**The DBMS Project Report**

**数据库课程设计实验报告**

目录

[Part 1: Group Member 小组成员 3](#_Toc388966585)

[Part 2: Environment 实验环境 3](#_Toc388966586)

[Part 3: Contents 实验内容 3](#_Toc388966587)

[3.1: 不带压缩地导入数据 3](#_Toc388966588)

[3.2: 外排序和 RLE 压缩 3](#_Toc388966589)

[3.3: JOIN 3](#_Toc388966590)

[Part 4: Design Ideas 实验思路和设计 3](#_Toc388966591)

[4.1实验思路 4](#_Toc388966592)

[4.2项目结构 4](#_Toc388966593)

[Part 5: Implementation 代码结构与实现 5](#_Toc388966594)

[5.1 结构简述 5](#_Toc388966595)

[5.2 整体结构 5](#_Toc388966596)

[5.2.1 Main 5](#_Toc388966597)

[5.2.2 OperandBrain 5](#_Toc388966598)

[5.2.3 PageManager 6](#_Toc388966599)

[5.2.4 SortBrain 7](#_Toc388966600)

[5.2.5 Page 8](#_Toc388966601)

[5.3 具体实现 9](#_Toc388966602)

[5.3.1 导入 9](#_Toc388966603)

[5.3.2 查询 10](#_Toc388966604)

[5.3.3 外排序 11](#_Toc388966605)

[5.3.4 压缩 13](#_Toc388966606)

[5.3.5 JOIN 14](#_Toc388966607)

[5.4 Contant 14](#_Toc388966608)

[Part 6: Performance 实验结果与性能比较 14](#_Toc388966609)

[Part 7: Inspiration 心得与体会 15](#_Toc388966610)

[7.1起始的准备 15](#_Toc388966611)

[7.2最初的想法 15](#_Toc388966612)

[7.3遇到的问题 15](#_Toc388966613)

[7.4感想与收获 16](#_Toc388966614)

# Part 1: Group Member 小组成员

|  |  |  |
| --- | --- | --- |
| Name | Sid | Email |
| 朱新强 | 12330441 | 617394495@qq.com |
| 庄泽帆 | 12330442 | sysu.zhuang@gamil.com |
| 钟盼盼 | 12330425 | 897269925@qq.com |

# Part 2: Environment 实验环境

|  |  |
| --- | --- |
| CPU | Intel(R) Core™ i7-2760QM |
| OS | Mac OS X 10.9 Mavericks, Ubuntu 13.04 |
| Compiler | g++ |
| Language | C++（不含STL） |

# Part 3: Contents 实验内容

## 3.1: 不带压缩地导入数据

实现分页存储机制。对于每一列，将所含的定长数据如同数组一样存储在磁盘上的页中 。每一页的大小固定。存储 ORDERS 表中的O\_ORDERKEY, O\_CUSTKEY, O\_TOTALPRICE, O\_SHIPPRIORITY。存储之后，提供接口，以便于查询记录。

## 3.2: 外排序和 RLE 压缩

对数据做外排序。并在排序之后，使用行程长度编码 (Run-Length Encoding) 方式压缩数据。

## 3.3: JOIN

将 ORDERS 表和 CUSTOMER 表以 CUSTKEY 进行 JOIN 操作。

# Part 4: Design Ideas 实验思路和设计

// 将给出我们对于整个项目中各个环节、流程、实现的具体思考过程、重要数据结构、函数伪代码等；

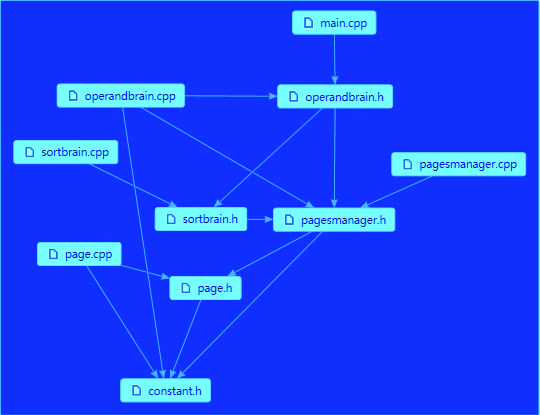
## 4.1实验思路

按照实验内容所给出的要求，C-STORE实验思路比较明确，分为以下几个步骤：

1. 导入：将数据导入并进行分页存储；
2. 外排序：先对页面内进行qsort，再对页面进行mergesort；
3. 压缩：对于排序后的CUSTKEY进行压缩，并写入文件中；
4. Join：将 ORDERS 表和 CUSTOMER 表以 CUSTKEY 进行 JOIN 操作；

## 4.2项目结构

UML结构图



# Part 5: Implementation 代码结构与实现

## 5.1 结构简述

这次整体采用面向对象的思路进行编程，对于每一个功能作为一个模块，对应建立不同的类来实现，并上下层次之间实现对接。这样的编程思想比较利于团队的协作，有便于分工，且在各自的代码出现bug需要修改的情况下，不会影响到其他人的代码的正确性。这样保证了统一性，加快了完成的效率。

## 5.2 整体结构

### 5.2.1 Main

在Main函数中接受命令行传入的字符串参数，并使用switch对接到join，load，compress等操作对应的函数实现中。

int main**(**int argc**,** const char **\*** argv**[])**

**{**

……

**switch** **(**argc**)** **{**

……

**case** 3**:**

**if** **(** strcmp**(** argv**[**1**],** LOAD\_DATA\_OPERATION\_NAME**)** **==** 0 **)** **{**

// db load filename

operandBrain**.**loadData**(** argv**[**2**]** **);**

**}** **else** **if** **(** strcmp**(**argv**[**1**],** QUERY\_OPERATION\_NAME**)** **==** 0 **)** **{**

operandBrain**.**queryInTable**(**argv**[**2**]);**

**}**

**break;**

……

**}**

**return** 0**;**

**}**

### 5.2.2 OperandBrain

与Main相对接，用于具体实现导入，查询，压缩等等操作。具体实现文件的开关，读写等。

class OperandBrain **{**

private**:**

pagesManager **\***\_pageManager**;**

SortBrain **\***\_sortBrain**;**

void externSort**(** const char**\*** tableName**,** const char**\*** column **);**

int pagesMemorySort**(** const char**\*** inputFile **);**

void mergeSortTempFiles**(** int count**,** const char**\*** outputFile **);**

public**:**

OperandBrain**();**

**~**OperandBrain**();**

void queryInTable**(** const char **\***tableName **);**

void loadData**(** const char **\***\_fileName **);**

void compressData**(** const char **\***tableName**,** const char**\*** column **);**

**};**

### 5.2.3 PageManager

用于处理一页内的操作，定义只是为了更加方便得调用page中的函数。其中包括对于页面的读写操作的函数，读写文件等函数，以及页面内排序的各种辅助函数，也包括外排序的相应的辅助函数。

class pagesManager **{**

private**:**

Page **\***pages**;**

/\*\*

\* sigletron instance

\* private constructor and destructor

\*/

static pagesManager **\***\_instance**;**

pagesManager**();**

**~**pagesManager**();**

//helper function for qsort

static int compare**(** const void **\***a**,** const void **\***b **);**

public**:**

static pagesManager **\***sharedManager**();**

//operations on one page

//just simple to call functions in Page

void writePageAtIndexToFile**(** size\_t index**,** FILE**\*** fptr **);**

void clearPageAtIndex**(** size\_t index **);**

void readFileToPageAtIndex**(** FILE **\***fptr**,** int index **);**

bool searchKeyInPageAtIndex**(** const char**\*** key**,** size\_t keySize**,** char **\***entryOutput**,** size\_t entrySize**,** int index **);**

//operations on pages

bool insertData**(** char **\***data**,** size\_t \_size**,** size\_t pageCondition **);**

int readFileToPages**(** FILE**\*** fptr**,** int begin**,** int end **);**

void clearPages**(** int begin**,** int end **);**

//extern sort helper functions

void sortPages**(** int begin**,** int end **);**

void megrePagesToFile**(** int begin**,** int end**,** FILE**\*** tempFptr **);**

void compressPagesToFile**(** FILE **\***fptr**,** int begin**,** int end **);**

**};**

### 5.2.4 SortBrain

考虑到外排序这个问题比较复杂，定义SortBrain这个类专门用于实现外排序。

class SortBrain **{**

private**:**

pagesManager **\***\_pageManager**;**

int pagesMemorySort**(** const char**\*** inputFile **);**

void mergeSortTempFiles**(** int count**,** const char**\*** outputFile **);**

public**:**

SortBrain**();**

**~**SortBrain**();**

void externSort**(** const char **\***inputFileName**,** const char**\*** outputFileName **);**

**};**

### 5.2.5 Page

定义了page这个类，集合了“处理插入数据到页存储”，“删除页中的内容”，“在页与文件中的读写操作”，“对页进行查找，压缩”等操作。在最低层实现页面内操作，在上层调用中使结构更加清晰。

class Page **{**

private**:**

char data**[**PAGE\_SIZE**];**

size\_t offSet**;**

public**:**

Page**();**

**~**Page**();**

bool insertDataToPage**(** char **\***data\_**,** size\_t size **);**

void clearPage**();**

void writePageToFile**(** FILE **\***fptr **);**

void readFileToPage**(** FILE **\***fptr **);**

bool searchInPage**(** const char **\***key**,** size\_t keySize**,** char**\*** **&** outputEntry**,** size\_t entrySize **);**

void compressToPage**(** Page **\***outputPage**,** int keySize**,** int entrySize**,** FILE **\***fptr **);**

inline bool isFull**()** **{** **return** offSet **>=** PAGE\_SIZE**;** **}**

inline bool isEmpty**()** **{** **return** offSet **<=** 0**;** **}**

inline char **\***getData**()** **{** **return** data**;** **}**

inline size\_t getOffset**()** **{** **return** offSet**;** **}**

inline void setOffset**(** int offset **)** **{** offSet **=** offset**;** **}**

**};**

## 5.3 具体实现

### 5.3.1 导入

进行一系列的文件打开和读取操作后，对指定的KEY进行导入，以下代码为对ORDERKEY的导入操作，其他KEY同。

**case** O\_ORDERKEY\_CONDITION**:**

primaryKey **=** atoi**(** currentProperity **);**

memcpy**(**tempArr**,** **&**primaryKey**,** **sizeof(**int**));**

**while** **(** **!**\_pageManager**->**insertData**((**char**\*)&**primaryKey**,** **sizeof(**int**),**pageIndexs**[**0**]** **)** **)** **{** //if insert fail

//write the page to file

\_pageManager**->**writePageAtIndexToFile**(** pageIndexs**[**0**],**orderKeyFptr **);**

//clear page

\_pageManager**->**clearPageAtIndex**(** pageIndexs**[**0**]** **);**

//re-insert

**}**

**break;**

导入到临时页面后再将页面写入文件中

void Page**::**writePageToFile**(** FILE **\***fptr **)** **{**

fwrite**(**data**,** **sizeof(**char**),** offSet**,** fptr**);**

**}**

### 5.3.2 查询

给定一个ORDERKEY，在页面中查询

**while** **(** **!**feof**(** orderKeyFptr **)** **)** **{**

//search key in page

**if** **(** **!**\_pageManager**->**searchKeyInPageAtIndex**(** **(**char**\*)&**orderKey**,** **sizeof(**int**),** **(**char**\*)&**orderKey**,** 0**,** 0 **)** **)** **{**

// if not found, read next page of data from file

\_pageManager**->**readFileToPageAtIndex**(** orderKeyFptr**,** 0 **);**

**}** **else** **{**

found **=** **true;**

**break;**

**}**

**}**

若查询到了，将其他三个KEY分别从文件中读取出来，下面代码为从CUSTKEY文件中读取CUSTKEY的部分

**if** **(** found **)** **{** //if find orderKey, it means the tuple can be found by the orderKey

//find relative custkey

//read one page of data from custkey file

\_pageManager**->**readFileToPageAtIndex**(**custKeyFptr**,** 0**);**

**while** **(** **!**feof**(** custKeyFptr **)** **)** **{**

//search key in page

**if** **(** **!**\_pageManager**->**searchKeyInPageAtIndex**((**char**\*)  
&**orderKey**,** **sizeof(**int**),** **(**char**\*)&**custKey**,** **sizeof(**int**),** 0**)** **)** **{**

// if not found, read next page of data from file

\_pageManager**->**readFileToPageAtIndex**(**custKeyFptr**,** 0**);**

**}** **else** **{**

**break;**

**}**

**}**

……

若查询不到，错误提示

**}** **else** **{** //can not find orderKey, the query fail

printf**(** "%d is not found!\n"**,** orderKey **);**

**}**

### 5.3.3 外排序

内部快速排序，直接用qsort完成

void pagesManager**::**sortPages**(** int begin**,** int end **)** **{**

**for** **(** int i **=** begin**;** i **<=** end**;** i**++** **)** **{**

qsort**(** pages**[**i**].**getData**(),** PAGE\_SIZE**/**8**,** 8**,** compare **);**

**}**

**}**

外部的merge，在while循环内，先找到最小值

**for** **(** int i **=** begin **;** i **<=** end**;** i**++** **)** **{**

//printPage( pages[i].getData() );

**if** **(** offsets**[** i **-** begin **]** **<** PAGE\_SIZE **)** **{**

//if this page is not been read at end

**if** **(** first **)** **{** //init the min

first **=** **false;**

min **=** i**;**

**}** **else** **if(** compare**(**pages**[**i**].**getData**()+**offsets**[**i**-**begin**],**

pages**[**min**].**getData**()+**offsets**[**min**-**begin**]** **)** **<** 0 **)** **{**

min **=** i**;**

**}**

**}**

**}** //for loop end, min save the value of the pageIndex of min

将选出来的最小的值插入到page当中，若page已满，则写入文件。

**while** **(** **!**lastPage**->**insertDataToPage**(**pages**[**min**].**getData**()+**offsets**[**min**-**begin**],** 8**)** **)** **{** //insert the min value to the output page

//if insert fail

//write page to file

lastPage**->**writePageToFile**(** tempFptr **);**

lastPage**->**clearPage**();**

**}**

offsets**[** min **-** begin **]** **+=** 8**;** //next entry

检查是否已经全部写入，退出条件

//exam if all pages has been read

int i**;**

**for** **(** i **=** begin **;** i **<=** end**;** i**++)** **{**

**if** **(**offsets**[** i **-** begin **]** **<** PAGE\_SIZE**)** **{** //if a page is not at the end

**break;**

**}**

**}**

**if** **(** i **==** end **+** 1 **)** **{**

**break;**

**}**

### 5.3.4 压缩

压缩需要先进行外排序，详见5.3.3，不加赘述。

读取页面并调用compressPagesToFile对其进行压缩操作。

**while** **(** readCount **==** NUMBER\_OF\_PAGE **-** 1 **)** **{** //while not end of file

//compress these pages data, and save to file

\_pageManager**->**compressPagesToFile**(** outputFptr**,** 0**,** NUMBER\_OF\_PAGE **-** 2 **);**

//read next pages

readCount **=** \_pageManager**->**readFileToPages**(** sortedFptr**,** 0**,** NUMBER\_OF\_PAGE **-** 2 **);**

**}**

具体将页面压入文件的实现，即 compressPagesToFile的函数实现

void pagesManager**::**compressPagesToFile**(** FILE **\***fptr**,** int begin**,** int end **)** **{**

Page **\***lastPage **=** **&**pages**[** end **+** 1 **];** //output page

**for** **(** int i **=** begin**;** i **<=** end**;** i**++** **)** **{**

//compress pages one by one

pages**[**i**].**compressToPage**(** lastPage**,** **sizeof(**int**),** **sizeof(**int**),** fptr **);**

**}**

//write remain output page

**if** **(** **!**lastPage**->**isEmpty**()** **)** **{**

lastPage**->**writePageToFile**(** fptr **);**

lastPage**->**clearPage**();**

**}**

**}**

计算压缩率

//compute compress rate

fseek**(** sortedFptr **,** 0**,** SEEK\_END **);**

long beforeSize **=** ftell**(** sortedFptr **);**

fclose**(** sortedFptr **);**

fseek**(** outputFptr**,** 0**,** SEEK\_END **);**

long compressedSize **=** ftell**(** outputFptr **);**

fclose**(** outputFptr **);**

rate **=** 1.0 **\*** compressedSize **/** **(** beforeSize **/** 2.0 **);**

### 5.3.5 JOIN

## 5.4 Contant

用于定义各种变量，优化代码让代码更加可视化，运用宏定义对各种常用的特定字符串进行重命名。也用于处理一些可以部分代替全部的字符串，方便程序的运行，减少开销。

#define JOIN\_OPERATION\_NAME "join"

#define LOAD\_DATA\_OPERATION\_NAME "load"

#define QUERY\_OPERATION\_NAME "retrieve"

……

inline SHIPPRIORITY shipPriorityMap**(** const char**\*** name **)** **{**

**switch** **(** name**[**0**]** **)** **{**

**case** '1'**:**

**return** URGENT**;**

**case** '2'**:**

**return** HIGH**;**

……

**}**

**return** NONE**;**

**}**

……

# Part 6: Performance 实验结果与性能比较

将给出基于上述代码执行相关操作的结果和时间性能上分析；

# Part 7: Inspiration 心得与体会

这次project无疑是今年面对的最大的挑战，之前完全没有听说过C-store的概念，而查阅了相关的参考资料之后，我们小组经历了讨论，确定实验思路，确定实验分工，到代码重构及完善各个阶段。特别是代码重构这个环节，可谓是几经波折，改了又改，改了再改。可以说组员在这次实验中都收获颇多！

## 7.1起始的准备

准备工作可谓是十分繁杂，由于之前翻译阶段每个人阅读参考的资料皆不尽相同，所以对于C-store的理解也建立在自己阅读的那一个部分上面。而两次讨论总是因为“知识背景不同”而讨论不出结果。最后才认识到了有统一的知识储备背景的重要性。于是我们小组成员多花了很多时间先把C-store介绍性的论文仔细地阅读，实际了解它一些具体实现的作用之后再来进行讨论，结论才慢慢有一些起色。

## 7.2最初的想法

最初看到这个问题，我们小组经过讨论觉得这是一个相对简单的项目，因为涉及的导入其实就是一个文件的分段写入，而外排序则由于有老师课堂的讲解，基本可以拆分成一个内部的快排加上一个外部的归并操作。而压缩在查阅了一些资料之后，也是比较容易解决的问题。

可以说，项目的前期我们感觉思路是比较明确的，所以讨论觉得要在一开始就规范化我们的代码，把各种接口，类的层级结构都规划好，然后按照面向对象的理念，尽力把它做成一个真正实用的数据库，让它可以在其他tbl或者其他的数据表格导入后也能进行相应的操作，并且可以对于不同的列不同的表明实现操作。

这样当然要求我们的代码需要有很明确的分层以及设计得比较合理的接口，让调试可以比较轻松地完成。从而实现代码的可维护性。而且要求代码的组织结构比较统一，使得后面的扩充操作可以不断地添加进去完善它让它适用于更广泛的输入数据。

## 7.3遇到的问题

一开始还是总体的结构不明确的问题。由于没有意识到外排等代码的复杂性，一开始都归在一个类中完成，结果由于实现的东西很多，代码写得十分冗长，可读性很差，但又已经临近第二阶段的提交时间，所以只能先完成了，再来重构代码。

于是完成了外排，压缩等函数之后，重新把他们归入不同的类中来实现，使代码的区分度更高，便于阅读和维护。

还有，一开始完成的代码由于考虑的东西比较少，后面重构的过程中就会发现其中的一些局限性，例如说函数虽然有传入参数，但是传入的参数却没有用到，这导致了代码中很多不安全和冗余。于是，在重构的过程其实也是重新修正各种函数的过程。

中间外排还出现了不小的问题，出现了比较大的BUG。而且显得无厘头。因为排序完居然比原来的文件小了将近50K。而这是之前的所有测试所没有遇到的问题。提出的几个假设也在接下来的调试中被否定掉。进制一度停滞不前。结果最后发现其实是测试的文件写错了。这也为我们敲响了警钟，不管是项目的代码，还是测试的代码都要保持严谨的态度，要不然因为测试代码的问题而不断地进行无谓的调试显然是十分不划算的！

## 7.4感想与收获

其实我们做这个项目前期的想法我觉得是没有问题的，就是我们要先定好框架，然后不断地充实它，然后再慢慢地让它实现更多的功能最后实现题目的要求甚至是做出一些扩展。

但是在实验过程中我们发现，有时候为了实现一些功能，往往会先不顾组织架构的设计，先把它完成了，然后再进行调整。其实后来觉得这是一种新的思路。就是其实一个项目一开始并不用把框架描述得那么细致，因为有一些细节的东西处理起来也许并不是一开始想象的那么简单。但代码的过程中往往是先把它实现出来，然后再去完善它，效率会高过直接就按照最后那种很高的标准去写。也许就是程序员经常说的“先做到在做好”吧。

还有在小组当中，及时的沟通显然是非常重要的。像我们小组中，虽然说比较主体的代码是由同一个人完成的，但是其他两个人也分别负责了一些小的模块。而因为代码都要与上一层的代码相对接，有时候由于阅读代码的偏差，有一些函数的功能理解不同，往往就会实现出另外一种效果。也许有时也能实现出来，但是无疑打乱了之前的思路而影响接下来的进展。而及时地沟通，及时地说出自己的理解，当意见发生碰撞时，也能及时地理解对方的思路，统一意见。有时思维的碰撞在沟通中还能得出一些更好的实现方法。

其实这次project也是对于编程能力的一次很好的训练，毕竟这学期也没有算法之类的课程，代码多少有一点忘记。特别是对于C++里面文件流的输入输出这方面的内容，确实已经遗忘了很多。这次实验就采用C++语言进行编写，debug的过程其实也会出现很多之前学习C++编程时遇到的低级错误等，就当是一次以实战为基础的很好的复习！

最后，在我们project进行的过程中也遇到了不少问题，感谢对我们提出的问题进行解答的同学，TA。也谢谢老师提供这样一个很好的机会让我们加深了对于数据库的理解与实战运用。这次project真的受益匪浅！