Precompiled Catalog Access Query

Ruogu Du, Tianxiong Wang, Ao Zeng
What We Have Done

- 75% Goal:
  - Table catalog sequential access with LLVM compilation and cached plan.
- 100% Goal:
  - Support get object for the rest kinds of catalog
- 125% Goal:
  - Support index scan
- 110% Goal:
  - Rebase our code onto the schema catalog refactoring
  - Add compilation support for insert, update and delete
- 125% Goal:
  - Fix (many) bugs in the existing code base to ensure correctness
What We Have Done

Support compiled sequential scan for fast catalog object lookups, insert, update and delete.

Also cache the compiled query for fast repeat execution.

Supported catalogs includes:

- DatabaseCatalog
- SchemaCatalog
- TableCatalog
- ColumnCatalog
- DatabaseMetricsCatalog
- IndexMetricsCatalog
- IndexCatalog
- LanguageCatalog
- ProcCatalog
- QueryHistoryCatalog
- QueryMetricsCatalog
- SettingCatalog
- TriggerCatalog
- ZoneMapCatalog
- ColumnStatsCatalog
- TableMetricsCatalog
Support Insertion/Deletion

Create Plan

Create insertion/deletion plans and perform binding

Compile/Check Cache

Check the plan in the query cache. Compile the plan if cache miss

Execute

Execute the compiled insertion/deletion plans
Create Insertion Plans

```cpp
std::vector<std::vector<ExpressionPtr>> tuples;
auto val0 = type::ValueFactory::GetIntegerValue(database_oid);
auto val1 = type::ValueFactory::GetVarcharValue(database_name, nullptr);

auto constant_expr_0 = new expression::ConstantValueExpression(
    val0);
auto constant_expr_1 = new expression::ConstantValueExpression(
    val1);

tuples.emplace_back();
auto &values = tuples[0];
values.emplace_back(ExpressionPtr(constant_expr_0));
values.emplace_back(ExpressionPtr(constant_expr_1));

return InsertTupleWithCompiledPlan(&tuples, txn);
```
Create Deletion Plans

```cpp
// cache miss, get from pg_database
std::vector<oid_t> column_ids(all_column_ids);

auto *db_oid_expr =
    new expression::TupleValueExpression(type::TypeId::INTEGER, 0,
                                          ColumnId::DATABASE_OID);

db_oid_expr->SetBoundOid(catalog_table_->GetDatabaseOid(),
                          catalog_table_->GetOid(),
                          ColumnId::DATABASE_OID);

expression::AbstractExpression *db_oid_const_expr =
    expression::ExpressionUtil::ConstantValueFactory(
        type::ValueFactory::GetIntegerValue(database_oid).Copy());

expression::AbstractExpression *db_oid_equality_expr =
    expression::ExpressionUtil::ComparisonFactory(
        ExpressionType::COMPARE_EQUAL, db_oid_expr, db_oid_const_expr);

expression::AbstractExpression *predicate = db_oid_equality_expr;

// evict cache
txn->catalog_cache.EvictDatabaseObject(database_oid);

return DeleteWithCompiledSeqScan(column_ids, predicate, txn);
```
What We Decided Not to Do

We still decide to pass the wrapped tuple to the catalog object constructor and copy the value out instead of passing a pointer to the constructor. Since the catalog object can serve as a value cache for future references while if we pass the pointer to the wrapped tuple, we need to convert the value to the right type for every get method.

```cpp
TableCatalogObject::TableCatalogObject(codecgen::WrappedTuple &wrapped_tuple, concurrency::TransactionContext *txn)
: table_oid(wrapped_tuple.GetValue(TableCatalog::ColumnId::TABLE_OID)
  .GetAs<oid_t>())
, table_name(wrapped_tuple.GetValue(TableCatalog::ColumnId::TABLE_NAME)
  .ToString(),
, schema_name(wrapped_tuple.GetValue(TableCatalog::ColumnId::SCHEMA_NAME)
  .ToString(),
, database_oid(wrapped_tuple.GetValue(TableCatalog::ColumnId::DATABASE_OID)
  .GetAs<oid_t>())
, version_id(wrapped_tuple.GetValue(TableCatalog::ColumnId::VERSION_ID)
  .GetAs<uint32_t>())
, index_objects(),
, index_names(),
, valid_index_objects(false),
, column_objects(),
, column_names(),
, valid_column_objects(false),
  txn(txn) {}
Bugs We Discovered until the Midterm

QueryCache Bug #1298

Solved by binding predicates to columns.

Bug in scan #1294

Solved by reordering the columns in the seq scan. We tried to modify the binding but this caused other problems :(
Bugs We Discovered until the Midterm

- Trigger test assumes order in table's trigger list #1293

Solved by modifying the test.
More Bugs

- ZoneMap bug when using cached compiled query #1362
  - Opened by Zeninma 4 hours ago · 0 comments

- Destruction of uninitialized Value in FillPredicateArray #1363
  - Opened by Zeninma 3 hours ago · 0 comments

  Solved by passing the expression pointer directly to the ShouldScanZoneMap method instead of using PredicateInfo.
More Bugs

❑ Assertion fail when insert and delete in the same transaction #1336

Kind-hearted Prashanth solved this by completing the implementation of delete.
Setting optimization level can slight promote the performance
Evaluation

[15721] Compiled Catalog Query Access ✓ class-project do not merge

#1339 opened 10 days ago by nwang57 • Review required
Evaluation
Evaluation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted index</td>
<td>0.004</td>
<td>0.005</td>
<td>0.005</td>
<td>0.004</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Compiled seq</td>
<td>0.007</td>
<td>0.009</td>
<td>0.011</td>
<td>0.012</td>
<td>1.490</td>
<td>3.740</td>
<td>4.804</td>
</tr>
<tr>
<td>Interpreted seq</td>
<td>0.009</td>
<td>0.012</td>
<td>0.014</td>
<td>0.021</td>
<td>0.031</td>
<td>0.123</td>
<td>0.265</td>
</tr>
</tbody>
</table>
Future Works

- Support Compiled IndexScan
- LLVM passes for compilation optimization