// +--
// | mycat 基于的是 MySQL5.7 系列，mycat1.6 系列，JDK1.7 系列
// | zookeeper,haproxy,keepalived,mysql,mycat-web
// +--
// | Version: 1.0
// +--
// | Author: JettJia <JettJia@qq.com>
// +--
// | Date: 2017.10.1
// +--

灰色底**黑色加粗**：	输入的系统命令，重要的提醒
灰色底：	输入的系统命令返回的信息或者配置文件文本信息
黄色底：	技巧或需要注意的注释信息
橙色底：	需要特别注意的地方
蓝色字体：	内容注释

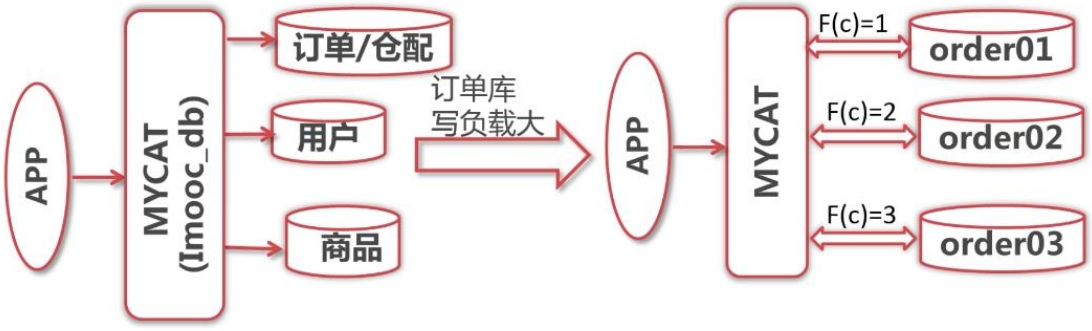
1 介绍

1.1 官网和 GitHub

http://www.mycat.io/
https://github.com/MyCATApache/Mycat-Server

1.2 mycat 作用

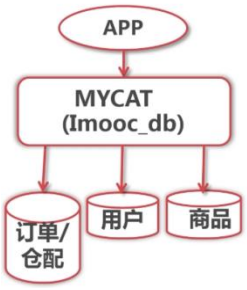
读写分离、垂直切分、水平切分、控制数据库连接数



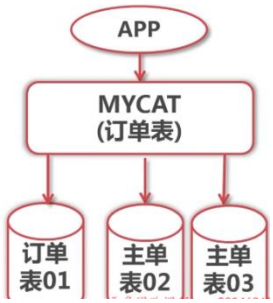
1.3 mycat 基本元素

逻辑库：
对应用来说相当于 MySQL 中的数据库

逻辑库可对应后端多个物理数据库
逻辑库中并不保存数据



逻辑表：
对于应用来说相当于 MySQL 中的数据表
逻辑表可对应后端多个物理数据库中的表
逻辑表中并不保存数据



逻辑表的类别：
分片表与非分片表按是否被分片划分
全局表，在所有分片中都存在的表
ER 关系表，按 ER 关系进行分片的表

2 MySQL 准备

演示环境的说明

主机名	IP	角色
node1	10.0.0.62	mycat mysql
node2	10.0.0.63	mysql
node3	10.0.0.64	mysql
node4	10.0.0.65	mysql

2.1 数据库安装

四个节点上都安装上 mysql 数据库

node1 节点同时安装上 mycat

4 个节点都安装上 mysql

安装 mysql，这里用的 cmake 编译安装的方式，安装的步骤和脚本如下：

```
[root@node1 ~]# mkdir -p /server/scripts/
[root@node1 scripts]# mkdir -p /home/tools/
[root@node1 scripts]# cd /home/tools/
[root@node1 tools]# rz
mysql-5.7.9.tar.gz
boost_1_59_0.tar.gz
```

这里的 mysql-5.7.9.tar.gz 和 boost_1_59_0.tar.gz 都是提前下载好的

执行安装脚本

```
[root@node1 tools]# sh /server/scripts/installMysql.sh
```

脚本内容如下：

```
#!/bin/bash
#################################################################################
########
#//
+----------------------------------------------------------------------------------------------------------------
-
#// | Xpress Install mysql for cmake style
#//
+----------------------------------------------------------------------------------------------------------------
-
#// | Version:      1.1
#//
+----------------------------------------------------------------------------------------------------------------
-
#// | Author:       JettJia <JettJia@qq.com>
#//
+----------------------------------------------------------------------------------------------------------------
-
#// | Date:         2017.10.1，2018.03.01
#//
+----------------------------------------------------------------------------------------------------------------
-
#// | default:
#// | mysql version 5.7.9
#// | mysql installation path : /application/mysql
#// | Management scripts : /server/scripts
#// | mysql pid: /application/mysql/run/mysql-fpm.pid
#// | mysql conf /etc/my.cnf
#// | Software management /etc/init.d/mysqld
```

```
#//
+----------------------------------------------------------------------------------------------------------
-

# Dead work
fileName="mysql-5.7.9"
package="mysql-5.7.9.tar.gz"
installDir="/application/mysql-5.7.9"
installLnDir="/application/mysql"

tools_dir=/home/tools
[ ! -d $tools_dir ] && mkdir /home/tools -p

scripts_dir=/server/scripts
[ ! -d $scripts_dir ] && mkdir /server/scripts -p

data_dir=/data/mysql
[ ! -d $data_dir ] && mkdir /data/mysql -p

id mysql >& /dev/null
if [ $? -ne 0 ];then
        groupadd mysql
        useradd -s /sbin/nologin -g mysql -M mysql
fi

# Install dependency environment
function dependency() {
        wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-6.repo
        yum -y install gcc gcc-c++ ncurses ncurses-devel cmake
        # boost
        mkdir -p /usr/local/boost
        cp $tools_dir/boost_1_59_0.tar.gz /usr/local/boost
}

# Install mysql
function installMysql() {
        local configure_str=$(cat <<EOF
        cmake . -DCMAKE_INSTALL_PREFIX=${installDir} \
        -DMYSQL_DATADIR=${data_dir} \
        -DDOWNLOAD_BOOST=1 \
        -DWITH_BOOST=/usr/local/boost/ \
        -DSYSCONFDIR=/etc \
        -DWITH_INNOBASE_STORAGE_ENGINE=1 \
        -DWITH_PARTITION_STORAGE_ENGINE=1 \
```

```
    -DWITH_FEDERATED_STORAGE_ENGINE=1 \
    -DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
    -DWITH_MYISAM_STORAGE_ENGINE=1 \
    -DENABLED_LOCAL_INFILE=1 \
    -DWITH_FAST_MUTEXES=1 \
    -DWITH_ZLIB=bundled \
    -DENABLE_DTRACE=0 \
    -DDEFAULT_CHARSET=utf8mb4 \
    -DDEFAULT_COLLATION=utf8mb4_general_ci \
    -DEXTRA_CHARSETS=gbk,gb2312,utf8,ascii \
    -DWITH_EMBEDDED_SERVER=1
EOF
)
    cd $tools_dir
    if [ ! -f "$package" ]; then
        echo "no $package"
        exit 2
    fi
    tar zxf $package
    cd $fileName
    $configure_str
    make -j `grep processor /proc/cpuinfo | wc -l`
    make install
    cd ../

    # create a link to php
    ln -s $installDir $installLnDir
    chown -R mysql.mysql ${installLnDir}
    chmod -R 1777 /tmp
    # my.cnf
    myCnf
    # Initialization mysql data
    ${installDir}/bin/mysqld --initialize-insecure --basedir=${installDir}
--datadir=${data_dir} --user=mysql
    # cp service
    /bin/cp /${installDir}/support-files/mysql.server /etc/init.d/mysqld
    chmod +x /etc/init.d/mysqld
    sed -i 's#/usr/local/mysql#${installLnDir}#g' ${installLnDir}/bin/mysqld_safe
/etc/init.d/mysqld
    # Initialize the global configuration
    if [ `grep mysql_bin /etc/profile|wc -l` -lt 1 ]; then
        echo '#mysql_bin' >> /etc/profile
        echo 'export PATH=/application/mysql/bin:$PATH' >> /etc/profile
    fi
```

```
    source /etc/profile
}

function myCnf(){
    cat > /etc/my.cnf <<EOF
[client]
port = 3306
socket = /tmp/mysql.sock
default-character-set = utf8mb4

[mysqld]
port = 3306
socket = /tmp/mysql.sock

basedir = /application/mysql
datadir = /data/mysql
pid-file = /data/mysql/mysql.pid
user = mysql
bind-address = 0.0.0.0
server-id = 1

init-connect = 'SET NAMES utf8mb4'
character-set-server = utf8mb4

#skip-name-resolve
#skip-networking
back_log = 300

max_connections = 1000
max_connect_errors = 6000
open_files_limit = 65535
table_open_cache = 128
max_allowed_packet = 4M
binlog_cache_size = 1M
max_heap_table_size = 8M
tmp_table_size = 16M

read_buffer_size = 2M
read_rnd_buffer_size = 8M
sort_buffer_size = 8M
join_buffer_size = 8M
key_buffer_size = 4M

thread_cache_size = 8
```

```
query_cache_type = 1
query_cache_size = 8M
query_cache_limit = 2M

ft_min_word_len = 4

log_bin = mysql-bin
binlog_format = mixed
expire_logs_days = 30

log_error = /data/mysql/mysql-error.log
slow_query_log = 1
long_query_time = 1
slow_query_log_file = /data/mysql/mysql-slow.log

performance_schema = 0
explicit_defaults_for_timestamp

#lower_case_table_names = 1

skip-external-locking

default_storage_engine = InnoDB
#default-storage-engine = MyISAM
innodb_file_per_table = 1
innodb_open_files = 500
innodb_buffer_pool_size = 64M
innodb_write_io_threads = 4
innodb_read_io_threads = 4
innodb_thread_concurrency = 0
innodb_purge_threads = 1
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 2M
innodb_log_file_size = 32M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120

bulk_insert_buffer_size = 8M
myisam_sort_buffer_size = 8M
myisam_max_sort_file_size = 10G
myisam_repair_threads = 1
```

```
interactive_timeout = 28800
wait_timeout = 28800

[mysqldump]
quick
max_allowed_packet = 16M

[myisamchk]
key_buffer_size = 8M
sort_buffer_size = 8M
read_buffer = 4M
write_buffer = 4M
EOF
}

function main () {
    dependency
    installMysql
    # start mysql
    /etc/init.d/mysqld start
    if [ $? -eq 0 ]; then
        echo "mysql install is ok"
        exit 0
    else
        echo "mysql install is error"
        exit 1
    fi
}
main
```

2.2 测试数据准备

说明：
基础数据库是 imooc_db.sql，目标，针对它进行垂直拆分，水平拆分等实际配置测试
下面是对测试的表的说明：
订单/仓配表：

order_customer_addr	用户地址表
order_master	订单主表
order_detail	订单详情表
order_cart	购物车表
region_info	地区信息表
warehouse_info	仓库信息表
warehouse_proudct	商品库存表
shipping_info	物流公司信息表

商品表：

product_brand_info	品牌信息表
product_category	商品分类表
product_supplier_info	供应商信息表
product_info	商品信息表
product_pic_info	商品图片信息表
product_comment	商品评论表

用户表：

customer_login	用户登陆表
customer_inf	用户信息表
customer_level_inf	用户级别信息表
customer_login_log	用户登录日志表
customer_point_log	用户积分日志表
customer_balance_log	用户余额变动表

3 mycat 环境准备

3.1 环境准备

MyCat 是使用 JAVA 语言进行编写开发，使用前需要先安装 JAVA 运行环境(JRE)。JDK 需要 7 以上

1）JDK 下载

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

注：JDK7 以上，或更高版本

2）MySQL 下载

https://dev.mysql.com/downloads/mysql/5.5.html#downloads

注：MyCat 支持多种数据库接入，如：MySQL、SQLServer、Oracle、MongoDB 等，推荐使用 MySql 做集群。

注：5.5 以上

3）MyCAT 项目主页

https://github.com/MyCATApache

注：MyCAT 相关源码、文档

3.2 环境安装和配置

1 JDK 安装配置

```
[root@node1 ~]# mkdir -p /home/tools
[root@node1 ~]# cd /home/tools/
[root@node1 tools]# tar xf jdk-7u80-linux-x64.tar.gz
[root@node1 tools]# mkdir -p /usr/local/java
[root@node1 tools]# mv jdk1.7.0_80/ /usr/local/java/
[root@node1 tools]# chown -R root.root /usr/local/java
[root@node1 tools]# cp /etc/profile /etc/profile.ori
[root@node1 tools]# vim /etc/profile
# java
export JAVA_HOME=/usr/local/java/jdk1.7.0_80
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
[root@node1 tools]# source /etc/profile

#检查是否安装成功
[root@node1 tools]# java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
```

2 mycat 安装配置

1 下载解压

```
[root@node1 tools]# wget
http://dl.mycat.io/1.6-RELEASE/Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz
[root@node1 tools]# tar -zxf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz
[root@node1 tools]# mv mycat/ /usr/local/
[root@node1 tools]# tree -L 1 /usr/local/mycat/
/usr/local/mycat/
├── bin
├── catlet
├── conf
├── lib
├── logs
└── version.txt
[root@node1 tools]# useradd mycat
[root@node1 tools]# chown -R mycat.mycat /usr/local/mycat/
#[root@node1 tools]# passwd mycat
#123456
```

2 配置 mycat 的环境变量

```
vim /etc/profile
#mycat 2018-05-01
export MYCAT_HOME=/usr/local/mycat
export PATH=$PATH:$MYCAT_HOME/bin
[root@node1 tools]# source /etc/profile
```

3 配置 IP 和主机名的映射

```
vi /etc/hosts
127.0.0.1      localhost localhost.localdomain node1
```

如果是在多台 Linux 系统中组建的 MyCAT 集群，那需要在 MyCAT Server 所在的服务器上配置对其他 ip 和主机名的映射，配置方式如下：

vi /etc/hosts

例如：我有 4 台机器，配置如下：

IP 主机名：

192.168.100.2 sam_node1

192.168.100.3 sam_server_2

192.168.100.4 sam_server_3

192.168.100.5 sam_server_4

编辑完后，保存文件。

4 修改 conf/wrapper.conf

```
[root@mycat tools]# vim /usr/local/mycat/conf/wrapper.conf
```

如下，默认文件是 2 个 G，修改为服务器的一半左右，根据实际情况调整

```
wrapper.java.additional.5=-XX:MaxDirectMemorySize=2G
```

这里虚拟机环境修改为 256M

```
wrapper.java.additional.5=-XX:MaxDirectMemorySize=256M
```

5 启动 mycat 服务

```
[root@mycat mycat]# mycat start
Starting Mycat-server...
```

#检查

```
[root@node1 mycat]# netstat -lntup|grep 8066
tcp        0      0 :::8066                     :::*
LISTEN      1288/java
```

3.3 mycat 目录说明

```
/usr/local/mycat/
├─── bin
├─── catlet
├─── conf
├─── lib
├─── logs
```

```
└── version.txt
```

bin 程序目录，存放了 window 版本和 linux 版本，除了提供封装成服务的版本之外，也提供了 nowrap 的 shell 脚本命令，方便大家选择和修改，进入到 bin 目录：
Linux 下运行：./mycat console,首先要 chmod +x *
注：mycat 支持的命令{ console | start | stop | restart | status | dump }

conf 目录下存放配置文件，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 或者通过 9066 端口 reload

lib 目录下主要存放 mycat 依赖的一些 jar 文件.

日志存放在 logs/mycat.log 中，每天一个文件，日志的配置是在 conf/log4j.xml 中，根据自己的需要，可以调整输出级别为 debug，debug 级别下，会输出更多的信息，方便排查问题.

注意：Linux 下部署安装 MySQL，默认不忽略表名大小写，需要手动到/etc/my.cnf 下配置 lower_case_table_names=1 使 Linux 环境下 MySQL 忽略表名大小写，否则使用 MyCAT 的时候会提示找不到表的错误！

4 核心配置详解

4.1 server.xml

mycat 需要的系统配置信息，配置系统相关参数，配置用户访问权限，配置 SQL 防火墙及 SQL 拦截功能

4.1.1 system 标签

这个标签内嵌套的所有 property 标签都与系统配置有关
如下格式：

```
<system>
    <property name="useSqlStat">0</property>
<system>
```

常用参数说明

参数	参数值	说明
charset	utf8	保证 mycat 的字符集和 mysql 的字符集是一致的

serverPort	8066	MyCAT 的默认端口，类似于 MySQL 的 3306 端口。
managerPort	9066	管理端口
nonePasswordLogin	0	
bingIp	0.0.0.0	
frontWriteQueueSize	2048	
txIsolation	2	
processors	8	
idleTimeout	1800000	指定连接的空闲时间超时长度，如果某个连接空闲时间超过该值，则将连接关闭并回收，单位为毫秒，默认值为 30 分钟
sqlExecuteTimeout	300	执行 sql 超时时间，默认为 300 秒
useSqlStat	0	1 为开启实时统计、0 为关闭
useGlobleTableCheck	0	1 为开启全局表一致性检测、0 为关闭
sequnceHandlerType	2	
defaultMaxLimit	100	
maxPacketSize	104857600	配置可以携带的数据量最大值，默认 16M

4.1.2 user 标签

作用：主要用于定义登录 mycat 的用户和权限

```
<user name="test">
   <property name="password">test</property>
   <property name="schemas">TESTDB</property>
   <property name="readOnly">true</property>
   <property name="benchmark">11111</property>
   <property name="usingDecrypt">1</property>
   <privileges check="false">
      <schema name="TESTDB" dml="0010" showTables="custome/mysql">
         <table name="tbl_user" dml="0110"/>
         <table name="tbl_dynamic" dml="1111"/>
      </schema>
   </privileges>
</user>
```

例如上面的例子中，我定义了一个用户，用户名为 test、密码也为 test，可访问的 schema 也只有 TESTDB 一个。 如果我在 schema.xml 中定义了多个 schema，那么这个用户是无法访问其他的 schema。在 mysql 客户端看来则是无法使用 use 切换到这个其他的数据库。

参数说明

参数	说明
name	指定用户名
password	指定密码，修改明文密码方式如下：
readOnle	true/false 来限制用户是否可读
schemas	控制用户可访问的 schema,多个用,隔开，例如： <property name="schemas">TESTDB,db1,db2</property>
Benchmark	Benchmark:mycat 连接服务降级处理： benchmark 基准，当前端的整体 connection 数达到基准值是，对来自该账户的请求开始拒绝连接，0 或不设表示不限制 例如 <property name="benchmark">1000</property>
usingDecrypt	是否对密码加密默认 0 否 如需要开启配置 1，同时使用加密程序对密码加密，加密命令为： 执行 mycat jar 程序： java -cp Mycat-server-1.6-RELEASE.jar io.mycat.util.DecryptUtil 0:root:123456 Mycat-server-1.4.1-dev.jar 为 mycat download 下载目录的 jar 1:host:user:password 中 0 为前端加密标志
privileges 子节点	对用户的 schema 及 下级的 table 进行精细化的 DML 权限控制
├── check	标识是否开启 DML 权限检查， 默认 false 标识不检查，当然 privileges 节点不配置，等同 check=false,由于 Mycat 一个用户的 schemas 属性可配置多个 schema ，所以 privileges 的下级节点 schema 节点同样可配置多个，对多库多表进行细粒度的 DML 权限控制

Schema/Table 上的 dml 属性描述:

参数	说明	示例（禁止增删改查）
dml	insert,update,select,delete	0000

注： 设置了 schema，但只设置了个别 table 或 未设置 table 的 DML，自动继承 schema 的 DML 属性

修改 root 明文 123456 的密码为加密的密码

```
[root@node1 ~]# cd /usr/local/mycat/lib/
[root@node1 lib]# java -cp Mycat-server-1.6-RELEASE.jar io.mycat.util.DecryptUtil
0:root:123456
GO0bnFVWrAuFgr1JMuMZkvfDNyTpoiGU7n/Wlsa151CirHQnANVk3NzE3FErx8v6pAc
O0ctX3xFecmSr+976QA==
```

拷贝加密的密码，修改 server.xml 里的配置

```
<user name="root">
<property name="usingDecrypt">1</property>
<property
name="password">GO0bnFVWrAuFgr1JMuMZkvfDNyTpoiGU7n/Wlsa151CirHQnANVk3NzE
3FErx8v6pAcO0ctX3xFecmSr+976QA==</property>
</user>
```

重新启动 mycat

```
mycat restart
```

登录 mycat

```
[root@node1 mycat]# mysql -uroot -p123456 -P9066 -h10.0.0.60
```

4.2 log4j2.xml

作用:

- ✧ 配置输出日志格式
- ✧ 配置输出日志的级别

日志格式:

```
<PatternLayout>
    <Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p [%t] (%l) - %m%n</Pattern>
</PatternLayout>
```

参数说明：

%d{yyyy-MM-dd HH:mm:ss.SSS} 表示日志的时间格式

%5p 表示输出的日志级别

%t 日志中记录线程名称

%m 输出代码中提示的信息

%n 输出一个回车换行符号，win 是 /r/n, linux 是/n

输出的日志级别：

```
<asyncRoot level="info" includeLocation="true">
</asyncRoot>
```

level 属性配置 mycat 日志级别，如下几种

all < trace < debug < info < warn < error < fatal < off

4.3 rule.xml

作用：

配置水平分片的分片规则

配置分片规则所对应的分片函数

rule.xml 里面就定义了我们对表进行拆分所涉及到的规则定义。我们可以灵活的对表使用不同的分片算法，或者对表使用相同的算法但具体的参数不同。这个文件里面主要有 tableRule 和 function 这两个标签。在具体使用过程中可以按照需求添加 tableRule 和 function。

4.3.1 tableRule 标签

这个标签定义表规则
如下：

```
<tableRule name="hash-mod-4_id">
    <rule>
        <columns>id</columns>
        <algorithm>hash-mod-4_id</algorithm>
    </rule>
</tableRule>
```

说明：
1）name 指定唯一的名字，用于标识不同的表规则。
这里设置的是 hash-mod-4_id，意思是 hash 取模的算法，并且分了 4 个，并对 id 列进行取模运算分片。见名知意。
2）内嵌的 rule 标签则指定对物理表中的哪一列进行拆分和使用什么路由算法。
3）columns 内指定要拆分的列名字。
4）algorithm 使用 function 标签中的 name 属性。连接表规则和具体路由算法。当然，多个表规则可以连接到同一个路由算法上。table 标签内使用。让逻辑表使用这个规则进行分片。

4.3.2 function 标签

```
<function name="hash-mod-4_id" class="io.mycat.route.function.PartitionByHashMod">
    <property name="count">4</property>
</function>
```

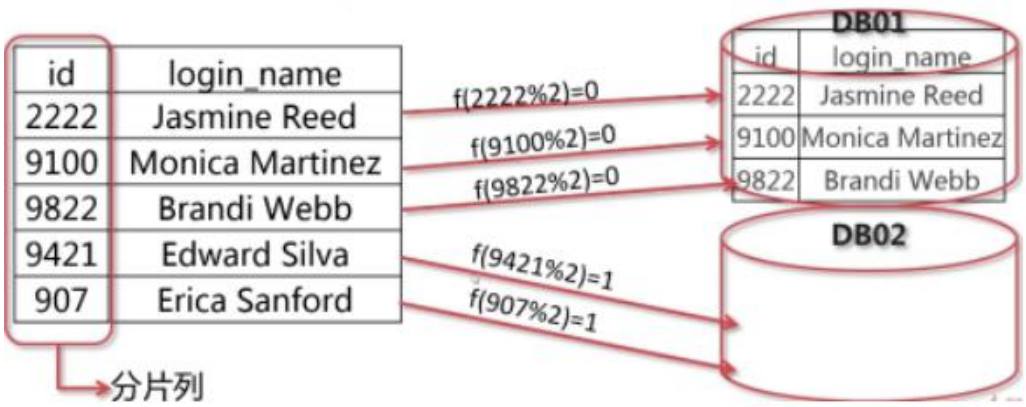
说明：
1）name 指定算法的名字。
2）class 制定路由算法具体的类名字。
3）property 为具体算法需要用到的一些属性。

4.3.3 分片算法

常见的有
简单取模-PartitionByMod
哈希取模-PartitionByHashMod
分片枚举-PartitionByFileMap
字符串范围取模分片
一致性 hash 分片

4.3.3.1 简单取模

示意图



示例：

```
<tableRule name="customer_login">
      <rule>
         <columns>cusotmer_id</columns>
         <algorithm>mod-long</algorithm>
      </rule>
 </tableRule>
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
      <property name="count">2</property>
</function>
```

对 customer_login 表进行分片，分片根据的列是 customer_login 表的 customer_id 列，分了 2 个片，分片的类方法是 io.mycat.route.function.PartitionByMod

说明：

1）可以用于分片列为整数类型的表

2）分片列 mod 分片基数

3）类全名：io.mycat.route.function.PartitionByMod

4.3.3.2 哈希取模

示例：

```
<tableRule name="customer_login">
      <rule>
         <columns>login_name</columns>
         <algorithm>mod-long</algorithm>
      </rule>
 </tableRule>
<function name="mod-long" class="io.mycat.route.function.PartitionByHashMod">
      <property name="count">2</property>
</function>
```

说明：

1）可以用于多种数据类型，如字符串，日期等进行分片

2）hash（分片列）mod 分片基数
3）类全名：io.mycat.route.function.PartitionByHashMod

4.3.3.3 枚举分片

示例：

```
<tableRule name="sharding-by-intfile">
      <rule>
         <columns>user_id</columns>
         <algorithm>hash-int</algorithm>
      </rule>
   </tableRule>
<function name="hash-int" class="io.mycat.route.function.PartitionByFileMap">
      <property name="mapFile">partition-hash-int.txt</property>
      <property name="type">0</property>
      <property name="defaultNode">0</property>
   </function>
```

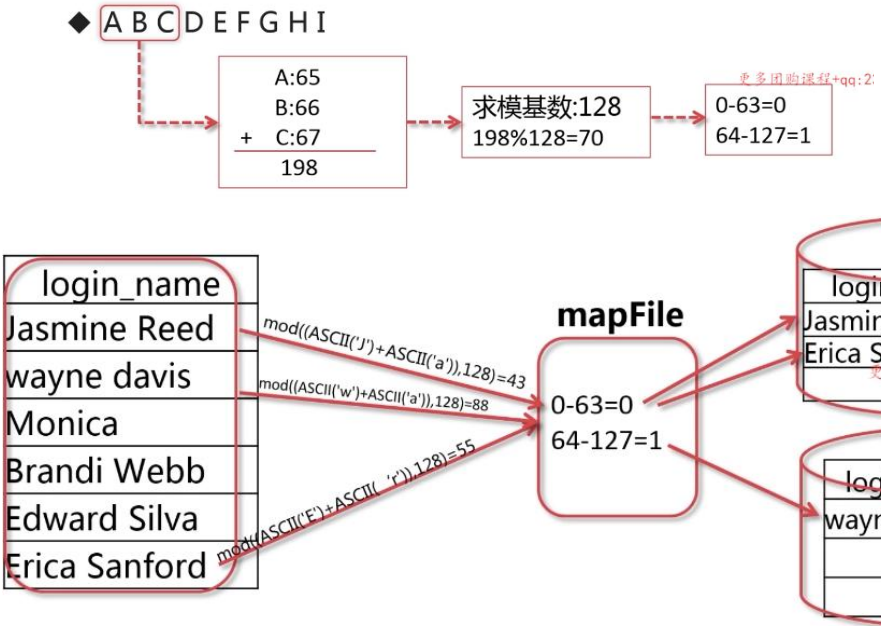
说明：
1）partition-hash-int.txt 配置：
10000=0
10010=1
2）上面 columns 标识将要分片的表字段，algorithm 分片函数，
其中分片函数配置中，mapFile 标识配置文件名称，type 默认值为 0，0 表示 Integer，非零表示 String，
3）所有的节点配置都是从 0 开始，及 0 代表节点 1
4）defaultNode
默认节点:>=0 表示启用默认节点，<0 表示不启用默认节点
默认节点的作用：枚举分片时，如果碰到不识别的枚举值，就让它路由到默认节点，如果不配置默认节点（defaultNode 值小于 0 表示不配置默认节点），碰到不识别的枚举值就会报错，like this：can't find datanode for sharding column:column_name val:ffffffff

可以根据可能的枚举值指定数据存储的位置
$MYCAT/conf 目录下增加 MapFile 配置枚举值同节点的对应关系
类全名：io.mycat.route.function.PartitionByFileMap

4.3.3.4 字符串范围取模分片

图示：

字符串范围取模分片-PartitionByPrefixPattern



示例：

```
<tableRule name="sharding-by-prefix-pattern">
      <rule>
            <columns>user_id</columns>
            <algorithm>sharding-by-prefix-pattern</algorithm>
      </rule>
</tableRule>
<function name="sharding-by-prefix-pattern"
class="io.mycat.route.function.PartitionByPrefixPattern">
      <property name="patternValue">128</property>
      <property name="prefixLength">2</property>
      <property name="mapFile">prefix-partition-pattern.txt</property>
</function>
```

说明：

可以根据指定字符串的前 N 个字符确定存储位置

$MYCAT/conf 增加 MapFile 配置取模范围同节点的对应关系

类全名：io.mycat.route.function.PartitionByPrefixPattern

4.3.3.5 一致性 hash 算法

```
<tableRule name="sharding-by-murmur">
        <rule>
          <columns>user_id</columns>
          <algorithm>murmur</algorithm>
        </rule>
    </tableRule>
```

```
<function name="murmur" class="io.mycat.route.function.PartitionByMurmurHash">
        <property name="seed">0</property>
        <!-- 默认是 0-->
        <property name="count">2</property>
        <!-- 要分片的数据库节点数量，必须指定，否则没法分片—>
        <property name="virtualBucketTimes">160</property>
        <!-- 一个实际的数据库节点被映射为这么多虚拟节点，默认是 160 倍，也就是虚拟节点数是物理节点数的 160 倍-->
        <!--
        <property name="weightMapFile">weightMapFile</property>
          节点的权重，没有指定权重的节点默认是 1。以 properties 文件的格式填写，以从 0 开始到 count-1 的整数值也就是节点索引为 key，以节点权重值为值。所有权重值必须是正整数，否则以 1 代替 -->
        <!--
        <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>
        用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的 murmur hash 值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西 -->
</function>
```

4.4 schema.xml

用途：
配置逻辑库和逻辑表
配置逻辑表所存储的数据节点
配置数据节点所对应的物理数据库服务器信息
Schema.xml 作为 MyCat 中重要的配置文件之一，管理着 MyCat 的逻辑库、表、分片规则、DataNode 以及 DataSource。弄懂这些配置，是正确使用 MyCat 的前提。

4.4.1 schema 标签

定义逻辑库

```
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100"></schema>
```

schema 标签用于定义 MyCat 实例中的逻辑库，MyCat 可以有多个逻辑库，每个逻辑库都有自己的相关配置。可以使用 schema 标签来划分这些不同的逻辑库。
如果不配置 schema 标签，所有的表配置，会属于同一个默认的逻辑库。

```
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">
    <table name="travelrecord" dataNode="dn1,dn2,dn3"
rule="auto-sharding-long" ></table>
</schema>
<schema name="USERDB" checkSQLschema="false" sqlMaxLimit="100">
    <table name="company" dataNode="dn10,dn11,dn12"
rule="auto-sharding-long" ></table>
```

```
</schema>
```

如上所示的配置就配置了两个不同的逻辑库，逻辑库的概念和 MYSQL 数据库中 Database 的概念相同，我们在查询这两个不同的逻辑库中表的时候需要切换到该逻辑库下才可以查询到所需要的表。

```
mysql> use USERDB;
Database changed
mysql> select * from company;
Empty set (0.09 sec)

mysql> select * from travelrecord;
ERROR 1064 (HY000): can't find table define in schema ,table:TRAVELRECORD schema
:USERDB
mysql>
```

如果你发现显示该错误信息，需要到 server.xml 添加该用户可以访问到的 schema 就可以了。

说明：

1）name 定义逻辑库的名字

2）checkSQLschema ，检查判断是否检查发给 mycat 的 SQL 是否含库名。操作的时候，最好不要带 db 的方式，如：db.table，会过滤掉 db；

3）sqlMaxLimit 限制返回结果集的行数，-1 表示关闭 limit 限制

4.4.2 table 标签

Table 标签定义了 MyCat 中的逻辑表，所有需要拆分的表都需要在这个标签中定义。

4.4.2.1 name 属性

定义逻辑表的表名，这个名字就如同我在数据库中执行 create table 命令指定的名字一样，同个 schema 标签中定义的名字必须唯一。

4.4.2.2 dataNode 属性

定义这个逻辑表所属的 dataNode，该属性的值需要和 dataNode 标签中 name 属性的值相互对应。如果需要定义的 dn 过多可以使用如下的方法减少配置：

```
<table name="travelrecord" dataNode="multipleDn$0-99,multipleDn2$100-199"
rule="auto-sharding-long" ></table>
<dataNode name="multipleDn" dataHost="localhost1" database="db$0-99" ></dataNode>
<dataNode name="multipleDn2" dataHost="localhost1" database=" db$0-99" ></dataNode>
```

这里需要注意的是 database 属性所指定的真实 database name 需要在后面添加一个，例如上面的例子中，我需要在真实的 mysql 上建立名称为 dbs0 到 dbs99 的 database。

4.4.2.3 rule 属性

该属性用于指定逻辑表要使用的规则名字，规则名字在 rule.xml 中定义，必须与 tableRule 标签中 name 属性属性值一一对应。

4.4.2.4 ruleRequired 属性

该属性用于指定表是否绑定分片规则，如果配置为 true，但没有配置具体 rule 的话 ，程序会报错。

4.4.2.5 primaryKey 属性

该逻辑表对应真实表的主键，例如：分片的规则是使用非主键进行分片的，那么在使用主键查询的时候，就会发送查询语句到所有配置的 DN 上，如果使用该属性配置真实表的主键。难么 MyCat 会缓存主键与具体 DN 的信息，那么再次使用非主键进行查询的时候就不会进行广播式的查询，就会直接发送语句给具体的 DN，但是尽管配置该属性，如果缓存并没有命中的话，还是会发送语句给具体的 DN，来获得数据。

4.4.2.6 type 属性

该属性定义了逻辑表的类型，目前逻辑表只有"全局表"和"普通表"两种类型。对应的配置：

- ✧ 全局表：global。
- ✧ 普通表：不指定该值为 globla 的所有表。

4.4.2.7 autoIncrement 属性

mysql 对非自增长主键，使用 last_insert_id()是不会返回结果的，只会返回 0。所以，只有定义了自增长主键的表才可以用 last_insert_id()返回主键值。

mycat 目前提供了自增长主键功能，但是如果对应的 mysql 节点上数据表，没有定义 auto_increment，那么在 mycat 层调用 last_insert_id()也是不会返回结果的。

由于 insert 操作的时候没有带入分片键，mycat 会先取下这个表对应的全局序列，然后赋值给分片键。这样才能正常的插入到数据库中，最后使用 last_insert_id()才会返回插入的分片键值。

如果要使用这个功能最好配合使用数据库模式的全局序列。

使用 autoIncrement="true" 指定这个表有使用自增长主键，这样 mycat 才会不抛出分片键找不到的异常。

使用 autoIncrement="false" 来禁用这个功能，当然你也可以直接删除掉这个属性。默认就是禁用的。

4.4.2.8 subTables

使用方式添加 subTables="t_order$1-2,t_order3"
目前分表 1.6 以后开始支持 并且 dataNode 在分表条件下只能配置一个，分表条件下不支持各种条件的 join 语句。

4.4.2.9 needAddLimit 属性

指定表是否需要自动的在每个语句后面加上 limit 限制。由于使用了分库分表，数据量有时会特别巨大。这时候执行查询语句，如果恰巧又忘记了加上数量限制的话。那么查询所有的数据出来，也够等上一小会儿的。
所以，mycat 就自动的为我们加上 LIMIT 100。当然，如果语句中有 limit，就不会在次添加了。
这个属性默认为 true,你也可以设置成 false 禁用掉默认行为。

4.4.3 dataNode 标签

dataNode 标签定义了 MyCat 中的数据节点，也就是我们通常说所的数据分片。一个 dataNode 标签就是一个独立的数据分片。
dataNode 定义逻辑表存储的物理数据库

```
<dataNode name="dn1" dataHost="lch3307" database="db1" ></dataNode>
```

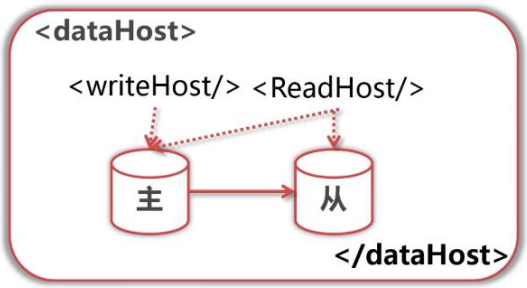
说明：
1）name　　属性定义数据节点的名称，必须唯一，我们需要在 table 标签上应用这个名字，来建立表与分片对应的关系。
2）dataHost 属性定义分片所在的物理主机。
3）database 属性定义物理数据库名称。

4.4.4 dataHost 标签

定义了具体的数据库实例、读写分离配置和心跳语句

4.4.4.1 dataHost 属性和作用

<dataHost>定义后端数据库主机信息




```
<dataHost name="mysql0103" maxCon="1000" minCon="10" balance="1" writeType="0"
dbType="mysql" dbDriver="native">
	<heartbeat>select user()</heartbeat>
	<writeHost host="192.168.1.3" url="192.168.1.3:3306" user="root"
password="123456">
		<readHost host="192.168.1.4" url="192.168.1.4:3306" user="root"
password="123456" />
	</writeHost>
</dataHost>
```

说明：

1）name 属性：

唯一标识 dataHost 标签，供上层的标签使用。

2）maxCon 属性：

指定每个读写实例连接池的最大连接。也就是说，标签内嵌套的 writeHost、readHost 标签都会使用这个属性的值来实例化出连接池的最大连接数。

3）minCon 属性：

指定每个读写实例连接池的最小连接，初始化连接池的大小。

4）balance 属性：

1. balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的 writeHost 上。

2. balance="1"，全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡，简单的说，当双主双从模式(M1->S1，M2->S2，并且 M1 与 M2 互为主备)，正常情况下，M2,S1,S2 都参与 select 语句的负载均衡。

3. balance="2"，所有读操作都随机的在 writeHost、readhost 上分发。

4. balance="3"，所有读请求随机的分发到 wiriterHost 对应的 readhost 执行，writerHost 不负担读压力，注意 balance=3 只在 1.4 及其以后版本有，1.3 没有。

5）writeType 属性：

负载均衡类型，目前的取值有 3 种：

1. writeType="0", 所有写操作发送到配置的第一个 writeHost，第一个挂了切到还生存的第二个 writeHost，重新启动后已切换后的为准，切换记录在配置文件中:dnindex.properties .

2. writeType="1"，所有写操作都随机的发送到配置的 writeHost，1.5 以后废弃不推荐。

switchType 属性

- -1 表示不自动切换

- 1 默认值，自动切换
- 2 基于 MySQL 主从同步的状态决定是否切换

6）dbType 属性：
指定后端连接的数据库类型，目前支持二进制的 mysql 协议，还有其他使用 JDBC 连接的数据库。例如：mongodb、oracle、spark 等。
7）dbDriver 属性：
指定连接后端数据库使用的 Driver，目前可选的值有 native 和 JDBC。使用 native 的话，因为这个值执行的是二进制的 mysql 协议，所以可以使用 mysql 和 maridb。其他类型的数据库则需要使用 JDBC 驱动来支持。
从 1.6 版本开始支持 postgresql 的 native 原始协议。
如果使用 JDBC 的话需要将符合 JDBC 4 标准的驱动 JAR 包放到 MYCAT\lib 目录下，并检查驱动 JAR 包中包括如下目录结构的文件：META-INF\services\java.sql.Driver。在这个文件内写上具体的 Driver 类名，例如：com.mysql.jdbc.Driver。
8）switchType 属性：
-1 表示不自动切换
1 默认值，自动切换
2 基于 MySQL 主从同步的状态决定是否切换
心跳语句为 show slave status
3 基于 MySQL galary cluster 的切换机制（适合集群）（1.4.1）
心跳语句为 show status like 'wsrep%'

9）tempReadHostAvailable 属性：
如果配置了这个属性 writeHost 下面的 readHost 仍旧可用，默认 0 可配置（0、1）。

4.4.4.2 heartbeat 标签

心跳检查语句
mysql: select user(),主从切换的语句必须是 show slave status

4.4.4.3 writeHost 标签、readHost 标签和其属性

指定后端数据库的相关配置给 mycat,用于实例化后端连接池，writeHost 指定写实例、readHost 指定读实例
在一个 dataHost 内可以定义多个 writeHost 和 readHost。但是，如果 writeHost 指定的后端数据库宕机，那么这个 writeHost 绑定的所有 readHost 都将不可用。另一方面，由于这个 writeHost 宕机系统会自动的检测到，并切换到备用的 writeHost 上去。
说明：
1）host 属性
用于标识不同实例，一般 writeHost 我们使用*M1，readHost 我们用*S1
2）url 属性
后端实例连接地址，如果是使用 native 的 dbDriver，则一般为 address:port 这种形式。用 JDBC 或其他的 dbDriver，则需要特殊指定。当使用 JDBC 时则可以这么写：jdbc:mysql://localhost:3306/。
3）user 属性

后端存储实例需要的用户名字
4）password 属性
后端存储实例需要的密码
5）weight 属性
权重 配置在 readhost 中作为读节点的权重（1.4 以后）
6）usingDecrypt 属性
是否对密码加密默认 0 否 如需要开启配置 1，同时使用加密程序对密码加密，加密命令为：

```
java -cp Mycat-server-1.6-RELEASE.jar io.mycat.util.DecryptUtil 0:root:123456
```

4.4.5 schema.xml 完整配置示例

完整配置

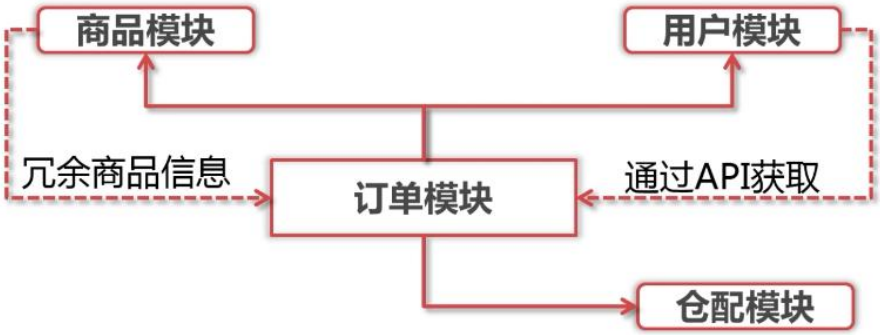
```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

    <schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">

        <table name="customer_login" primaryKey="customer_id" dataNode="logindb01,logindb02" rule="
        customer_login"/>
    </schema>


    <dataNode name="logindb01" dataHost="mysql0103" database="login_db01" />
    <dataNode name="logindb02" dataHost="mysql0103" database="login_db02" />

    <dataHost name="mysql0103" maxCon="1000" minCon="10" balance="1"
              writeType="0" dbType="mysql" dbDriver="native" switchType="1">
        <heartbeat>select user()</heartbeat>
        <writeHost host="192.168.1.3" url="192.168.1.3:3306" user="im_mycat" password="123456">
           <readHost host="192.168.1.4" url="192.168.1.4:3306" user="im_user" password="123456" />
        </writeHost>
        <writeHost host="192.168.1.4" url="192.168.1.4:3306" user="im_user" password="123456" />
    </dataHost>


</mycat:schema>
```

5 实战之垂直分库

5.1 垂直分库说明

相关联的数据，模块化划分，例如：用户模块，商品模块，订单/仓配模块；
垂直分库的步骤：
1. 手机分析业务模块间的关系
2. 复制数据库到其他实例
3. 配置 mycat 垂直分库
4. 通过 mycat 访问 DB
5. 删除原库中已迁移表

收集分析业务模块间的关系



5.2 复制数据库到其他实例

1 备份原数据库并记录相关事物点
2 在原数据库中建立复制用户
3 在新实例上恢复备份数据库
4 在新实例上配置复制链路
5 在新实例上启动复制

5.3 环境说明

演示环境的说明

主机名	IP	角色	数据库
node1	10.0.0.62	mycat mysql	imooc_db
node2	10.0.0.63	mysql	order_db
node3	10.0.0.64	mysql	product_db
node4	10.0.0.65	mysql	customer_db

5.4 MySQL 复制实战

0 主节点修改配置重启 MySQL

```
[root@node1 tools]# egrep "log_bin|server-id" /etc/my.cnf
server-id = 1
log_bin = mysql-bin
# 从节点 server-id 分别是唯一的，比如 3,5,7;从节点的 log_bin 关闭掉
```

1 导入事先准备好的数据

imooc_db.sql，将数据创建到 node1 节点

```
mysql -uroot -p123456 -e "create database imooc_db"
mysql -uroot -p123456 imooc_db < imooc_db.sql
mysql -uroot -p123456
mysql> show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| imooc_db           |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

2 主节点创建同步账号，授权

node1:

```
[root@node1 tools]# mysql -uroot -p123456 -e "grant replication slave on *.* to
'im_repl'@'10.0.0.%' identified by '123456'"
[root@node1 tools]# mysql -uroot -p123456 -e "flush privileges"
[root@node1 tools]# mysql -uroot -p123456 -e "show master status"
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000001 |    34854 |              |                  |                   |
|
+------------------+----------+--------------+------------------+-------------------+
```

3 主库数据备份

备份 node1 节点的全量数据

```
[root@node1 tools]# mysqldump -uroot -p123456 --master-data=2 --single-transaction
--routines --triggers --events imooc_db > bak_imooc.sql
```

4 三个几点数据同步

拷贝到其他三个节点

```
root@node1 tools]# scp bak_imooc.sql root@10.0.0.63:/root
root@node1 tools]# scp bak_imooc.sql root@10.0.0.64:/root
root@node1 tools]# scp bak_imooc.sql root@10.0.0.65:/root
```

三个节点都导入数据，注意库名称，已经修改，三个节点的库名都不同

node2:

```
[root@node2 ~]# mysql -uroot -p123456 -e "create database order_db"
[root@node2 ~]# mysql -uroot -p123456 order_db < bak_imooc.sql
[root@node2 ~]# mysql -uroot -p123456 -e "show databases;"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| order_db           |
```

```
| performance_schema |
| test               |
+--------------------+
```

node3:

```
[root@node3 ~]# mysql -uroot -p123456 -e "create database product_db"
[root@node3 ~]# mysql -uroot -p123456 product_db < bak_imooc.sql
```

node4:

```
[root@node4 ~]# mysql -uroot -p123456 -e "create database customer_db"
[root@node4 ~]# mysql -uroot -p123456 customer_db <bak_imooc.sql
```

5 三个从节点创建复制链路

node2,node3, node4

```
mysql -uroot -p123456 -e "change master to
master_host='10.0.0.62',master_user='im_repl',master_password='123456',master_port=3306,
master_log_file='mysql-bin.000001',master_log_pos=35040;"
```

6 垂直拆分，数据库名，表名都不一致，设置主从复制过滤

mysql 5.7

node2:

```
mysql -uroot -p123456 -e "change replication filter
replicate_rewrite_db=((imooc_db,order_db));"
```

\h change replication filter 查看帮助

node3:

```
mysql -uroot -p123456 -e "change replication filter
replicate_rewrite_db=((imooc_db,product_db));"
```

node4:

```
mysql -uroot -p123456 -e "change replication filter
replicate_rewrite_db=((imooc_db,customer_db));"
```

mysql5.5

node2:

```
vi /etc/my.cnf
replicate-rewrite-db=imooc_db->order_db
```

node3:

```
vi /etc/my.cnf
replicate-rewrite-db=product_db->order_db
```

node4:

```
vi /etc/my.cnf
replicate-rewrite-db=customer_db->order_db
```

7 启动三个从节点

```
mysql -uroot -p123456 -e "show slave status\G"
mysql -uroot -p123456 -e "start slave;"
mysql -uroot -p123456 -e "show slave status\G" |egrep
"Slave_IO_Running|Slave_SQL_Running|Seconds_Behind_Master"

                  Slave_IO_Running: Yes
                 Slave_SQL_Running: Yes
           Seconds_Behind_Master: 0
```

8 测试

在主库上修改数据，从库上查看

node1:

```
[root@node1 tools]# mysql -uroot -p123456 -e "create database slave_test_db"
[root@node1 tools]# mysql -uroot -p123456 -e "show databases"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| imooc_db           |
| mysql              |
| performance_schema |
| slave_test_db      |
| test               |
+--------------------+
```

node2,3,4:

```
[root@node2 ~]# mysql -uroot -p123456 -e "show databases"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| order_db           |
| performance_schema |
| slave_test_db      |
| test               |
+--------------------+
```

5.5 mycat 垂直切分配置

- ✧ 配置 mycat 垂直分库步骤
- ✧ 使用 schema.xml 配置逻辑库
- ✧ 使用 server.xml 配置系统变量及用户权限
- ✧ 由于没有用到水平分片，所以不用配置 rule.xml

5.5.1 schema.xml 配置逻辑库

1 主节点，创建同步的用户

node1:

```
[root@node1 ~]# mysql -uroot -p123456 -e "create user im_mycat@'10.0.0.%' identified by '123456'"
[root@node1 ~]# mysql -uroot -p123456 -e "grant select,insert,update,delete on *.* to im_mycat@'10.0.0.%'"
[root@node1 tools]#    mysql -uroot -p123456 -e "flush privileges;"
```

主节点配置 schema.xml 的 dataHost

下面只是测试，用的一组 writeHost，生产环境会有主从，可以先配置 dataHost 信息，再配置 dataNode 信息，最后配置 schema 和 table 信息；因为他们是一层层依赖的，从上到下，那么配置的时候，可以从下到上配置；

schema.xml 配置如下：

[root@node1 ~]# vim /usr/local/mycat/conf/schema.xml

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
    <schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">
        <table name="order_master" primaryKey="order_id" dataNode="ordb" />
        <table name="order_detail" primaryKey="order_detail_id" dataNode="ordb" />
        <table name="order_cart" primaryKey="cart_id" dataNode="ordb" />
        <table name="order_customer_addr" primaryKey="customer_addr_id" dataNode="ordb" />
        <table name="region_info" primaryKey="region_id" dataNode="ordb" />
        <table name="shipping_info" primaryKey="ship_id" dataNode="ordb" />
        <table name="warehouse_info" primaryKey="w_id" dataNode="ordb" />
        <table name="warehouse_proudct" primaryKey="wp_id" dataNode="ordb" />

        <table name="product_brand_info" primaryKey="brand_id" dataNode="prodb" />
        <table name="product_category" primaryKey="category_id" dataNode="prodb" />
        <table name="product_comment" primaryKey="comment_id" dataNode="prodb" />
        <table name="product_info" primaryKey="product_id" dataNode="prodb" />
        <table name="product_supplier_info" primaryKey="supplier_id" dataNode="prodb" />
        <table name="product_pic_info" primaryKey="product_pic_id" dataNode="prodb" />

        <table name="customer_balance_log" primaryKey="balance_id" dataNode="custdb" />
```

```
        <table name="customer_inf" primaryKey="customer_inf_id" dataNode="custdb"
/>
        <table name="customer_level_inf" primaryKey="balance_id" dataNode="custdb"
/>
        <table name="customer_login" primaryKey="login_id" dataNode="custdb" />
        <table name="customer_login_log" primaryKey="login_id" dataNode="custdb" />
        <table name="customer_point_log" primaryKey="point_id" dataNode="custdb" />

    </schema>

    <dataNode name="ordb" dataHost="mysql0103" database="order_db" />
    <dataNode name="prodb" dataHost="mysql0104" database="product_db" />
    <dataNode name="custdb" dataHost="mysql0105" database="customer_db" />

    <dataHost name="mysql0103" maxCon="1000" minCon="10" balance="3"
writeType="0" dbType="mysql" dbDriver="native"    switchType="1">
        <heartbeat>select user()</heartbeat>
        <writeHost host="10.0.0.63" url="10.0.0.63:3306" user="im_mycat"
password="123456"></writeHost>
    </dataHost>
    <dataHost name="mysql0104" maxCon="1000" minCon="10" balance="3"
writeType="0" dbType="mysql" dbDriver="native"    switchType="1">
        <heartbeat>select user()</heartbeat>
        <writeHost host="10.0.0.64" url="10.0.0.64:3306" user="im_mycat"
password="123456"></writeHost>
    </dataHost>
    <dataHost name="mysql0105" maxCon="1000" minCon="10" balance="3"
writeType="0" dbType="mysql" dbDriver="native"    switchType="1">
        <heartbeat>select user()</heartbeat>
        <writeHost host="10.0.0.65" url="10.0.0.65:3306" user="im_mycat"
password="123456"></writeHost>
    </dataHost>
</mycat:schema>
```

5.5.2 server.xml 配置系统变量及用户权限

[root@node1 ~]# vim /usr/local/mycat/conf/server.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>
        <property name="managerPort">9066</property>
        <property name="nonePasswordLogin">0</property>
```

```
        <property name="bindIp">0.0.0.0</property>
        <property name="frontWriteQueueSize">2048</property>
        <property name="charset">utf8</property>
        <property name="txIsolation">2</property>
        <property name="processors">8</property>
        <property name="idleTimeout">1800000</property>
        <property name="sqlExecuteTimeout">300</property>
        <property name="useGlobleTableCheck">0</property>
        <property name="sequnceHandlerType">2</property>
        <property name="defaultMaxLimit">100</property>
        <property name="maxPacketSize">104857600</property>
    </system>

    <user name="app_imooc" defaultAccount="true">
        <property name="password">123456</property>
        <property name="schemas">imooc_db</property>
    </user>

</mycat:server>
```

可以对 app_imooc 用户的密码，进行加密，这个是 mycat 的管理账号

5.5.3 后续工作

1)切换应用通过 mycat 连接数据库,数据管理也是 mycat

2)从库上验证数据是否可以查询
比如 node4 验证:

```
[root@node4 ~]# mysql -uroot -p123456 -e "use customer_db;select count(*) from
product_brand_info;"
+----------+
| count(*) |
+----------+
|       17 |
+----------+
```

5.5.4 启动 mycat

```
mycat restart
```

查看日志，看启动是否正常,查看日志解决问题

```
[root@node1 ~]# netstat -lntup|grep 8066
tcp        0      0 :::8066                     :::*
LISTEN      68411/java
```

5.5.5 mycat 验证配置

从节点登陆验证

```
[root@node4 ~]# mysql -uapp_imooc -p123456 -P8066 -h10.0.0.62
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.29-mycat-1.6.5-release-20180122220033 MyCat Server (OpenCloundDB)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

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 input statement.

mysql>
```

黄色部分，是 mycat 本部信息，表示正常

逻辑库信息

```
mysql> show databases;
+----------+
| DATABASE |
+----------+
| imooc_db |
+----------+
```

逻辑表信息

```
mysql> use imooc_db;
Database changed
mysql> show tables;
+----------------------+
| Tables in imooc_db   |
+----------------------+
| customer_balance_log |
| customer_inf         |
| customer_level_inf   |
| customer_login       |
| customer_login_log   |
| customer_point_log   |
| order_cart           |
| order_customer_addr  |
| order_detail         |
| order_master         |
```

```
| product_brand_info       |
| product_category         |
| product_comment          |
| product_info             |
| product_pic_info         |
| product_supplier_info |
| region_info              |
| shipping_info            |
| warehouse_info           |
| warehouse_proudct        |
+----------------------+
20 rows in set (0.01 sec)
```

5.5.6 清理多余数据

node 2,3,4 上的多余数据清理
1 停掉从节点的主从同步
node 2,3,4 都关闭

```
mysql -uroot -p123456 -e "stop slave"
mysql -uroot -p123456 -e "reset slave all"
mysql -uroot -p123456 -e "show slave status\G"
```

清理每个节点多余的数据

提示：表很多的话，用脚本处理，生产环境，一定要进行备份

节点 node2 最后保留的表 如下：

```
[root@node2 ~]# mysql -uroot -p123456 -e "use order_db;drop tables
customer_balance_log;drop tables customer_inf;drop tables customer_level_inf; drop tables
customer_login;drop tables customer_login_log;drop tables customer_point_log;drop tables
product_brand_info;drop tables product_category;drop tables product_comment;drop tables
product_info;drop tables product_pic_info;drop tables product_supplier_info;"

[root@node2 ~]# mysql -uroot -p123456 -e "use order_db;show tables;"
+--------------------+
| Tables_in_order_db |
+--------------------+
| order_cart         |
| order_customer_addr |
| order_detail       |
| order_master       |
| region_info        |
| serial             |
| shipping_info      |
| warehouse_info     |
```

```
| warehouse_proudct     |
+----------------------+
```

节点 3 保留的表

```
[root@node3 ~]# mysql -uroot -p123456 -e "use product_db;drop tables
customer_balance_log;drop tables customer_inf;drop tables customer_level_inf; drop tables
customer_login;drop tables customer_login_log;drop tables customer_point_log;drop tables
order_cart; drop tables order_customer_addr;drop tables order_detail; drop tables
order_master; drop tables serial;drop tables shipping_info;drop tables warehouse_info;drop
tables warehouse_proudct;drop tables region_info;"

[root@node3 ~]# mysql -uroot -p123456 -e "use product_db;show tables;"
+----------------------+
| Tables_in_product_db    |
+----------------------+
| product_brand_info       |
| product_category          |
| product_comment             |
| product_info                 |
| product_pic_info           |
| product_supplier_info |
+----------------------+
```

节点 4 保留的表

```
[root@node4 ~]# mysql -uroot -p123456 -e "use customer_db;drop tables order_cart;drop
tables order_customer_addr; drop tables order_detail; drop tables order_master;drop tables
region_info;drop tables serial; drop tables shipping_info; drop tables warehouse_info; drop
tables warehouse_proudct; drop tables product_brand_info; drop tables product_category;
drop tables product_comment; drop tables product_info;drop tables product_pic_info;drop
tables product_supplier_info;"

[root@node4 ~]# mysql -uroot -p123456 -e "use customer_db;show tables;"
+----------------------+
| Tables_in_customer_db |
+----------------------+
| customer_balance_log    |
| customer_inf                |
| customer_level_inf       |
| customer_login              |
| customer_login_log       |
| customer_point_log       |
+----------------------+
```

登陆验证

```
[root@node4 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> show databases;
+----------+
| DATABASE |
+----------+
| imooc_db |
+----------+
mysql> use imooc_db;
Database changed
mysql> show tables;
+-----------------------+
| Tables in imooc_db    |
+-----------------------+
| customer_balance_log  |
| customer_inf          |
| customer_level_inf    |
| customer_login        |
| customer_login_log    |
| customer_point_log    |
| order_cart            |
| order_customer_addr   |
| order_detail          |
| order_master          |
| product_brand_info    |
| product_category      |
| product_comment       |
| product_info          |
| product_pic_info      |
| product_supplier_info |
| region_info           |
| shipping_info         |
| warehouse_info        |
| warehouse_proudct     |
+-----------------------+
mysql> select count(*) from product_brand_info;
mysql> select count(*) from product_brand_info;
+--------+
| COUNT0 |
+--------+
|     17 |
+--------+
```

5.5.7 跨分片查询

5.5.7.1 案例

节点 4，查询 sql 如下：跨分片查询

跨分片查询报错

```
mysql> select supplier_name,b.region_name as '省',c.region_name as '市',d.region_name as '区'
from product_supplier_info a
join region_info b on b.region_id=a.province
join region_info c on c.region_id=a.city
join region_info d on d.region_id=a.district;

ERROR 1064 (HY000): invalid route in sql, multi tables found but datanode has no intersection    sql:select supplier_name,b.region_name as '省',c.region_name as '市',d.region_name as '区'
from product_supplier_info a
join region_info b on b.region_id=a.province
join region_info c on c.region_id=a.city
join region_info d on d.region_id=a.district
```

5.5.7.2 解决方法

方式 1：前端 api 调用，请求数据频次高，不适合
方式 2：表数据冗余设计，数据冗余过多，修过的冗余字段会很多
方式 3：全局表方式

5.5.7.3 全局表解决方法步骤

1）全局表：将要连表查询的数据，region_info，拷贝到节点 3,4 中
节点 2 导出数据
node2:

```
[root@node2 ~]# mysqldump -uroot -p123456 order_db region_info > region_info
```

备份到节点 3 ，节点 4
node2:

```
[root@node2 ~]# scp region_info root@10.0.0.64:/root
[root@node2 ~]# scp region_info root@10.0.0.65:/root
```

节点 3,4 分别导入备份数据
node3:

```
[root@node3 ~]# mysql -uroot -p123456 product_db < region_info
```

node4:

```
[root@node4 ~]# mysql -uroot -p123456 customer_db < region_info
```

登陆查看

```
[root@node4 ~]# mysql -uroot -p123456
mysql> use customer_db
mysql> show tables;
+----------------------+
| Tables_in_customer_db |
+----------------------+
| customer_balance_log    |
| customer_inf              |
| customer_level_inf       |
| customer_login            |
| customer_login_log       |
| customer_point_log       |
| region_info                |
+----------------------+
```

2）修改逻辑表 schema.xml

node1 节点:

[root@node1 conf]# vim /usr/local/mycat/conf/schema.xml

```
<table name="order_master" primaryKey="order_id" dataNode="ordb" />
<table name="order_detail" primaryKey="order_detail_id" dataNode="ordb" />
<table name="order_cart" primaryKey="cart_id" dataNode="ordb" />
<table name="order_customer_addr" primaryKey="customer_addr_id" dataNode="ordb" />
<table name="region_info" primaryKey="region_id" dataNode="ordb,prodb,custdb" type="global" />
<table name="shipping_info" primaryKey="ship_id" dataNode="ordb" />
<table name="warehouse_info" primaryKey="w_id" dataNode="ordb" />
<table name="warehouse_proudct" primaryKey="wp_id" dataNode="ordb" />

<table name="product_brand_info" primaryKey="brand_id" dataNode="prodb" />
<table name="product_category" primaryKey="category_id" dataNode="prodb" />
```

重启 mycat

```
[root@node1 conf]# mycat restart
```

检验跨分片关联查询

```
[root@node4 ~]#    mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
use imooc_db;
mysql> select supplier_name,b.region_name as '省',c.region_name as '市',d.region_name as '区'
  from product_supplier_info a
join region_info b on b.region_id=a.province
join region_info c on c.region_id=a.city
join region_info d on d.region_id=a.district;
+---------------+--------+--------+-------------+
```

```
| supplier_name | 省        | 市        | 区              |
+---------------+--------+--------+-------------+
| 供应商-1         | 上海     | 黄浦     | 外滩 18 号      |
+---------------+--------+--------+-------------+
```

提示: 创建了全局表，一定要通过 mycat 来管理数据，否则全局表的数据，会出现不一致

测试：节点 4 上，修改数据

```
mysql> update region_info set region_name='中华国' where region_id=1;
```

报错，解决如下：

```
mysql> update region_info set region_name='中国' where region_id=1;
ERROR 1105 (HY000): The MySQL server is running with the --read-only option so it cannot execute this statement
```

关闭节点 2，3，4 read_only 属性

mysql 端

```
Type 'help;' or '\h' for help. Type '\c' to

mysql> set global read_only=off;
```

登陆的是 mycat 端，查看数据已经修改

```
[root@node1 conf]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> use imooc_db;
mysql> select * from region_info where region_id=1;
+-----------+-----------+-------------+--------------+
| region_id | parent_id | region_name | region_level |
+-----------+-----------+-------------+--------------+
|         1 |         0 | 中华国          |            1 |
+-----------+-----------+-------------+--------------+
```

登陆 2,3,4 节点 mysql 端 ，查看物理数据库信息

```
mysql -uroot -p123456
mysql> use order_db;
mysql> select * from region_info where region_id=1;
+-----------+-----------+-------------+--------------+
| region_id | parent_id | region_name | region_level |
+-----------+-----------+-------------+--------------+
|         1 |         0 | 中华国          |            1 |
+-----------+-----------+-------------+--------------+
```

5.5.8 垂直切分的优缺点

垂直切分的优点

数据库的拆分简单明了，拆分规则明确

应用程序模块化清晰明确，整合容易

数据维护方便易行，容易定位

垂直切分的缺点

部分表关联无法在数据库级别完成，需要在程序中完成

对于访问极其频繁且数据量超大的表仍然存在性能瓶颈

切分达到一定程度之后，扩展性会遇到限制

解决跨分片关联的方式

使用 mycat 全局表

冗余部分关键数据

使用 API 的方式获取

6 实战之水平分库

6.1 水平分库介绍

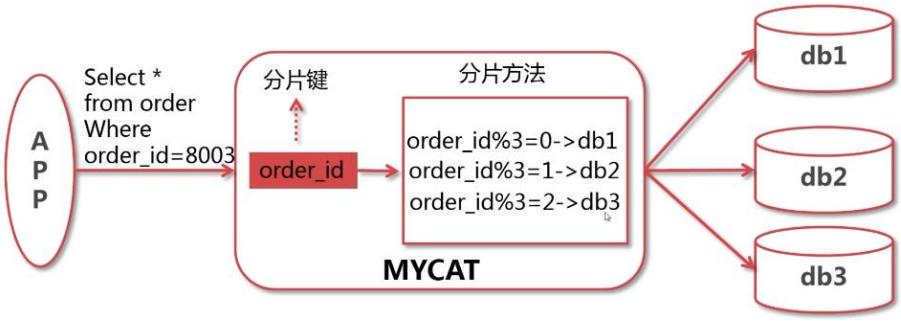
分片原则

能不切分尽量不要切分

选择合适的切分规则和分片键

尽量避免跨分片 join 操作

分片后如何处理查询



水平分片的步骤

1 根据业务状态确定要进行水平切分的表

2 分析业务模型选择分片键及分片算法

3 使用 MyCAT 部署分片集群

4 测试分片集群

5 业务及数据迁移

如何选择分片键

尽可能的比较均匀分布数据到各个节点上

该业务字段是最频繁的或者最重要的查询条件

分析业务模型选择分片键及分片算法

对订单相关表进行水平切分

以 customer_id 做为分片键
采用简单取模分片算法

部署 mycat 分片集群说明

使用 schema.xml 配置逻辑库及逻辑表
使用 rule.xml 配置分片表的分片规则
使用 server.xml 配置访问用户及权限

演示环境说明

主机名	IP	角色	数据库
node1	10.0.0.62	mycat mysql	
node2	10.0.0.63	mysql	orderdb01 orderdb02
node3	10.0.0.64	mysql	orderdb03 orderdb04

6.2 水平切分实践

配置水平分片

0）准备测试数据

node2：

```
[root@node2 ~]# mysql -uroot -p123456 -e "create database orderdb01"
[root@node2 ~]# mysql -uroot -p123456 -e "create database orderdb02"
[root@node2 ~]# mysql -uroot -p123456
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| order_db           |
| orderdb01          |
| orderdb02          |
| performance_schema |
| sys                |
+--------------------+
mysql> use orderdb01;
mysql>
```

node3:

```
[root@node3 ~]# mysql -uroot -p123456 -e "create database orderdb03"
[root@node3 ~]# mysql -uroot -p123456 -e "create database orderdb04"
```

```
# 登录 node2,node3; 分别为 orderdb01、orderdb02、orderdb03、orderdb04 创建如下表：

CREATE TABLE order_master (
    order_id INT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '订单 ID',
    order_sn BIGINT UNSIGNED NOT NULL COMMENT '订单编号
yyyymmddnnnnnnnn',
    customer_id INT UNSIGNED NOT NULL COMMENT '下单人 ID',
    shipping_user VARCHAR (20) NOT NULL COMMENT '收货人姓名',
    province SMALLINT NOT NULL COMMENT '收货人所在省',
    city SMALLINT NOT NULL COMMENT '收货人所在市',
    district SMALLINT NOT NULL COMMENT '收货人所在区',
    address VARCHAR (100) NOT NULL COMMENT '收货人详细地址',
    payment_method TINYINT NOT NULL COMMENT '支付方式:1 现金,2 余额,3 网银,4
支付宝,5 微信',
    order_money DECIMAL (8, 2) NOT NULL COMMENT '订单金额',
    district_money DECIMAL (8, 2) NOT NULL DEFAULT 0.00 COMMENT '优惠金额',
    shipping_money DECIMAL (8, 2) NOT NULL DEFAULT 0.00 COMMENT '运费金额
',
    payment_money DECIMAL (8, 2) NOT NULL DEFAULT 0.00 COMMENT '支付金额
',
    shipping_comp_name VARCHAR (10) COMMENT '快递公司名称',
    shipping_sn VARCHAR (50) COMMENT '快递单号',
    create_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT '下单时间',
    shipping_time datetime COMMENT '发货时间',
    pay_time datetime COMMENT '支付时间',
    receive_time datetime COMMENT '收货时间',
    order_status TINYINT NOT NULL DEFAULT 0 COMMENT '订单状态',
    order_point INT UNSIGNED NOT NULL DEFAULT 0 COMMENT '订单积分',
    invoice_title VARCHAR (100) COMMENT '发票抬头',
    modified_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP COMMENT '最后修改时间',
    PRIMARY KEY pk_orderid (order_id)
) ENGINE = INNODB COMMENT '订单主表';
```

1）修改 schema.xml

```
[root@node1 ~]# vim /usr/local/mycat/conf/schema.xml
```

增加 dataNode

```
<dataNode name="orderdb01" dataHost="mysql0103" database="orderdb01" />
<dataNode name="orderdb02" dataHost="mysql0103" database="orderdb02" />
<dataNode name="orderdb03" dataHost="mysql0104" database="orderdb03" />
<dataNode name="orderdb04" dataHost="mysql0104" database="orderdb04" />
```

table 修改

针对要修改的 order_master 表，进行配置

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
        <schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">
        <table name="order_master" primaryKey="order_id" dataNode="orderdb01,orderdb02,orderdb03,orderdb04" rule="order_master" />
        <table name="order_detail" primaryKey="order_detail_id" dataNode="ordb" />
        <table name="order_cart" primaryKey="cart_id" dataNode="ordb" />
        <table name="order_customer_addr" primaryKey="customer_addr_id" dataNode="ordb" />
```

2）修改 rule.xml

```
[root@node1 ~]# vim /usr/local/mycat/conf/rule.xml
<!DOCTYPE mycat:rule SYSTEM "rule.dtd">
<mycat:rule xmlns:mycat="http://io.mycat/">
        <tableRule name="order_master">
                <rule>
                        <columns>customer_id</columns>
                        <algorithm>mod-long</algorithm>
                </rule>
        </tableRule>
        <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
                <property name="count">4</property>
        </function>
</mycat:rule>
```

3）重启 mycat

```
[root@node1 ~]# mycat restart
```

查看日志

```
[root@node1 ~]# tail -f /usr/local/mycat/logs/wrapper.log
```

4）查看配置效果

创建订单数据：

```
mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_method,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shipping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0001',5636,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17 10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
```

```
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0002',2808,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0003',6977,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0004',7586,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0005',4711,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0006',6235,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0007',4855,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
```

```
values('0008',3131,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17 10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');
```

5）查看逻辑库里的数据，是否进行了数据分片

node2:

```
mysql -uroot -p123456
```

查看 orderdb01 的数据

```
mysql> use orderdb01;
mysql> select customer_id,order_sn,order_id from order_master;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
+-------------+----------+----------+
|        5636 |        1 |        1 |
|        2808 |        2 |        2 |
+-------------+----------+----------+
mysql> select mod(5636,4);
+-------------+
| mod(5636,4) |
+-------------+
|           0 |
+-------------+
```

查看 orderdb02 的数据

```
mysql> use orderdb02;
mysql> select customer_id,order_sn,order_id from order_master;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
+-------------+----------+----------+
|        6977 |        3 |       10 |
+-------------+----------+----------+
mysql> select mod(6977,4);
+-------------+
| mod(6977,4) |
+-------------+
|           1 |
+-------------+
```

node3:

查看 orderdb03 的数据

```
mysql -uroot -p123456
mysql> use orderdb03;
mysql> select customer_id,order_sn,order_id from order_master;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
```

```
+-------------+----------+----------+
|        7586 |        4 |        1 |
+-------------+----------+----------+
mysql> select mod(7586,4);
+-------------+
| mod(7586,4) |
+-------------+
|           2 |
+-------------+
```

查看 orderdb04 的数据

```
mysql> use orderdb04;
mysql> select customer_id,order_sn,order_id from order_master;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
+-------------+----------+----------+
|        4711 |        5 |        1 |
|        6235 |        6 |        2 |
|        4855 |        7 |        3 |
|        3131 |        8 |        4 |
+-------------+----------+----------+
```

登录 mycat 查看数据

```
mysql> select customer_id,order_sn,order_id from order_master;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
+-------------+----------+----------+
|        6977 |        3 |       10 |
|        7586 |        4 |        1 |
|        5636 |        1 |        1 |
|        4711 |        5 |        1 |
|        6235 |        6 |        2 |
|        4855 |        7 |        3 |
|        3131 |        8 |        4 |
|        2808 |        2 |        2 |
+-------------+----------+----------+
```

发现 order_id 有重复的

全局自增 ID

在实现分库分表的情况下，数据库自增主键已无法保证自增主键的全局唯一。为此，MyCat 提供了全局 sequence，并且提供了包含本地配置和数据库配置等多种实现方式。

下面以数据库配置方式实现来讲解

在数据库中建立一张表，存放 sequence 名称(name)，sequence 当前值(current_value)，步长

(increment int 类型每次读取多少个 sequence，假设为 K)等信息；

1）创建记录自增 ID 的数据库

node1:

登录 mysql，创建 mycat 表，用来记录要全局同步的

```
[root@node1 conf]# mysql -uroot -p123456 -P3306
mysql> create database mycat;
```

将 conf 目录下的 dbseq.sql 导入到上面创建的表 mycat

```
[root@node1 conf]# pwd
/usr/local/mycat/conf
[root@node1 conf]# mysql -uroot -p123456 mycat < dbseq.sql
```

登录 MySQL，查看

```
[root@node1 conf]# mysql -uroot -p123456 -P3306
mysql> use mycat
mysql> show tables;
+----------------+
| Tables_in_mycat |
+----------------+
| MYCAT_SEQUENCE    |
+----------------+
mysql> select * from MYCAT_SEQUENCE;
+--------+---------------+-----------+
| name      | current_value | increment |
+--------+---------------+-----------+
| GLOBAL |                     1 |              1 |
+--------+---------------+-----------+
```

2）修改 server.xml

```
<property name="sequnceHandlerType">1</property>
```

1 是数据库的方式

```
<?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>
            <property name="managerPort">9066</property>
            <property name="nonePasswordLogin">0</property>
            <property name="bindIp">0.0.0.0</property>
            <property name="frontWriteQueueSize">2048</property>

            <property name="charset">utf8</property>
            <property name="txIsolation">2</property>
            <property name="processors">8</property>
            <property name="idleTimeout">1800000</property>
            <property name="sqlExecuteTimeout">300</property>
            <property name="useSqlStat">0</property>
            <property name="useGlobleTableCheck">0</property>
            <property name="sequnceHandlerType">1</property>
            <property name="defaultMaxLimit">100</property>
            <property name="maxPacketSize">104857600</property>
        </system>

        <user name="app_imooc" defaultAccount="true">
            <property name="usingDecrypt">1</property>
```

3）修改 schema.xml

增加 mycat 数据库的节点

[root@node1 conf]# vim schema.xml

dataHost 信息

```
<dataHost name="mysql0102" maxCon="1000" minCon="10" balance="3" writeType="0"
dbType="mysql" dbDriver="native"    swit
chType="1">
            <heartbeat>select user()</heartbeat>
            <writeHost host="10.0.0.62" url="10.0.0.62:3306" user="im_mycat"
password="123456"></writeHost>
</dataHost>
```

dataNode 信息

```
<dataNode name="mycat" dataHost="mysql0102" database="mycat" />
```

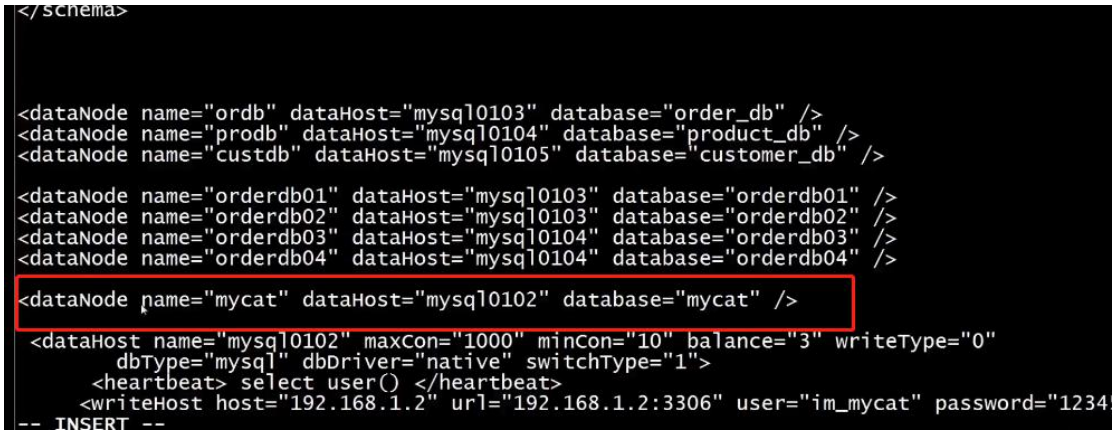


table 信息

```
# 开启全局自增
 <table name="order_master" primaryKey="order_id"
dataNode="orderdb01,orderdb02,orderdb03,orderdb04" rule="order_m
aster"    autoIncrement="true"/>
```

4）查看上面配置的用户在 node1 上的权限，增加 execute 权限

```
[root@node1 ~]# mysql -uroot -p123456
mysql> select user,host from mysql.user;
+-----------+-----------+
| user          | host          |
+-----------+-----------+
| im_mycat    | 10.0.0.%    |
| im_repl      | 10.0.0.%    |
| mysql.sys | localhost |
| root          | localhost |
+-----------+-----------+
mysql> show grants for im_mycat@'10.0.0.%';
```

```
+-------------------------------------------------------------------+
| Grants for im_mycat@10.0.0.%                                      |
+-------------------------------------------------------------------+
| GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'im_mycat'@'10.0.0.%' |
+-------------------------------------------------------------------+

#增加 execute 权限
grant execute on *.* to 'im_mycat'@'10.0.0.%';

# 给表 MYCAT_SEQUENCE 增加要记录的数据
insert into MYCAT_SEQUENCE values ('ORDER_MASTER',1,1);
mysql> select * from MYCAT_SEQUENCE;
+--------------+---------------+-----------+
| name         | current_value | increment |
+--------------+---------------+-----------+
| GLOBAL       |             1 |         1 |
| ORDER_MASTER |             1 |         1 |
+--------------+---------------+-----------+
```

5）sequence_db_conf.properties 相关配置,指定 sequence 相关配置在哪个节点上

```
[root@node1 conf]# vim sequence_db_conf.properties
#sequence stored in datanode
GLOBAL=mycat
ORDER_MASTER=mycat
```

6）测试验证

删除之前的测试数据

```
[root@node1 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> use imooc_db;
mysql> select count(*) from order_master;
+--------+
| COUNT0 |
+--------+
|      8 |
+--------+
mysql> delete from order_master;
mysql> select count(*) from order_master;
+--------+
| COUNT0 |
+--------+
|      0 |
+--------+
#重新导入数据
```

```
insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0001',5636,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0002',2808,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0003',6977,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0004',7586,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0005',4711,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0006',6235,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
```

```
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0007',4855,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0008',3131,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');
```

```
#全局 ID 生效，order_id 恢复正常
mysql> select customer_id,order_sn,order_id from order_master order by order_id;
+-------------+----------+----------+
| customer_id | order_sn | order_id |
+-------------+----------+----------+
|        5636 |        1 |        2 |
|        2808 |        2 |        3 |
|        6977 |        3 |        4 |
|        7586 |        4 |        5 |
|        4711 |        5 |        6 |
|        6235 |        6 |        7 |
|        4855 |        7 |        8 |
|        3131 |        8 |        9 |
+-------------+----------+----------+
```

ER 分片

跨分片查询有问题

```
mysql> select * from order_master a join order_detail b on a.order_id=b.order_id;
ERROR 1064 (HY000): invalid route in sql, multi tables found but datanode has no
intersection    sql:select * from order_master a join order_detail b on a.order_id=b.order_id
```

ER 分片表解决

1）在 orderdb01,02,03,04 上建立 order_detail 表

建表语句如下

```
CREATE TABLE order_detail (
    order_detail_id INT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '自增
主键 ID,订单详情表 ID',
    order_id INT UNSIGNED NOT NULL COMMENT '订单表 ID',
    product_id INT UNSIGNED NOT NULL COMMENT '订单商品 ID',
    product_name VARCHAR (50) NOT NULL COMMENT '商品名称',
    product_cnt INT NOT NULL DEFAULT 1 COMMENT '购买商品数量',
```

```
    product_price DECIMAL (8, 2) NOT NULL COMMENT '购买商品单价',
    average_cost DECIMAL (8, 2) NOT NULL DEFAULT 0.00 COMMENT '平均成本价格',
    weight FLOAT COMMENT '商品重量',
    fee_money DECIMAL (8, 2) NOT NULL DEFAULT 0.00 COMMENT '优惠分摊金额',
    w_id INT UNSIGNED NOT NULL COMMENT '仓库 ID',
    modified_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '最后修改时间',
    PRIMARY KEY pk_orderdetailid (order_detail_id)
) ENGINE = INNODB COMMENT '订单详情表';
```

node2:

```
mysql -uroot -p123456
mysql> use orderdb01;
mysql> #执行上面的建表语句
mysql> use orderdb02;
mysql> #执行上面的建表语句
```

node3:

```
mysql -uroot -p123456
mysql> use orderdb03;
mysql> #执行上面的建表语句
mysql> use orderdb04;
mysql> #执行上面的建表语句
mysql> show tables;
+--------------------+
| Tables_in_orderdb02 |
+--------------------+
| order_detail               |
| order_master               |
+--------------------+
```

2）删除之前的测试数据

```
[root@node1 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> use imooc_db;
mysql> delete from order_master;
Query OK, 8 rows affected (0.02 sec)
```

3）修改 schemal.xml

```
<table name="order_master" primaryKey="order_id"
dataNode="orderdb01,orderdb02,orderdb03,orderdb04" rule="order_m
aster"    autoIncrement="true">
```

```
                    <childTable name="order_detail" primaryKey="order_detail_id"
joinKey="order_id" parentKey="order_id" auto
Increment="true"/>
            </table>
```

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="imooc_db" checkSQLschema="false" sqlMaxLimit="100">

 <table name="order_master" primaryKey="order_id" dataNode="orderdb01,orderdb02,orderdb03,
orderdb04" rule="order_master" autoIncrement="true">
     <childTable name="order_detail" primaryKey="order_detail_id" joinKey="order_id" parer
tKey="order_id" autoIncrement="true" />
</table>

 <table name="order_detail" primaryKey="order_detail_id" dataNode="ordb" />
 <table name="order_cart" primaryKey="cart_id" dataNode="ordb" />
 <table name="order_customer_addr" primaryKey="customer_addr_id" dataNode="ordb" />
 <table name="region_info" primaryKey="region_id" dataNode="ordb,prodb,custdb" type="globa
l" />
<table name="shipping_info" primaryKey="ship_id" dataNode="ordb" />
<table name="warehouse_info" primaryKey="w_id" dataNode="ordb" />
<table name="warehouse_proudct" primaryKey="wp_id" dataNode="ordb" />

<table name="product_brand_infos" primaryKey="brand_id" dataNode="prodb" />
<table name="product_category" primaryKey="category_id" dataNode="prodb" />
<table name="product_comment" primaryKey="comment_id" dataNode="prodb" />
<table name="product_info" primaryKey="product_id" dataNode="prodb" />
-- INSERT --
```

删除 order_detail_id 的定义

```
<table name="order_detail" primaryKey="order_detail_id" dataNode="ordb" />
 <table name="order_customer_addr" primaryKey="customer_addr_id" dataNode="ordb" />
 <table name="region_info" primaryKey="region_id" dataNode="ordb,prodb,custdb" type="
" />
<table name="shipping_info" primaryKey="ship_id" dataNode="ordb" />
<table name="warehouse_info" primaryKey="w_id" dataNode="ordb" />
<table name="warehouse_proudct" primaryKey="wp_id" dataNode="ordb" />

<table name="product_brand_infos" primaryKey="brand_id" dataNode="prodb" />
<table name="product_category" primaryKey="category_id" dataNode="prodb" />
<table name="product_comment" primaryKey="comment_id" dataNode="prodb" />
```

4）sequence_db_conf.properties 相关配置,指定 sequence 相关配置在哪个节点上

```
[root@node1 conf]# vim sequence_db_conf.properties
#sequence stored in datanode
GLOBAL=mycat
ORDER_MASTER=mycat
ORDER_DETAIL=mycat
```

5）给表 MYCAT_SEQUENCE 增加要记录的数据

```
[root@node1 conf]# mysql -uroot -p123456
insert into MYCAT_SEQUENCE values ('ORDER_MASTER',1,1);
mysql> select * from MYCAT_SEQUENCE;
+--------------+---------------+-----------+
| name         | current_value | increment |
+--------------+---------------+-----------+
| GLOBAL       |             1 |         1 |
| ORDER_MASTER |             1 |         1 |
+--------------+---------------+-----------+
```

```
mysql> use mycat;
mysql> insert into MYCAT_SEQUENCE values('ORDER_DETAIL',1,1);
mysql> select * from MYCAT_SEQUENCE;
+--------------+---------------+-----------+
| name         | current_value | increment |
+--------------+---------------+-----------+
| GLOBAL       |             1 |         1 |
| ORDER_DETAIL |             1 |         1 |
| ORDER_MASTER |             9 |         1 |
+--------------+---------------+-----------+

# 将前面测试数据记录 ID 值，置为 1
mysql> update MYCAT_SEQUENCE set current_value=1 where
name='ORDER_MASTER';
mysql> select * from MYCAT_SEQUENCE;
+--------------+---------------+-----------+
| name         | current_value | increment |
+--------------+---------------+-----------+
| GLOBAL       |             1 |         1 |
| ORDER_DETAIL |             1 |         1 |
| ORDER_MASTER |             1 |         1 |
+--------------+---------------+-----------+
```

6）重启 mycat 查看状态

```
[root@node1 conf]# mycat restart
[root@node1 conf]# tail -f ../logs/wrapper.log
STATUS | wrapper    | 2018/06/30 17:39:59 | --> Wrapper Started as Daemon
STATUS | wrapper    | 2018/06/30 17:40:00 | Launching a JVM...
INFO   | jvm 1      | 2018/06/30 17:40:00 | Wrapper (Version 3.2.3)
http://wrapper.tanukisoftware.org
INFO   | jvm 1      | 2018/06/30 17:40:00 |     Copyright 1999-2006 Tanuki Software, Inc.
All Rights Reserved.
INFO   | jvm 1      | 2018/06/30 17:40:00 |
INFO   | jvm 1      | 2018/06/30 17:40:01 | log4j:WARN No appenders could be found for
logger (io.mycat.memory.MyCatMemory).
INFO   | jvm 1      | 2018/06/30 17:40:01 | log4j:WARN Please initialize the log4j system
properly.
INFO   | jvm 1      | 2018/06/30 17:40:01 | log4j:WARN See
http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
INFO   | jvm 1      | 2018/06/30 17:40:01 | MyCAT Server startup successfully. see logs in
logs/mycat.log
```

7）导入测试数据

```
[root@node1 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
```

```
# 清空上面的测试数据
mysql> delete from order_master;
Query OK, 0 rows affected (0.09 sec)

mysql> delete from order_detail;
Query OK, 0 rows affected (0.02 sec)

# 导入测试数据
# 订单主表
insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0001',5636,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0002',2808,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0003',6977,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0004',7586,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0005',4711,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');
```

```
insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0006',6235,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0007',4855,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');

insert into
order_master(order_sn,customer_id,shipping_user,province,city,district,address,payment_met
hod,order_money,district_money,shipping_money,payment_money,shipping_comp_name,shi
pping_sn,shipping_time,pay_time,receive_time,order_status,order_point,invoice_title)
values('0008',3131,'张三',2,3,4,'外滩',1,200,0.00,0.00,0.00,'顺丰','kd_no123','2014-07-17
10:53:15','2014-07-17 10:53:15','2014-07-17 10:53:15',0,200,'fp_no567');



# 订单详情表
insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(2,2,'长裙
2',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(3,2,'长裙
3',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(4,2,'长裙
4',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(5,2,'长裙
5',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');
```

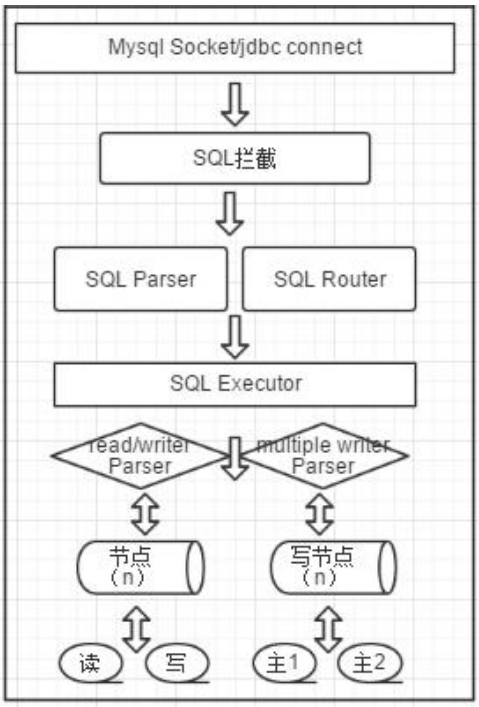
```
insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(6,2,'长裙
6',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(7,2,'长裙
7',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(8,2,'长裙
8',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');

insert into
order_detail(order_id,product_id,product_name,product_cnt,product_price,average_cost,weig
ht,fee_money,w_id,modified_time) values(9,2,'长裙
1',4,50.00,30.00,0.1,0.00,100,'2014-07-17 10:53:15');
```

```
mysql> select count(*) from order_master;
+--------+
| COUNT0 |
+--------+
|      8 |
+--------+
mysql> select count(*) from order_detail;
+--------+
| COUNT0 |
+--------+
|      8 |
+--------+
mysql> select order_id,order_sn,mod(customer_id,4) from order_master;
+----------+----------+--------------------+
| order_id | order_sn | mod(customer_id, 4) |
+----------+----------+--------------------+
|        5 |        4 |                  2 |
|        6 |        5 |                  3 |
|        7 |        6 |                  3 |
|        8 |        7 |                  3 |
|        9 |        8 |                  3 |
|        4 |        3 |                  1 |
|        2 |        1 |                  0 |
```

```
|          3 |          2 |                          0 |
+----------+----------+--------------------+

# node2 orderdb01 查看结果:
mysql> use orderdb01;
mysql> select order_id,order_sn,customer_id from order_master;
+----------+----------+-------------+
| order_id | order_sn | customer_id |
+----------+----------+-------------+
|          2 |          1 |        5636 |
|          3 |          2 |        2808 |
+----------+----------+-------------+
mysql> select order_id from order_detail;
+----------+
| order_id |
+----------+
|          2 |
|          3 |
+----------+
#上面表明，配置的 ER 分片，成功；数据在一个分片上

# node1: 验证连表查询
[root@node1 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> select a.order_id,order_sn ,product_name from order_master a join order_detail b on
a.order_id=b.order_id;
+----------+----------+--------------+
| order_id | order_sn | product_name |
+----------+----------+--------------+
|          5 |          4 | 长裙 5       |
|          6 |          5 | 长裙 6       |
|          7 |          6 | 长裙 7       |
|          8 |          7 | 长裙 8       |
|          9 |          8 | 长裙 1       |
|          4 |          3 | 长裙 4       |
|          2 |          1 | 长裙 2       |
|          3 |          2 | 长裙 3       |
+----------+----------+--------------+
# ok ,测试完毕
```

7 其他功能

7.1 SQL 拦截



1 作用

监控记录数据写入操作

SQL 审计

2 在 server.xml 配置

```
<system>
	<property
name="sqlInterceptor">io.mycat.server.interceptor.impl.StatisticsSqlInterceptor</property>
	<property name="sqlInterceptorType">update,insert,delete</property>
	<property name="sqlInterceptorFile">/tmp/sql.txt</property>
</system>
```

3 实际操作

[root@node1 conf]# vim server.xml

```
<!DOCTYPE mycat:server SYSTEM "server.dtd">
<mycat:server xmlns:mycat="http://io.mycat/">
        <system>
            <property name="serverPort">8066</property>
            <property name="managerPort">9066</property>
            <property name="nonePasswordLogin">0</property>
            <property name="bindIp">0.0.0.0</property>
            <property name="frontWriteQueueSize">2048</property>

            <property name="charset">utf8</property>
            <property name="txIsolation">2</property>
            <property name="processors">8</property>
            <property name="idleTimeout">1800000</property>
            <property name="sqlExecuteTimeout">300</property>
            <property name="useSqlStat">0</property>
            <property name="useGlobleTableCheck">0</property>
            <property name="sequnceHandlerType">1</property>
            <property name="defaultMaxLimit">100</property>
            <property name="maxPacketSize">104857600</property>

            <property name="sqlInterceptor">io.mycat.server.interceptor.impl.StatisticsSql
Interceptor</property>
            <property name="sqlInterceptorType">UPDATE,DELETE,INSERT</property>
            <property name="sqlInterceptorFile">/tmp/sql.txt</property>

"server.xml" 36L, 1565C written
```

重启 mycat

```
[root@node1 ~]# mycat restart
[root@node1 ~]# mysql -uapp_imooc -p123456 -h10.0.0.62 -P8066
mysql> use imooc_db;
mysql> select count(*) from order_master;
+--------+
| COUNT0 |
+--------+
|      8 |
+--------+
```

```
mysql > select * from order_master limit 1\G
*************************** 1. row ***************************
             order_id: 2
             order_sn: 1
          customer_id: 5636
       shipping_user: 张三
             province: 2
                 city: 3
             district: 4
              address: 外滩
      payment_method: 1
         order_money: 200.00
      district_money: 0.00
      shipping_money: 0.00
       payment_money: 0.00
shipping_comp_name: 顺丰
         shipping_sn: kd_no123
         create_time: 2018-06-30 17:43:24
       shipping_time: 2014-07-17 10:53:15
            pay_time: 2014-07-17 10:53:15
```

```
          receive_time: 2014-07-17 10:53:15
          order_status: 0
            order_point: 200
        invoice_title: fp_no567
        modified_time: 2018-06-30 17:43:24

# 修改地址
mysql> update order_master set address="东方明珠" where order_id=2;
```

验证配置

```
[root@node1 conf]# ls /tmp/
hsperfdata_root    sql2018-06-30.txt
[root@node1 conf]# cat /tmp/sql2018-06-30.txt
UPDATE:update order_master set address="东方明珠" where order_id=2
```

7.2 SQL 防火墙

1 白名单和 SQL 黑名单说明：
统一控制哪些用户可以通过哪些主机访问后端数据库
统一屏蔽一些 SQL 语句，要强安全控制

2 在 server.xml 配置

```
<firewall>
      <whitehost>
            <host user="root" host="127.0.0.1"></host>
      </whitehost>
      <blacklist check="true">
            <property name="noneBaseStatementAllow">true</property>
            <property name="deleteWhereNoneCheck">true</property>
      </blacklist>
</firewall>
```

3 实际操作

```
<firewall>
      <whitehost>
            <host user="app_imooc" host="10.0.0.65"></host>
      </whitehost>
      <blacklist check="true">
            <property name="deleteWhereNoneCheck">true</property>
      </blacklist>
</firewall>
```

```
            <property name="processors">8</property>
            <property name="idleTimeout">1800000</property>
            <property name="sqlExecuteTimeout">300</property>
            <property name="useSqlStat">0</property>
            <property name="useGlobleTableCheck">0</property>
            <property name="sequnceHandlerType">1</property>
            <property name="defaultMaxLimit">100</property>
            <property name="maxPacketSize">104857600</property>

            <property name="sqlInterceptor">io.mycat.server.interceptor.impl.StatisticsS
Interceptor</property>
            <property name="sqlInterceptorType">UPDATE,DELETE,INSERT</property>
            <property name="sqlInterceptorFile">/tmp/sql.txt</property>

        </system>

        <firewall>
           <whitehost>
                <host user="app_imooc" host="192.168.1.5"></host>
           </whitehost>

        </firewall>
"server.xml" 45L, 1721C written
```

```
    <firewall>
       <whitehost>
            <host user="app_imooc" host="192.168.1.5"></host>
       </whitehost>
       <blacklist check="true">
         <property name="deleteWhereNoneCheck">true</property>
       </blacklist>
    </firewall>
```

测试：只有 node4 才可以登陆；并且：没有 where 条件的不能删除

8 MyCAT 高可用集群

8.1 高可用介绍

使用 zookeeper 启动 mycat

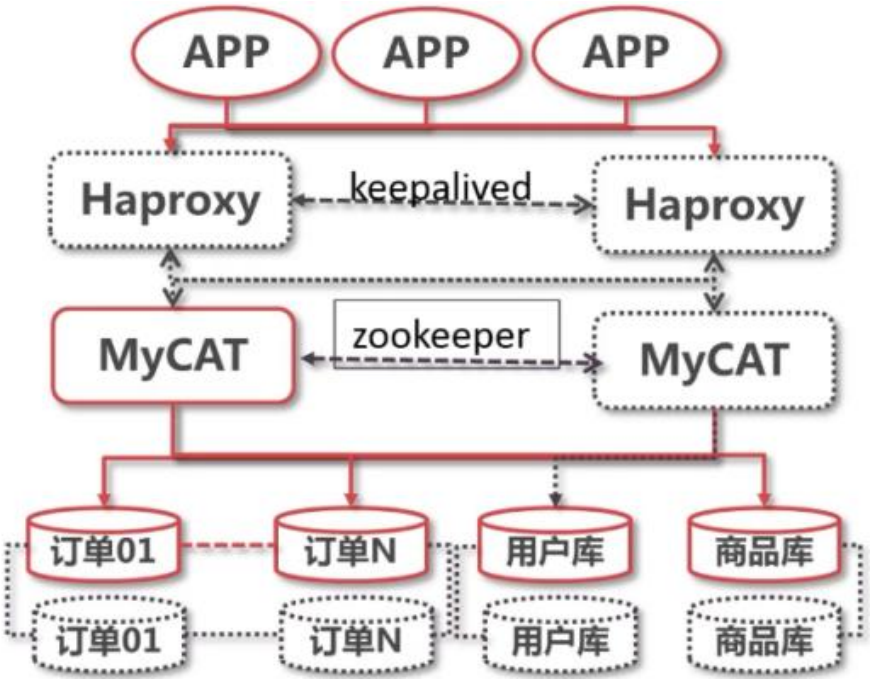
实战 mycat+zk+haproxy+mysql 主从高可用架构

高可用系统的基本要求

系统架构中不存在单点问题

可用最大限度的保障服务的可用性

架构目标



8.2 zookeeper

ZooKeeper 是一个分布式的，开放源码的分布式应用程序协调服务，是 Google 的 Chubby 一个开源的实现，它是集群的管理者，监视着集群中各个节点的状态根据节点提交的反馈进行下一步合理操作。最终，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

操作步骤

1 建立 zookeeper 集群

2 初始化 mycat 配置到 zk 集群

3 配置 mycat 支持 zk 启动

4 启动 mycat

演示环境说明

主机名	IP	角色
node1	10.0.0.62	mycat，mysql，zk
node2	10.0.0.63	mysql，zk
node3	10.0.0.64	mysql，zk
node4	10.0.0.65	mycat，mysql

8.2.1 建立 zookeeper 集群

1）安装 jdk 环境

zk 依赖于 jdk 环境，node2,node3,node4 安装 jdk

```
mkdir -p /home/tools
cd /home/tools/
tar xf jdk-7u80-linux-x64.tar.gz
mkdir -p /usr/local/java
mv jdk1.7.0_80/ /usr/local/java/
chown -R root.root /usr/local/java
cp /etc/profile /etc/profile.ori

vim /etc/profile
# java
export JAVA_HOME=/usr/local/java/jdk1.7.0_80
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
[root@node1 tools]# source /etc/profile

#检查是否安装成功
java -version
```

2）下载安装 zk

node1:

下载 zookeeper

```
[root@node1 ~]# cd /home/tools/
[root@node1 tools]# rz
zookeeper-3.4.12.tar.gz
```

安装 zookeeper

```
[root@node1 tools]# tar zxf zookeeper-3.4.12.tar.gz
[root@node1 tools]# mv zookeeper-3.4.12 /usr/local/zookeeper
```

3）配置

```
[root@node1 tools]# cd /usr/local/zookeeper
[root@node1 zookeeper]# mkdir data
[root@node1 zookeeper]# echo 0 > data/myid
[root@node1 zookeeper]# cd conf/
[root@node1 conf]# ls
configuration.xsl    log4j.properties    zoo_sample.cfg
[root@node1 conf]# cp zoo_sample.cfg zoo.cfg
[root@node1 conf]# vim zoo.cfg
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
```

```
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/usr/local/zookeeper/data
# the port at which the clients will connect
clientPort=2181
server.0=10.0.0.62:2888:3888
server.1=10.0.0.63:2888:3888
server.2=10.0.0.64:2888:3888
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
```

4）同步 zookeeper 目录到其它 2 个节点 node2,node3，修改 myid

```
rsync -avzP /usr/local/zookeeper/ root@10.0.0.63:/usr/local/zookeeper/
rsync -avzP /usr/local/zookeeper/ root@10.0.0.64:/usr/local/zookeeper/
```

修改 myid 的值分别为 1,2

5）启动 zk

node1,node2,node3

```
/usr/local/zookeeper/bin/zkServer.sh start
[root@node1 zookeeper]# netstat -lntup|grep 2181
tcp        0      0 :::2181                     :::*
LISTEN      2822/java
```

6）node4 安装 mycat

```
tar -zxf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz
mv mycat/ /usr/local/
useradd mycat
chown -R mycat.mycat /usr/local/mycat/
```

```
vim /etc/profile
#mycat 2018-05-01
export MYCAT_HOME=/usr/local/mycat
export PATH=$PATH:$MYCAT_HOME/bin

source /etc/profile

vi /etc/hosts
127.0.0.1      localhost localhost.localdomain node4
```

8.2.2 初始化 mycat 配置到 zk 集群

7）将原来配置好的 node1 中 mycat 配置拷贝到/usr/local/mycat/conf/zkconf/下:

```
[root@node1 zookeeper]# cd /usr/local/mycat/conf
[root@node1 conf]# cp schema.xml server.xml rule.xml sequence_db_conf.properties zkconf/
cp：是否覆盖"zkconf/schema.xml"？  y
cp：是否覆盖"zkconf/server.xml"？  y
cp：是否覆盖"zkconf/rule.xml"？  y
cp：是否覆盖"zkconf/sequence_db_conf.properties"？  y
```

8）node1 节点,初始化 zk 配置

```
[root@node1 zookeeper]# sh /usr/local/mycat/bin/init_zk_data.sh
```

9）登录 zk 集群查看验证

```
[root@node1 zookeeper]# sh /usr/local/zookeeper/bin/zkCli.sh
[zk: localhost:2181(CONNECTED) 0] ls /mycat/mycat-cluster-1
[schema, cache, sequences, command, migrate, lock, line, server, bindata, ruledata, rules]
[zk: localhost:2181(CONNECTED) 1] ls /mycat/mycat-cluster-1/schema
[schema, dataHost, dataNode]
[zk: localhost:2181(CONNECTED) 3] get /mycat/mycat-cluster-1/schema/dataHost
[{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0102","writeType":0,"switchTyp
e":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.62","url":"10.0.0.62:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0103","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.63","url":"10.0.0.63:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0104","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.64","url":"10.0.0.64:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0105","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.65","url":"10.0.0.65:3306","password":"123456","user":
"im_mycat"}]}]
```

```
cZxid = 0x300000047
ctime = Tue Jul 03 10:54:31 CST 2018
mZxid = 0x300000048
mtime = Tue Jul 03 10:54:31 CST 2018
pZxid = 0x300000047
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 997
numChildren = 0
```

node2 查看

```
[root@node2 ~]#    sh /usr/local/zookeeper/bin/zkCli.sh
[zk: localhost:2181(CONNECTED) 0] ls /mycat/mycat-cluster-1/schema
[schema, dataHost, dataNode]
[zk: localhost:2181(CONNECTED) 1] get /mycat/mycat-cluster-1/schema/dataHost
[{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0102","writeType":0,"switchTyp
e":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.62","url":"10.0.0.62:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0103","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.63","url":"10.0.0.63:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0104","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.64","url":"10.0.0.64:3306","password":"123456","user":
"im_mycat"}]},{"balance":3,"maxCon":1000,"minCon":10,"name":"mysql0105","writeType":
0,"switchType":1,"dbType":"mysql","dbDriver":"native","heartbeat":"select
user()","writeHost":[{"host":"10.0.0.65","url":"10.0.0.65:3306","password":"123456","user":
"im_mycat"}]}]
cZxid = 0x300000047
ctime = Tue Jul 03 10:54:31 CST 2018
mZxid = 0x300000048
mtime = Tue Jul 03 10:54:31 CST 2018
pZxid = 0x300000047
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 997
numChildren = 0
```

8.2.3 配置 mycat 支持 zk 启动

10）配置 zk，启动 mycat

配置 myid.properties， node1,node4 都需要修改

```
[root@node1 conf]# vim /usr/local/mycat/conf/myid.properties
loadZk=true
zkURL=10.0.0.62:2181,10.0.0.63:2181,10.0.0.64:2181
clusterId=mycat-cluster-1
myid=mycat_01
clusterSize=2
clusterNodes=mycat_01,mycat_04
#server   booster   ;   booster install on db same server,will reset all minCon to 2
type=server
boosterDataHosts=dataHost1
```

```
[root@node4 conf]# vim /usr/local/mycat/conf/myid.properties
loadZk=true
zkURL=10.0.0.62:2181,10.0.0.63:2181,10.0.0.64:2181
clusterId=mycat-cluster-1
myid=mycat_04
clusterSize=2
clusterNodes=mycat_01,mycat_04
#server   booster   ;   booster install on db same server,will reset all minCon to 2
type=server
boosterDataHosts=dataHost1
```

8.2.4 启动 mycat，验证

11）重启 node1,node4 mycat，查看验证配置

```
mycat restart
```

node4 查看配置

```
[root@node4 mycat]# tail -f /usr/local/mycat/conf/schema.xml
```

查看配置信息，发现 node4 和 node1 一样

8.3 haproxy&keepalived

操作步骤

1 安装 haproxy

2 使用 keepalived 监控 haproxy

3 配置 haproxy 监控 mycat

4 配置应用通过 VIP 访问 haproxy

演示环境说明

主机名	IP	角色
node1	10.0.0.62	mycat，mysql，zk，haproxy，keepalived
node2	10.0.0.63	mysql，zk
node3	10.0.0.64	mysql，zk
node4	10.0.0.65	mycat，mysql，haproxy，keepalived

8.3.1 安装配置 haproxy

1 安装

node1

```
yum install haproxy -y
ls /etc/haproxy
```

node1:

```
[root@node1 conf]# cd /etc/haproxy/
[root@node1 haproxy]# vim haproxy.cfg
global
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
```

```
        timeout client          1m
        timeout server          1m
        timeout http-keep-alive 10s
        timeout check           10s
        maxconn                 3000

listen admin_status
        bind 0.0.0.0:48800    #VIP
        stats uri /admin-status
        stats auth admin:admin

listen allmycat_service
        bind 0.0.0.0:8096
        mode tcp
        option tcplog
        option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
        balance roundrobin
        server mycat_01 10.0.0.62:8066 check port 48700 inter 5s rise 2 fall 3
        server mycat_04 10.0.0.65:8066 check port 48700 inter 5s rise 2 fall 3

listen allmycat_admin
        bind 0.0.0.0:8097
        mode tcp
        option tcplog
        option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
        balance roundrobin
        server mycat_01 10.0.0.62:9066 check port 48700 inter 5s rise 2 fall 3
        server mycat_04 10.0.0.65:9066 check port 48700 inter 5s rise 2 fall 3
```

2 增加 MyCat 存活状态检测服务配置

```
[root@node1 haproxy]# yum install xinetd -y
```

```
[root@node1 haproxy]# vim /etc/services
#在末尾增加如下
mycatchk        48700/tcp                       # mycatchk
```

增加 MyCat 存活状态检测服务配置

```
[root@node1 haproxy]# vim /etc/xinetd.d/mycatchk
service mycatchk
{
flags = REUSE
socket_type = stream
port = 48700
wait = no
```

```
user = root
server = /usr/local/bin/mycat_status
log_on_failure +=USERID
disable = no
}
```

添加 /usr/local/bin/mycat_status 服务脚本

```
[root@node1 haproxy]# vim /usr/local/bin/mycat_status
#!/bin/bash
#/usr/local/bin/mycat_status.sh
# This script checks if a mycat server is healthy running on localhost. It will
# return:
#
# "HTTP/1.x 200 OK\r" (if mycat is running smoothly)
#
# "HTTP/1.x 503 Internal Server Error\r" (else)
mycat=`/usr/local/mycat/bin/mycat status |grep 'not running'| wc -l`
if [ "$mycat" = "0" ];
then
        /bin/echo -en "HTTP/1.1 200 OK\r\n"
        /bin/echo -en "Content-Type: text/plain\r\n"
        /bin/echo -en "Connection: close\r\n"
        /bin/echo -en "Content-Length: 40\r\n"
        /bin/echo -en "\r\n"
        /bin/echo -en "MyCAT    Cluster Node is synced.\r\n"
        exit 0
else
        /bin/echo -en "HTTP/1.1 503 Service Unavailable\r\n"
        /bin/echo -en "Content-Type: text/plain\r\n"
        /bin/echo -en "Connection: close\r\n"
        /bin/echo -en "Content-Length: 44\r\n"
        /bin/echo -en "\r\n"
        /bin/echo -en "MyCAT Cluster Node is not synced.\r\n"
        exit 1
fi
```

```
[root@node1 haproxy]# chmod a+x /usr/local/bin/mycat_status
```

3 重启 xinetd 检查

```
[root@node1 haproxy]# service xinetd restart
停止 xinetd：                                              [确定]
正在启动 xinetd：                                          [确定]
[root@node1 haproxy]# netstat -lntup | grep 48700
tcp        0      0 :::48700                :::*                    LISTEN 6376/xinetd
```

4 eth0 绑定 VIP

```
[root@node1 haproxy]# ifconfig eth0:1 10.0.0.10/24
[root@node1 haproxy]# ifconfig
```

5 启动 haproxy，检测

```
[root@node4 haproxy]# haproxy -f /etc/haproxy/haproxy.cfg
```

ps -ef 查看

```
[root@node1 haproxy]# ps -ef|grep haproxy
haproxy     89180          1    0 16:21 ?              00:00:00 haproxy -f
/etc/haproxy/haproxy.cfg
```

6 node4 操作同上，但是不用增加 VIP

7 node3 测试

```
[root@node3 ~]# mysql -uapp_imooc -p123456 -h10.0.0.10 -P8096
#提示：10.0.0.10 是 VIP
mysql> show databases;
+----------+
| DATABASE |
+----------+
| imooc_db |
+----------+
mysql> use imooc_db;
mysql> select count(*) from order_master;
+--------+
| COUNT0 |
+--------+
|          8 |
+--------+
```

8.3.2 keepalived

node1:

1 安装 keepalived

```
[root@node1 haproxy]# yum install keepalived -y
```

2 配置 keepalived

```
[root@node1 haproxy]# cd /etc/keepalived/
[root@node1 keepalived]# vim keepalived.conf
! Configuration Fileforkeepalived
vrrp_script chk_http_port {
        script"/etc/keepalived/check_haproxy.sh"
        interval 2
        254
```

```
        weight 2
}
vrrp_instance VI_1 {
        state MASTER
        interface eth0
        virtual_router_id 51
        priority 150
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1111
        }
        track_script {
                chk_http_port
        }
        virtual_ipaddress {
                10.0.0.10 dev eth0 scope global
        }
}
```

3 检查 haproxy 脚本

```
[root@node1 keepalived]# vim check_haproxy.sh
#!/bin/bash
STARTHAPROXY="/usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg"
STOPKEEPALIVED="/etc/init.d/keepalived stop"
#STOPKEEPALIVED="/usr/bin/systemctl stop keepalived"
LOGFILE="/var/log/keepalived-haproxy-state.log"
echo "[check_haproxy status]" >> $LOGFILE
A=`ps -C haproxy --no-header |wc -l`
echo "[check_haproxy status]" >> $LOGFILE
date >> $LOGFILE
if [ $A -eq 0 ];then
    echo $STARTHAPROXY >> $LOGFILE
    $STARTHAPROXY >> $LOGFILE 2>&1
    sleep 5
fi
if [ `ps -C haproxy --no-header |wc -l` -eq 0 ];then
    exit 0
else
    exit 1
fi
```

```
[root@node1 keepalived]# chmod a+x check_haproxy.sh
```

4 启动 keepalived

```
[root@node1 keepalived]# /etc/init.d/keepalived start
正在启动 keepalived：                                                    [确定]
[root@node1 keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 00:0c:29:9e:49:84 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.62/24 brd 10.0.0.255 scope global eth0
    inet 10.0.0.10/32 brd 10.0.0.255 scope global secondary eth0:1
    inet6 fe80::20c:29ff:fe9e:4984/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 00:0c:29:9e:49:8e brd ff:ff:ff:ff:ff:ff
```

node4:

```
yum install keepalived -y
[root@node4 haproxy]# cd /etc/keepalived/
[root@node4 keepalived]# vim keepalived.conf
! Configuration Fileforkeepalived
vrrp_script chk_http_port {
        script"/etc/keepalived/check_haproxy.sh"
        interval 2
        254
        weight 2
}
vrrp_instance VI_1 {
        state BACKUP
        interface eth0
        virtual_router_id 51
        priority 120
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1111
        }
        track_script {
                chk_http_port
        }
        virtual_ipaddress {
```

```
                    10.0.0.10 dev eth0 scope global
            }
}
```

```
vi check_haproxy.sh
#!/bin/bash
STARTHAPROXY="/usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg"
STOPKEEPALIVED="/etc/init.d/keepalived stop"
#STOPKEEPALIVED="/usr/bin/systemctl stop keepalived"
LOGFILE="/var/log/keepalived-haproxy-state.log"
echo "[check_haproxy status]" >> $LOGFILE
A=`ps -C haproxy --no-header |wc -l`
echo "[check_haproxy status]" >> $LOGFILE
date >> $LOGFILE
if [ $A -eq 0 ];then
    echo $STARTHAPROXY >> $LOGFILE
    $STARTHAPROXY >> $LOGFILE 2>&1
    sleep 5
fi
if [ `ps -C haproxy --no-header |wc -l` -eq 0 ];then
    exit 0
else
    exit 1
fi
[root@node1 keepalived]# chmod a+x check_haproxy.sh
```

```
[root@node4 keepalived]# /etc/init.d/keepalived start
正在启动 keepalived：                                    [确定]
```

8.3.3 模拟故障

node1 模拟故障

```
[root@node1 keepalived]# /etc/init.d/keepalived stop
停止 keepalived：                                        [确定]
[root@node1 keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:9e:49:84 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.62/24 brd 10.0.0.255 scope global eth0
    inet6 fe80::20c:29ff:fe9e:4984/64 scope link
```

valid_lft forever preferred_lft forever

3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

link/ether 00:0c:29:9e:49:8e brd ff:ff:ff:ff:ff:ff

node4:

```
[root@node4 keepalived]# ip addr
```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 00:0c:29:5d:65:61 brd ff:ff:ff:ff:ff:ff

inet 10.0.0.65/24 brd 10.0.0.255 scope global eth0

inet 10.0.0.10/32 brd 10.0.0.255 scope global eth0

inet6 fe80::20c:29ff:fe5d:6561/64 scope link

valid_lft forever preferred_lft forever

3: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

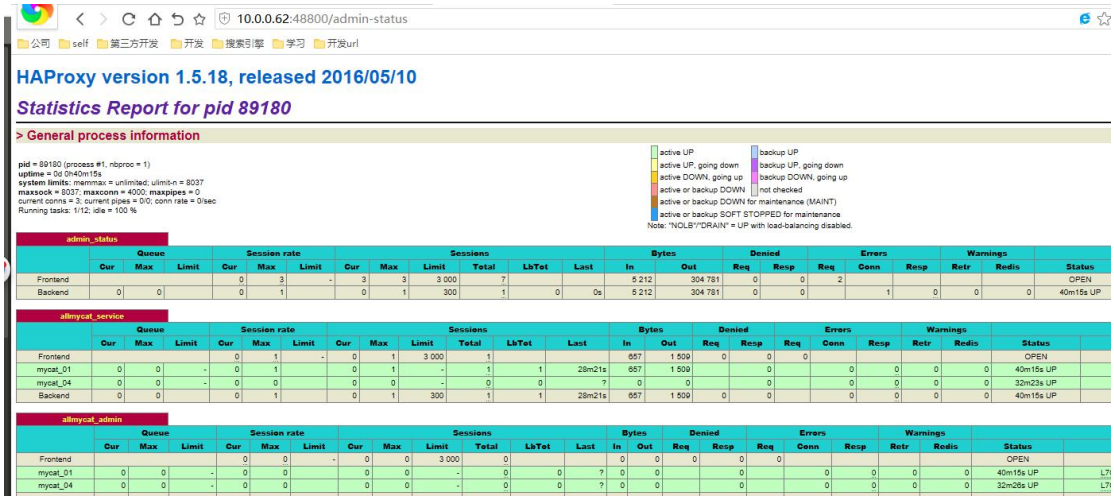
link/ether 00:0c:29:5d:65:6b brd ff:ff:ff:ff:ff:ff

node1 节点恢复

```
[root@node1 keepalived]# /etc/init.d/keepalived start
```

补充

haproxy 页面



8.5 mycat 读写分离

一主一从用 mycat，一主多从 MHA 等

操作步骤

1 配置各个节点的 mysql 主从复制

2 配置 mycat 对后端 DB 进行读写分离
3 滚动应用 mycat 配置
演示环境说明

主机名	IP	角色
node1	10.0.0.62	mycat，mysql，zk，haproxy，keepalived
node2	10.0.0.63	mysql，zk
node3	10.0.0.64	mysql，zk
node4	10.0.0.65	mycat，mysql，haproxy，keepalived
node5	10.0.0.66	mysql

8.5.1 主从复制

这里配置 node4 节点上的数据 customer_db orderdb03 数据库进行主从复制演示
node4:

```
[root@node4 ~]# mkdir /server/backup/ -p
[root@node4 backup]# cd /server/backup/
[root@node4 backup]# mysqldump --master-data=2 --single-transaction -uroot -p --databases
customer_db orderdb03 > node4.sql
[root@node4 backup]# scp node4.sql root@10.0.0.66:/server/backup
[root@node4 backup]# mysql -uroot -p123456
mysql> use mysql;
mysql> select user,host from user;
+-----------+-----------+
| user          | host          |
+-----------+-----------+
| im_mycat    | 10.0.0.%    |
| mysql.sys | localhost |
| root          | localhost |
+-----------+-----------+
mysql> create user im_repl@'10.0.0.%' identified by '123456';
mysql> grant replication slave on *.* to im_repl@'10.0.0.%';
```

node5:

```
[root@node5 backup]# mysql -uroot -p < node4.sql
mysql> show databases;
mysql> show databases;
+--------------------+
| Database                  |
+--------------------+
| information_schema |
| customer_db              |
| mysql                       |
```

```
| orderdb03                      |
| performance_schema |
| sys                            |
+--------------------+

mysql>change master to
master_host='10.0.0.65',master_user='im_repl',master_password='123456',master_port=3306,
master_log_file='mysql-bin.000001',master_log_pos=35040;
mysql> start slave;
mysql> show slave status\G
# 创建 mycat 管理用户
mysql>create user im_mycat@'10.0.0.%' identified by '123456';
mysql> grant select,update,insert,delete,execute on *.* to im_repl@'10.0.0.%';
```

8.5.2 schema.xml

node1

```
[root@node1 keepalived]# cd /usr/local/mycat/conf/zkconf/
[root@node1 zkconf]# vim schema.xml
```

readHost 标签

增加一个 read，同时，node4 挂掉，node5 可以继续接管 write；mycat 一主一从才可以。一主多从，不要这样配置

```
<dataHost name="mysql0105" maxCon="1000" minCon="10" balance="3" writeType="0"
          dbType="mysql" dbDriver="native" switchType="1">
         <heartbeat> select user() </heartbeat>
         <writeHost host="192.168.1.5" url="192.168.1.5:3306" user="im_mycat" password="12
3456" >
          <readHost host="192.168.1.6" url="192.168.1.6:3306" user="im_mycat" password="1234
56" />
         </writeHost>
         <writeHost host="192.168.1.6" url="192.168.1.6:3306" user="im_mycat" password="123
456" />
"schema.xml" [dos] 76L, 3901C written
```

初始化配置，数据通过 zk，分发到 node4

```
/usr/local/mycat/bin/init_zk_data.sh
```

node1 节点，mycat/conf 目录下检查 schemal.xml 的配置

```
    <dataHost balance="3" maxCon="1000" minCon="10" name="mysql0105" writeType="0" switchT
ype="1" dbType="mysql" dbDriver="native">
        <heartbeat> select user() </heartbeat>
        <writeHost host="192.168.1.5" url="192.168.1.5:3306" password="123456" user="im_my
cat">
            <readHost host="192.168.1.6" url="192.168.1.6:3306" password="123456" user="im
_mycat"/>
        </writeHost>
        <writeHost host="192.168.1.6" url="192.168.1.6:3306" password="123456" user="im_my
cat"/>
    </dataHost>
```

node4 同样检查

重启 node1,node4 节点 mycat

```
mycat restart
```

9 MyCAT 管理及监控

9.1 管理命令行

参考：https://www.abcdocker.com/abcdocker/72

通过管理端口管理 mycat

管理端口由 server.xml 配置，默认 9066

管理用户由 server.xml 配置

mysql -u -p -P9066 -hmycat-host

演示：

```
[root@node1 ~]# mysql -uapp_imooc -p -P9066 -h127.0.0.1
#  查看 mycat 命令帮助
mysql> show @@help;

#更新配置文件，推荐用 mycat 轮询重启。配置两台 Mycat
mysql> reload @@config;

#  查看逻辑库
mysql> show @@databases;
+----------+
| DATABASE |
+----------+
| imooc_db |
+----------+

#  数据节点
mysql> show @@datanode;
```

NAME	DATHOST	INDEX	TYPE	ACTIVE	IDLE	SIZE	EXECUTE	TOTAL_TIME	MAX_TIME	MAX_SQL	RECOVERY_TIME
custdb	mysql0105/customer_db	0	mysql	0	10	1000	892	0	0	0	-1
mycat	mysql0102/mycat	0	mysql	0	8	1000	893	0	0	0	-1
ordb	mysql0103/order_db	0	mysql	0	8	1000	890	0	0	0	-1
orderdb01	mysql0103/orderdb01	0	mysql	0	0	1000	4	0	0	0	-1
orderdb02	mysql0103/orderdb02	0	mysql	0	2	1000	4	0	0	0	-1
orderdb03	mysql0104/orderdb03	0	mysql	0	0	1000	4	0	0	0	-1
orderdb04	mysql0104/orderdb04	0	mysql	0	2	1000	4	0	0	0	-1
prodb	mysql0104/product_db	0	mysql	0	8	1000	890	0	0	0	-1

```
#  查看单个节点
mysql> show @@datanode where schema=imooc_db\G

#  心跳信息
mysql> show @@heartbeat;
```

NAME	TYPE	HOST	PORT	RS_CODE	RETRY	STATUS	TIMEOUT	EXECUTE_TIME	LAST_ACTIVE_TIME	STOP
10.0.0.65	mysql	10.0.0.65	3306	1	0	idle	0	1,1,1	2018-07-03 18:07:49	false
10.0.0.63	mysql	10.0.0.63	3306	1	0	idle	0	1,1,1	2018-07-03 18:07:49	false
10.0.0.64	mysql	10.0.0.64	3306	1	0	idle	0	1,1,1	2018-07-03 18:07:49	false
10.0.0.62	mysql	10.0.0.62	3306	1	0	idle	0	0,0,0	2018-07-03 18:07:49	false

mycat 前端连接信息
mysql> show @@connection;
*************************** 1. row ***************************
PROCESSOR: Processor0
ID: 2
HOST: 127.0.0.1
PORT: 9066
LOCAL_PORT: 36379
USER: app_imooc
SCHEMA: NULL
CHARSET: utf8:46
NET_IN: 393
NET_OUT: 6889
ALIVE_TIME(S): 534
RECV_BUFFER: 4096
SEND_QUEUE: 0
txlevel:
autocommit:

杀掉连接
mysql> kill @@connection 2;

#mycat 后端连接状态
mysql> show @@backend;

缓存
mysql> show @@cache;

数据节点对应 mysql 信息
mysql> show @@datasource;

DATANODE	NAME	TYPE	HOST	PORT	W/R	ACTIVE	IDLE	SIZE	EXECUTE	READ_LOAD	WRITE_LOAD
prodb	10.0.0.64	mysql	10.0.0.64	3306	W	0	10	1000	938	2	0
ordb	10.0.0.63	mysql	10.0.0.63	3306	W	0	10	1000	938	2	0
orderdb03	10.0.0.64	mysql	10.0.0.64	3306	W	0	10	1000	938	2	0
orderdb02	10.0.0.63	mysql	10.0.0.63	3306	W	0	10	1000	938	2	0
orderdb01	10.0.0.63	mysql	10.0.0.63	3306	W	0	10	1000	938	2	0
mycat	10.0.0.62	mysql	10.0.0.62	3306	W	0	8	1000	933	0	0

```
| orderdb04 | 10.0.0.64 | mysql | 10.0.0.64 | 3306 | W       |          0 |     10 | 1000 |        938 |               2 |               0 |
| custdb    | 10.0.0.65 | mysql | 10.0.0.65 | 3306 | W       |          0 |     10 | 1000 |        932 |               0 |               0 |
+-----------+-----------+-------+-----------+------+------+---------+------+------+---------+-----------+-----------+
```

9.2 web 管理

通过 mycat-web 监控 mycat
操作步骤：
1 下载 mycat-web 并解压
2 安装 jdk-1.7 运行环境
3 安装 zookeeper
4 启动 mycat-web

node3:
1 下载 mycat-web 并解压

```
[root@node3 ~]# cd /home/tools/
[root@node3 tools]# wget
http://dl.mycat.io/mycat-web-1.0/Mycat-web-1.0-SNAPSHOT-20170102153329-linux.tar.gz
[root@node3 tools]# tar zxf Mycat-web-1.0-SNAPSHOT-20170102153329-linux.tar.gz
[root@node3 tools]# mv mycat-web /usr/local/
```

2 配置 web-mycat

```
[root@node4 WEB-INF]# pwd
/usr/local/mycat-web/mycat-web/WEB-INF/classes
[root@node3 classes]# vim mycat.properties
```

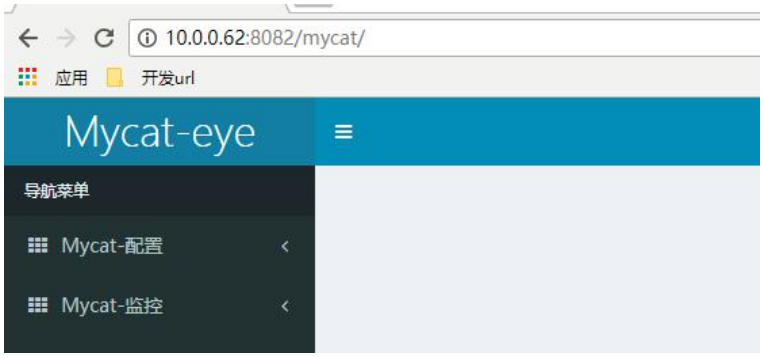
```
#
#Mon Jan 16 15:37:36 CST 2012
show.period=3000000
zookeeper=10.0.0.62:2181

mycat_warn_mail=[{"cc"\:"sohudo@mycat.io","index"\:1,"mangerPort"\:"465","
.com","to"\:"9183838@qq.com"}]
##sql\u4E0A\u7EBF\u76F8\u5173\u914D\u7F6E
sqlonline.server=192.168.80.128
sqlonline.user=root
sqlonline.passwd=123456
~
```

启动：

```
[root@node3 classes]# sh /usr/local/mycat-web/start.sh
#新开一个窗口查看
[root@node1 mycat-web]# netstat -lntup|grep 8082
tcp              0          0 :::8082                                 :::*
LISTEN          51169/java
```

2 web 演示

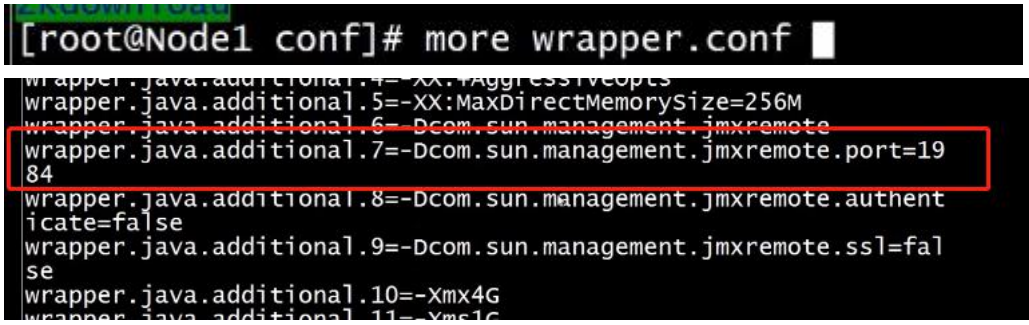
http://10.0.0.62:8082/mycat/



添加 mycat 服务

mycat -vm 配置

查看端口号



mysql 管理

schemal.xml 中有配置,将里面的信息配置过来

10 MyCAT 集群优化

10.1 linux 系统参数优化

编辑/etc/sysctl.conf 优化内核相关参数

```
net.ipv4.tcp_max_syn_backlog = 655350000
net.core.netdev_max_backlog = 327680000
net.core.somaxconn = 327680
```

```
net.core.wmem_default = 838860800
net.core.wmem_max = 167772160
net.core.rmem_default = 838860800
net.core.rmem_max = 167772160

net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_orphans = 3276800
net.ipv4.tcp_keepalive_time = 120

net.ipv4.tcp_max_tw_buckets = 180
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_fin_timeout = 10

net.ipv4.ip_local_port_range = 1024 65535
vm.swappiness = 10
net.nf_conntrack_max = 6553500
net.netfilter.nf_conntrack_max = 6553500
net.netfilter.nf_conntrack_tcp_timeout_established = 300
```

编辑/etc/security/limits.conf 修改资源限制

```
# End of file
* soft nofile 65535
* hard nofile 65535
```

10.2 mycat 集群优化

jvm 参数优化

conf/wrapper.conf

```
wrapper.java.additional.5=-XX:MaxDrrectMemorySize=4G
```

对于 mycat 独立服务器而言占系统内存的一半

mycat 系统参数优化

frontWriteQueueSize	指定前端写队列大小	2048
processors	系统可用线程的数量	根据 CPU 压力可以使 CPU*4
processorBufferPool	指定所使用的 ByteBuffer 池的总字节容量，单位为字节	2097152B
processorBufferChunk	指定所使用的单个 ByteBuffer 的容量，单位为字节	4096B
processorExecutor	每个 processor 的线程池大小	可以为 16-64

日志级别的优化

```
<asyncRoot level=”info” includeLocation=”true”>
```

10.3 mysql 优化

```
[client]
port = 3306
socket = /data/mysql_data/mysql.sock
default-character-set = utf8mb4

[mysqld]
# Skip #
skip_name_resolve               = 1
skip-external-locking    = 1
# General #
user = mysql
default_storage_engine = InnoDB
character-set-server = utf8
socket = /data/mysql_data/mysql.sock
pid_file = /data/mysql_data/mysqld.pid
basedir = /application/mysql
port = 3306
bind-address = 0.0.0.0
log-warnings = 2
explicit_defaults_for_timestamp = off
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES

# Myisam #
key_buffer_size = 32M
myisam_recover = FORCE,BACKUP

# SAFETY #
max_allowed_packet = 16M
max_connetc_errors = 1000000
sysdate_is_now = 1
#innode = FORCE
#innodb_strict_mode = 1

# Replice #
server-id = 1
relay_log = /data/sql_log/relay_log/mysqld-relay-bin
# rpl_semi_sync_master_enabled=1
# rpl_semi_sync_master_timeout=200    # 0.2 second
gtid_mode = on
```

```
enforce-gtid-consistency
log-slave-updates

# Data storage #
datadir = /data/mysql_data
tmpdir = /data/mysql_tmp

# Binary logging #
log_bin = /data/sql_log/bin_log/mysql-bin
max_binlog_size = 1000M
binlog_format = row
expire_logs_days = 7
sync_binlog = 1

# Caches and limits #
tmp_table_size = 32M
max_heap_table_size = 32M
query_cache_type = 0
query_cache_size = 0
max_connections = 4000
thread_cache_size = 2048
open_files_limit = 65535
table_definition_cache = 4096
table_open_cache = 4096
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 2M
thread_concurrency = 24
join_buffer_size = 1M
thread_stack = 512k
max_length_for_sort_data = 16k

# innodb #
innodb_flush_method = O_DIRECT
innodb_log_buffer_size = 16M
innodb_flush_log_at_trx_commit = 2
innodb_file_per_table = 1
innodb_buffer_pool_size = 192G
innodb_buffer_pool_instances = 8
innodb_stats_on_metadata = off
innodb_open_files = 8192
innodb_read_io_threads = 8
innodb_write_io_threads = 16
innodb_thread_concurrency = 0
```

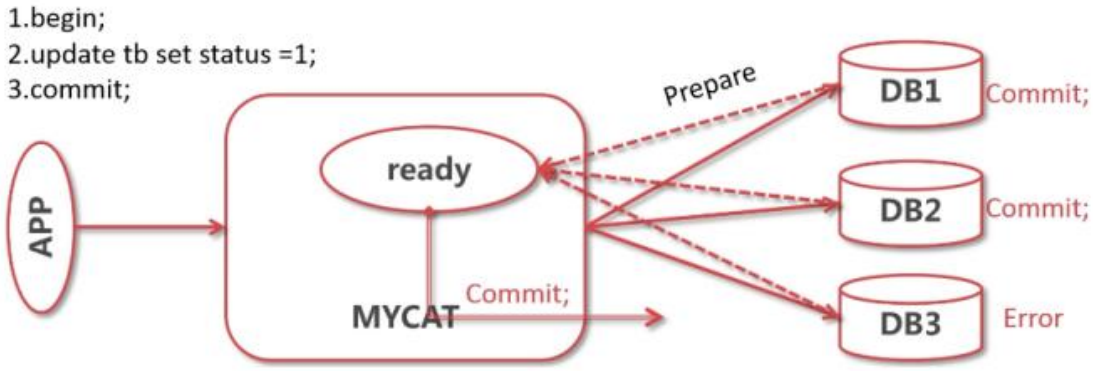
```
innodb_lock_wait_timeout = 60
innodb_old_blocks_time = 1000
innodb_use_native_aio = 1
innodb_purge_threads = 1
innodb_change_buffering = all
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_data_file_path = ibdata1:1024M:autoextend

# logging #
log_error = /data/sql_log/mysql-error.log
log_queries_not_using_indexes = 1
#slow_query_log = 1
slow_query_log_file = /data/sql_log/slowlog.log

# timeout #
interactive_timeout = 30
wait_timeout = 30

[mysqldump]
quick
max_allowed_packet = 16M

[mysql]
no-auto-rehash
# remove the next comment character if you are not familiar with SQL
# safe-updates
```

11 MyCAT 的限制

不支持的 SQL 语句

- create table like xxx/ create table select xxx
- 跨库多表关联查询，子查询
- select for update/ select lock in share mode
- select into outfile/inot var_name
- 多表 update 或是 update 分片键
- 跨分片 update/delete [order by] limit

mycat 对事务的有限支持



mycat 只支持弱分布式事务
如事务 commit 后某节点失效则无法保证事务的一致性

mycat 不使用的场景
使用到不支持的 SQL 的场景
需要跨分片关联查询的场景
需要保证跨分片事务强一致性的场景