基于Mycat的分库分表

2016年12月

目录

[1 背景 3](#_Toc470600244)

[2 Mycat是什么 3](#_Toc470600245)

[3 原理 4](#_Toc470600246)

[4 部署图 4](#_Toc470600247)

[5 定义介绍 5](#_Toc470600248)

[5.1 逻辑库（schema） 5](#_Toc470600249)

[5.2 逻辑表（table） 5](#_Toc470600250)

[5.3 节点主机（dataHost） 5](#_Toc470600251)

[5.4 分片节点（dataNode） 6](#_Toc470600252)

[6 配置文件 6](#_Toc470600253)

[6.1 Server.xml 6](#_Toc470600254)

[6.2 Schema.xml 7](#_Toc470600255)

[6.3 Rule.xml 9](#_Toc470600256)

[7 简单应用 10](#_Toc470600257)

[7.1 代理功能 11](#_Toc470600258)

[7.2 读写分离 13](#_Toc470600259)

[7.3 分库分表 15](#_Toc470600260)

[7.3.1 垂直切分 15](#_Toc470600261)

[7.3.2 水平切分 17](#_Toc470600262)

[7.4 高可用 19](#_Toc470600263)

# 背景

Mycat是一款数据库中间件，其前生为Cobar（阿里巴巴研发的关系型数据的分布式处理系统）。Cobar在使用过程中发现存在一些比较严重的漏洞，如：假死、高可用的BUG、阻塞、自动切换的缺陷等，经过Mycat发起人第一次改良，Mycat诞生，开源后，更多的人参与到开发中，最终Mycat发展成为一个由众多软件公司的实例架构师和资深开发人员维护的社区型开源软件。

Mycat是基于开源Cobar演变而来，并对Cobar的代码进行了彻底的重构，使用NIO重构了网络模块，并且优化了Buffer内核，增强了聚合，Join等基本特性，同事兼容绝大多数数据库成文通用的数据库中间件。1.4版本后，完全脱离Cobar内核，结合Mycat集群管理，自动扩容，智能优化，成为高性能的中间件。

# Mycat是什么

* **对 DBA 来说**

Mycat就是MySQL Server，而Mycat后面连接的才是MySQL Server，就如MySQL的存储引擎：InnoDB，MyISAM等，因此，Mycat本身不存储数据，数据是在后端的MySQL上存储的，数据的可靠性以及事务都是由MySQL保证，简单地说，Mycat可以理解为MySQL的辅助工具，提升MySQL的能力。

* **对 软件工程师 来说**

Mycat就是一个近似等于MySQL的数据库服务器，你可以用连接MySQL的方式去连接Mycat（端口不同），大多数情况下，可以使用熟悉的对象映射框架使用Mycat。

针对分片表，建议尽量使用基础的SQL语句，因为这样能达到最佳性能，特别是千万级数据的情况下。

* **对 架构师 来说**

Mycat是一个强大的数据库中间件，不仅仅可以用作读写分离、以及表的分表分库、容灾备份，而且可以用于多租户应用开发、云平台基础设施、让架构具备很强的适应性和灵活性。

* **在大数据时代，为什么选择数据库系统，而非Hadoop等？**

Hadoop这种系统，更适合的是千亿以上的数据规模，而千亿以下的数据规模，仍然是数据库系统领域的专长。

# 原理

Mycat的原理中最重要的一词是“拦截”，它拦截由客户端发送过来的SQL语句，并对SQL进行一系列的分析：如分片分析、路由分析、读写分离分析、缓存分析等，然后将此SQL发往后端的真实数据库。再将真实数据库返回的信息，进行适当处理后，最终返回给用户。

# 部署图



# 定义介绍

## 逻辑库（schema）

逻辑库，由Mycat虚拟出来的数据库，在物理数据库中并不一定存在。逻辑库是由一个或者多个物理数据库中的schema构成。对业务开发人员来讲，可以将逻辑库看作实际的数据库。

## 逻辑表（table）

既然有逻辑库，那举就会有逻辑表，分布式数据库中，对应用来该，读写的表就是逻辑表。逻辑表，可以理解为被切分在多个库中的数据总和，也可以是单个数据库的一张表数据。

* 分片表

分片表：是指那些原有很大数据，被切分到多个数据库中的表。每个分片都有这张表的部分数据，所有的分片构成了完成的数据。

* 非分片表

在实际的应用数据库中，并非所有的表数据都很大，这些没有被分片的表，统称为非分片表。

* 全局表

业务系统中，往往还存在大量的类似字典表的表，这些表和业务表有着密切的关系；这些表有：数据很少变动，数据量不大的特点，如果业务表存在跨库分片，字典表和业务表的关联关系就需要靠数据冗余来解决，即：每个库中都存在全量字典表。这类表称为全局表。

## 节点主机（dataHost）

数据被切分后，每个分片节点都不一定会独占一台主机，一台服务器上可以存在多个分片数据库，这样一个或者多个分片所在的主机就被称为节点主机，节点主机下，可指定读写节点。

## 分片节点（dataNode）

一个大表被切分后，被分到不同分片数据上，每个分片所在的数据库就是分片节点。

# 配置文件

## Server.xml

Server.xml文件主要设置mycat的参数，包括端口设置，用户设置，访问权限设置，以及事务控制等。

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mycat:server SYSTEM "server.dtd">  
<mycat:server xmlns:mycat="http://io.mycat/">  
 <system>  
 <property name="useSqlStat">0</property>  
 <property name="useGlobleTableCheck">0</property>  
 <property name="sequnceHandlerType">2</property>  
 <property name="processorBufferPoolType">0</property>  
 <property name="serverPort">8066</property>  
 <property name="managerPort">9066</property>  
 <property name="idleTimeout">300000</property>  
 <property name="bindIp">0.0.0.0</property>  
 <property name="frontWriteQueueSize">4096</property>  
 <property name="processors">32</property>  
 <property name="handleDistributedTransactions">0</property>  
 <property name="useOffHeapForMerge">1</property>  
 <property name="memoryPageSize">1m</property>  
 <property name="spillsFileBufferSize">1k</property>  
 <property name="useStreamOutput">0</property>  
 <property name="systemReserveMemorySize">384m</property>  
 <property name="useZKSwitch">true</property>  
 </system>  
  
 <user name="root">  
 <property name="password">root</property>  
 <property name="schemas">TESTDB</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>  
</mycat:server>

## Schema.xml

Schema.xml主要用于定义节点主机（dataHost）、数据节点（dataNode）、虚拟库（schema），以及虚拟表（table）。

<?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="dn">

<table />  
 </schema>  
 <dataNode name="dn" dataHost="localhost1" database="mycatdb" />  
 <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  
 writeType="0" dbType="mysql" dbDriver="native" switchType="1">  
 <heartbeat>select user()</heartbeat>  
 <!-- can have multi write hosts -->  
 <writeHost host="hostM1" url="localhost:3306" user="root"  
 password="root">  
 <!-- can have multi read hosts -->  
 <readHost host="hostS2" url="localhost:3306" user="root" password="root" />  
 </writeHost>  
 </dataHost>  
</mycat:schema>

**schema标签**：定义逻辑库

|  |  |
| --- | --- |
| **属性名称** | **说明** |
| name | 逻辑库的名称 |
| checkSQLschema | 当执行select \* from TESTDB.order时，mycat会自动将TESTDB去掉，否则，会因TESTDB在实际库中不存在而报错。 |
| sqlMaxLimit | 当sql没指定limit的时候，默认返回100条数据，避免因数据过大，导致数据输出过多。 |
| dataNode | 指定分片节点，可指定多个 |

子标签table，指定分片的表或者全局表，未声明的表为非分片表。

**dataNode标签**：定义分片节点。

|  |  |
| --- | --- |
| **属性名称** | **说明** |
| name | 分片节点的名称 |
| dataHost | 分片主机的名称 |
| database | 实际数据库库称 |
|  |  |

**dataHost标签**：定义分片主机

|  |  |
| --- | --- |
| **属性名称** | **说明** |
| name | 分片主机的名称 |
| maxCon | 每个读写实例的最大连接数 |
| minCon | 每个读写实例的最小连接数 |
| balance | 负载均衡类型  0：不开启读写分离  1：除了在用的写节点外的所有节点参与读请求的分发  2：所有的读都随机分发到writehost和readhost  3：读请求在writehost对应的readhost上随机分发 |
| writeType | 写请求的分发类型  -1：不自动切换  1：默认值，轮训，第一个挂了以后，切换到第二个，以此类推，重启后不恢复到第一个writeHost  2：所有的writeHost都参与读请求随机分发 |
| dbType | 数据库类型，如：mongodb、oracle、spark等 |
| dbDriver | 指定连接后端数据库使用的Driver，目前可选的值有native和JDBC，native是二进制的mysql协议，可使用mysql和maridb。其他类型的数据库需要使用JDBC驱动来支持，此时，该值为具体的具体的Driver类名，如：com.mysql.jdbc.Driver |

**heartbeat标签**：检测节点是否正常

mysql：select user();

oracle：select \* from dual;

**writeHost标签**：定义写节点

|  |  |
| --- | --- |
| **属性名称** | **说明** |
| host | 名称，用于标识不同的实例，writeHost一般使用\*M1 |
| url | 后端实例连接地址，如果使用native的dbDriver，一般为ip:port的形式，如果使用JDBC，则需要jdbc连接地址，如：jdbc:mysql://ip:port/ |
| user | 后端实例的登陆用户名 |
| password | 后端实例的登陆密码 |
|  |  |
|  |  |

**readHost标签**：定义读节点

|  |  |
| --- | --- |
| **属性名称** | **说明** |
| host | 名称，用于标识不同的实例，writeHost一般使用\*S1 |
|  | 其他属性参考writeHost标签 |
|  |  |

## Rule.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mycat:rule SYSTEM "rule.dtd">  
<mycat:rule xmlns:mycat="http://io.mycat/">  
 <tableRule name="sharding-by-intfile">  
 <rule>  
 <columns>sharding\_id</columns>  
 <algorithm>hash-int</algorithm>  
 </rule>  
 </tableRule>  
 <tableRule name="auto-sharding-long">  
 <rule>  
 <columns>id</columns>  
 <algorithm>rang-long</algorithm>  
 </rule>  
 </tableRule>  
  
 <function name="hash-int"  
 class="io.mycat.route.function.PartitionByFileMap">  
 <property name="mapFile">partition-hash-int.txt</property>  
 </function>  
 <function name="rang-long"  
 class="io.mycat.route.function.AutoPartitionByLong">  
 <property name="mapFile">autopartition-long.txt</property>  
 </function>  
</mycat:rule>

**tableRule标签**：定义一个规则

|  |  |
| --- | --- |
| **属性** | **说明** |
| name | 规则名称 |

**columns标签**：字段的名

**algorithm标签**：算法名称

**function标签**：定义算法名称

简单介绍两种算法：

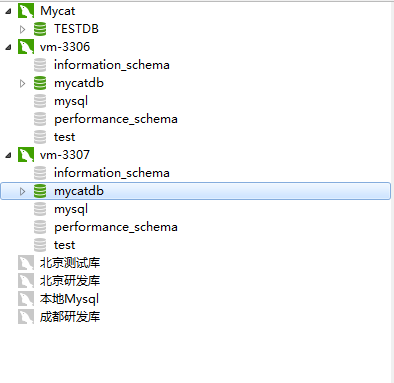
sharding-by-intfile：分片枚举，比如当字段值为1时，存入一个数据库中，当为0时，存入另一个数据库中

auto-sharding-long：主键范围，按照范围进行自动分片，比如0-200的数据在分片节点1,200-400的数据在分片节点2，依次类推，根据数据库服务器的性能，评估数据存储量，此种方式，每个分片表一个独立的自增长ID机制，确保记录的连续性。conf/ autopartition-long.txt中定义了分段范围与分片ID的关系。

# 简单应用

为了能够直观的了解Mycat的作用，下面做几个简单的实验：

前提：在虚拟机（192.168.197.130）上装两个mysql实例，端口分别为3306、3307，都创建测试数据库mycatdb，部署Mycat并启动，使用navicat连接数据库实例和mycat，效果如下：

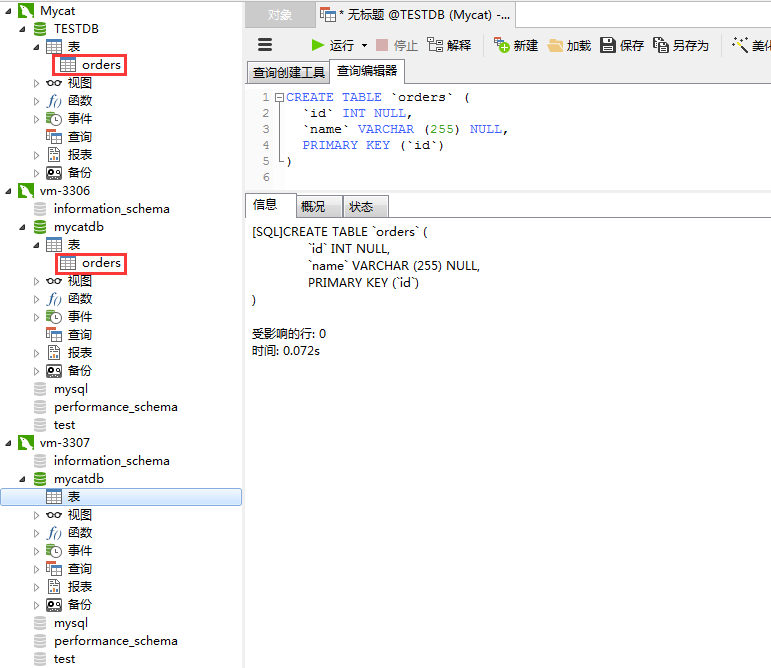


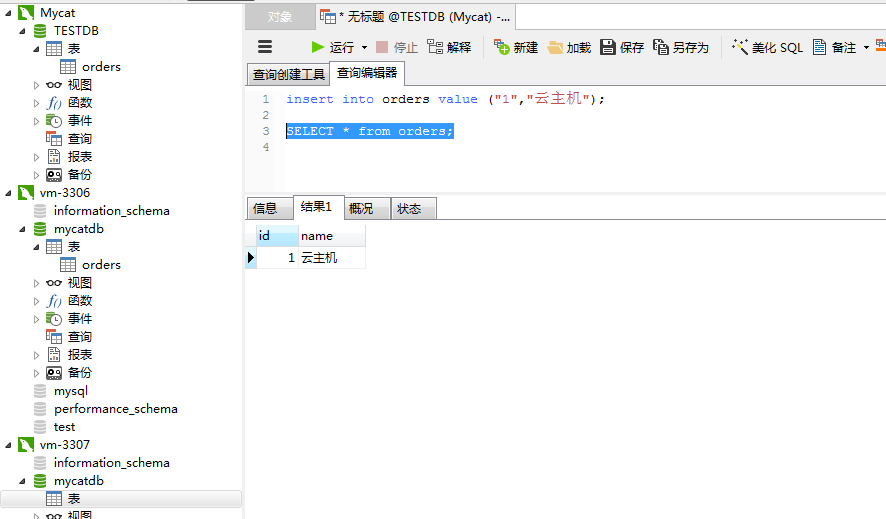
## 代理功能

场景：mycat仅代理3306端口服务

Schema.xml配置：

<?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="v\_host1" database="mycatdb" />  
 <dataHost name="v\_host1" maxCon="1000" minCon="10" balance="0"  
 writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <!-- can have multi write hosts -->  
 <writeHost host="hostM1" url="localhost:3306" user="root" password="root">  
 <!-- can have multi read hosts -->  
 <readHost host="hostS2" url="localhost:3306" user="root" password="root" />  
 </writeHost>  
 </dataHost>  
</mycat:schema>



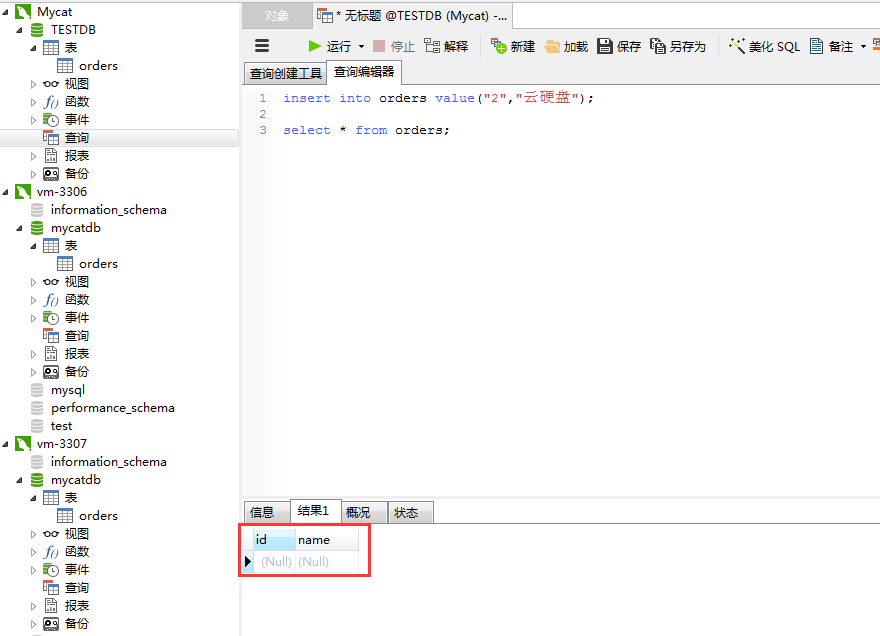


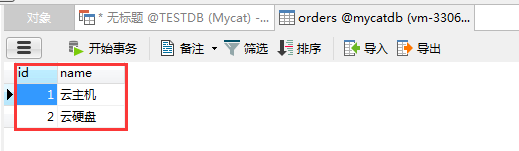
## 读写分离

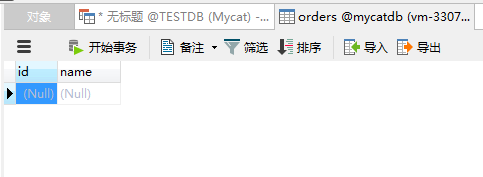
场景：读请求被分发到3307上，写请求被分发到3306上

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="dn1">  
 </schema>  
 <dataNode name="dn1" dataHost="v\_host1" database="mycatdb" />  
 <dataHost name="v\_host1" maxCon="1000" minCon="10" balance="1"  
 writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <!-- can have multi write hosts -->  
 <writeHost host="hostM1" url="localhost:3306" user="root" password="root">  
 <!-- can have multi read hosts -->  
 <readHost host="hostS2" url="localhost:3307" user="root" password="root" />  
 </writeHost>  
 </dataHost>  
</mycat:schema>







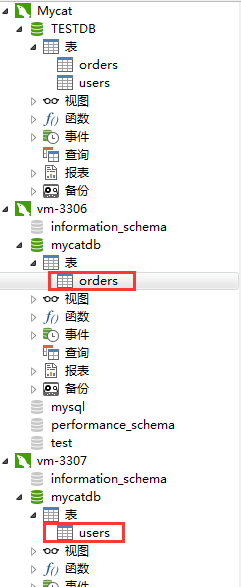
## 分库分表

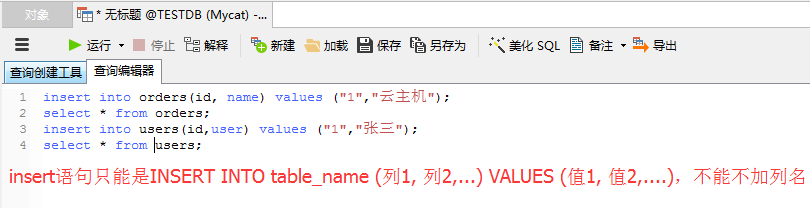
### 垂直切分

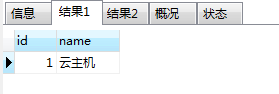
场景：同一应用的两张表，被分别存放到两个库中，orders表存放于3306库中，users存放于3307库中，可通过mycat读写两张表，如访问一个数据库；

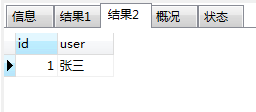
schema.xml：

<?xml version="1.0"?>  
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">  
<mycat:schema xmlns:mycat="http://io.mycat/">  
  
 <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">  
 <table name="orders" primaryKey="ID" dataNode="dn1"/>  
 <table name="users" primaryKey="ID" dataNode="dn2"/>  
 </schema>  
   
 <dataNode name="dn1" dataHost="v\_host1" database="mycatdb" />  
 <dataNode name="dn2" dataHost="v\_host2" database="mycatdb" />  
   
 <dataHost name="v\_host1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="localhost:3306" user="root" password="root">  
 <readHost host="hostS2" url="localhost:3306" user="root" password="root" />  
 </writeHost>   
 </dataHost>  
  
 <dataHost name="v\_host2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="localhost:3307" user="root" password="root">  
 <readHost host="hostS2" url="localhost:3307" user="root" password="root" />  
 </writeHost>   
 </dataHost>  
</mycat:schema>

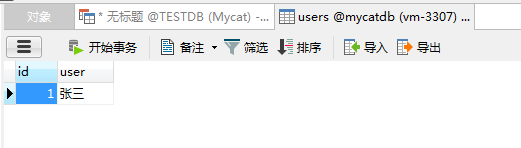












### 水平切分

场景：一张表的数据，被切分到两个库中，如：orders表，当ID的值在100以内，则数据存放于3306，当ID的值在100-200之间，则存放于3307上。

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">  
 <table name="orders" primaryKey="ID" autoIncrement="true" dataNode="dn1,dn2" rule="auto-sharding-long" />  
 </schema>  
   
 <dataNode name="dn1" dataHost="v\_host1" database="mycatdb" />  
 <dataNode name="dn2" dataHost="v\_host2" database="mycatdb" />  
   
 <dataHost name="v\_host1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="localhost:3306" user="root" password="root">  
 <readHost host="hostS2" url="localhost:3306" user="root" password="root" />  
 </writeHost>  
 </dataHost>  
  
 <dataHost name="v\_host2" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="localhost:3307" user="root" password="root">  
 <readHost host="hostS2" url="localhost:3307" user="root" password="root" />  
 </writeHost>   
 </dataHost>   
</mycat:schema>

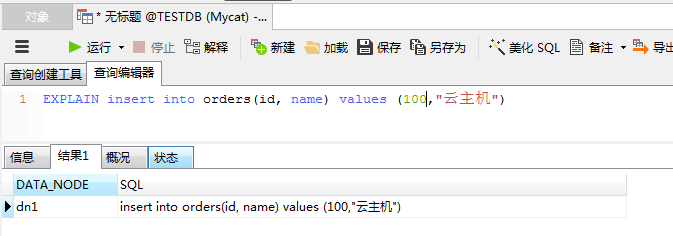
autopartition-long.txt

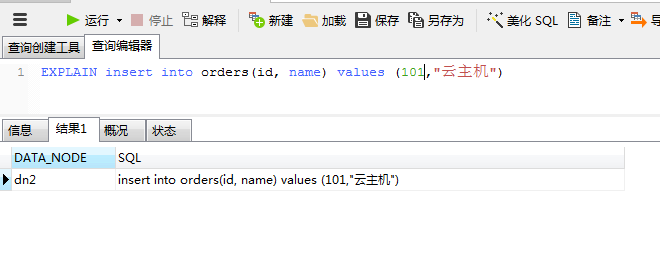
# range start-end ,data node index

# K=1000,M=10000.

0-100=0

101-200=1





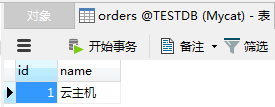
## 高可用

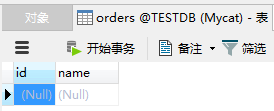
场景：两个数据库，当一个数据库挂掉，另一个数据库接收请求，自动替换已经挂掉的节点。

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="dn">  
 </schema>  
   
 <dataNode name="dn" dataHost="v\_host1" database="mycatdb" />  
   
 <dataHost name="v\_host1" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="mysql" dbDriver="native">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="hostM1" url="localhost:3306" user="root" password="root">  
 <readHost host="hostS1" url="localhost:3306" user="root" password="root" />  
 </writeHost>  
 <writeHost host="hostM2" url="localhost:3307" user="root" password="root">  
 <readHost host="hostS2" url="localhost:3307" user="root" password="root" />  
 </writeHost>   
 </dataHost>  
</mycat:schema>

多次查询orders表，发现有时候有数据，有时候没有数据，说明请求是随机分发到3306和3307上的





把3306服务停掉，再次在TESTDB上查询orders表，一直返回的是空数据，说明，因3306服务停止，无法分发到3306上，所以一直往3307上分发。