《软件架构与中间件》实验指导书

——实验3：数据层架构实验

实验目的

1）学习使用Mycat和Sharding-JDBC实现数据分库分表
2）学习使用Redis数据库实现数据缓存
3）能够灵活应用Mycat或Sharding-JDBC实现分库分表架构到实际系统
4）能够灵活应用Redis实现数据缓存架构到实际系统

实验要求

1）2人结对成组
2）实验3.1、3.2、3.3均为必做
3）结合《软件过程与工具》课程中进销存系统(或其他实际软件系统)进行数据层架构重构，实现根据业务垂直划分的数据库分库分表；面向海量数据带来的数据检索慢问题，实现数据库水平分片，达到数据检索的性能提升；利用缓存架构实现数据读取的性能提升。
4）应给出关键过程的细节。

实验3. 1 Mycat数据库分库分表实验

1. 基本开发环境准备
   - 安装 MySQL Server 5.5及以上

注：“mycat安装目录/lib”中默认放置的是MySQL5.0的JDBC驱动(mysql-connector-java-5.x.x.jar)，如果使用8.0以上版本，下载8.0驱动(https://dev.mysql.com/downloads/connector/j/ 选择platform independent)，删除5.0驱动，替换驱动文件。

   - 安装jdk1.8以上（建议使用jdk11）
   - 安装 Mycat
     - 下载地址http://dl.mycat.org.cn/1.6.7.5/2020-4-10/
     - 选择自己计算机对应的版本
     - 下载完成后解压到合适位置
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

配置文件修改需要重启Mycat。

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。

Server.xml示例：

```
<?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按照实际物理数据库配置填写。

```
<?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
的三个库上注：本例中采用在一个MySQL服务器上创建三个数据库的形式模拟分片；实际应用中可以
在多个服务器上配置多个数据库，修改dataHost的地址和database名称即可-->
    <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。

```
<!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>
```

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;
```

5. 启动Mycat 服务，测试插入数据

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。详细信息查看/log目录中的日志文件。

提示MyCAT Server startup successfully则启动成功。

- 连 接 Mycat

```
.\mysql.exe -uroot -p123456 -h 127.0.0.1 -P8066 -DTESTDB
```

- 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(id,name) values(11,'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)
```

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。

6. 在学会上述基本操作基础上，回答下述问题。

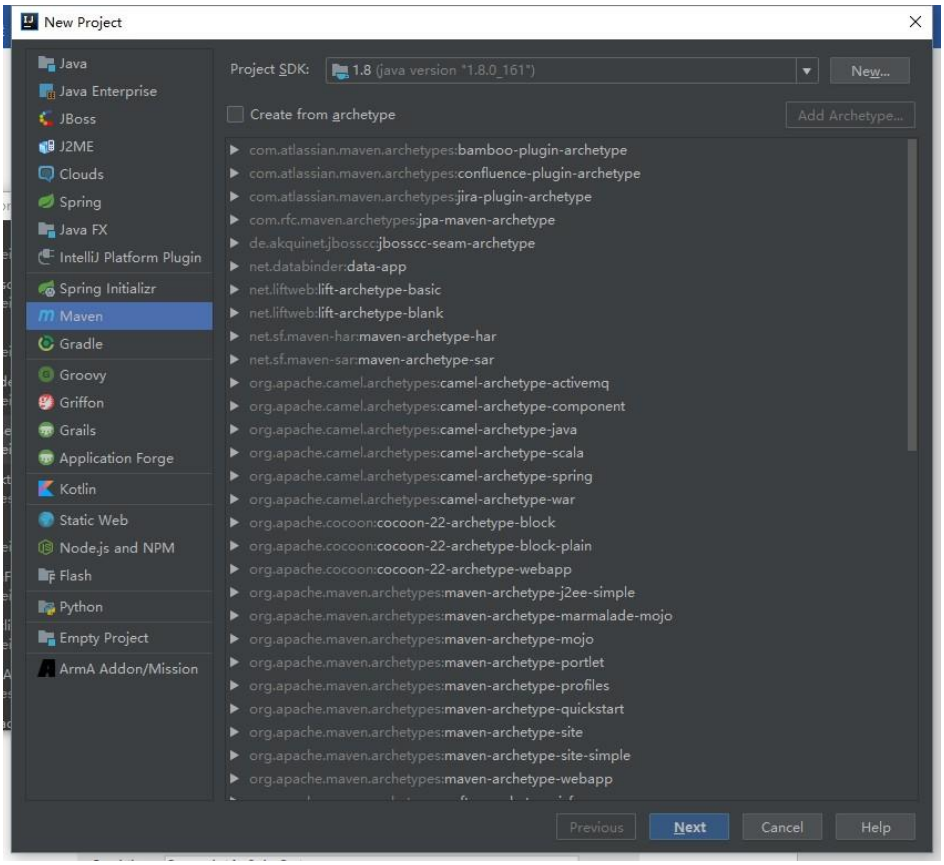
1）请给出Mycat配置安装过程中遇到的问题和解决方案。

2）请详析Mycat的分库分表原理和操作方法。

3）请在进销存系统(或其他实际软件系统)创建具有复杂表结构和含有较大数据量的数据库表，并基于此库表描述分库分表的结果，且验证分库分表的效果。

实验3.2 Sharding-JDBC数据库分库分表实验

1. 基本开发环境准备
   - 安装Java编程IDE，推荐Eclipse或者IntelliJ IDEA(社区版，或者注册学生账户获得专用版免费使用权)。
   - 安装jdk1.8以上（建议使用jdk11）
   - 安装 MySQL Server 5.5 及以上
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，其中外部类库依赖修改后内容如下：

```
 <dependencies>
        <dependency>
            <groupId>org.apache.shardingsphere</groupId>
            <artifactId>sharding-jdbc-core</artifactId>
            <version>4.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-dbcp2</artifactId>
            <version>2.8.0</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.25</version>
        </dependency>
    </dependencies>
```

配置好后等待 maven 导入依赖。

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 db4;
create database db4;
use db4;
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 db5;
create database db5;
use db5;
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 db6;
create database db6;
use db6;
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;
```

4. 程序编写

- 按照以下示例代码编写文件/src/main/java/Demo.java

```
import
org.apache.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。注意下面文件中的缩进。

```
dataSources: # 配置数据源列表,必须是有效的jdbc配置
  db4: !!org.apache.commons.dbcp2.BasicDataSource # 数据源名称
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db4
    username: root  # MySQL 用户名
    password: a12345  # MySQL用户的明文密码

  db5: !!org.apache.commons.dbcp2.BasicDataSource # 数据源名称
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db5
    username: root  # MySQL 用户名
    password: a12345  # MySQL 用户的明文密码

  db6: !!org.apache.commons.dbcp2.BasicDataSource # 数据源名称
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db6
    username: root  # MySQL 用户名
    password: a12345  # MySQL 用户的明文密码

shardingRule: # sharding 的配置
  tables:  # 配置表sharding的主要位置
    users:
      actualDataNodes: db4.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${5..6}.item # 表示存在 db5.item,db6.item
      databaseStrategy: # sharding 规 则
        inline:
          shardingColumn: id # 列 名
          algorithmExpression: db${id % 2 + 5}  # 例如: id=1 时表示 db6
```

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 db4;
Database changed
mysql> select * from users;
+----+----------+
| id | name     |
+----+----------+
|  1 | zhangsan |
+----+----------+
```

```
1 row in set (0.00 sec)

mysql> use db5;
Database changed
mysql> select * from item;
+----+-------+
| id | value |
+----+-------+
|  2 |   200 |
+----+-------+
1 row in set (0.01 sec)

mysql> use db6;
Database changed
mysql> select * from item;
+----+-------+
| id | value |
+----+-------+
|  1 |   100 |
+----+-------+
1 row in set (0.00 sec)
```

6. 在学会上述基本操作基础上，回答下述问题。

1）请给出Sharding-JDBC配置安装过程中遇到的问题和解决方案。

2）请详析Sharding-JDBC的分库分表原理和操作方法。

3）请在进销存系统(或其他实际软件系统)创建具有复杂表结构和含有较大数据量的数据库表，并基于此库表描述分库分表的结果，且验证分库分表的效果。

实验3.3 Redis数据缓存实验

1. 基本开发环境准备

- 安装Java编程环境。
- 安装Redis
  - 选项1：直接安装Redis（分windows版和Linux版）
  - 选项2：使用docker安装
- Redis for Windows安装（选项1）

1) 解压安装文件到任意文件夹，如C:\redis
2) 启动redis-server：

```
$ cd C:\redis
$ redis-server.exe redis.windows.conf
```

3) 使用命令行方式redis-cli，或 类似于Another Redis Desktop Manager这类GUI工具操作

```
$ cd c:\redis
$ redis-cli.exe
127.0.0.1:6379> set foo bar
OK
127.0.0.1:6379> get foo
"bar"
```

- redis for Linux/Mac latest安装（选项1）

1) 下载后解压、编译:

```
$ tar xzf redis-5.0.4.tar.gz
$ cd redis-5.0.4
$ make
```

此时可执行文件在`redis-5.0.4/src` 目录下

2) 启动redis-server：

```
$ src/redis-server
```

3) 使用redis-cli:

```
$ src/redis-cli
redis> set foo bar
OK
redis> get foo
"bar"
```

- 使用docker安装redis（选项2）

1) 安装docker

docker for Windows 下载地址 - 要求 64 位windows 10 专业版或企业版

docker for Mac 下载地址 - 要求 Mac OS Sierra 10.12 以上

docker Linux 安装命令：

```
$ curl -fsSL get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh --mirror Aliyun
```

2) 安装和启动redis

```
$ docker pull redis
```

```
$ docker run --name my-redis -p 6379:6379 redis:latest
```

3) 启动redis-cli

docker exec -ti d0b86 redis-cli -h 127.0.0.1 -p 6379 //d0b86是docker的id，可通过“docker ps”命令查看；127.0.0.1是本地IP，根据具体情况替换

2. 利用redis-cli客户端完成操作

- 数据类型：string

```
# 赋 值 set [key] [value]
 127.0.0.1:6379> set name zhangsan
OK

# 取 值 get [key]
 127.0.0.1:6379> get name
"zhangsan"

# 删 除 del [key]
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> get name
(nil)
```

- 数据类型：hash

hash 可以存储多个键值对之间的映射

```
# 赋值
 # hset [key] [field] [value]
 # hmset [key] [field1] [value1] [field2] [value2]...
# 取值
 # hget [key] [field]
 # hmget [key] [field1] [field2] ...
 127.0.0.1:6379> hset myhash name zhangsan
 (integer) 1
 127.0.0.1:6379> hget myhash name
"zhangsan"
127.0.0.1:6379> hmset myhash name zhangsan age 18 class 2
OK
127.0.0.1:6379> hmget myhash name age class
1) "zhangsan"
2) "18"
3) "2"

# 删除 hdel [key] [field]
```

```
127.0.0.1:6379> hdel myhash name
(integer) 1
127.0.0.1:6379> hget myhash name
(nil)
```

- 数据类型：list

list 的顺序是按照插入的顺序，可以在头部跟尾部插入数据

```
# lpush [key] [value1] [value2] ...
# rpush [key] [value1] [value2] ...
# 查看
# lrange [key] [start_index] [stop_index]
127.0.0.1:6379> lpush mylist a b c
(integer) 3
127.0.0.1:6379> lpush mylist 1 2 3
(integer) 6
127.0.0.1:6379> lrange mylist 0 -1
1) "3"
2) "2"
3) "1"
4) "c"
5) "b"
6) "a"
127.0.0.1:6379> rpush mylist d e f
(integer) 9
127.0.0.1:6379> lrange mylist 0 -1
1) "3"
2) "2"
3) "1"
4) "c"
5) "b"
6) "a"
7) "d"
8) "e"
9) "f"

# 指定位置添加 lset [key] [index] [value]
127.0.0.1:6379> lset mylist 3 x
OK
127.0.0.1:6379> lrange mylist 0 -1
1) "3"
2) "2"
3) "1"
4) "x"
5) "b"
```

```
6) "a"
7) "d"
8) "e"
9) "f"

# 两端弹出
# lpop [key]
# rpop [key]
127.0.0.1:6379> lpop mylist
"3"
127.0.0.1:6379> rpop mylist
```

- 数据类型：set

set 中不允许出现重复的元素，没有顺序

```
# 添加 sadd [key] [member1] [member2] ...
# 删除 srem [key] [member1] [member2] ...
# 查看 smembers [key]
 127.0.0.1:6379> sadd myset a b c 1 2 3
 (integer) 6
 127.0.0.1:6379> sadd myset a
 (integer) 0
 127.0.0.1:6379> srem myset a b c
 (integer) 3
 127.0.0.1:6379> smembers myset
 1) "1"
 2) "3"
 3) "2"
```

- 数据类型：sorted set

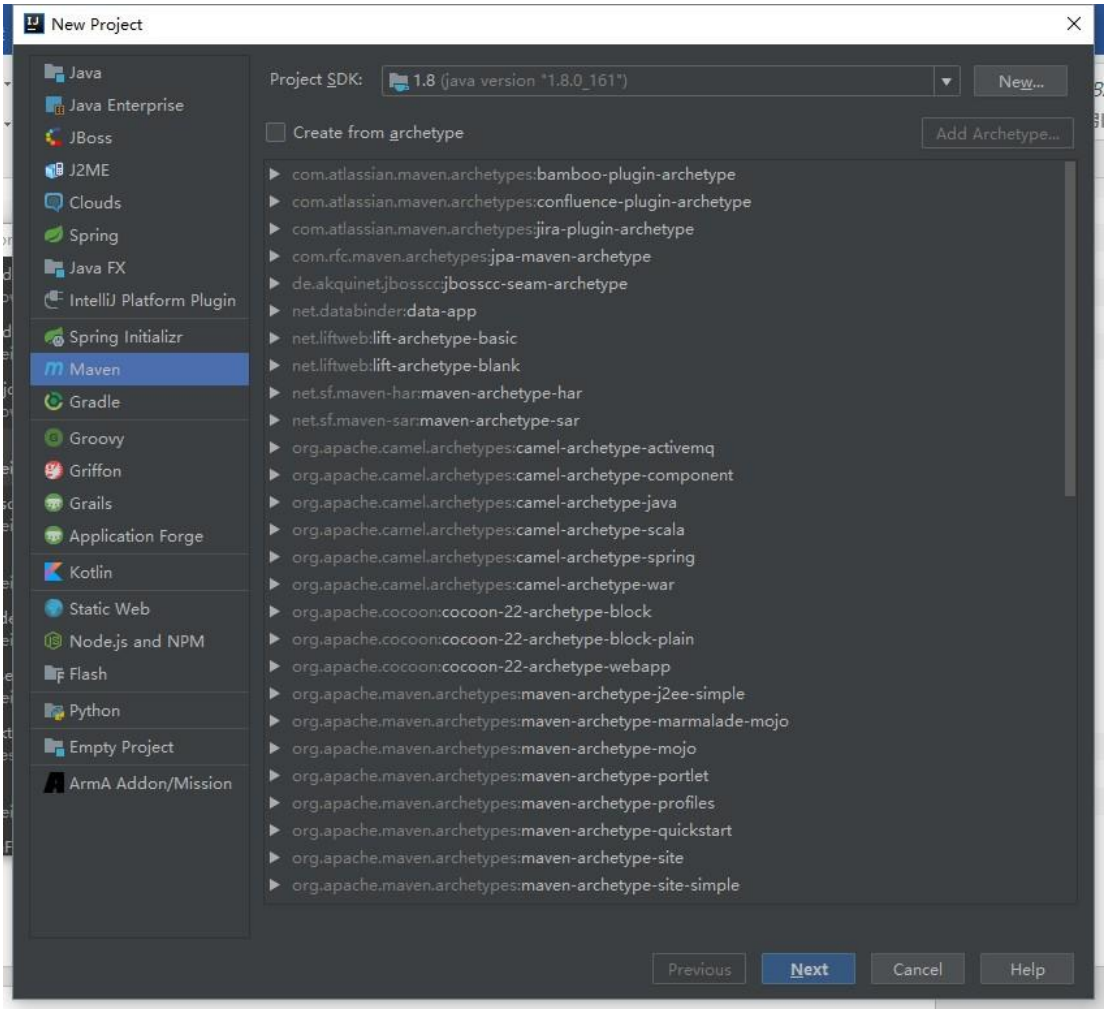
sorted set有顺序，从小到大排序

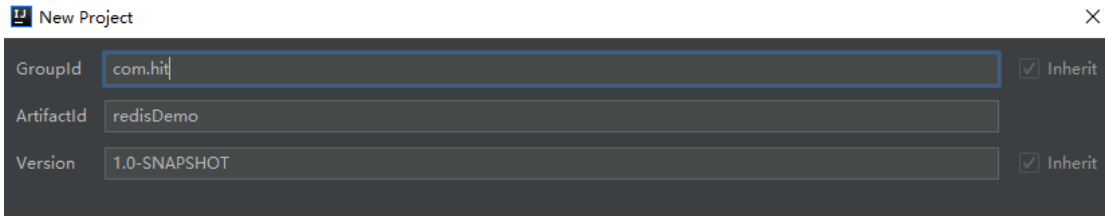
```
# 添 加 zadd [key] [score1] [member1] [score2] [member2] ...
 127.0.0.1:6379> zadd mysset 10 zhangsan 30 lisi 20 wangwu
 (integer) 3
# 查看 zrange [key] [start_index] [end_index] <withscores>
 127.0.0.1:6379> zrange mysset 0 -1
 1) "zhangsan"
 2) "wangwu"
 3) "lisi"
 127.0.0.1:6379> zrange mysset 0 -1 withscores
 1) "zhangsan"
 2) "10"
```

```
3) "wangwu"
4) "20"
5) "lisi"
6) "30"
# 查看score: zscore [key] [member]
127.0.0.1:6379> zscore mysset zhangsan
"10"
# 删除 zrem [key] [member1] [member2] ...
127.0.0.1:6379> zrem mysset zhangsan
(integer) 1
```

3. redis java client: Jedis 的使用

- 使用idea创建maven项目





- 修改 pom.xml添加jedis依赖如下：

```
<dependencies>
  <dependency>
      <groupId>redis.clients</groupId>
      <artifactId>jedis</artifactId>
      <version>3.6.0</version>
  </dependency>
</dependencies>
```

- 新建文件/src/main/java/RedisDemo.java，文件内容如下：

```
import redis.clients.jedis.Jedis;
public class RedisDemo {
    public static void main(String[] args) {
        Jedis jedis = new Jedis("127.0.0.1", 6379, 100000);
        jedis.set("foo", "bar");
        System.out.println(jedis.get("foo"));
    }
}
```

运行RedisDemo.main
输出：bar

4. 在学会上述基本操作基础上，回答下述问题。

1）请给出Redis配置安装过程中遇到的问题和解决方案。

2）请详析Redis的缓存清洗策略，数据迁移及扩容策略，面向缓存雪崩、穿透等问题的策略。

3）请在进销存系统(或其他实际软件系统)设计一个简单场景，实现缓存读写操作，缓存更新操作，给出缓存的效果，分析2问题中相关策略的效果。