数据库系统概论项目报告

潘子睿　　余任杰
2020010960　　2020010966

2022 年 12 月 9 日

目录

1 项目架构

SimDB是一个简单的关系型数据库系统，其能够支持一些基本的SQL语句，完成对数据库的插入、删除、更新、查找等操作。项目的最底层是一个**页式文件管理系统**，在其上构建了**记录管理**和**索引管理**两个模块，分别用来维护数据库中的一条条记录以及在某些记录上建立的索引。 **系统管理**模块通过调用记录管理和索引管理两个模块的接口，实现了具体操作数据库的各项功能。最后，在**查询解析**模块中通过Antlr4解析SQL指令，并将其交给系统管理模块执行。

本项目源代码目前位于https://github.com/pzrain/SimDB，并可能会在本课程结课后开源。

1.1 支持功能

1.1.1 数据结构

数据库当前支持的数据结构包括：

- 整型（INT）
- 浮点型（FLOAT）
- 字符串型（VARCHAR）

其中VARCHAR仅支持定长字符串。

1.1.2 SQL语句

1.1.3 其他

1.2 代码测试

2 编译与运行

本项目基于CMake进行自动构建。需要Antlr4依赖，以及编译器支持C++17特性。使用时，将CMakeLists.txt中的ANTLR4_RUNTIME_DIRECTORY的值设置为Antlr4运行时库antlr4-runtime.h所在的目录，并将ANTLR4_RUNTIME_SHARED_LIBRARY的值设置为Antlr4运行时动态链接库libantlr4-runtime.so所在位置。
成功配置好Antlr4后，在项目根目录下执行以下命令：

```
./run.sh -c
```

即可自动进行编译并运行，生成的可执行文件SimDB位于在./bin目录下。

3 系统功能

3.1 页式文件系统

本部分代码位于./src/filesystem，需要注意的是，本部分直接使用了课程实验文档附录中提供的参考实现代码（对其中部分接口做了略微修改）。

数据库是被设计用来存储大量数据的系统，数据库中一个文件的大小甚至可能超过计算机的内存。因此，需要一个**页式文件管理系统**来管理数据库的各个文件，以及一个**缓存**机制，将操作的多个页面缓存在内存中，只在需要时进行替换和写回，以提高读写的效率。参考实现中使用的替换算法为**最近最少使用算法**（LRU）。

3.2 记录管理

记录管理模块是整个数据库系统中相对比较底层的模块之一。其负责管理存入数据库的一条条记录，具体而言，记录管理模块需要支持的功能包括：将某一记录存放在某一文件的特定位置处，并维护好该记录的位置信息、是否为空的标记等；根据指定的位置，从存放数据的文件中取出指定的记录；将某一指定的记录从文件中删除；修改文件中某一指定位置的记录，实际上就是先读取记录，修改后再写回到原来的位置。

同时，本模块还负责维护一张表的结构，具体而言，需要支持的功能包括：增加一张表；删除一张表；给原有的表增加/删除一个表项；修改原有的表中的一个表项等。为了实现以上的所有功能，记录管理模块会调用页式文件系统中定义的各个接口，从而完成实际的文件I/O操作。

在本模块的具体实现中，一个数据库以文件夹的形式存储，该数据库下的一张表以单个文件的形式存放在对应文件夹下。在处理某一文件中的多个内存页时，本模块将其分为两类，分别为表头页和记录页。前者用来存放一张表以及其对应文件的**元数据**，包括表的列数，每列的具体要求，以及该文件的总页数，第一个有空闲位置的页等信息。后者则用来存放对应表下的具体记录。

3.2.1 记录结构

一条记录包含多个表项的具体内容。记录实际上存在两种结构，分别为**序列化**和**反序列化**后的结果，实际存储在文件中的记录是序列化后的结果。

序列化 序列化后的结果为字节的序列，也即将记录中的各个表项拼接在一起，组成一个char数组。该字符数组的前两位字节被用来判断记录中各个表项是否为空值（Null）。

反序列化 反序列化即为将字节的序列整理成为更易操作的格式，实际过程中对记录的修改都是基于反序列化的形式。反序列化会将字节数组的各个表项提取出来，构造成一个链表。

记录位置 每条记录的位置被维护成一个**二元组**(pageId，slotId)，表示该记录被存放在第pageId页上的第slotId槽中。

3.2.2 表头页结构

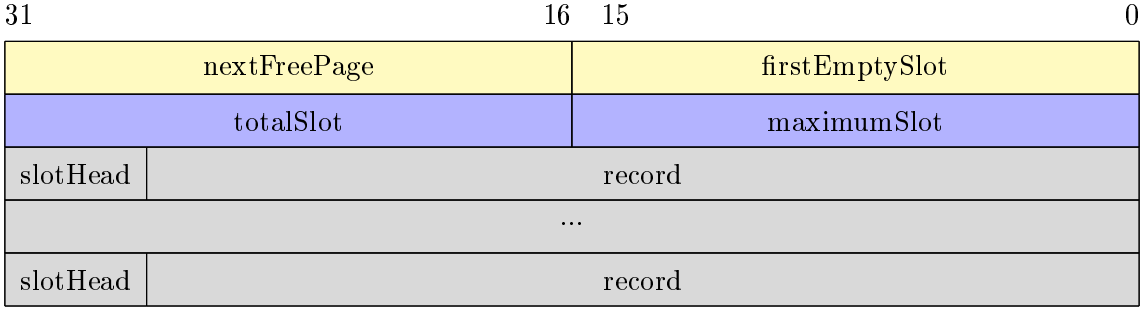
每个文件对应的第一个内存页被处理成**表头页**。表头页中存储的各个字段如下

名称	占用字节数	描述
valid	1	该页是否已经初始化
colNum	1	表中的项数
entryHead	1	第一项在entrys中对应的下标
firstNotFullPage	2	第一个非满页的页码
recordLen	2	表中定长记录的长度
totalPageNumber	2	当前总页数
recordSize	4	一页上所能存放的记录数
recordNum	4	总记录数
entrys[TAB_MAX_COL_NUM]	-	各表项的具体描述
tableName[TAB_MAX_NAME_LEN]	-	各表项的名称

其中，每个表项使用一个类TableEntry来描述，表中的entrys即为TableEntry的数组。TableEntry类的具体实现，参见附录中的各模块接口详细说明。

3.2.3 记录页结构

文件对应的各个内存页中，除了第一个为表头页外，其余均为**记录页**。记录页的结构组织如下：



记录页中的nextFreePage，以及表头页中的firstNotFullPage，将所有已分配的空闲页串成了一张链表。这样，在插入一条记录时，可以迅速找到有空闲位置的页；删除记录时，如果该页上的记录已经被删完了，就将其添加到空闲页链表的尾部。 totalSlot记录的为当前页面内记录的总数，如果其达到了maximumSlot，说明页面已被填满，需要将其从空闲页链表中去除。对于非空闲页，该域中的内容可以是任意值。

如上图所示，记录页中的前8个字节用于记录和空闲页管理以及页内槽数相关的信息，剩下的空间中会**紧密**排列着各条记录。由于记录是定长的，因此一旦记录的长度确定，就可以计算出页面上最多能存放的记录总数，以及每条记录存放位置的偏移。与空闲页的管理

类似，空闲槽也被组织成链表，对任一个空闲槽，其对应的下一个空闲槽的编号被记录在`slotHead`中。链表尾部空闲槽该域的值为`-1`。而对于非空闲（已经写入了记录的）的槽，其`slotHead`域会被记录为`SLOT_DIRTY`以进行区分。

3.3 索引管理

3.4 系统管理

3.5 查询解析

4 项目分工

- 记录管理、索引管理：潘子睿
- 系统管理、查询解析：余任杰

5 参考资料

以下列出了本项目完成过程中所参考的部分资料，包括课程的实验文档、往届的实现，以及相关网站等。

- 往届实现的数据库项目
- CS346 Redbase project
- B^+树的定义与实现
- 使用`Antlr4`完成`SQL`的语法解析
- 课程实验文档