课程内容介绍

1、基本概念
（1）什么是 Sharding Sphere
（2）分库分表

2、Sharding-JDBC 分库分表操作

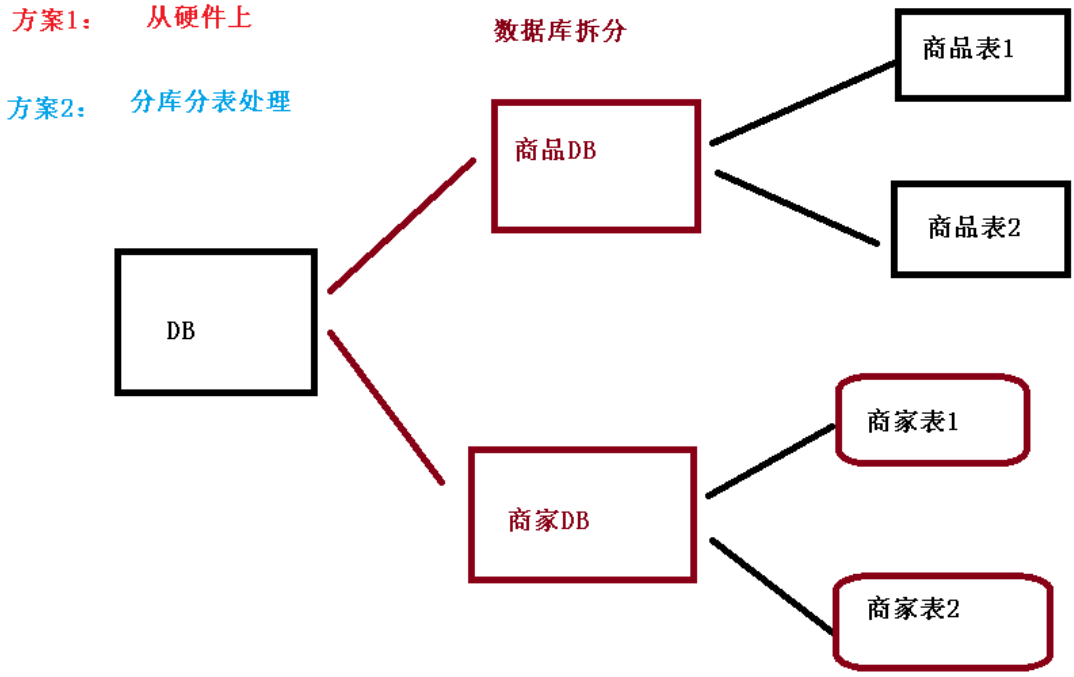
3、Sharding-Proxy 分库分表操作

什么是 ShardingSphere

1、一套开源的分布式数据库中间件解决方案
2、有三个产品：Sharding-JDBC 和 Sharding-Proxy
3、定位为关系型数据库中间件，合理在分布式环境下使用关系型数据库操作

什么是分库分表

1、数据库数据量不可控的，随着时间和业务发展，造成表里面数据越来越多，如果再去对数据库表 curd 操作时候，造成性能问题。
2、方案 1：从硬件上
3、方案 2：分库分表
* 为了解决由于数据量过大而造成数据库性能降低问题。



分库分表的方式

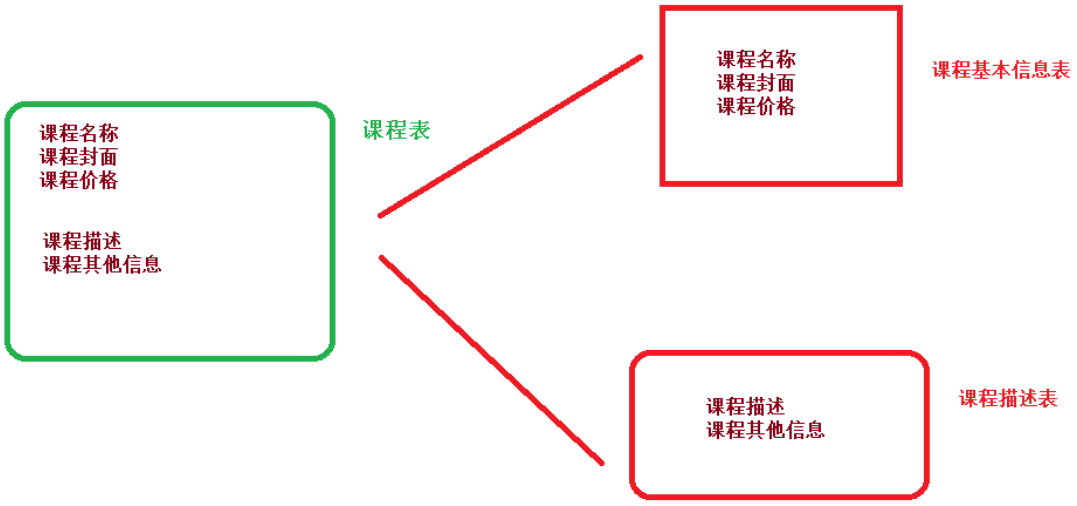
1、分库分表有两种方式：垂直切分和水平切分

2、垂直切分：垂直分表和垂直分库

3、水平切分：水平分表和水平分库

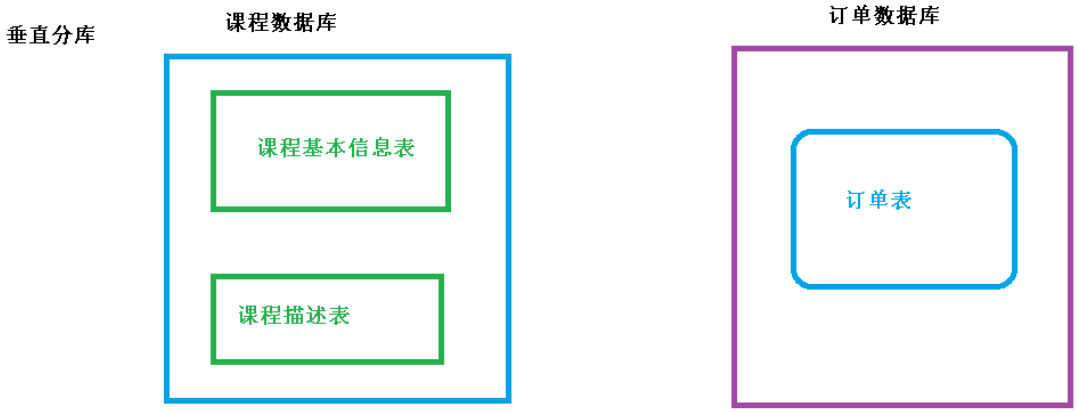
4、垂直分表

（1）操作数据库中某张表，把这张表中一部分字段数据存到一张新表里面，再把这张表另一部分字段数据存到另外一张表里面

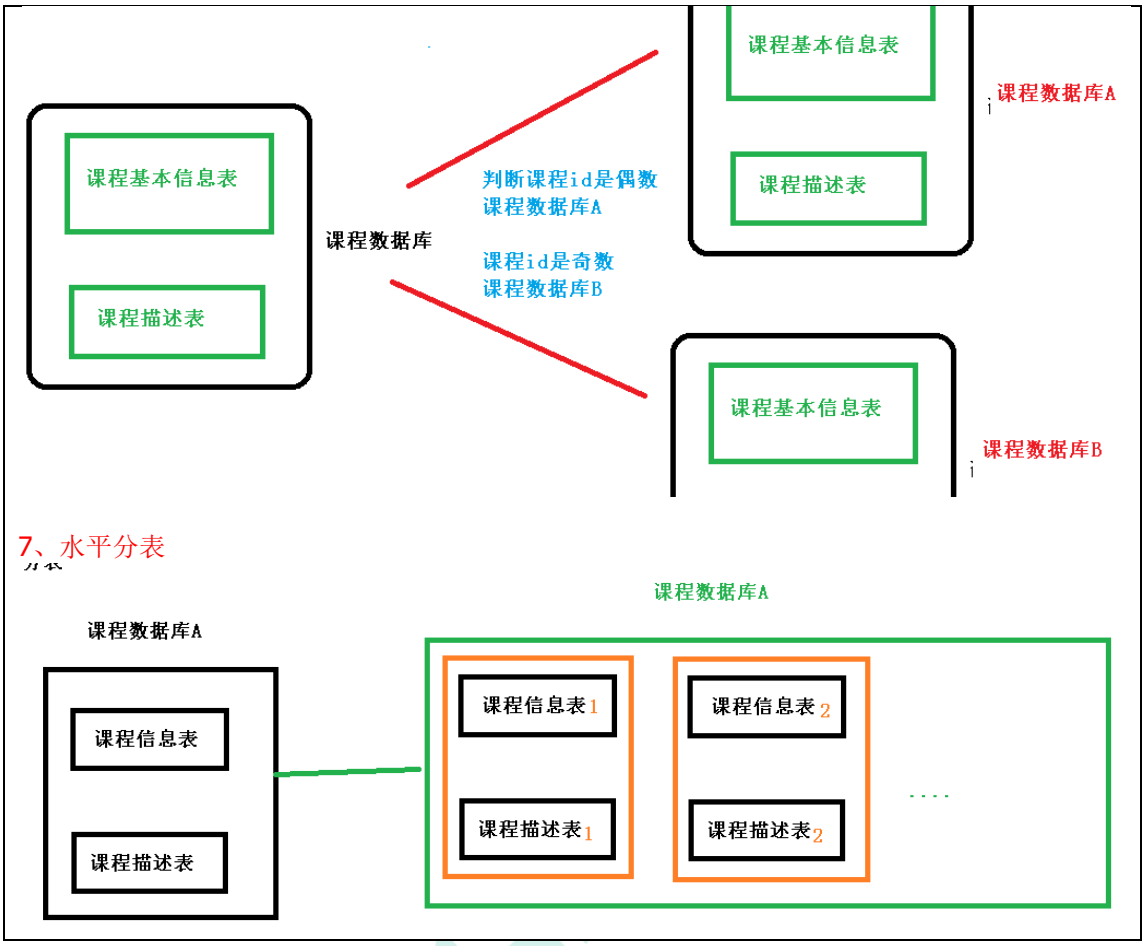


5、垂直分库

（1）把单一数据库按照业务进行划分，专库专表



6、水平分库



分库分表应用和问题

1、应用

（1）在数据库设计时候考虑垂直分库和垂直分表

（2）随着数据库数据量增加，不要马上考虑做水平切分，首先考虑缓存处理，读写分离，使用索引等等方式，如果这些方式不能根本解决问题了，再考虑做水平分库和水平分表

2、分库分表问题

（1）跨节点连接查询问题（分页、排序）

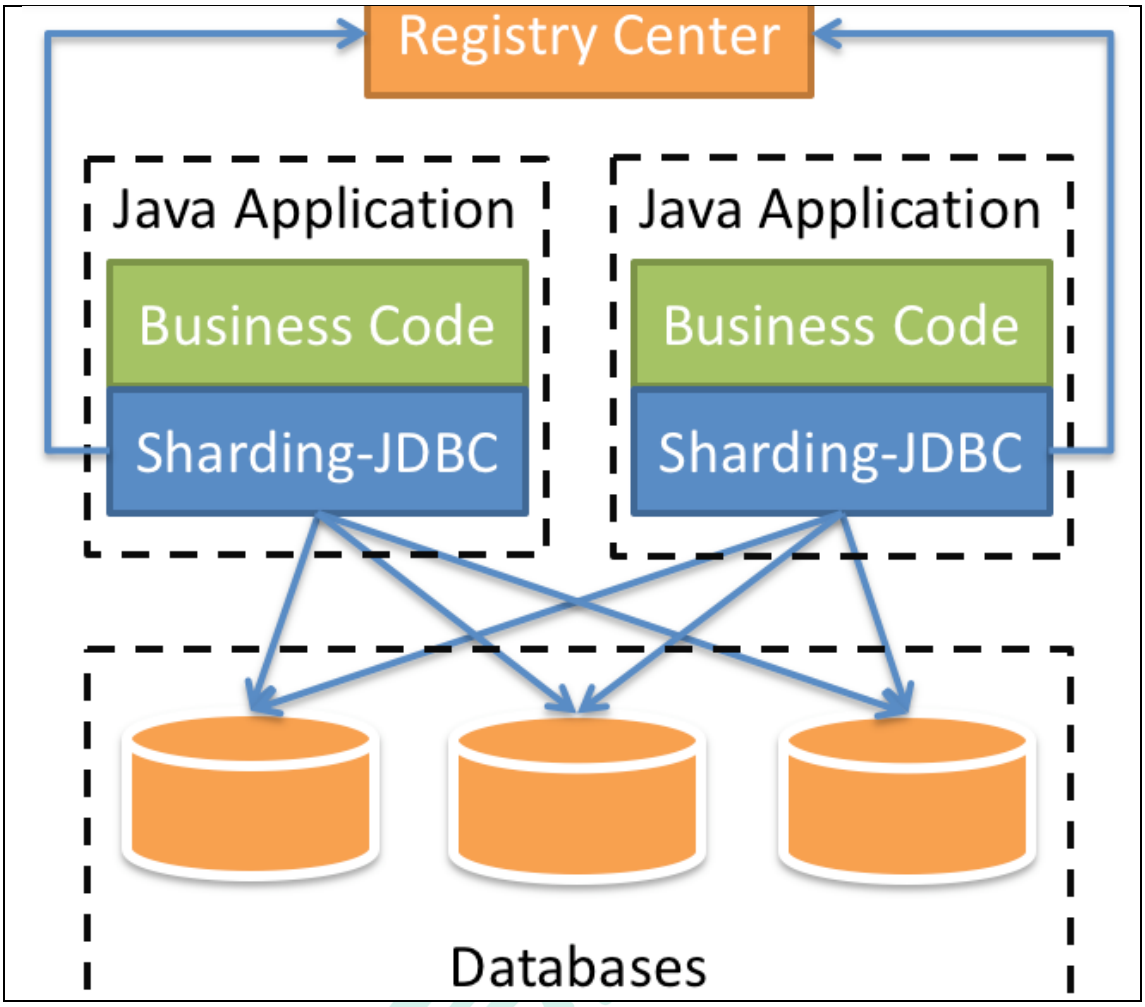
（2）多数据源管理问题

Sharding-JDBC 简介

1、是轻量级的 java 框架，是增强版的 JDBC 驱动

2、Sharding-JDBC

（1）主要目的是：简化对分库分表之后数据相关操作

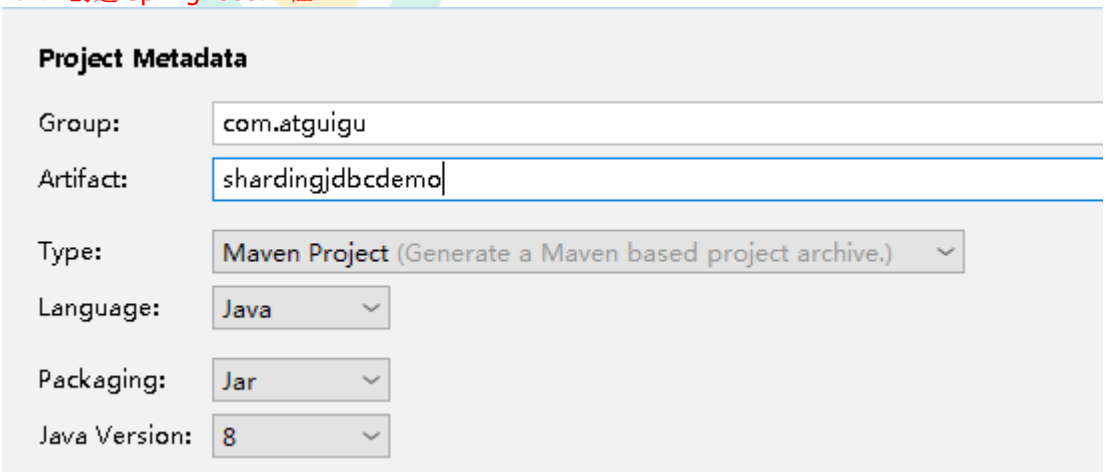


Sharding-JDBC 实现水平分表

1、搭建环境

（1）技术：SpringBoot 2.2.1+ MyBatisPlus + Sharding-JDBC + Druid 连接池

（2）创建 SpringBoot 工程



（3）修改工程 SpringBoot 版本 2.2.1

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.1.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

（4）引入需要的依赖

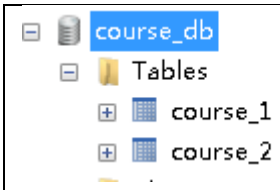
```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>

    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid-spring-boot-starter</artifactId>
        <version>1.1.20</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.shardingsphere</groupId>
        <artifactId>sharding-jdbc-spring-boot-starter</artifactId>
        <version>4.0.0-RC1</version>
    </dependency>
    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactid>
        <version>3.0.5</version>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>
</dependencies>
```

2、按照水平分表的方式，创建数据库和数据库表

（1）创建数据库 course_db

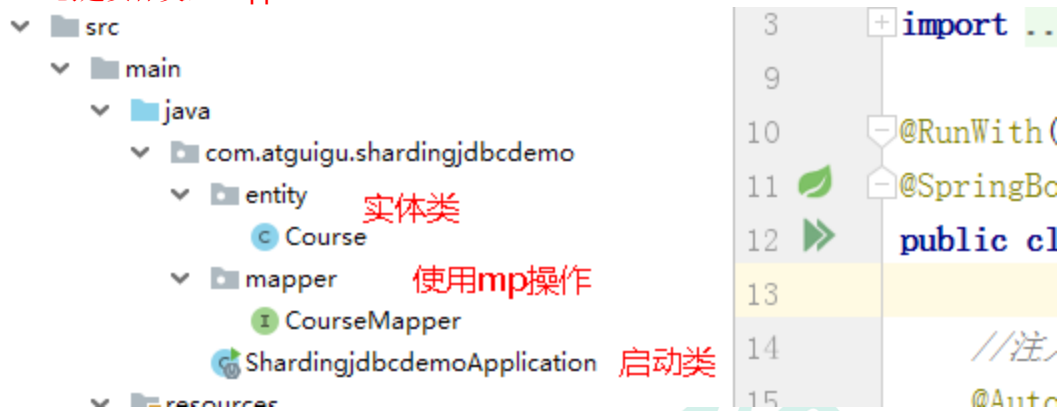
（2）在数据库创建两张表 course_1 和 course_2

（3）约定规则：如果添加课程 id 是偶数把数据添加 course_1，如果奇数添加到 course_2



3、编写代码实现对分库分表后数据的操作

（1）创建实体类，mapper



4、配置 Sharding-JDBC 分片策略

（1）在项目 application.properties 配置文件中进行配置


```
# shardingjdbc 分片策略
# 配置数据源，给数据源起名称
spring.shardingsphere.datasource.names=m1

# 一个实体类对应两张表，覆盖
spring.main.allow-bean-definition-overriding=true

#配置数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m1.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m1.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m1.url=jdbc:mysql://localhost:3306/course_db?
serverTimezone=GMT%2B8
spring.shardingsphere.datasource.m1.username=root
```

```
spring.shardingsphere.datasource.m1.password=root

#指定 course 表分布情况，配置表在哪个数据库里面，表名称都是什么  m1.course_1 ，m1.course_2
spring.shardingsphere.sharding.tables.course.actual-data-nodes=m1.course_$->{1..2}

# 指定 course 表里面主键 cid 生成策略  SNOWFLAKE
spring.shardingsphere.sharding.tables.course.key-generator.column=cid
spring.shardingsphere.sharding.tables.course.key-generator.type=SNOWFLAKE

# 指定分片策略  约定 cid 值偶数添加到 course_1 表，如果 cid 是奇数添加到 course_2 表
spring.shardingsphere.sharding.tables.course.table-strategy.inline.sharding-column=cid
spring.shardingsphere.sharding.tables.course.table-strategy.inline.algorithm-expression=course_$->{cid % 2 + 1}

# 打开 sql 输出日志
spring.shardingsphere.props.sql.show=true
```

5、编写测试代码

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class ShardingjdbcdemoApplicationTests {

    //注入 mapper
    @Autowired
    private CourseMapper courseMapper;

    //添加课程的方法
    @Test
    public void addCourse() {
        for(int i=1;i<=10;i++) {
            Course course = new Course();
            course.setCname("java"+i);
            course.setUserId(100L);
            course.setCstatus("Normal"+i);
            courseMapper.insert(course);
        }
    }

    //查询课程的方法
    @Test
    public void findCourse() {
        QueryWrapper<Course>  wrapper = new QueryWrapper<>();
        wrapper.eq("cid",465114665106538497L);
        Course course = courseMapper.selectOne(wrapper);
        System.out.println(course);
    }
}
```

（1）上面测试代码执行，报错了

```
Description:

The bean 'dataSource', defined in class path resource [org/apache/shardingsphere/shardingjdbc/spring/boot/SpringBootConf

Action:

Consider renaming one of the beans or enabling overriding by setting spring.main.allow-bean-definition-overriding=true
```

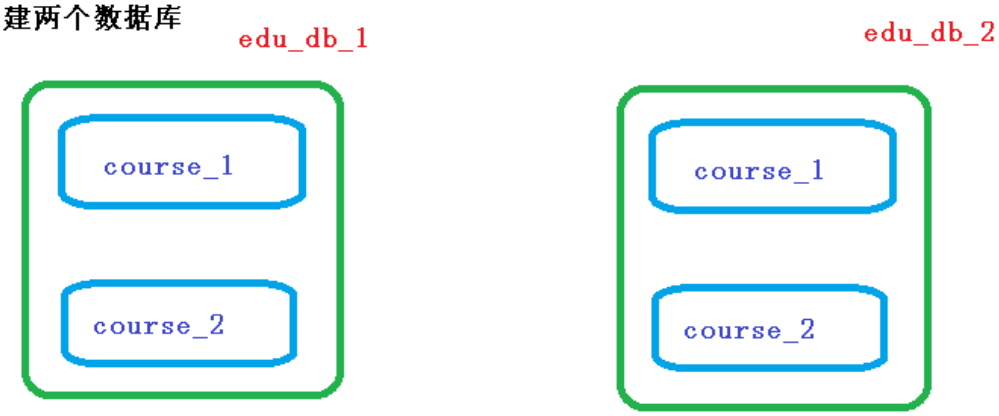
（2）解决方案，在配置文件中添加一行配置

```
# 一个实体类对应两张表，覆盖
spring.main.allow-bean-definition-overriding=true
```

Sharding-JDBC 实现水平分库

1、需求分析

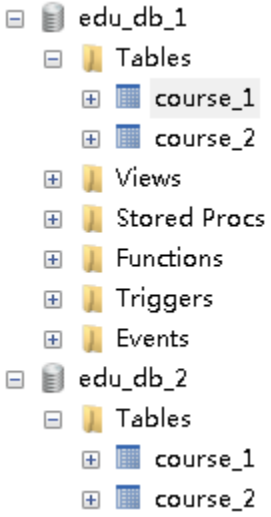
1、创建两个数据库



数据库规则：（1）userid为偶数数据添加edu_db_1数据库
为奇数数据添加edu_db_2数据库

表规则：（1）cid为偶数数据添加course_1表
为奇数数据添加course_2表

2、创建数据库和表



3、在 SpringBoot 配置文件配置数据库分片规则

```
# shardingjdbc 分片策略
```

```
# 配置数据源，给数据源起名称,
# 水平分库，配置两个数据源
spring.shardingsphere.datasource.names=m1,m2

# 一个实体类对应两张表，覆盖
spring.main.allow-bean-definition-overriding=true

#配置第一个数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m1.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m1.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m1.url=jdbc:mysql://localhost:3306/edu_db_1?s
erverTimezone=GMT%2B8
spring.shardingsphere.datasource.m1.username=root
spring.shardingsphere.datasource.m1.password=root

#配置第二个数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m2.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m2.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m2.url=jdbc:mysql://localhost:3306/edu_db_2?s
erverTimezone=GMT%2B8
spring.shardingsphere.datasource.m2.username=root
spring.shardingsphere.datasource.m2.password=root

#指定数据库分布情况，数据库里面表分布情况
# m1  m2     course_1 course_2
spring.shardingsphere.sharding.tables.course.actual-data-nodes=m$-
>{1..2}.course_$->{1..2}

# 指定 course 表里面主键 cid 生成策略  SNOWFLAKE
spring.shardingsphere.sharding.tables.course.key-generator.column=cid
spring.shardingsphere.sharding.tables.course.key-generator.type=SNOWFLAKE

# 指定表分片策略  约定 cid 值偶数添加到 course_1 表，如果 cid 是奇数添加到
course_2 表
spring.shardingsphere.sharding.tables.course.table-strategy.inline.sharding-
column=cid
spring.shardingsphere.sharding.tables.course.table-strategy.inline.algorithm-
expression=course_$->{cid % 2 + 1}

# 指定数据库分片策略 约定 user_id 是偶数添加 m1，是奇数添加 m2
#spring.shardingsphere.sharding.default-database-strategy.inline.sharding-
column=user_id
#spring.shardingsphere.sharding.default-database-strategy.inline.algorithm-
expression=m$->{user_id % 2 + 1}
spring.shardingsphere.sharding.tables.course.database-
strategy.inline..sharding-column=user_id
spring.shardingsphere.sharding.tables.course.database-
strategy.inline.algorithm-expression=m$->{user_id % 2 + 1}
```

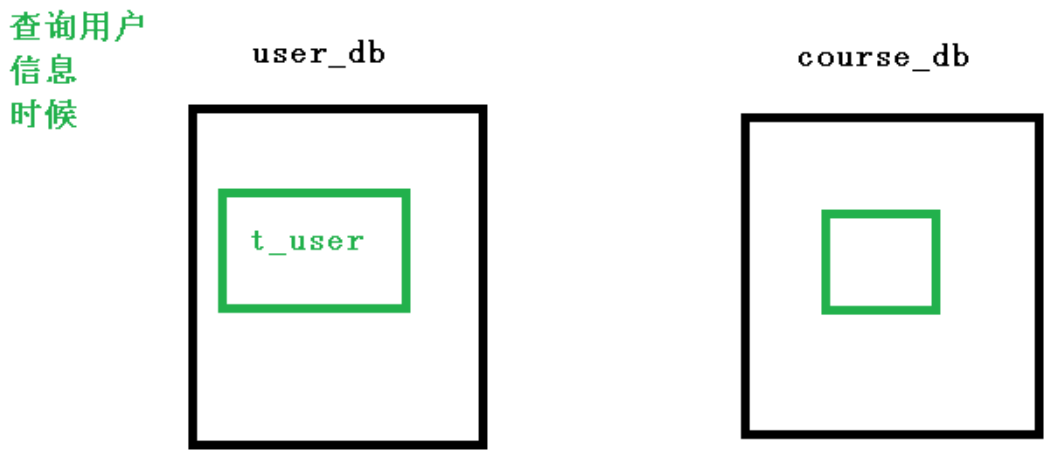
```
# 打开 sql 输出日志
spring.shardingsphere.props.sql.show=true
```

4、编写测试方法

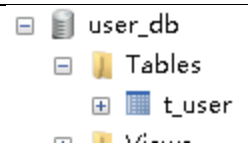
```
//=====================测试水平分库====================
//添加操作
@Test
public void addCourseDb() {
    Course course = new Course();
    course.setCname("javademo1");
    //分库根据 user_id
    course.setUserId(111L);
    course.setCstatus("Normal1");
    courseMapper.insert(course);
}

//查询操作
@Test
public void findCourseDb() {
    QueryWrapper<Course>  wrapper = new QueryWrapper<>();
    //设置 userid 值
    wrapper.eq("user_id",100L);
    //设置 cid 值
    wrapper.eq("cid",465162909769531393L);
    Course course = courseMapper.selectOne(wrapper);
    System.out.println(course);
}
```

Sharding-JDBC 实现垂直分库

1、需求分析



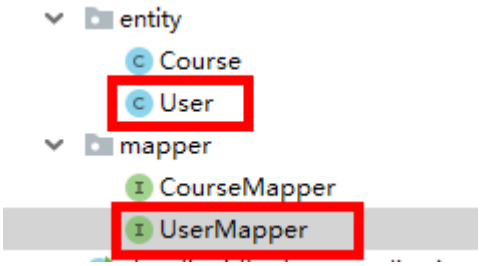
2、创建数据库和表



3、编写操作代码

（1）创建 user 实体类和 mapper

```
@Data
@TableName(value = "t_user")  //指定对应表
public class User {
    private Long userId;
    private String username;
    private String ustatus;
}
```



（2）配置垂直分库策略

* 在 application.properties 进行配置

```
# shardingjdbc 分片策略
# 配置数据源，给数据源起名称,
# 水平分库，配置两个数据源
spring.shardingsphere.datasource.names=m1,m2,m0

# 一个实体类对应两张表，覆盖
spring.main.allow-bean-definition-overriding=true

#配置第一个数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m1.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m1.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m1.url=jdbc:mysql://localhost:3306/edu_db_1?s
erverTimezone=GMT%2B8
spring.shardingsphere.datasource.m1.username=root
spring.shardingsphere.datasource.m1.password=root

#配置第二个数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m2.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m2.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m2.url=jdbc:mysql://localhost:3306/edu_db_2?s
erverTimezone=GMT%2B8
spring.shardingsphere.datasource.m2.username=root
spring.shardingsphere.datasource.m2.password=root
```

```
#配置第三个数据源具体内容，包含连接池，驱动，地址，用户名和密码
spring.shardingsphere.datasource.m0.type=com.alibaba.druid.pool.DruidDataSourc
e
spring.shardingsphere.datasource.m0.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.m0.url=jdbc:mysql://localhost:3306/user_db?se
rverTimezone=GMT%2B8
spring.shardingsphere.datasource.m0.username=root
spring.shardingsphere.datasource.m0.password=root

# 配置 user_db 数据库里面 t_user 专库专表
spring.shardingsphere.sharding.tables.t_user.actual-data-nodes=m$->{0}.t_user

# 指定 course 表里面主键 cid 生成策略  SNOWFLAKE
spring.shardingsphere.sharding.tables.t_user.key-generator.column=user_id
spring.shardingsphere.sharding.tables.t_user.key-generator.type=SNOWFLAKE

# 指定表分片策略  约定 cid 值偶数添加到 course_1 表，如果 cid 是奇数添加到
course_2 表
spring.shardingsphere.sharding.tables.t_user.table-strategy.inline.sharding-
column=user_id
spring.shardingsphere.sharding.tables.t_user.table-strategy.inline.algorithm-
expression=t_user
```

（3）编写测试代码

```
//注入 user 的 mapper
@Autowired
private UserMapper userMapper;

//=====================测试垂直分库=================
//添加操作
@Test
public void addUserDb() {
    User user = new User();
    user.setUsername("lucy");
    user.setUstatus("a");
    userMapper.insert(user);
}
```

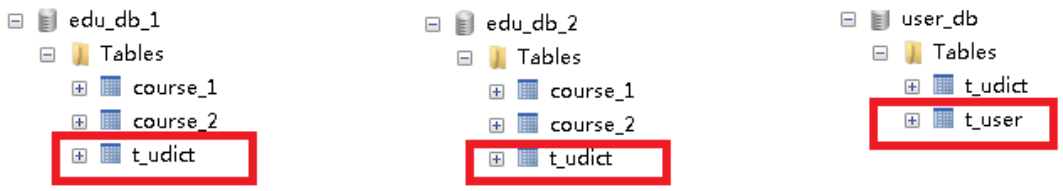
Sharding-JDBC 操作公共表

1、公共表

（1）存储固定数据的表，表数据很少发生变化，查询时候经常进行关联

（2）在每个数据库中创建出相同结构公共表

2、在多个数据库都创建相同结构公共表



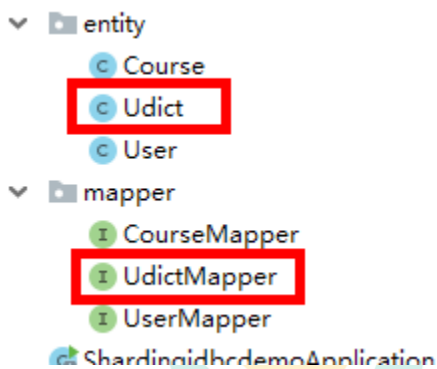
3、在项目配置文件 application.properties 进行公共表配置

```
# 配置公共表
spring.shardingsphere.sharding.broadcast-tables=t_udict
spring.shardingsphere.sharding.tables.t_udict.key-generator.column=dictid
spring.shardingsphere.sharding.tables.t_udict.key-generator.type=SNOWFLAKE
```

4、编写测试代码

（1）创建新实体类和 mapper

```
@Data
@TableName(value = "t_udict")
public class Udict {
    private Long dictid;
    private String ustatus;
    private String uvalue;
}
```



（2）编写添加和删除方法进行测试

```
@Autowired
private UdictMapper udictMapper;

//=====================测试公共表==================
//添加操作
@Test
public void addDict() {
    Udict udict = new Udict();
    udict.setUstatus("a");
    udict.setUvalue("已启用");
    udictMapper.insert(udict);
}

//删除操作
@Test
public void deleteDict() {
```

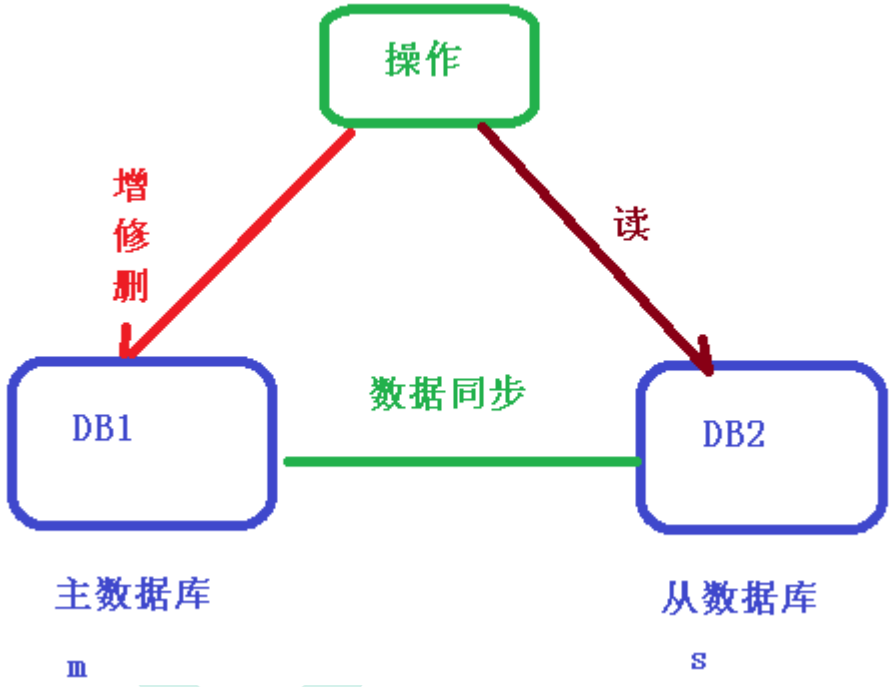
```
    QueryWrapper<Udict>  wrapper = new QueryWrapper<>();
    //设置 userid 值
    wrapper.eq("dictid",465191484111454209L);
    udictMapper.delete(wrapper);
}
```

Sharding-JDBC 实现读写分离

1、读写分离概念

为了确保数据库产品的稳定性，很多数据库拥有双机热备功能。也就是，第一台数据库服务器，是对外提供增删改业务的生产服务器；第二台数据库服务器，主要进行读的操作。

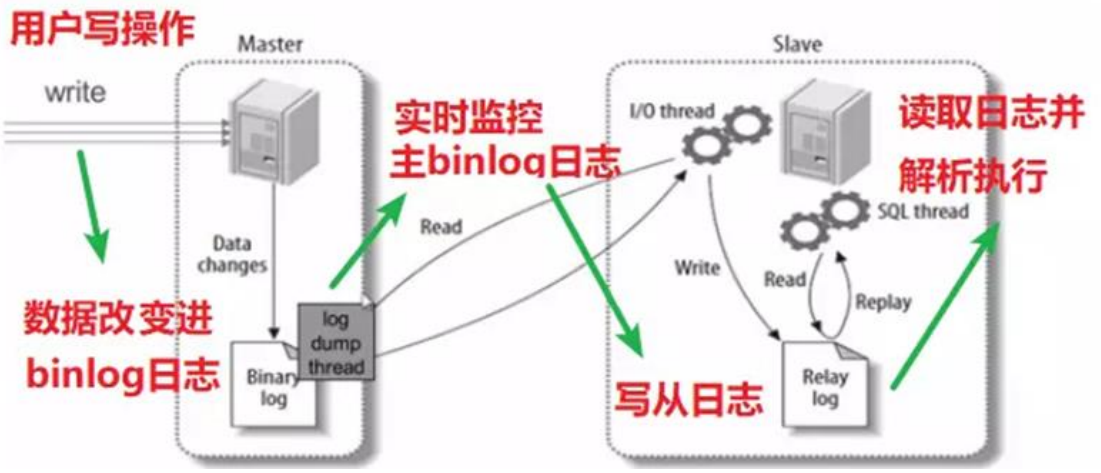
原理：让主数据库（master）处理事务性增、改、删操作，而从数据库（slave）处理SELECT查询操作。



读写分离原理：

主从复制：当主服务器有写入（insert/update/delete）语句时候，从服务器自动获取

读写分离：insert/update/delete语句操作一台服务器，select操作另一个服务器

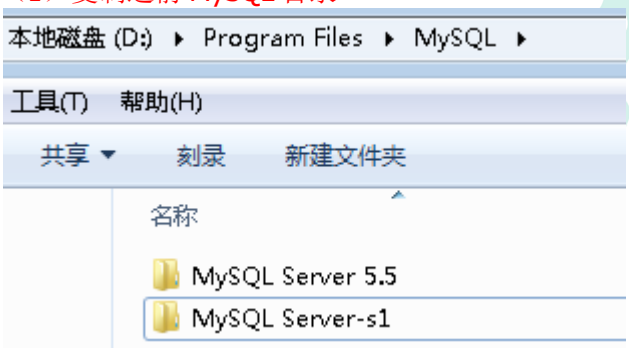


Sharding-JDBC 通过 sql 语句语义分析，实现读写分离过程，不会做数据同步

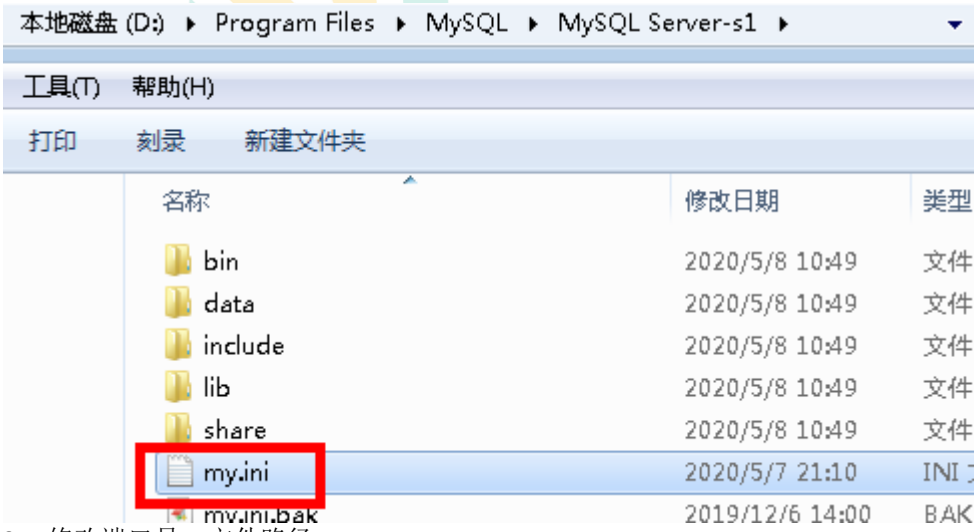
2、MySQL 配置读写分离

第一步 创建两个 MySQL 数据库服务，并且启动两个 MySQL 服务

（1）复制之前 MySQL 目录



（2）修改复制之后配置文件



- 修改端口号，文件路径
- 需要把数据文件目录再复制一份


```
[client]

port=3307

# The TCP/IP Port the MySQL Server will listen on
port=3307


#Path to installation directory. All paths are usually resol
basedir="D:/Program Files/MySQL/MySQL Server-s1/"

#Path to the database root
datadir="C:/ProgramData/MySQL/MySQL Server-s1/Data/"
```

（3）把复制修改之后从数据库在 windows 安装服务

* 使用命令：mysqld install mysqls1 --defaults-file="D:\Program Files\MySQL\MySQL Server-s1\my.ini"

```
D:\Program Files\MySQL\MySQL Server-s1\bin>mysqld install mysqls1 --defaults-fil
e="D:\Program Files\MySQL\MySQL Server-s1\my.ini"
Service successfully installed.
```

MySQL	已启动	自动	本地系统
mysqls1	已启动	自动	本地系统

第二步 配置 MySQL 主从服务器

（1）在主服务器配置文件

```
[mysqld]
#开启日志
log-bin = mysql-bin
#设置服务id，主从不能一致
server-id = 1
#设置需要同步的数据库
binlog-do-db=user_db
#屏蔽系统库同步
binlog-ignore-db=mysql
binlog-ignore-db=information_schema
binlog-ignore-db=performance_schema
```

（2）在从服务器配置文件

```
[mysqld]
#开启日志
log-bin = mysql-bin
#设置服务id，主从不能一致
```

```
server-id = 2
#设置需要同步的数据库
replicate_wild_do_table=user_db.%
#屏蔽系统库同步
replicate_wild_ignore_table=mysql.%
replicate_wild_ignore_table=information_schema.%
replicate_wild_ignore_table=performance_schema.%
```

（3）把主和从服务器重启

第三步 创建用于主从复制的账号

```
#切换至主库bin目录，登录主库
mysql -h localhost -uroot -p
#授权主备复制专用账号
GRANT REPLICATION SLAVE ON *.* TO 'db_sync'@'%' IDENTIFIED BY 'db_sync';
#刷新权限
FLUSH PRIVILEGES;
```

%	db_sync	*9D8D15B4F5FDEA3A135CE1C93A0FF2F9EB2EDA86

```
#确认位点 记录下文件名以及位点
show master status;
```

File	Position	Binlog_Do_DB	Binlog_Ignore_DB
mysql-bin.000177	107	user_db	mysql,information_schema,performance_schema

第四步 主从数据同步设置

```
#切换至从库bin目录，登录从库
mysql -h localhost -P3307 -uroot -p

#先停止同步
STOP SLAVE;

#修改从库指向到主库，使用上一步记录的文件名以及位点
CHANGE MASTER TO
master_host = 'localhost',
master_user = 'db_sync',
master_password = 'db_sync',
master_log_file = 'mysql-bin.000177',
master_log_pos = 107;

#启动同步
START SLAVE;

#查看Slave_IO_Runing和Slave_SQL_Runing字段值都为Yes，表示同步配置成功。如果不为Yes，请排查相关异常。
show slave status
```

Relay_Master_Log_File	Slave_IO_Running	Slave_SQL_Running	R
mysql-bin.000177	Yes	Yes	

3、Sharding-JDBC 操作

（1）配置读写分离策略

```
# user_db 从服务器
spring.shardingsphere.datasource.s0.type=com.alibaba.druid.pool.DruidDataSource
spring.shardingsphere.datasource.s0.driver-class-name=com.mysql.cj.jdbc.Driver
spring.shardingsphere.datasource.s0.url=jdbc:mysql://localhost:3307/user_db?serverTimezone=GMT%2B8
spring.shardingsphere.datasource.s0.username=root
```

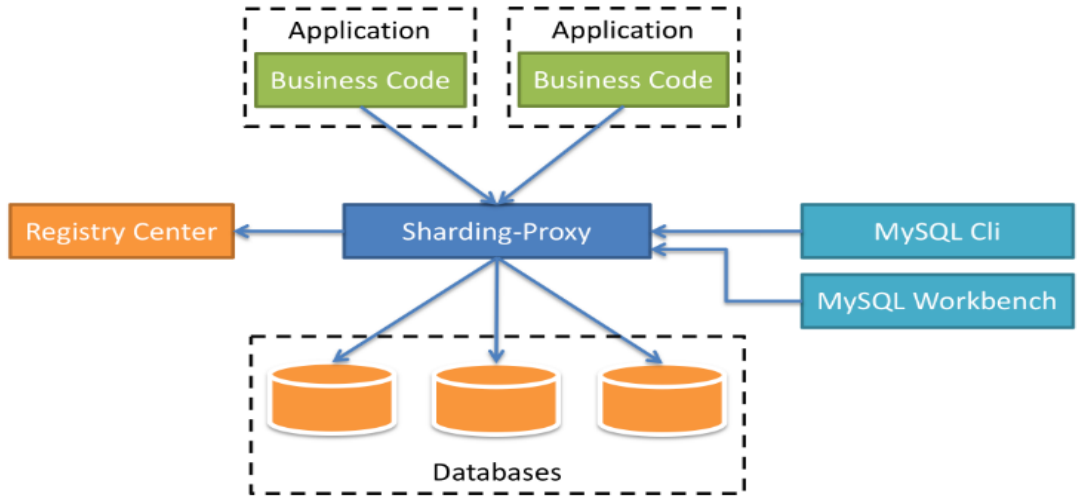
```
spring.shardingsphere.datasource.s0.password=root

# 主库从库逻辑数据源定义 ds0 为 user_db
spring.shardingsphere.sharding.master-slave-rules.ds0.master-data-source-name=m0
spring.shardingsphere.sharding.master-slave-rules.ds0.slave-data-source-names=s0

# 配置 user_db 数据库里面 t_user 专库专表
#spring.shardingsphere.sharding.tables.t_user.actual-data-nodes=m$->{0}.t_user
# t_user 分表策略，固定分配至 ds0 的 t_user 真实表
spring.shardingsphere.sharding.tables.t_user.actual-data-nodes=ds0.t_user
```

（2）编写测试代码

```
//添加操作
@Test
public void addUserDb() {
    User user = new User();
    user.setUsername("lucymary");
    user.setUstatus("a");
    userMapper.insert(user);
}

//查询操作
@Test
public void findUserDb() {
    QueryWrapper<User>  wrapper = new QueryWrapper<>();
    //设置 userid 值
    wrapper.eq("user_id",465508031619137537L);
    User user = userMapper.selectOne(wrapper);
    System.out.println(user);
}
```

Sharding-Proxy 简介

1、定位为透明的数据库代理端



2、Sharding-Proxy 独立应用，需要安装服务，进行分库分表或者读写分离配置，启动使用

3、安装

（1）下载安装软件

ShardingSphere > 下载

ShardingSphere的发布版包括源码包及其对应的二进制包。由于下载内容分布在镜像服务上，所以下载后应该进行GPG或SHA-512校验，以此来保证内容没有被篡改。

版本	发布日期	源码下载	Sharding-JDBC下载	Sharding-Proxy下载	Sharding-UI下载	Sharding-Scaling 下载
4.0.1	2020 Mar 9	source (asc sha512)	binary (asc sha512)	binary (asc sha512)	binary (asc sha512)	
4.1.0	2020 Apr 30	source (asc sha512)	binary (asc sha512)	binary (asc sha512)		binary (asc sha512)

HTTP

https://downloads.apache.org/incubator/shardingsphere/4.0.1/apache-shardingsphere-incubating-4.0.1-sharding-proxy-bin.tar.gz

（2）把下载之后压缩文件，解压，启动 bin 目录启动文件就可以了

apache-shardingsphere-incubating-4.0.1-sharding-proxy-bin

查看(V) 工具(T) 帮助(H)

到库中 共享 刻录 新建文件夹

名称	修改日期
bin	2020/5/8 15:36
conf	2020/5/8 16:47
lib	2020/5/8 15:47
licenses	2020/5/8 14:25
DISCLAIMER	2020/2/25 17:54
LICENSE	2020/2/25 17:54
NOTICE	2020/2/25 17:54
README.txt	2020/2/25 17:54

Sharding-Proxy 配置（分表）

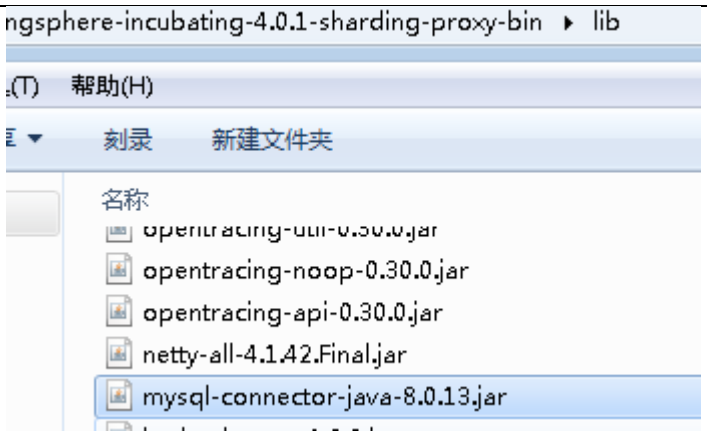
1、进入 conf 目录，修改文件 server.yaml，打开两段内容注释


```
authentication:
  users:
    root:
      password: root
    sharding:
      password: sharding
      authorizedSchemas: sharding_db

props:
  max.connections.size.per.query: 1
  acceptor.size: 16  # The default value is available process
  executor.size: 16  # Infinite by default.
  proxy.frontend.flush.threshold: 128  # The default value is
    # LOCAL: Proxy will run with LOCAL transaction.
    # XA: Proxy will run with XA transaction.
    # BASE: Proxy will run with B.A.S.E transaction.
  proxy.transaction.type: LOCAL
  proxy.opentracing.enabled: false
  query.with.cipher.column: true
  sql.show: false
```

2、进入 conf 目录，修改 config-sharding.yaml

（1）复制 mysql 驱动 jar 包到 lib 目录

```
If you want to connect to MySQL, you should manually copy MySQL driver to lib directory.
```

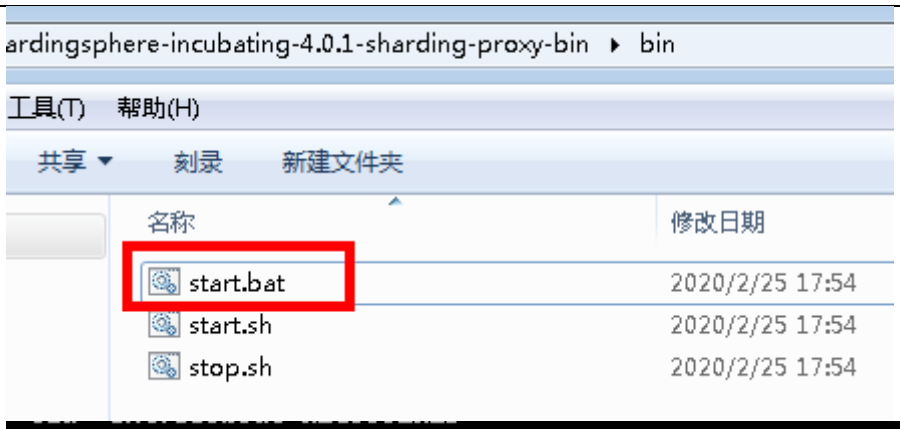


（2）配置分库分表规则

```
schemaName: sharding_db

dataSources:
  ds_0:
    url: jdbc:mysql://127.0.0.1:3306/edu_1?serverTimezone=UTC&useSSL=false
    username: root
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50

shardingRule:
  tables:
    t_order:
      actualDataNodes: ds_${0}.t_order_${0..1}
      tableStrategy:
        inline:
          shardingColumn: order_id
          algorithmExpression: t_order_${order_id % 2}
      keyGenerator:
        type: SNOWFLAKE
        column: order_id
  bindingTables:
    - t_order
  defaultDatabaseStrategy:
    inline:
      shardingColumn: user_id
      algorithmExpression: ds_${0}
  defaultTableStrategy:
    none:
```

3、启动 Sharding-Proxy 服务

（1）Sharding-Proxy 默认端口号 3307




```
[INFO ] 17:31:34.650 [nioEventLoopGroup-2-1] i.n.handler.logging.Lo
- [id: 0x47008bed] BIND: 0.0.0.0/0.0.0.0:3307
[INFO ] 17:31:34.650 [nioEventLoopGroup-2-1] i.n.handler.logging.Lo
- [id: 0x47008bed, L:/0:0:0:0:0:0:0:0:3307] ACTIVE
```

4、通过 Sharding-Proxy 启动端口进行连接

（1）打开 cmd 窗口连接 Sharding-Proxy，连接方式和连接 mysql 一样的

```
C:\Users\lenovo>mysql -P3307 -uroot -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.4-Sharding-Proxy 4.0.0

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All right

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current inp

mysql>
```

（2）进行 sql 命令操作看到只有一个库

```
mysql> show databases;
+--------------+
| Database     |
+--------------+
| sharding_db  |
+--------------+
1 row in set (0.01 sec)
```

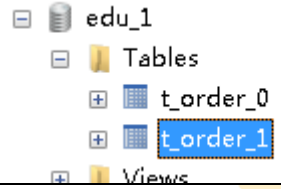
（3）在 sharding_db 数据库创建表

```
mysql> CREATE TABLE IF NOT EXISTS ds_0.t_order (order_id BIGINT NOT NULL, user_i
d INT NOT NULL, status VARCHAR(50), PRIMARY KEY (order_id));
Query OK, 0 rows affected (0.27 sec)

mysql> use sharding_db;
Database changed
mysql> show tables;
+------------------+
| Tables_in_edu_1 |
+------------------+
| t_order          |
+------------------+
1 row in set (0.01 sec)
```

（4）向表添加一条记录

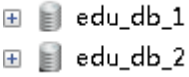
```
mysql> INSERT INTO t_order (order_id,user_id,status) VALUES (11,1,'init');
Query OK, 1 row affected (0.30 sec)

mysql> select * from t_order;
+----------+---------+--------+
| order_id | user_id | status |
+----------+---------+--------+
|       11 |       1 | init   |
+----------+---------+--------+
1 row in set (0.01 sec)
```

5、回到本地 3306 端口实际数据库中，看到已经创建好了表和添加数据



Sharding-Proxy 配置（分库）

1、创建两个数据库



2、找到 conf 目录，config-sharding.yaml

```
schemaName: sharding_db

dataSources:
  ds_0:
    url: jdbc:mysql://127.0.0.1:3306/edu_db_1?serverTimezone=UTC&useSSL=false
    username: root
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50
  ds_1:
    url: jdbc:mysql://127.0.0.1:3306/edu_db_2?serverTimezone=UTC&useSSL=false
    username: root
```

```
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50

shardingRule:
  tables:
    t_order:
      actualDataNodes: ds_${0..1}.t_order_${1..2}
      tableStrategy:
        inline:
          shardingColumn: order_id
          algorithmExpression: t_order_${order_id % 2 + 1}
      keyGenerator:
        type: SNOWFLAKE
        column: order_id
  bindingTables:
    - t_order
  defaultDatabaseStrategy:
    inline:
      shardingColumn: user_id
      algorithmExpression: ds_${user_id % 2}
  defaultTableStrategy:
    none:
```

3、启动 Sharding-Proxy 服务

```
[INFO ] 18:10:10.307 [nioEventLoopGroup-2-1] i.n.handler.logging.Log
- [id: 0xa1b09624] BIND: 0.0.0.0/0.0.0.0:3307
[INFO ] 18:10:10.307 [nioEventLoopGroup-2-1] i.n.handler.logging.Log
- [id: 0xa1b09624, L:/0:0:0:0:0:0:0:0:3307] ACTIVE
```

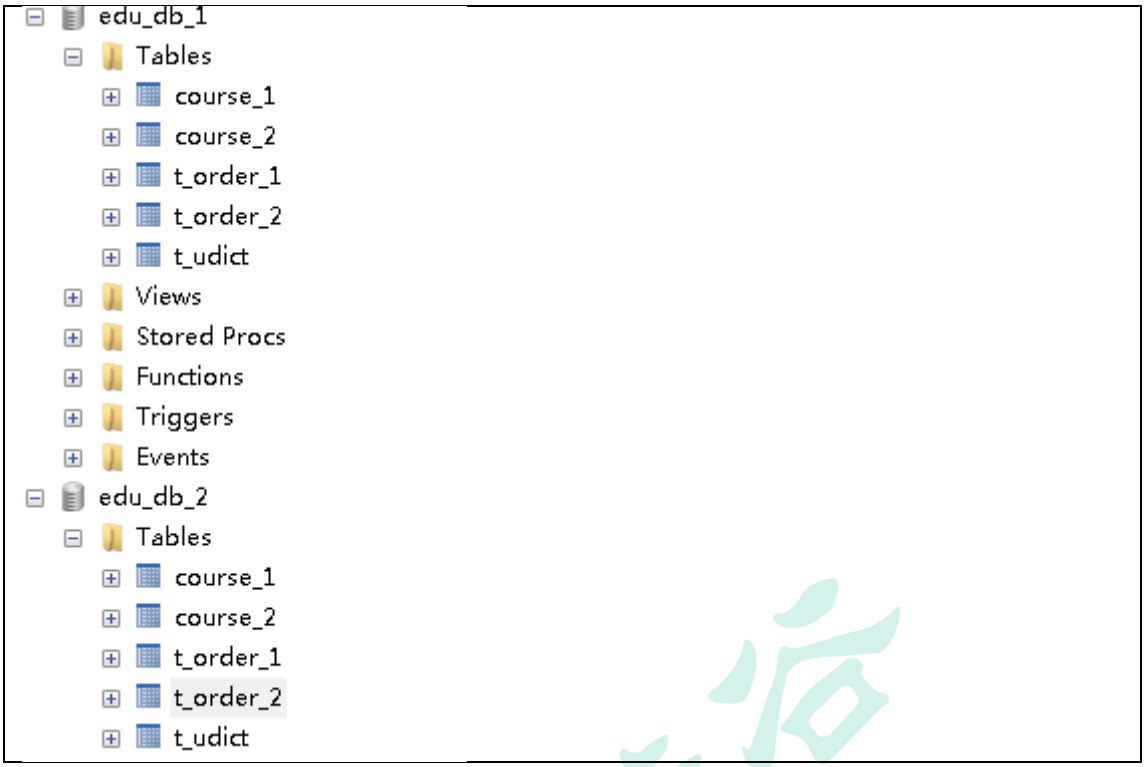
4、打开 cmd 仓库，连接 Sharding-Proxy 服务

```
C:\Users\lenovo>mysql -P3307 -uroot -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
```

（1）创建数据库表，向表添加记录

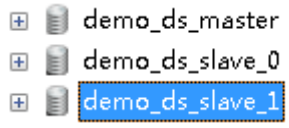
```
mysql> CREATE TABLE IF NOT EXISTS ds_0.t_order (order_id BIGINT NOT NULL, user_i
d INT NOT NULL, status VARCHAR(50), PRIMARY KEY (order_id));
Query OK, 0 rows affected (0.48 sec)

mysql> INSERT INTO t_order (order_id,user_id,status) VALUES (11,1,'init');
Query OK, 1 row affected (0.19 sec)
```

（2）连接本地 3306 的 MySql 数据库服务器，表已经创建出来，表里面有数据

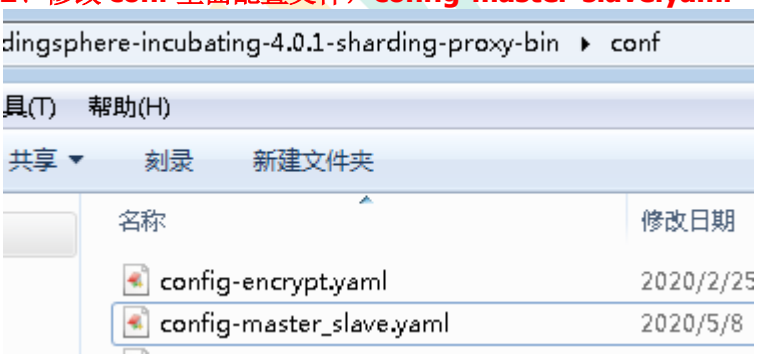


Sharding-Proxy 配置（读写分离）

1、创建三个数据



2、修改 conf 里面配置文件，config-master-slave.yaml




```
schemaName: master_slave_db

dataSources:
  master_ds:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_master?serverTimezone=UTC&useSSL=false
    username: root
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50
  slave_ds_0:
```

```
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_slave_0?serverTimezone=UTC&useSSL=false
    username: root
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50
  slave_ds_1:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_slave_1?serverTimezone=UTC&useSSL=false
    username: root
    password: root
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 50

masterSlaveRule:
  name: ms_ds
  masterDataSourceName: master_ds
  slaveDataSourceNames:
    - slave_ds_0
    - slave_ds_1
```

3、启动 Sharding-Proxy 服务

```
[INFO ] 18:28:20.996 [nioEventLoopGroup-2-1] i.n.handler.logging.Loggi
- [id: 0xdbfaa07b] BIND: 0.0.0.0/0.0.0.0:3307
[INFO ] 18:28:20.996 [nioEventLoopGroup-2-1] i.n.handler.logging.Loggi
- [id: 0xdbfaa07b, L:/0:0:0:0:0:0:0:0:3307] ACTIVE
```

4、通过 cmd 连接 Sharding-Proxy，进行创建表和添加记录操作

```
mysql> show databases;
+-----------------+
| Database        |
+-----------------+
| master_slave_db |
| sharding_db     |
+-----------------+
```

（1）在主数据库和从数据库里面，都创建数据库表

```
mysql> use master_slave_db;
Database changed
mysql> CREATE TABLE IF NOT EXISTS demo_ds_master.t_order (order_id BIGINT NOT NU
LL, user_id INT NOT NULL, status VARCHAR(50), PRIMARY KEY (order_id));
Query OK, 0 rows affected (0.32 sec)

mysql> CREATE TABLE IF NOT EXISTS demo_ds_slave_0.t_order (order_id BIGINT NOT N
ULL, user_id INT NOT NULL, status VARCHAR(50), PRIMARY KEY (order_id));
Query OK, 0 rows affected (0.19 sec)

mysql> CREATE TABLE IF NOT EXISTS demo_ds_slave_1.t_order (order_id BIGINT NOT N
ULL, user_id INT NOT NULL, status VARCHAR(50), PRIMARY KEY (order_id));
Query OK, 0 rows affected (0.10 sec)
```

（2）向表添加记录，不指定向哪个库添加

* 把添加数据添加到主数据库里面

```
mysql> INSERT INTO t_order (order_id,user_id,status) VALUES (11,1,'init');
Query OK, 1 row affected (0.22 sec)
```

（3）查询数据库表数据，不指定查询哪个库
* 直接执行查询从库里面的数据

```
mysql> select * from t_order;
Empty set (0.06 sec)
```

课程总结

一、基本概念
1、什么是 Sharding Sphere
2、什么是分库分表
（1）水平切分和垂直切分

二、Sharding-JDBC
1、什么是 Sharding-JDBC
2、使用 Sharding-JDBC 水平切分
3、使用 Sharding-JDBC 垂直切分
4、使用 Sharding-JDBC 操作公共表
5、使用使用 Sharding-JDBC 读写分离

三、Sharding-Proxy
1、什么是 Sharding-Proxy
2、使用 Sharding-Proxy 分库分表
3、使用 Sharding-Proxy 读写分离