《数据库系统概论》作业报告

计02 刘明道 2020011156
计04 高焕昂 2020010951

Course project for *Introduction to Database Management System (2022 Fall)*.

1 使用说明

1.1 运行环境

- Linux
- C++ 20
- CMake >= 3.22

1.2 运行步骤

- 检查 `./utils/setup_env.sh`，安装相应依赖
- 使用以下命令构建项目

```
cd cmake-build-debug && make main
```

2 完成功能一览

- **基本功能**：基本运行、系统管理、查询解析、完整性约束、模式管理、索引模块
- **扩展功能**：多表 JOIN、高级查询、扩展数据类型、高级索引：UNIQUE 约束、高级完整性约束：NULL、联合主外键、联合索引

3 实验内容

3.1 文件管理

和文件管理系统有关的代码主要位于 `src/io` 文件夹下。

- `FileSystem` 类是一个页式文件管理系统，对标准库函数提供的文件输入输出函数进行了一次封装，对外提供了按页为粒度对文件进行读写的接口。
- `Page` 类表示实际的一页数据，提供了额外的标志位标志其是否为脏页，同时提供了 `Lock()` 函数和 `Release()` 函数为上层缓存系统在换页时标记本页是否可以被交换。
- `BufferSystem` 类是一个基于 LRU 算法的缓存系统，这也是数据库系统进行文件读写的接口。缓存系统将部分页缓存在内存中，并使用 LRU 替换算法在适当的时机调用文件管理系统提供的函数将被替换页写入硬盘。

3.2 记录管理

和记录管理系统有关的代码主要位于 `src/record` 文件夹下。

- `DataPage` 是对 `Page` 类的一次封装，其页面布局遵循以下规则，可以同时支持定长记录和变长记录，并对外开放了查看剩余空间和根据槽号插入记录、更新记录与删除记录的接口。删除后，我们默认将尾指针置为 -1。

```
/** DataPage Layout
 * | PageHeader |
 * | Slot 0 | Slot 1 | ... | Slot <slot_count - 1> |
 * | Free Space | ...
 * | Pointer to free space | Pointer to <Slot <slot_count - 1>> | ... |
Pointer to <Slot 0> |
 */
```

- `FreeSpaceManager` 类将一张表的各个数据页面的剩余空间组织起来，在面对新记录作为输入，需要申请分配一定的空间的时候，可以将具有对应空间的页面快速地查询出来。具体来说，我们采用了文档中提出的链表实现，将 4096 Byte 的页面划分成 256 个不同的状态，然后使用 (页号 → 状态)，(状态 → 页号列表) 两种映射来辅助查询与修改工作。
- `Field` 类定义了表列的类别，包括 `Int`，`Float`，`Char`，`VarChar` 与 `Date` 这几种。同时，其中定义的 `PrimaryKey`，`ForeignKey`，`UniqueKey`，`IndexKey` 类分别定义了主键、外键、唯一性约束与索引的结构。
- `Record` 类是对一行记录的定义，其由 `Field` 的向量组成。在序列化与反序列化时，我们使用 `NullBitmap` 来记录某一列是否为空。
- `TableMeta` 类是对一张表元信息的表述，其 Layout 如下所示：

```
/**
 * TableMeta Layout
 * Page 0: Fixed info (ensured that will not exceed one page)
 * Page 1 ~ Page n: Foreign Keys (each foreign key has fixed length)
 * Page (n + 1) ~ Page m: Index Keys
 * Page (m + 1) ~ Page k: FieldMetas
 * Page (k + 1) ~ Page ([total_page_count] - 1): FSM Pages
 */
```

3.3 索引管理

和索引管理系统有关的代码主要位于 `src/index` 文件夹下。索引管理系统和记录管理系统很相似，下面做具体介绍：

- `IndexPage` 类是对 `Page` 类的一次封装，其页面布局遵循以下规则，注意到内部节点页和叶节点页对应的布局结构不同。对外其表现类似于一个向量，提供了根据槽号拿出记录，插入记录，更新记录，删除记录，切片选取记录，范围插入与范围删除记录的接口。

```
/*
 ** Index Page Layout (Leaf)
 * | PageHeader |
 * | IndexRecord <0> | IndexRecord <1> | ... | IndexRecord <n-1> |
 * where IndexRecord <i> == (PageID <i>, SlotID <i>, Key <i>)
 *
 * Index Page Layout (Internal)
 * | PageHeader |
 * | IndexRecord <0> | IndexRecord <1> | ... | IndexRecord <n-1> |
 * where IndexRecord <i> == (PageID <i>, Key <i>)
 *
 * For every internal node <j>, we ensure that all elements in the subtree
rooted at <j>
 * are less than or equal to Key <j>.
 *
 * The methods of IndexPage act like a vector. ()
 */
```

- `IndexField` 类是对可以被索引的字段相应的索引记录的描述，目前支持 `IndexINT`，`IndexINT2` 两种类型，分别对应记录类型的 `Int` 型与两列 `Int` 型的联合索引。其为内部节点与叶结点提供了两套 `FromSrc` 方法与 `Write` 方法，这是因为在联合索引的实现中，内部节点只需存储第一个列的关键码，而叶结点需要以第一列为主排序条件，第二列为次排序条件进行排序。
- `IndexFile` 类整体将 `IndexPage` 类的对象作为结点，将其组织成一颗 B+ 树的形式。对外其提供了索引查找的接口，以 (叶结点页号, 槽号) 的二元组作为迭代器提供沿下侧链遍历所有索引叶结点页的方法。具体来说，其包括记录的读取、插入、删除、范围删除与更新。在内部节点间查询时我们使用二分查找的方法。
- `IndexRecord` 类是索引记录项，同样分为针对内部节点的索引记录和针对叶结点的索引记录。其提供了从 `Record` 类对象转换的方法。
- `IndexMeta` 类是对某个索引系统的元信息的定义，包括 B+ 树的阶数，目前索引系统的页数，根页面页号，索引类型等等。这里我们在创建索引系统时，默认采用的阶数为最大程度利用叶结点页空间的阶数值。

3.4 查询处理

和查询处理系统有关的代码主要位于 `src/node` 文件夹下与 `src/system/WhereConditions.h` 中，我们构造的查询计划树的可能结点类型如下：

- `TrivialScanNode` 和 `IndexScanNode` 两种结点用于扫描任务，其中前者按照数据页的存储顺序进行扫描筛选，后者按照索引进行扫描筛选并在条件合适时提前终止。
- `NestedJoinNode` 和 `HashJoinNode` 两种表合并结点，分别对应文档中提到的嵌套循环连接与哈希链接两种表连接方法。
- `ProjectNode` 计算结点用于对选取出的列进行投影。
- `AggregateNode` 计算结点用于实现查询算法的聚合查询、分组查询功能。
- `OffsetLimitNode` 计算结点可以实现查询算法的 OFFSET 与 LIMIT 功能。
- `LikeCondition` 与 `InSubqueryCondition` 用于实现模糊查询与嵌套查询。

3.5 解析器

语法解析使用实验框架提供的默认语法文件 `src/grammar/SQL.g4`，使用 antlr 库进行解析，之后生成具有 Vistor 模式的文件。我们继承了自动生成的 Visitor 方法，编写了 `src/system/DBVistor.h`，以实现语法解析的功能。

3.6 系统管理

我们系统管理的有关代码位于 `src/system/DBSystem.h` 与 `src/system/DBSystem.cpp`，这是程序逻辑的主要部分。这里支持了创建、删除、调整当前正在使用的数据库的相关接口，同时对解析器解析出的其他命令进行了简单的一步封装。

3.7 命令行交互前端

```
(base) ubuntu@ubuntu-Lenovo-Legion-Y9000K2021H [11时42分44秒] [/home/c7w/dbs-tut
orial/cmake-build-debug/bin] [master]
-> % ./front
[dbs-tutorial] project by lambda & c7w
Usage: Enter to submit, Ctrl-D to exit
       Multi-line editing and history are supported
sql> CREATE DATABASE abc;                                          1,21  ]
Database created
(0.00 sec)

sql> USE abc;                                                      1,9   ]
Database changed
(0.00 sec)

sql> CREATE TABLE a1;                                              1,17  ]
line 1:15 mismatched input ';' expecting '('
2023-Jan-06 11:43:18.612314 [error] Syntax Error
Syntax Error
sql>                                                               1,1   ]
```

交互前端我们使用了 cpp-terminal 库。其支持历史记录回溯，查看当前行号列号等功能。用户可以在这里输入支持的 SQL 语句，然后交由 `DBVisitor` 中的对应函数进行解析。

4 测试程序

我们编写了若干测试程序位于 `tests` 文件夹下。其中在 `tests/python/` 下我们可以创建 Python 文件生成相应的测例 SQL；在 `tests/python_utils` 下我们提供了连接 MySQL 服务器的客户端，方便与我们数据库的输出结果进行对拍； 在 `tests/sql` 文件夹下是我们生成的测例 SQL 语句代码。此外，我们还支持使用 cpp 文件进行测试的形式。

5 其它

5.1 小组成员分工

- 刘明道：系统与文件管理、记录管理、查询解析
- 高焕昂：索引管理、记录管理

5.2 参考资料

- 实验文档
- CPP-Terminal
- 除上述资料外，未参考任何其他资料或开源代码