Sequences

15-721 Project #3 Final Presentation

Adrian Chang (adrianlc) Tianhan Hu (tianhanh) Zian Ke (ziank)

Overview of Sequences

- Global counters that can be used as auto-increment keys for tables.
- Support functionalities:
  - **`nextval`** function (shared across session)
  - **`currval`** function (last nextval in the local session)
- Example usage:
  - **`Terminal 1: nextval -> 2, Terminal 2: nextval -> 3, Terminal 1: nextval -> 4`**
  - **`Terminal 1: currval -> 2, Terminal 2: nextval -> 3, Terminal 1: currval -> 2`**

Our Progress

Establish the end-to-end **`CREATE SEQUENCE`** and **`DROP SEQUENCE`** pipeline.

Store sequences in database catalog **`pg_class`** and **`pg_sequence`**.

Implement built-in functions **`nextval`** and **`currval`** with expected behaviors when called upon a sequence.

Modified Files

1. Parser (postgresparser.cpp)
2. Binder (bind_node_visitor.cpp)
3. Optimizer (logical_operators.cpp, physical_operators.cpp, query_to_operator_transformer.cpp, plan_generator.cpp)
4. Planner (create_sequence_plan_node.cpp, drop_sequence_plan_node.cpp)
5. TrafficCop (traffic_cop.cpp)
6. Execution (ddl_executors.cpp, vm.cpp)
7. Catalog (catalog_accessor.cpp, database_catalog.cpp, pg_sequence.h)
8. Built-in Function (sema_builtin.cpp, string_functions.cpp, builtins.h)
9. Unit tests (parser_test, binder_test, logical_operator_test, physical_operator_test, operator_transformer_test, catalog_test, ddl_executors_test, SequenceTest.java)

Quality of Code

- **`CREATE SEQUENCE`** and **`DROP SEQUENCE`** pipeline code is strong.
- **`nextval`** implementation is functionally correct. However, due to the limitation of built-in functions, **`nextval`** has to be called from a table.
- **`currval`** needs strengthening. Currently stored in unordered_map (from session_id to session_local_currval). Garbage collection not supported yet. Will consider using temp tables for each session.
- High test coverage (unit tests for most modules we have modified, JUnit integration test for the basic usage of sequences).

Demo

Next Steps

1. Implement **`minvalue`**, **`maxvalue`**, **`increment`** ...
    - https://www.postgresql.org/docs/12/sql-createsequence.html
2. Benchmarks against PostgreSQL.
3. Implement **`cache`** option.

Thank you!

Any questions?