MiniSQL阶段报告4

——Catalog Manager

第七小组

4.1实验概述

在MiniSQL的设计中，Catalog Manager负责管理和维护数据库的所有模式信息，包括所有表的定义信息，表中每个字段的定义信息，数据库中所有索引的定义等。它将被创建，修改，删除的模式信息重新持久化到数据库文件中。同时Catalog Manager为上层的执行器提供公共接口从而执行器可以获取目录信息并生成执行计划。

4.2目录元信息

数据库定义的表和索引在内存中以TableInfo和IndexInfo的形式表现，他们维护了与之对应的表或索引的元信息和操作对象。为了能够将所有表和索引的定义信息持久化到数据库文件并在重启时从数据库中恢复，我们需要为表和索引的元信息TableMetadata和IndexMetadata实现序列化和反序列化操作。在序列化时，我们为每一个表和索引都分配了一个数据页，所以要设计一个数据页和数据对象CatalogMeta来记录和管理这些表和元信息具体存储的数据页位置，同样CatalogMeta也需要被序列化到数据页上。

实现

- CatalogMeta::SerializeTo(*buf)
  - 依次将CatalogMeta中记录的两个map，分别是用来记录表的id与它所在的数据页和索引的id与它所在的数据页，序列化到数据库文件中的第CATALOG_META_PAGE_ID号数据页中。
- CatalogMeta::GetSerializedSize()
  - 返回CaltalogMeta序列化的字节偏移量。
- CatalogMeta::DeserializeFrom(*buf, *heap)
  - 将buf指向的内容依次反序列化成两个map：table_meta_pages,index_meta_pages。
- IndexMetadata::SerializeTo(*buf)
  - 依次将IndexMetadata中的index_id_,index_name_,table_id_,key_map_写入buf指向的内容。
- IndexMetadata::GetSerializedSize()
  - 返回IndexMetadata序列化的字节偏移量。
- IndexMetadata::DeserializeFrom(*buf,*&index_meta, *heap)
  - 将buf指向的内容依次反序列化成index_id_,index_name_,table_id_,key_map_。
- TableMetadata::SerializeTo(*buf)
  - 依次将table_id_,table_name_,root_page_id,schema_以及primarykey序列化，其中schema_调用Schema::SerializeTo(*buf)序列化，并写入buf指向的内容。
- TableMetadata::GetSerializedSize()
  - 返回TableMetadata序列化的字节偏移量。
- TableMetadata::DeserializeFrom(*buf,*&table_meta, *heap)
  - 将buf指向的内容依次反序列化成table_id_,table_name_,root_page_id,schema_调用Schema::DeserializeFrom(*buf,*&schema, *heap),以及primarykey。
- IndexInfo::Init(*index_meta_data,&table_info,*buffer_pool_manager)
  - 通过index_meta_data,table_info,调用Schema::ShallowCopySchema(table_info->GetSchema(),meta_data->GetKeyMapping, heap_)得到key_schema_来初始化IndexInfo。调用CreateIndex(buffer_pool_manager)来创建一个索引对象。

4.3表和索引的管理

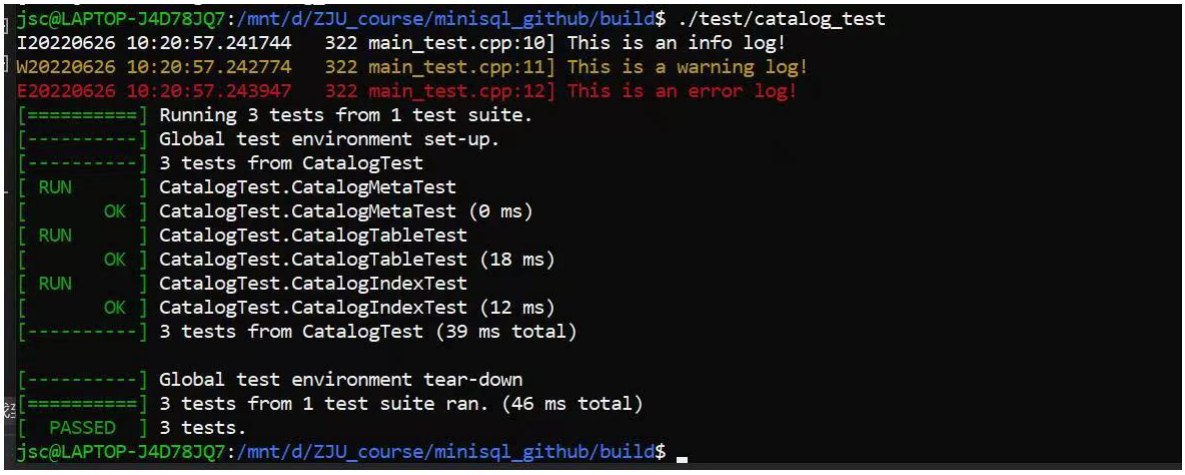
CatalogManager类维护和持久化数据库中所有表和索引的信息。在数据库初次打开时初始化所有数据，在后续打开时加载所有表和索引的信息。同时该类还需对上层模块提供对指定数据表和指定数据索引的操作方式。

实现:

- `CatalogManager::CatalogManager(BufferPoolManager *buffer_pook_manager,LockManager *lock_manager,LogManager *log_manager,bool init)`
  - 在第一次调用时初始化创建变量catalog_meta_，在后续调用时通过在缓冲池中获取数据页反序列化catalog_meta_。通过得到的catalog_meta_反序列化得到所有的tableinfo，indexinfo并且更新关于表和索引的两个map。
- `CatalogManager::~CatalogManager()`
  - 将更改过的新的表的数据和索引的数据重新序列化到内存池中。
- `CatalogManager:CreateTable(const string &table_name,TableSchema *schema,std::vector<Column>primary_key,Transaction *txn,TableInfo *&table_info)`
  - 创建一个新的表，创建这个表的元信息，并将它序列化到内存池中，同时创建一个指向表的信息的指针，并且在CatalogManager中的map中更新相关的值。
- `CatalogManager:GetTable(const string &table_name, TableInfo *&table_info)`
  - 在记录表信息的两个map里寻找所需要的表的信息，并且将指向它的表信息的指针返回。
- `CatalogManager:GetTables(vector<TableInfo *>&tables) const`
  - 找到多张表的信息并返回。
- `CatalogManager::CreateIndex(const std::string &table_name,const string &index_name,const std::vector<std::string>&index_keys,Transaction *txn,IndexInfo *&index_info)`
  - 先检查是否存在一个在该表里的相同的指针。通过指向索引信息的指针创建一个key_map，并且创建索引的元信息，序列化到内存池中，更新指向表的信息的指针，并且在CatalogManager中的map中更新相关的值。最后根据表的内容和前面得到的内容创建这个新的索引。
- `CatalogManager::GetIndex(const std::string &table_name,const string &index_name,IndexInfo *&index_info)`
  - 在记录index信息的两个map里寻找需要的索引的信息并返回。
- `CatalogManager::GetTableIndexs(const std::string &table_name,vector<IndexInfo *>&indexs)`
  - 找到多个索引的信息并返回。
- `CatalogManager:DropTable(const string &table_name)`
  - 删除表的所有信息及它包含的索引的信息
- `CatalogManager:DropIndex(const string &table_name,const string &index_name)`
  - 删除某个表中的相关索引信息

  `CatalogManager:FlushCatalogMetaPage()const`
  - 更新存放catalog序列化的数据页的信息
- `CatalogManager:LoadTable(const table_id_t table_id,const page_id_t page_id)`
  - 更新catalog数据中记录表的id与之对应序列化的数据页的id的map
- `CatalogManager:LoadIndex(const index_id_t index_id,const page_id_t page_id)`
  - 更新catalog数据中记录索引的id与之对应序列化的数据页的id的map

测试：


jsc@LAPTOP-J4D78JQ7:/mnt/d/ZJU_course/minisql_github/build$ ./test/catalog_test
I20220626 10:20:57.241744 322 main_test.cpp:10] This is an info log!
W20220626 10:20:57.242774 322 main_test.cpp:11] This is a warning log!
E20220626 10:20:57.243947 322 main_test.cpp:12] This is an error log!
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from CatalogTest
[RUN] CatalogTest.CatalogMetaTest
[OK] CatalogTest.CatalogMetaTest (0 ms)
[RUN] CatalogTest.CatalogTableTest
[OK] CatalogTest.CatalogTableTest (18 ms)
[RUN] CatalogTest.CatalogIndexTest
[OK] CatalogTest.CatalogIndexTest (12 ms)
[----------] 3 tests from CatalogTest (39 ms total)
[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (46 ms total)
[PASSED] 3 tests.
jsc@LAPTOP-J4D78JQ7:/mnt/d/ZJU_course/minisql_github/build$