运行程序

首先清空bin/data下全部文件，然后命令行进入bin目录下

运行server：

```
java -jar server.jar
```

运行client：

```
java -jar client.jar
```

我们的server写死在本地（127.0.0.1）的6667端口上，只能同时运行一个服务器。但是客户端可以同时运行多个。

进入客户端之后，必须先输入

```
connect;
```

建立连接。之后会弹出提示让你输入账号和密码，由于目前我们没有实现权限管理，随意输入即可。

在建立连接成功之后，就可以直接输入sql代码来操作了。

关闭客户端之前必须输入

```
disconnect;
```

关闭连接。

sql代码执行

我们支持以下sql语句（不区分大小写）：

创建数据库

```
CREATE DATABASE University
```

切换数据库

```
USE University
```

注意，如果一开始没有任何数据库，则create一个数据库之后就会自动使用它。否则必须显式使用use来切换。

删除数据库

```
DROP DATABASE University
```

创建表

```
CREATE TABLE tableName(attrName1 Type1, attrName2 Type2,…, attrNameN TypeN NOT NULL, PRIMARY KEY(attrName1))
```

我们支持，而且仅支持“NOT NULL”和“PRIMARY KEY”这两个关键字，Type为Int，Long，Float，Double，String（必须给出长度）之一。主键仅在某一列定义。

示例:

```
CREATE TABLE person (name String(256), ID Int not null, PRIMARY KEY(ID))
```

删除表

```
DROP TABLE tableName
```

展示某张表的模式信息

```
SHOW TABLE tableName
```

我们展示的模式信息如下

```
Table Name: person
Column Name, Column Type, Primary, Is Null, Max Length
name,STRING,0,false,256
id,INT,1,true,-1
```

第一行是表名，第三行和之后代表表的每一列的名称，类型，是否主键，是否非空等信息

插入数据

```
INSERT INTO tableName(attrName1, attrName2,…, attrNameN VALUES (attrValue1, attrValue2,…, attrValueN)
```

字符串需要用英文单引号包围，示例

```
INSERT INTO person VALUES ('Bob', 15)
```

会正常插入

```
INSERT INTO person(name) VALUES ('Bob')
```

会提示字段ID不能为空。

删除数据

```
DELETE FROM tableName WHERE attrName1 = attrValue1 and attrName2 =
attrValue2
```

示例

```
delete from person where name='Bob' and id=15
```

更新数据

```
UPDATE tableName SET attrName = attrValue WHERE attrName = attrValue
```

示例

```
update person set name='sgl' where id>14
```

增加，删除，更新数据都会报告增/删/改了多少条数据

查询数据：单一表

```
SELECT attrName1, attrName2, … attrNameN FROM tableName WHERE attrName1 =
attrValue1 and attrname2=attrvalue2
```

示例

```
select name from person where id>14 or name='sgl'
```

查询数据：多表

```
SELECT tableName1.AttrName1, tableName1.AttrName2…, tableName2.AttrName1,
tableName2.AttrName2,… FROM tableName1 JOIN tableName2 ON
tableName1.attrName1 = tableName2.attrName2 WHERE attrName1 = attrValue
```

示例

```
select name,student.id,GPA from student join grade on student.id=grade.id
where GPA>3.7
```

以上查询输出格式如下：

```
ThssDB>select name, id from person
name, id
----------------------------------------------------------------
Anna, 20
Bob, 22
Cindy, 30
```

第一行是待查询的列名，和select中的列名一一对应。第而行和之后代表结果的每一行对应列的数据。

注意：以上delete，update，select用到的所有where和on条件都支持"<","<=",">=","=","<>"6种比较类型和&&,||两种逻辑（不限个数，考虑优先级），而被join的表不限个数。

事务

单事务

事务命令很简单，使用`begin transaction`可以开启事务，开启事务后会出现(T)标识作为事务模式的标志；使用`commit`可以提交事务。

根据事务的定义，只支持在事务中使用`insert, delete, update, select`这几种DML，不应该在事务中使用DDL等语言，使用示例如下：

```
ThssDB>begin transaction
start transaction
It costs 3 ms.

ThssDB(T)>INSERT INTO person VALUES ('wsj', 100)
Inserted 1 rows.
It costs 38680 ms.

ThssDB(T)>select * from person
person.name, person.id
-----------------------------------------------------------------
Amy, 3
Alice, 20
sgl, 55
wsj, 100
It costs 1 ms.

ThssDB(T)>delete from person where name='Alice'
Deleted 1 items.
It costs 2 ms.

ThssDB(T)>select * from person
person.name, person.id
-----------------------------------------------------------------
Amy, 3
sgl, 55
wsj, 100
It costs 1 ms.

ThssDB(T)>commit
commit transaction
It costs 2 ms.

ThssDB>select * from person
person.name, person.id
-----------------------------------------------------------------
Amy, 3
sgl, 55
wsj, 100
It costs 2 ms.
```

多事务

多事务实现了`read commited`隔离级别，解决了脏读问题，在一个事务对表进行了修改操作后，会将整个表上写锁，当其他事务在此时进行读写等操作，则会被阻塞并按照顺序进入执行队列，在进行了修改操作的事务被提交后才会按顺序执行其它事务；如果一个事务只对表进行了读操作，则在读后并不影响其它事务读取该表。

单事务的WAL机制

本数据库采用了单事务的WAL机制用于保证事务的原子性。当数据库进行`insert, delete, update`操作时，数据库会将这些操作命令记录到log文件中，`begin transaction, commit`命令也会被记录，通过这样的WAL机制，即使是commit后的数据也不会立刻被写入磁盘，只有在log文件大于一定大小时，在commit后会写入磁盘，清空log并完成实际上的持久化。在数据库启动时，会读取log文件并按照日志的命令对数据库进行恢复；当某个事务未执行完（未commit）就出现了中断，则读取log时服务器会只读取到这个未执行完的事务的前一句，并且将之后未执行完的事务log删除，从而保证事务的原子性。

下图是log文件实例：

```
INSERT INTO person VALUES ('Bob', 15)
INSERT INTO person VALUES ('Alice', 20)
begin transaction
INSERT INTO person VALUES ('Amy', 3)
INSERT INTO person VALUES ('Lucy', 55)
delete from person where name='Bob'
update person set name='sgl' where id=55
commit
begin transaction
INSERT INTO person VALUES ('wsj', 100)
delete from person where name='Alice'
commit

```

重启server后会读取wal恢复数据库

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
log file size: 358 Byte
Read WAL log to recover database.
read 12 lines
[Thread-0] INFO cn.edu.thssdb.server.ThssDB - Starting ThssDB ...
```