

数据库系统概论

课程项目报告

王宣润 2016010676

2020.01

1 总述

本次大作业我实现了一个完成了所有基础功能的简单数据库系统。它有以下功能:

- 支持database的增删查操作
- 支持对数据表表的变动(alter系列语句)
- 支持数据表行(Entry)的增删改查
- 完善的主键外键约束和数据完整性约束
- 支持int,float,varchar,date四种基础数据类型

```
>> select * from relcat;
```

relName	recordSize	attrCount	indexCount	primaryCount
relcat	60	5	0	0
attrcat	140	10	0	0
fkcat	396	10	0	0
idxcat	129	3	0	0
part	166	9	0	1
region	183	3	0	1
nation	187	4	0	1
supplier	197	7	0	1
customer	224	8	0	1
partsupp	216	5	0	2
orders	146	9	0	1
lineitem	166	16	0	2

```
Total = 12 entries.  
>> _
```

Figure 1: 数据库运行截图

该项目开源在我的github(https://github.com/WXR1998/DBMS_Basic)上。

2 系统架构设计

遵循官方结构设计, 该DBMS分为四个大的部分: 记录管理模块、索引管理模块、系统管理模块、以及查询解析模块。四个系统分别在代码目录中存放在record、index、system、parser目录下。但由于代码编写后期逻辑开始变得有些混乱, 这些功能可能划分得并不是那么清晰。

3 各模块详细设计

3.1 记录管理

本模块旨在建立一个从Record到数据表Entry的桥梁, 提供若干数据格式转换接口和协议实现的编码。每个文件详细的情况说明如下表所示。

文件名	功能和方法
FileScan	文件扫描器。开启一个Record扫描，逐个扫描某个文件里的所有记录，返回下一个满足条件的
FileHandler	文件操作主句柄，利用单例模式，给上层系统提供了一个操作文件的接口，创建、修改文件的时候会根据定义的数据格式协议维护记录的完整性
FirstPageInfo	维护文件第一页的元信息
PageInfo	维护文件第二页到最后一页的信息、直接操作下层的缓存页式文件系统
Record	记录类
RecordID	RID类
SingleFileHandler	对于单个文件的操作句柄，依附于单个文件，给FileHandler类提供接口支持

每个数据表都是一个文件，每个文件都保存了两种页：首页和内容页。首页保存了有关这个数据表的元信息，如文件中有多少个Record，文件有多少页、这个文件中Record的长度等。为了实现方便和提高效率，每个文件保存的Record的长度都是确定且不可改变的。后面紧跟的内容页每页分为3个部分：页头、槽(Slot)Bitmap、数据槽。一旦一条Record的长度确定，一页能分成多少个槽也就确定了。页头存放该页的相关元数据，Bitmap记录了每个槽的占用情况，每个槽都可以填入一条定长Record。这种实现方式实际上就否决掉了Varchar变长字符串的可能性：因为变长的字符串势必会导致变长Record，这样就无法用这种方式来保存文件。

3.2 索引管理

索引管理模块通过对给定数据表的某一列的所有键值建立一棵B+树来加速查询。由于索引本质上也需要用Record保存到文件中，所以在我实现的B+树中，用每个B+树节点在索引文件中的RID来唯一地指代这个节点。每个节点对应的Record需要保存 K 对键的值(其中 K 是B+树的阶数)：用于比较的键值和在索引文件中的孩子节点RID。B+树的叶子节点没有孩子节点，所以将它们的孩子节点空间用于指代数据表文件中该条Record对应的RID。这样无论是叶子节点还是非叶子节点，保存它们的Record长度是相同的，便于我们运用记录管理模块进行保存。

另外为了简化设计起见，我实现的数据库系统只支持单列索引，对于复合索引的输入命令我将直接忽略。这个模块的文件详细情况说明如下表所示。

文件名	功能和方法
BNode	B+树节点类，还负责把B+树节点转化成Record
IndexHandler	索引操作主句柄，利用单例模式，给上层系统提供了一个操作索引的接口，索引增删、增删查改Entry的时候会调用
SingleIndexHandler	对于单个索引的操作句柄，依附于单个表的单个Attribute，给IndexHandler提供接口支持

3.3 系统管理

系统管理是本数据库管理系统最复杂的一个模块，它负责给所有的操作编码，赋予其语义信息。为维护整个数据库系统，我维护了几张系统表。系统表里记载了这个数据库系统的下列关键信息：

- **relcat** table表，记录了当前database里存放了哪些table(relation)，还记载了它们是否有主键、是否存在索引
- **attrcat** attribute表，记录了当前database里每个table的每个attribute的元信息，如类型、占用字节数、Record中的偏移等
- **idxcat** 索引表，记录了当前database每个索引的名称和依附到哪个table的哪个relation

- **fkcat** 外键表，记录了当前database每个外键的名称和依附到的主表、从表及其对应的attributes。注意，为了实现的方便，我的数据库系统规定联合外键的attribute数量不能超过3个

每个数据库是一个文件夹，数据库里每个table是一个文件，每个索引也是一个单独的文件。下面逐个介绍重要语句需要执行的操作。

3.3.1 SHOW DATABASES

遍历存放数据库的文件夹，找出所有目录的名字，剔除掉系统目录之后输出。

3.3.2 CREATE DATABASE

建立文件夹，建立新的系统表。

3.3.3 SHOW TABLES

遍历系统表relcat并输出结果。

3.3.4 CREATE TABLE

维护relcat, attrcat，建立对应文件，填充元信息。

3.3.5 DROP TABLE

检查数据完整性，维护relcat, attrcat，删除对应文件。

3.3.6 DESC TABLE

遍历系统表attrcat，找到当前table对应的行并输出。

3.3.7 INSERT INTO TABLE VALUES ...

外键检查主表是否有对应项，主键检查是否重复，插入Record，插入索引项（若有）。

3.3.8 DELETE FROM TABLE WHERE ...

通过where子句筛选要删除的Entry，外键检查是否被指向，删除Record，删除索引项（若有）。

3.3.9 UPDATE TABLE SET ...

由于维护数据完整性过于复杂，我没有做where子句。检查set子句是否会更改主键键值并破坏主键数据完整性，检查set子句是否会更改外键键值并破坏指向关系，依次修改每一行的这一个attr的值并保存，修改索引（若有）。

3.3.10 SELECT SLCTR FROM TABLES WHERE ...

先通过where子句中常量比较子句初筛一遍结果（若有索引则使用，否则暴力查找）。然后再用SLCTR筛出需要的列。支持多表查询的笛卡尔积。

3.3.11 CREATE INDEX

维护idxcat，遍历表的每一个Entry建立B+树。

3.3.12 ALTER TABLE ADD ...

将表中所有Entry提取到内存，删掉所有索引，删掉这个表，加入一个新的表，其中新表比旧表多一列，再将所有Entry对应的列置NULL或默认值（若有），依次插回这些Entry，检查主键完整性（若有），加回所有索引。这里不能保留索引的原因是重新插入可能会使得RID变化。

3.3.13 ALTER TABLE DROP ...

检查是否是主键，检查是否是外键，将表中所有Entry提取到内存，删掉所有索引，删掉这个表，加入一个新的表，其中新表比旧表少一列，将所有Entry对应的列删去，依次插回这些Entry，加回所有索引。

3.3.14 ALTER TABLE CHANGE ...

相当于调用ALTER TABLE DROP再调用ALTER TABLE ADD。

3.3.15 ALTER TABLE RENAME ...

检查重名，重命名文件，重命名索引名，在系统表里查找原名并全部改为新名。

3.3.16 ALTER TABLE ADD PRIMARY KEY

检查是否有重复值，这里我的方法是这样的：首先取出所有Record，按照主键的键值排序，然后检查相邻的两个Record是否在主键上相同。维护系统表relcat和attrcat。

3.3.17 ALTER TABLE DROP PRIMARY KEY

检查是否是其他表的外键，维护系统表relcat和attrcat。

3.3.18 ALTER TABLE ADD FOREIGN KEY

检查重名，检查主表项是否是主表的主键，逐Entry检查是否满足外键数据完整性，维护系统表fkcat。

3.3.19 ALTER TABLE DROP FOREIGN KEY

维护系统表fkcat。

这个模块的文件详细情况说明如下表所示。

文件名	功能和方法
DBHandle	数据库操作句柄，响应数据库操作语句
Printer	打印数据库查询结果，其中我对格式做了很多特殊处理，如列宽设置等，其显示效果和MySQL相似，见图1
RecordDescriptor	可以存到文件的二进制数据 到 可以按类读取的对象之间 的转换器
SystemManager	系统操作主句柄，所有上述操作都来自这个文件

3.4 查询解析模块

使用YACC和Flex工具进行词法/语法分析。参照编译原理PA1，在这个模块里我只需编写parser.l和parser.y，YACC和Flex就会自动生成cpp文件。具体实现可以参考上述这两个文件。另外抽象语法树遍历的过程都定义在Tree中。

4 主要接口说明

请参见3章节。

5 实验结果

由于没有实现字节层面上的copy，导致导入效率很低，在当场测试时我只在小规模数据集上运行。取得了良好的结果，通过了所有基础要求的测试。

在项目根目录下运行

```
>> bash run.sh all
```

即可自动编译并运行数据库。

6 小组分工

本小组是单人小组，所有工作都由我一人完成。

7 感想

整个数据库一共8500行，这应该是我本科做过的最大的项目了。一开始我还对自己能否完成这个巨大的项目没有信心，曾经一度想要退课，但还好坚持下来了。最后爆肝了一个多星期终于完成了全部的基础要求内容。写完这个项目，我对数据库的运作机制、各种特殊情况的判断有了更深刻的理解，同时也体会到要开发一个完整的、鲁棒性强的数据库系统需要耗费多少人力。但由于我代码写得过于随心所欲，基本没怎么考虑运行效率问题，所以没能在大规模数据集上跑，是一个小小的遗憾了。

8 参考文献和工程

1. 马也 的数据库工程 <https://git.net9.org/maye9999/database>
2. 杜政晓 的数据库工程 <https://github.com/duzx16/MyDB>
3. 数据库大作业详细说明
4. parser语法规则