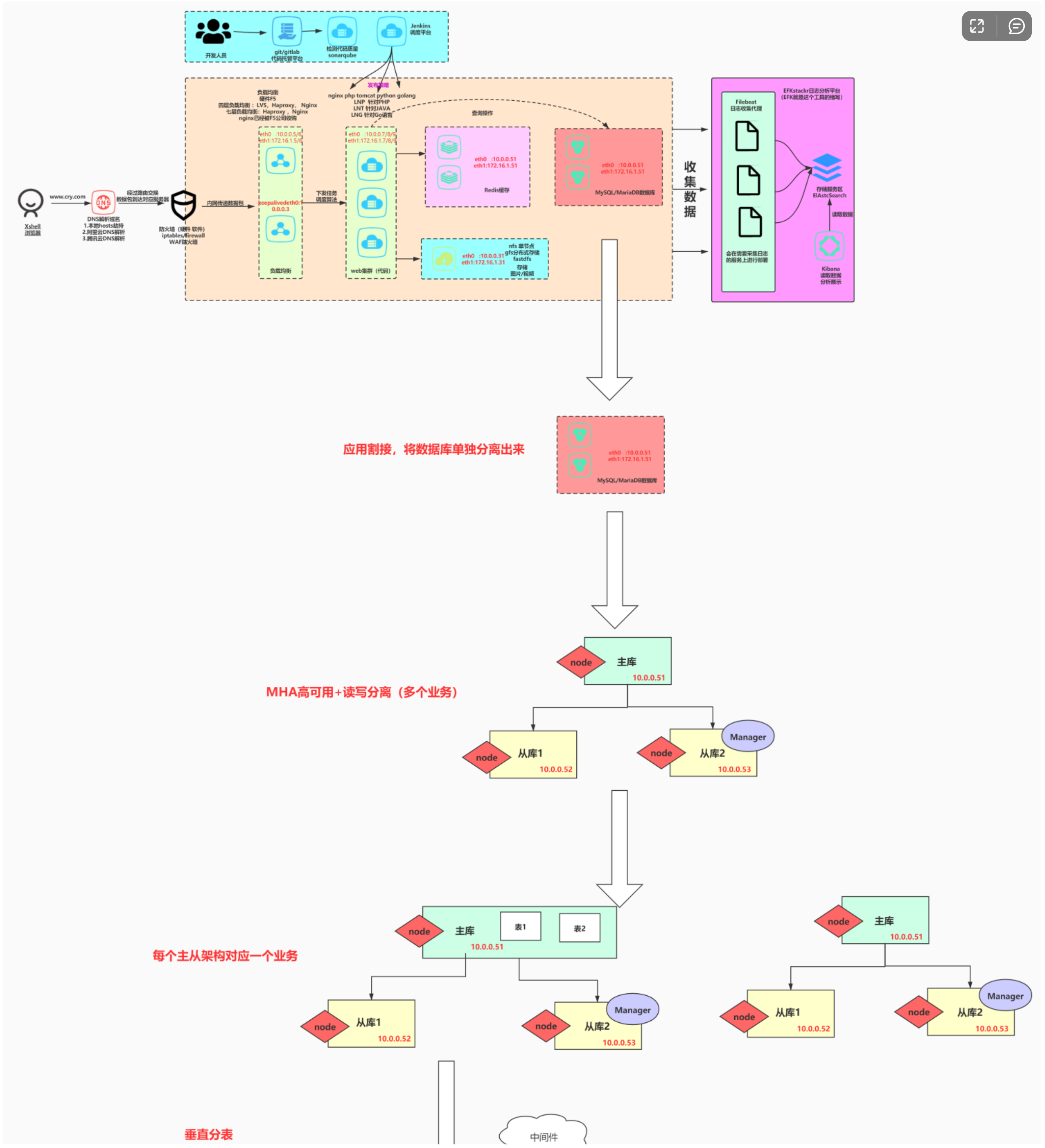
33.MySQL分布式架构-Mycat√

1.数据库架构的演变

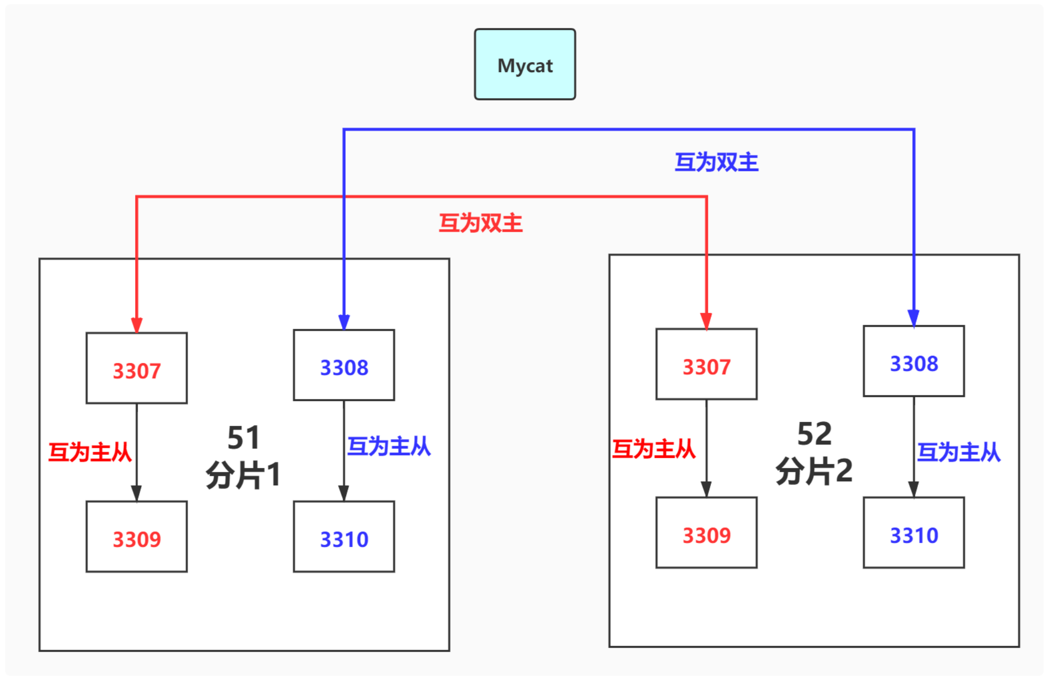

开发人员
git/gitlab 代码托管平台
检测代码质量 sonarqube
Jenkins 调度平台
发布策略
负载均衡
硬件F5
四层负载均衡：LVS，Haproxy，Nginx
七层负载均衡：Haproxy，Nginx
nginx已经被F5公司收购
nginx php tomcat python golang
LNP 针对PHP
LNT 针对JAVA
LNG 针对Go语言
查询操作
Xshell 浏览器
www.cry.com
DNS解析域名
1.本地hosts劫持
2.阿里云DNS解析
3.腾讯云DNS解析
经过路由交换 数据包到达对应服务器
防火墙（硬件 软件）
iptables/firewall
WAF防火墙
内网传递数据包
eth0 :10.0.0.5/6
eth1:172.16.1.5/6
keepalivedeth0:10.0.0.3
负载均衡
下发任务 调度算法
eth0 :10.0.0.7/8/9
eth1:172.16.1.7/8/9
web集群（代码）
eth0 :10.0.0.51
eth1:172.16.1.51
Redis缓存
eth0 :10.0.0.51
eth1:172.16.1.51
MySQL/MariaDB数据库
eth0 :10.0.0.31
eth1:172.16.1.31
nfs 单节点
gfs分布式存储
fastdfs
存储
图片/视频
收集数据
EFKstackr日志分析平台
(EFK就是这个工具的缩写)
Filebeat
日志收集代理
会在需要采集日志的服务上进行部署
存储服务区
ElAstcSearch
读取数据
Kibana
读取数据
分析展示
应用割接，将数据库单独分离出来
eth0 :10.0.0.51
eth1:172.16.1.51
MySQL/MariaDB数据库
MHA高可用+读写分离（多个业务）
node
主库
10.0.0.51
node
从库1
10.0.0.52
node
从库2
10.0.0.53
Manager
每个主从架构对应一个业务
node
主库
表1
表2
10.0.0.51
node
从库1
10.0.0.52
node
从库2
10.0.0.53
Manager
node
主库
10.0.0.51
node
从库1
10.0.0.52
node
从库2
10.0.0.53
Manager
垂直分表
中间件

```bash
1.遵守Mysql原生协议，跨语言，跨数据库的通用中间件代理
2.基于心跳的自动故障切换，支持读写分离，支持MySQL一双主多从，以及一主多从
3.有效管理数据源连接，基于数据分库，而不是分表的模式
4.基于NIO实现，有效管理线程，高并发问题
5.支持数据的多片自动路由与聚合，支持sum,count,max等常用的聚合函数
6.支持2表join，甚至基于caltlet的多表join
7.支持通过全局表，ER关系的分片策略，实现了高效的多表join查询
8.支持多租户方案
9.支持分布式事务
10.支持全局序列号，解决分布式下的主键生成问题
11.分布规则丰富，插件化开发，易于扩展
12.强大的web，命令行监控
13.支持前端所为mysql通用代理，后端JDBC支持oracle ,DB2, SQL server,mongodb
14.集群基于zookeeper管理，在线升级 扩容 智能优化
```

3.Mycat基础架构准备

两台虚拟机（2.5~3G内存）db01 db02

每台创建四个mysql实例：3307 3308 3309 3310



3.1 删除历史环境

```bash
pkill mysqld
rm -rf /data/33*
\mv /etc/my.cnf /etc/my.cnf.bak
```

3.2 创建相关目录初始化数据

```bash
mkdir /data/33{07..10}/data -p
mysqld --initialize-insecure  --user=mysql --datadir=/data/3307/data --basedir=/usr/local/mysql
mysqld --initialize-insecure  --user=mysql --datadir=/data/3308/data --basedir=/usr/local/mysql
mysqld --initialize-insecure  --user=mysql --datadir=/data/3309/data --basedir=/usr/local/mysql
mysqld --initialize-insecure  --user=mysql --datadir=/data/3310/data --basedir=/usr/local/mysql
```

3.3 准备配置文件和启动脚本

Bash Copy

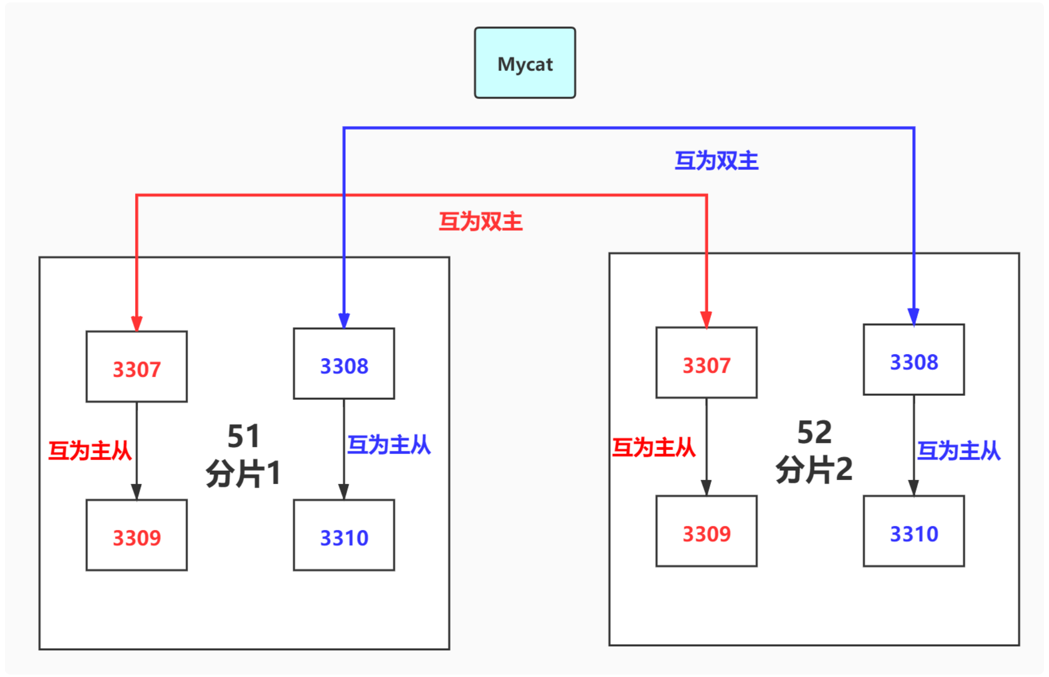
```bash
========db01==============
cat >/data/3307/my.cnf<<EOF
[mysqld]
basedir=/usr/local/mysql
datadir=/data/3307/data
socket=/data/3307/mysql.sock
port=3307
log-error=/data/3307/mysql.log
log_bin=/data/3307/mysql-bin
binlog_format=row
skip-name-resolve
server-id=7
gtid-mode=on
enforce-gtid-consistency=true
log-slave-updates=1
EOF

cat >/data/3308/my.cnf<<EOF
[mysqld]
basedir=/usr/local/mysql
datadir=/data/3308/data
port=3308
socket=/data/3308/mysql.sock
log-error=/data/3308/mysql.log
log_bin=/data/3308/mysql-bin
binlog_format=row
skip-name-resolve
server-id=8
gtid-mode=on
enforce-gtid-consistency=true
log-slave-updates=1
EOF

cat >/data/3309/my.cnf<<EOF
[mysqld]
basedir=/usr/local/mysql
```

3.4 修改权限，启动多实例

```bash
chown -R mysql.mysql /data/*
systemctl start mysqld3307
systemctl start mysqld3308
systemctl start mysqld3309
systemctl start mysqld3310

mysql -S /data/3307/mysql.sock -e "show variables like 'server_id'"
mysql -S /data/3308/mysql.sock -e "show variables like 'server_id'"
mysql -S /data/3309/mysql.sock -e "show variables like 'server_id'"
mysql -S /data/3310/mysql.sock -e "show variables like 'server_id'"
==========================db01===================================
[root@db01 ~]# mysql -S /data/3307/mysql.sock -e "show variables like 'server_id'"
ke 'server_id'"+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 7     |
+---------------+-------+
[root@db01 ~]# mysql -S /data/3308/mysql.sock -e "show variables like 'server_id'"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 8     |
+---------------+-------+
[root@db01 ~]# mysql -S /data/3309/mysql.sock -e "show variables like 'server_id'"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 9     |
+---------------+-------+
[root@db01 ~]# mysql -S /data/3310/mysql.sock -e "show variables like 'server_id'"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 10    |
+---------------+-------+

==========================db02===================================
[root@db02 ~]# mysql -S /data/3307/mysql.sock -e "show variables like 'server_id'"
ke 'server_id'"+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 17    |
+---------------+-------+
[root@db02 ~]# mysql -S /data/3308/mysql.sock -e "show variables like 'server_id'"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 18    |
+---------------+-------+
[root@db02 ~]# mysql -S /data/3309/mysql.sock -e "show variables like 'server_id'"
```

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 19    |
+---------------+-------+
[root@db02 ~]# mysql -S /data/3310/mysql.sock -e "show variables like 'server_id'"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 20    |
+---------------+-------+

```

3.5 开始配置主从环境



shard1(分片1)

10.0.0.51:3307 <-----> 10.0.0.52:3307 互为双主

```bash
# db02
mysql  -S /data/3307/mysql.sock -e "create user repl@'10.0.0.%' identified with mysql_native_password by '123

mysql  -S /data/3307/mysql.sock -e "create user root@'10.0.0.%' identified with mysql_native_password by '123

# db01
mysql  -S /data/3307/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.52', MASTER_PORT=3307, MASTER_AUTO_P(
mysql  -S /data/3307/mysql.sock -e "start slave;"
mysql  -S /data/3307/mysql.sock -e "show slave status\G"|grep Running:

# db02

mysql  -S /data/3307/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.51', MASTER_PORT=3307, MASTER_AUTO_P(
mysql  -S /data/3307/mysql.sock -e "start slave;"
mysql  -S /data/3307/mysql.sock -e "show slave status\G"|grep Running:


```

10.0.0.51:3309 ------> 10.0.0.51:3307 互为主从

```bash
# db01

mysql  -S /data/3309/mysql.sock  -e "CHANGE MASTER TO MASTER_HOST='10.0.0.51', MASTER_PORT=3307, MASTER_AUTO_P(
mysql  -S /data/3309/mysql.sock  -e "start slave;"
mysql  -S /data/3309/mysql.sock  -e "show slave status\G"|grep Running:
```

10.0.0.52:3309 ------> 10.0.0.52:3307 互为主从

```bash
mysql  -S /data/3309/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.52', MASTER_PORT=3307, MASTER_AUTO_POS
mysql  -S /data/3309/mysql.sock -e "start slave;"
mysql  -S /data/3309/mysql.sock -e "show slave status\G"|grep Running:
```

shard2(分片2)

10.0.0.52:3308 <-----> 10.0.0.51:3308 互为双主

Bash Copy

```bash
# db01

mysql  -S /data/3308/mysql.sock -e "create user repl@'10.0.0.%' identified with mysql_native_password by '123

mysql  -S /data/3308/mysql.sock -e "create user root@'10.0.0.%' identified with mysql_native_password by '123

# db02

mysql  -S /data/3308/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.51', MASTER_PORT=3308, MASTER_AUTO_P
mysql  -S /data/3308/mysql.sock -e "start slave;"
mysql  -S /data/3308/mysql.sock -e "show slave status\G"|grep Running:

# db01

mysql  -S /data/3308/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.52', MASTER_PORT=3308, MASTER_AUTO_P
mysql  -S /data/3308/mysql.sock -e "start slave;"
mysql  -S /data/3308/mysql.sock -e "show slave status\G"|grep Running:

```

10.0.0.52:3310 ----–> 10.0.0.52:3308 互为主从

Bash Copy

```bash
#db02
mysql  -S /data/3310/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.52', MASTER_PORT=3308, MASTER_AUTO_POS
mysql  -S /data/3310/mysql.sock -e "start slave;"
mysql  -S /data/3310/mysql.sock -e "show slave status\G"|grep Running:
```

10.0.0.51:3310 ----–> 10.0.0.51:3308 互为主从

Bash Copy

```bash
# db01

mysql  -S /data/3310/mysql.sock -e "CHANGE MASTER TO MASTER_HOST='10.0.0.51', MASTER_PORT=3308, MASTER_AUTO_POS
mysql  -S /data/3310/mysql.sock -e "start slave;"
mysql  -S /data/3310/mysql.sock -e "show slave status\G"|grep Running:
```

3.6 检测主从状态

```bash
mysql -S /data/3307/mysql.sock -e "show slave status\G"|grep Yes
mysql -S /data/3308/mysql.sock -e "show slave status\G"|grep Yes
mysql -S /data/3309/mysql.sock -e "show slave status\G"|grep Yes
mysql -S /data/3310/mysql.sock -e "show slave status\G"|grep Yes

注：如果中间出现错误，在每个节点进行执行以下命令，从2.6从头执行
mysql -S /data/3307/mysql.sock -e "stop slave; reset slave all;"
mysql -S /data/3308/mysql.sock -e "stop slave; reset slave all;"
mysql -S /data/3309/mysql.sock -e "stop slave; reset slave all;"
mysql -S /data/3310/mysql.sock -e "stop slave; reset slave all;"

```

4.Mycat安装（db02节点上）

4.0 简介

开源组织和社区开发人员，在淘宝cobar（TDDL）基础上二次开发。Mycat后来被爱可生改写成了DBLE

4.1 预先安装Java运行环境

```bash
yum install -y java
```

4.2 上传软件包

Mycat-server-1.6.7.4-release-20200105164103-linux .tar.gz（21.8 MB）

```bash
cd /opt
tar xf Mycat-server-1.6.7.4-release-20200105164103-linux_.tar.gz
```

4.3 启动和连接

4.3.1 配置环境变量

```bash
vim /etc/profile
export PATH=/opt/mycat/bin:$PATH
source /etc/profile
```

4.3.2 启动

```bash
mycat start
mycat端口号8066
[root@db02 mycat]# netstat -lntup|grep java
tcp        0      0 127.0.0.1:32000         0.0.0.0:*               LISTEN      4050/java
tcp6       0      0 :::1984                 :::*                    LISTEN      4050/java
tcp6       0      0 :::8066                 :::*                    LISTEN      4050/java
tcp6       0      0 :::10276                :::*                    LISTEN      4050/java
tcp6       0      0 :::9066                 :::*                    LISTEN      4050/java
tcp6       0      0 :::8586                 :::*                    LISTEN      4050/java
```

4.3.3 连接mycat

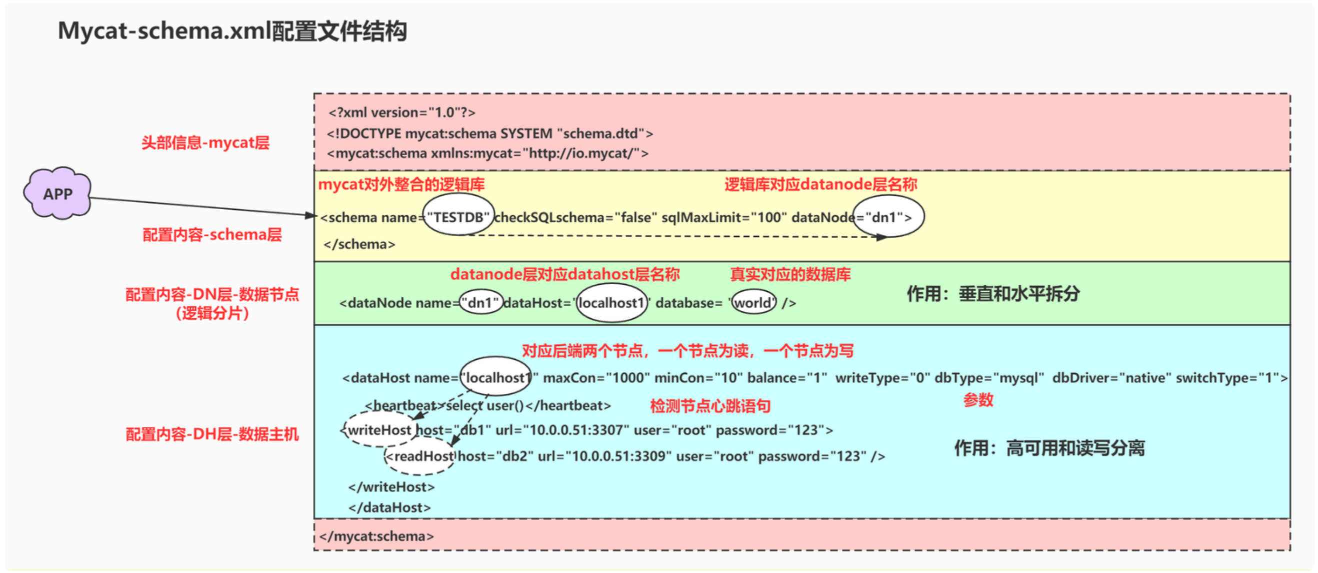
```bash
8.0之前
mysql -uroot -p123456 -h 127.0.0.1 -P8066

8.0之后:
mysql -uroot -p123456 -h10.0.0.52 -P8066 --default-auth=mysql_native_password
```

5.Mycat软件结构

```bash
[root@db02 mycat]# cd /opt/mycat/
[root@db02 mycat]# ll
总用量 12
drwxr-xr-x 2 root root  190 5月  20 20:11 bin
drwxrwxrwx 2 root root    6 10月 22 2019 catlet
drwxrwxrwx 4 root root 4096 5月  20 20:11 conf
drwxr-xr-x 2 root root 4096 5月  20 20:11 lib
drwxrwxrwx 2 root root   77 5月  20 20:15 logs
drwxr-xr-x 2 root root    6 5月  20 20:15 tmlogs
-rwxrwxrwx 1 root root  227 1月   5 2020 version.txt



1.bin   存放可程序的目录
2.conf  存放配置文件的目录----->重点关注

conf目录下
2.1  schema.xml
主配置文件：节点信息、读写分离、高可用设置、调用分片策略..
2.2 rule.xml
分片策略的定义、功能、使用用方法
2.3 server.xml
mycat服务有关配置： 用户、网络、权限、策略、资源...
2.4  xx.txt文件
分片参数定义文件
2.5 log4j2.xml
Mycat 相关日志记录配置

3.lib   存放库的目录
4.logs  存放日志目录----->重点关注
logs目录下
4.1
wrapper.log : 启动日志
mycat.log   : 工作日志
```

6.Mycat 核心配置文件

6.1 schema.xml配置文件结构


Mycat-schema.xml配置文件结构
头部信息-mycat层
APP
配置内容-schema层
配置内容-DN层-数据节点（逻辑分片）
配置内容-DH层-数据主机
mycat对外整合的逻辑库
逻辑库对应datanode层名称
datanode层对应datahost层名称
真实对应的数据库
作用：垂直和水平拆分
对应后端两个节点，一个节点为读，一个节点为写
参数
检测节点心跳语句
作用：高可用和读写分离

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">
</schema>
  <dataNode name="dn1" dataHost='localhost1' database= 'world' />
  <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver="native" switchType="1">
     <heartbeat>select user()</heartbeat>
  <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
       <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
   </writeHost>
   </dataHost>
</mycat:schema>
```

```
# 逻辑库:
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">
</schema>
# DN数据节点（逻辑分片）：数据节点（逻辑分片）：
    <dataNode name="dn1" dataHost="localhost1" database= "world" />
作用:
  垂直和水平查分。
# DH 数据主机
作用:  高可用和读写分离
    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver=
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    </dataHost>
</mycat:schema>

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


# 1.逻辑库配置
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">
</schema>


# 2. DN，分片定义
    <dataNode name="dn1" dataHost="localhost1" database= "world" />


#3. DH节点定义
    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver=
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    </dataHost>

</mycat:schema>

```

6.2 schema.xml配置文件模板

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">
</schema>
    <dataNode name="dn1" dataHost="localhost1" database= "world" />
    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver=
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    </dataHost>
</mycat:schema>
```

6.3 配置mycat,测试读写分离

6.3.1 修改mycat主配置文件

Mycat软件装在db02节点上了

```
[root@db02 ~]# cd /opt/mycat/conf
[root@db02 ~]# mv schema.xml schema.xml.bak
[root@db02 ~]# vim schema.xml
使用模板配置
```

6.3.2 测试环境的准备

```
对单个节点两个主库实列上传数据，会同步到另一个节点的双主实列，也会同步到本节点的从库实列中
 mysql -S /data/3307/mysql.sock -e  "source /opt/world.sql"
 mysql -S /data/3308/mysql.sock -e  "source /opt/world.sql"
```

6.3.3 重启mycat

```
[root@db02 ~]# mycat restart
Stopping Mycat-server...
Stopped Mycat-server.
Starting Mycat-server...
```

6.3.4 连接安装在db02节点上的mycat进行读写分离测试

```bash
根据mycat配置文件进行读写分离
也可以使用MHA+proxysql基于sql语句方式的读写分离
写操作---->10.0.0.51:3307
[root@db02 conf]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "begin ;
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|           7 |
+-------------+
读操作---->10.0.0.51:33
[root@db02 conf]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "select @
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|           9 |
+-------------+

```

6.4 Mycat高可用配置（db02节点上）

mycat高可用要配合gtid和增强半同步一起使用，防止异步复制中间的数据丢失

6.4.0 测试环境的准备

```bash
对单个节点两个主库实列上传数据，会同步到另一个节点的双主实列，也会同步到本节点的从库实列中
 mysql -S /data/3307/mysql.sock -e  "source /opt/world.sql"
 mysql -S /data/3308/mysql.sock -e  "source /opt/world.sql"
```

6.4.1 修改配置文件为高可用

```bash
[root@db02 ~]# cd /opt/mycat/conf/
[root@db02 conf]# mv schema.xml schema.xml.bak1
[root@db02 conf]# vim schema.xml

<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">
</schema>
    <dataNode name="sh1" dataHost="oldguo1" database= "world" />
    <dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver="na
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    <writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />
    </writeHost>
    </dataHost>
</mycat:schema>

primary  writehost    : 负责写操作的writehost
standby  writeHost    : 和readhost一样，只提供读服务
当写节点宕机后，后面跟的readhost也不提供服务，这时候standby的writehost就提供写服务，后面跟的readhost提供读服务
备注:
1.db1作为主写时 db02 db03 db04都是丛写 一主三从
2.db01宕机 db02配置也会生效 因为db01和db02是一组,高可用成为db03为主写 db04为从读
3.db01修复好 会成为从读节点
```

6.4.2 测试读写分离

Bash Copy

```bash
1.重启mycat
mycat restart
2.测试读 （一主三从 会在三个读节点轮询，减少了读压力）
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "select @@
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|          19 |
+-------------+
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "select @@
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|          17 |
+-------------+
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "select @@
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|           9 |
+-------------+
3.测试写操作，只会在主节点（10.0.0.51：3307）
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "begin ; s
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|           7 |
+-------------+
```

6.4.3 测试高可用

```bash
1.停止10.0.0.51: 3307主节点实列
systemctl stop mysqld3307
2.等待一会
3.10.0.0.51: 3307主节点转化到10.0.0.52: 3307
写
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "begin ; s
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|          17 |
+-------------+
读 只有10.0.0.52: 3309
[root@db02 opt]# mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password  -e "select @@
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+
| @@server_id |
+-------------+
|          19 |
+-------------+
4.修复10.0.0.51: 3307
10.0.0.51: 3307 和 10.0.0.51: 3309 都会是读节点
重新构成一主三从
10.0.0.52: 3307 主-写
10.0.0.51: 3307 从-读
10.0.0.51: 3309 从-读
10.0.0.52: 3309 从-读
```

6.5 Mycat配置文件中参数介绍

```bash
 <dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver="nati
```

6.5.1 balance属性

```bash
读操作负载均衡类型，目前的取值有3种：
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。
使用场景：不用！！！
2. balance="1"，全部的readHost与standby 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。
使用场景：默认取值为1，常用
3. balance="2"，所有读操作都随机的在writeHost、readhost上分发。
使用场景：当主库压力较小时，可以让主库也进行读操作
```

6.5.2 writeType属性

```bash
写操作，负载均衡类型，目前的取值有2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主，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，但不推荐使用
```

6.5.3 switchType属性

```bash
-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
```

其他参数

6.5.4 连接有关

```bash
maxCon="1000"：最大的并发连接数
minCon="10" ：mycat在启动之后，会在后端节点上自动开启的连接线程 （预备连接） 根据后端数据库内存大小设置！！！
```

6.5.5 tempReadHostAvailable="1"

```bash
这个一主一从时（1个writehost，1个readhost时），可以开启这个参数，如果2个writehost，2个readhost时
```

6.5.6 sqlMaxLimit="100"

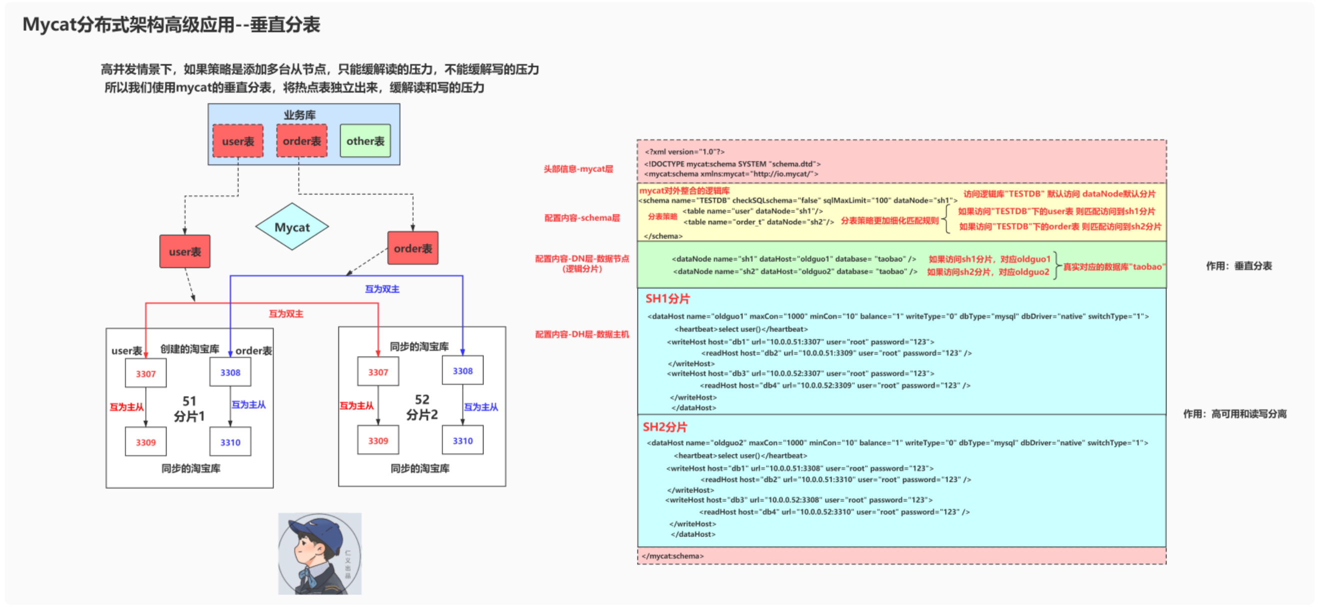
```
显示sql结果最多显示多少行
```

6.5.7 <heartbeat>select user()</heartbeat>

```
监测心跳的语句 select user()
```

7.Mycat分布式架构高级应用--垂直分表

7.1 原理图


Mycat分布式架构高级应用--垂直分表
高并发情景下，如果策略是添加多台从节点，只能缓解读的压力，不能缓解写的压力
所以我们使用mycat的垂直分表，将热点表独立出来，缓解读和写的压力
业务库
user表
order表
other表
Mycat
user表
order表
互为双主
互为双主
user表
创建的淘宝库
order表
3307
3308
互为主从
51
分片1
互为主从
3309
3310
同步的淘宝库
同步的淘宝库
3307
3308
互为主从
52
分片2
互为主从
3309
3310
同步的淘宝库
头部信息-mycat层
配置内容-schema层
配置内容-DN层-数据节点（逻辑分片）
配置内容-DH层-数据主机
mycat对外整合的逻辑库
分表策略
分表策略更加细化匹配规则
访问逻辑库"TESTDB" 默认访问 dataNode默认分片
如果访问"TESTDB"下的user表 则匹配访问到sh1分片
如果访问"TESTDB"下的order表 则匹配访问到sh2分片
如果访问sh1分片，对应oldguo1
如果访问sh2分片，对应oldguo2
真实对应的数据库"taobao"
SH1分片
SH2分片
作用：垂直分表
作用：高可用和读写分离

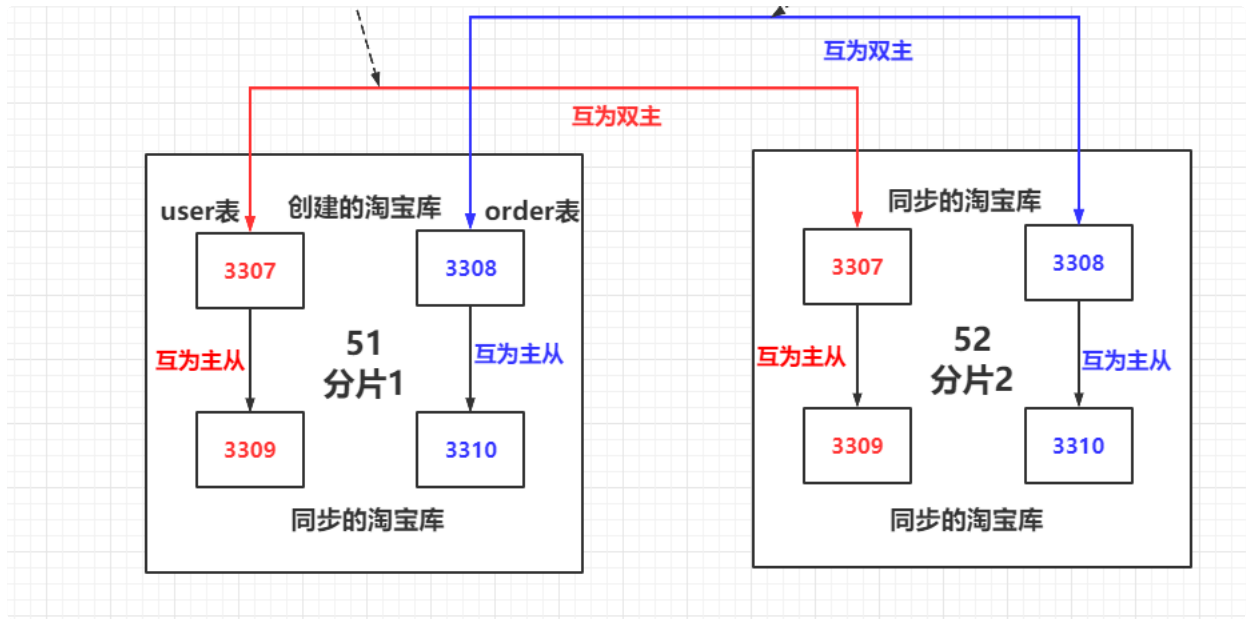
```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">
        <table name="user" dataNode="sh1"/>
        <table name="order_t" dataNode="sh2"/>
</schema>
        <dataNode name="sh1" dataHost="oldguo1" database= "taobao" />
        <dataNode name="sh2" dataHost="oldguo2" database= "taobao" />
 <dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    <writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />
    </writeHost>
    </dataHost>
 <dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />
    </writeHost>
    <writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />
    </writeHost>
    </dataHost>
</mycat:schema>
```

7.2 实现垂直分表

7.2.1 编辑配置文件

```
[root@db02 ~]# cd /opt/mycat/conf/
[root@db02 conf]# mv schema.xml schema.xml.1
[root@db02 conf]# vim schema.xml
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">
        <table name="user" dataNode="sh1"/>
        <table name="order_t" dataNode="sh2"/>
</schema>
    <dataNode name="sh1" dataHost="oldguo1" database= "taobao" />
    <dataNode name="sh2" dataHost="oldguo2" database= "taobao" />
    <dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver="na
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
    </writeHost>
    <writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />
    </writeHost>
    </dataHost>

    <dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql"  dbDriver="na
        <heartbeat>select user()</heartbeat>
    <writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />
    </writeHost>
    <writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />
    </writeHost>
    </dataHost>

</mycat:schema>
```

7.2.2 准备垂直分表功能的库和表



```bash
创建测试库（在两个分片的主节点上）
mysql -S /data/3307/mysql.sock -e "create database taobao charset utf8;"
mysql -S /data/3308/mysql.sock -e "create database taobao charset utf8;"

创建测试库下需要垂直分开的表（一个分片主节点对应一个表）
mysql -S /data/3307/mysql.sock -e "use taobao;create table user(id int,name varchar(20))";
mysql -S /data/3308/mysql.sock -e "use taobao;create table order_t(id int,name varchar(20))"
```

7.2.3 重启Mycat

```bash
mycat restart
```

7.2.4 测试–连接到mycat,分表对两张表插入数据

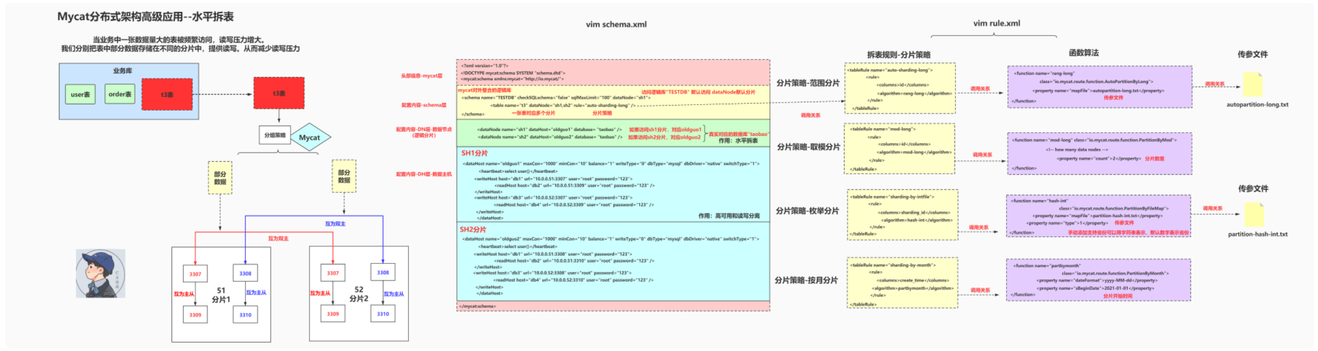
```bash
1.登陆到mycat
mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password
2.查看mycat下库（逻辑库）
mysql> show databases;
+----------+
| DATABASE |
+----------+
| TESTDB   |
+----------+
3.进入逻辑库查看表，mycat已经将两个物理层面分开的表，逻辑整合为一个库下
mysql> show tables;
+------------------+
| Tables_in_taobao |
+------------------+
| order_t          |
| user             |
+------------------+
4.两张表插入数据
4.1 对user表
insert into user values(1,'a');
insert into user values(2,'b');
insert into user values(3,'c');
commit;
4.2 对order表
insert into order_t values(1,'x'),(2,'y');
commit;
```

7.2.5 查看垂直分表结果

```bash
mysql -S /data/3307/mysql.sock -e "show tables from taobao"
+------------------+
| Tables_in_taobao |
+------------------+
| user             |
+------------------+
mysql -S /data/3308/mysql.sock -e "show tables from taobao"
+------------------+
| Tables_in_taobao |
+------------------+
| order_t          |
+------------------+
```

8.Mycat分布式架构高级应用--水平拆表

8.0 原理图



```xml
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
  <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="sh1">
          <table name="t3" dataNode="sh1,sh2" rule="auto-sharding-long" />
  </schema>
      <dataNode name="sh1" dataHost="oldguo1" database= "taobao" />
      <dataNode name="sh2" dataHost="oldguo2" database= "taobao" />
  <dataHost name="oldguo1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">
        <heartbeat>select user()</heartbeat>
     <writeHost host="db1" url="10.0.0.51:3307" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3309" user="root" password="123" />
     </writeHost>
     <writeHost host="db3" url="10.0.0.52:3307" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3309" user="root" password="123" />
      </writeHost>
      </dataHost>
  <dataHost name="oldguo2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native" switchType="1">
        <heartbeat>select user()</heartbeat>
     <writeHost host="db1" url="10.0.0.51:3308" user="root" password="123">
            <readHost host="db2" url="10.0.0.51:3310" user="root" password="123" />
     </writeHost>
     <writeHost host="db3" url="10.0.0.52:3308" user="root" password="123">
            <readHost host="db4" url="10.0.0.52:3310" user="root" password="123" />
      </writeHost>
      </dataHost>
</mycat:schema>
```

```xml
<tableRule name="auto-sharding-long">
        <rule>
              <columns>id</columns>
              <algorithm>rang-long</algorithm>
        </rule>
  </tableRule>
<function name="rang-long"
              class="io.mycat.route.function.AutoPartitionByLong">
        <property name="mapFile">autopartition-long.txt</property>
</function>

<tableRule name="mod-long">
        <rule>
              <columns>id</columns>
              <algorithm>mod-long</algorithm>
        </rule>
</tableRule>
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
        <!-- how many data nodes -->
              <property name="count">2</property>
</function>

<tableRule name="sharding-by-intfile">
        <rule>
              <columns>sharding_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">1</property>
</function>

<tableRule name="sharding-by-month">
        <rule>
              <columns>create_time</columns>
            <algorithm>partbymonth</algorithm>
        </rule>
</tableRule>
<function name="partbymonth"
              class="io.mycat.route.function.PartitionByMonth">
        <property name="dateFormat">yyyy-MM-dd</property>
        <property name="sBeginDate">2021-01-01</property>
</function>
```

8.1 重要概念

8.1.1 分片策略

```bash
分片策略 ：Mycat几乎融合经典业务中大部分的分片策略。Mycat已经开发了相应算法，非常方便调用。
分片策略种类：
1.范围分片
2.取模  取余        num/3  0 1 2
3.枚举
4.日期范围
5.HASH
等等
```

8.1.2 .分片键

```bash
作为分片策略条件的列
```

8.2 分片策略-范围分片(auto-sharding-long)

8.2.1 使用场景

```bash
一张2000w数据行的表
1.行数非常的多，2000w（1-1000w:sh1 1000-2000w:sh2)
2.访问非常的频繁，用户顺序访问较多
```

8.2.2 修改配置文件，定制分片策略

Bash Copy

```bash
cd /opt/mycat/conf
cp schema.xml schema.xml.2

schmea.xml中的配置调用用rule.xml的分片策略auto-sharding-long，分片策略调用函数算法rang-long，函数算法的传参通过autopartit
vim schema.xml
添加分片策略:
<table name="t3" dataNode="sh1,sh2" rule="auto-sharding-long" />
  表名             对应多个分片       分片策略


支持的分片策略会在rule.xml中存在
vim rule.xml
 <tableRule name="auto-sharding-long">
                <rule>
                        <columns>id</columns>            分片键
                        <algorithm>rang-long</algorithm> 函数算法
                </rule>
        </tableRule>
函数算法
 <function name="rang-long"
                          class="io.mycat.route.function.AutoPartitionByLong">
                <property name="mapFile">autopartition-long.txt</property>  传参数通过一个txt文件
        </function>

 vim autopartition-long.txt
# range start-end ,data node index  range范围分片 起始-终点  节点号码（索引）
# K=1000,M=10000 一千用k表示 一万用M表示
根据分片键id列进行范围划分
0-10=0 id大于0，小于等于10 匹配0分片
10-20=1 id大于10，小于等于20 匹配1分片
```

8.2.3 创建测试表

Bash Copy

```bash
mysql -S /data/3307/mysql.sock -e "use taobao;create table t3 (id int not null primary key auto_increment,name
mysql -S /data/3308/mysql.sock  -e "use taobao;create table t3 (id int not null primary key auto_increment,name
```

8.2.4 重启mycat

```bash
mycat restart
```

8.2.5 测试表插入数据

```bash
mysql -uroot -p123456 -h 10.0.0.52 -P 8066  --default-auth=mysql_native_password
use TESTDB
insert into t3(id,name) values(1,'a');
insert into t3(id,name) values(2,'b');
insert into t3(id,name) values(3,'c');
insert into t3(id,name) values(4,'d');
insert into t3(id,name) values(11,'aa');
insert into t3(id,name) values(12,'bb');
insert into t3(id,name) values(13,'cc');
insert into t3(id,name) values(14,'dd');
```

8.2.6 查看各分片数据部署情况

```bash
一张表的数据拆分在两个分片中
[root@db01 ~]# mysql -S /data/3307/mysql.sock  -e "select * from taobao.t3"
+----+------+
| id | name |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
+----+------+
[root@db01 ~]#
[root@db01 ~]# mysql -S /data/3308/mysql.sock  -e "select * from taobao.t3"
+----+------+
| id | name |
+----+------+
| 11 | aa   |
| 12 | bb   |
| 13 | cc   |
| 14 | dd   |
+----+------+

mycat视角还是一张表
mysql> show tables;
+------------------+
| Tables_in_taobao |
+------------------+
| t3               |
+------------------+
1 row in set (0.01 sec)

mysql> select * from t3;
+----+------+
| id | name |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
| 11 | aa   |
| 12 | bb   |
| 13 | cc   |
| 14 | dd   |
+----+------+
```

8.3 分片策略-取模分片（mod-long）

解决：范围分片中连续的范围为一个分片，出现访问压力集中在同一分片上。取模分片会将分片数据打散减少访问压力

8.3.1 取模算法

```bash
1%3 1
2%3 2
3%3 0
4%3 1
5%3 2
任何正整数数字和N（正整数）取模，得的值永远都是 0~N-1

id % 分片数量取模
N  % 5  =  0-4   idx
取模分片方式：分片键（一个列）与节点数量（分片数量）进行取余，得到余数，将数据写入对应节点。（关注点）
```

8.3.2 创建测试表

```bash
mysql -S /data/3307/mysql.sock
create table taobao.t5 (
id  int not null  primary key auto_increment,
name varchar(20) not null  ) charset utf8 ;

mysql -S /data/3308/mysql.sock
create table taobao.t5 (
id  int not null  primary key auto_increment,
name varchar(20) not null) charset utf8 ;
```

8.3.3 修改配置文件，定制分片策略

①修改mycat配置文件 schema.xml

```bash
vim schema.xml
添加:
<table name="t5" dataNode="sh1,sh2" rule="mod-long" />
```

② 查看rule.xml对应的分片规则

```bash
vim rule.xml
<tableRule name="mod-long">
                <rule>
                        <columns>id</columns>
                        <algorithm>mod-long</algorithm>
                </rule>
        </tableRule>

 <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
                <!-- how many data nodes -->  多少个数据节点
                <property name="count">2</property> 我们有两个数据节点（分片数量）
        </function>
```

8.3.4 重启mycat

```bash
mycat restart
```

8.3.5 登陆mycat录入数据

```bash
mysql -uroot -p123456 -h 10.0.0.52 -P8066 --default-auth=mysql_native_password
use TESTDB
insert into t5(id,name) values(1,'a');
insert into t5(id,name) values(2,'b');
insert into t5(id,name) values(3,'c');
insert into t5(id,name) values(4,'d');
insert into t5(id,name) values(6,'x'),(8,'y'),(10,'z');

mysql> select * from t5;
+----+------+
| id | name |
+----+------+
|  2 | b    |
|  4 | d    |
|  6 | x    |
|  8 | y    |
| 10 | z    |
|  1 | a    |
|  3 | c    |
+----+------+
```

8.3.6 查看各分片数据部署情况

```bash
[root@db02 conf]# mysql -S /data/3308/mysql.sock  -e "select * from taobao.t5"
+----+------+
| id | name |
+----+------+
|  1 | a    |
|  3 | c    |
+----+------+
[root@db02 conf]# mysql -S /data/3307/mysql.sock  -e "select * from taobao.t5"
+----+------+
| id | name |
+----+------+
|  2 | b    |
|  4 | d    |
|  6 | x    |
|  8 | y    |
| 10 | z    |
+----+------+
```

8.4 分片策略–枚举(区域)分片（sharding–by–intfile）

8.4.1 创建测试表（根据区域进行对分片的录入数据）

```bash
mysql -S /data/3307/mysql.sock -e "use taobao;create table t6 (id int not null primary key auto_increment,name
mysql -S /data/3308/mysql.sock -e "use taobao;create table t6 (id int not null primary key auto_increment,name
```

8.4.2 修改配置文件，定制分片策略

① 修改mycat配置文件 schema.xml

```bash
vim schema.xml
<table name="t6" dataNode="sh1,sh2" rule="sharding-by-intfile" />
```

② 查看rule.xml对应的分片规则

```bash
vim rule.xml
<tableRule name="sharding-by-intfile">
                <rule>
                        <columns>name</columns>   修改表中表示省份的列名
                        <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">1</property>                           手动添加支持省份可以用字符串表示，默认数

        </function>

vim partition-hash-int.txt 默认文件中是使用数字表示省份，开启type后，可以使用字符串表示省份
bj=0
sh=1
DEFAULT_NODE=1 默认分片（不满足条件默认加入1中）
```

8.4.3 重启mycat

```bash
mycat restart
```

8.4.4 登陆mycat录入数据

```bash
mysql -uroot -p123456 -h10.0.0.52 -P8066 --default-auth=mysql_native_password
use TESTDB
insert into t6(id,name) values(1,'bj');
insert into t6(id,name) values(2,'sh');
insert into t6(id,name) values(3,'bj');
insert into t6(id,name) values(4,'sh');
insert into t6(id,name) values(5,'tj');

mysql> select * from t6;
+----+------+
| id | name |
+----+------+
|  1 | bj   |
|  3 | bj   |
|  2 | sh   |
|  4 | sh   |
|  5 | tj   |
+----+------+
```

8.4.5 查看各分片数据部署情况

```bash
北京区域
[root@db02 conf]# mysql -S /data/3307/mysql.sock  -e "select * from taobao.t6"
+----+------+
| id | name |
+----+------+
|  1 | bj   |
|  3 | bj   |
+----+------+
上海区域+没有定义的区域走默认分片的信息
[root@db02 conf]# mysql -S /data/3308/mysql.sock  -e "select * from taobao.t6"
+----+------+
| id | name |
+----+------+
|  2 | sh   |
|  4 | sh   |
|  5 | tj   |
+----+------+
```

8.5 分片策略–按月分片(sharding–by–month)

8.5.1 创建测试表

```bash
mysql -S /data/3307/mysql.sock
create table taobao.t4 (
id  int not null  primary key auto_increment,
name varchar(20) not null ,
create_time datetime not null ) charset utf8 ;

mysql -S /data/3308/mysql.sock
create table taobao.t4 (
id  int not null  primary key auto_increment,
name varchar(20) not null ,
create_time datetime not null ) charset utf8 ;
```

8.5.2 修改配置文件，定制分片策略

① 修改mycat配置文件 schema.xml

```bash
vim schema.xml
添加:
<table name="t4" dataNode="sh1,sh2" rule="sharding-by-month" />
```

② 查看rule.xml对应的分片规则

```bash
vim rule.xml
<tableRule name="sharding-by-month">
                <rule>
                        <columns>create_time</columns>
                        <algorithm>partbymonth</algorithm>
                </rule>
        </tableRule>

<function name="partbymonth"
                          class="io.mycat.route.function.PartitionByMonth">
                <property name="dateFormat">yyyy-MM-dd</property>
                <property name="sBeginDate">2021-01-01</property>  修改为2021  每个月一个分片
        </function>
```

8.5.3 重启mycat

```bash
mycat restart
```

8.5.4 登陆mycat录入数据

```bash
mysql -uroot -p123456 -h10.0.0.52 -P8066 --default-auth=mysql_native_password
use TESTDB
insert into t4(id,name,create_time) values(1,'a','2021-01-01 10:00:00');
insert into t4(id,name,create_time) values(2,'b','2021-02-02 10:00:00');
insert into t4(id,name,create_time) values(3,'c','2021-01-02 10:00:00');
insert into t4(id,name,create_time) values(4,'d','2021-02-12 10:00:00');
insert into t4(id,name,create_time) values(5,'e','2021-01-22 10:00:00');
commit;

mysql> select * from t4;
+----+------+---------------------+
| id | name | create_time         |
+----+------+---------------------+
|  1 | a    | 2021-01-01 10:00:00 |
|  3 | c    | 2021-01-02 10:00:00 |
|  5 | e    | 2021-01-22 10:00:00 |
|  2 | b    | 2021-02-02 10:00:00 |
|  4 | d    | 2021-02-12 10:00:00 |
+----+------+---------------------+
```

8.5.5 查看各分片数据部署情况

```bash
分片应该有12个，一个月对应一个分片存储数据
一月
[root@db02 conf]# mysql -S /data/3307/mysql.sock -e "select * from taobao.t4;"
+----+------+---------------------+
| id | name | create_time         |
+----+------+---------------------+
|  1 | a    | 2021-01-01 10:00:00 |
|  3 | c    | 2021-01-02 10:00:00 |
|  5 | e    | 2021-01-22 10:00:00 |
+----+------+---------------------+
二月
[root@db02 conf]# mysql -S /data/3308/mysql.sock -e "select * from taobao.t4;"
+----+------+---------------------+
| id | name | create_time         |
+----+------+---------------------+
|  2 | b    | 2021-02-02 10:00:00 |
|  4 | d    | 2021-02-12 10:00:00 |
+----+------+---------------------+
```

9.Mycat分布式架构处理“多表查询”解决方案-全局表

9.1 简介

```
当面临多表查询情况，mycat会出现跨分片join，mycat会先将后端节点数据进行整合，再进行多表连接，提供查询。性能上会消耗很多。
使用场景：
如果你的业务中有些数据类似于数据字典，比如配置文件的配置，
常用业务的配置或者数据量不大很少变动的表，这些表往往不是特别大，
而且大部分的业务场景都会用到，那么这种表适合于Mycat全局表，无须对数据进行切分，
要在所有的分片上保存一份数据即可，Mycat 在Join操作中，业务表与全局表进行Join聚合会优先选择相同分片内的全局表join，
避免跨库Join，在进行数据插入操作时，mycat将把数据分发到全局表对应的所有分片执行，在进行数据读取时候将会随机获取一个节点读取数据。
```

9.2 Mycat全局表配置

9.2.1 创建测试表

9.2.2 设置全局表策略

9.2.3 重启mycat

9.2.4 登陆mycat录入数据

9.2.5 查看各分片数据部署情况

全局表在每个分片上的数据都是一致的

10.Mycat分布式架构处理“多表查询”解决方案–ER（关系）分片

10.1 简介

10.2 ER分片配置

10.2.1 创建测试表

Bash | Copy

10.2.2 修改配置文件，定制分片策略

① 修改mycat配置文件 schema.xml

Bash | Copy

② 修改rule.xml对应的分片规则

Bash | Copy

10.2.3 重启mycat

Bash | Copy

10.2.4 登陆mycat录入数据

Bash | Copy

10.2.5 查看各分片数据部署情况

Bash | Copy

Mycat%E2%88%9A%20%7C%201.%E6%95%B0%E6%8D%AE%E5%BA%93%E6%9E%B6%E6%9E%84%E7%9A%84%E6%BC%94%E5%8F%982.Mycat%E7%AE%80%E4%BB%8B1.%E9%81%B5%E5%AE%88