# SQL Views


pm jat @ daiict


# SQL Views

- Relations that are defined using CREATE TABLE statement actually hold the data, and are called “stored relations” or “base relations”.
- They actually hold the data.
- There can another type of relations in relational databases that do not hold data; there data are generated by executing associated query.
- Such relations are referred as Views or Virtual Relations.
- Views and relations can be used in same manner while expressing queries.
- PS: the term “Table” often refer to base relation while term “View” refers to Virtual relations.


# Create and using a View

- Views are created using **CREATE VIEW** statement
- Example:

```
CREATE VIEW research_projects AS
        SELECT project.* FROM project
                NATURAL JOIN department
                WHERE dname = 'Research';
```

- Query from:

```
SELECT * FROM research_projects;
```

The query associated with view gets executed and result is given to FROM clause in this query.


# Create and using a View

- Views can be treated like any other base relation in queries; for example following query will result employees working on research projects-

  SELECT employee.* FROM research_projects NATURAL JOIN WORKS_ON JOIN EMPLOYEE ON essn = ssn;

- Find average salary of people who work on research projects -

  SELECT avg(salary) FROM research_projects NATURAL JOIN WORKS_ON JOIN EMPLOYEE ON essn = ssn;


# Create and using a View

- A variation to create view is “CREATE OR REPLACE VIEW”- allows to replace existing definition of view as long target schema remains same.

```
CREATE OR REPLACE VIEW research_projects
AS
    SELECT project.* FROM project
         NATURAL JOIN department
         WHERE dname = 'Research';
```


# Examples

- You can rename name of attributes while creating views-
  **CREATE VIEW pay (name, annual_pay) AS**
  **SELECT fname, salary * 12**
  **FROM employee;**
- Later you query the view, as-
  **SELECT name, annual_pay FROM PAY;**
- Off course renaming could have been done as following as well -
  **CREATE VIEW pay AS SELECT fname AS name, salary*12 AS annual_pay FROM employee;**


# DROP View

```
DROP VIEW PAY;
```


# Benefits of Views?

- To hide data from users;
  - for example we may not like to return salary column when SELECT * FROM EMPLOYEE is requested. Instead they do not know the name of actual relation and what they know is name of a view that does not include salary column.
- Views can be used to encapsulate complex queries
  - complex queries are created by experts and saved as views. Application developer (like java developer) just write simple queries on views.
- Adds to “logical data independence”
  - If applications deal with views, we can change the underlying schema without affecting applications


# Updatable views

- What if we try to update view relation as following-

```
UPDATE research_projects set pname =
'ABC' where pno = 1;
```

- If this sort of update are permitted then the view is updatable.


# Updatable views - example

- UPDATE EMP SET DOB = ‘1987-11-11’ where ssn = 123;


# Updatable views - example

- You can rename name of attributes while creating views-
  **CREATE VIEW pay (name, annual_pay) AS**
  **SELECT fname, salary * 12**
  **FROM employee;**
- Later you query the view, as-
  **SELECT name, annual_pay FROM PAY;**
- **UPDATE pay SET name = 'ABCD' where ssn=1234;**
- **UPDATE pay SET annual_pay = 234566 where ssn=1234;**
- **INSERT INTO PAY VALUES(…)**


# Updatable views

- An updatable view is one, that you can use to insert, update, or delete underlying base table rows
- SQL does define some rules that determines what view can be updatable. But those rules are complex and we do not plan to discuss them here.
- However in general we can say that if attributes in view-relation map to distinct base relations with no ambiguity then the view can be, by default, updatable.
- PostgreSQL views are not updatable


# Making views updatable

- By defining “INSTEAD OF trigger” for update operation on a view, we can make them updatable.

- “INSTEAD OF Trigger” is typically a “stored procedure” that actually automatically gets executed whenever update is requested on a relation.


# Why we want updatable views

- Sometimes end-users do not know whether the relation they are working are base relation or virtual relation

- They may require to permitted to update certain views, without letting them name of actual underlying base relations.


# Materialized Views

- Views of which results are also stored (and kept updated based on some strategy)
- Materialized views are useful where source table is very huge (in terms of billions of tuples), and query is complex or slow.
- Data warehouses basically contain a large number of materialized views (and primarily talk about various strategies of computing and maintaining complex views over large distributed databases)