Views

DBMS Assignment-03

View (VIRTUAL TABLE/RELATION)

- A view is simply any SELECT query that has been given a name. For this reason, a view is sometimes called a **"Named SQL query"**. Through this name we can access the select statement.
- A table is where the data is actually stored, with a fixed structure defined by a table definition, whereas the view is a Named Sql Query which can represent stored data from the tables in a way convenient for it's consumer.
- A view takes up **no storage space** other than for the definition of the view in the data dictionary.
- A view contains **no data**. All the data it shows comes from the base tables.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

The basic **CREATE VIEW** syntax is as follows −

```
CREATE VIEW view_name AS
        SELECT column1, column2.....
           FROM table_name
              WHERE [condition];
```

Example

Consider the EMPLOYEE table having the following records −

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID
111	Steven	King	24000	20
112	John	Hopkings	12000	30
113	Alexander	Roy	10000	20
114	Carlie	Nayer	23000	20
115	Julies	Ceaser	8000	40
116	James	Mathew	9000	30
117	Andrew	Matt	5500	30
118	Sunil	Pal	25000	20
119	Roshan	Kumar	15000	40
120	Rahul	Kapoor	16000	40

Following is an example to create a view from the EMPLOYEE table. This view would be used to have customer FIRST_NAME and SALARY from the EMPLOYEE table.

```
SQL > CREATE VIEW EMPLOYEE_VIEW AS
    SELECT FIRST_NAME, SALARY
    FROM EMPLOYEE;
```

```
SQL > SELECT * FROM EMPLOYEE_VIEW;
```

```
SQL > SELECT * FROM EMPLOYEE_VIEW;
```

FIRST_NAME	SALARY
Steven	24000
John	12000
Alexander	10000
Carlie	23000
Julies	8000
James	9000
Andrew	5500
Sunil	25000
Roshan	15000
Rahul	16000

Updating a View

A view can be updated under certain conditions which are given below −

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.

Example to update View

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the salary of John.

```
SQL > UPDATE EMPLOYEE_VIEW SET SALARY = 13500 WHERE name = 'John';
```

This would ultimately update the base table EMPLOYEE and the same would reflect in the view itself.

Inserting Rows into a View

- To insert a row, a view has to satisfy all the constraints on any column of underlying table over which it's created.
- A view must contain the entire Not Null column from underlying table. The failure to include any columns that have NOT NULL constraints from the underlying table in the view will result in the non-execution of the INSERT statement.
- If there is any column with referential key constraint in underlying table then view must contain that column and a proper value must be provided to that column while performing Insert dml on the View and many more.
- Thus to perform an INSERT dml on a view, it has to include all the columns which either have NOT NULL constraint or Referential constraint or any other mandatory constraint

Deleting Rows from a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having SALARY = 24000.

```
SQL > DELETE FROM EMPLOYEE_VIEW WHERE SALARY = 24000;
```

This would ultimately delete a row from the base table EMPLOYEE and the same would reflect in the view itself.

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below −

```
DROP VIEW view_name;
```

Purpose

- **Security:**
- views can be used to give access to selective data in a table. Views allow you to implement an additional security layer. They help you hide certain columns and rows from the underlying tables and expose only needed data to the appropriate users.
- **Views help simplify data retrieval significantly:**
- First, you build a complex query, test it carefully, and encapsulate the query in a view. Then, you can access the data of the underlying tables through the view instead of rewriting the whole query again and again.
- I.e. Developers can use a view instead of a difficult query to see the results they need.
- So, if you have a complex query that is used frequently, it makes sense to make a view out of it. It simplifies the work of the developers and can improve efficiency.
- Or we can also say that: A view **hides implementation complexity**. The user can select from the view with a simple SQL, unaware that the view is based internally on a join between multiple tables.

Simplify data Retrieval

```
SELECT customerid, customername FROM customers WHERE countryid='US';
```

```
CREATE VIEW view_uscustomers AS SELECT customerid, customername FROM customers WHERE countryid='US';
```

```
SELECT * FROM view_uscustomers WHERE customerid BETWEEN 100 AND 200;
```

And Oracle will transform the query into this:

```
SELECT * FROM (select customerid, customername from customers WHERE countryid='US') WHERE customerid BETWEEN 100 AND 200;
```

```
SELECT
    name AS customer,
    SUM( quantity * unit_price ) sales_amount,
    EXTRACT(
        YEAR
    FROM
        order_date
    ) YEAR
FROM
    orders
INNER JOIN order_items
        USING(order_id)
INNER JOIN customers
        USING(customer_id)
WHERE
    status = 'Shipped'
GROUP BY
    name,
    EXTRACT(
        YEAR
    FROM
        order_date
    );
```

CUSTOMER	SALES_AMOUNT	YEAR
International Paper	613735.06	2015
AutoNation	968134.3	2017
Becton Dickinson	484279.39	2016
Plains GP Holdings	655593.37	2016
Supervalu	394765.27	2017
Centene	417049.24	2017
CenturyLink	711307.09	2016
Abbott Laboratories	697288.63	2016
AutoNation	236531.07	2016
Jabil Circuit	508588.59	2017

- This query is quite complex. Typing it over and over again is time-consuming and potentially cause a mistake. To simplify it, you can create a view based on this query:.

```
CREATE OR REPLACE VIEW customer_sales AS
SELECT
    name AS customer,
    SUM( quantity * unit_price ) sales_amount,
    EXTRACT(
        YEAR
    FROM
        order_date
    ) YEAR
FROM
    orders
INNER JOIN order_items
        USING(order_id)
INNER JOIN customers
        USING(customer_id)
WHERE
    status = 'Shipped'
GROUP BY
    name,
    EXTRACT(
        YEAR
    FROM
        order_date
    );
```

Now, you can easily retrieve the sales by the customer in 2017 with a more simple query:

```
SELECT
    customer,
    sales_amount
FROM
    customer_sales
WHERE
    YEAR = 2017
ORDER BY
    sales_amount DESC;
```

CUSTOMER	SALES_AMOUNT
Raytheon	2406081.53
NextEra Energy	1449154.87
Southern	1304990.28
General Mills	1081679.88
AutoNation	968134.3
Emerson Electric	952330.09
Progressive	950118.04
Aflac	698612.98
Goodyear Tire & Rubber	676068.67
AutoZone	645379.54
Jabil Circuit	508588.59

Constraints

With Read Only: If view is created with "*WITH READ ONLY*" **no data manipulation** is allowed in any condition. **Only selects** are allowed against the views.

With check Option: when a view is created with "*WITH CHECK OPTION*" operations such as insertion or updation are allowed based on the condition provided in the where clause. It is used for restricted DML operation.

CREATE [**OR REPLACE VIEW**] view_name **AS** select_statement
WITH CHECK OPTION;

Constraints

```
Create table jbt_employee(name varchar2(20), address varchar2(20), phone number)
```

Table Created

```
CREATE OR REPLACE FORCE VIEW jbt_read_only_view AS SELECT * FROM jbt_employee
WHERE phone < 8 WITH READ ONLY;
```

View Created with "*WITH READ ONLY*"

```
CREATE OR REPLACE FORCE VIEW jbt_check_option_view AS SELECT * FROM jbt_employee
WHERE phone < 8 WITH CHECK OPTION;
```

View Created with "*WITH CHECK ONLY*"

```
insert into jbt_check_option_view values('name', 'address', 9);
Error Thrown : ORA-01402: view WITH CHECK OPTION where-clause violation
```

Note*: Query is trying to insert a row which view itself can not access

After changing the query and providing the phone *number < 8*

```
insert into jbt_check_option_view values('name', 'address', 7); Successfully Fired.
```

Now row will be created as it can be accessed by view itself(Based on Where clause).

Regular and Materialized Views

Regular view: Short name for a query, no additional space is used here.

- It stores only the name of the query or metadata of the query will be stored it wont store the data of the result. The result of a query is not physically stored.
- It always fetches data from the base table.
- Any changes made in the base table will automatically reflect in the view.

Materialized view: Similar to creating table whose data will refresh periodically based on data query used for creating the view.

- Whenever you create the materialized view. It stores the result of the query on the physical storage the way it stores a table
- It fetches data from local data stored in the materialized view not from the base table.
- Any changes made in the base table wont reflect in the M-view immediately
- To get the data from the base tables into M-view we have a concept called “Refresh”.