实验一：MySQL 关系数据库管理系统

及 SQL 语言的使用

高宏 邹兆年

1. 实验目的

掌握 MySQL 关系数据库管理系统的基本命令，并熟练使用 SQL 语言管理 MySQL 数据库。掌握 SQL 语言的使用方法，学会使用 SQL 语言进行关系数据库查询，特别是聚集查询、连接查询和嵌套查询。

2. 实验环境

Windows 操作系统、MySQL，Code Blocks 编程环境。

3. 实验内容

3.1 实验任务

创建关系数据库 COMPANY，使用 SQL 语言完成如下查询：

1：参加了项目名为“SQL Project”的员工名字；
2：在“Research Department”工作且工资低于 3000 元的员工名字和地址；
3：没有参加项目编号为 P1 的项目的员工姓名；
4：由张红领导的工作人员的姓名和所在部门的名字；
5：至少参加了项目编号为 P1 和 P2 的项目的员工号；
6：参加了全部项目的员工号码和姓名；
7：员工平均工资低于 3000 元的部门名称；
8：至少参与了 3 个项目且工作总时间不超过 8 小时的员工名字；
9：每个部门的员工小时平均工资；

3.2 关系数据库 COMPANY 介绍

创建关系数据库 COMPANY，其模式如下（下划线表示关系的主键）：

关系 EMPLOYEE (ENAME, ESSN, ADDRESS, SALARY, SUPERSSN, DNO)
ENAME: 工作人员名字，
ESSN: 工作人员身份证号，
ADDRESS: 工作人员住址，
SALARY: 工作人员工资，
SUPERSSN: 工作人员直接领导的身份证号，
DNO: 所属部门号

关系 DEPARTMENT (DNAME, DNO, MGRSSN, MGRSTARTDATE)

实验二：使用高级语言操作 MySQL 数据库

高宏 邹兆年

1. 实验目的

学会使用高级语言访问 MySQL 数据库，并进行查询。

2. 实验环境

MySQL 关系数据库管理系统、C++编译器。

本次实验主要利用 C 语言访问 MySQL 数据库，也可以使用 JAVA，PHP 等其他语言。不允许使用 ORM。

3. 实验内容

3.1 实验任务

在上次上机实验课建立的 COMPANY 数据库上，用 C 语言编写程序，完成如下查询，程序的命令行参数为：

company_query –q <Number> -p [Parameters]

其中，Number 代表待执行查询的序号，Parameters 为第 Number 号查询需要的参数列表。

待执行的 9 个查询为如下：

1：参加了项目编号为%PNO%的项目的员工号，其中%PNO%为 C 语言编写的程序的输入参数；

2：参加了项目名为%PNAME%的员工名字，其中%PNAME%为 C 语言编写的程序的输入参数；

3：在%DNAME%工作的所有工作人员的名字和地址，其中%DNAME%为 C 语言编写的程序的输入参数；

4：在%DNAME%工作且工资低于%SALARY%元的员工名字和地址，其中%DNAME%和%SALARY%为 C 语言编写的程序的输入参数；

5：没有参加项目编号为%PNO%的项目的员工姓名，其中%PNO%为 C 语言编写的程序的输入参数；

6：由%ENAME%领导的工作人员的姓名和所在部门的名字，，其中%ENAME%为 C 语言编写的程序的输入参数；

7：至少参加了项目编号为%PNO1%和%PNO2%的项目的员工号，其中%PNO1%和%PNO2%为 C 语言编写的程序的输入参数；

8：员工平均工资低于%SALARY%元的部门名称，其中%SALARY%为 C 语言编写的程序的输入参数；

9：至少参与了%N%个项目且工作总时间不超过%HOURS%小时的员工名字，其中%N%和%SALARY%为 C 语言编写的程序的输入参数；

3.2 建立 MySQL 工程方法

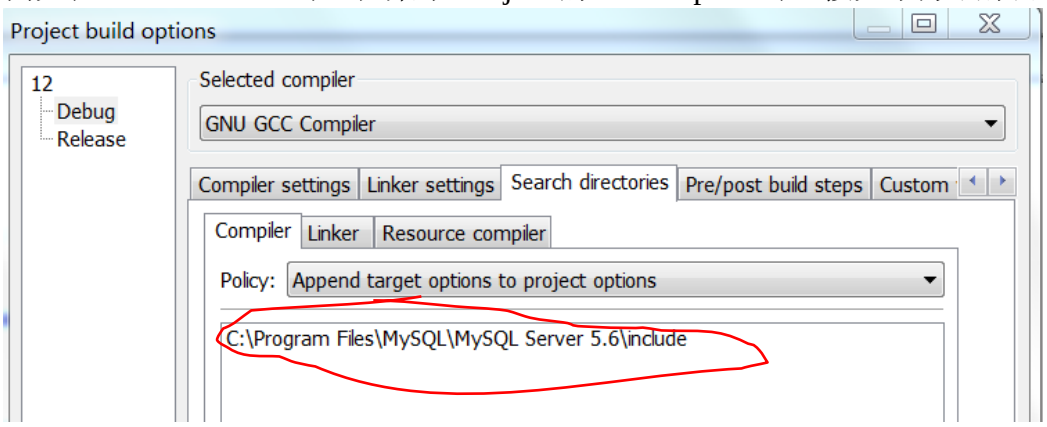
了解编程相关的资源文件。MySQL 安装路径下的 include 文件夹存放 C 语言编译相关的头文件（例如重要的 mysql.h），lib 文件夹存放相关的静态库（例如重要的 opt/libmysql.lib）

注：MySQL Server5.1.41 不用添加，MySQL Server5.6 需要自行添加以下路径。

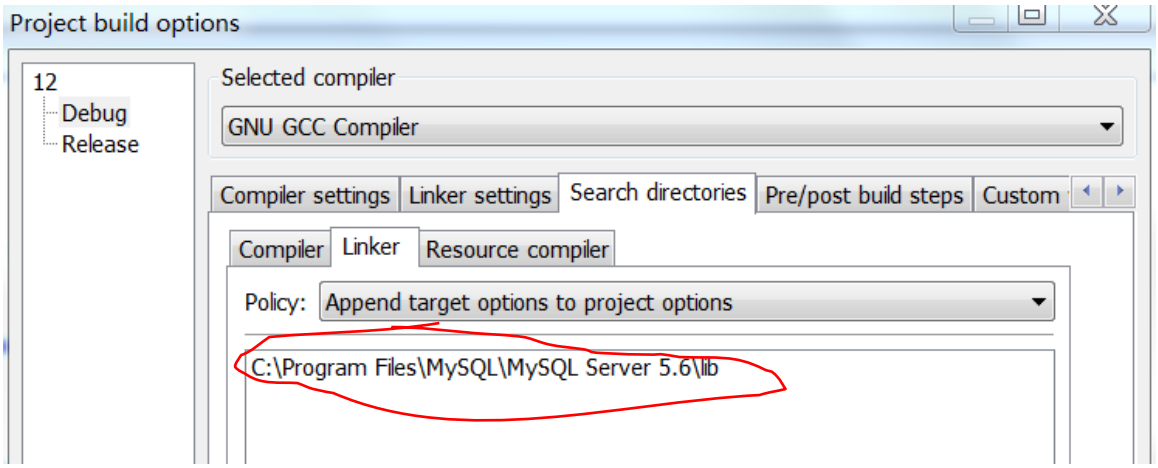
使用 C 语言编程环境（Code::Blocks 或 Visual C++）建立空白工程。在该工程的编译和链接配置中，首先添加引用路径"%MySQL 安装路径%/include/"

其中"%MySQL 安装路径%"为 MySQL 安装路径，在实验中心的计算机上为 C:/Program Files/MySQL/include/

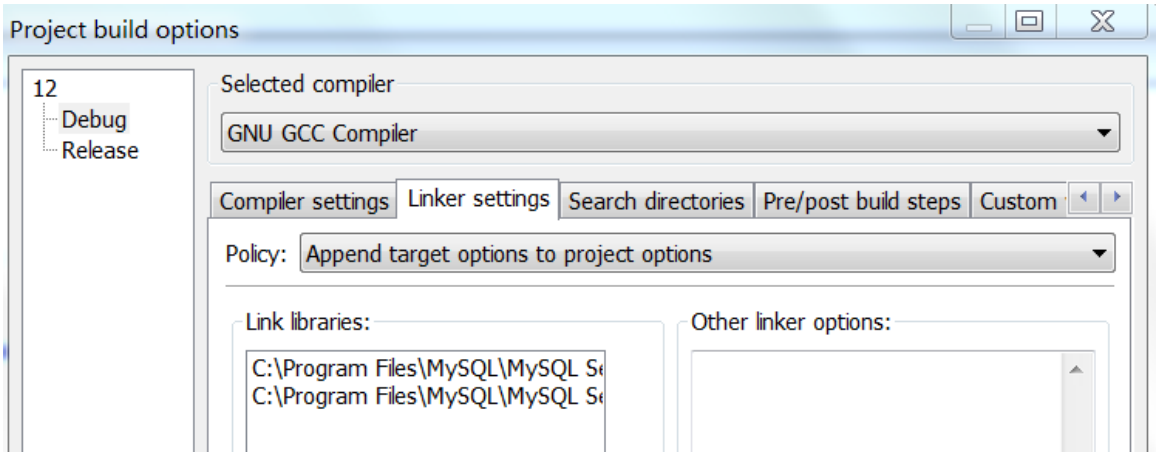
例如在 Code blocks 中，在菜单 Project 的 Build option 中，按如下方法添加:



然后，添加静态库路径 %MySQL 安装路径%/lib/



最后，添加需要的库文件%MySQL 安装路径%/lib/下的库文件 libmysql.lib 和 mysqlclient.lib



3.3 连接、断开 MySQL 服务器

在程序中必须加入#define __LCC__，以避免在 Windows 操作系统下进行编译时产生的错误。然后，在程序中加入#include <mysql.h>，让编译器能够找到所有 MySQL Client 的函数定义。

随后，使用 mysql_init 函数准备连接，mysql_init 函数的声明为

MYSQL *mysql_init(MYSQL *mysql);

其参数和返回值均为类型为 MYSQL 的结构体的指针。MYSQL 结构体是连接 MySQL 服务器的句柄，每一个句柄代表程序与 MySQL 服务器唯一的连接，所有 insert，select，delete 等操作都需要 MySQL 句柄作为参数以识别不同用户、数据库以及程序。若 mysql_init 操作成功，则返回刚刚初始化的句柄；否则，返回 NULL。

下面是一个简单的例子。

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv)
{
   MYSQL mysql_conn; /* Connection handle */

   if (mysql_init(&mysql_conn) != NULL)
{
      printf("Init succeeds!\n");
   }
```

```
else
{
      printf("Init fails!\n");
   }
   return 0;
}
```

编译并运行该程序，若句柄初始化成功，则打印”Init succeeds!”；否则，打印”Init fails!”。

在句柄被初始化后，就可以使用 mysql_real_connect 函数或者 mysql_connect 函数连接 MySQL 服务器了，这两个函数的声明如下：

```
MYSQL *mysql_real_connect(
   MYSQL *mysql, const char *host, const char *user,
   const char *passwd, const char *db, unsigned int port,
   const char *socket, unsigned long client_flag);
```

其中，第 1 个参数是 MySQL 句柄，第 2 至第 4 个参数分别为 host name、user name 及 password，第 5 个参数是待使用的数据库的名字，你也可把第 5 个参数设为 NULL，之后使用 mysql_select_db 函数来选择数据库，第 6 个参数是 MySQL 服务器的连接池，通常把它设为 MYSQL_PORT，第 7 个参数是 MySQL 服务器及 client 之间传输的渠道（socket 或 named pipe），但 MySQL 服务器会根据 host name 建立另一渠道，除非你有真的需要，否则你需在第 7 个参数填上 NULL。最后是 client_flag，包括压缩协议、查询协议、加密协议等。

与 MySQL 服务器的连接成功后，使用 mysql_close 函数断开连接，其函数声明为：

```
int mysql_close(MYSQL*);
```

下面是一个简单的例子：

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv) {
  MYSQL mysql_conn; /* Connection handle */

    if (mysql_init(&mysql_conn) != NULL) {
       if (mysql_real_connect(
            &mysql_conn, "localhost", "root",
```

```
            NULL, "company", MYSQL_PORT,
            NULL, 0) != NULL)
          printf("GOOD!\n");
       else
          printf("Connection fails.\n");
    } else {
       printf("Initialization fails.\n");
       return -1;
    }

    mysql_close(&mysql_conn);
    return 0;
}
```

至此，我们学会了如何连接、断开 MySQL 服务器。

3.4 查询 MySQL 数据库

在成功连接 MySQL 服务器后，便可以使用 mysql_query 函数查询数据库。mysql_query 函数的声明如下：

```
int mysql_query(MYSQL *mysql, const char *query);
```

第 1 个参数是 MySQL 连接的句柄，第 2 个参数是 SQL 语句。如果查询成功，mysql_query 返回 0，否则返回非 0。

如果你的查询语句不需要结果返回，例如 DELETE、UPDAE、INSERT 等，mysql_query 被执行后便完成整个操作了；如果你要执行 SELECT、SHOW、DESCRIBE 等，在存取结果前，必须使用 mysql_store_result 函数建立查询结果的句柄。mysql_store_result 的函数声明如下：

```
MYSQL_RES *mysql_store_result(MYSQL *mysql);
```

其输入参数为 MySQL 的连接句柄，其输出为 MYSQL_RES 结构体指针类型。MYSQL_RES 结构体表示的是 MySQL 数据库传递回来的查询结果，但我们并不会直接读取 MYSQL_RES 的数据，而是使用 MYSQL_ROW，也即是以行为单位读取数据。

例子如下所示：

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <mysql.h>
```

```
int main(int argc, char **argv) {
   MYSQL mysql_conn; /* Connection handle */
   MYSQL_RES *mysql_result;

   if (mysql_init(&mysql_conn) != NULL) {
      if (mysql_real_connect(
            &mysql_conn, "localhost", "root",
            NULL, "company", MYSQL_PORT,
            NULL, 0) != NULL) {
         mysql_query(
            &mysql_conn, "select * from table1");
         mysql_result = mysql_store_result(&mysql_conn);

         /* Free the result to release the heap memory*/
         mysql_free_result(mysql_result);
      } else {
         printf("Connection fails.\n");
      }
   } else {
      printf("Initialization fails.\n");
      return -1;
   }

   mysql_close(&mysql_conn);
   return 0;
}
```

请注意 mysql_result 是一个指针。因为 mysql_store_result 函数会自动分配内存存储查询结果，所以需要执行 mysql_free_result(MYSQL_RES*)来释放内存。

3.5 提取查询结构

在提取结果前必须使用上面介绍过的 mysql_store_result 函数为查询结果分配内存，然后使用 mysql_fetch_row 函数逐行提取数据。mysql_fetch_row 函数的声明如下：

MYSQL_ROW mysql_fetch_row(MYSQL_RES *result);

其输入参数为查询结果句柄，输出的类型为 MYSQL_ROW。MYSQL_ROW 是一个数组结构，数组中每一个元素依次为该元组各个属性上的值。

结果的元组数可以用 mysql_num_rows 函数返回。mysql_num_rows 函数的声明为

int mysql_num_rows(MYSQL_RES*);

其输入参数为查询结果句柄，输出结果为元组数量。

下面是一个简单的例子：

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <malloc.h>
#include <mysql.h>

int main(int argc, char **argv) {
   MYSQL mysql_conn; /* Connection handle */
   MYSQL_RES *mysql_result; /* Result handle */
   MYSQL_ROW mysql_row; /* Row data */
   int f1, f2, num_row, num_col;

   if (mysql_init(&mysql_conn) != NULL) {
      if (mysql_real_connect(
         &mysql_conn, "localhost", "root",
         NULL, "company", MYSQL_PORT, NULL, 0) != NULL){
         if (mysql_query(
               &mysql_conn,
               "select * from testtable limit 10") == 0) {
            mysql_result = mysql_store_result(&mysql_conn);
            num_row = mysql_num_rows(mysql_result);
            num_col = mysql_num_fields(mysql_result);

            for (f1 = 0; f1 < num_row; f1++) {
               mysql_row = mysql_fetch_row(mysql_result);

               for (f2 = 0; f2 < num_col; f2++)
                  printf(
                     "[Row %d, Col %d] ==> [%s]\n",
                     f1+1, f2+1, mysql_row[f2]);
            }
         } else
            printf("Query fails\n");
      } else {
            int i = mysql_errno(&mysql_conn);
            const *s = mysql_error(&mysql_conn);
            printf("Connection fails(ERROR %d): %s\n", i, s);
      }
```

```
    } else
        printf("Initialization fails\n");

    mysql_free_result(mysql_result);
    mysql_close(&mysql_conn);
    return 0;
}
```

请注意如果数据库很大，而又没有用 mysql_free_result 函数释放内存的话，很容易发生内存溢出的问题。

4. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》

实验三：数据库系统开发

高宏 邹兆年

1. 实验目的

在熟练掌握 MySQL 基本命令、SQL 语言以及用 C 语言编写 MySQL 操作程序的基础上，学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计。

2. 实验环境

MySQL 关系数据库管理系统、C++编译器。
本次实验可使用 C，C++，JAVA，PHP 或其他语言均可。

3. 实验内容

开发一个数据库系统，可以参考教材的例子。

3.1 要求

1. 该系统的 E-R 图至少包括 8 个实体和 7 个联系（必须有一对一联系、一对多联系、多对多联系）。
2. 在设计的关系中需要体现关系完整性约束：主键约束、外键约束，空值约束。
3. 对几个常用的查询创建视图、并且在数据库中为常用的属性（非主键）建立索引。
4. 该系统功能必须包括：插入、删除、连接查询、嵌套查询、分组查询。其中插入，删除操作需体现关系表的完整性约束，例如插入空值、重复值时需给予提示或警告等。
5. 包含事务管理（如在程序中显示保证事务操作的原子性）、触发器功能。

3.2 作业检查

1. 检查系统的 E-R 图，关系的完整性约束，索引，视图。
2. 整个系统（插入、删除、查询、加分项）。

3.3 数据库系统示例

考虑建立一个简单的社会网络系统。逻辑上，该系统具有如下功能：

1. 用户可以在该系统中注册、修改个人基本信息，包括姓名、性别、出生日期、电子邮箱、通讯地址、用户密码（注意，一个用户可以注册多个电子邮箱，但作为用户名使用的只能有一个）；
2. 用户可以在该系统中录入、修改个人经历，包括教育经历（教育级别、起止年月、学校名称、学位）、工作经历（工作单位、起止时间、职位）。注意，用户可以全部、部分或不录入个人经历信息；
3. 用户可以在该系统中搜索、添加、删除好友，还可以添加、修改、删除好友分组，向好友分组中添加、删除好友；

4. 用户可以在该系统中发表、更改、删除日志，系统记录日志的发表或最后更新时间；
5. 用户可以对好友日志或其他可回复的日志进行回复，系统记录回复信息的发表时间、内容、被回复的用户，注意，若用户删除自己发表的日志，则相关回复信息也将全部被删除；
6. 用户可以对好友信息或其他可回复的信息进行回复，系统记录回复信息的发表时间、内容、被回复的用户；
7. 用户可以分享好友的日志或其他公开的可以分享的日志，并对该日志进行评论，系统记录分享时间、评论时间和评论内容；

4. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》

实验四：查询处理算法的实现

高宏 邹兆年

1. 实验目的

掌握关系连接操作的实现算法，理解算法的 I/O 复杂性，使用高级语言语言实现重要的关系连接操作算法。

2. 实验环境

使用本实验提供函数库或者使用其他语言重构函数库实现均可。

3. 实验内容

3.1 实验任务

关系 R 具有两个属性 A 和 B，其中 A 和 B 的属性值均为 int 型（4 个字节），A 的值域为[1, 40]，B 的值域为[1, 1000]。

关系 S 具有两个属性 C 和 D，其中 C 和 D 的属性值均为 int 型（4 个字节）。C 的值域为[20, 60]，D 的值域为[1, 1000]。

1）实现关系选择算法：基于 ExtMem 程序库，使用 C 实现关系选择算法，选出 R.A=40 或 S.C=60 的元组，并将结果存放在磁盘上。要求实现至少三种选择算法，包括线性搜索算法，二元搜索算法，和任意一种索引算法（使用 B+/B-树可加分）。

2）实现关系投影算法：基于 ExtMem 程序库，使用 C 语言实现关系投影算法，对关系 R 上的 A 属性进行投影，并将结果存放在磁盘上。

3）实现连接操作算法：基于 ExtMem 程序库，使用 C 语言实现连接操作算法，对关系 R 和 S 计算 R.A 连接 S.C，并将结果存放在磁盘上。要求实现三种连接操作算法：Nest-Loop- Join 算法，Sort-Merge-Join 算法，Hash-Join 算法。

注意：

（**1**）需要在检查前准备好相应的数据，详见 **3.3** 节。

（**2**）每人需要完成上述全部算法，根据完成情况给分。

（**3**）本次实验不需要提交实验报告，但需要提交实验代码。

（**3**）如果自创函数库，使用其他高级程序语言实现选择、投影、连接算法亦可。

3.2 ExtMem 程序库介绍

ExtMem 程序库是一个专门为本课程编写的模拟外存磁盘块存储和存取的程序库，由 C 语言开发。ExtMem 程序库的功能包括内存缓冲区管理、磁盘块读/写它提供了 1 个数据结构和 7 个 API 函数。

ExtMem 程序库定义了 Buffer 数据类型，包含如下 6 个域：

- numIO：外存 I/O 次数；
- bufSize：缓冲区大小（单位：字节）；
- blkSize：块的大小（单位：字节）；
- numAllBlk：缓冲区内可存放的最多块数；
- numFreeBlk：缓冲区内可用的块数；
- data：缓冲区内存区域。

缓冲区内每个块的大小为 blkSize 个字节，其最后 4 个字节用来存放其后继磁盘块的地址（在 ExtMem 库中，我们 4 个字节来记录磁盘块地址，地址在程序中为 unsigned int 类型。若无后继磁盘块，则置为 0），其余(blkSize – 4)个字节用于存放块内的记录。

ExtMem 库提供了如下 API 函数：

- Buffer *initBuffer(size_t bufSize, size_t blkSize, Buffer *buf);
  初始化缓冲区，其输入参数 bufSize 为缓冲区大小（单位：字节），blkSize 为块的大小（单位：字节），buf 为指向待初始化的缓冲区的指针。若缓冲区初始化成功，则该函数返回指向该缓冲区的地址；否则，返回 NULL。
- void freeBuffer(Buffer *buf);
  释放缓冲区 buf 占用的内存空间。
- unsigned char *getNewBlockInBuffer(Buffer *buf);
  在缓冲区 buf 中申请一个新的块。若申请成功，则返回该块的起始地址；否则，返回 NULL。
- void freeBlockInBuffer(unsigned char *blk, Buffer *buf);
  解除块 blk 对缓冲区内存的占用，即将 blk 占据的内存区域标记为可用。
- int dropBlockOnDisk(unsigned int addr);
  从磁盘上删除地址为 addr 的磁盘块内的数据。若删除成功，则返回 0；否则，返回-1。
- unsigned char *readBlockFromDisk(unsigned int addr, Buffer *buf);
  将磁盘上地址为 addr 的磁盘块读入缓冲区 buf。若读取成功，则返回缓冲区内该块的地址；否则，返回 NULL。同时，缓冲区 buf 的 I/O 次数加 1。
- int writeBlockToDisk(unsigned char *blkPtr, unsigned int addr, Buffer *buf);
  将缓冲区 buf 内的块 blk 写入磁盘上地址为 addr 的磁盘块。若写入成功，则返回 0；否则，返回-1。同时，缓冲区 buf 的 I/O 次数加 1。

文件 test.c 中给出了 ExtMem 库使用方法的一个具体示例。

声明：ExtMem 库是为本课程专门开发的模拟外存磁盘块存储和存取的程序库，不保证其能够真正实现对磁盘块的存取操作，同时也不保证其排除一切软件错误。本课程及 ExtMem 开发者不会对使用该程序库所导致的一切错误负责。

3.3 数据准备

使用 ExtMem 程序库建立两个关系 R 和 S 的物理存储。关系的物理存储形式为磁盘块序列 $B_1, B_2, \ldots, B_n$，其中 B_i 的最后 4 个字节存放 B_{i+1} 的地址。

即 R 和 S 的每个元组的大小均为 8 个字节。

块的大小设置为 64 个字节，缓冲区大小设置为 512+8=520 个字节。这样，每块可存放 7 个元组和 1 个后继磁盘块地址，缓冲区内可最多存放 8 个块。

编写程序，随机生成关系 R 和 S，使得 R 中包含 16 * 7 = 112 个元组，S 中包含 32 * 7 = 224 个元组。

3.4 补充说明

实验要求限制使用的内存大小。即算法的操作应在 8 个块的内存缓冲区实现，不能再另开较大内存空间存放数据。必要的变量、非存放数据的较小数组等可以存放在内存的其他地址（非缓冲区）。

4. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》

DNAME: 部门名，
DNO: 部门号，
MGRSSN: 部门领导身份证号，
MGRSTARTDATE: 部门领导开始领导工作的日期

关系 PROJECT (PNAME, PNO, PLOCATION, DNO)
PNAME: 工程项目名，
PNO: 工程项目号，
PLOCATION: 工程项目所在地，
DNO: 工程项目所属部门号

关系 WORKS_ON (ESSN, PNO, HOURS)
ESSN: 工作人员身份证号，
PNO: 工程项目号，
HOURS: 工作小时数

3.3 数据准备

向创建的数据库 COMPANY 中添加数据，以备后续查询使用。

要求数据库中至少包含 50 个员工，5 个部门，10 项工程，并且必须包含“研发部”、编号为 P1 和 P2 的项目、名叫张红的员工。

4. MySQL 手册

4.1 MySQL 基本命令

1) 连接 MySQL 服务器
在命令行下输入 mysql –h localhost –u root –p
当显示“Enter password:”时，输入密码 mysql

2) 查看 MySQL 中有哪些数据库
mysql> show databases; （注意 SQL 语句结尾的分号!）

3) 使用数据库 mysql
mysql> use mysql;

4) 查看当前使用的数据库
mysql> select database();

5) 查看数据库 mysql 中有哪些关系
mysql> show tables;

6) 查看数据库 mysql 中关系的模式（以关系 user 为例）
mysql> describe user;

7) 使用 help 命令来了解其他命令和变量类型等的含义

```
mysql> help;
mysql> help use;
```

8) 使用 SQL 语言在数据库 mysql 上进行简单查询

```
mysql> select * from user;
mysql> select user, host, password from user;
mysql> select count(*) from user;
mysql> select count(*) as ucount from user;
```

9) 取消命令

若要取消一条正在编辑命令，键入\c 并回车

10) 断开 MySQL 服务器连接

```
mysql> quit  或  mysql> exit
```

4.2 使用 SQL 语言管理 MySQL 数据库

1) 创建数据库 menagerie

```
mysql> CREATE DATABASE menagerie;
```

2) 查看数据库 menagerie 是否创建成功

```
mysql> show databases;
```

3) 使用 menagerie 数据库

```
mysql> use menagerie;
```

4) 创建宠物信息关系 pet，包含宠物名字、主人、种类、性别、出生和死亡日期

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

5) 查看关系 pet 是否创建成功

```
mysql> show tables;
```

6) 查看关系 pet 的模式

```
mysql> describe pet;
```

7) 向关系 pet 中插入元组

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

8) 从文件中批量导入数据

设 D:\pet.txt 是一个由字符 Tab 分隔的文本文件

```
Fluffy    Harold   cat   f     1993-02-04   \N
```

```
Claws   Gwen    cat   m    1994-03-17  \N
Buffy   Harold  dog f     1989-05-13  \N
Fang    Benny   dog m     1990-08-27  \N
Bowser  Diane   dog m     1998-08-31  1995-07-29
Chirpy  Gwen    bird f    1998-09-11  \N
WhistlerGwen    bird \N   1997-12-09  \N
Slim    Benny   snake    m    1996-04-29  \N
```

在 MySQL 提示符下执行

```
mysql> LOAD DATA LOCAL INFILE "D:\pet.txt" INTO TABLE pet;
```

9) 查询关系 pet 中所有元组

```
mysql> SELECT * FROM pet;
```

10) 更改关系 pet 中的数据

```
mysql> UPDATE pet SET birth = "1989-08-31" WHERE name = "Bowser";
```

11) 查询名叫 Bowser 的宠物信息

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
```

12) 查询所有 90 后小猫和小狗的信息

```
mysql> SELECT * FROM pet WHERE birth >= "1990-1-1" AND
    -> (species = "dog" OR species = "cat");
```

13) 查询所有宠物的主人

```
mysql> SELECT owner FROM pet;
mysql> SELECT DISTINCT owner FROM pet;
```

14) 查询所有宠物的名字及生日，并按其年龄递增排序

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
```

15) 查询所有宠物的名字及生日，并按其年龄递减排序

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
```

16) 查询所有宠物的名字、种类及生日，并先按种类名称递增排序，同种宠物按年龄递减排序

```
mysql> SELECT name, species, birth FROM pet ORDER BY species, birth DESC;
```

17) 查询所有活着的宠物的信息

```
mysql> SELECT * FROM pet WHERE death IS NULL;
```

18) 查询关系 pet 中有多少只宠物

```
mysql> SELECT COUNT(*) FROM pet;
```

19) 查询每个主人有多少只宠物

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
```

20) 查询每种宠物的数量

```
mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
```

21) 查询雄狗和雌狗的数量

```
mysql> SELECT species, sex, COUNT(*) FROM pet
    -> WHERE species = "dog" GROUP BY species, sex;
```

22) 查询每种宠物的最大年龄

```
mysql> SELECT species, MAX(birth) FROM pet GROUP BY species;
```

5. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》

《MySQL 中文参考手册》（MySQLBook.chm）