
MyCat

一、MyCat 简介

1 什么是 MyCat

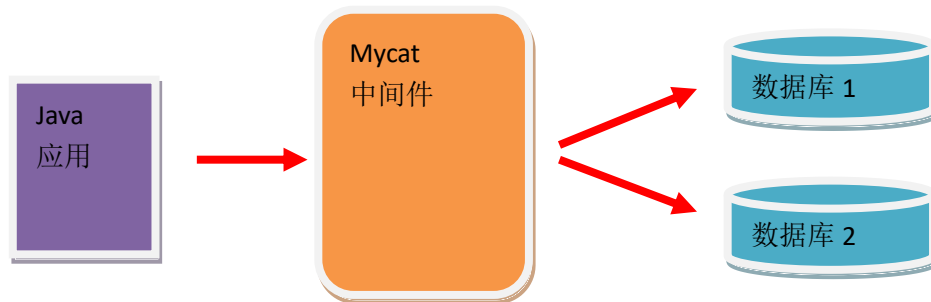
MyCat 是目前最流行的基于 java 语言编写的数据库中间件，是一个实现了 MySQL 协议的服务器，前端用户可以把它看作是一个数据库代理，用 MySQL 客户端工具和命令行访问，而其后端可以用 MySQL 原生协议与多个 MySQL 服务器通信，也可以用 JDBC 协议与大多数主流数据库服务器通信，其核心功能是分库分表。配合数据库的主从模式还可实现读写分离。

MyCat 是基于阿里开源的 Cobar 产品而研发，Cobar 的稳定性、可靠性、优秀的架构和性能以及众多成熟的使用案例使得 MyCat 变得非常的强大。

MyCat 发展到目前的版本，已经不是一个单纯的 MySQL 代理了，它的后端可以支持 MySQL、SQL Server、Oracle、DB2、PostgreSQL 等主流数据库，也支持 MongoDB 这种新型 NoSQL 方式的存储，未来还会支持更多类型的存储。而在最终用户看来，无论是那种存储方式，在 MyCat 里，都是一个传统的数据库表，支持标准的 SQL 语句进行数据的操作，这样一来，对前端业务系统来说，可以大幅降低开发难度，提升开发速度。

MyCat 官网：<http://www.mycat.io/>

2 使用 Mycat 后的结构图



3 使用 Mycat 的优势

3.1 数据量级

单一的 MySQL 其数据存储量级和操作量级有限。

MyCat 可以管理若干 MySQL 数据库,同时实现数据的存储和操作。

3.2 开源性质

Mycat 是 java 编写的中间件. 开源,免费.

有非常多的人和组织对 Mycat 实行开发,维护,管理,更新.

Mycat 版本提升较快,可以跟随环境发展.如果有问题,可以快速解决.

Mycat 有开源网站和开源社区.且有官方发布的电子书籍.

Mycat 是阿里原应用 corba 转型而来的.

3.3 市场应用

2015 年左右,Mycat 在互联网应用中占比非常高.

二、 MyCat 中的概念

1 切分

逻辑上的切分. 在物理层面,是使用多库[database],多表[table]实现的切分.

1.1 纵向切分/垂直切分

就是把原本存储于一个库的数据存储到多个库上。

由于对数据库的读写都是对同一个库进行操作,所以单库并不能解决大规模并发写入的问题。

例如,我们会建立定义数据库 workDB、商品数据库 payDB、用户数据库 userDB、日志数据库 logDB 等,分别用于存储项目数据定义表、商品定义表、用户数据表、日志数据表等。

优点

1) 减少增量数据写入时的锁对查询的影响。

2) 由于单表数量下降,常见的查询操作由于减少了需要扫描的记录,使得单表单次查询所需的检索行数变少,减少了磁盘 IO,时延变短。

缺点: 无法解决单表数据量太大的问题。

1.2 横向切分/水平切分

把原本存储于一个表的数据分块存储到多个表上。当一个表中的数据量过大时,我们可以把该表的数据按照某种规则,进行划分,然后存储到多个结构相同的表,和不同的库上。

例如,我们 userDB 中的 userTable 中数据量很大,那么可以把 userDB 切分为结构相同的多个 userDB: part0DB、part1DB 等,再将 userDB 上的 userTable,切分为很多 userTable: userTable0、userTable1 等,然后将这些表按照一定的规则存储到多个 userDB 上。

优点

1) 单表的并发能力提高了,磁盘 I/O 性能也提高了。

2) 如果出现高并发的话,总表可以根据不同的查询,将并发压力分到不同的小表里面。

缺点: 无法实现表连接查询。

2 逻辑库-Schema

Mycat 中定义的 database.是逻辑上存在的.但是物理上是不存在的.
主要是针对纵向切分提供的概念.

3 逻辑表-table

Mycat 中定义的 table.是逻辑上存在,物理上是不存在的.
主要是针对横向切分提供的概念.

4 默认端口

MySQL 默认端口是 3306

Mycat 默认端口是 8066

tomcat 默认端口是 8080

Oracle 默认端口是 1521

nginx 默认端口是 80

http 协议默认端口 80

redis 默认端口 6379

5 数据主机 - dataHost

物理 MySQL 存放的主机地址.可以使用主机名,IP,域名定义.

6 数据节点 - dataNode

配置物理的 database. 数据保存的物理节点.就是 database.

7 分片规则

当控制数据的时候,如何访问物理 database 和 table.

就是访问 dataHost 和 dataNode 的算法.

在 Mycat 处理具体的数据 CRUD 的时候,如何访问 dataHost 和 dataNode 的算法.如:哈希算法,crc32 算法等.

三、 MyCat 的使用

1 读写分离

原理: 需要搭建主从模式, 让主数据库 (master) 处理事务性增、改、删操作 (INSERT、UPDATE、DELETE), 而从数据库 (slave) 处理 SELECT 查询操作。

Mycat 配合数据库本身的复制功能, 可以解决读写分离的问题。

2 主从备份概念

什么是主从备份: 就是一种主备模式的数据库应用.

主库(Master)数据与备库(Slave)数据完全一致.

实现数据的多重备份, 保证数据的安全.

可以在 Master[InnoDB]和 Slave[MyISAM]中使用不同的数据库引擎,实现读写的分离

2.1 MySQL 5.5, 5.6 版本后本身支持主从备份

在老旧版本的 MySQL 数据库系统中,不支持主从备份,需要安装额外的 RPM 包.

如果需要安装 RPM,只能在一个位置节点安装.

2.2 主从备份目的

2.2.1 实现主备模式

保证数据的安全. 尽量避免数据丢失的可能.

2.2.2 实现读写分离

使用不同的数据库引擎,实现读写分离.提高所有的操作效率.

InnoDB 使用 DML 语法操作. MyISAM 使用 DQL 语法操作.

2.3 主从备份效果

2.3.1 主库操作同步到备库

所有对 Master 的操作,都会同步到 Slave 中.

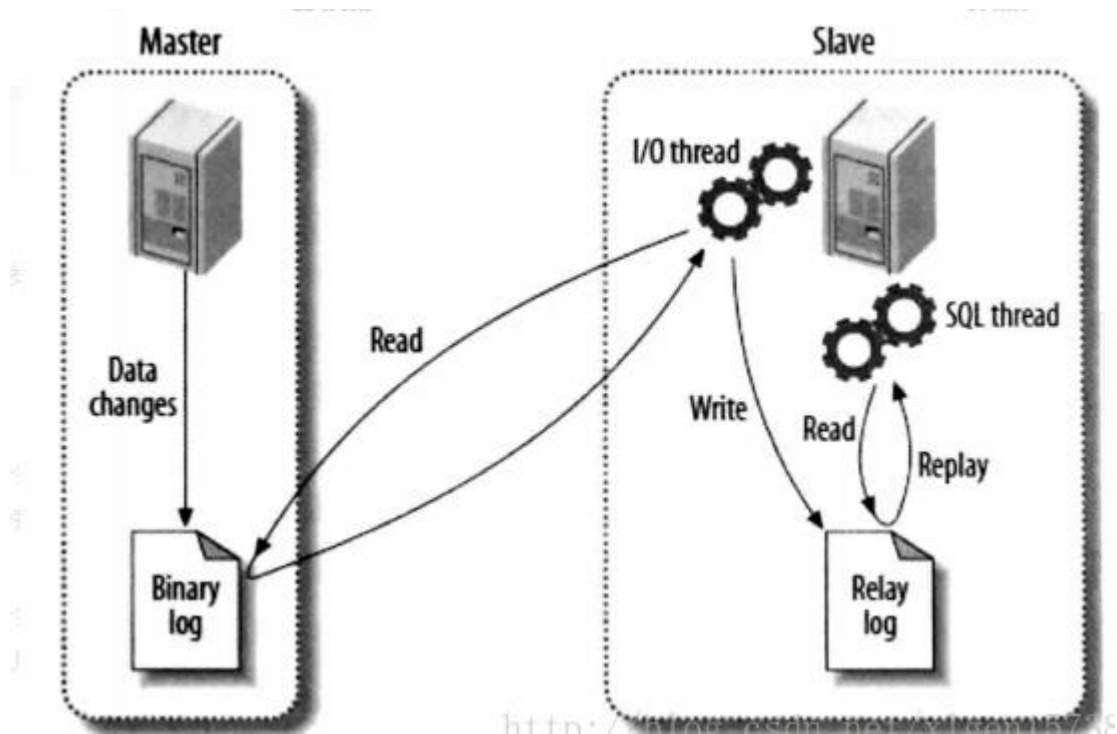
*如果 Master 和 Slave 天生上环境不同,那么对 Master 的操作,可能会在 Slave 中出现错误
如: 在创建主从模式之前,Master 有 database : db1, db2, db3. Slave 有 database: db1, db2.*

创建主从模式.现在的情况 Master 和 Slave 天生不同.

主从模式创建成功后,在 Master 中 drop database db3. Slave 中抛出数据库 SQL 异常.后续所有的命令不能同步.

一旦出现错误. 只能重新实现主从模式.

2.4 主从模式下的逻辑图



3 MySQL 的主从模式搭建

3.1 安装 MySQL

已安装

主库: 192.168.70.148

从库: 192.168.70.149

3.2 主从备份配置

3.3 Master[主库]配置

3.3.1 修改 Master 配置文件

路径: /etc/my.cnf

命令: vim /etc/my.cnf

3.3.2 server_id

本环境中 server_id 是 1

MySQL 服务唯一标识

配置要求:

server_id 任意配置,只要是数字即可

server_id Master 唯一标识数字必须小于 Slave 唯一标识数字.

3.3.3 log_bin

本环境中 log_bin 值 : master_log

开启日志功能以及日志文件命名,log_bin=master_log

变量的值就是日志文件名称.是日志文件名称的主体.

MySQL 数据库自动增加文件名后缀和文件类型.

3.3.4 重启 MySQL

service mysqld restart

3.3.5 配置 Master

3.3.5.1 访问 MySQL

mysql -uusername -ppassword

3.3.5.2 创建用户

在 MySQL 数据库中,为不存在的用户授权,就是同步创建用户并授权.

此用户是从库访问主库使用的用户

ip 地址不能写为%. 因为主从备份中,当前创建的用户,是给从库 Slave 访问主库 Master 使用的.用户必须有指定的访问地址.不能是通用地址.

```
grant all privileges on *.* to 'username'@'ip' identified by 'password' with grant option;
```

```
flush privileges;
```

```
grant all privileges on *.* to 'myslave'@'192.168.70.149' identified by 'myslave' with grant option;
flush privileges;
```

3.3.5.3 查看用户

```
use mysql;
```

```
select host, name from user;
```

```
mysql> select host,user from user;
+-----+-----+
| host          | user  |
+-----+-----+
| 127.0.0.1     | root  |
| 192.168.70.149 | myslave |
| ::1          | root  |
| localhost     | root  |
| localhost     | root  |
| localhost.localdomain | root  |
| localhost.localdomain | root  |
+-----+-----+
```

3.3.5.4 查看 Master 信息

show master status;

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| master_log.000001 | 428      |              |                  |                   |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

3.3.5.5 关闭防火墙或在防火墙中开放 3306 端口

3.4 Slave[从库]配置

3.4.1 修改 Slave 配置文件

/etc/my.cnf

3.4.2 server_id

唯一标识, 本环境中配置为 :2

3.4.3 重启 MySQL 服务

service mysqld restart

3.4.4 配置 Slave

3.4.4.1 访问 mysql

mysql -uusername -ppassword

3.4.4.2 停止 Slave 功能

stop slave

3.4.4.3 配置主库信息

需要修改的数据是依据 Master 信息修改的. ip 是 Master 所在物理机 IP. 用户名和密码是 Master 提供的 Slave 访问用户名和密码. 日志文件是在 Master 中查看的主库信息提供的. 在 Master 中使用命令 show master status 查看日志文件名称.

change master to master_host='ip', master_user='username', master_password='password', master_log_file='log_file_name';

change	master	to
master_host='192.168.70.148',master_user='myslave',master_password='myslave',master_log_file='master_log.000001';		

3.4.4.4 启动 Slave 功能

start slave;

3.4.4.5 查看 Slave 配置

show slave status \G;

```
mysql> show slave status \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.70.148
Master_User: myslave
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: master-log.000001
Read_Master_Log_Pos: 427
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 591
Relay_Master_Log_File: master-log.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 427
Relay_Log_Space: 765
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0 最后一次错误的 IO 请求编号
```



```
Last_IO_Error:
Last_SQL_Errno: 0 最后一次错误的执行 SQL 命令编号.
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_UUID: 9ee988ac-8751-11e7-8a95-000c2953ac06
Master_Info_File: /var/lib/mysql/master.info
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O
thread to update it
Master_Retry_Count: 86400
Master_Bind:
Last_IO_Error_Timestamp:
Last_SQL_Error_Timestamp:
Master_SSL_Crl:
Master_SSL_Crlpath:
Retrieved_Gtid_Set:
Executed_Gtid_Set:
Auto_Position: 0
1 row in set (0.00 sec)
```

3.5 测试主从

新建库

```
create database demo1 default character set utf8;
```

新建表

```
CREATE TABLE `t_users` (
  `id` int(11) NOT NULL,
  `name` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

添加数据

```
insert into users values(1,'admin')
```

4 安装 MyCat

4.1 安装环境

192.168.70.150

4.2 需要配置 JDK

已安装

4.3 在主数据库和从数据库都需要完成

4.3.1 放开 3306 端口

4.3.2 保证 root 用户可以被 mycat 访问

在 Mycat 中通过 Master 数据库的 root 用户访问 Master 数据库.

```
grant all privileges on *.* to 'root'@'%' identified by 'root' with grant option;
```

```
flush privileges;
```

4.4 解压上传的 Mycat 压缩包

```
tar -zxvf Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz
```

4.5 将解压后的文件夹复制到 /usr/local/mycat

4.6 MyCat 目录介绍

bin 目录里是启动脚本

conf 目录里是配置文件

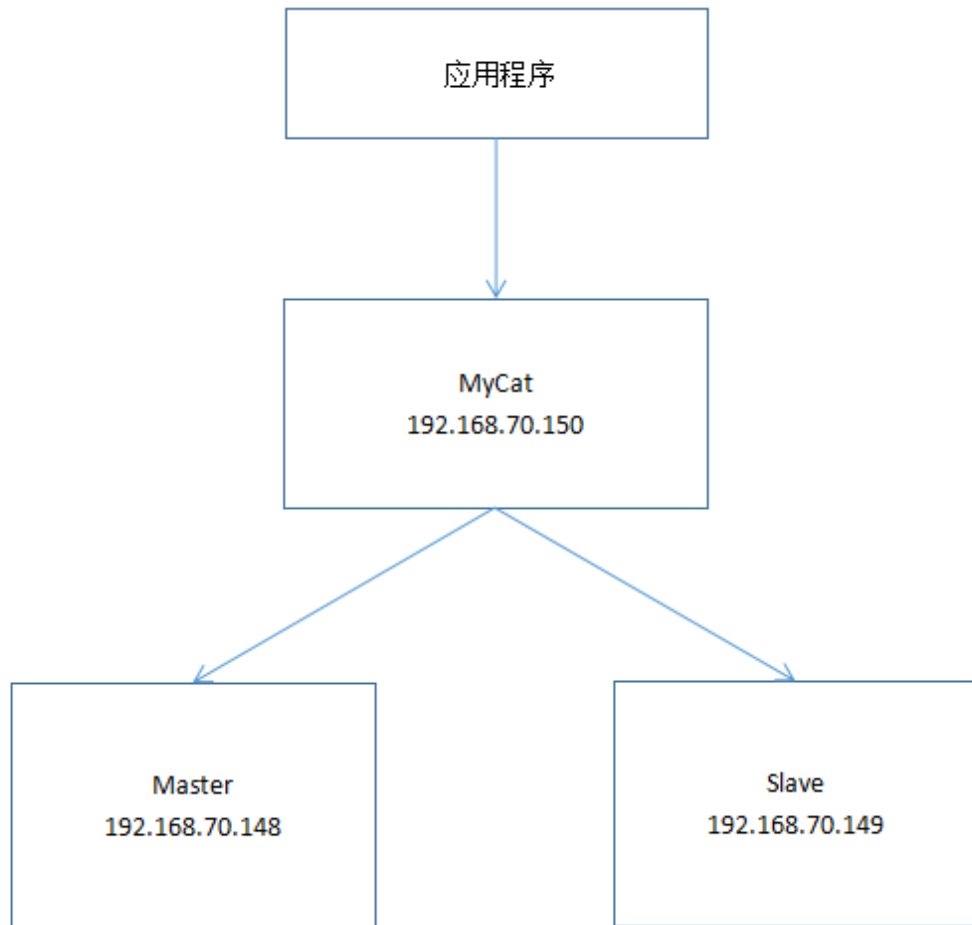
catlet 为 Mycat 的一个扩展功能

lib 目录里是 Mycat 和它的依赖 jar

logs 目录里是 console.log 用来保存控制台日志，和 mycat.log 用来保存 mycat 的 log4j 日志

5 MyCat 配置文件

Mycat 的架构其实很好理解，Mycat 是代理，Mycat 后面就是物理数据库。和 Web 服务器的 Nginx 类似。对于使用者来说，访问的都是 Mycat，不会接触到后端的数据库。我们现在做一个主从、读写分离。结构如下图：



Mycat 的配置文件都在 conf 目录里面，这里介绍几个常用的文件

文件	说明
server.xml	MyCat 的配置文件，设置账号、参数等
schema.xml	MyCat 对应的物理数据库和数据库表的配置
rule.xml	MyCat 分片（分库分表）规则

5.1 server.xml

常见修改内容:

```
<property name="serverPort">8066</property> <!-- Mycat 服务端口号 -->
<property name="managerPort">9066</property><!-- Mycat 管理端口号 -->
<user name="root"><!-- mycat 用户名 -->
    <property name="password">密码</property>
    <property name="schemas">用户可访问逻辑库名</property>

    <!-- 表级 DML 权限设置 -->
    <!-- 不检查 SQL 语法结果 -->
    <privileges check="false">
        <schema name="逻辑库名" dml="0110" >
```

```

        <table name="逻辑表名" dml="0000"></table>
        <table name="tb02" dml="1111"></table>

    </schema>

</privileges>
-->

</user>

<user name="user"><!-- 其他用户名 -->
    <property name="password">密码</property>
    <property name="schemas">可访问逻辑库名</property>
    <property name="readOnly">是否只读</property>

</user>

```

5.1.1 配置 Mycat 服务信息

如: Mycat 中的用户,用户可以访问的逻辑库,可以访问的逻辑表,服务的端口号等

user	用户配置节点
--name	登录的用户名, 也就是连接 Mycat 的用户名
--password	登录的密码, 也就是连接 Mycat 的密码
--schemas	逻辑库名, 这里会和 schema.xml 中的配置关联, 多个用逗号分开, 例如需要这个用户管理两个数据库 db1,db2, 则配置 db1,db2
--privileges	配置用户针对表的增删改查的权限

默认配置了一个账号 root 密码也是 123456,针对数据库 TESTDB,读写权限都有, 没有针对表做任何特殊的权限。

5.1.2 配置权限

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

dml 权限顺序为: insert(新增),update(修改),select(查询),delete(删除),0000--> 1111,0 为禁止权限, 1 为开启权限。

5.2 schema.xml

schema.xml 是最主要的配置文件, 首先看默认的配置文

```

<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
    <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">
        <table name="t_user" dataNode="dn1,dn2,dn3" rule="crc32slot" />
    </schema>
</mycat:schema>

```

```
</schema>
<dataNode name="dn1" dataHost="localhost1" database="db1" />
<dataNode name="dn2" dataHost="localhost1" database="db2" />
<dataNode name="dn3" dataHost="localhost1" database="db3" />
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
            writeType="0"      dbType="mysql"      dbDriver="native"
switchType="1"  slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <writeHost host="hostM1" url="localhost:3306" user="root"
                password="root">
    <readHost   host="hostS2"   url="192.168.1.200:3306"   user="root"
password="root" />
    </writeHost>
</dataHost>
</mycat:schema>
```

5.2.1 用于定义逻辑库和逻辑表的配置文件

在配置文件中可以定义读写分离,逻辑库,逻辑表,dataHost,dataNode 等信息.

schema	配置逻辑库，name 与 server.xml 中 schema 对应
dataNode	定义数据节点的标签，也就是分库相关配置
dataHost	物理数据库，真正存储数据的数据库

5.2.2 节点与属性介绍

5.2.2.1 标签 schema

配置逻辑库的标签

5.2.2.1.1 属性 name

逻辑库名称

5.2.2.1.2 属性 checkSQLschema

是否检测 SQL 语法中的 schema 信息.
如: Mycat 逻辑库名称 A, dataNode 名称 B
SQL : select * from A.table;
checkSQLschema 值是 true, Mycat 发送到数据库的 SQL 是 select * from table;
checkSQLschema 只是 false,Mycat 发送的数据库的 SQL 是 select * from A.table;

5.2.2.1.3 sqlMaxLimit

Mycat 在执行 SQL 的时候,如果 SQL 语句中没有 limit 子句.自动增加 limit 子句. 避免一次性得到过多的数据,影响效率. limit 子句的限制数量默认配置为 100.如果 SQL 中有具体的 limit

子句,当前属性失效.

SQL : select * from table . mycat 解析后: select * from table limit 100

SQL : select * from table limit 10 . mycat 不做任何操作修改.

5.2.2.2 标签 table

定义逻辑表的标签

5.2.2.2.1 属性 name

逻辑表名

5.2.2.2.2 属性 dataNode

数据节点名称. 即物理数据库中的 database 名称.多个名称使用逗号分隔.

5.2.2.2.3 属性 rule

分片规则名称.具体的规则名称参考 rule.xml 配置文件.

5.2.2.3 标签 dataNode

定义数据节点的标签

5.2.2.3.1 属性 name

数据节点名称, 是定义的逻辑名称,对应具体的物理数据库 database

5.2.2.3.2 属性 dataHost

引用 dataHost 标签的 name 值,代表使用的物理数据库所在位置和配置信息.

5.2.2.3.3 属性 database

在 dataHost 物理机中,具体的物理数据库 database 名称.

5.2.2.4 dataHost 标签

定义数据主机的标签

5.2.2.4.1 属性 name

定义逻辑上的数据主机名称

5.2.2.4.2 属性 maxCon/minCon

最大连接数, max connections

最小连接数, min connections

5.2.2.4.3 属性 dbType

数据库类型 : mysql 数据库

5.2.2.4.4 属性 dbDriver

数据库驱动类型, native, 使用 mycat 提供的本地驱动.

5.2.2.5 dataHost 子标签 writeHost

写数据的数据库定义标签. 实现读写分离操作.

5.2.2.5.1 属性 host

数据库命名

5.2.2.5.2 属性 url

数据库访问路径

5.2.2.5.3 属性 user

数据库访问用户名

5.2.2.5.4 属性 password

访问用户密码

5.2.2.6 writeHost 子标签 readHost

5.2.2.6.1 属性 host

数据库命名

5.2.2.6.2 属性 url

数据库访问路径

5.2.2.6.3 属性 user

数据库访问用户名

5.2.2.6.4 属性 password

5.2.3 rule.xml

用于定义分片规则的配置文件.

mycat 默认的分片规则: 以 500 万为单位, 实现分片规则.

逻辑库 A 对应 dataNode - db1 和 db2. 1-500 万保存在 db1 中, 500 万零 1 到 1000 万保存在 db2 中, 1000 万零 1 到 1500 万保存在 db1 中. 依次类推.

```
<tableRule name="rule1">
    <rule>
        <columns>id</columns>
        <algorithm>func1</algorithm>
```

```
</rule>
</tableRule>
```

5.2.3.1 tableRule

name	属性指定唯一的名字，用于标识不同的分片规则。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分片算法
columns	指定要拆分的列名字
algorithm	使用 function 标签中的 name 属性。连接表规则 and 具体分片算法。 table 标签内使用。让逻辑表使用这个规则进行分片

5.2.3.2 function

```
<function name="func1" class="io.mycat.route.function.PartitionByLong">
    <property name="partitionCount">8</property>
    <property name="partitionLength">128</property>
</function>
```

name	指定算法的名字
class	制定分片算法具体的类名字
property	为具体算法需要用到的一些属性

6 实现读写分离

6.1 配置读写分离

6.1.1 Schema.xml

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

    <schema name="suibian" checkSQLschema="false" sqlMaxLimit="100">
        <table name="t_users" dataNode="dn1" />
    </schema>
    <dataNode name="dn1" dataHost="localhost1" database="demo1" />
    <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
        writeType="0" dbType="mysql" dbDriver="native" switchType="1"
slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <!-- can have multi write hosts -->
```



```
<writeHost host="hostM1" url="192.168.70.148:3306" user="root"
          password="root">
  <!-- can have multi read hosts -->
  <readHost host="hostS2" url="192.168.70.149:3306" user="root"
password="root" />
</writeHost>
</dataHost>
</mycat:schema>
```

6.1.2 Server.xml

```
<user name="root">
  <property name="password">123456</property>
  <property name="schemas">suibian</property>

  <!-- 表级 DML 权限设置 -->
  <!--
  <privileges check="false">
    <schema name="TESTDB" dml="0110" >
      <table name="tb01" dml="0000"></table>
      <table name="tb02" dml="1111"></table>
    </schema>
  </privileges>
  -->
</user>

<user name="user">
  <property name="password">user</property>
  <property name="schemas">suibian</property>
  <property name="readOnly">true</property>
</user>
```

6.1.3 测试读写分离

6.1.3.1 启动 Mycat 命令

bin/mycat start

6.1.3.2 停止命令

bin/mycat stop

6.1.3.3 重启命令

bin/mycat restart

6.1.3.4 查看 MyCat 状态

bin/mycat status

6.1.3.5 访问方式

可以使用命令行访问或客户端软件访问。

6.1.3.6 命令行访问方式

mysql -u 用户名 -p 密码 -hmycat 主机 IP -P8066

链接成功后,可以当做 MySQL 数据库使用。

访问约束

6.1.3.7 查看 Mycat 日志

logs/wrapper.log

日志中记录的是所有的 mycat 操作。查看的时候主要看异常信息 caused by 信息

6.1.3.8 balance

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

balance=" 1", 全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡

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

balance=" 3", 所有读请求随机的分发到 writeHost 对应的 readhost 执行,writerHost 不负担读压力

7 MyCat 分库

7.1 分片规则

7.1.1 auto-sharding-long 范围约定

以 500 万为单位,实现分片规则。

逻辑库 A 对应 dataNode - db1 和 db2. 1-500 万保存在 db1 中, 500 万零 1 到 1000 万保存在 db2 中,1000 万零 1 到 1500 万保存在 db1 中.依次类推。

7.1.2 crc32slot 规则

在 CRUD 操作时,根据具体数据的 crc32 算法计算,数据应该保存在哪一个 dataNode 中

7.2 配置分片规则需要注意的地方

1) <columns>id</columns>中推荐配置主键列

2) 所有的 tableRule 只能使用一次。如果需要为多个表配置相同的分片规则,那么需要在此重新定义该规则。

3) 在 crc32Slot 算法中的分片数量一旦给定, MyCat 会将该分片数量和 slot 的取值范围

保存到文件中。在次修改分片数量时是不会生效的，需要将该文件删除。文件位置位于 **conf** 目录中的 **ruledata** 目录中。

7.3 配置分库

7.3.1 需求：

- 1) 在 master 中创建 3 个数据库
- 2) 在 MyCat 中配置分库

7.3.2 创建数据库

```
create database demo1 default character set utf8;
create database demo2 default character set utf8;
create database demo3 default character set utf8;
```

创建 t_users 表

```
CREATE TABLE `t_users` (
  `id` int(11) NOT NULL,
  `name` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

7.3.3 修改 Schema.xml

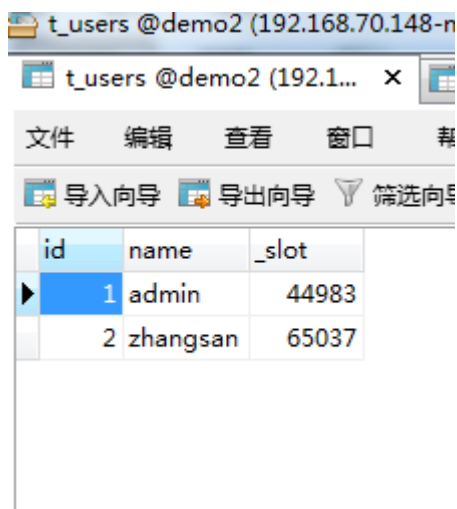
```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

  <schema name="suibian" checkSQLschema="false" sqlMaxLimit="100">
    <table name="t_users" dataNode="dn1,dn2,dn3" rule="crc32slot" />
  </schema>

  <dataNode name="dn1" dataHost="localhost1" database="demo1" />
  <dataNode name="dn2" dataHost="localhost1" database="demo2" />
  <dataNode name="dn3" dataHost="localhost1" database="demo3" />
  <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"
    writeType="0" dbType="mysql" dbDriver="native" switchType="1"
    slaveThreshold="100">
    <heartbeat>select user()/heartbeat>
    <!-- can have multi write hosts -->
    <writeHost host="hostM1" url="192.168.70.148:3306" user="root"
      password="root">
```

```
<!-- can have multi read hosts -->
<readHost    host="hostS2"    url="192.168.70.149:3306"    user="root"
password="root" />
</writeHost>
</dataHost>
</mycat:schema>
```

7.3.4 测试



The screenshot shows a MySQL client window titled 't_users @demo2 (192.168.70.148-n)'. The window has a menu bar with '文件', '编辑', '查看', '窗口', and '帮助'. Below the menu bar is a toolbar with icons for '导入向导', '导出向导', and '筛选向导'. The main area displays a table with three columns: 'id', 'name', and '_slot'. The table contains two rows of data.

id	name	_slot
1	admin	44983
2	zhangsan	65037

7.4 注意:

- 1) 使用 MyCat 实现分库时，先在 MyCat 中定义逻辑库与逻辑表，然后在 MyCat 的链接中执行创建表的命令必须要在 MyCat 中运行。因为 MyCat 在创建表时，会在表中添加一个新的列，列名为_slot。
- 2) 使用 MyCat 插入数据时，语句中必须要指定所有的列。即便是一个完全项插入也不允许省略列名。