|  |  |  |
| --- | --- | --- |
| 2021春《数据库系统》实验报告 | | |
| 实验3：查询执行器 | | |
| 姓名： 卢兑玧 | 学号：L170300901 | 班级：1803501 |

1. 实验目的

在实验2实现的BadgerDB缓冲池管理器(buffer pool manager)的基础上，本次实验继续实现BadgerDB 的存储管理器(storage manager)和查询执⾏器(query executor)，具体完成以下内容：

使⽤堆⽂件(heap file)存储关系，实现向关系中插⼊(insert)元组和删除(delete)元组的功能。

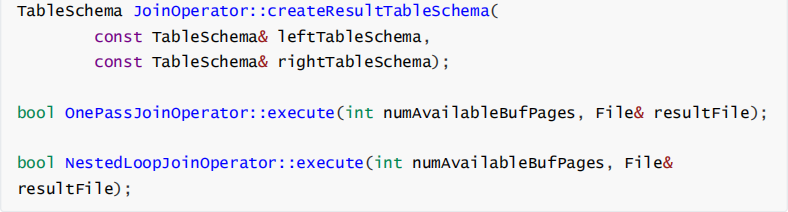
实现⾃然连接(natural join)操作算法，对两个关系进⾏⾃然连接，具体实现⼀趟连接(One-Pass Join)、基于块的嵌套循环连接(Block-based Nested Loop Join)、Grace哈希连接(Grace Hash Join) 等算法。Grace哈希连接算法的实现为选作。

2. 实验准备

见实验指导书。根据实验指导书来完成。

3. 实验内容

在本次实验中，你需要实现TableSchema, HeapFileManager, OnePassJoinExecutor, NestedLoopJoinOperator类中的若⼲⽅法，完成BadgerDB的存储管理和查询执⾏功能。具体需要实现的⽅法如下：

**schema .cpp**

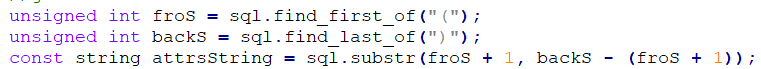
（1）

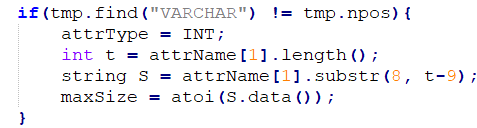
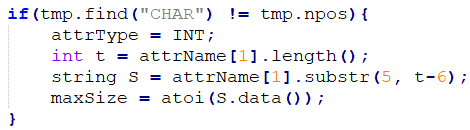
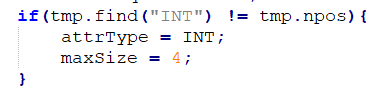
首先在schema.h中添加一个split函数的声明，用来按照指定字符切割字符串：返回vector类型变量：



然后再在schema.cpp中实现split，输入要切割的字符串cutting和切割字符sign，遍历cutting，如果碰到sign，则在vector遍历中加入到此符号之前一位的子串，最后得到分割结果。

1. 

首先将输入的SQL语句按照空格“ ”分割，则table的名字即为分割后得到的vector变量的第三个值，然后如果名字中包含#着##，则为临时表。之后找到前后小括号的位置，得到属性部分：

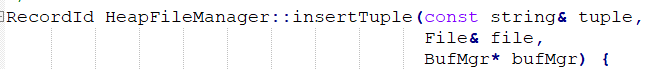
将属性按照“,”分割，即可得到各个属性值的定义，遍历属性vector对所有属性值进行如下操作：按照空格分割，vector中第一个即为属性名字。再使用find找到属性的类型，并分类进行最大值的获取

然后再用find看属性是否是NULL、UNIQUE的即可。

1. 

输出表名、是否为临时表、属性总数量、属性名字、类型、最大值、是否为NULL，是否UNIQUE。

**storage.cpp**

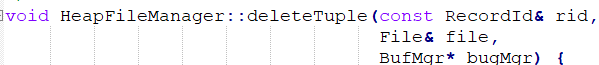
1. 

要插入元组，就要知道可用的recordid。

首先，运用文件的迭代器遍历文件的页

在遍历中，首先根据迭代器得到遍历到的页的id，然后运用缓冲池的readpage方法得到页对象。如果空间充足，则运用insertRecord方法得到插入的recordid，将缓冲池的此页设置为dirty。

一旦发现页号为无效值或者无充足空间，则跳出遍历从缓冲池申请页来插入元组，并根据页的insertRecord方法得到recordid，再将缓冲池的此页设置为dirty即可。

1. 

删除元组，已知recordid，则找到所在页即可，且recordid包含page\_number，所以直接运用缓冲池的readpage方法得到页对象，运用页的deleteRecord方法即可删除元组

1. 

由于实验规定元组传递时传递的不是字符串，而是字节序列，所以首先设定一个将元组字符串转换成字节序列的方法。arrInfo参数表示insert的属性值，len表示属性值的maxsize。

声明一个字符串流类型的参数ans，然后循环遍历长度len（i），内套循环遍历长度char类型的长度\*8（j），表示bit数如果arrInfo的第i位不是‘\0’，那么则将它与0x80作位与操作，0x80为10000000，每内循环一次0x80则往右移j位，目的在于逐位提取attInfo的bit值

如果到了尽头，则后面加X即可。最终返回ans的string值。

1. 

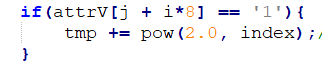
首先声明一个变量bitSequence，用来存储元组的字节序列。跟分析create语句一样，首先根据“ ”分离字符串，然后得到的vector变量的第三个值即为要插入元组的表的名字，然后根据名字得到tableid，之后将tableid写入bitSequence。然后用跟之前同样的方法获得属性值，之后将属性值根据“,”分割后得到不同属性的值。遍历不同属性的值，根据值从表模式中得到属性的maxsize，再将maxsize值加到bSeqSize变量中，然后此时用之前的BitsSequence方法获取bits序列，再加入到变量中attrsBitsSeq中，为后来加入bitSequence变量做准备。

遍历完成后，用之前的bSeqSize变量再加上64，代表元组头（id和size）的长度，将其加入到bitSequence变量中，最后再将attrsBitsSeq加入bitSequence变量即可得到最终答案。

**executor.cpp**

首先设置几个全局变量：表示进行连接的属性，即两个表中的共有属性；存储共有属性值--含有该值的元组的键值对。

1. 

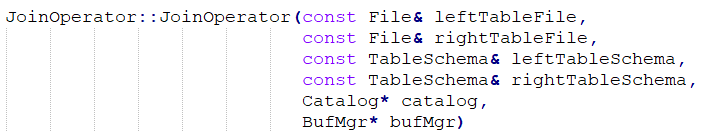
进行元组的bitSequence的翻译。首先遍历得到的bitSequence，将空位补充的X都给去掉。然后将剩下的序列遍历，遍历长度为序列长度/8：，内部嵌套一个循环，目的是从最低位开始算二进制数的值。最后再将这个值转换为字符即可。

1. 

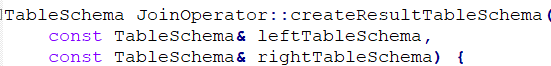
比较两个文件的页数多少。运用迭代器对L和R文件进行迭代，并设置参数累加计算页数，最终，如果L文件页数少于等于R，则返回true，否则返回false

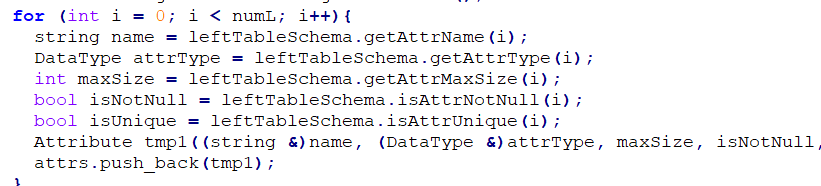
1. 

打印数据库。首先获取tableFile和缓冲池，然后刷一遍文件，再利用file的迭代器进行迭代完成数据库表的读取。获取并打印表模式的操作与schema.cpp中的print函数相同。之后根据迭代器获得pageid进行while循环，pageid等于0时跳出。根据缓冲池的readpage方法得到page的对象，获取pageid和page中的slotid再进行一个while循环，slotid等于0时跳出。根据pageid和slotid构建recordid之后通过recordid获得元组字节序列，经过翻译后遍历输出即可。

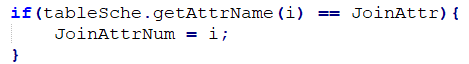
1. 

这个方法为构建连接后表模式的操作，调用createResultTableSchema即可，当leftfile更小时createResultTableSchema参数为（left，right），否则反过来即可。

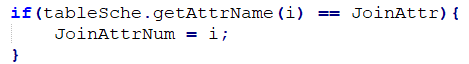
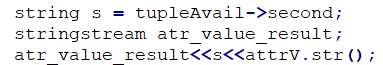
1. 

这个方法为构建连接后表模式的操作。首先根据自然连接的属性可知要先获取左右表的属性都有哪些，再从中找到共有属性，再进行连接即可。思路为先将其中一个表的全部属性放入，再遍历另一个表的属性，如果与某个已经加入的属性相等，则这个属性不再加入，且这个属性即为共有属性，把这个属性的名字付给一开始声明的全局变量。这样即可完成这个方法。最终

1. 

该方法将表模式写入缓冲池中并记录共有属性值--含有该值的元组的键值对。首先根据page对象获得pageid和起始slotid，然后获取属性总数并进行遍历，找到两个表共有属性的下标，再进行while循环，当slotid等于0时跳出。循环中首先构建recordid，然后以此获得元组的字节序列，翻译成字符串之后进行分割操作，遍历将属性值依次写入字符串流变量中，最后将共有属性值--连接元组的键值对写入全局变量JoinAttrs中即可

1. 

目的在于将输入的page与缓冲池中的数据进行连接，并返回连接表中元组的个数。已知page对象，可以根据page对象获得起始slotid和pageid，然后获取属性总数并进行遍历，找到两个表共有属性的下标再进行while循环，当slotid等于0时跳出。循环中首先构建recordid，然后以此获得元组的字节序列，翻译成字符串之后进行分割操作，遍历将属性值依次写入字符串流变量中。先用count方法看此共有属性值是否在JoinAttrs中，若在的话就继续如下操作：迭代遍历所有key值为此共有属性值的键值对，将它的值写入一stringstream变量，然后再把之前得到的页中的元组值写入，然后这里便得到了insert语句的属性值部分，再构建insert语句，运用之前实现的HeapFileManager::createTupleFromSQLStatementinsertTuple和方法即可把该元组插入新的连接表中，再把连接表中元组的个数加一即可。

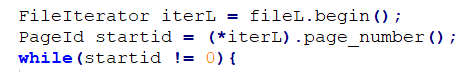
最后，再将新数据从缓冲池刷到文件中即可

1. 

首先将连接结果表模式和表名加入目录中，然后声明变量分别为左表文件、右表文件、左表模式和右表模式。然后对比大小，以进行下面的操作。为了不影响之后的操作，首先清空全局变量JoinAttrs。然后用迭代器迭代较小的表文件，遍历将该表文件所有的页都读入到缓冲池中，再将读写数加1：，然后把其写入缓冲池中并记录共有属性值--含有该值的元组的键值对。将缓冲池已使用页数加1：。之后再用迭代器遍历较大的表文件，将其中的页依次读入缓冲池，并将此页与较小表文件的每一页进行连接操作，这样即可完成一次连接操作。

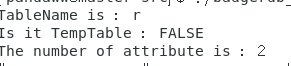
1. 

首先将连接结果表模式和表名加入目录中，然后声明变量分别为左表文件、右表文件、左表模式和右表模式。然后对比大小，以进行下面的操作。根据基于块的嵌套循环连接的原理可知，需要将较小的文件的M-1块写入缓冲池，M即为缓冲池中可用页数。为了不影响之后的操作，首先清空全局变量JoinAttrs。然后循环遍历较小的表文件，将其中的M-1个页写入缓冲池

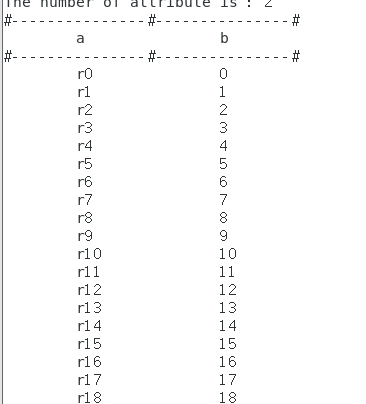
，再在这个循环内部用迭代器遍历较大的表文件，将其中的页依次读入缓冲池，并将此页与较小表文件的这一轮循环所写入的页进行连接操作，这样即可完成基于块的嵌套循环连接操作。

4. 实验结果

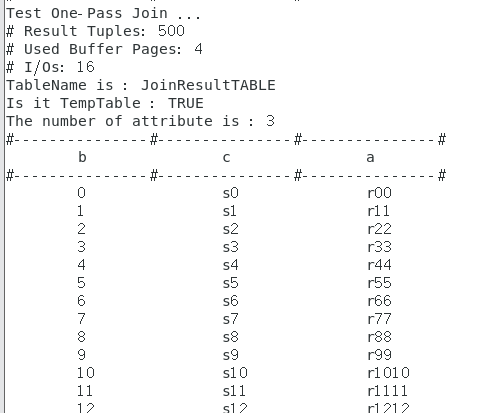
打印数据库模式：



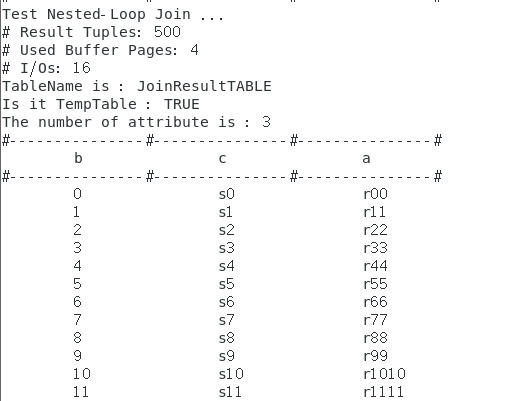
打印数据库：



一次连接结果：



嵌套连接结果：



测试结果：



5. 总结与体会

通过这次实验，我对于磁盘上数据库的操作细节有了更加清晰的认识，有助于我更深入地了解具体步骤。同时我还对几种连接方式进行了具体实现，对上课内容的理解有了更深的认识。这次实验设计的非常好。如果能再对所有需要用到的已实现方法进行介绍就更好了。