Final Report: SQL Features

Shuxin Lin (shuxinl@andrew.cmu.edu)
https://www.linkedin.com/in/shuxin-lin/
Yuan Luo (yluo2@andrew.cmu.edu)
https://www.linkedin.com/in/yuanluo-cmu/

Overview

- 75% work done
  - ✓ Issue 551
  - ✓ Issue 552
- 100% work done
  - ✓ Issue 562
  - ✓ Issue 543
  - ✓ Issue 401
  - Issue 561
- 125% work done
  - ✓ Issue 515
- Benchmark & testing
- Pull Request URL:
- https://github.com/cmu-db/peloton/pull/669

75%: WORK DONE
✓ Issue 551: support NULL insertion

Be able to insert a tuple with NULL values into table.

Before 1st code review

- Support simple_optimizer

After 1st code review

- Support new_optimizer
- Support codegen (llvm)

75%: WORK DONE
✓ Issue 551: support NULL insertion

Be able to insert a tuple with NULL values into table.

```
postgres=# select * from xxx;
 a | b
---+---
 1 |
 2 | 3
(2 rows)

postgres=# select * from xxx where b is null;
 a | b
---+---
 1 |
(1 row)
```

Before 1st code review

- Support simple_optimizer

After 1st code review

- Support new_optimizer
- Support codegen (llvm)

75%: WORK DONE
✓Issue#552: Support "is null" and "is not null" in where clause

Be able to evaluate whether a value is (not) NULL is where clause

Before 1st code review

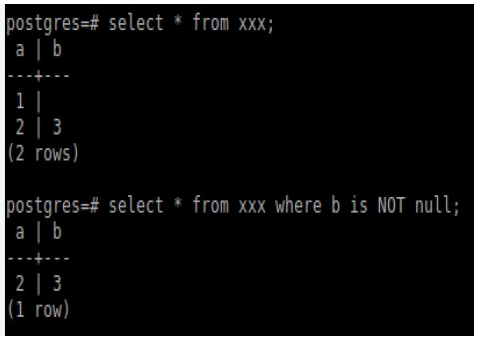
- Support simple_optimizer

After 1st code review

- Support new_optimizer
- Support codegen (llvm)

75%: WORK DONE
✓Issue#552: Support "is null" and "is not null" in where clause

Be able to evaluate whether a value is (not) NULL is where clause



Before 1st code review

- Support simple_optimizer

After 1st code review

- Support new_optimizer
- Support codegen (llvm)

100%: WORK DONE
✓Issue#562: GROUP BY is broken

```
postgres=# select * from xxx;
 id | name  | salary
----+-------+--------
  1 | Mike  |   1000
  2 | Jane  |   2000
  3 | Tom   |   3000
  4 | Kelly |   4000
  5 | Lucy  |   3000
  6 | Tim   |   2000
(6 rows)

postgres=# select count(id), salary from xxx group by salary;
 count | salary
-------+--------
     1 |   1000
     2 |   2000
     1 |   4000
     2 |   3000
(4 rows)
```

After 1st code review

- Support new_optimizer
- With bug: VARCHAR anomaly

100%: WORK DONE
✓Issue#543: Cannot use exprs inside of aggregate functions

After 1st code review

- Support new_optimizer
- SELECT SUM(a+b) from table;

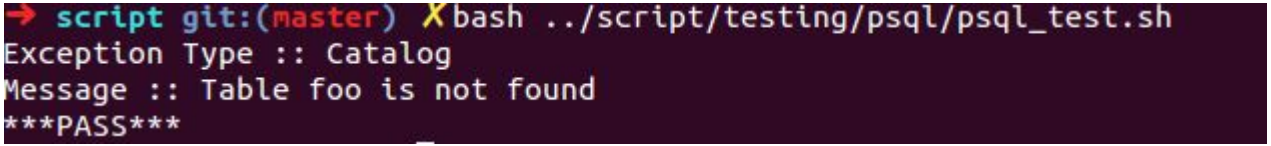
```
postgres=# select * from xxx;
 a | b
---+---
 1 |
 2 | 3
 3 | 5
(3 rows)

postgres=# select sum(a+b) from xxx;
 sum
-----
 13
(1 row)

postgres=# select avg(a*b) from xxx;
 avg
------
 10.5
(1 row)
```

100%: WORK DONE
✓Issue#401: Inconsistent results with basic1.sql

Inconsistent results in Jenkins and in local.

After 1st code review

- PASS on command-line



100%: WORK DONE
Issue#561: Support IN in where clause

Support Array_type for posgresparser in Expr_Transform

After 1st code review

- Done: the codes in src/parser & src/type
- Not done: the codes in src/codegen -> need a translator

125%: WORK DONE
Issue#515: Add TRUNCATE SQL command

After 1st code review

- Run using Postgresparser and new_optimizer

125%: WORK DONE
Issue#515: TRUNCATE

```
postgres=# create table a(id int, value int);
create table a(id int, value int) 0
postgres=# insert into a values(1, 2);
insert into a values(1, 2) 1
postgres=# select * from a;
 id | value
----+-------
  1 |     2
(1 row)

postgres=# truncate table a;
truncate table a 1
postgres=# select * from a;
 id | value
----+-------
(0 rows)
```

BENCHMARK AND TESTING

Before 1st code review

- Wrote 4 unit test for new features in test/sql/new_feature_test.cpp
- All tests passed.

After 1st code review

- Wrote 3 unit test for new features in test/type/array_type_test.cpp, test/sql/truncate_test.cpp, test/sql/aggregation_sql_test.cpp, test/sql/group_by_sql_test.cpp
- All tests pass under new_optimizer

More on WHERE IN and TRUNCATE

We implemented where in using ARRAY type and wrote test cases to test the constant value expression evaluate method with ARRAY type, and comparison expression with tuples the array. Test cases all passed, however, running where in command will encounter seg fault. We spent days to figure out the seg fault reason. Since the unit test for the internal implementation of where in passed, we think the seg fault may from the steps after the parser tree, like optimizer or other part of the process.

For truncate, the current implementation assume truncate is equivalent to “delete from table”, and no not implement exception. And later when we trying to do truncate we realize we need to deallocate the tiles. But we didn’t understand the how the tile group is managed, so we found out it’s difficult to do it.