What is Materialized View?

It is a one type of view which contains the result of a query. It caches the result of complex query and you can access the result data like a normal table. It requires physical space to store generated data. Once we put any complex query in Materialized View, we can access that query and data without disturbing a physical base table. Only one thing you should do is: Periodically refresh your Materialized View to get newly inserted data from the base table.

It is more efficient to use materialized views if query involves summaries, large or multiple joins or both. It is a pre-computed table comprising aggregated or joined data from fact. Also known as a summary or aggregate table and mainly used for improving query performance or providing replicated data.

Key Features

Use Index & Partition

In PostgreSQL, You can create a Materialized View and can refresh it. Let me show you, full practical on this.

Create a table with sample data:

```
create table test (id int , name varchar)

select * from test t ;

insert into test values(2,'Bibek');

insert into test values(1,'Umesh');

select * from test t ;
```

```
create materialized view  test_vw
```

```
as
select *
from TEST t ;

SELECT * FROM test_vw;
REFRESH MATERIALIZED VIEW test_vw WITH DATA;

---In Oracle
DROP  MATERIALIZED VIEW test_vw;
create materialized view  test_vw
as
select *
from TEST t ;

SELECT * FROM test_vw;
INSERT INTO TEST values(1,'ram');
SELECT * FROM TEST t ;

BEGIN
DBMS_SNAPSHOT.REFRESH('test_vw');
END;
```

Refresh a materialized view:

Once you create a materialized view, you should also refresh it otherwise newly inserted values of the table will not update in this view.

Because It acts like a physical table and once your base table update, you should refresh the data of the materialized view.

-----Concurrently Materialized View ---

```
drop table t_demo cascade;

CREATE TABLE t_demo (grp int, data numeric);

INSERT INTO t_demo SELECT 1, random()

FROM generate_series(1, 5);


select * from t_demo; --insert


CREATE MATERIALIZED VIEW mat_view AS

        SELECT   grp, avg(data), count(*)

        FROM     t_demo

        GROUP BY 1; --default with data


 select * from mat_view;--- summary


 CREATE OR REPLACE FUNCTION refresh_test1_mv()
```

```
  RETURNS TRIGGER LANGUAGE plpgsql
  AS $$
  BEGIN
  REFRESH MATERIALIZED VIEW CONCURRENTLY mat_view ;
  RETURN NULL;
  END $$;


  CREATE TRIGGER refresh_test1_mv_trigger
  AFTER INSERT OR UPDATE OR DELETE OR TRUNCATE
  ON t_demo
  FOR EACH STATEMENT
  EXECUTE PROCEDURE refresh_test1_mv();


CREATE UNIQUE INDEX idx_grp ON mat_view (grp);
INSERT INTO t_demo SELECT 1, random()
FROM generate_series(1, 5);

select * from mat_view;
```

Difference between view and materialized view?

View	Materialized view
1) A view is a virtual table from one or more base tables which is never stored on disk.	1) Materialized view is a physical copy of the base table and stored on disk.
2) Views are updated every time when the base tables are updated.	2) Updated manually or using triggers.
3) Views are slow processing. (If you update any content in View, it is reflected in the original table, and if any changes had been done to the original base table, it would reflect in its View. But this makes the performance of a View slower.)	3) Fast processing. (Materialized View responds faster in comparison to View. It is because the materialized view is precomputed.)
4) Views are not pre-computed as needed to compute each time when it is used or accessed.So, always we have updated data in view.	4) But unlike View, the Materialized View is precomputed and stored on a disk like an object, and they are not updated each time they are used. Instead,the materialized view has to be updated manually or with the help of triggers. The process of updating Materialized View is called Materialized View Maintenance.
5) Advantages:- i) No storage space needed.	5) Advantages:- i) Faster than views.
ii) Restrict users from accessing information in a database. iii) Reduces the complexity of queries by getting data from several tables into a single customized View.	
6) All views are not updateable.	6) Need to update manually or by using triggers.

Conclusion

A “materialized view” is a database object which stores the result of a precalculated database query and makes it easy to refresh this result as needed. Materialized views are an integral feature of pretty much all advanced database systems. **The results are stored physically**, making it acting like a cache.

Anyone who is looking to improve the performance of queries in their Data Warehouse or database should seriously consider implementing materialized views if they can pre-compute the results of some queries. They can help to speed up large calculations – or at least to cache them.