Databases Design. Introduction to SQL

LECTURE 12

CTE. Views

Common Table Expression

- A **CTE (Common Table Expression)** is a temporary result set which you can reference within another SQL statement.

- Common Table Expressions are temporary in the sense that they only exist during the execution of the query.

- Common Table Expressions are typically used to simplify complex joins and subqueries in PostgreSQL.

Syntax

```
WITH cte_name AS (
          cte_query_definition)
statement;
```

- Use the CTE like a table in SQL statements.

Example 1 with subquery

```
SELECT *
FROM ( SELECT group_id, count(*) AS
                                    num_of_stud
          FROM Students
         GROUP BY group_id) StudNum
WHERE num_of_stud > 20;
```

Example 1 with CTE

```
WITH StudNum AS (
    SELECT group_id, count(*) AS num_of_stud
    FROM Students
    GROUP BY group_id
)

SELECT *
FROM StudNum
WHERE num_of_stud > 20;
```

Example 2 with subquery

```
SELECT *
FROM
(SELECT count(*) FROM students) students,
(SELECT count(*) FROM teachers) teachers;
```

Example 2 with CTE

```
WITH Stud AS (
   SELECT count(*)
   FROM students),
Teach AS (
   SELECT count(*)
   FROM teachers)

SELECT *
FROM Stud, Teach;
```

Example 3: CTE with join

```
WITH StudNum AS (
     SELECT group_id, count(*) AS num_of_stud
     FROM Students
     GROUP BY group_id
)

SELECT g.name, s.num_of_stud
FROM  Groups g
INNER JOIN StudNum  s
ON g.group_id = s.group_id;
```

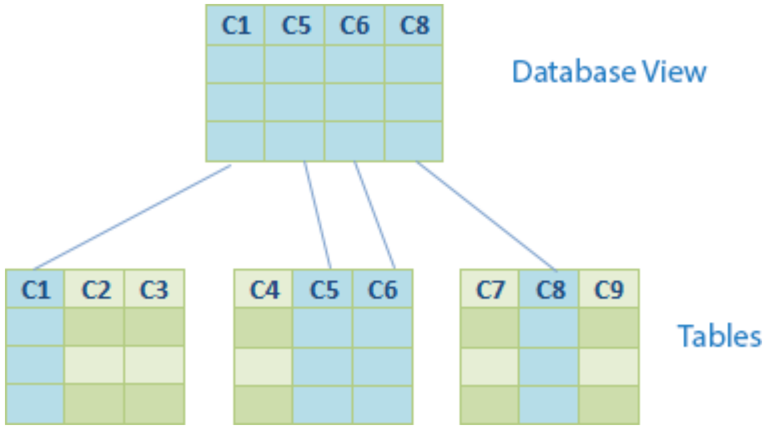
CTE advantages

- Improve readability of the complex queries. You can use CTEs to organize complex queries in a more organized and readable manner.
- Ability to create recursive queries. Recursive queries are queries that reference themselves.

View

View is a virtual table based on the result-set of an SQL statement.

View contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.



View

- Views do not physically exist.
- Views are virtual tables.

- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

- A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

- Views are supported by all main DBMS.

Use of view: case 1

- In some cases, we may not want users to view all information in a table(s).
- Users need to be restricted from accessing this information.

Use of view: case 2

- In other case, a complex set of relational tables does not lend itself to easy use by non-database professionals.

- Consider a clerk at the library performing an audit. This clerk is only interested in the names of each member and the number of books those member have borrowed.
- Should this clerk have to write complex queries involving aggregate functions and joins over multiple tables?
- Probably not.

Use of views

Views allow users to do the following:

- Restrict access to the data such that a user can only see limited data instead of complete table.
- Structure data in a way that users or classes of users find natural or intuitive.
- Summarize data from various tables, which can be used to generate reports.

CREATE VIEW

- A view is created using the CREATE VIEW command in SQL with a SELECT statement on the defining tables.

- Syntax:

```
CREATE VIEW view_name
AS
SELECT …;
```

View example

- Example: create a view named Students_info that stored only the first and last name from the Students table.

```
CREATE VIEW Students_info
AS
SELECT fname, lname
FROM Students;
```

CREATE OR REPLACE VIEW

- CREATE OR REPLACE VIEW is similar, but if a view of the same name already exists, it is replaced.
- The new query must generate the same columns that were generated by the existing view query (that is, the same column names in the same order and with the same data types), but it may add additional columns to the end of the list.
- The calculations giving rise to the output columns may be completely different.
- CREATE OR REPLACE VIEW is a PostgreSQL language extension.

CREATE OR REPLACE VIEW

Syntax:

```
CREATE OR REPLACE VIEW view_name
AS
SELECT ….;
```

Example:

```
CREATE OR REPLACE VIEW Students_info
AS
SELECT fname, lname, stud_id
FROM Students;
```

View with joining

- Views may also be built by joining many tables.

- Example: create a view that contains last name and group’s name of each student.

```
CREATE VIEW Students_groups
AS
SELECT s.stud_id, s.lname, g.group_id, g.name
FROM Students s, Groups g
WHERE s.group_id = g.group_id;
```

View deletion

- Views can be deleted using the DROP VIEW statement.
- Example: delete the Students_groups view created on the previous slide.

```
DROP VIEW Students_groups;
```

View updating

- Updates to views are not simple.
- Recall that views are *virtual tables* – they do not physically exist.
- Any updates to views must be mapped onto the defining tables. If an update cannot be mapped, then a view is unupdatable.

View updating

Note. For a view to be updatable, the DBMS must be able to trace any row or column back to its row or column in the source table.

- In general, a view is updatable if it contains a single table and contains a primary key.

- Generally, a view is not updatable if it contains a join operation.

- A view is definitely not updatable if it involves an aggregate function or a subquery.

View updating

- Use DML command to update view Students_info:

```
UPDATE Students_info
SET fname = 'Alan'
WHERE stud_id = 3;
```

- It’s identical to operation in the physical table Students.

View updating

- Suppose we slightly altered the view to include only students with group_id = 2.

```
CREATE VIEW Group_2
AS
SELECT stud_id, fname, lname, group_id
FROM Students
WHERE group_id = 2;
```

View

- One problem with updatable views are the rows that we attempt to insert may violate the selection condition.
- Suppose we tried to update the view to change the student’s group_id to 3.

- Will that student still be part of the view?
- No, that student will not be part of the view. That row will ***migrate*** from the view.

Using views as physical tables

- Views can be used like any other real tables in database. Also you can build view based on other views.

  Creating of the view:

  ```
  CREATE VIEW Students_info
  AS
  SELECT fname, lname
  FROM Students;
  ```

- Using of the view:

  ```
  SELECT * FROM Students_info;
  ```

Database Security: Access Control

- The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users.
- The user is not aware of the existence of any attributes or rows that are missing from the view.
- A view can be defined over several relations with a user being granted the appropriate privilege to use it, but not to use the base relations.
- In this way, using a view is more restrictive than simply having certain privileges granted to a user on the base relation(s).

Summary

- A view is the dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a *virtual relation* that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.
- Views can represent a subset of the data contained in a table.
  - A view can limit the degree of exposure of the tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.
  - Views can join and simplify multiple tables into a single virtual table. Views can hide the complexity of data.
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.

Books

- Connolly, Thomas M. Database Systems: A Practical Approach to Design, Implementation, and Management / Thomas M. Connolly, Carolyn E. Begg.- United States of America: Pearson Education
- Garcia-Molina, H. Database system: The Complete Book / Hector Garcia-Molina.- United States of America: Pearson Prentice Hall
- Sharma, N. Database Fundamentals: A book for the community by the community / Neeraj Sharma, Liviu Perniu.- Canada

- www.postgresql.org/docs/manuals/
- www.postgresql.org/docs/books/