4 CATALOG MANAGER

4.1 实验概述

Catalog Manager 负责管理和维护数据库的所有模式信息，包括：

- 数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。
- 表中每个字段的定义信息，包括字段类型、是否唯一等。
- 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

这些模式信息在被创建、修改和删除后还应被持久化到数据库文件中。此外，Catalog Manager还需要为上层的执行器Executor提供公共接口以供执行器获取目录信息并生成执行计划。

4.2 目录元信息

数据库中定义的表和索引在内存中以`TableInfo`和`IndexInfo`的形式表现，它们分别定义于`src/include/catalog/table.h`和`src/include/catalog/indexes.h`，其维护了与之对应的表或索引的元信息和操作对象。以`IndexInfo`为例，它包含了这个索引定义时的元信息`meta_data_`，该索引对应的表信息`table_info_`，该索引的模式信息`key_schema_`和索引操作对象`index_`。除元信息`meta_data_`外，其它的信息（如`key_schema_`、`table_info_`等）都是通过反序列化后的元信息生成的。也就是说，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文件中恢复，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。它们与`TableInfo`和`IndexInfo`定义在相同文件中。在序列化时，为了简便处理，我们为每一个表和索引都分配一个单独的数据页用于存储序列化数据。因此，在这样的设计下，我们同样需要一个数据页和数据对象`CatalogMeta`（定义在`src/include/catalog/catalog.h`）来记录和管理这些表和索引的元信息被存储在哪个数据页中。`CatalogMeta`的信息将会被序列化到数据库文件的第`CATALOG_META_PAGE_ID`号数据页中（逻辑意义上），`CATALOG_META_PAGE_ID`默认值为0。

在本节中，你需要了解与Catalog相关的元信息的序列化和反序列化操作，并实现GetSerializedSize()函数：

- `CatalogMeta::GetSerializedSize()`
- `IndexMetadata::GetSerializedSize()`
- `TableMetadata::GetSerializedSize()`
- `IndexInfo::Init(*index_meta_data, *table_info, *buffer_pool_manager)`: 传入事先创建好的`IndexMetadata`和从`CatalogManager`中获取到的`TableInfo`，创建索引本身的`key_schema_`和`Index`对象。这里的`key_schema_`可以通过`Schema::ShallowCopySchema`来创建，且`key_schema_`中包含的列与`TableSchema`中的列共享同一份存储。

4.3 评论

陆天2022-05-12 15:08

instance.h 文件的 DBStorageEngine 的构造器中，当第一次进行初始化存储引擎（init设为true）时，默认会创建空白的CATALOG META PAGE，并将缓存池中的Catalog meta页进行unpin且将dirty标为false，这是否可以理解为：要求当CatlogMeta内的两个map为空时CatlogMeta的序列化结果必须为空？我认为这是不对的，因为：

1这与自主实现序列化方法的理念相悖
2这与CATALOG_METADATA_MAGIC_NUM的存在相矛盾
3存在导致数据不一致错误的可能，考虑初始化DBStorageEngine 后立刻关闭的情况

YingChengJun2022-05-12 20:50

“要求当CatlogMeta内的两个map为空时CatlogMeta的序列化结果必须为空”不是这么理解的，我在我们班的实验课上是提到过就算是空的也要写入一些标记（比如size=0）。不过这里的设计确实没有考虑到初始化DB后立即关闭这一场景。一个可行的修改方案是，为CatalogManager添加InitCatalogMetaPage的成员函数，然后在DBStorageEngine Init的时候调用它，把CatalogMeta序列化进去（这时序列化了一个MAGIC_NUM以及两个map的size）。

实验设计得比较仓促，难免存在错误或者考虑欠妥的地方，感谢提出的反馈！

陆天2022-05-13 15:35

CatalogMeta::GetNextTableId 以及 CatalogMeta::GetNextIndexId 存在问题，详见

https://git.zju.edu.cn/zjucsdb/minisql/-/merge_requests/1

YingChengJun2022-05-13 20:49

看了一下我这边的实现是，先在初始化Catalog时加载到内存中（内存中用的原子变量），然后需要获取新的ID的时候再通过FetchAdd加1，所以在实现上产生了一些出入。
MR中的实现也是合理的，已合并。

愚者千虑2022-05-29 22:02

请问在GetNextTableId()方法中，不需要考虑旧的Table被drop的情况吗？还是说drop掉的Table不占用资源，直接采用更靠后的TableId也不会有浪费？

GetNextIndexId()同理

YingChengJun2022-05-29 22:13

TableID的空间很大，不考虑复用旧的TableID