实验题目	查询执行器实现			实验日期	2022 年 5 月 1 日
班级	1903104	学号	1190201421	姓名	张瑞

CS33503 数据库系统实验

实验检查记录

实验结果的正确性(60%)		表达能力(10%)	
实验过程的规范性(10%)		实验报告(20%)	
加分(5%)		总成绩(100%)	

实验报告

一、实验目的（介绍实验目的）

1. 掌握各种关系代数操作的实现算法，特别是连接操作的实现算法。
2. 在实验 2 完成的缓冲区管理器的基础上，使用 C++面向对象程序设计方法实现查询执行器。

二、实验环境（介绍实验使用的硬件设备、软件系统、开发工具等）

1. 硬件设备：Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz; 8GB RAM
2. 软件系统：Windows 10
3. 开发工具：Vmware 16

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

1. 阅读源代码

为了完成本次实验，需要先阅读源代码，了解相关定义与实现。下面简要介绍几个对本次实验比较重要的方法。

HeapFileManager 类的 createTupleFromSQLStatement 提供了将 SQL 语句翻译为元组的方法，其将元组各个属性的值按顺序存储，并进行一个 4 字节对齐的操作，空位用 0 补齐。

HeapFileManager 类的 insertTuple 提供了将元组存入文件的方法，其将元组插入文件中有空余位置的页面中。

JoinOperator 类的 getCommonAttributes 提供了提取左关系与右关系共同属性的方法。

JoinOperator 类的 construct_search_key 提供了提取左（或右）关系中“共同属性”所对应的属性值的方法。

JoinOperator 类的 joinTuples 提供了将左关系中一个元组与右关系中另一元组进行连接，得到新的元组表示的方法。

2. 实现 NestedLoopJoinOperator 类的 execute 方法

该方法实现的是基于块的嵌套循环连接操作。假设可用内存块数为 M，则该方法的算法思路如下：

```
for 左关系 R 的每 M-1 块 do
    将这 M-1 块读入缓冲池
    for 右关系 S 的每 1 块 P do
        将 P 读入缓冲池
        for P 中的每条元组 s do
```

实验题目	查询执行器实现			实验日期	2022 年 5 月 1 日
班级	1903104	学号	1190201421	姓名	张瑞

for 缓冲池中能与 r 进行连接的元组 r do

连接 r 与 s，并将连接结果暂存

将最终得到的全部连接结果写入指定文件

上述算法的最终实现代码如下：

```
bool left_flag = true;
bool right_flag = true;
while (iter != leftTableFile.end()) {
  // 每次将左关系的M-1个块读入缓冲池
  vector<badgerdb::Page*> in_buffer;
  for (int i = 0; i < numAvailableBufPages - 1; i++) {
    badgerdb::Page page = *iter;
    badgerdb::Page* buffered_page;
    bufMgr->readPage(&leftTableFile, page.page_number(), buffered_page);
    numIOs++;
    if (left_flag) {
      numUsedBufPages++;
    }
    in_buffer.push_back(buffered_page);
    iter++;
    if (iter == leftTableFile.end()) {
      break;
    }
  }
  left_flag = false;
  // 每次将右关系的1个块读入缓冲池
  for (badgerdb::FileIterator iter2 = rightTableFile.begin(); iter2 != rightTableFile.end(); iter2++) {
    badgerdb::Page page = *iter2;
    badgerdb::Page* rpage;
    bufMgr->readPage(&rightTableFile, page.page_number(), rpage);
    numIOs++;
    if (right_flag) {
      numUsedBufPages++;
      right_flag = false;
    }
    // 遍历右关系中的每一条元组
    for (badgerdb::PageIterator piter = rpage->begin(); piter != rpage->end(); piter++) {
      string rightKey = *piter;
      string search_key_right = construct_search_key(rightKey, common_attrs, rightTableSchema);
      // 遍历查找能与右关系元组进行连接的左关系元组
      for(int i = 0; i < in_buffer.size(); i++) {
        badgerdb::Page* lpage = in_buffer[i];
        for(badgerdb::PageIterator piter2 = lpage->begin(); piter2 != lpage->end(); piter2++) {
          string leftKey = *piter2;
          string search_key_left = construct_search_key(leftKey, common_attrs, leftTableSchema);
          // 相同属性相等则进行连接
          if(search_key_left == search_key_right) {
            string result_tuple = joinTuples(leftKey, rightKey, leftTableSchema, rightTableSchema);
            resTuples.push_back(result_tuple);
          }
        }
      }
    }
  }
}

// 将连接结果写入文件
for(int i = 0; i < resTuples.size(); i++) {
  HeapFileManager::insertTuple(resTuples[i], resultFile, bufMgr);
}
```

3. 实验结果

首先运行 make 编译项目，然后运行./badgerdb_main，结果如下：

```
Test Nested-Loop Join ...
# Result Tuples: 500
# Used Buffer Pages: 3
# I/Os: 3
```

可以看到，最终的连接结果包含 500 个元组，使用了 3 个缓冲区页面，进行了 3 次 IO。

实验题目	查询执行器实现			实验日期	2022 年 5 月 1 日
班级	1903104	学号	1190201421	姓名	张瑞

四、实验结论（总结实验发现及结论）

查询执行器能执行多种操作，例如选择操作、投影操作、去重操作、聚集操作、集合操作和连接操作等。其中，连接操作又有多种执行算法，例如：一趟连接算法、嵌套循环连接算法、排序归并连接算法、哈希连接算法和索引连接算法。本次实验在实验 2 的基础上，通过 C++编程实现了 BadgerDB 查询执行器中的连接操作，并通过基于块的嵌套循环连接算法来实现。在实验的过程中，我复习了查询执行器的相关知识，对嵌套循环连接算法有了更为深刻的理解。