浙江大学

本科实验报告

课程名称：	大规模信息系统构建技术导论
实验名称：	分布式 MiniSQL
姓　　名：	李毅桐、王粤龙、米博宇
学　　院：	计算机学院
系：	软件工程系
专　　业：	软件工程
学　　号：	3200102888
指导教师：	鲍凌峰

2023 年　5　月　15　日

浙江大学实验报告

实验名称：分布式 MiniSQL 其他模块 实验类型：设计实验

同组学生：李毅桐，王粤龙，米博宇 实验地点：

一、实验内容

- 设计分布式 MiniSQL 的其他模块：
- WorkerRunnable 类作为服务器端的多线程封装。
- DbOperations 类封装了支持的所有数据库操作函数，并负责直接与 MySQL 数据库进行交互。

二、功能描述

1. WorkerRunnable

实现了 Runnable 接口，作为 Region Server 的多线程包装。用于在同一进程中创建多个 Region Server 线程，用于模拟分布式效果。

2. DbOperations

DbOperations 类定义了执行查询、新建/删除数据表、更新/插入/删除数据表的操作函数，并与实际的 MySQL 数据库建立连接。这些函数可以被 Master 和 Region 服务器直接调用，进行数据库操作并获得结果。

三、接口说明

DbOperations 类提供了 runSelect，runUpdate，insertTableData 接口，分别支持查询、删除/新建/更改数据表。getTableNames 函数用于获取一个 Region 服务器上的所有数据表。

四、工作原理

1. runSelect 函数

该函数接受一个查询的 SQL 语句。首先在 MySQL 连接上执行这个 SQL 语句并

得到结果。然后将结果构造为一个 List<Map<String, Object>>类型的结果。结果 List 的每个元素代表查询结果的一行，Map 的 key 为列名，value 为该列的值，分别为 String 和 Object 类型。最后将查询结果序列化为字符串并返回。

2. runUpdate 函数

接受一个创建/删除数据表和更改数据表的 SQL 语句。直接在在 MySQL 连接上执行这个 SQL 语句，并返回执行结果。如果操作为更改数据表，执行结果需要附加影响的行数。

3. getTableNames 函数

直接执行 SHOW TABLES 语句，并将所有表名构造为 List 后返回。

4. insertTableData 函数

该函数接受一个需要插入的数据表名和所有需要插入的行，行的类型为 List<Map<String, Object>>（含义同上）。构造一个 INSERT 的 prepare Statement，然后将参数中的具体值逐个写入 prepareStatement。执行 prepareStatement 即可完成插入。

5. runShow 函数

使用 SQL 的 SHOW CREATE TABLE 操作获取数据表的 DDL，用于复制操作。

五、单元测试

测试在名为 id_student 的数据表中进行，其 DDL 为

```
CREATE TABLE `id_student` (
  `Id` int DEFAULT NULL,
  `Name` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

insertTableData 函数

输入：构造一个 List<Map<String, Object>>作为参数传入 insertTableData 函数，作为要插入的记录。
预期输出：无输出，数据库中该记录被成功插入。

实际输入：

```
List<Map<String, Object>> tabledata = new ArrayList<>();
Map<String, Object> row1 = new HashMap<String, Object>();
row1.put("Id", 1);
row1.put("Name", "Alice");
tabledata.add(row1);
insertTableData(connection, "id_student", tabledata);
```

输出：
无输出，但记录被成功插入。

```
mysql> select * from id_student;
+------+-------+
| Id   | Name  |
+------+-------+
|    1 | Alice |
+------+-------+
```

是否通过：通过

runUpdate 函数

建表

输入 SQL：“CREATE TABLE id_student (Id INTEGER, Name VARCHAR(255))”
预期输出：“execute success!”，数据库中出现 id_student 表
实际输出：“execute success!”，查看数据库发现 id_student 表出现
是否通过：通过

更改

参数：“UPDATE id_student SET Name=“Bob” WHERE Id=1”
预期输出：“1 rows of table are affected.”,表中记录被修改
实际输出：“1 rows of table are affected.”,查看数据库，发现 id_student 表中记录的 Name 被修改为 Bob
是否通过：通过

删除

参数：“DELETE. FROM id_student WHERE Id=1”
预期输出：“1 rows of table are affected.”,表中记录被删除
实际输出：“1 rows of table are affected.”,查看数据库，发现 id_student 表中记录被删除。

是否通过：通过

删除表

参数：“DROP TABLE id_student”
预期输出：“execute success!”，id_student 被删除
实际输出：“execute success!”,查看数据库，发现 id_student 表被删除。
是否通过：通过

runShow 函数

参数：“show create table id_student”
预期输出：id_student 表的 DDL
实际输出：

```
CREATE TABLE `id_student` (
  `Id` int DEFAULT NULL,
  `Name` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

是否通过：通过

runSelect 函数

此时 id_student 表中信息：

```
mysql> select * from id_student;
+------+-------+
| Id   | Name  |
+------+-------+
|    1 | Alice |
|    2 | Bob   |
+------+-------+
```

参数：“SELECT * FROM id_student”
预期输出：含有 id_student 表的查询结果的自定义消息
实际输出：
SELECT##Id: 1, Name: Alice,
Id: 2, Name: Bob,
是否通过：通过

getTableNames 函数

输入：无
预期输出：数据库内所有数据表的名称组成的列表
实际输出：与预期一致，完整显示了所有数据表名
是否通过：通过

六、开发心得

这个模块用到了大量的 JDBC 操作和函数，让我学到了一些开发经验。此外，由于这个模块为多个其他模块提供接口，我们选择了共同开发的方式，每个人都参与了部分工作。而这一部分的函数由于大多是向外提供的接口，也让我们体会到了接口设计的重要性。