- Some of most commonly used scalar functions are:-
  - LCase()
  - UCase()
  - Now()
  - Length()
  - Sysdate()
  - Floor()
  - CEIL()
  - ROUND()
  - FORMAT()

View :- Views are tables whose content are taken or derived from other tables. View themself donot contain the data. View just act as window through which certain content can be view. User can also manopulate the content of original table through these view but in reality the view does not exist in database as a stored set of value.

Syntax :-

Create OR Replace view view_name As
<select statement>

emp_ID	First Name	email	Phone No	Dept_id

Employee

• View is not a table, it is just a window (temporary window) through which we can see the value.

Page No.
Date / 20

(i) Create view emp_view As select Emp_ID, First_Name, Dept_id from employee;

(ii) When dep id is visible. →
create view emp_view As select emp_ID, First_Name, Dept_ID from employee where dept_id = 20;

→ There are following reasons to use the views (exists)
* views restict extract to database.
* user can only view the some fields of a database.
* Crucial data in database table can be controled by using the view.
* View allow same data to be seen by different user in different way

→ Types Of views :-
(1) Horizontal Oriented view
(2) vertical view
(3) Row / Column subset view
(4) grouped view
(5) joint view

* Horizental view :- In the view horizental subset of source table is taken in view.

For example :-
Create view emp_view As select * from employee where emp_id In(101, 102, 103);

Page No.
Date / 20

* Vertical view :- In the view vertical subset of source table is taken to view.

For example :-

create view emp_view As select emp_id, First Name, email from employee;

* Row/column subset view :- It is combination of horizontal and vertical view.

For example :-

create view emp_view As select emp_ID, first_name, email from employee where emp_ID In(101,103,104,

* group view :- It is created based on group by query. The group rows of data and produce one row in the result corresponding to each group. So the row in the view do not have one to one correspondence with the rows of table for this reason the group view can not be updated.

For example :-

create view view_group by (dept, No of Emp)
As select department, count (employee id)
from employee_Manages
Group by department;

* Joined Views :- It is created by specifying more than one table in the query or in simple words, it contains

SELECT with joining of tables. The rows in this view don't have a one to one correspondence with the rows of the source table. For this reason, joined view cannot be updated.

For example :-

```
Create view view_cust_join as
select a.Cust_id, b.cust_first_Name, b.cust_last_Name, Amount_in_dollars
from Customer_loan a, customer_details b
where a.cust_id = b.cust_id;
```

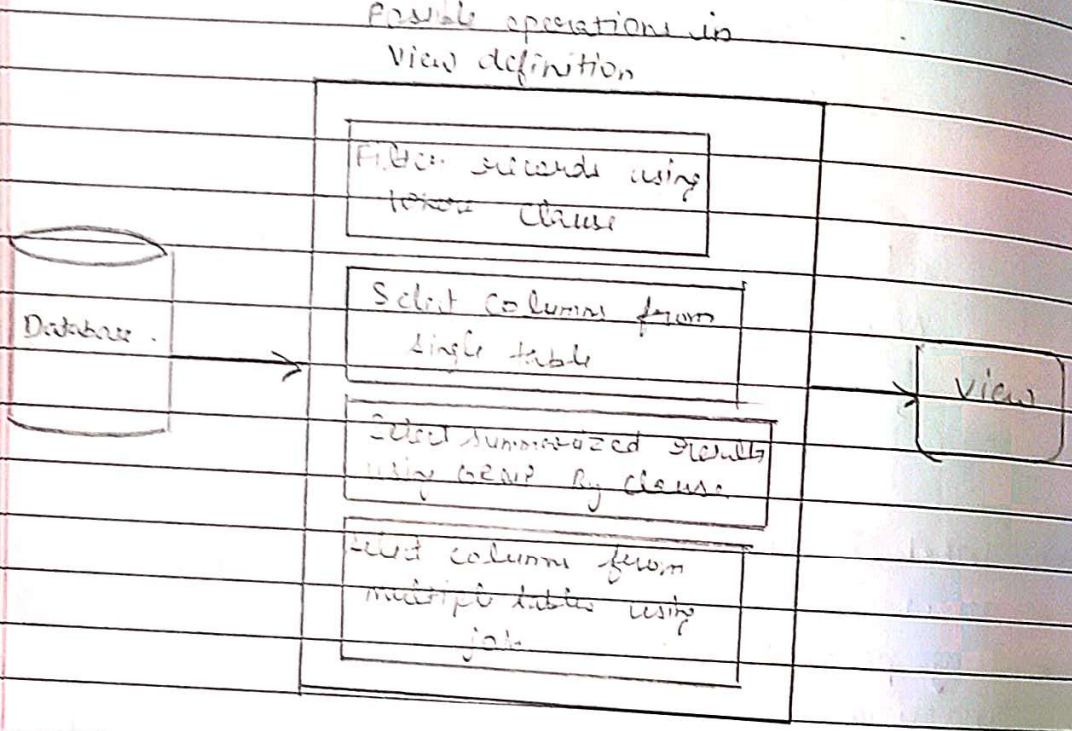
→ What is a view?

* Views are a special version of table in SQL.
* They provide a virtual table environment for various complex operations. You can select data from multiple tables, or you can select specific data based on certain criteria in views.
* It does not hold the actual data; it hold only definition of the view in the data dictionary.
* The view is a query stored in the data dictionary, on which the user can query just like they do on table.
* It does not use the physical memory, only query is stored in the data dictionary.
* It is computed dynamically, whenever the user perform any query on it.
* Changes made at any point in view are reflected in the actual base table.

→ The view has primarily two purposes:-

① Simplify the complex SQL queries.

② Provide restriction to users from accessing sensitive data.



→ Types of Views :-

- Simple View :- A view based on only a single table, which doesn't contain GROUP BY clause and any functions.
- Complex View :- A view based on multiple tables, which contain GROUP BY clause and functions.
- Inline View :- A view based on a subquery in FROM clause, that subquery creates a temporary

table and simplifies the complex query.

- Materialized View :- A view that stores the definition as well as data. It creates replicas of data by storing it physically.

* Inline view :-
Inline view is a select statement in the ~~from~~ From clause of another select statement to create temporary table that could be referenced by the select statement. Inline views are ~~writing~~ initialised for writing complex sql query without join or subquery Operation. This is called a temporary table and known as inline view.

The temporary table is called inline view

- Syntax :-

```
select column names.....
From (Sub queries)
where condition ;
```

A subquery is also a select query

* Materialized view : It replicate the retrieve data physically. This replicated data can be reused without exicuting the view again, this type of view is also known as snap short. It reduce the processing time

to segregate the whole data. It help secure user to replicate data locally and improve the very performence.

For example we have a have employee table with following attribute - emp_id, emp_name, dept_id and salary

Emp_id	Emp_name	dept_id	salary
1001	David	2	40000
1002	Mark	1	5000
1003	John	1	6000
1004	Steef	2	7000
105		2	8000

Employee

- Syntax :-

Create Materialized view emp_view As select emp_id, name, dept_id from employee where dept_id=2.

O/P →

Emp_id	Emp_name	dept_ID
1001	David	2
104	steef	2
105		2

~~Index~~ :-

⇒ Data Manipulation in view :- Once a view is created it can be manipulated just like any other table and user can perform insert, update and delete query on the view. for eg → create a customer view on a customer details table and also

Page No.
Date / 20

perform the insert, delete and update queries
Consider customer detail table with following attributes - cust_id, name, phone, email & address

cust_id	name	phone	email	Address

Customer details

(1) create view cust_view As
select * from customer_details
where cust_id In (101, 103, 105);

(2) Insert into cust_view values (107, 'ABC', 985...3xx, 'abc@gmail.com', 'Delh');

(3) Delete from cust_view where cust_id = 107;

delete → record from table

(4) Update cust_view
set name = 'Ajay'
where cust_ID = 107;

Apart all various conditions rules # for updating a view

Updating a view :- depending on the implementation being used view may or may not update table. A view is update table if -

(1) distinct keyword is not specified in the query used to create the view.

horizontal, vertical, row view → view update

(2) The From clause specified only one source table.

(3.) The select list does not contain expressions.
(4) The where clause does not include a subquery
(5) The query does not include a group by or having clause

Droping a view :- * Views are droped in single way as tables are droped.

- Syntax :-

DROP view view_name;

* If some other views depends upon this view then following statement can be used -

Drop view view_name Cascade ;

Disadvantage And Advantage of view :-

Index :- * Index is a way to store and search a record in a table.
* Indexes are used to improve the speed with which the records can be located and retrieve from the table.
* The oracle retrieves the row in a table in one of the two method or ways.
  - By ROWID
  - By full table scan

* The address of an index is called row id which are internally generated and maintain

Page No.
Date / 20

0 or 1

in binary values. With row id, oracle can search for data on disc quickly.

* The row id format is used by oracle is BBBBBBB RRRR FFFF.
  - BBBBBBB: represent block no. on which record is stored
  - RRRR: represent record no. in each data block
  - FFFF: represent unique no. given to each data file

Uses of indexes :- * It can be used to ensure uniqueness on a data base.

* Indexes can also ~~used~~ boost performance on reaching for records in a table.

Types of indexes :- Duplicate index and unique indexes.

* Duplicate Indexes :- * It allows duplicate value for index columns.

Duplicate indexes can be simple duplicate index or composite duplicate index.

1. Simple duplicate index :- Index is create on a single column but it can have duplicate value.

° Syntax :-

~~simple duplicate index~~
create index index_name.
on table_Name (column_Name);

° for eg :-

create index emp_index
On emp(ename);

2. Composite duplicate index :- Index created on a multiple column and can have duplicate values.

- Syntax :-

```
create index Index_Name
On table_Name (column_Name1, column_Name2
```

- eg :-

```
create index emp_index
On emp (ename, sal);
```

* Unique index :- It does not allow duplicate values for index column.
It can be further classified into
- simple unique index
- composite unique index

1. Simple Unique Index :- It is the index create on a single column but it cannot have duplicate values.

- Syntax :-

```
Create Unique Index Index_Name.
On table_Name (column_Name);
```

- eg :-

```
create Unique Index emp_index.
On emp (emp_id);
```

→ if we use ename (in which duplicate value take place) so it show arrow

2. Composite Unique Index :- It is a index create on multiple column with no dupl value.

Page No.
Date / 20

Syntax :-

create Unique index index_Name
On table_name (column_Name1, column_Name2, ...);

eg :-

create Unique index emp_index
On emp(emp_id, email);

user can create multiple databases

Dropping an existing index :-

When an Index is no longer needed in database the developer can remove it with the use of drop command.

Syntax :- Drop Index Index Name;

for eg :-

Drop Index emp_index;

Index guidlines :- There are some guidlines for creating efficient index -

* Don't use indexes on small table, a full table scan would be fine.

* Create a primary key for all the tables an index will be created by default

* Index the column that are involved in multitable join operation.

* Index column that are used frequently in where clause.

* Index column that are involved a order by, group by, union and distinct operation.

* Column that are frequently to update are bad for indexes.

some time will take place either we use (create) index or not in case of small table

* Column that have long character strings are bad for indexes.

Oracle index scan :-

1. Oracle index are implimented by using the data structure

(1) B-Tree

(2) Bitmap Index

* B-Tree :- B-Tree stand for balanced tree, most common type of database index.
* A B-Tree index is an order list of value divided into ranges by associating key with a row or range of row.
* B-Tree provide excellent retrieval performance for wide range of query including - exact match and range searches

for example :- Consider an employee table with following attributes - emp id, ename, salary and dept id.

emp-id	ename	salary	deptid

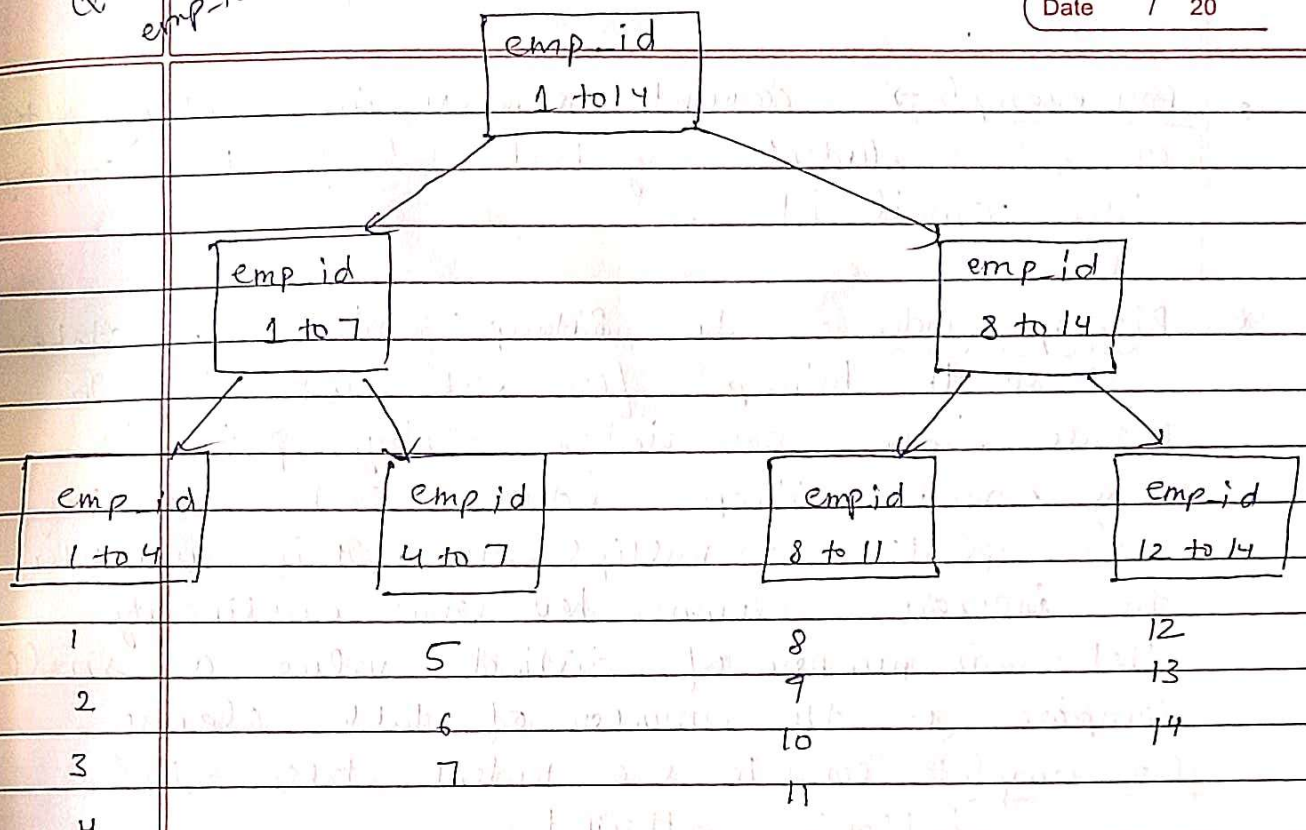
Step-1 :- create index emp_index On emp_table(emp_id);

Step-2 :- select rowid, e * from emp_table e;

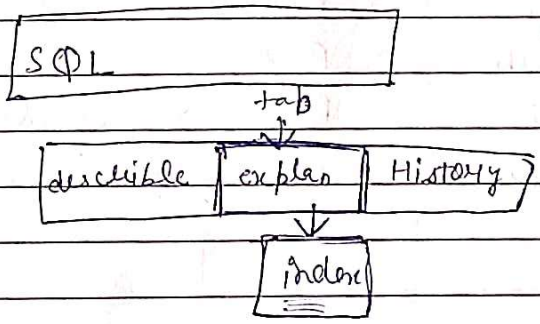
(ii) let we have a table having 1 to 14 emp id

→ B-Tree
it is used to search the element or information in the table in minimum time.

Q- search emp_id = 5



→ How to find weither index is used by query ?



in this way we can check where the index exist.

another way to check index created on the table

select * from user_indexes from table_name = 'emp_table';

→ When to create a B-tree index ?

B-Tree indexes are suitable for column containing heigh number of unique values.

For example → account number in banking system, student registration number, employee id, email id.

* Bitmap index :- In Bitmap index, the database stored the bitmap for each index key. In B-tree index one index entry point to single row in bitmap index each index key store pointer to multiple row. It is used when the indexed column have low cardinality that is number of distinct value are small compare to the number of table clause.

For example :- Consider a student table with following attributes.

S_id	S_name	Result	
101	Ajay	P	
102	Akash	F	
103	Mohit	P	← Students
104	Raman	P	
105	Ajay	F	

→ How Bitmap index data is stored ?

Step 1 :- create bitmap index std_index on students (Result);

Step 2 :- select * from student where result In ('P', 'F');

This query will search into the bitmap index to find the records according to the condition specified in where clause. It may no need

to scan the source table.

→ When to use bitmap index?
* table column with low cardinality value.
* Very less DML activities preferably read only table.
* multiple low cardinality column.

Sequences :- A sequence is a database object used to generate sequences numbers for rows in the table. It can be used for producing unique primary key value. A sequence is created using the CREATE sequence command.

mostly used for serial no

o Syntax :-

```
Create Sequence Sequence_Name
[INCREMENT BY <Integer Value>]
[START WITH <Integer Value>]
[MINVALUE <Integer value>]
[MAXVALUE <Integer Value>]
[CYCLE | NO CYCLE];
```

all bracket parts are optional

o For example: Create a sequence employee_numbers startwith with value 100 and increment by 2

```
Create Sequence Empnumbers
Increment By 2
Start with 100;
```

if we not use increment statement then by default 1

* Increment by clause is optional and default value is 1.

Page No.
Date / 20

* A Pseudo column NEXTVAL is used to extract the next number in line from a specified sequence. Another Pseudo column CURRVAL is used to extract the current sequence number.

* For eg :- To view the next number in sequence use following query.

Select empnumber.NEXTVAL from dual;

O/P → 102

* In same way CURRVAL column is used to know the last value returned by NEXTVAL column.

* The following statement will insert 102 into the employee table. (emp table)

For example :-

Insert into emp (empno, ename, job, do, sal
values (empnumber.CURRVAL, 'Mohit', 'Manager',
'12-Aug-23', 50000);