

**SYIT Sem III**  
**Database Management System**  
**Practical #6**

# Revision of Practical 5

Joins

Subqueries and DML Commands

# Contents of the Practical 6:

Views with its  
Operations

Types of  
Views

SET operators

# What is View?

- View is the simply subset of table which are stored logically in a database means a view is a virtual table in the database whose contents are defined by a query.
- To the database user, the view appears just like a real table, with a set of named columns and rows of data.
- SQL creates the illusion of the view by giving the view a name like a table name and storing the definition of the view in the database.
- Views are used for security purpose in databases, views restricts the user from viewing certain column and rows means by using view we can apply the restriction on accessing the particular rows and columns for specific user.
- Views display only those data which are mentioned in the query, so it shows only data which is returned by the query that is defined at the time of creation of the View.

# Views Contd..

- Views that used only for looking at table data are called as **Read-Only views**.
- Views that are used to look at the table data as well as Insert, Update and Delete table data is called an **Updateable View**.

➤ **SYNTAX** – CREATE VIEW ViewName AS SELECT field 1, field 2.... From existing\_table\_name;

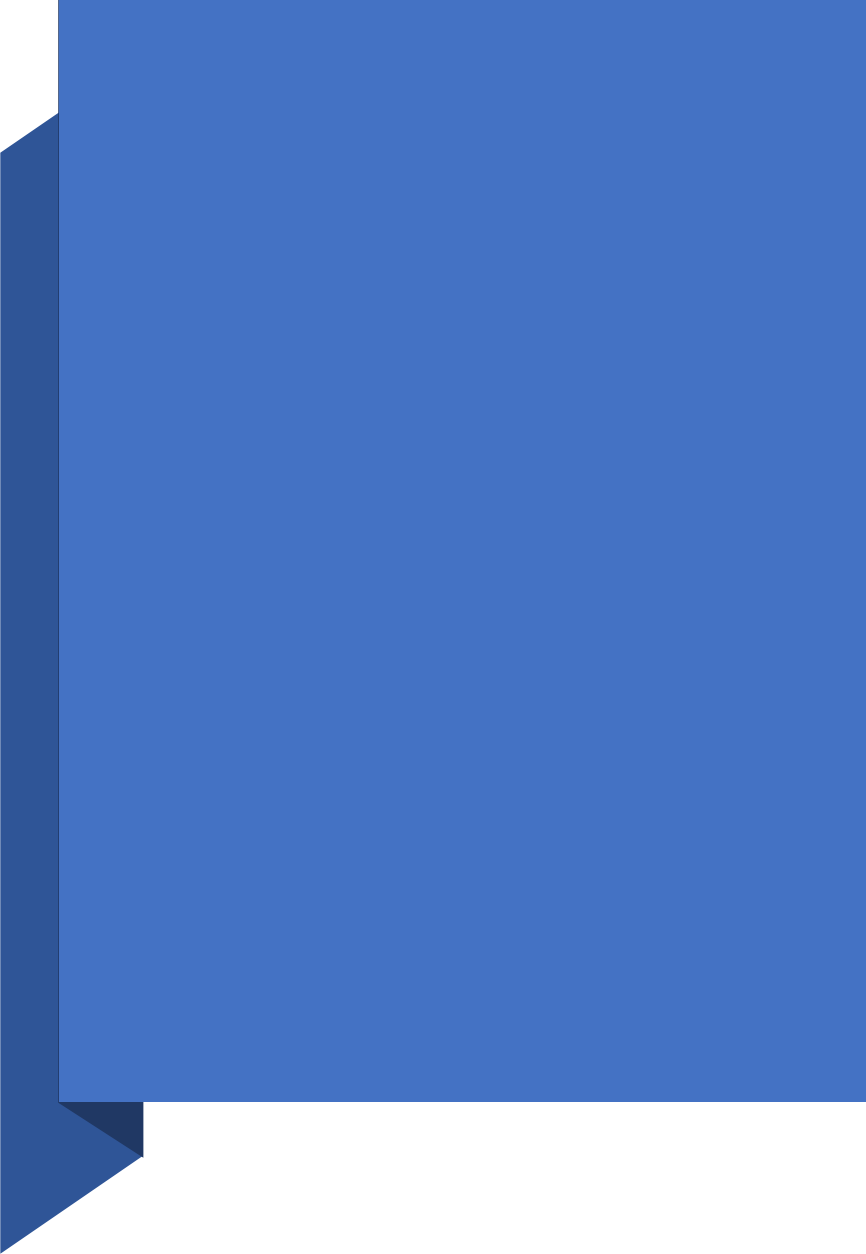
consider table emp

- Create following table emp and insert the given four records to it.

select \* from emp;

- o/p:

eid	ename	esal	did
1	neha	20000	1
2	neeta	25000	2
3	priya	30000	2
4	raj	35000	4

- 
- **EXAMPLE** – CREATE VIEW emp\_view AS SELECT \* FROM emp where did=2;
  - **VIEW CALL** – SELECT \* FROM ViewName;
  - Eg: select \* from emp\_view;

## ➤ create a view to display employee id and name

```
create view empdet as(select eid, ename from emp);
```

```
select * from empdet;
```

eid	ename
1	neha
2	neeta
3	priya
4	Raj

## ➤ Inserting a row into View

```
insert into empdet values(5,'ankita');
```

```
select * from empdet; //from view
```

output:

eid	ename
1	neha
2	neeta
3	priya
4	raj
5	ankita

# Output of main base table emp

```
select * from emp; //from table
```

eid	ename	esal	did
1	neha	20000	1
2	neeta	25000	2
3	priya	30000	2
4	raj	35000	4
5	ankita	NULL	NULL



# Deleting a row from view

- delete from empdet where eid=5;
- select \* from empdet; //from view
- o/p:

eid	ename
-----	-------

1	neha
---	------

2	neeta
---	-------

3	priya
---	-------

4	raj
---	-----

select \* from emp; //from table

o/p:

eid	ename	esal	did
-----	-------	------	-----

1	neha	20000	1
---	------	-------	---

2	neeta	25000	2
---	-------	-------	---

3	priya	30000	2
---	-------	-------	---

4	raj	35000	4
---	-----	-------	---

# Types of Views

- Consider faculty table

- select \* from faculty;

- o/p:

• Fid	Fname	Branch	Sal
• 1	A	IT	10000
• 2	B	IT	20000
• 3	C	CS	50000
• 4	D	CS	40000
• 5	E	CS	80000

# 1. Horizontal view

- **A horizontal view cuts the source tables horizontally to create the view. It creates a horizontal view of the source tables.**
- It is used when different names have to be displayed in a view.
- create view IT\_faculty as(select \* from faculty where branch='IT');
- Select \* from IT\_faculty;
- o/p:

Fid	Fname	Branch	Sal
1	A	IT	10000
2	B	IT	20000

- create view CS\_faculty as(select \* from faculty where branch='CS');
- Select \* from CS\_faculty;
- o/p:

Fid	FName	Branch	Sal
3	C	CS	50000
4	D	CS	40000
5	E	CS	80000

## 2. Vertical view

- **A vertical view cuts the source tables vertically to create the view.**
- create view info as (select Fid, FName from faculty);
- Select \* from info;
- o/p:

Fid	FName
1	A
2	B
3	C
4	D
5	E

### 3. Row/Column Subset View

- A row/column subset view slices the table horizontally as well as vertically to create a view. Only some columns and some rows of the source table are in this view .
- EXAMPLE –
- create view rc1 as (select Fid, FName from faculty where branch='CS');
- Select \* from rc1;
- o/p:

Fid	FName
3	C
4	D
5	E

## 4. Grouped View

- **A query specified in a view with a group by clause is called as grouped view.**
- Create following table faculty
- Select \* from faculty;

Fid	Fname	Branch	Sal
1	A	IT	10000
2	B	IT	20000
3	C	CS	50000
4	D	CS	40000
5	E	CS	80000

## 4.Grouped view contd....

- create view faculty\_group as (select branch,max(sal) as maximum from faculty group by branch);
- select \* from faculty\_group;

- Output:

branch	maximum
IT	20000
CS	80000

## 5. Joined View

- **Joined views are created by specifying multi table query that draws data from multiple tables.**
- EXAMPLE – Consider table faculty.
- Select \* from faculty;
- | Fid | Fname | Branch | Sal   |
|-----|-------|--------|-------|
| 1   | A     | IT     | 10000 |
| 2   | B     | IT     | 20000 |
| 3   | C     | CS     | 50000 |
| 4   | D     | CS     | 40000 |
| 5   | E     | CS     | 80000 |
- create table dept(deptno number, branch varchar(20));
- Select \* from dept;
- | deptno | branch |
|--------|--------|
| 10     | IT     |
| 20     | CS     |



## 5. Joined View contd..

Syntax:

- CREATE VIEW JoinView AS Select D.branch, F.Fname From dept D, faculty F WHERE F.branch=D.branch;
- Select \* from JoinView;
- o/p:
- Branch      fname
- IT            A
- IT            B
- CS            C
- CS            D
- CS            E

# Read Only view

- **We can create a view with read-only option to restrict access to the view.**
- Syntax to create a view with Read-Only Access  
CREATE or REPLACE view view\_name AS  
SELECT column\_name(s)  
FROM table\_name  
WHERE condition with read only
- The above syntax will create view for read-only purpose, we cannot insert or Update or delete data into read-only view. It will throw an error.
- create view facultydetails as(select fid,fname from faculty ) with read only;
- select \* from facultydetails;
- insert into facultydetails values(6, 'F');//can't execute
- update facultydetails set fname='G' where fid=5; //can't execute
- delete from facultydetails where fid=2; // //can't execute

# UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

- The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
- The SELECT statement should not have the DISTINCT keyword.
- The View should have all NOT NULL values.
- The view should not be created using nested queries or complex queries.
- The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.
- The **UPDATE** query is used to update the record of view.
- SYNTAX: UPDATE view\_name Set field\_name=new\_value  
Where condition;
- Example:
- Update IT\_Faculty set sal=30000 where fid=1;

## Dropping a view

### **Syntax:**

```
drop view view_name
```

### **Example:**

```
drop view IT_faculty;  
drop view CS_faculty;
```

# Table vs View

Table	View
Hold actual data.	Contains columns which reference the base table for the data.
It occupies memory space.	It does not occupy memory space except a data dictionary entry.
A table can contain its own data.	Views contain data from multiple tables.
A record of a table can be inserted, updated and deleted unless it violates any integrity constraints.	There are limitations and restrictions to update and delete from views.
You can add integrity constraints to a table.	You cannot add integrity constraints to a view.

A large orange circle is positioned on the left side of the slide, partially cut off by the edge.

# Set operators

---

Set operators are used to join the results of two (or more) SELECT statements.

The SET operators are

---

1.UNION

---

2.UNION ALL

---

3.INTERSECT

---

4.MINUS

# SET Operators

The **UNION** set operator returns the combined results of the two SELECT statements. Essentially, it removes duplicates from the results i.e. only one row will be listed for each duplicated result.

To counter this behaviour, use the UNION ALL set operator which retains the duplicates in the final result.

**INTERSECT** lists only records that are common to both the SELECT queries

**MINUS** set operator removes the second query's results from the output if they are also found in the first query's results.

# 1.UNION

- When multiple SELECT queries are joined using UNION operator, Oracle displays the combined result from all the compounded SELECT queries, after removing all duplicates and in sorted order (ascending by default), without ignoring the NULL values.



## Example:

```
select * from prod_det
```

o/p:

PID	PNAME	QUN	PRICE
1001	pendrive	100	900
1002	harddisk	200	4000
1003	headphone	1000	15000
1004	DVD	20	1000
1005	speaker	60	2400

## Example contd..

---

```
select * from sales_det
```

---

o/p:

---

SALES_NO	PID	QUN	PRICE	CUST_NAME
2001	1001	50	900	Sanvi
2002	1004	10	900	Anvi
2003	1003	120	15000	Manvi
2005	1002	40	4000	Akash

# Example contd..

---

```
select pid from prod_det
```

---

```
union
```

---

```
select pid from sales_det
```

---

```
o/p:
```

---

```
PID
```

---

```
1001
```

---

```
1002
```

---

```
1003
```

---

```
1004
```

---

```
1005
```

## 2.UNION ALL

- UNION and UNION ALL are similar in their functioning with a slight difference. But UNION ALL gives the result set without removing duplication and sorting the data. For example: In above query UNION is replaced by UNION ALL to see the effect.





Example:

---

```
select pid from prod_det
```

---

```
union all
```

---

```
select pid from sales_det
```

---

```
o/p:
```

---

```
PID
```

---

```
1001
```

---

```
1002
```

---

```
1003
```

---

```
1004
```

---

```
1005
```

---

```
1001
```

---

```
1004
```

---

```
1003
```

---

```
1002
```

### 3. INTERSECT

- Using INTERSECT operator, Oracle displays the common rows from both the SELECT statements, with no duplicates and data arranged in sorted order (ascending by default).



A large orange circle is positioned on the left side of the slide, partially cut off by the edge.

Example:

---

```
select pid from prod_det
```

---

```
intersect
```

---

```
select pid from sales_det
```

---

```
PID
```

---

```
1001
```

---

```
1002
```

---

```
1003
```

---

```
1004
```

## 4. MINUS

- Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data arranged in ascending order by default.





A large orange circle is positioned on the left side of the slide, partially cut off by the edge.

Example:

---

```
select pid from prod_det
```

---

Minus

---

```
select pid from sales_det
```

---

o/p:

---

PID

---

1005

**Thank You!!!**