**Oracle 19c:**

- Oracle is a relational database which is used to store the data or information permanently into secondary storage devices.

- If you want to operate oracle database then we are using non-procedural language – SQL and also we can use procedural language pl/sql.

-All Organization stores data or information for future purpose.

**Data:**   
It is a collection of raw facts.  
ex: student marks  
customer names

**Information:**

information is nothing but meaningful data or process data, when we are processing data, then we are achieving meaning full result, this is called information.

ex: student mark sheet.  
 invoice of customer.

**data Store:**

it is a place where are can store data or information.

1) papers & books.

2) Flat Files (.csv, .txt, excel.....)

3) Database

**1) Flat Files:**

It is a traditional mechanism which is used to store data or information permanently into secondary devices.

Drawbacks:

. Data retrieval

. Data Redundancy(duplicate)

. Data Integrity

. Data security

. Data Indexing.

**Data Retrieval:**

If we want to retrieve data from flat file then we must develop application programs in high level language, where as if we want to retrieve data from database then we are using SQL.

**Data Redundancy(duplicate/repeated):**

. In Flat files sometimes we are maintaining multiple copy of the same data in different location.

. this data is also called as duplicate data or redundant data.

. if flat file whenever we are modifying this data is one location, it is not affected in other location. This is called inconsistency; that’s why flat files don’t maintain consistency on data automatically.

. Databases automatically maintain consistent data through transactions every transaction internally having four properties;

***these are also called ACID property.***

. If you want to reduce duplicate data in database, then we are using “Normalization Process”

**Data Integrity:**

Integrity means to maintain proper data. If want to maintain proper data in database then we are using constraints (Primary key, foreign key,) where as a flat file we must develop application program in high level language for maintaining valid data

**Data Security:**

Data stores in flat files can’t be secured because flat files do not provide security mechanism, whereas database provide role based security.

**Data Indexing:**

If we want to retrieve data very fast from database, then we are using indexing mechanism on database. Whereas flat files don’t support indexing mechanism.

**Database Management System (DBMS)**It is a collection of programs(software) written to manage software.

Example:  
 ORACLE 19c –latest version 21c   
 DB2, Teradata

Sql Server

MySQL

SQL-Lite

Informix

Sybase....

- In **flat files** mechanism every application programme maintain its own file separate from other application programme in the organization

**Data Management in Flat Files**

File2

File1

File3

Operating system (File system)

App Prog

App Prog

App Prog

Whenever we are installing DBMS s/w into our system can automatically some places is created in hard disk is called Database.  
and also automatically an use interface is created through this user interface we can directly interacting with an database or through the application programme we are indirectly with database.

App Prog3

App Prog1 **Oracle**

App Prog2   
**Sql server**

Database Management System

Operating system (File system)

User interface

|  |  |
| --- | --- |
|  |  |
|  |  |

username

pasword

**Database**:

it is an organized collection of interrelated data.   
**or**  
 It s a collection of structured data.

Every data base having 2 types of structure:  
 1) Logical Structure  
 2) Physical Structure

1. **Logical Structure:** A Structure which is not visible in operating system is called logical Structure. Logical structure contains tables, views, sequence, synonyms. Etc. logical structure is handle by database developer or either database administrator.
2. **Physical structure:**

A structure which is visible in operating system is called physical structure. Physical structure handled by DBA only.

**DBMS Architecture:**

ANSI (American National standard institute) as established three level architecture for DBMS

They are:

1. Conceptual Level
2. External level
3. Internal level

View3

View2

View1

Conceptual Level

Internal Level

Database

**Data Independence:**

Data independence is nothing, but upper level is affected by changes in lower level.

DBMS architecture is having 2 types of data independence:

1. **Logical Data independence**
2. **Physical Data independence**

**Logical Data Independence:**

Changes to the conceptual level do not require changes to the external level is called logical data Independence.

Ex: adding a new entity in conceptual level shouldn’t affect in external level.

**Physical Data independence:**

Changes in the internal level, then do not require changes to the conceptual level is called physical data independence.

Ex: Adding a new entity in internal in internal level shouldn’t affect in conceptual level.

**Data Model**

How data is represented at the conceptual level defined by means of ***data model***.

In the History of Database design three data model have been used, they are-

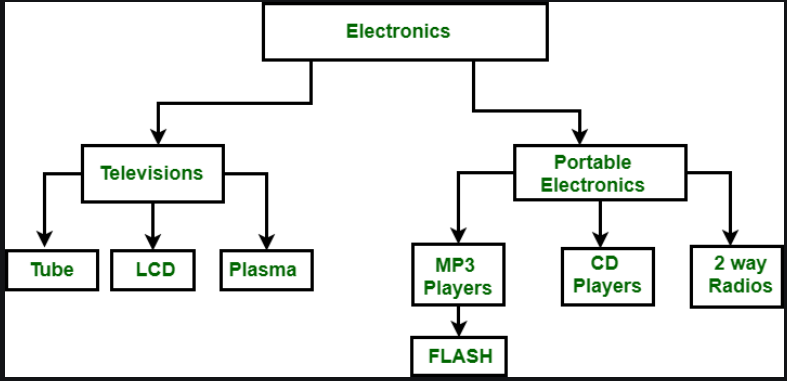
DBMS 🡪 RDBMS

1. Hierarchical data Model
2. Network Data Model
3. Relational Data Model

**Hierarchical Data Model: (*1960’s onwards)***In this model data organize in tree structure format, in this data model data is represented in format of records.

Hierarchical Data model is implemented based on one to many (1: M) relation between parent, child records. Based on this relation many child records having single parent records.

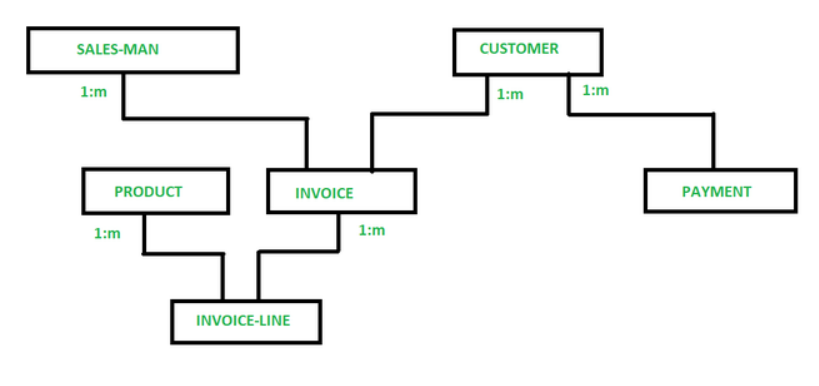
That why in this Data Model always child records are repeated. That why in this data model having more duplicate data.



**Network Model:**

In 1970’s CODASYL (Conference on Data System Language) committee introduced network Data Model.

In this data model, having fewer duplicate data because in this model, parent child relationship implemented based on ***many to many*** relationship(M:M). in this model also data is represented in the format of records and records type is same as table in relational data model.



**Relational Data Base Model (RDBS):**

In 1970’s E.F.CODD introduced relational data model. This data model consists of relations, this relation is in the form of table.

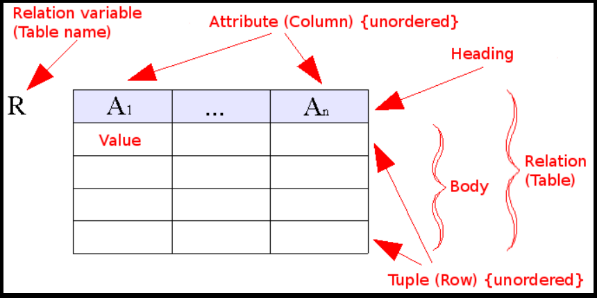
That’s why in relational model we can store the data in 2-D table.

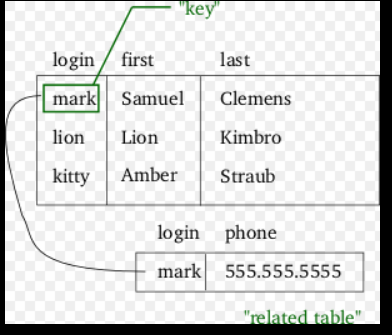
Relational data model mainly consists of 3-Components these are –

1. Collection of Objects

Eg: Tables, Views, indexes, synonyms, clusters etc…

1. Set of Operators
2. Set of Integrity rules





1970 EF CODD Written a paper “Relational Data Model for large Shared databanks” [ALPHA Language]

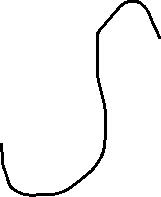
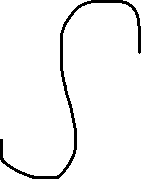


Oracle -RSI-1978

Oracle corp

INGRESS

System/R-IBM

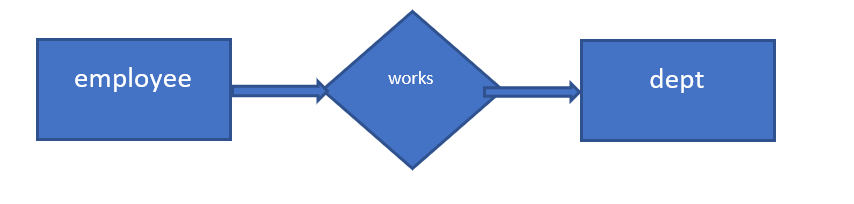


### **RDBMS vs. DBMS**

In general, databases store sets of data that can be queried for use in other applications. A database management system supports the development, administration and use of database platforms.

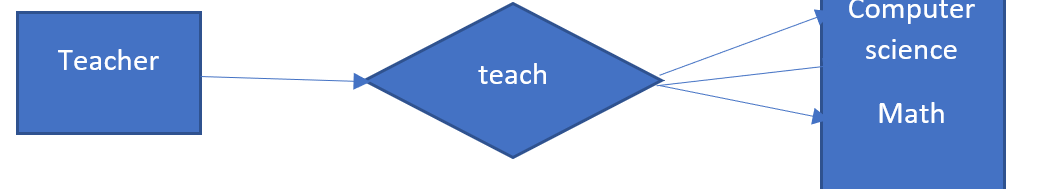
An RDBMS is a type of [database management system](https://searchsqlserver.techtarget.com/definition/database-management-system) (DBMS) that stores data in a row-based table structure which connects related data elements. An RDBMS includes functions that maintain the **security, accuracy, integrity and consistency** of the data. This is different than the file storage used in a DBMS.

**Entity Relationships:**In Any database having the following types of relationships, they are  
1) 1 : 1 (one : one)  
2) 1:M (one : Many)   
3) M:1 (Many : one)  
4) M:M (Many : Many)

**1 :1 Relationship:**

In this relationship, one employee having works for only 1 dept.

1:M Relationship:

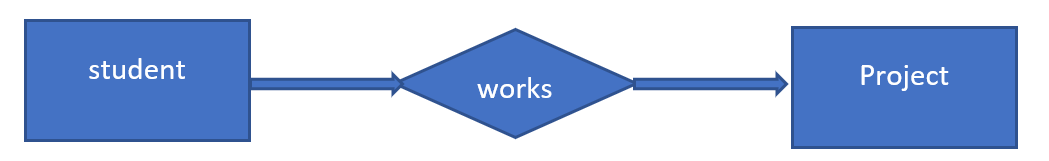


**M:M Relationship:**



One book may have many author’s, and an author can have multiple books

**M:1 Relationship:**



Many students work in a single project, this is many to one relationship

**SQL (Structured Query Language)**

1. SQL is a non-procedure language which is used to operate all Relational Databases
2. In 1970’s E.F.CODD Introduced DSL/ANSI SQL Language, which is used to operate relational Database.
3. In IBM System/R team introduced simplified version of dsl/Alpha language “**Square**” again IBM changed from “***Square***” to ***SEQUEL***” (Structured English Query Language) and again IBM changed from ***SEQUEL*** to ***SQL***

**What is table?**

The data in RDBMS is stored in database objects called tables. The table is a collection of related data entries and it consists of **columns** and **rows**. Remember, a table is the most common and simplest form of data storage in a relational database.

Following is the example of a CUSTOMERS table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | RAMESH | 30 | DELHI | 10000 |
| 2 | VIGNESH | 28 | CHENNAI | 15000 |
| 3 | RAVI | 29 | BENGALORE | 2000 |

**What is field?**

Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY. **A field is a column in a table** that is designed to maintain specific information about every record in the table.

**What is record or row?**

A record, also called a row of data, is each individual entry that exists in a table. For example, there are 7 records in the above CUSTOMERS table. Following is a single row of data or record in the CUSTOMERS table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | RAMESH | 30 | DELHI | 10000 |

**What is column?**

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

For example, a column in the CUSTOMERS table is ADDRESS, which represents location description and would

consist of the following

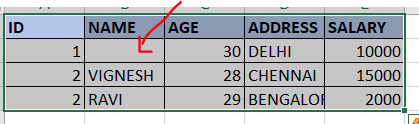
|  |
| --- |
| **NAME** |
| RAMESH |
| VIGNESH |
| RAVI |

**What is NULL value?**

A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a *field with no value*.

It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

A field with a NULL value is one that has been left blank during record creation



SQL is defined with different Sub Languages to operate the structured data.

They are,

1. **DDL (Data Definition Language)** 
   1. *. Create*
   2. *. Alter*
   3. *. Drop*
   4. *. Truncate*
   5. *. Rename (Oracle 9i)*
2. **DML (Data Manipulation Language)**
   1. *Insert*
   2. *Update*
   3. *Delete*
   4. *Merge (Oracle 9i)*
3. **DQL (Data query language) or DRL (Data retrieval Language)**
   1. *Select*
4. **TCL (Transaction Control Language)**
   1. *Commit*
   2. *Rollback*
   3. *Savepoint*
5. **DCL (Data Control Language)**
   1. Grant
   2. Revoke

**Data types:**

Data Types identifies type of data within a table column. Oracle having different type of datatypes,

1. **Number (P, S):**

P 🡺 Precision (total no. of digits)

S 🡪 Scale

It is used to store fixed, floating point numbers.

**Syntax**: *Column\_Name Number(p,s)*

Ex: sql> Create table test( sno number(7,2) );

Sql> Insert into test valus(12345.67)

Sql> select \* from test;

Ouput: 12345.67

Sql> insert into test values (123456.7);

Error: value larger than precision allowed for this column.

**Note**: *whenever we are using number(p,s) format then we are not allowed to insert more than (p-s) no. of digits before decimal point*.

Example : number(p,s) 🡺 p-s => number(7,2)

7-2 => 5

12345.67

123456.7 – error

Note: whenever we are using Number(p,s) and also when we are try to insert more no.of digits after decimal point then oracle server doesn’t return any error.

In this case oracle server internally round at number based on the maximum size specified in scale within declaration

Number(p):

It is used to store fixed numbers.  
**Syntax:**   
 *Column\_Name Number(P)*

Ex: SQL> Create table test1 (sno number(7));

SQL> insert into test1 values (99.90)  
 SQL> select \* from test1;  
 result: 100

SQL> insert into test1 values (99.40)

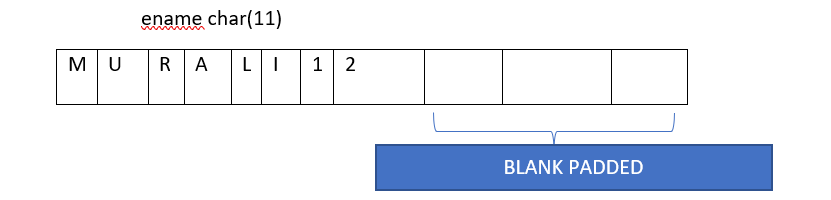
**Note:** *Oracle number datatype maximum precision is up to 38 digits*

1. **Char():**

It is used to store fixed length alpha numeric data in bytes. Maximum limit is up to 2000 bytes.

Syntax***: Column\_name Char(maxsize)***

Default Character datatype having 1 Byte size. When we are trying to store less no.of bytes then the data type size specified in character Datatype. Then oracle server automatically adding blank spaces in the place of remaining bytes after end of the string. This is Called **Blank Padded** mechanism. To overcome this problem oracle provided varchar2 datatype.



1. **Varchar2 –(maxsize)**

Oracle 7. Introduced varchar2 datatype it is used to store variable length of alphanumeric data in bytes. Max size is 4000 bytes.

Syntax**: Column\_Name varchar2(maxsize)**

Note: when we are trying to store less no of bytes, then data type size specified in varchar2 datatype. Then oracle doesn’t add blank spaces at the end of the string. That’s why this datatype doesn’t waste the disk space.

Name varchar2(10)  
 -------------------------  
 MURALI

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| M | U | R | A | L | I |  |

1. **VARCHAR (MAXSIZE)**

Prior to oracle 7.0 oracle having varchar datatype, it is also same as varchar2 datatype. Varchar also stores variable length alphanumeric data in bytes, but max size of the varchar datatype is up to 2000 bytes.

Syntax:  
***Colmn\_Name varchar(maxsize)***

1. **Date:**

It is used to store data in oracle date format, in oracle by default

Date format is “DD-MON-YY’. Ex: 10-Jan-22

Syntax: ***Column\_Name Date***

**SQL**

1. **Data Definition Language (DDL):**
2. Create
3. Alter
4. Drop
5. Truncate
6. Rename (oracle 9i)

DDL Commands are used to define structure of the table.

1. **Create:**

It is used to create the database objects like tables, views, synonyms, indexes….

Creating a table:

Syntax:  
 ***create table table\_name (column1 datatype(size), column1 datatype(size)…..);***

Ex:   
 sql> create table first (sno number(10),sname varchar2(10));  
   
 To view the structure of the table:

Syntax: ***desc table\_name***

Ex: desc first;

1. **Alter:**

It is used to change existing table structure

Alter can perform below operations:

* *Add*
* *Modify*
* *Drop*
* **Add:**

It is used to add columns into existing table.

*Syntax:* ***Alter table table\_name add(col1 datatype(size),col2 datatype(size2)……….)*;**

**Alter table test add(address varchar2(30),country char(2));**

* **Modify:**

It is used to changed column datatype or datatype size only.

**Syntax:**

*Alter table table\_name modify(col1 datatype(size),col2 datatype(size)……);*

ex: *Alter table test modify(address varchar2(50),country varchar(10));*

* **Drop:** it is used to drop columns from existed table.

it has 2 methods:

**Method 1:** - if we want to drop single column at a time without using parenthesis, then we are using following command

**Syntax:** **Alter table table\_name drop column column\_name;**

*Ex: alter table first drop column address;*

**Method2:**- if we want to drop single or multiple columns at a time with using parenthesis, then we are using following

syntax:

**Alter table table\_name drop(col1,col2,col3…**…..);

Ex: *alter table first drop (address,country);*

Sql> desc first;

**Note:** in all databases we can’t drop all columns in the table.

1. **Drop:**

It is used to remove the Table/ objects from the database permanently.

Syntax:

*Drop object object\_name  
or* ***drop table table\_name***

*or drop view view\_name*

ex: desc first;

sql> drop table first;

testing:

sql> desc first;

error: object first doesn’t exist.

1. **Truncate:**

Oracle 7.0 introduced truncate table command; it is used to delete all rows permanently from the table.

Syntax:

*Truncate table <table\_name>;*

;

Example:

SQL> create table test1 as select \* from first;

SQL> select \* from first;

SQL> truncate table first;

Testing:   
 SQL> select \* from first;

No rows selected.

SQL> desc first;

1. **Rename:**

It used to rename a table.  
syntax:  
 **rename old\_tablename to new\_table\_name;**

Ex:   
sql> **rename first to second;**

ds

**Renaming a column-(oracle 9i):**

Syntax:  
***alter table table\_name rename column old\_column\_name to new\_column\_name;***

Ex:

SQL> alter table first rename column address to add;

SQL> Select \* from first;

Note: in all databases by default all DDL commands are automatically COMMITTED(save).

**Data manipulation language (DML) :**

it is used for manipulating the data within a table.

­

1. Insert
2. Update
3. Delete
4. Mer**ge**

**Insert:** it is used to insert the data into the table.

**Menthod1:** syntax*: insert into <table\_name> values(value1,value2,value3……);*

Ex: select \* from first;

Insert into first values(10,’izaz’);

Insert into first values(20,’ani’);

**Method2: (*using substitutional operator)***

**&** - enter values for

**Syn***: insert into table\_name values (&column1,&column2,……..);*

Ex: insert into first values (&sno,&sname);  
enter the values for sno: 20  
enter the values for sname:shahid

Sql> / [this gives the previous commands]

**Method3: (skipping columns)**

Syntax: *Insert into table\_name (col1,col2,…) values (value1,value2…..);*

Ex: insert into first (sno,sname,address) values (100,’abc’,’nellore’);

1. **Update:** it is used to change the data in a table.

Syntax:

***Update table\_name set column\_name=new\_value where***

***Column\_name=old\_value;***

Ex: Update first set country=’IN’ where sno=10;

1. **Delete:**

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

Syn*: delete from table; --> all rows will delete*

*Delete from table where condition ;--> only specified condition records only will delete*

Sql> delete from first;  
 rows deleted.

Or

Sql> delete from first where sno=10;

**To get it back data:-**   
 sql> rollback;

Sql> select \* from first;

**Rollback:** if we need to restore the deleted data using rollback command achieve.

**Commit:**  
 if we use commit command the traction success fully complete which means can’t rollback the data.

**Difference between delete and truncate:  
Delete:**

1.Whenever we are using **delete** from the table then automatically deleted the data internally stored in buffer, we can also get it back by using ***rollback*** *command*.

2. where delete particular data using **where** clause in **delete** statement

**Truncate:**

1. Whenever we are using **truncate** table, then all rows are permanently deleted we can’t get it back this data by using rollback also because truncate is a ddl command and ddl command **transactions are automatically committed**.

2. truncate command will not support **where** clause condition.

**DQL (Data query Language) Or DRL (Data Retrieval language):**

**Select:**

Select command is used to fetch the data from database tables.

**Syn:** *select \* from table\_name;*

**Syn:** *select col1,col2,…….. from tablename where <condition> group by colum\_name having <condtion> order by column\_name [asc/desc];*

1. Select all column & all rows then use ‘\*’ symbal
2. Select all columns & particular rows then used “*where” clause condition*
3. select particular columns (col1,col2…) & all rows
4. select particular column & particular rows.

== > **Creating a new table from existing table /coping a table from another table: (iq)**

Syn: create table table\_name as select \* from <existing table>;

Ex:  
Sql> create table first1 as select \* from first1.

== > **creating a new table from existing table without copying data: (iq)**

**Syn:** *create table new\_table\_name as select \* from <old\_table> where <false\_condition>;  
ex:* Sql> create table first2 as select \* from first where 1=2;

Sql> select \* from first2;

Result: no rows selected

Sql> desc first2;

**Operators used in select statement/Oracle SQL Operators:**

1. **Athematic operators**

\* ---- Multiplication

+ ---- Addition

* ---- Subtraction (minus)

/ ---- Division

1. **Relational Operators**

= 🡪 equals to 1=1 T  
< 🡪 less than 2<3 T  
> 🡪 greater than 3>2 T  
<= 🡪 less than equals to 2<=3 T, 3<=2 F  
>= 🡪 greater than equals to 2>=1 T, 2<=1 F  
<> 🡪 not equals to 1<> 2  
!= 🡪 not equals to

1. **Logical Operators:***And   
   Or   
   Not*
2. **Special Operators**

**Note:** *Arithmetic operators are used in* ***number, date datatype*** *columns.*

**Athematic Operators examples**

**Q) Write a query to display ename, sal, ann\_sal from emp table?**

Ans> select ename, sal, sal\*12 as annual\_sal from emp;

**Q) Write a query to display ename, sal, total\_month\_sal from emp table?**

SQL> select ename,sal,comm,sal+comm as total\_sal from emp;

**Assignments:**

**Q) Write a query to display ename,sal and display sal with out commission?**

**Q) Write a query to display ename,sal and his/her day salary?**

**Relational Operators Examples:  
Q ) WAQ to display the employees except job as CLERK from emp table?;**

Ans) select \* from emp where job <>'CLERK';

**Q) WAQ to display the employee who are getting more salary than 2000 from emp table?** Ans) select \* from emp where sal>2000;

***Logical Operators Examples:***

1=1(T) AND 2=2 (T) 🡪 T  
1=1(T) AND 2=3(F) 🡪 F  
1=2(F) AND 2=2(T) 🡪 F  
1=2(F) AND 2=3(F) 🡪 F

1=1(T) OR 2=2 (T) 🡪 T  
1=1(T) OR 2=3(F) 🡪 T  
1=2(F) OR 2=2(T) 🡪 T  
1=2(F) OR 2=3(F) 🡪 F

***AND***:  
Q) **WAQ whose job ‘CLERK’ having more than 2000 salary from emp table?**

Sql> select \* from emp where job='CLERK' AND SAL>1000;

***OR:***  
Sql> select \* from emp where job='CLERK' OR SAL>1000;

Q) **WAQ whose job ‘CLERK’ or ‘Saleman’ from emp table?**sql> select \* from emp where job=’CLERK’ OR job=’SALESMAN’;

**Q) WAQ to display an employees whose deptno is 20,30 and 40 from emp table?**sql> select \* from emp where deptno=20 or deptno=30 or deptno=40;

**4- Special Operators:**

Oracle provides following types of special operators, these are

|  |  |
| --- | --- |
| **in** | **Not in** |
| **Between** | **Not Between** |
| **Is Null** | **Is Not Null** |
| **Like** | **Not like** |

1. **IN:**

It is used to pick the values one by one from list of values. Generally, we can also use “IN” operator in place of “OR” operator when we are retrieving multiple values from single columns in this case IN operator performance is very high compare to “OR” operator.

Syntax:  
 *select \* from table\_name where column\_name* ***in*** *(list of values);*

*Any datatype*

Ex: select \* from emp where deptno in (20,30);

Ex: select \* from emp where ename in (‘SMITH’,’ALLEN’,’MILLER’);

Select \* from emp where deptno =20 or deptno=30;

**Note:** In all DB’s “*not in* “ operator doesn’t work with *null* values.

Ex: select \* from emp where deptno not in (10,20,null);

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

* In all DB’s whenever we are using arithmetic operators and null value then it will become null.

Ex: **null + 50 = null**

Ex: WAQ to display ename,sal,comm,sal+comm from emp where ename is SMITH?

SQL> select ename,sal,comm ,sal+comm as total\_sal from emp where ename= 'KING';

ENAME SAL COMM TOTAL\_SAL

---------- ---------- ---------- ----------

KING 6000

In this case to handle these kinds of issue to handle oracle provide special null handing functions.

1. **Between:**

It is used to retrieve range of values.

**Syntax:**

*Select \* from table\_name where column\_name between low\_value and highvalue;*

Ex:

WAQ to display the employees who are getting SAL between 2000 – 5000 from emp table?

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

*We can write the above query using Relational Operators as well:*

SQL> select \* from emp where sal >=2000 and sal<=5000;

**Not Between:** it is reverse to Between operator

Ex: select \* from emp where sal not between 2000 and 5000;

1. IS NULL, IS NOT NULL:

These two special operators are used in **where** condition only, these operators are used to test whether a column having null value or not.

Note: In all databases we are not allowed to used relational operators along with null value within with where clause condition.

**Syntax:** *select \* from table\_name where column\_name is null;  
 or  
 select \* from table\_name where column\_name is not null;*

**Q) WAQ to display employees who are not getting commission from emp?**

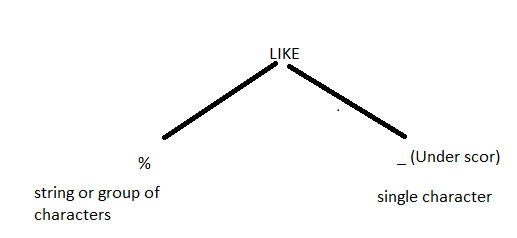
Sql> select \* from emp where comm is null;

**Q) WAQ to display employees who are getting commission from emp?**

Sql> select \* from emp where comm is not null;

**Like, Not Like:**

This is operators are used to search data based on character pattern. like operator performance is very high compare to predefine searching function.



Along with like operator we are using two special operators they are ***% and \_*** (underscore)

These operators are also called wildcard characters.  
  
syntax:

**Select \* From table\_name where column\_name like ‘character pattern’;**

Ex: waq to display the employees whose ename starts with ‘A’ from emp table?

Select \* from emp where ename like ‘A%’

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

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

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

**Like operator with underscore:**

select \* from emp where ename like '\_\_A%';

**Scenarios:**   
1) waq to display the employess whose ename having M in any position within ename column from emp able.

1. Waq to display employees whose ename second letter is start from ‘L’ from emp table?
2. Waq to display employees whose ename 4th letter is ‘M’ from emp table.
3. Waq to display employee who are joining in the month of dec from emp table?
4. Wsq to display the employees who are joining in the year 81 from emp table?

**Concatenation Operator (||)**

Using this operator, we can concatenate more than one string or character

Ex: SQL> select ename||' monthly salary is '|| sal from emp;

SQL> select ename || ' job role is '||job||' and his salary is '||sal from emp;

SQL> select 'hello'||' '||'world' from dual;

**NVL():**

NVL is predefined function which are used to replace or substitute user define value in place of null.

Syntax: nvl(exp1,exp2) --- exp- expression

Here exp1 ,exp2 must belongs to same datatype.

If exp1 is null then it will returns exp2 otherwise it returns exp1;

Ex: select nvl(null,20) from dual; output - 20

Select nvl(10,20) from dual; output – 10

SQL> select ename,sal,comm ,nvl(comm,0) as comm\_test,sal+ nvl(comm,0) as total\_sal from emp;

ENAME SAL COMM COMM\_TEST TOTAL\_SAL

---------- ---------- ---------- ---------- ----------

KING 6000 0 6000

**Interview question:**

How to handle null values? *Ans – using* ***nvl()*** *function*

**NOT IN():** Not In operator is reverse to the ‘IN’ operator

Ex: select \* from emp where deptno not in (10,20);

**FUNCTIONS:**

. Functions are used to solve particular task.

. functions are also return a value.  
. Oracle having 2 types of functions.

1. Predefined functions
2. User-defined functions.

**Predefined functions:**

1. Number functions
2. Character functions
3. Date Functions
4. Group functions (or) aggregate functions

**Number Functions:**

These functions operate over number data.

1. **Abs()** : it is used to convert -ve sign into +ve sign

Ex: sql> select abs(-50) from dual;  
result: 50

**Dual:** dual is a predefine virtual table which contains one column and one row.

Dual table is used to test predefine; user define functions functionality.

Ex: sql> select nvl(null,0) from dual;  
 result: 0

Ex: sql> select nvl(20,10) from dual;  
 result: 20

**ii) mod(m,n) :** it gives reminder after M/N

ex: select mod(10,3) from dual;

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

ex: select round(1.7) from dual;

sql> select round (1.23456,3) from dual;

result: 1.235

50 ->

**iv) trunc ()**: it truncates given floated value number m based on n.

ex: sql> select trunc(1.8) from dual;

result: 1

sql> select trunc(1.23456,3) from dual;

op: 1.234

**ceil(), floor():** these functions always return integer, ceil returns nearest, greatest integer where as floor returns nearest lowest integer.

Ex: select ceil(1.4) from dual; result : 2

Sql> Select ceil(1.9) from dual; result :2

Sql> select floor(1.9) from dual; result :1

**Greatest (exp1,exp2…) , least(exp1,exp2…) :**

Greatest returns maximum value among given expression where as least returns minimum value among given expression

Ex: sql> select greatest (5,6,7,8) from dual;  
 sql> select least(12,4,29,2) from dual;

**B. Character Functions/String Functions:**

**1) upper():** it is used to covert string into upper case and column into upper case

Ex: select upper(‘my first char function’) from dual;

UPPER('MYFIRSTCHARFUNC

----------------------

MY FIRST CHAR FUNCTION

**2) lower():**  it is used to covert upper case values into Lower case

Ex: select lower(‘THIS IS MY FIRST CHAR FUCTION’) from dual;

LOWER('THISISMYFIRSTCHARFUCTI

-----------------------------

this is my first char fuction

Sql> Select lower(ename) from emp;

**3) Initcap():** it returns 1st letter is capital letter and remaining letters are small case

Ex: select initcap(ename) from emp;

Ex2: select initcap(‘my first sql init fuction’) from dual;

**Length():** always it returns number datatype it returns total length of string including spaces.

Ex: select length(ename) from EMP;

Ex: select length(‘ab cd’) from dual;

**Lpad():** it will fills remaining spaces with specified character on the left side of given string. Here always second parameter returns total length of string:

*Syntx: lpad(columnname,total\_length,’filled character’)*

Ex: select lpad(‘abcd’,10,’#’) from dual;

**Rpad:** it is reverse to the lpad

Ex: select rpad(‘abcd’ ,10,’$’) from dual;

**LTRIM,RTRIM,TRIM() functions:**

Ltrim – trim left side white spaces or characters

Rtrim – trim right side white spaces or characters

Trim () – trim left and right side white spaces.

Ex: select ltrim (‘ssmissthss’, ‘s’) from dual;

Op: missthss

Ex: select job,ltrim(job,’CL’) from emp;

Ex: select rtrim(‘SSMISSTHSS’,’S’) from dual;

Op: SSMISSTH

Ex: select trim(‘S’ from ‘SSTHSSMISS’) from dual;

Note : we can use trim for remove the leading & trailing spaces.

**Translate() , replace():**

Translate is used to replace character by character whereas replace() us used to replaces character string by string.

Ex: select translate(‘INDIA’,’IN’,’XY’) from dual; result: XYDXA

Ex: select replace(‘INDIA’,’IN’,’XYZ’) from dual; result: XYZDIA

Ex: select replace (‘A B C’,’ ‘, ‘PQR’) from dual;

Ex: select job, replace(job,’SALEMAN’,’MARETING’) from dual;

**Concat():** to concatenate given 2 strings

Sql> select concat(‘wel’,’come’) from dual;

Op: welcome

**SUBSTR():**

It will extract portion of the string within given string based on last 2 parameters.  
**syn:** *substr(column\_name, starting\_position, no.of characters from position)*

*column\_name –* input data.  
 *starting\_position - > -ve or +ve  
 no.of characters from position -> always from left to right (Always +ve)*

ex: select substr(‘ABCDEFGHI’,1,3) from dual;  
 Result: ABC

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | B | C | D | E | F | G | H | I |

Ex: select substr(‘AB CDEFGHI’,1,3) from dual;

Result: AB

Ex: select substr(‘ABCDFGH’,-1,3) from dual;

SQL> select substr('ABCDFGH',-2,3) from dual;  
Result: GH

SQL> select substr('ABCDFGH',-3,3) from dual;

Result: FGH

SQL> select substr('ABCDFGH',-4,5) from dual;

Result: DFGH

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | A | B | C | D | E | F | G | H |
|  | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

Q1) WAQ to display the emplyees whose ename 2nd letter would be ‘LA’ from emp table by using substr();

Answer: SELECT ENAME FROM EMP WHERE SUBSTR(ENAME,2,2) ='LA';

Q2) WAQ to display the employees whose ename length is 5 characters from emp table by using length();

Result: SELECT \* FROM EMP WHERE LENGTH(ENAME)=5

**INSTR():**

Instr always returns number datatype, i.e it returns position of the delimiter position of the delimiter, position of the character, position of the string within given string.

Syn : **instr (column\_name, ‘str’, [searching position], [number of occurrences from position])**

***Column\_name*** : given input string  
 ***Str***  : searching string  
 ***Searching Position***: position of the index (-ve or +ve)   
***number of occurrences from position*** : --------- > (no of occurrences from the position)  
 < ---------

Always instr() returns position based on last 2 parameters but oracle server counting number of characters left side first characters onwards.

Ex: select instr(‘ABC\*D’,’\*’) from dual;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| A | B | C | D | E | F | G | H | C | D | I | J | K | L | C | D | M | N | P |
| -19 | -18 | -17 | -16 | -15 | -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-5,2) from dual;

Result: 9  
SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-5,3) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-5,3)

--------------------------------------

3

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-4,2) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-4,2)

--------------------------------------

9

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-4,3) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-4,3)

--------------------------------------

3

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-4,5) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-4,5)

--------------------------------------

0

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-4,4) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-4,4)

--------------------------------------

0

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-3,2) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-3,2)

--------------------------------------

9

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-3,3) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-3,3)

--------------------------------------

3

SQL> select instr('ABCDEFGHCDIJKLCDNMP','CD',-3,1) from dual;

INSTR('ABCDEFGHCDIJKLCDNMP','CD',-3,1)

--------------------------------------

15

Q) Waq to get the below data by using **substr** and **instr** functions? \*\*\*  
input data:  
[saishaik123@gmail.com](mailto:saishaik123@gmail.com)  
[izan11@gmail.com](mailto:izan11@gmail.com)  
[shahid@gmail.com](mailto:shahid@gmail.com)  
[anil1@gmail.com](mailto:anil1@gmail.com)  
[Radha@gmail.com](mailto:Radha@gmail.com)  
  
**output:**saishaik123  
izaz11  
shahid  
anil  
Radha

**Answer:**   
SQL> select mailname, substr(mailname,1,instr(mailname,'@')-1) name from email;

**Q) write a query to display the data which is start and end character is same? \*\*\***

select \* from frstlstmtch where substr(col1,1,1)=substr(col1,-1,1);



**FOR REVERSE:**

SELECT COL1,SUBSTR(COL1,1,1) FIRST\_CHAR,SUBSTR(COL1,-1,1) LAST\_CHAR FROM frstlstmtch

WHERE SUBSTR(COL1,1,1)<>SUBSTR(COL1,-1,1);

**Date Functions:**

In oracle by default date format is DD-MON-YY

It has following date functions; they are:

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

**Sysdate:** it returns current date of the system in oracle date format  
sql> select sysdate from dual;

**Add\_months():** it is used to add or remove no.of months from the specified date based on 2nd parameter.

Syntax: *add\_months(date,number)*

Eg: select add\_months(sysdate,5) from dual;

Eg: select add\_months(sysdate,1) from dual;

Eg: select add\_months(sysdate,-1) from dual;

**Last\_day():** it returns last date of the specified month

Syntax: last\_day(date)

Eg: select last\_day(sysdate) from dual;

**Next\_day():** it returns next occurrence day from the specified date based on 2nd parameter

Syntax: next\_day(date,’day’)

Eg: select next\_day(sysdate,’monday’) from dual;

**Months\_between():** it returns number datatype , that is it returns no.of months between 2 specified dates.

Syntax: months\_between(date1,date2)

Note: here date1>date2 otherwise this fx returns -ve sign;

Eg: select ename, round(months\_between(sysdate,hiredate)) from emp;

For More Date Functions Follow the below Link:

<https://docs.oracle.com/cd/E17952_01/mysql-5.7-en/date-and-time-functions.html>

**Date Athematic:**

1) Date + Number –> it will allow   
2) Date – Number -- > it will allow   
3) Date1 + Date2 -- > not allow  
4) Date1- date2 -- > it will allow

Eg: select sysdate +1 from daul;

Eg: select sysdate-1 from dual;

Eg: select sysdate -sysdate from dual;

**Date conversion Functions:**

1) to\_char()

2) to\_date()

**To\_char():**

It is used to convert oracle date type into character type , that is , it converts date type into date string.

Eg: select sysdate from dual;

Eg: Select to\_char(sysdate,’dd/mm/yy’) from dual;

Eg: select to\_char(sysdate,’DAY’) from daul;

Eg: select to\_char(sysdate,’day’) from dual;

D DAY  
DD DY  
DDD MONTH   
MM MON  
YY YEAR   
YYYY Q  
HH  
MI  
SS

**Q) How to find the current quarter of the date?**

SELECT TO\_CHAR(date '2035-07-20', 'Q')

FROM DUAL;

Or

SELECT TO\_CHAR(hiredate, 'Q') FROM emp;

**Q) write a query to display the current quarter and previous quarter from emp table?**

SELECT hiredate

, TO\_CHAR ( hiredate , 'YYYY"0"Q') AS current\_qtr

, TO\_CHAR ( ADD\_MONTHS (hiredate, -3), 'YYYY"0"Q') AS previous\_qtr

FROM emp ORDER BY hiredate;

select sysdate,to\_char(sysdate,'yyyy') year1,to\_char(sysdate,'mm') mon ,to\_char(sysdate,'dd') day1, to\_char(sysdate,'Q') from dual;

**TO\_DATE():**

It is used to convert date string into oracle date type.

Eg: select to\_date(‘12/jun/05’) from daul;

SQL> select '09-FEB-05' + 5 from dual; 🡪 error

SQL> select to\_date('09-FEB-05') + 5 from dual; 🡪 14-FEB-05

**Group functions (): \*\*\*\*\*\*\*\*\*\***

In all databases group () functions operate over no. of values in a column and returns single value.  
oracle Server having following group functions.

1) max()  
2) min()  
3) avg()  
4) sum()  
5) count(\*)  
6) count(Column\_Name)

**Max():**

It returns maximum value from a column table.  
  
eg: select max(sal) from emp;  
  
eg: select max(hiredate) from emp;

Eg: select max(ename) from emp;

**Min():**

It returns minimum value from a column table.

eg: select min(sal) from emp;  
  
eg: select min(hiredate) from emp;

Eg: select min(ename) from emp;

**Avg():**

It returns average from number datatype column.

Eg: select avg(sal) from emp;

Eg: select avg(comm) from emp;

**Sum():**

It returns total from number datatype column

Eg: select sum(sal) from emp;

Eg: select sum(comm) from emp;

**Count(\*) :**

It counts no. of a rows in the table.

Eg: select count(\*) from emp;

**Count (columname):**

It counts no.of ***not null*** values in a column.

Eg: select count(comm) from emp;

Eg: select count(mgr) from emp;

**Interview Question:**

I have a data like below the table contain -- single row comment

/\* -- multirow comment

Tab\_1 -- this is table name

col1

1

1

null

\*/

select count(\*) from tab\_1; -- what is the output

select count(1) from tab\_1; -- what is the output

select count(col1) from tab\_1; -- what is the output

create table tab\_1(col1 varchar(2)) ;

insert into tab\_1 values ('1');

insert into tab\_1 values ('1');

insert into tab\_1 values (null);

select col1 from tab\_1;

select rowid,col1 from tab\_1;

select count(\*) from tab\_1; -- o/p: 3

select count(1) from tab\_1; -- o/p: 3

select count(col1) from tab\_1; -- o/p :2

select \* from emp

select 2 from emp;

select count(1) from emp;

select empno,1,3 from emp;

**GROUP BY Clause:**group by class is used to arrange similar data items into set of logical groups. Whenever we are **using group by class** database server selects similar data items from a table column and then reduces no. of data item in each group.

Syntax:  
***select col1,col2,col3,…… from table\_name group by col1,col2,col2;***

**Q) waq to display no. of employees in each department from emp table by using group by ?**

Answer) select deptno, count(\*) from emp group by deptno;

**Assignments:**

1) waq to display no. of employees in each job from emp table by using group by function?

2) waq to find max, min, avg salary from emp table?

3) waq to display each department total salary/sum salary from emp table?

**Execution process:**

In all databases whenever we are submitting **group by** classes *then database server first executes after group by class* ***columns***and also then results are stored in result-set table then only database servers select that result set table data based on specified columns after select.

|  |  |
| --- | --- |
| **deptno** | **Job** |
| -- | -- |
| -- | -- |
| -- | -- |

**SQL>** select deptno,job from emp group by deptno,job;

Step1: Group by deptno, job

Step2: select deptno, job ..

Result\_set table

**Note:** all databases we are trying to display group function with normal function, then database server returns errors , to over come this problem we must use group by class.

**Step1:** select max(Sal) from emp;

**Step2:** select deptno, sum(sal) from emp;

Error: not a single-group group function

**Solution:** select deptno,sum(sal) from emp group by deptno;

**Having: -- this is using for filter – of groping rows**

After group by clause, we are not allowed to use **where class** in place of this one ANSI SQL provided another clause ***having***. Which is restrict after the clause.

Generally, if we want to restrict rows in a able then we are using where clause. Whereas if we want to restrict group by then we must use having clause.

Generally, we are not allowed to use group function in where clause where a in having clause we can also use group function.

**Note:**

**Where & having** both are filter conditions only

***Where*** *– filters only raw data ( have to use very next to table name / before group by cluase)*

***Having*** *– filter only grouped data. (have to use after group by only)*

Q) WAQ to display those dept total sal is more than 9000 from emp table using group by function?

select deptno,sum(sal) from emp group by deptno having sum(sal)>9500;

**Assignments:**

1) Waq to display year, no of employee per year in which more than 1 employee was hired from emp table by using group by function?

Answer: SQL> *select to\_char(hiredate,'yyyy'),count(\*) no\_of\_emps from emp group by to\_char(hiredate,'yyyy') having count(\*)>1;*

2) waq to display those deptno, sum(sal) from emp table those dept having more than 3 employees by using having clause?

SQL> select deptno,sum(sal) sumsal,count(\*) no\_of\_emps from emp group by deptno having count(\*)>3;

**Order by:**

Order by clause is used to either arrange in ascending order by descending order along with ***order by***clause

We are using using 2 keywords , they are, **ASC** or ***DESC***

Default order by cluse having ascending order

**Syntax:** select \* from table\_name order by colum\_name [asc/desc];

Eg: select sal from emp order by sal desc;

Eg select \* from emp order by ename;

SQL> select deptno,sal from emp order by deptno ,sal desc;

SELECT DEPTNO,SUM(SAL) TOTAL\_SAL FROM EMP GROUP BY DEPTNO HAVING SUM(SAL)>9000 order by DEPTNO desc;

**Order of the SQL query Execution: (interview question)**

1) From  
2) Where   
3) Group by   
4) Having   
5) Select   
6) order by

Syntax:

*Select col1,col2,….  
from table\_name  
where <condition>  
group by Column\_name  
having <condition>  
order by col [asc/desc];*

***Understand the below assignment:***

Eg: select deptno,count(\*) from emp   
 where sal>1000  
 group by deptno  
 having count(\*)>3  
 order by deptno desc;

**JOINS**

1) Inner Join / equi Join  
2) Left Outer Join  
3) Right Outer Join   
4) Full Outer Join  
5) Self Join   
6) Cross Join /Cartesian join 2\* 2 =4

**1) Inner Join / Equi Join**

Inner join returns matching rows from both input datasets, *here join conditional column must belongs to same datatype*. When tables having common columns then only we are allowed to use inner join.

*Syntax:  
 select a.col1,a.col2,b.col3,b.col4 from* ***table1*** *a inner join* ***table2*** *b on  
 a.col1=b.col1;*

Eg: *select*

*e.empno,*

*e.ename,*

*e.job,*

*e.sal,*

*e.deptno,*

*d.dname,*

*d.loc*

*from emp e inner join dept d*

*on e.deptno=d.deptno*

***assignment:*** WAQ to display the employees who are working in ‘Chicago’ Location from emp and dept tables using inner join .

***Left outer Join:***

It will return matched records from the both datasets and unmatched records from left dataset.

Eg: *select*

*e.empno,*

*e.ename,*

*e.job,*

*e.sal,*

*e.deptno,*

*d.dname,*

*d.loc*

*from emp e left join dept d*

*on e.deptno=d.deptno;*

SQL> select a.id,a.name,b.loc from test1 a left join test3 b on a.id=b.id;

***Right Join:***

It returns matched records from both datasets and unmatched records from Right dataset.

select

e.empno,

e.ename,

e.job,

e.sal,

e.deptno,

d.dname,

d.loc

from emp e right join dept d

on e.deptno=d.deptno;

SQL> select a.id,a.name,b.loc from test1 a right join test3 b on a.id=b.id;

**Full outer Join:**

Returns Matched and unmatched records from both datasets

select

e.empno,

e.ename,

e.deptno,

d.loc

from emp e full join dept d

on e.deptno=d.deptno;

SQL> select a.id,a.name,b.loc from test1 a full join test3 b on a.id=b.id;

**Natural Join:**

This join also returns matching rows only. this is similar to inner join, but it gives high performance compared to inner join.

In this join not required to used join condition explicitly. But in this case resource tables must have a common column based on this common column oracle server internally establish joining condition .

*Syn:*

*Select \* from table1* ***natural join*** *table2;*

Eg: select \* from emp natural join dept;

**Cross join:**

Cross join will return Cartesian result.

Which mean left dataset completely multiply with right dataset.

When emp table – 14

Dept table – 4

14\* 4 => 56;

**SELF JOIN: (imp interview question)**

. joining a table itself is called self join

. Here joining conditional column must belongs to same datatype in same table.

. Generally if we want to compare 2 column values from *different tables* and also if we want to retrieve matching rows then we are only using equi-join/inner join,

where as we want to compare two different column values within a *same table* then we must use self join, but here these column must belongs to same datatype.

. before we are using self join we must create alias name in from clause. These alias name must be different names.

. these alias names internally behave like a extract table when query execution time.

Syn:   
*select aliasname1.col1,aliasname2.col1….. from table\_name aliasname1, table\_name aliasname2 where aliasname1.col1=aliasname2.col1;*

Q) WAQ to display ename and mgrname from emp table by using self join?

Answer: *SQL> select e1.ename "employees", e2.ename "manager"*

*from*

*emp e1,*

*emp e2*

*where e1.mgr=e2.empno;*

**Assignments:**

1) WAQ to display the employees who are getting more salary than their manager salary from emp table by using self join?

SQL> select

e1.ename "employees",

e2.ename "manager",

e1.sal emp\_Sal,

e2.sal mgr\_sal

from

emp e1,

emp e2

where e1.mgr=e2.empno

and e1.sal>e2.sal;

2) WAQ to display the employees who are joining before their manager from emp table by using self join?

select

e1.ename employees,

e1.hiredate emp\_hiredate,

e2.ename manager,

e2.hiredate mgr\_hiredate

from

emp e1,

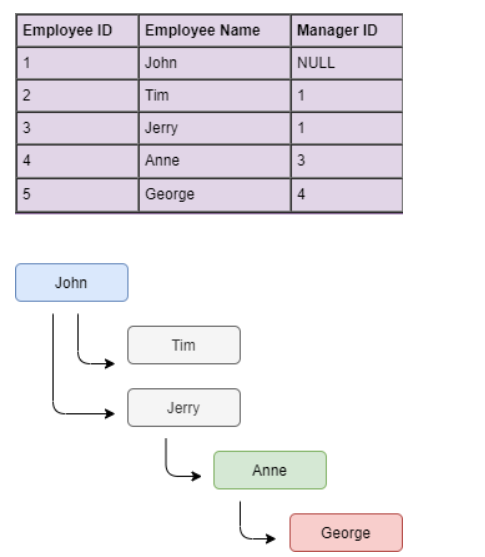
emp e2

where

e1.mgr=e2.empno

and e1.hiredate<e2.hiredate;

**self job** better example:



SELECT emp.employee\_id , emp.last\_name , emp.manager\_id ,

       mng.last\_name

FROM   employees emp ,employees mng

WHERE  emp.manager\_id = mng.employee\_id

**Set Operators: \*\*\*\*\***

Oracle provided following types of Set operators:

1) Union

2) Union All

3) Intersect

4) Minus

1) **Union** : it returns unique values from both datatsets and also sorting those values

Eg: select \* from emp where deptno=10

Union   
 select \* from emp where deptno=20;

SQL> select \* from test1

2 union

3 select \* from test3;

2) **Union all :** Unique + duplicate values

SQL> select \* from test1

Union all

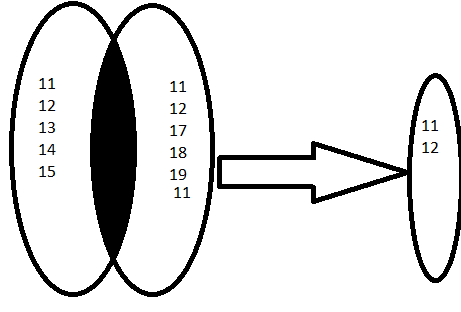
select \* from test3;

**3) Intersect:** it returns common values or marched records with distinct values (unique)

select \* from test1

intersect

select \* from test3;



**4. Minus:**

It returns values from first query those values are not in second query.

select \* from test1

minus

select \* from test3;

test1{1,2,3,4)  
test2{2,5,7}

tests Minus Test2 {1,3,4}

**conversion functions: (type conversion)**

converting one data type into another data type is called conversation.

Oracle having 2 types of conversions-

*a) implicit (or) automatic conversion*

*b) explicit conversion*

**a) Implicit/ automatic conversion:**

In oracle when an expression having string, string representing **pure number** automatically converted character type into number type.

Eg: select sal+’100’ from emp;

**Note**:

-in oracle whenever we are passing number into character function then oracle server automatically converts number type into character type.

Ex: select length(22222) from dual;

Op: 5

In oracle whenever we are passing date string into predefine date() functions then oracle server automatically converts date string into date type. But here passed parameter must be default date format.

Eg: select last\_day (’15-aug-05’) from dual;

***Explicit conversion:***

1) decode () \*\*\*

2) case statement \*\*\*

3) to\_number()

4) to\_char()

5) to\_date()

**1) decode():**

Decode() is a conversion function which is used to decoding the values. Decode conversion function Is same as **if-then-else** construct in pl-sql.

**Syn:** *decode(colum\_name, value1, stmts1, value2, stmt2, stmts);*

Eg:

select deptno, decode(deptno,10,'ten',20,'twenty',30,'thirty','others') dcode from emp;

**Assignment:**

Q) Update employee comm in emp table based on following conditions.

a) if job=’CLERK’ then update comm is 10% of sal

b) if job =’SALEMAN’ then update comm is 20% of sal else comm is 30% of sal

**SQL>** UPDATE EMP1 SET COMM=DECODE (JOB,'CLERK',SAL\*0.1,'SALESMAN',SAL\*0.2,SAL\*0.3);

(OR )

**SQL>** UPDATE EMP1 SET COMM=CASE WHEN JOB='CLERK' THEN SAL\*0.1

WHEN JOB='SALESMAN' THEN SAL\*0.2

ELSE SAL\*0.3 END;

**2) case statement:**

- case statement also used to decoding the values

- oracle 8.0 introduced case statement where as oracle 8i introduced case-conditional statement.

- case conditional statement are also called as searched case statement.

- case statement performance is very high compare to decode statement.

|  |
| --- |
| ***Syn:***  ***Case***  ***When column\_condition then statemen1***  ***When column\_condition then statement2***  ***--***  ***--***  ***Else statements end.*** |

Eg: select ename sal,

Case when sal<1000 then ‘low salary’

When sal between 1000 and 3000 then ‘medium salary’

When sal in (3500,3800,4000) then ‘special salary’

Else ‘others salary’ end salary\_Structure

from emp;

**Converting number into words:**

In oracle **to\_char** having **jsp** format which takes Julian date and then spell-out.

Eg: select to\_char(sysdate,’JSP’) from dual;

TWO MILLION FOUR HUNDRED FIFTY-NINE THOUSAND SIX HUNDRED NINE

SQL> select sysdate, to\_char(sysdate,'JSP') from dual;

SYSDATE TO\_CHAR(SYSDATE,'JSP')

--------- ------------------------------------------------------------------------------

29-JAN-22 TWO MILLION FOUR HUNDRED FIFTY-NINE THOUSAND SIX HUNDRED NINE

SQL> select sysdate, to\_char(sysdate,'J') from dual;

SYSDATE TO\_CHAR

--------- -------

29-JAN-22 2459609

**Julian date:**

In oracle Julian date is number of days since **Jan 1, 4712 BC**. Julian day always display as number , these Julian dates are used in oracle to convert a no. into works by using ***to\_date, to\_char*** functions.

If we want to convert into any no to word then first, we must convert given no’s into Julian date, if we want to convert any no. into Julian date then we must use format **‘J’** with to\_date function then only that Julian date converted into word by using to\_char JSP format.

Eg: SQL> select to\_char(to\_date(1234,'J'),'JSP') from dual;

**3) To\_number():**

Converting string representing a number value with format into number without format then only we are allowed to use to\_number() function.

Eg: select ‘$56.8’+3 from dual; -- it will through error

Eg: select to\_number(‘$56.8’)+3 from dual; -- error

When ever are using to\_number also we use a second parameter as same as first parameter format by using predefine format elements.

**Sol:** select to\_number (‘$56.8’,’$99.9’) +3 from dual;

**Null Value functions: \*\*\*\*\*\***

**1)** NVL(exp1,exp2) function:

It replaces null values into specified values.

Eg: SQL> *select sal, comm, nvl(comm,10) comm from emp;*

**2) coalesce():**

Oracle 9i introduced coalesce() function, it is an ANSI function, this function accepts n no.of expressions.

**Syntax:**

Coalese(exp1,exp2,exp3,……..)

This function returns first normal value from given expression.

SQL> select sal,comm, coalesce(comm,0) comm1 ,nvl(comm,0) comm2 from emp

CREATE TABLE CLSC\_TST(COL1 VARCHAR(4),COL2 VARCHAR(4),COL3 VARCHAR(4));

INSERT INTO CLSC\_TST VALUES ('1','','','');

INSERT INTO CLSC\_TST VALUES ('','1','','');

INSERT INTO CLSC\_TST VALUES ('','','1','');

INSERT INTO CLSC\_TST VALUES ('','','','1');

INSERT INTO CLSC\_TST VALUES ('','','','');

select coalesce(col1,col2,col3,col4) from CLSC\_TST;

**Difference between nvl(), coalesce(): \*\*\***

* **NVL** is an oracle function where as **coalese** is an ANSI function and also **coalese** performance is very high as compared to NVL function.
* NVL function internally uses implicit conversions i.e NVL function returns a value if the exp1,exp2 is not belong to same datatype also if exp2 automatically converted into exp1 where as in **coalese** function exp1,exp2 must belongs to same datatype.

**How to handle the null values in a table? *By using Nvl or coalesce***

**Constraints:**

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

- Generally, constraints are created on table columns:-

- Oracle server having following types of constrains

1) NOT NULL

2) UNIQUE

3) Primary Key

4) Foreign Key

5) Check

6) DEFAULT

Q) how many types of Constraints?

In all databases all above constraints are defined into two ways

*i) Column Level*

*ii) Table Level*

**i) Column Level:**

In this method we are defining constraints in individual columns i.e. whenever we are defining column then only, we are specifying constraints type.

Syntax:

Create table table\_name(col1 datatype(size) constraint\_type,

Col2 datatype(size) constraint\_type,……….);

**ii) Table Level:**

In this method we are defining constraints on group of columns i.e., first we are specifying all columns and last only we are specifying constraints type, along with group of columns.

Syntax:

Create table table\_name(col1 datatype(size), col1 datatype(size),……… constraint\_type(col1,col2));

**1) NOT NULL:**

-

*-NOT NULL doesn’t allow null values but it allows duplicate values*.

Column Level:

Eg: create table z1(sno number(10) not null, sname varchar(10));

**INSERT** **INTO** Z1 **VALUES**(1,'ABC');

1 row inserted

**INSERT** **INTO** Z1 **VALUES**(**NULL**,'ABC');

**SQL** Error [1400] [23000]: ORA-01400: cannot **insert** **NULL** **into** ("C##SCOTT"."Z1"."SNO")

**2) UNIQUE:**

- unique constraints doesn’t accept duplicate value but accept null values.

**Note:** Whenever we are creating unique constraints internally oracle server automatically creates b-tree indexes for those columns.

***Column Level:***

**INSERT** **INTO** Z2 **VALUES**(1,'ABC');

1 row inserted

**INSERT** **INTO** Z2 **VALUES**(1,'ABC');

**SQL** Error [1] [23000]: ORA-00001: **unique** **constraint** (C##SCOTT.SYS\_C007803) violated

***Table Level:***

**CREATE** **TABLE** Z3(SNO **NUMBER**(10),SNAME **VARCHAR**(10), LOC **VARCHAR**(10), **UNIQUE** (SNO,SNAME));

**INSERT** **INTO** Z3 **VALUES**(1,'ABC','HYD');

**INSERT** **INTO** Z3 **VALUES**(1,'ABC','HYD');

**SQL** Error [1] [23000]: ORA-00001: **unique** **constraint** (C##SCOTT.SYS\_C007804) violated

**INSERT** **INTO** Z3 **VALUES**(1,'BBC','HYD');

**3) Primary Key  
*primary key doesn’t accept duplicate values & null values***

And primary key constraint we can use only once in a table.  
Whenever we are creating primary key also oracle server automatically creates B Tree index on those columns.

Primary Key = NOT NULL + UNIQUE

Column Level:

**CREATE** **TABLE** Z4 (SNO **NUMBER**(10) **PRIMARY** **KEY**, SNAME **VARCHAR**(10))

**SQL**> **INSERT** **INTO** Z4 **VALUES** (1,'ABC');

1 **row** created.

**SQL**> **INSERT** **INTO** Z4 **VALUES** (1,'BBC');

ERROR **at** line 1:

ORA-00001: **unique** **constraint** (C##SCOTT.SYS\_C007805) violated

**SQL**> **SELECT** \* **FROM** Z4;

SNO SNAME

---------- ----------

1 ABC

**SQL**> **INSERT** **INTO** Z4 **VALUES** (**NULL**,'BBC');

ERROR **at** line 1:

ORA-01400: cannot **insert** **NULL** **into** ("C##SCOTT"."Z4"."SNO")

Table Level:  
**CREATE** **TABLE** Z5(SNO **NUMBER**(10), SNAME **VARCHAR**(10),LOC **VARCHAR**(10), **PRIMARY** **KEY**(SNO,SNAME));

**SQL**> **INSERT** **INTO** Z5 **VALUES**(1,'ABC','HYD');

Composite Primary key

1 **row** created.

**SQL**> **SELECT** \* **FROM** Z5;

SNO SNAME LOC

---------- ---------- ----------

1 ABC HYD

**SQL**> **INSERT** **INTO** Z5 **VALUES**(1,'ABC','HYD');

ERROR **at** line 1:

ORA-00001: **unique** **constraint** (C##SCOTT.SYS\_C007806) violated

**SQL**> **INSERT** **INTO** Z5 **VALUES**(**NULL**,'ABC','HYD');

ERROR **at** line 1:

ORA-01400: cannot **insert** **NULL** **into** ("C##SCOTT"."Z5"."SNO")

**4) foreign key:**

if you want to establish relationship on base tables then we are using referential integrity constraints foreign key.

One Table foreign key must belong to another table primary key and also these two columns must belong to same datatype.

Always foreign key values based on primary key values only.

Generally primary key doesn’t accept duplicate, null values whereas foreign key accepts duplicates & null values.

**Column level (references)**

Syntax:   
create table <table\_name> (col1 datatype(size) references <master\_table\_name>(primary key col\_name))

|  |
| --- |
| Sql> create table w4(sno number(10), references z4); |

**Table level:**

Syntax:

|  |
| --- |
| Create table table\_name (col1 datatype(size), col2 datatype(size),….  foreign key(col1,col2…..)  references master\_table\_name(primary\_key col)); |

Eg: create table z6(sno number(10), name varchar(10), foreign key(sno,sname) references z5)

Whenever we are establishing relation between table by using foreign key then oracle server automatically violates following 2 rules, these are,

1) deletion in master table

2) insertion in master table

**SQL**> **CREATE** **TABLE** DEPT\_DEMO(DNO **NUMBER**(10) **PRIMARY** **KEY** ,DNAME **VARCHAR**(10));

**Table** created.

**SQL**> **SELECT** \* **FROM** DEPT\_DEMO;

DNO DNAME

---------- ----------

10 ACC

20 SALES

30 MECH

**SQL**> **CREATE** **TABLE** EMP\_CHILD (EMPNO **NUMBER**,ENAME **VARCHAR**(10),DNO **NUMBER**(10) **REFERENCES** DEPT\_DEMO);

**Table** created.

**SQL**> **DESC** EMP\_CHILD

Name **Null**? **Type**

----------------------------------------------------------------------------------- -------- --------------------------------------------------------

EMPNO **NUMBER**

ENAME **VARCHAR2**(10)

DNO **NUMBER**(10)

**SQL**> **INSERT** **INTO** EMP\_CHILD(1000,'ABC',10);

\*

ERROR **at** line 1:

ORA-00928: missing **SELECT** keyword

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC',10);

1 **row** created.

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC',20);

1 **row** created.

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC','');

1 **row** created.

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC',10);

1 **row** created.

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC',30);

1 **row** created.

**SQL**> **INSERT** **INTO** EMP\_CHILD **VALUES**(1000,'ABC',40);

\*

ERROR **at** line 1:

ORA-02291: integrity **constraint** (C##SCOTT.SYS\_C007808) violated - parent **key** **not** **found**

CREATE TABLE EMP\_CHILD ( EMPNO NUMBER, ENAME VARCHAR(10),SAL NUMBER, DNO NUMBER REFERENCES DEPT\_DEMO(DNO));

CREATE TABLE EMP\_CHILD1 ( EMPNO NUMBER, ENAME VARCHAR(10),SAL NUMBER, DNO NUMBER,

FOREIGN KEY (DNO) REFERENCES DEPT\_DEMO(DNO));

**5) CHECK:**

Check constraints are used to define logical condition according to client business rules.

In Oracle Check constraints doesn’t work with sysdate function.

Sytax:

|  |
| --- |
| Create table table\_name(col1 datatype(size) check(logical\_condition),  Col2 datatype(size) ,,,,,); |

**SQL**> **CREATE** **TABLE** CHK(ENO **NUMBER**(10),ENAME **VARCHAR**(10),SAL **NUMBER**(10) **CHECK**(SAL<10000));

**Table** created.

**SQL**> **insert** **into** chk **values**(1000,'abc',5000);

1 **row** created.

**SQL**> **insert** **into** chk **values**(2000,'bbc',9000);

1 **row** created.

**SQL**> **insert** **into** chk **values**(3000,'cbc',10000);

**insert** **into** chk **values**(3000,'cbc',10000)

\*

ERROR **at** line 1:

ORA-02290: **check** **constraint** (C##SCOTT.SYS\_C007809) violated

SUB QUERIES:

. Query within another query is called subquery (or) nested query.

. Subqueries are used to retrieve data from single or multiple tables based on more than one step process.

. All Database systems having 2 types of subqueries, they are:

1) Non-Corelated sub queries /sub queries

2) Co-related sub queries

. In Non Corelated subqueries child queries are executed first then only parent queries executed, whereas co-related subqueries parent query will execute first then only child queries executed.

**1) Non – Corelated sub queries /sub queries:**

Non – corelated sub queries mainly consist of 2 parts,

a) child queries (or) inner query

b) parent query (or) outer query

**a) child queries:**

A query which provides values to another query is called subquery.

**b) parent queries:**

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

Non-Corelated sub-queries are 4 types, they are:

i) Single row sub query

ii) multiple row sub query

iii) multiple column sub query

iv) inline views

**Main definition:** in subquery first child query will execute and get the result, then that result will assign to the parent query, based on that result parent query will execute.

**Syntax:** select \* from <parant\_tabel> where <operators> (select \* from child\_qry);

**Q) WAQ to display employees who are getting more salary than average sal from emp table?**

select \* from emp where sal>(select avg(sal) from emp);

in the above queries, this is a single row sub query because child query returns single value. in single row sub query, we are using ***=, < , >, <=, >= , between*** operators.

**Execution process:**

Step1: sql> select avg(sal) from emp;  
 op: 2158  
step2: sql> select \* from emp where sal> 2158

**Q) WAQ to display the employees who are working in sales dept from emp table?**

SELECT \* FROM EMP WHERE DEPTNO=(select DEPTNO from dept where dname='SALES');

1> 90 % same columns  
2> 5% group fx

3> 3% expression

4> 2**% different columns.**

**Q) WAQ to display senior most employee details from emp table?**

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

**Assignments:**

**1) WAQ to display the employees who are getting same as ‘SMITH’ deptno from dept table?**

Answer)   
 *SELECT \* FROM EMP WHERE DEPTNO=(select DEPTNO from emp where ename='SMITH');*

**2) WAQ to display the employee who are getting more salary than the highest salary paid employee in 20th dept in emp table?**

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

**3) WAQ to display deptname, of the highest paid employee from emp, dept table?**

SELECT DNAME FROM DEPT WHERE

DEPTNO=(SELECT DEPTNO FROM EMP WHERE SAL=(SELECT MAX(SAL) FROM EMP));

**3) WAQ to display deptname AND ENAME , of the highest paid employee from emp, dept table?**

SELECT ENAME,DNAME,SAL FROM EMP E INNER JOIN DEPT D ON E.DEPTNO=D.DEPTNO

AND SAL=(SELECT MAX(SAL) FROM EMP);

**4) WAQ to display 2nd max salary from emp table?**

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

**5) WAQ to display 2nd highest salary details from emp table?**

select \* from emp where sal=(

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

**6) WAQ to display lowest avg salary job from emp table by using group by function?**

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

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

**7) WAQ to display job, avg salary which job avg salary more than clerk job avg salary?**

SELECT JOB, AVG(SAL) FROM EMP GROUP BY JOB

HAVING AVG(SAL)>(select avg(sal) from emp where job='CLERK')

**8) WAQ to display the employees who are working under ‘blake’ as manager by using empno, mgr column from emp table?**

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

**Multiple row subquery:**

When ever child query returns this type of subquery is called multiple row subquery.

In all databases we are using **IN, ALL ,ANY** operators in multiple row sub query.

Note: we can also used ‘ IN ‘ Operator in single row sub query.

**Q) WAQ to display the employee details who are getting max(sal) from each department from emp table?**

select \* from emp where sal=(select max(sal) from emp group by job);

ERROR at line 1:

ORA-01427: single-row subquery returns more than one row

This is the multiple row subquery because there child query returns multiple rows in subquery. In this scenario have to use IN operator

select \* from emp where sal IN (select max(sal) from emp group by job);

**Assignment:**

Q) WAQ to displays the employees who are working in ‘SALES’ or ‘RESEARCH’ department from dept ,emp table by using subquery.

SQL> select \* from emp where deptno in (select deptno from dept where dname IN ('SALES','RESEARCH'));

**Multiple Column Subquery:**

In all databases we can also compare multiple columns of child values with the multiple column values of parent query these type of subquery is called multiple column subquery. In this case must specify parent query where condition column within parenthesis()

**Syn:**

|  |
| --- |
| Select \* from table\_name where (col1,col2..) in (select col1,col12…from table\_nm); |

**Q) WAQ to display the employees who’s job ,mgr match with the job, mgr of the employee ‘scott’ from emp;**

select \* from emp where (job,mgr) in(select job,mgr from emp where ename='SCOTT');

**Assignments:**

**1) WAQ to display emp who are getting max sal from each dept from emp table using mutli column subquery?**

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

**2) WAQ to display ename, dname, sal of the employees whose sal ,match with the sal, comm of the employees working in the location ‘DALLAS’?**

SELECT ENAME,DNAME,SAL,LOC FROM EMP E, DEPT D

WHERE E.DEPTNO=D.DEPTNO

AND (SAL,NVL(COMM,0)) IN

(SELECT SAL,NVL(COMM,0) FROM EMP WHERE DEPTNO IN

(SELECT DEPTNO FROM DEPT WHERE LOC='DALLAS'));

**3) WAQ a multiple column subquery to display senior most employee in each job from emp table?**

select \* from emp where (job,hiredate) in(select job,min(hiredate) from emp group by job);

**TOP-N analysis:**

1) Inline Views \*\*\*\*

2) Rownum

3) Rowid

**1) Inline Views:**

Inline views are a special type of query having a sub query in **FROM** clause of the parent query.

Generally, we are not allowed to use order by clause in child query , to over come this problem, oracle introduced sub-queries. In from Clause in parent query.

these types of queries are also called as inline views

**syntax:**

select \* from (subquery);

generally, we are not allowed to use column alias name in where clause if we want to use column alias name in where clause then we must use inline views. Because in this case alias name behaves like a actual column name.

Eg:

Sql> Select ename,sal, sal\*12 annsal from emp where annsal>3000;

ERROR at line 1:

ORA-00933: SQL command not properly ended

Sql> Select \* from (select ename,sal,sal\*12 annsal from emp) where annsal>3000;

**2) Rownum:**

. Rownum is a pseudo column which behaves like a table column.

. rownum is used to restrict no.of rows in a table.

. These pseudo columns are belongs to all databases.

. it will generate temporary individual unique values for each select row

These are rownum, rowid……

Sql> select rownum,ename from emp;

Generally rownum having temporary values.

Ex: select rownum, ename from emp where deptno=10;

Q) WAQ to display first row from emp table by using row num?

Answer) select \* from emp where rownum=1

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

no rows selected

generally rownum doesn’t work with more than 1 positive integer values. It works with **< , <= operators**

SQL> select \* from emp where rownum<=2;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7839 KING PRESIDENT 17-NOV-81 6000 10

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

**Assignments:**

**1) WAQ to display first five highest salary employees from emp table by using rownum?**

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

**2) waq to display 5th highest salary emp from emp table by using rownum?**

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

MINUS

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

**3) waq to display 2nd row from emp table by using row num?**

select \* from emp where rownum<=2

minus

select \* from emp where rownum<=1;

**4) waq to display rows between 1 to 5 from emp table using rownum?**

SQL> select \* from emp where rownum between 1 and 5;

**5) waq to display rows between 3 to 7 from emp table using rownum?**

select \* from emp where rownum <=7

minus

select \* from emp where rownum <=3;

**6) waq to display last 2 rows from emp table by using rownum?**

select \* from emp

minus

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

**7) waq to display 2nd row from emp table by using rownum alias name?**

SQL> select \* from (select rownum rn, e.\* from emp e ) where RN=2;

**8) waq to display rows between 3 to 7 from emp table using rownum alias name?**

SQL> SELECT \* FROM (SELECT ROWNUM RN,E.\* FROM EMP E ) WHERE RN BETWEEN 3 AND 7;

**9) waq to display 2nd 3rd 7th 10th records from emp table using rownum alias name?**

SQL> SELECT \* FROM (SELECT ROWNUM RN,E.\* FROM EMP E ) where RN IN (2,3,7,10) ;

**10) waq to display 1st row ,last row from emp table by using rownum alias name?**

SELECT \* FROM (SELECT ROWNUM RN,E.\* FROM EMP E)

WHERE RN=1 OR RN=(SELECT COUNT(\*) FROM EMP)

**11) waq to display even no.of records from emp table by using rownum alias name?**

SQL> SELECT \* FROM (SELECT ROWNUM RN, E.\* FROM EMP E) WHERE MOD(RN,2)=0;

**12) waq to display 5th highest salary employee from emp table by using rownum alias name?**

SELECT \* FROM (

SELECT ROWNUM RN ,A.\* FROM

(SELECT \* FROM EMP ORDER BY SAL DESC ) A ) B WHERE RN=5;

ROWID:

----------

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

. generally rownum having tempary values where as rowid having fixed values.

. when ever we are inserting data into a table then oracle server automatically generates an unique identification number for identify the record uniquely,for each and every row this is called rowid.

ex: select rownum,rowid ,ename from emp;

. in oracle if you want to retrieve data very fastly then using rowid.

sql> select min(rowid) from emp;

sql> select max(rowid) from emp;

. In oracle by default rowid having in ascending order.

Q) WAQ to display first row from emp table by using rowid ?

select \* from emp where rowid=(select min(rowid) from emp);

Q) waq to display last row from emp table by using rowid?

select \* from emp where rowid=(select max(rowid) from emp);

Note: we can also use rowid in order by clause with in analytical function.

scenarios:

----------

1)waq to dislay 2nd row from emp table by using rowid?

2) waq to display last 2 records from employee table by using analytical function rowid.

3)waq to display 2nd row from each deptno by using anlytical function using rowid from emp table?

4) waq to display duplicate data from emp table?

5)waq to delete duplicate rows from the above table using row\_id?

or

delete duplicate row except 1 row from the above table.

**Analytical Functions: \*\*\*\*\***

Oracle having followed analytical functions:

1) ROW\_NUMBER()

2) RANK()

3) DENSE\_RANK()

syntax:

analytical\_function\_name() over([partition by column\_name] order by column\_name [asc/desc])

**=> Row\_number()** analytical function automatically assigns different rank\_number when values are same

**=> rank()** analytical function automatically assigns same rank numbers when values are same and also skip the next consecutive number(1,1,3,3,5)

**=> dense\_rank()** analytical function assign same number and doesn’t skip the sequence.

**Row\_number:**

Dno ename sal rn

20 a 100 1

20 b 100 2

20 c 100 3

30 d 200 4

**Rank():**

Dno ename sal rnk

20 a 100 1

20 b 100 1

20 c 100 1

30 d 200 4

30 e 100 4  
40 f 100 6

**Dense\_Rank():**

Dno ename sal dnsernk

20 a 100 1

20 b 100 1

20 c 100 1

30 d 200 2

40 e 300 3

40 e 600 3

50 e 600 4

**Q) WAQ to display the employee details highest salary to lowest salary and also automatically assign rank to each dept from table by using analytical function?**

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

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

**It will generate sequence number**

**Q) WAQ to display 2nd highest salary employee in each dept from emp table by using analytical function?**

SQL> select \* from (

select deptno,empno,ename,sal, row\_number() over(partition by deptno order by sal desc) rn from emp) where rn=2;

correct result is:

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

DEPTNO ENAME SAL RN

---------- ---------- ---------- ----------

10 CLARK 2450 2

20 JONES 2975 2

30 ALLEN 1600 2

**Q) WAQ to display 5th highest salary employee from emp table by using analytical function?**

SQL> select \* from (select deptno,ename,sal,dense\_rank() over(order by sal desc) rn from emp) where rn=5;

DEPTNO ENAME SAL RN

---------- ---------- ---------- ----------

10 CLARK 2450 5

**Q) WAQ to display nth highest salary employee from emp table by using analytical function?**

SQL> select \* from (select deptno,empno,ename,sal, row\_number() over( order by sal desc) rn from emp) where rn=&n;

**ROWID:**

. row\_id is a pseudo column; it behaves like a table column.

. generally, rownum having temporary values whereas rowid having fixed values.

. whenever we are inserting data into table then oracle server automatically generates a unique identification number for identify the records uniquely , this is called rowid.

Eg: select rownum, rowid, ename from emp where deptno=10;

.In Oracle if you want to retrieve data very fast then using rowid.

. We can also use min, Max function into rowid.

Eg: select min(rowid) from emp;

Select max(rowid) from emp;

. In Oracle by default rowid having ascending order.

Q) WAQ to display first row from emp table by using rowid?

Select \* from emp where rowid=(select min(rowid) from emp);

**Q) WAQ to display duplicate data from emp?**

SQL> delete from emp\_dup where (deptno,rowid) not in (select deptno,min(rowid) from emp\_dup group by deptno);

**Co-related subquery:**

. Generally, in no-correlated subquery, child query is executed first then parent query is executed.

. Whereas in co-related subquery first parent query executed first, then after child query is executed, that result will assign to parent query again.

. Generally, in non-corelated subquery child query is executed only once per parent query.

whereas co-related subquery child query is executed for each row per parent query table.

. In co-relate subquery we must create an alias name for parent query table and then pass the alias name in child query in “where” condition.

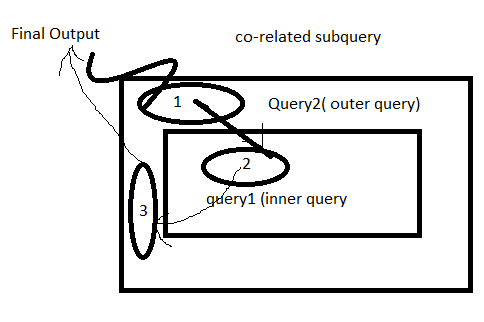
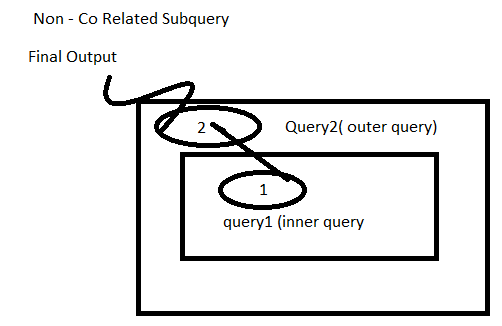
|  |
| --- |
| **Syntax:**  Select \* from table\_name aliasname  Where column\_name=(select \* from table\_name where columnname=aliasname.colname); |

. Generally, co-related subquery used in de-normalization process. in this process we are using co-related update.

If we want to modify 1 table column value based on another table then only, we are using co-related subquery updates.

|  |
| --- |
| **Syntax:**  Update table\_name1 aliasname set  Col\_name=(select column\_name from table\_name1 aliasname1  Where aliasname1.col\_name=aliasname2.column\_name); |

. whenever we are submitting co-related subquery into the database server, then database server gets candidate row from parent query table then control passed into child query where condition and then based on execution values, it compares value with parent query.



**Q) waq to display 1st highest salary employee from emp table by using co-related subquery?**

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

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7839 KING PRESIDENT 17-NOV-81 6000 10

Q) waq to display 2st highest salary employee from emp table by using co-related subquery?

outer query ==> inner query matching ==> outer query result.

select \* from emp e1 where 2=(select count(distinct sal) from emp e2 where e2.sal>=e1.sal);

**note**: whenever we are using co-related subquery in inner query should not be there any duplicates

**assignments:**

**1.WAQ to display nth highest salary employee from the emp table by using corelates subquery.**

SQL> select \* from emp e1 where &n=(select count(distinct sal) from emp e2 where e2.sal>=e1.sal);

**2. WAQ to display the employees who are getting more salary than the av(sal) of their job from emp table by using co-related subquery?**

select \* from emp e1 where sal>(select avg(sal) from emp where job=e1.job);

**Exists operator:**

. Exists operator always return Boolean value either **true** or **false**

. Exists operator performance is very high compared to **IN** operator

. Exists operator used in co-related subquery

. Exists operator is used in where condition only, when we are using **exists** operator then we are not allowed to use column name along with exists operator.

|  |
| --- |
| Syntax:  Select \* from table\_name aliasname where exists  (select \* from table\_name where tablename.col\_name=aliasname.col\_name); |

Eg: exists {1,2,3,4} = True

Exists { } = false

**Q) WAQ to display department from dept table those dept having employees in emp table by using corelated subquery exists operator.**

Select \* from dept d where exists (select \* from emp where deptno=d.deptno);

**Above same query is in non-corelated subquery?**

Select \* from dept where deptno in (select deptno from emp);

**Note:** Exists operator perform very fast compared to IN operator

**Assignments:**

Q) waq to display the employees who are getting same salary as ‘SCOTT’ from emp table by using corelated sub query exists operator.

Q) WAQ to display those departments from dept table doesnot have a employee in employee table by using co-related subquery exists operator?

Q) WAQ to display those from dept table doesn’t have any employee in emp table by using no-corelated sub query?

**ALL, ANY subquery special operators used In non-corelated subquery?**

IN -> It returns same values in the list

**< ALL** or **> ALL** 🡪 it satisfies all values in the list (AND)

>ANY or <ANY 🡪 it satisfies any values in the list. (OR)

**Q) WAQ to display the emp who are getting more salary than the all salary of the ‘CLERK’ from emp by using subquery special operator?**

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

Select \* from emp where **sal > any**(select sal from emp where job=’CLERK’)

*ALL = AND Operator*

*ANY = OR operator*

**CTE - (common table expression)**

**Syntax:**

A common table expression (CTE) is a named temporary result set that exists within the scope of a single statement and that can be referred to later within that statement, possibly multiple times

WITH

cte1 AS (SELECT a, b FROM table1),

cte2 AS (SELECT c, d FROM table2)

SELECT b, d FROM cte1 JOIN cte2

WHERE cte1.a = cte2.c;

Ex:

WITH E AS (SELECT \* FROM EMP ) ,

D AS (SELECT \* FROM DEPT)

SELECT

E.EMPNO,

E.ENAME,

D.DNAME,

D.LOC

FROM E INNER JOIN D ON E.DEPTNO=D.DEPTNO

WHERE E.DEPTNO=10;

**Recursive CTE:**

A recursive common table expression is one having a subquery that refers to its own name. For example:

WITH recrsive\_cte (n) AS

(

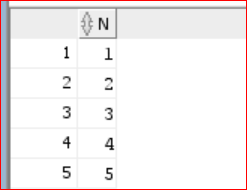
SELECT 10 from dual

UNION ALL

SELECT n + 1 FROM recrsive\_cte WHERE n < 15

)

SELECT \* FROM recrsive\_cte;



**Ref-** [**https://docs.oracle.com/cd/E17952\_01/mysql-8.0-en/with.html**](https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/with.html)

**Temp Table:**

A TEMPORARY table is visible only within the current session, and is dropped automatically when the session is closed. This means that two different sessions can use the same temporary table name without conflicting with each other or with an existing non- TEMPORARY table of the same name.

## **LEAD() function**

LEAD() is an [analytic function](https://www.oracletutorial.com/oracle-analytic-functions/) that allows you to access the following row from the current row without using a [self-join](https://www.oracletutorial.com/oracle-basics/oracle-self-join/).

LEAD(expression [, offset ] [, default ])

OVER (

[ query\_partition\_clause ]

order\_by\_clause

)

**Lag:**

LAG() is an [analytic function](https://www.oracletutorial.com/oracle-analytic-functions/) that allows you to access the row at a given offset prior to the current row without using a [self-join](https://www.oracletutorial.com/oracle-basics/oracle-self-join/)

**LEAST and LAG Examples:**

SQL) select segment,country,product,sale\_date,gross\_sales,discount,lead(gross\_sales) over(order by sale\_DAte,gross\_Sales) gross\_prev from Financial;

SQL) select segment,country,product,sale\_date,gross\_sales,discount,lag(gross\_sales) over(order by sale\_DAte,gross\_Sales) gross\_prev from Financial;

SQL) select ename,job,sal,lead(sal) over(order by hiredate,sal) sal\_prev from emp1;

SQL) select ename,job,sal,lag(sal) over(order by hiredate,sal) sal\_prev from emp1;

# **LISTAGG()**

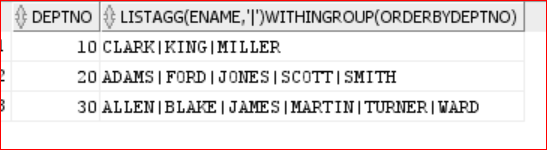
LISTAGG() function is an [aggregation function](https://www.oracletutorial.com/oracle-aggregate-functions/) that transforms data from multiple rows into a single list of values separated by a specified delimiter.

LISTAGG (

[ALL] column\_name [,delimiter]

) WITHIN GROUP( ORDER BY sort\_expressions);

select deptno,listagg(ename,'|') within group (order by deptno) from emp group by deptno



**VIEWS:**

. Views is a database object which is used to provides authority level of security.

. Generally, data security point of view database administrator creating views from the base table and those views given to the no.of users.

. **Generally, views don’t store data that’s why views are also called as virtual or window of the table.**

. Generally, views are created from base table, based on the base table views are categorized into 2 types:

*a) Simple view*

*b) Complex View*

select \* from emp where depnto=30

drop ?

**a) Simple View:**

Simple view is a view which is created from only one base table whereas complex table view is a view which is created from no.of base tables.

**Syntax:**

Create or replace view view\_name

As

Select statement from base\_table;

**Eg:**

SQL> create or replace view v1

as

select \* from emp1 where deptno=10;

select \* from v1;

**DML Operation on simple view?**

In oracle we can also perform DML operator through simple view to base table based on following restrictions.

1) If a simple view having group functions group by clause, rownum, distinct , set operators, join then we can’t perform DML operations through simple view to the base table.

2) we must include base table not null column into the view then only we are allow to perform insertion operation through simple view to the base table.

Ex1:

SQL> create or replace view v1

as

select \* from emp1 where deptno=10;

SQL> select \* from v1;

SQL> Insert into v1(empno,ename,deptno) values (1,’anil’,30);

Select \* from emp;

Ex:2

Create or replace view v2

As

Select sal,deptno from emp where deptno=10

Select \* from v2;

**COMPLEX VIEWS:**

Complex view is a view which is created multiple base tables.

create or replace view v5

as

select ename,sal,dname,loc from emp,dept

where emp.deptno=dept.deptno;

SQL> select \* from emp;

Generally, we can’t perform DML operation through complex view to base table.

SQL> update v5 set ename='abc' where ename='SMITH';

**Materialized view:**

* M.View used in data warehousing application.
* Generally views doesn’t stored data whereas materialized views store data
* Generally materialized views are used to improved performance of the joined or aggregable queries
* M.Views are store data same like base table but when are refreshing M.views synchronize the data based on base table.

Syntax:

Create materialized view view\_name

As

Select \* from tables

Ex:

Create materialized view mv1

As

Select \* from emp1;

Sql> select \* from mv1;

**Materialized view refesh:**

execute dbms\_mview.refresh('EMP\_mz1');

**difference between views and m.views**

**views**

1) View doesn't' store the data

2) security purpose

3) when we are dropping base table then can’t be accessible

4) Through the view we can perform DML Operation

**materialized views.**

1) materialized view stores data. like physical table

2) improve performance purpose

3) when we are dropping base table also materialized view

4) we can't' perform DML operation

**SEQUENCE:**

* Sequence is a database object, which is used to generates sequence number automatically.
* Generally, sequences are used to generates primary key values automatically.
* once sequence are created, then no.of users simultaneously access that sequence.
* Sequence is an independent database object.

***Syntax:***

*Create sequence sequence\_name*

*Start with n*

*Increment by n*

*Minvalue n*

*Maxvalue n*

*Cycle/nocycle*

*Cache/nocache;*

* If we want to generate sequence value or access sequence value then we are using following 2 pseudo columns.

1) currval

2) nextval

Syntax: sequence\_name.currval

Syntax: sequence\_name.nexval

**Note**: pseudo columns are used in select,insert,update,delete statement in oracle. If we want to generates sequence values by using select statements then we must use dual table.

Syntax: select sequence\_name.curval from dual;

Syntax: Select sequence\_name.nextval from dual;

Eg:

Create sequence s2

Start with 5

Increment by 2

Maxvalue 20;

Sql> select s1.currval from dual;

Error –

Sql> select s1.nextval from dual;

Sql> select s1.nextval from dual;

Sql> select s1.nextval from dual;

Ex: create table test(sno number(10) default s1.nextval primary key, name varchar2(10));

Sql> Insert into test(name) values (‘abc’);

# **Display Sequence of Numbers in SQL Using LEVEL**

<https://docs.oracle.com/cd/B12037_01/server.101/b10759/queries003.htm>

**INDEX:**

An index is a database object, using this we can fetch the data/retrieve the data very fast. It will improve the query performance. An index creates an entry for each value that appears in the indexed columns. By default, Oracle creates B-tree indexes.

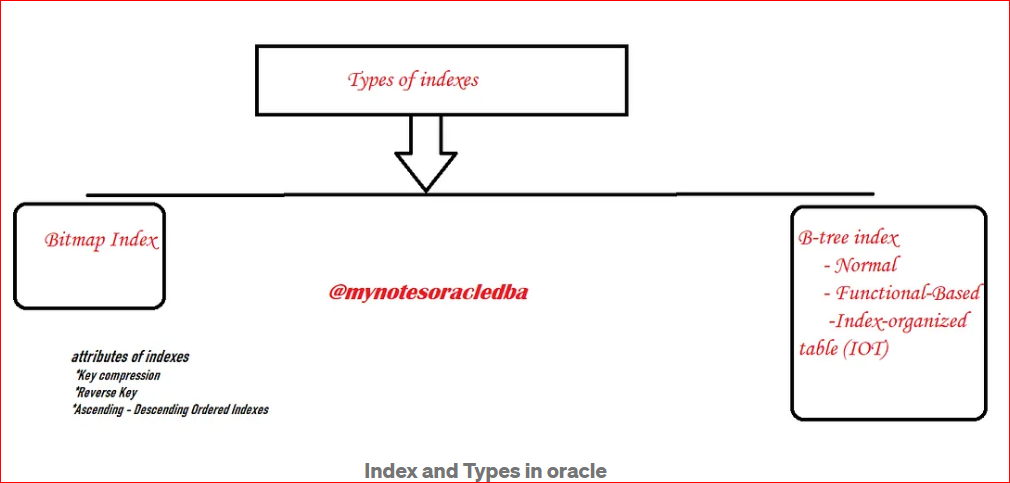
**Notes:**

. When we are requesting data by using **where clause** or **order by clause** then only **DB server searching for Indexes.**

. *In oracle whenever* ***where*** *clause having* ***not equal to (or) Is null or is not null operators t****hen oracle server doesn’t search for indexes*. If those columns having already indexes also.

Index uses following methods to locate the desired information

* Table Scan/full table scan
* Index Scan



**Oracle having different types of indexes:**

1) **B Tree Indexes (binary tree)**

* Index Organized index
* Reverse key index
* Descending index
* B-Tree Cluster index (Instead of pointing to a row, the key points to the block that contains rows related to the cluster key)

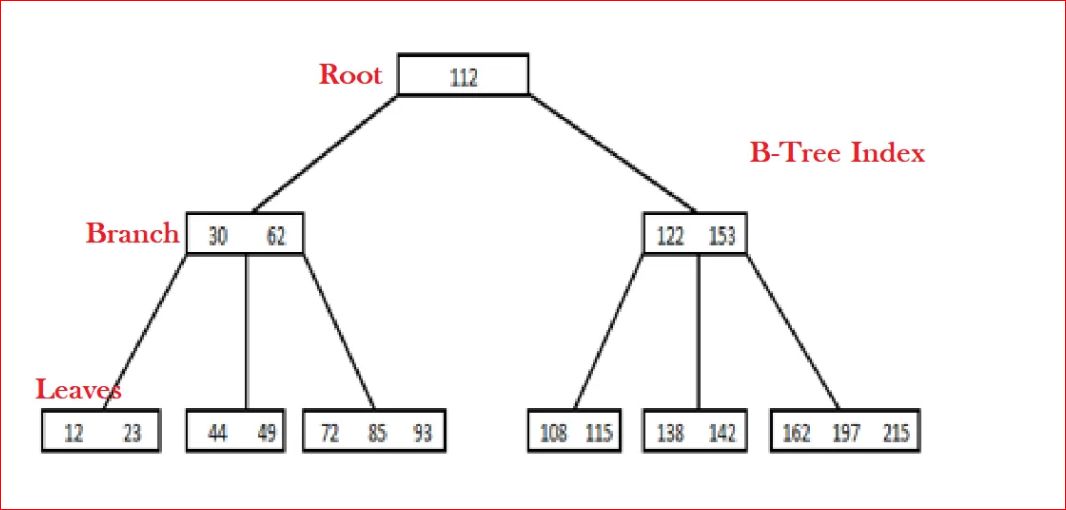
2) **Bitmap Indexes**

* Function Based Index
* Application Based Index
* Unique index

**A) B Tree Indexes**:

default indexes are Btree indexes (Primary key = b-tree index)

* It is a default index in the oracle database when we write a CREATE INDEX, it creates the B-Tree index.
* It is commonly used for OLTP type databases.
* It is recommended when the selectivity of the column is high in the table
* Let’s assume we have employee details it contains a huge volume of rows in case we are looking at email details B-tree is good for email columns
* This index has a root, branch and leaves.
* In this example, we have each one branch from the right and left sides.
* In the first branch has data less than 100, the second has the data between 108 and 215
* Here this interval is stored in the root and based on the related interval, the search is directed to the relevant branch. And these branches also have the same mechanism



**Syntax:** Create Index index\_name on table\_name(column name);

**Unique index:**

**Syntax:**

CREATE [UNIQUE] INDEX index\_name

ON table\_name (column1, column2, ... column\_n)

[ COMPUTE STATISTICS ];

**UNIQUE**

It indicates that the combination of values in the indexed columns must be unique.

Eg: create index in1 on emp1(ename);

Sql> Desc user\_indexes

Sql> select index\_name, index\_type from user\_indexs where table\_name=’emp’;

**##INDEXES:**

CREATE TABLE SALES

(

ID INT,

ORDER\_ID INT,

PROD\_CODE VARCHAR2(30),

AMOUNT INT,

DATE\_OF\_SALE\_DATE DATE

);

INSERT INTO SALES

SELECT

ROWNUM,FLOOR(DBMS\_RANDOM.VALUE(90,9000)),

DBMS\_RANDOM.STRING('U',3),FLOOR(DBMS\_RANDOM.VALUE(90,000)),

TRUNC(SYSDATE)-FLOOR(DBMS\_RANDOM.VALUE(10,9000))

FROM DUAL

CONNECT BY LEVEL<=50000;

Before index explain plan:

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=1;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

After index explain plan:

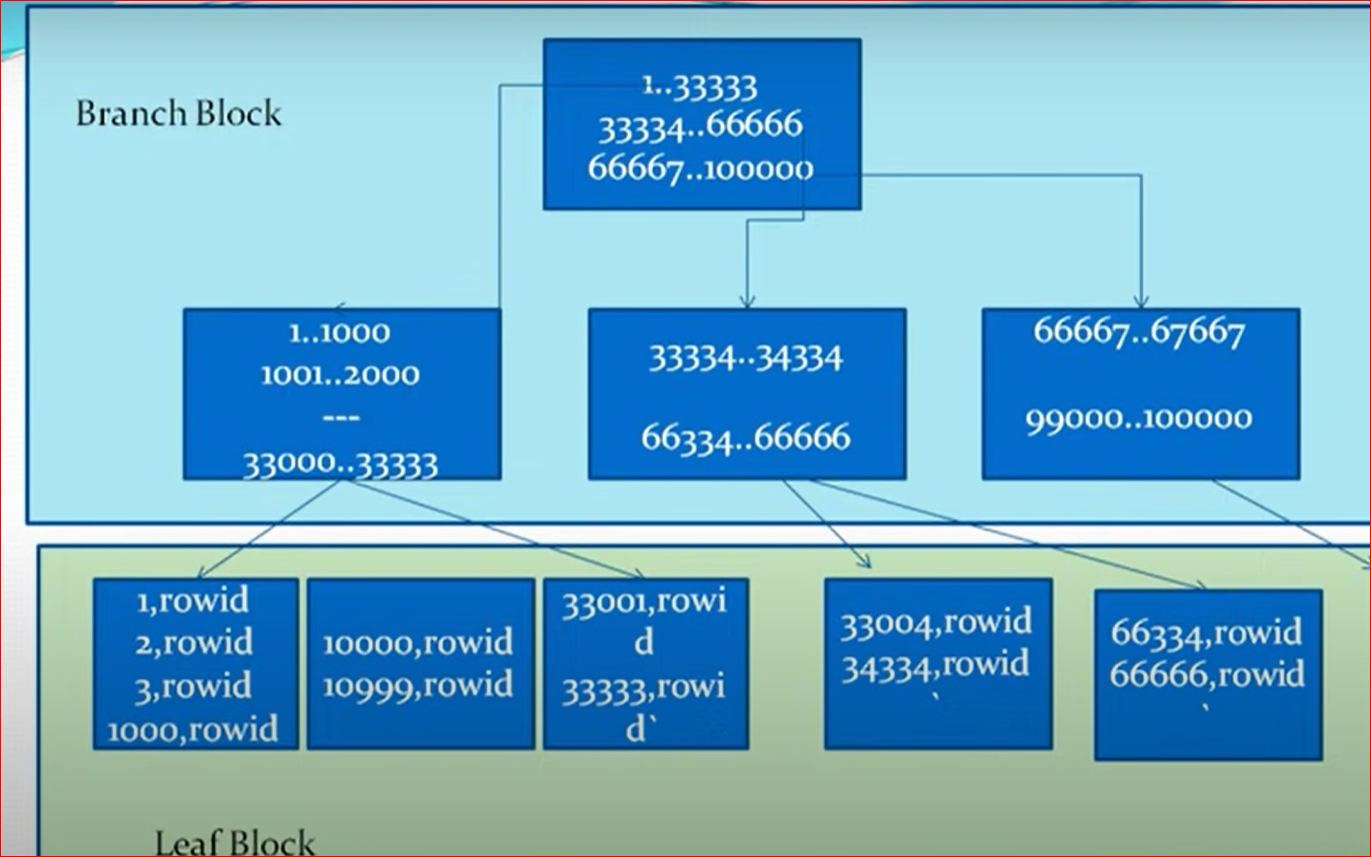
CREATE UNIQUE INDEX IDX\_SALES\_ID ON SALES(ID);

ANALYZE INDEX IDX\_SALES\_ID VALIDATE STRUCTURE;

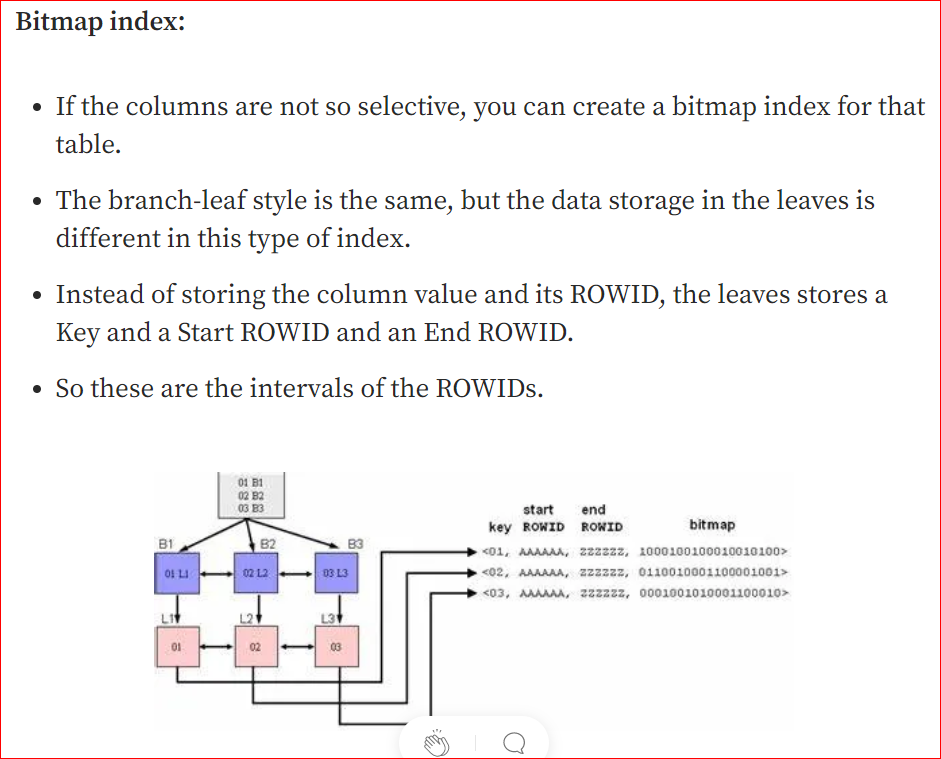
EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=1;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

SELECT \* FROM USER\_SEGMENTS WHERE segment\_name='IDX\_SALES\_ID';

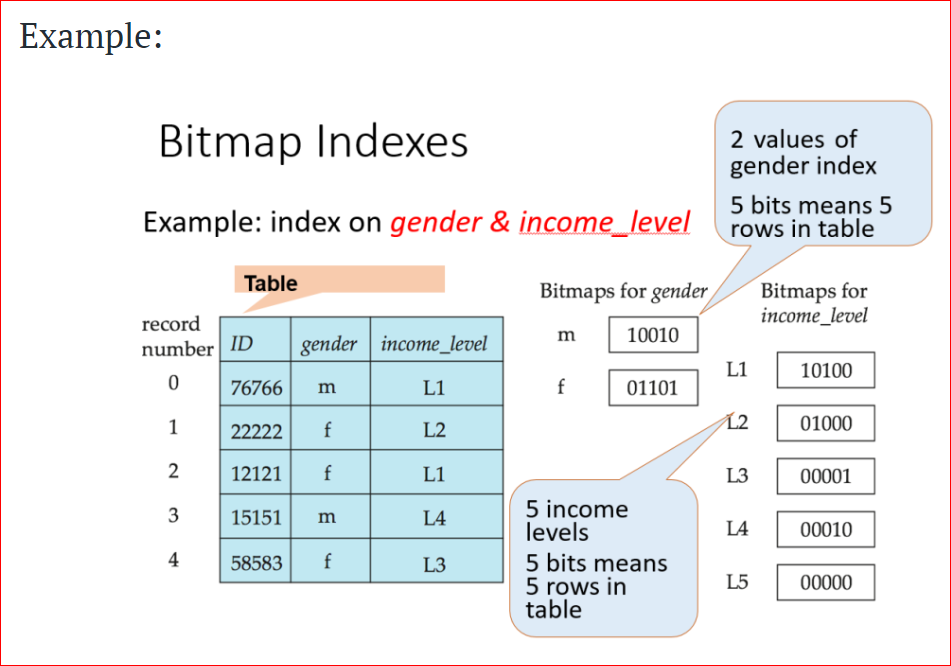
**Default B-Tree index look like**

**Bitmap Index:**



### bitmap index?

A bitmap index is a special type of index designed for efficient query processing on multiple keys. It is a binary valued two-dimensional array created with an indexed column for every record in the table. Bitmap indexes use bit arrays (commonly called bitmaps) and answer queries by performing bitwise logical operations on these bitmaps. As the number of distinct values increases, the size of the bitmap increases exponentially.  The bit in a row of bitmap is “1” if the record has the value v for the indexed attribute, or “0” otherwise.



## Collect Statistics on an Index

If you forgot to collect statistics on the index when you first created it or you want to update the statistics, you can always use the ALTER INDEX command to collect statistics at a later date.

### **Syntax**

The syntax for collecting statistics on an index in Oracle/PLSQL is:

ALTER INDEX index\_name

REBUILD COMPUTE STATISTICS;

**index\_name**

The index in which to collect statistics.

### **Example**

Let's look at an example of how to collect statistics for an index in Oracle/PLSQL.

For example:

ALTER INDEX supplier\_idx

REBUILD COMPUTE STATISTICS;

In this example, we're collecting statistics for the index called supplier\_idx.

## Drop an Index

### **Syntax**

The syntax for dropping an index in Oracle/PLSQL is:

DROP INDEX index\_name;

**index\_name**

The name of the index to drop.

### **Example**

Let's look at an example of how to drop an index in Oracle/PLSQL.

For example:

DROP INDEX supplier\_idx;

In this example, we're dropping an index called supplier\_idx.

**Cluster Index Ref:**

[**https://docs.oracle.com/database/121/CNCPT/tablecls.htm#CNCPT-GUID-CC31365B-83B0-4E09-A047-BF1B79AC887A**](https://docs.oracle.com/database/121/CNCPT/tablecls.htm#CNCPT-GUID-CC31365B-83B0-4E09-A047-BF1B79AC887A)

A **table cluster** is a group of tables that share common columns and store related data in the same blocks.

When tables are clustered, a single data block can contain rows from multiple tables. For example, a block can store rows from both the employees and departments tables rather than from only a single table.

CREATE CLUSTER employees\_departments\_cluster

(department\_id NUMBER(4))

SIZE 512;

CREATE INDEX idx\_emp\_dept\_cluster

ON CLUSTER employees\_departments\_cluster;

**What is a Clustered Index?**

In SQL Server, the primary key constraint automatically builds a clustered index on a column. According to the protocol, there will be only one clustered index per table. A clustered index is used to define the order or to sort the table or arrange the data by alphabetical order just like a dictionary.

**What is a Non Clustered Index?**

A non-clustered index collects the data at one place and records at another place. The index carries pointers to the position of that data. According to the protocol, a single table can include many non-clustered indexes.

**Difference between Clustered and Non-clustered index :** \*\*\*

|  |  |  |
| --- | --- | --- |
| **S.No** | **Clustered** | **Non-clustered** |
| 1 | A clustered index is used to define the order or to sort the table or arrange the data by alphabetical order just like a dictionary. | A non-clustered index collects the data at one place and records at another place. |
| 2 | It is faster than a non-clustered index. | It is slower than the clustered index. |
| 3 | It demands less memory to execute the operation. | It demands more memory to execute the operations. |
| 4 | It permits you to save data sheets in the leaf nodes of the index. | It never saves data sheets in the leaf nodes of the index. |
| 5 | A single table can consist of a sole cluster index. | It can consist of multiple non-clustered indexes. |
| 6 | It has the natural ability to store data on the disk. | It does not have the natural strength to store data on the disk |

**Ref:**

<https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/>

<https://www.geeksforgeeks.org/difference-between-clustered-and-non-clustered-index/>

can try to get more details about cluster and non-cluster index

**Partitions table**

Partition tables are used to improve the query performance of the application in backup and recovery process.(DBA)

Partition tables are created by database administrator in **every large database**.

Partition tables are used in data warehousing applications; partitions table are created based on **key column**; this column is also called partition key.

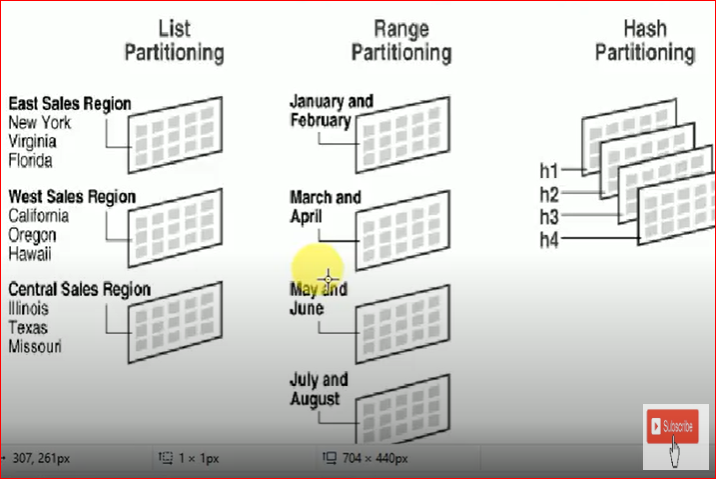
In oracle if we want to view particular partitions then we are use following sysntax

Select \* from table\_name partition (partion\_name1,partition\_name2….)

1) Range Partitions

2) list partitions

3) hash partitions.



**1) Range partitions:**

Create table table\_name (col1 datatype (size),…)

Partition by range (key\_column)

(partition partition\_name1 values less than(value),…

partition partition\_name2 values less than(value),…

);

SQL> create table test\_partition(sno number(10), name varchar2(10), sal number(10))

Partition by range (sal)

(Partition p1 values less than(100), partition p2 values less than (200), partition p3 values less than (3000));

**2.list partitions:**

Oracle 9i introduced list partition, using list partition we can also create partition based on character datatype column.

In this method partitions are created based on list of values.

Syntax:

Create table (col\_name1 datatype(size)…..)

Partition by list (key\_col)

(partition partition\_name1 values (val1,val2..),……)

SQL> create table test\_list1(sno number(10), name varchar2(10))

Partition by list (name)

(Partition p1 values ('ap','ch'), partition p2 values ('us','uk','canada'), partition p3 values (default));

Insert into table\_name values (…….)

SQL> select \* from test\_list partition (p3);

**HASH Partitions:**

Hash Partitions are automatically created by oracle server based on hash algorithm, in this case we are specifying no.of partitions explicitly

Syntax:

Create table table\_name (col1 datatype(size),….)

Partition by hash(key\_col)

Partition anynumber;

create table test\_hash(sno number(10),sal number(10))

partition by hash(sal)

partitions 5;

SQL> insert into test\_hash values(3,3000);

SQL> desc user\_tab\_partitions;

SQL> select partition\_name from user\_tab\_partitions where table\_name='TEST\_HASH';

SQL> SELECT \* FROM TEST\_HASH PARTITION(SYS\_P1046);

**NORMALIZATION:**

. Normalization is specific process which is used to decomposing a table into no.of tables. This process automatically reduces duplicate/redundant data and also automatically avoids insertion, updating & deletion problems.

. In design phase of SDLC database designers design LOGICAL MODEL of the data base in this logical model only database designers was normalization process by using normal forms

. E F Codd introduced the normal forms.

**Normal Form:**

1) 1st Normal Form

2) 2nd Normal From

3) 3rd Normal Form

4) BCNF Normal Form (Boyee Code)

5) 4th Normal Form

6) 5th Normal Form

**1ST Normal form:**

If a table is in 1st normal form in that table in each column should be **atomic** and also identifying a key record uniquely used key.

Item\_table(not in 1st NF)

|  |  |  |  |
| --- | --- | --- | --- |
| Item\_Name | color | price | tax |
| marker | red,gree | 20 | 0.2 |
| pen | blue,black | 30 | 0.3 |

Candidate key

|  |  |  |  |
| --- | --- | --- | --- |
| Item\_Name | color | price | Tax |
| marker | red | 20 | 0.2 |
| marker | green | 20 | 0.2 |
| pen | blue | 30 | 0.3 |
| pen | black | 30 | 0.3 |

**2nd Normal Form:**

. If table in 1st and also all non-key attributes are fully , functionally dependent on total candidate key.

. Always 1st NF deals with atomicity whereas 2nd NF deals with relationship between key and no-key attributes.

.If a table in 1st NF and also the table having any partially no-key attributes then the table not in 2nd NF.

|  |  |  |  |
| --- | --- | --- | --- |
| candidate key | |  |  |
| Item\_Name | color | price | tax |
| marker | red | 20 | 0.2 |
| marker | green | 20 | 0.2 |
| pen | blue | 30 | 0.3 |
| pen | black | 30 | 0.3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| item\_color |  |  |  |  |  |
| |  | | --- | | FRK | |  |  | Pk | Item\_price | |
| Item\_Name | color |  | Item\_Name | price | tax |
| marker | red |  | Marker | 40 | 0.2 |
| marker | green |  | Pen | 30 | 0.2 |
| pen | blue |  |  |  |  |
| pen | black |  |  |  |  |

Select

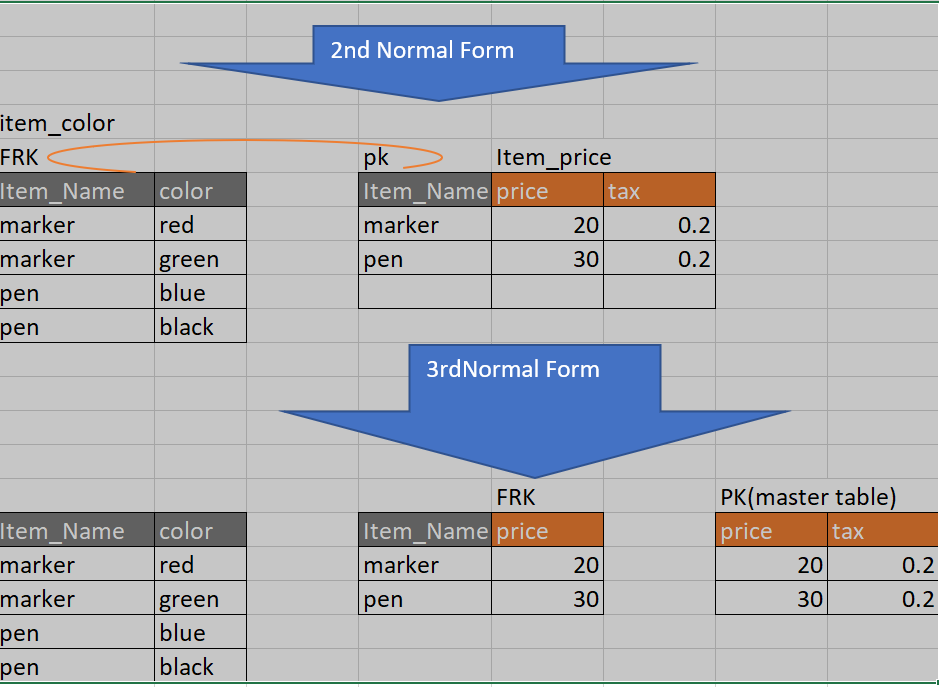
a.item\_name,a.color,b.price,b.tax

From item\_color a inner join item\_price b on a.item\_name=b.item\_name

**3rd Normal Form:**

If a table is in 2nd NF also all non-key attributes are only dependent on total candidate key.

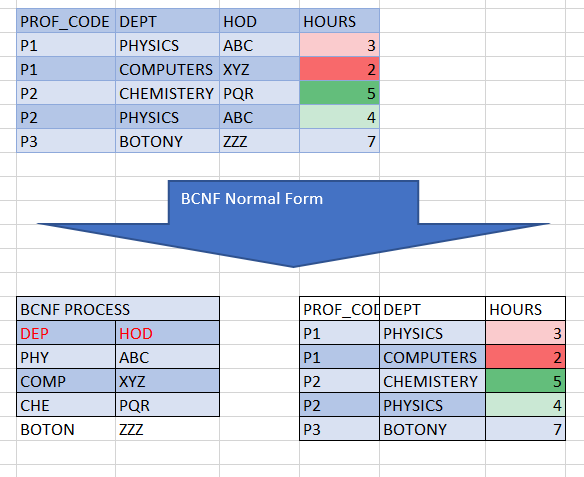
If a table is in 2nd NF also any non-key attributes which depends on another non-key attributes, then that table is in 3rd NF



**BCNF Normal Form:**

If a table is in BCNF in that table every determinant is a candidate key.

Whenever table have multiple composite candidate keys and also those candidate keys are overlapped, and also one candidate key , non -key attributes which depends on another candidate key non key attributes then only we are using BCNF process.



**What is the De-normalization?**

Traditionally data is stored in normalized databases, in which multiple separate tables are maintained in a relational database to minimize the redundant data. Therefore, whenever we have to access data from multiple tables, we need to perform complex and costly join operations on the required tables.

**Demoralization** does not mean that the database is never normalized. It is just a method of optimizing databases that is implemented after normalization has been achieved.

Demoralisation will maintain the redundancy

Normalization is reducing the redundancy

Ref: <https://www.scaler.com/topics/denormalization-in-dbms/>

Advantages:

**Enhanced Query Performance**

**More convenient to manage database**

**Facilitate and accelerate reporting**

## **Difference between Normalization and Denormalization**

The following table highlights the important differences between Normalization and Denormalization −

|  |  |  |
| --- | --- | --- |
| **Factor** | **Normalization** | **Denormalization** |
| Implementation | Normalization is used to remove redundant data from the database and to store non-redundant and consistent data into it. | Denormalization is the process of adding some redundant data to a database that has been normalized, so as to improve the read performance (execution time) of the database |
| Focus | Normalization mainly focuses on clearing the database from unused data and to reduce the data redundancy and inconsistency. | The real goal of denormalization is to achieve the faster execution of the queries by introducing redundancy. |
| Number of Tables | During Normalization, data is reduced, so there will be a decrease in the number of tables. | During Denormalization, data is integrated into the same database and hence there will be an increase in the number of tables. |
| Memory consumption | Normalization uses optimized memory and hence faster in performance. | Denormalization introduces some sort of wastage of memory. |
| Data integrity | Normalization maintains data integrity, i.e., any addition or deletion of data from the table will not create any mismatch in the relationship of the tables. | Denormalization does not maintain any data integrity. |
| Where to use | Normalization is generally used where a number of insert/update/delete operations are performed and joins of those tables are not expensive. | Denormalization is used where joins are expensive and frequent queries are executed on the tables. |

**Pivot & Unpivot Concept:**

<https://oracle-base.com/articles/11g/pivot-and-unpivot-operators-11gr1>

<https://doyensys.com/blogs/converting-rows-to-columns-using-oracle-pivot-clause/>

<https://www.databasestar.com/oracle-sql-pivot/>

<http://www.dba-oracle.com/t_pivot_examples.htm>

<https://www.oracletutorial.com/oracle-basics/oracle-pivot/>

<https://www.techonthenet.com/oracle/pivot.php>

**RegEx Commands Concept:**

<https://www.complexsql.com/oracle-regular-expression-examples-regexp_like-regexp_count-regexp_instr-regexp_substr-regexp_replace/>

**Performance tuning Techniques:**

* [12 Query optimization tips for better performance](https://blog.devart.com/how-to-optimize-sql-query.html#query-optimization-tips-for-better-performance)
  + [Tip 1: Add missing indexes](https://blog.devart.com/how-to-optimize-sql-query.html#missing-indexes)
  + [Tip 2: Check for unused indexes](https://blog.devart.com/how-to-optimize-sql-query.html#Non-used-indexes)
  + [Tip 3: Avoid using multiple OR in the FILTER predicate](https://blog.devart.com/how-to-optimize-sql-query.html#or-in-join-predicate)
  + [Tip 4: Use wildcards at the end of a phrase only](https://blog.devart.com/how-to-optimize-sql-query.html#use-wildcards)
  + [Tip 5: Avoid too many JOINs](https://blog.devart.com/how-to-optimize-sql-query.html#high-table-count)
  + [Tip 6: Avoid using SELECT DISTINCT](https://blog.devart.com/how-to-optimize-sql-query.html#avoid-using-select-distinct)
  + [Tip 7: Use SELECT fields instead of SELECT \*](https://blog.devart.com/how-to-optimize-sql-query.html#use-select-fields-instead-of-select-all)
  + [Tip 9: Run the query during off-peak hours](https://blog.devart.com/how-to-optimize-sql-query.html#run-query-during-offpeak-hours)
  + [Tip 11: Minimize large write operations](https://blog.devart.com/how-to-optimize-sql-query.html#minimize-large-write-operations)
  + [Tip 12: Create joins with INNER JOIN (not WHERE)](https://blog.devart.com/how-to-optimize-sql-query.html#create-joins-with-inner-join)
  + Tip 13: Use **explain plan** and identify the bottle neck

-------------------- THE END -------------------------

**Indexes practice and how to see this in IO results in Explain plan:**

select \* from sales;

drop table sales;

create table sales (

id int,

order\_id int,

product\_code varchar2(3),

amount int,

order\_of\_sale date

);

INSERT INTO SALES

select rownum, FLOOR(DBMS\_RANDOM.VALUE(90,9000)),

DBMS\_RANDOM.STRING('U',3),FLOOR(DBMS\_RANDOM.VALUE(90,9000)),

TRUNC(SYSDATE)-FLOOR(DBMS\_RANDOM.VALUE(10,900))

FROM DUAL

CONNECT BY LEVEL <=100000;

SELECT \* FROM SALES;

--SELECT SYSDATE,TRUNC(SYSDATE)-FLOOR(DBMS\_RANDOM.VALUE(10,900)),FLOOR(DBMS\_RANDOM.VALUE(10,900)) FROM DUAL;

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=1;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

CREATE UNIQUE INDEX IDX\_SALES\_ID ON SALES(ID);

ANALYZE INDEX IDX\_SALES\_ID VALIDATE STRUCTURE;

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=1;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

-- INDEXES ARE Physically stored in the database as an objects

select \* from user\_segments where segment\_name='IDX\_SALES\_ID';

--basiscally btree index will sort the data.

--(idexes contain brank block(root node & block node) and leafe block (it contain rows wwith rowid)

-- shall we create indexs on every column?

.) let's not do that

.) index may increase select peformance in certain cases but they hamper insert update and deletes

.) create index when you are continuesly using the select statements

--USABLE AND NON-USABLE INDEX IN ORACLE

NOTE: ORACLE consider which are usable in table, they are not consider which are not in non-usable.

ALTER INDEX IDX\_SALES\_ID unusable;

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=100;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY); -- here indexs are unusable and will turn into full scan.

select \* from user\_segments where segment\_name='IDX\_SALES\_ID'; -- this obeject drop from the database when we did it as unsual;

Re-build the index:

Alter index idx\_sales\_id REBUILD;

select \* from user\_segments where segment\_name='IDX\_SALES\_ID';

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=100;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

--VISIBALE AND INVISIABLE INDEX:

ALTER INDEX idx\_sales\_id INVISIBLE;

select \* from user\_segments where segment\_name='IDX\_SALES\_ID';

EXPLAIN PLAN FOR SELECT \* FROM SALES WHERE ID=100;

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

NOTE: unusable WILL DROP THE INDEX,BUT IVISIBLE JUST HIDE THE INDEX IT WONT' DLETE FROM THE DATABASE.