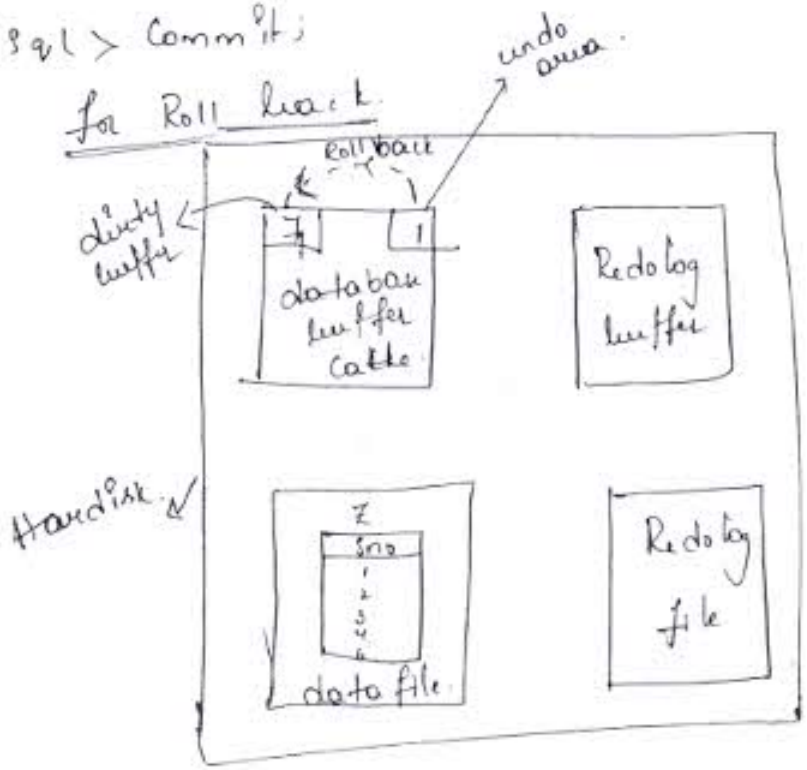
Library cache also stores cache values defined in sequences and also dictionary cache stores ddl related data dictionary information.

whenever we are requesting database objects server process checks whether those objects are available in database buffer cache. If those objects are not available dbwr (data base writer) process fetches the database objects from data files & store it into database buffer cache



Sql > update Z set
Sno = 7 where Sno = 1;

Sql > Commit;



17/8/2012

There are 3 types of structures supported by oracle
1.) Physical structure
2.) Logical structure
3.) Logical storage structure

Physical structure Contains physical files. They are
1.) data files
2.) Control files
3.) Log files
This structure is handled by DBA

Logical structure Contains tables, views, indexes, sequences etc.
This structure is handled by developers

Logical Storage Structure :- It is handled by database admin
- iterator, this structure Contains
1.) Tablespace 2.) Segment 3.) Extent 4.) Block

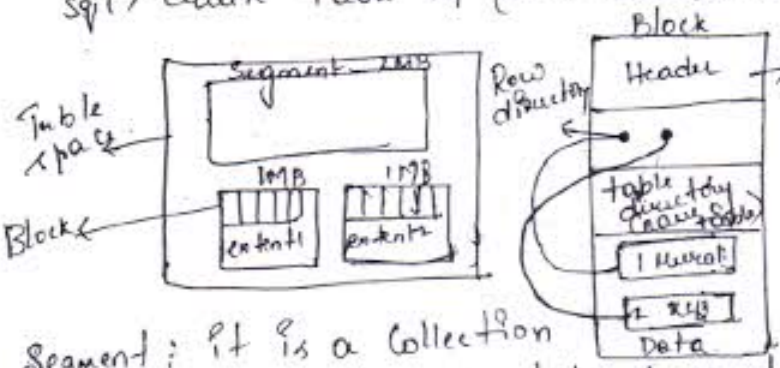
Tablespace :- It is a collection of data files, one data files belong to one table space only, whenever we are installing oracle automatically 6 table spaces are created, if we want to learn oracle min two table spaces are required they are
1.) System table space 2.) System auxillary table space.

System table space Contains metadata i.e it stores all data dictionaries information

Database administrator also creates table spaces explicitly by using following syntax.

Create tablespace tablespace name as 'path of data files'

sql> Create table T1 (sno number(10), name varchar2(10));



Segment: It is a collection of extends that form a data base object like tables, views, sequence etc. There are 4 types of segments supported by oracle.

they are
1.) data segment 2.) Index segment
3.) Temporary segment 4.) undo segment.

sql> insert into T1 values (1, 'murali');
sql> insert into T1 values (2, 'xyz');

Synonyms :- synonym is an alias name created by database administrator security purpose.

synonyms hides another schema username, object name, database link.

There are two types of synonyms supported by object they are
1.) public synonyms 2.) private synonyms

Before creating public synonym, we must give privilisation to user otherwise it will rise an error.

syntax: grant public synonym to user1, user2 ....

Ex :- sql> Conn sys as sysdba;
Enter password: sys
Connected
sql> grant create public synonym to u1;

By default synonyms are private

syntax :- Create synonym synonymname for username.objectname @database link;

Ex :- scott/tiger
sql> grant all on emp to u1, murali;

u1/u1
sql> select * from scott.emp;
sql> Create synonym www for scott.emp;
sql> select * from www;
sql> Conn sys as sysdba;
password: sys
sql> grant create public synonym to u1
sql> Conn u1/u1
sql> Create public synonym qq for scott.emp

murali/murali
sql> select * from www
error; www is not a public synonym

sql> select * from qq

18/8/2014

Indexes:

→ It is a database object, which is used to retrieve data from database very quickly. Thats why index mechanism always improves performance of the app.

→ Indexes are created by DBA's

→ Generally Indexes are created on table columns

→ Generally indexes are created in two ways.

1.) automatically.

2.) Manually.

Automatically: whenever we are creating primary key, unique key oracle server automatically creates B-tree index on those columns.

Manually: we can also create indexes manually using create index privilage.

Syntax: Create index indexname on
tablename (col1, col2 .....);

Note:

Whenever we are using where clause (or) order by clause then the oracle server searches for indexes in data dictionary but if where clause contains not equal to, is null, is not null operators, then oracle server does not search for indexes.

There are two types of indexes supported by oracle.

1.) b-tree indexes.

2.) Bitmap Indexes.

→ whenever we are using where clause, those columns does not have indexes, then oracle server uses full table scan whereas when indexes are available oracle server uses index scan

B-tree indexes:

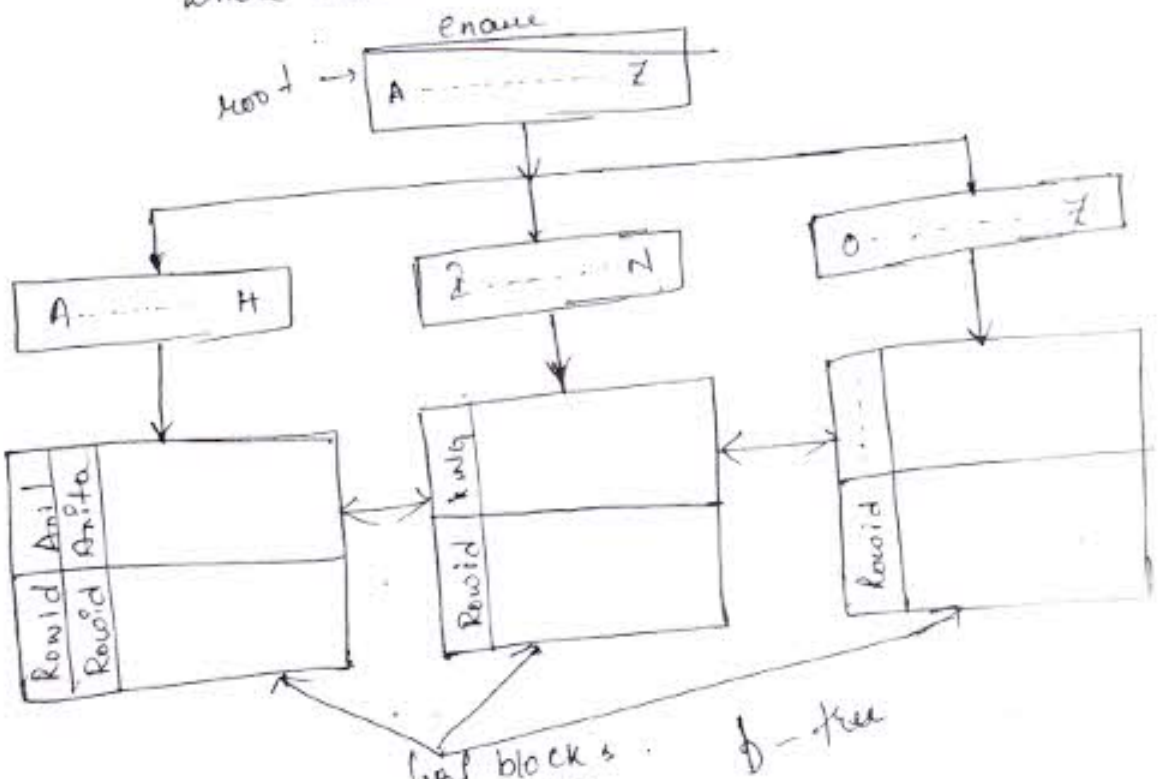
→ In oracle by default indexes are b-tree indexes

→ whenever a btree index is created oracle server automatically creates btree structure.

→ In this btree structure leaf block stores actual data along with rowid

→ whenever we are requesting data using where clause oracle server uses index scan in btree structure, that's why it retrieves data very fast.

```
SQL> select * from emp
     where ename = 'KING';
```



→ All indexes information stored under user_indexes data dictionary

```
Sql > desc user_indexes;
sql > select INDEX_NAME, INDEX_TYPE
      from user_INDEXES
      where TABLE_NAME = 'EMP';
```

INDEX_NAME	INDEX_TYPE
PK_EMP	NORMAL
IN1	NORMAL

→ To view indexes along with column names we use user_ind_column

* if we want to view columnnames along with indexnames we use user_IND_Columns data dictionary

```
Sql > desc User_Ind_Columns;
sql > select index_name, column_name
      from user_ind_Columns
      where table_name = 'EMP'
```

Index_name	Column_name
Pk_EMP	empno
IN1	ename

→ There are two types of btree indexes supported by oracle
1.) Non-unique indexes.
2.) unique indexes

By default indexes are non-unique indexes

Unique Indexes:

Whenever we are creating a unique key Oracle server creates unique indexes on those columns

→ we can also create unique indexes explicitly using following syntax: Create unique index indexname on tablename (columnname);

Note: we cannot create unique index on duplicate value columns.

Bitmap Indexes:

Oracle 7.3 introduced bitmap indexes, bitmap indexes are used in data warehousing applications.

Generally bitmap indexes are created on low cardinality columns

These columns have less possibility values, more no. times (duplicate values)

whenever we are creating bitmap index automatically oracle server creates bitmap index

whenever user using logical operators on where clause dirc -tly bits are affected and resultant bitmap is automati -cally converted into record's using a bitmap function

Syntax: Create bitmap index indexname on tablename (columnname);

If duplicate values are present we use bitmap index.

Create Bitmap index IN3 ON EMP(JOB)

Bit Map Index

Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Clerk Sales	1	0	0	0	0	0	0	0	0	0	1	1	0	1
Salesman	0	1	1	0	0	0	0	0	0	1	0	0	0	0
Manager	0	0	0	1	0	1	1	0	0	0	0	0	0	0
Analyst	0	0	0	0	0	0	0	1	0	0	0	0	1	0
President														

Function Based Indexes:

Oracle 8i introduced function based indexes. Function based indexes are by default btree indexes.

Generally when a where clause contains functions, expressions oracle server don't search for indexes of those columns having already having indexes also

To overcome these problems oracle 8i introduced function based indexes

Syn:

Create index Indexname on tablename
(function name (Columnname));

Ex: create index fng on emp (upper (ename));

sql> select * from emp where upper(ename) = 'SMITH';