《软件架构与中间件》实验指导书

——实验三：数据分库分表的设计与实现

➢ 实验 1 Mycat 使用

1.1 基本开发环境准备

- 安装 MySQL Server 5.5 及以上
- 安装 jdk1.6 以上（建议使用 1.7 或 1.8）
- 安装 Mycat
  - 下载地址 http://dl.mycat.io/1.6.7.1/
  - 下载完成后解压到合适位置

1.2 Mycat 目录说明

/bin ：启动目录

/conf ：配置目录存放配置文件

--server.xml：是 Mycat 服务器参数调整和用户授权的配置文件。

--schema.xml：是逻辑库定义和表以及分片定义的配置文件。

--rule.xml： 是分片规则的配置文件，分片规则的具体一些参数信息单独存放为文件，也在这个目录下，配置文件修改需要重启 MyCAT。

1.3 Mycat 配置

- Mycat 系统参数配置 （/conf/server.xml）

所有的 Mycat 参数变量都是配置在 server.xml 文件中，system 标签下配置所有的参数，如果需要配置某个变量添加相应的配置即可，例如添加启动端口 8066，默认为 8066。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:server SYSTEM "server.dtd">
<mycat:server xmlns:mycat="http://io.mycat/">
    <system>
        <property name="serverPort">8066</property>
        <!-- 0为需要密码登陆、1为不需要密码登陆 ,默认为0，设置为1则需要指定默认账户-->
        <property name="nonePasswordLogin">0</property>
        <property name="useHandshakeV10">1</property>
    </system>
    <!-- 设置登陆 Mycat 的用户名,密码,逻辑库  -->
    <user name="root" defaultAccount="true">
        <property name="password">123456</property>
        <property name="schemas">TESTDB</property>
    </user>
</mycat:server>
```

- 逻辑库、表分片配置 （/conf/schema.xml）

Mycat 作为一个中间件，实现 MySQL 协议，那么对前端应用连接来说就是一个数据库，也就有数据库的配置，Mycat 的数据库配置是在 schema.xml 中配置，配置好后映射到 server.xml 里面的用户就可以了。

注意：WriteHost 中 url, username, password 按照自己实际物理数据库配置填写。

```xml
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
    <!-- 配置逻辑库 TESTDB  -->
    <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">
        <!-- 逻辑表配置，name 代表表名，dataNode 代表对应的分片 -->
        <table name="users" primaryKey="id"  dataNode="dn1" />
    <table name="item" primaryKey="id" dataNode="dn2,dn3" rule="rule1" />

    </schema>
    <!-- 设置 dataNode 对应的数据库,及 mycat 连接的地址 dataHost -->
    <!-- 配置分片(dataNode) -->
    <!-- 表切分后需要配置映射到哪几个数据库中，Mycat 的分片实际上就是库的别名 -->
    <!-- 例如上面例子配置了三个分片 dn1,dn2,dn3 分别对应到物理机映射 dataHost localhost1 的两个库上 -->
    <dataNode name="dn1" dataHost="localhost1" database="db1" />
```

```
    <dataNode name="dn2" dataHost="localhost1" database="db2" />
    <dataNode name="dn3" dataHost="localhost1" database="db3" />
    <!-- 配置物理库分片映射 -->
    <!-- Mycat 作为数据库代理需要逻辑库，逻辑用户，表切分后需要配置分片，分片也就需要映射到真实的物理主机上 -->
    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
              writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">
        <!-- heartbeat 标签代表 Mycat 需要对物理库心跳检测的语句 -->
        <heartbeat>select user()</heartbeat>
        <!-- writeHost 此标签代表 一个逻辑主机（dataHost）对应的后端的物理主机映射 -->
        <writeHost host="hostM1" url="localhost:3306" user="root" password="123456" />
    </dataHost>
</mycat:schema>
```

- Mycat 表切分规则配置 （/conf/rule.xml）

数据切分中作为表切分规则中最重要的配置，表的切分方式决定了数据切分后的性能好坏，因此也是最重要的配置。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:rule SYSTEM "rule.dtd">
<mycat:rule xmlns:mycat="http://io.mycat/">
    <!-- name 为 schema.xml 中 table 标签中对应的 rule="name1" ,也就是配置表的分片规则 -->
    <!-- columns 是表的切分字段 -->
    <!-- algorithm 是规则对应的切分规则：映射到 function 的 name -->
    <tableRule name="rule1">
        <rule>
            <columns>id</columns>
            <algorithm>func1</algorithm>
        </rule>
    </tableRule>
    <!-- function 配置是分片规则的配置 -->
    <!-- name 为切分规则的名称，名字任意取，但是需要与 tableRule 中匹配 -->
    <!-- class 是切分规则对应的切分类，写死，需要哪种规则则配置哪种 -->
    <!-- property 标签是切分规则对应的不同属性，不同的切分规则配置不同 -->
    <function name="func1" class="io.mycat.route.function.PartitionByLong">
        <property name="partitionCount">2</property>
        <property name="partitionLength">512</property>
    </function>
```

```
</mycat:rule>
```

1.4 物理数据库配置

- 登录物理数据库 MySQL，创建数据库与表

```
drop database if exists db1;
create database db1;
use db1;
 CREATE TABLE users (
    id INT NOT NULL AUTO_INCREMENT,
    name varchar(50) NOT NULL default '',
    indate DATETIME NOT NULL default CURRENT_TIMESTAMP,
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;

drop database if exists db2;
create database db2;
use db2;
 CREATE TABLE item (
    id INT NOT NULL AUTO_INCREMENT,
    value INT NOT NULL default 0,
    indate DATETIME NOT NULL default CURRENT_TIMESTAMP,
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;

drop database if exists db3;
create database db3;
use db3;
CREATE TABLE item (
    id INT NOT NULL AUTO_INCREMENT,
    value INT NOT NULL default 0,
    indate DATETIME NOT NULL default CURRENT_TIMESTAMP,
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

1.5 启动 Mycat 服务，测试插入数据

- 运行 Mycat

linux:

```
./mycat start # 启动
```

```
./mycat stop # 停止
./mycat console # 前台运行
./mycat restart # 重启服务
./mycat pause # 暂停
./mycat status # 查看启动状态
```

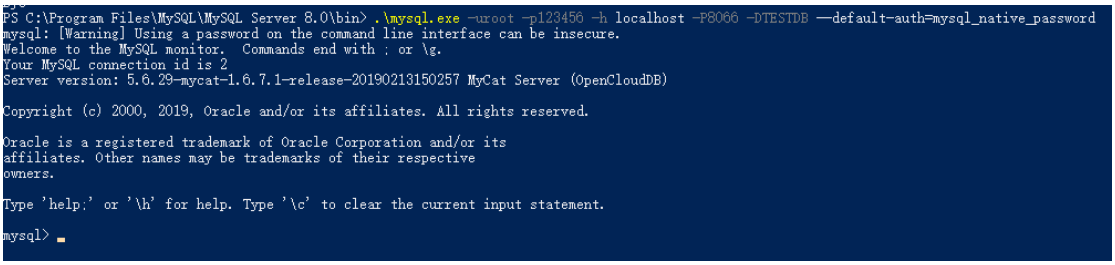
windows:

直接运行 Mycat 安装目录下/bin/startup_nowrap.bat，如果出现闪退，在 cmd 命令行运行，查看出错原因。

提示 MyCAT Server startup successfully. 则启动成功。

- 连接 Mycat

```
.\mysql.exe -uroot -p123456 -h localhost -P8066 -DTESTDB --default-auth=mysql_native_password
```



- 插入数据

连接 Mycat 后测试插入三条数据：

```
mysql> use TESTDB;
Database changed
mysql> show tables;
+------------------+
| Tables in TESTDB |
+------------------+
| item             |
| users            |
+------------------+
2 rows in set (0.00 sec)

mysql> insert into users(name) values('zhangsan');
Query OK, 1 row affected (0.12 sec)
```

```
mysql> insert into item(id, value) values(1, 100);
Query OK, 1 row affected (0.09 sec)

mysql> insert into item(id, value) values(512, 100);
Query OK, 1 row affected (0.07 sec)

mysql> explain insert into users(name) values('zhangsan');
+-----------+-------------------------------------------+
| DATA_NODE | SQL                                       |
+-----------+-------------------------------------------+
| dn1       | insert into users(name) values('zhangsan') |
+-----------+-------------------------------------------+
1 row in set (0.00 sec)

mysql> explain insert into item(id, value) values(1, 100);
+-----------+-------------------------------------------------+
| DATA_NODE | SQL                                             |
+-----------+-------------------------------------------------+
| dn2       | INSERT INTO item (id, value) VALUES ('1', 100) |
+-----------+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> explain insert into item(id, value) values(512, 100);
+-----------+---------------------------------------------------+
| DATA_NODE | SQL                                               |
+-----------+---------------------------------------------------+
| dn3       | INSERT INTO item (id, value) VALUES ('512', 100) |
+-----------+---------------------------------------------------+
1 row in set (0.00 sec)
```

然后登录物理数据库，查看是否插入成功：

```
mysql> use db1;
Database changed
mysql> select * from users;
+----+----------+---------------------+
| id | name     | indate              |
+----+----------+---------------------+
| 11 | zhangsan | 2019-04-14 13:44:22 |
+----+----------+---------------------+
1 row in set (0.07 sec)

mysql> use db2;
```

```
Database changed
mysql> select * from item;
+----+-------+---------------------+
| id | value | indate              |
+----+-------+---------------------+
|  1 |   100 | 2019-04-14 13:44:35 |
+----+-------+---------------------+
1 row in set (0.06 sec)

mysql> use db3;
Database changed
mysql> select * from item;
+-----+-------+---------------------+
| id  | value | indate              |
+-----+-------+---------------------+
| 512 |   100 | 2019-04-14 13:44:49 |
+-----+-------+---------------------+
1 row in set (0.06 sec)
```

插入的 users 表中的数据全部在 db1 中，而 item 表中的数据分布在 db2 和 db3 中。这样就根据实际的路由策略进行了分表。

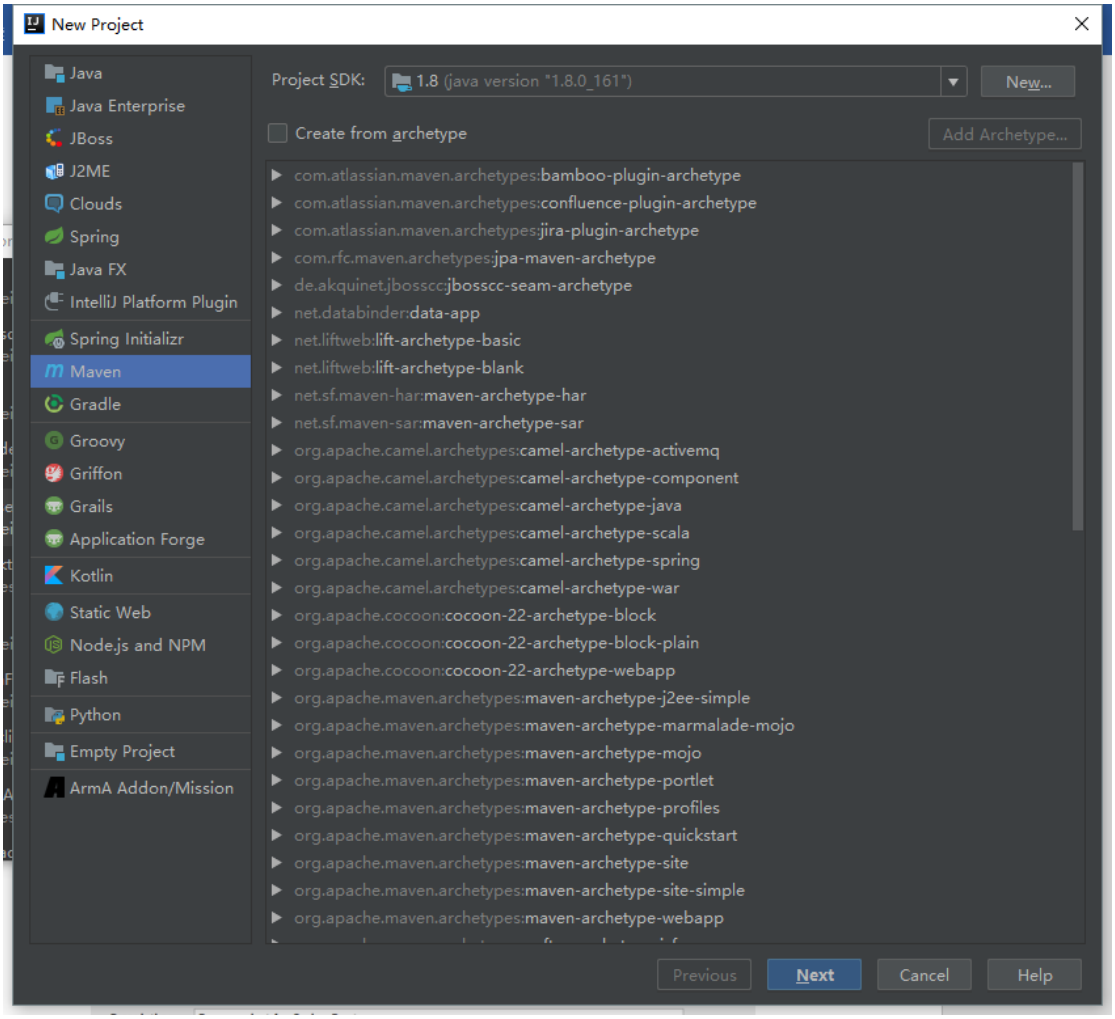
➢ 实验 2 Sharding-JDBC 使用

2.1 基本开发环境准备

- 安装 IntelliJ IDEA，并注册学生账户以获得免费使用权。
- 安装 jdk1.6 以上（建议使用 1.7 或 1.8）
- 安装 MySQL Server 5.5 及以上

2.2 项目创建

- 打开 idea，选择 Create New Project
- 项目配置，选择 Maven，Project SDK 选择 1.8，选择 Next



- 继续项目配置，选择 Next，选择默认项直到 Finish 生成项目。

- 配置依赖，修改 pom.xml，修改后内容如下：

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <project xmlns="http://maven.apache.org/POM/4.0.0"
3.          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
   apache.org/xsd/maven-4.0.0.xsd">
5.     <modelVersion>4.0.0</modelVersion>
6.
7.     <groupId>com.hit</groupId>
```

```
    <artifactId>shardingjdbcDemo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>

        <dependency>
            <groupId>io.shardingsphere</groupId>
            <artifactId>sharding-jdbc-core</artifactId>
            <version>3.1.0</version>
        </dependency>
    </dependencies>
</project>
```

配置好后等待 maven 导入依赖。

2.3 物理数据库配置

- 登录物理数据库 MySQL，创建数据库与表

```
drop database if exists db1;
create database db1;
use db1;
 CREATE TABLE users (
    id INT NOT NULL AUTO_INCREMENT,
    name varchar(50) NOT NULL default '',
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;

drop database if exists db2;
create database db2;
use db2;
 CREATE TABLE item (
    id INT NOT NULL AUTO_INCREMENT,
```

```
    value INT NOT NULL default 0,
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;

drop database if exists db3;
create database db3;
use db3;
CREATE TABLE item (
    id INT NOT NULL AUTO_INCREMENT,
    value INT NOT NULL default 0,
    PRIMARY KEY (id)
)AUTO_INCREMENT= 1 ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2.4 程序编写

- 按照以下代码编写文件/src/main/java/Demo.java

```
import io.shardingsphere.shardingjdbc.api.yaml.YamlShardingDataSourceFactory;

import javax.sql.DataSource;
import java.io.File;
import java.io.IOException;
import java.sql.*;

public class Demo {
    public static void main(String[] args) throws SQLException, IOException {
        File conf = new File("./src/main/resources/conf.yml");
        DataSource dataSource = YamlShardingDataSourceFactory.createDataSource(conf);
        Connection conn = dataSource.getConnection();
        Statement stmt = conn.createStatement();

        System.out.println(stmt.executeUpdate("insert into users(id, name) values(1, 'zhangsan')"));
        System.out.println(stmt.executeUpdate("insert into item(id, value) values(1, 100)"));
        System.out.println(stmt.executeUpdate("insert into item(id, value) values(2, 200)"));
    }
}
```

- 按照以下代码编写文件/src/main/resources/conf.yml

按照自己的实际数据库配置填写 url, username, password 属性

```
dataSources:  # 配置数据源列表,必须是有效的 jdbc 配置,目前仅支持 MySQL 与 PostgreSQL
  db1: !!org.apache.commons.dbcp.BasicDataSource  # 数据源名称
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db1  # 这里的要求合法的 jdbc 连接串即可,目前尚未兼容 MySQL 8.x
    username: root  # MySQL 用户名
    password: 123456  # MySQL 用户的明文密码
  db2: !!org.apache.commons.dbcp.BasicDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db2
    username: root
    password: 123456
  db3: !!org.apache.commons.dbcp.BasicDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db3
    username: root
    password: 123456

shardingRule:  # sharding 的配置
  tables:  # 配置表 sharding 的主要位置
    users:
      actualDataNodes: db1.users  # sharidng 表对应的数据源以及物理名称,需要用表达式处理,表示表实际上在哪些数据源存在
    item:
      actualDataNodes: db${2..3}.item  # 表示存在 db2.item, db3.item
      databaseStrategy:  # sharding 规则
        inline:
          shardingColumn: id  # 列名
          algorithmExpression: db${id % 2 + 2}  # 例如：id=1 时表示 db3
```

2.5 程序运行

- 运行 Demo.main()，成功后登录物理数据库查看是否插入成功。

```
mysql> use db1;
Database changed
mysql> select * from users;
+----+----------+
| id | name     |
+----+----------+
|  1 | zhangsan |
+----+----------+
1 row in set (0.00 sec)

mysql> use db2;
Database changed
mysql> select * from item;
+----+-------+
| id | value |
+----+-------+
|  2 |   200 |
+----+-------+
1 row in set (0.01 sec)

mysql> use db3;
Database changed
mysql> select * from item;
+----+-------+
| id | value |
+----+-------+
|  1 |   100 |
+----+-------+
1 row in set (0.00 sec)
```