

数据库用户文档

运行配置

首先清空/Data下全部文件。

- 运行server:
直接运行 `src/main/java/cn/edu/thssdb/server/ThssDB.java` , 即可打开server。
- 运行client:

直接运行 `src/main/java/cn/edu/thssdb/client/Client.java` , 即可打开client。

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

```
1 | connect;
```

建立连接。之后会弹出提示让你输入账号和密码, 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

```
1 | quit;
```

退出客户端。

语句执行

支持以下sql语句且不区分大小写:

创建数据库:

```
1 | create database database_name;
```

需要先创建数据库, 才能执行操作, 其他命令会提示目前没有存在的数据库。在创建数据库后会自动将当前的数据库切换到创建的数据库。

切换数据库, 如果database不存在会报错提醒。

```
1 | use database database_name;
```

删除数据库, 如果database不存在会报错提醒。

```
1 | drop database database_name
```

打印所有的database下的table的元数据。

```
1 | show databases
```

展示特定database下table的元数据。

```
1 | show database database_name
```

创建表格，支持多种Type: Int, Long, Float, Double, Strin。支持 Not null 和 PRIMARY KEY

```
1 | CREATE TABLE tableName(attrName1 Type1, attrName2 Type2,..., attrNameN TypeN  
  | NOT NULL, PRIMARY KEY(attrName1));
```

删除表格

```
1 | DROP TABLE tableName;
```

展示表格的schema:

```
1 | SHOW TABLE tableName
```

增加数据

```
1 | INSERT INTO tableName VALUES (value1, value2,...)
```

也可以通过按列插值

```
1 | INSERT INTO table_name (column1, column2,...) VALUES (value1, value2,...)
```

更新数据

where语句还支持attrName与attrName之间的判断和AND/OR逻辑判断。

在插入发现问题的时候也会重新回滚到正常的状态。

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

查询数据

可以通过select进行单表和多表查询:

所有的查询的Where和On语句都支持AND/OR逻辑运算符来实现多种条件。

所有的查询都支持 select distinct 除重。

- 单表查询

支持从表格中查询特定数据:

```
1 | SELECT attrName1, attrName2, ... attrNameN FROM tableName [ WHERE attrName1 =  
  | attrValue]
```

支持从表格中通过通配符 * 查询所有数据:

```
1 | SELECT * FROM tableName [ WHERE attrName1 = attrValue]
```

- 多表查询

ON和WHERE都支持AND/OR多条件逻辑连接。
支持属性和属性之间比较。

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

支持多表的JOIN:

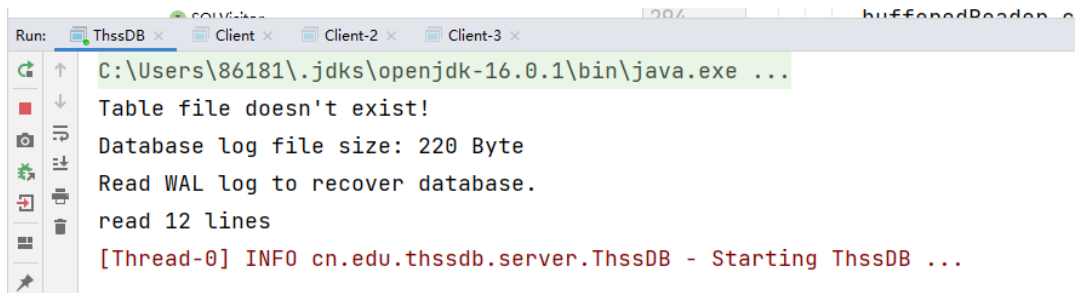
```
1 SELECT tableName1.AttrName1...,tableName2.AttrName1...tableName3.AttrName1...FROM
   tableName1 JOIN tableName2 JOIN tableName3 ON tableName1.attrName1 =
   tableName2.attrName2... [ WHERE attrName1 = attrValue ]...
```

事务部分

- 单个session的事务
 - 使用begin transaction开启事务
 - 开启事务后会在命令行有(in transaction)标识, 提示处于事务状态
 - 使用commit提交事务
 - 只支持在事务中使用insert, delete, update, select这四种语句
- 示例:

```
connect success!
sessionId is 1
It costs 4981 ms.
ThssDB>use test;
Successfully switch to database: test
It costs 87 ms.
ThssDB>begin transaction;
successfully begin transaction
It costs 3 ms.
ThssDB>(in transaction)insert into person values(10);
Successfully inserted data into the table: person in database: test
It costs 3 ms.
ThssDB>(in transaction)commit;
Successfully commit
It costs 3 ms.
ThssDB>quit;
Successfully quited.
```

- 单个session的WAL机制恢复
 - 系统因故障而崩溃后, 重启server时会自动读取data下的文件, 重新读取.data中记载的数据库及表格信息, 执行.log中记录的指令, 进行自动恢复:



```
Run: ThssDB x Client x Client-2 x Client-3 x
C:\Users\86181\jdk\openjdk-16.0.1\bin\java.exe ...
Table file doesn't exist!
Database log file size: 220 Byte
Read WAL log to recover database.
read 12 lines
[Thread-0] INFO cn.edu.thssdb.server.ThssDB - Starting ThssDB ...
```

- 多个session的事务并发
 - 可以通过add configuration同时开启多个客户端，连接到服务器
 - 采用共享锁和排他锁，实现read committed隔离级别
 - 当一个事务获取了某表的写锁时，其它希望对该表进行读写操作的事务将进入队列进行繁忙等待，在该事务被提交后才会释放写锁，从阻塞队列中按顺序执行；当一个事务只对表进行读操作时，读后并不影响其它事务读取。