Materialized View Definition and Maintenance in PostgreSQL

Group 2

Amita Girija
Namrata Sayali

Problem

Problem Statement

Implement Creation and Maintenance of Materialized Views in PostgreSQL.

Scope

Scope

- Adding commands (CREATE, DROP)

Scope

Scope

- Adding commands (CREATE, DROP)
- Creating materialized views

Scope

Scope

- Adding commands (CREATE, DROP)
- Creating materialized views
- Writing triggers for updation of views after modifications on base table

Scope

Scope

- Adding commands (CREATE, DROP)
- Creating materialized views
- Writing triggers for updation of views after modifications on base table
- Dropping materialized views

Design and Implementation

1. Grammar
2. Creation
3. Maintenance
4. Dropping

Grammar

Adding following commands

- CREATE MATERIALIZED VIEW viewname AS SELECT column_list (with aliases if any) FROM from_items WHERE conditions_list GROUP BY expression HAVING conditions_list
- DROP MATERIALIZED VIEW viewname

Creation and Maintenance

Steps

1. Creating a table corresponding to this materialized view.
2. Maintenance
   - A function to update view as per view definition
   - A trigger to invoke this function after insert/delete/update on base tables

Dropping

Dropping Views

- Drop existing view
- Drop functions and triggers on it (kept for maintenance)

Approaches

Approach one : Multiple query strings

- Main query strings : CREATE and DROP Commands
- Form query strings for CREATE FUNCTION, CREATE TRIGGER and DROP FUNCTION
- Let parser handle rest of the things

Approach two : Multiple parse trees

- Main parse trees : CREATE and DROP
- Generate parse trees for CREATE FUNCTION, CREATE TRIGGER and DROP FUNCTION

What we have done?

Approach 2

In parser module, we generate multiple parse trees and return the list. Advantages of this over the first approach are as follows:

- The entire operation remains atomic (creation of view, functions and corersponding triggers)
- All the operations which are possible in select statement are supported in view creation

Implementation

Grammar and Creation

- Adding a production in gram.y for CREATE command
- Adding code for parse tree generation for this production

Maintenance

- Adding code for parse tree generation for function updating the view.
- Adding code for parse tree generation for trigger executing the function

Dropping

- Adding a production in gram.y for DROP command
- Adding code for parse tree generation for dropping the view
- Adding code for parse tree generation for dropping functions (and triggers) defined on the view

References

Web

- http://www.jonathangardner.net/PostgreSQL/materialized_views/matviews.html
- http://doxygen.postgresql.org/