# mycat介绍

数据库的集群,分布式高可用的结构,需要引入中间件的软件,实现中间状态的计算,连接后端数据库完成客户端的所有功能需求sql语句的执行.将客户端与服务端数据库切分开来.

mycat是一个国内企业级别高性能数据库中间件.京东,ali都在使用.

# mycat特点

### 高性能的读写分离（速度非常快）

对于数据库集群来讲,主节点可以写数据,同时也可以读,从节点不能写数据,只能读数据.

多个节点组成的一个主从结构,可以实现读和写的分离操作---读写分离.

读写分离的存在能够大大提升数据库主从集群的使用效率.

select显然是读,一般大量分配给从节点

insert/drop/delete显然是写,全部分配给主节点

### 实现水平并行计算（水平拆分）

#### 单机数据库结构：

一个库多个表格存在

多个库，存在多个表格

#### 垂直拆分：

将不同的单机结构中的库数据，拆分到多个服务器中。由于微服务的存在，可以非常简单的实现垂直拆分（每一个拆分的结构中表格是完整的）

|  |
| --- |
| 计算机生成了可选文字: eaSydb tcart tuser tproduct torder tcart 垂直拆分是不切分 格 刂多分片 tuser tproduct torder |

#### 水平拆分：

将同一个数据库中同一个表的数据切分到多个数据库中（也就是表记录的拆分，分到不同数据库存储），出现的数据分片（拆分表格），如果一个表格数据库需要特别大的量级，比如30亿条，没有任何一个数据库技术可以支持这么大的表格数据（oracle 单表数据量亿级别，mysql单表数据量千万级别）,这样用水平拆分后逻辑上才能支持

|  |
| --- |
| 计算机生成了可选文字: mysq| tproduct 2亿条 tproduct tproduct t_product t_pr uct |

由于一个表格的整体数据被切分到了不同的真实数据库管理的一部分，那么实际运用中一定存在数据分片的计算。

mycat可以实现水平分片，大表总数据量百亿，管理和计算后端非常庞大的数据库集群。

官方数据：mycat的权威指南千亿级别。一般千亿级别的数据，都会采用大数据存储技术实现存储。

# mycat运行原理

mycay在客户端与数据库之间形成一道计算的桥梁，接受客户端的sql语句，完成计算，实现后端数据库的访问

|  |
| --- |
|  |

1. mycat启动时，就生成了管理连接后端真实数据库的所有datasource。
2. 客户端只能连接mycat发送sql语句。
3. mycat接受客户端发送的sql语句，执行各种计算（包括计算sql对应哪个分片、计算读写分离等），拿到了数据源，获取连接发送sql语句。
4. 数据库拿到mycat发送的sql开始执行。
5. mycat将数据库返回结果返回给客户端。

客户端还以为是mycat执行的sql获取数据结果。

# mycat处理假死逻辑

cobar是mycat的前身，是一个早期版本比较流行的中间件，没有解决后端连接的假死发生，2011出现了mycat解决了后端假死问题。

mycat不太需要关注假死的·现象，因为mycat可以非常高效的使用系统资源几乎不会出现假死现象。

### 什么是假死？

当中间件的线程资源在运行时被占满了，导致新的请求无法连接后端数据库（原因是没有线程可以分配），这时中间件错误的判断是数据库宕机导致的—这种情况称之为假死。

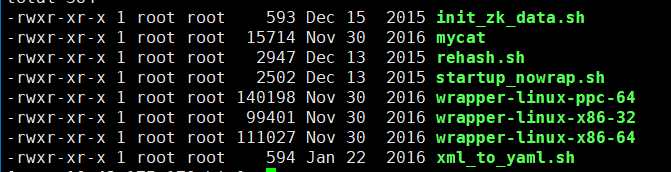
# mycat安装目录介绍

安装目录/home/software/mycat,目录结构如下:

|  |
| --- |
| 计算机生成了可选文字: 18 13 18 18 13 30 bin 22：20 2015 ate 22：21 0n 22：20飞土b 2015 2016 VerS10n到txt |

### bin

叫bin的文件夹一般都是运行命令脚本



可以在bin目录下执行（配置环境变量后可在任意地方运行）：

mycat console 控制台运行（一定bin文件夹下）

mycat start 后台运行

mycat stop 停止

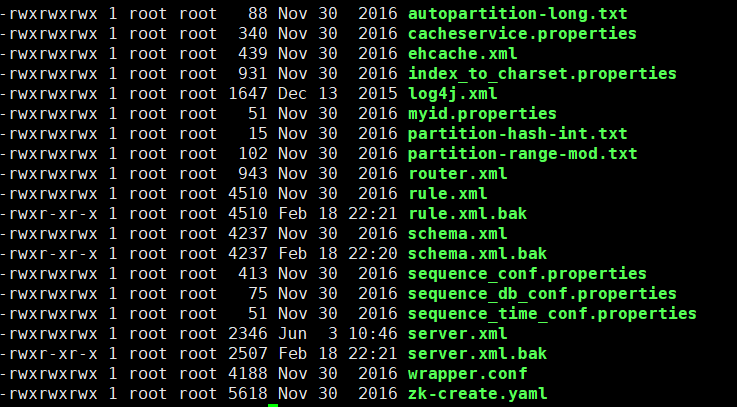
mycat restart 重启

### catlet

外部插件

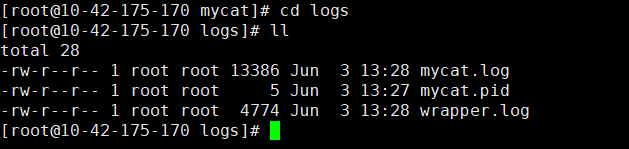
### conf

mycat的配置文件



### logs

运行日志（控制台运行可以直接输出日志）



# mycat的使用

### 登录（和mysql类型，端口8066）

linux 执行mysql命令(mycat支持默认环境mysql)

|  |
| --- |
| [root@10-42-147-110 ~]# mysql -utest -ptest -P8066 -h10.42.175.170 |

-u用户

-p密码

-P端口

-h ip

* sqlYog登录mycat

没有配置好后端数据的配置文件之前.不要使用sqlYog会卡死.

只要在连接属性配置 ip username password port

# mycat的配置文件详解

|  |
| --- |
| 计算机生成了可选文字: 需求： 准åmycat数据库 TESTDB t1：数据量大分片 t2：数据量不大 dn1 mycat TESTDB <schema> <dataNodename="dn1"database="dbl" db1-slave db1-master 双向备份 <tablename=tl dataNode="dn1,dn2" t2 <tablename="t2" 操亻乍TESTDB dataNode="dn1"> 中的tlt2 dn2 <writeHost> db2 master <writeHost> 囗 dataHost localhostl dataHost localhost2 sl<readHost> s2<readHost> s3<readHost> <dataNodename="dn2"database="db2"> |

在mycat根目录有conf文件夹，其中有三个xml（server、schema、rule）配置，都会使用到。

重要 一旦涉及到修改xml，习惯备份原文件。

|  |
| --- |
|  |

### server.xml

#### 标签结构

|  |
| --- |
| system  property  user  property  quarantine  whitehost  blacklist |
|  |

#### 标签和属性

system：

内容与mycat进程启动时占用的资源，内部配置属性有关 ，见上图

* + 1. property标签：可以配置压缩格式，配置端口号，配置占用系统的线程资源等等。

user：

mycat登录的用户配置。权限配置等

下面是user下<propertiy>可以配置的name

* + 1. name：用户名
    2. password：用户密码
    3. schemas：该用户可以访问的mycat中的数据库。可以访问多个数据库以 ，隔开，必须是mycat中存在的库(在schema.xml中配置才有)，如果有任何一个库不存在，将会报错
    4. readOnly：用户只读

例：test用户的密码test，可以访问TESTDB数据库

|  |
| --- |
| <user name="test">  <property name="password">test</property>  <property name="schemas">TESTDB</property>  </user> |

* 1. quarantine：防火墙配置
     1. whitehost：ip白名单

只有客户端ip地址在白名单范围内，才被允许访问mycat

|  |
| --- |
| <host host="127.0.0.1" user="mycat"/> |

上例中：只有客户端是从127.0.0.1的ip来访问mycat，并且使用mycat用户登录才能登录成功

* + 1. blacklist：sql黑名单 （规则见附录）

在黑名单中列出的sql规则，都不可以使用。例如：drop不允许执行，没有where条件的delete不能执行。

|  |
| --- |
| <blacklist check="false"></blacklist> |

标签中值是空的不检查

例如：限制客户端不能使用select \* 这种语句

|  |
| --- |
| <blacklist check="true">selectAllColumnAllow<blacklist> |

### schema.xml

#### 标签结构

|  |
| --- |
| 1. schema    1. table       1. childTable          1. childTable 2. dataNode 3. dataHost    1. heartbeat    2. writeHost       1. readHost |
|  |

#### 标签和属性

##### schema标签：

可以在一个schema.xml配置多个schema的标签，每一个都表示客户端可以看到一个数据库--不是真实的数据库 ，可以在mycat中show databases查询。

|  |
| --- |
| <schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100"> |

**属性**：

1. name：客户端可以看到和使用的数据库名称。sehema.xml中配置的多个schema标签，对应server.xml用户互配置标签中schemas属性。
2. checkSQLschema：可以配置true和false，表示所有sql语句是否需要添加表格的数据库名称。例如：表格 student,所在库 db1.配置fasle，sql发送到mycat，select \* from student.不会拼接db1。但是如果是true，变成select \* from db1.student.在mycat中存在多个库，多个表格，可以唯一定位表格的名称。
3. sqlMaxLimit：整数值，当sql语句做批量查询，mycat防止性能浪费，在判断sql语句语句中没有limit关键字自动拼接limit 0,100 查询前100条。

**table子标签：**

在一个schema标签中，存在的数据库表格，可以有多个table标签，表示多个表格。

|  |
| --- |
| <table name="t1\_student" primaryKey="s\_id" dataNode="dn1"/> |

属性

1. name：表格名称，所有表格数据都一定来自真实库，表格名称要和真实库的表格名一致，并且同一个schema标签中只能存在唯一一个名称table
2. primaryKey：主键名称，对应表格的真实数据中的主键名字，默认是id，当不是id时，需要配置这个属性
3. dataNode：table表格数据，可以根据数据量大小，实现水平切分，对应的数据分片计算绑定dataNode标签，这里可以设置当前表格被切分到了几个数据分片中,可以对应一个分片，可以对应多个分片。 指定dataNode标签的name。
4. rule：当表格需要进行对应多个分片数据切分时，指定切分数据数据分片计算逻辑，可以给配置rule规则，例如：auto-sharding-long 整数范围约束,表示以主键某个字段的整数做分片计算，0-500万对应第一个分片，500万-1000万对应第二个分片，1000万-1500万对应第三个分片。字符串可以使用一致性hash sharding-by-murmur. 指向rule.xml中<tableRule>的name。

##### dataNode标签

指向真实的数据节点（可以理解为真的数据库）

在mycat管理一个主从数据库集群作为分片使用，需要将集群包装在一个数据分片对象中，一个dataNode标签标示一个数据分片--最主要的作用就是计算分片的，利用名称，下标实现分片计算。

|  |
| --- |
| <dataNode name="dn1" dataHost="localhost1" database="db1" /> |

属性

1. name：分片名称，按照配置顺序每个dataNode还有一个下标从0开始，可以做一致性hash计算。
2. dataHost：一个分片不负责数据库集群的管理，只是绑定数据库集群管理对象dataHost，dataHost使用名字来绑定。
3. database：当前分片绑定真实数据库集群使用，真实数据库中database可能有多个，当前分片用的是哪个库（垂直划分），指定真实数据库库名称

##### dataHost标签

负责管理一个真正的数据库主从集群。连接池，读写分离都是由这个标签的对象实现的计算。

|  |
| --- |
| <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100"> |

**属性**

name：当前dataHost的名字，绑定dataNode时使用

maxCon：当前dataHost管理的数据主从集群每个节点的连接池中最大连接

minCon：最小连接

balance：读写分离的读逻辑，见mycat读写分离详解

writeType：读写分离的写逻辑，见mycat读写分离详解

dbType：默认mysql，数据库软件类型

dbDriver：默认mysql叫做native，如果是其他数据库给定driver全路径值

switchType：故障转移有关，见mycat故障转移详解

slaveThreshold：100是毫秒数，表示当前主从集群，从节点sql延迟100毫秒以上时，将不会使用该节点处理读数据逻辑

**heartbeat：**

dataHost连接使用后端数据时，实现心跳检测sql语句，一般有2中常用的，select user(),show slave status(只有使用该语句，slaveThreshold：100才能生效)

|  |
| --- |
| <heartbeat>select user()</heartbeat> |

**writeHost：**

写主机，表示在一个主从集群中的真实库连接标签。只能在其中配置主节点

|  |
| --- |
| <writeHost host="hostM1" url="10.202.4.39:3306" user="root" password="sf123456"> |

属性

host：代号名称，相当于name 一般会使用默认结构 hostM1 第一个主节点

url：ip：port连接

user：登录数据用户名

password：登录密码

**readHost：**

读主机，表示主从集群中真实库连接，主从节点都可以在这里配置，一般都是使用从节点。

|  |
| --- |
| <readHost host="hostS2" url="192.168.1.200：3306" user="root" password="xxx" /> |

属性

host：代号名称，相当于name 一般会使用默认结构 hostM1S1 第一个主节点第一个从节点

url：ip：port连接

user：登录数据用户名

password：登录密码

# mycat入门案例（实现代理中间件）

### 环境准备

一个后端数据库，这里用mstest来测试

一个mycat服务

### 功能描述

通过客户端(mysql登录,sqlYog)登录mycat。

访问mycat的一个逻辑数据库TESTDB。

看到使用一个表格tb\_user,不需要分片,查询所有数据都来自于后端一个真是库。

### 结构图

|  |
| --- |
| 计算机生成了可选文字: mycat151．60 TESTDB tbuser mysql sqIYog 48．69 mstest tbuser |

### 配置xml文件

#### Server.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mycat:server SYSTEM "server.dtd">  <mycat:server xmlns:mycat="http://org.opencloudb/">          <system>                  <property name="defaultSqlParser">druidparser</property>          </system>          <user name="root">                  <property name="password">root</property>                  <property name="schemas">TESTDB</property>          </user>  </mycat:server> |

#### Schema.xml

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://org.opencloudb/" >          <!--mycat only one logic database TESTDB-->          <schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100">                  <!--tb\_user-->                  <table name="tb\_user" primaryKey="id" dataNode="dn1"/>          </schema>          <dataNode name="dn1" dataHost="localhost1" database="mstest"/>          <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="M1" url="10.9.48.69:3306"                  user="root" password="root"/>          </dataHost>  </mycat:schema> |

### 测试

### 常见问题

* xml的语法编写配置补熟练,有语法错误.

从mycat启动日志中,找到Caused by错误提示关键字，mycat console可以直接从控制台看到日志

|  |
| --- |
| 计算机生成了可选文字: java:105） jvm1 r·java：73） jvm1 atorg.opencIoudb.MycatServer.<cIinit>(MycatServe more jvm1 丨Causedby：org.xml.sax.SAXParseException;lineNumber: 8；column heelementtype"table"mustbeterminatedb ythematchingend-tag"</table>" jvm1 atC0m·Sun·rg.apache.xerces.internaI.utiI.ErrorH andlerWrapper。CreateSAXParseException（ErrorHandlerWrapper。java：20 |

测试登录mycat访问后端真实库的表格资源,有一个常见的问题

* writeHost/readHost 提供的datasource创建的必要属性
  + url
  + user
  + password

给错了,所以mycat无法创建后端数据库的datasource导致你无法使用mycat执行操作表格增删查改的功能

ERROR 3009 (HY000): java.lang.IllegalArgumentException: Invalid DataSource:0

# mycat读写分离

数据某个分片中，实现的主从高可用备份,通过对主的写操作，对从的读操作将读写分离执行提升集群使用效率

读写分离和故障转移都是基于主从结构实现(mycat至少管理一个后端主从结构)

* 之前主从复制课程,实现了一主一丛的数据库结构,测试读写分离,故障转移,为了测试效果,把搭建好的主从给断开

### 准备环境

* 准备2个后端数据库,并且将已有的主从断开
  + 10.9.151.60:3306 10.9.48.69:3306
  + 到从节点 执行 stop slave
* 准备一个mycat的软件

一个云主机运行了一个后端数据库

令外一个云主机运行另另一个后端数据库

同时运行了mycat

### 结构图

|  |
| --- |
| 计算机生成了可选文字: mycat151．60 dnldatanode |

### 配置文件schema.xml

|  |
| --- |
| 经过测试发现,配置了2个writeHost标签,但是读和写的操作都是在第一个writeHost进行的.  因为balance为0，表示不开启读写分配 |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://org.opencloudb/" >          <!--mycat only one logic database TESTDB-->          <schema name="TESTDB" checkSQLschema="true" sqlMaxLimit="100">                  <!--tb\_user-->                  <table name="tb\_user" primaryKey="id" dataNode="dn1"/>          </schema>          <dataNode name="dn1" dataHost="localhost1" database="mstest"/>          <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="M1" url="10.9.48.69:3306"                  user="root" password="root">                  </writeHost>                  <writeHost host="M2" url="10.9.151.60:3306"                  user="root" password="root">                  </writeHost>          </dataHost>  </mycat:schema> |

### 读写分离逻辑属性

一个dataHost标签中,由balance和writeType控制读写逻辑.

**writeType**：实现写逻辑控制，二个值

0：默认值,在index下标为0的writeHost数据库中写数据

1：mycat1.5版本以上的已经不推荐使用了,随机的在所有的writeHost进行写操作,覆盖balance的读逻辑,在所有\*\*Host进行随机的读

**balance**： 单独控制一个主从结构的读写分离的读逻辑,想让balance值生效,writeType不能是1

0：默认值,不开启读的分离,只会在index=0的writeHost进行读

1：读的操作,除了index=0的writeHost,其他的所有Host都随机读.当读并发超级高时,所有后端是数据库读都承受很大压力时,第一个writeHost才会参与一部分读的分离.

2：随机在所有\*\*Host进行读

3：随机在所有的ReadHost进行读,没有readHost时,只会在第一个writeHost进行读

# Mycat 故障转移

一个数据分片的读写工作，由于正在使用的某个节点出现故障，连接转向一个备份的节点实现读写

读写分离和故障转移都是基于主从结构实现(mycat至少管理一个后端主从结构)

* 之前主从复制课程,实现了一主一丛的数据库结构,测试读写分离,故障转移,为了测试效果,把搭建好的主从给断开
* 目的:通过高可用来提升一个数据分片的可靠性.

### switchType属性

dataHost管理的主从数据库结构,可以通过switchType决定故障转移逻辑.

1：默认值,当正在通过写功能的host(index=0的writeHost)故障,开启故障转移,将index=1的writeHost顶替(替换index下标).

-1：不开启故障转移

### 测试

通过前面读写分离案例,对第一个writeHost实现宕机,观察读写是否正常.

# Mycat一致性hash算法

|  |
| --- |
|  |

基础:hash环

数据映射: 分片对象dataNode 名字 字段值

对应关系:字段值的整数顺时针寻找最近dataNode整数

分片计算完成,后续还有读写分离计算.

# Mycat跨分片的表格设计

数据库中很多个表格的时候,可以实现关联查询,比如商品表格和商品分类就可以设计成关联的表格.像这样的表格如果涉及到分片表格的配置,需要考虑底层数据是否跨分片.

mycat不支持数据跨分片的,必须在业务层或者mycat配置中解决这个问题.

### 全局表

#### 应用场景

* + tb\_product：商品详情表
    - 保存了一个项目的所有商品数据，大表（数据非常多），在mycat应该配置成分片表格 。
  + tb\_category：商品分类表
    - 保存了当前电商项目的所有商品分类数据(京东大概1000多个商品分类)，不需要设计成分片表格，可以设计成非分片。
  + 两张表格的关系

tb\_product

|  |  |  |
| --- | --- | --- |
| id | product\_name | c\_id(分类id属于哪个分类) |
| 1 | 乐事薯片 | 1 |
| 2 | 可口可乐 | 1 |
| 3 | 美年达 | 1 |
| 600w | 海尔电视 | 2 |
| 601w | 海信电视 | 2 |
| 602w | 康佳彩电 | 2 |

tb\_category

|  |  |
| --- | --- |
| id | cat\_name |
| 1 | 食品饮料 |
| 2 | 电视 |

关联查询

select \* from tb\_category left join tb\_product on tb\_category.id=tb\_product.c\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | cat\_name | id | product\_name | c\_id |
| 1 | 食品饮料 | 1 | 乐事薯片 | 1 |
| 1 | 食品饮料 | 2 | 可口可乐 | 1 |
| 1 | 食品饮料 | 3 | 美年达 | 1 |
| 2 | 电视 | 600w | 海尔电视 | 2 |
| 2 | 电视 | 601w | 海信电视 | 2 |
| 2 | 电视 | 602w | 康佳彩电 | 2 |

#### 跨分片问题

Category表无需分片，那么他的数据写到第一个writehost， 而product表数据太多需要分片，Mycat计算数据分片时，根据ID计算，假如分到了二个数据库中，那么在mycat关联查询时，而关联的category数据在另一个数据库中，这样查询出来关联的category数据为空，因为mycat不支持这种查询方式。

|  |
| --- |
| D:\untitled.png |

由于mycat底层不支持跨分片的查询,所以对于上述一些数据 600w 601w 602w这3个商品关联查询时没有返回的分类数据.

#### 解决办法

既然原因是category表只在一个分片中，那么将category表复制到所有分片不就行了吗？这就是全局表。

|  |
| --- |
| D:\untitled.png |

全局表配置：下列中tb\_category就是全局表，type的值为global，数据复制到dn1和dn2，而且没有分片既没有rule属性。

|  |
| --- |
| <table name="tb\_product" primaryKey="id" dataNode="dn1,dn2" rule="auto-sharding-long"/>  <table name="tb\_category" primaryKey="id" dataNode="dn1,dn2" type="global"/> |

一张表格在配置table标签时,给指定了多个分片,但是没有指定rule计算分片规则,默认就会将表格新增数据同步到所有分片.

#### 企业的全局表格

* + 企业中使用mycat为了不出现跨分片的查询,将一些工具字典表设计成全局表格的形式,这样一大批业务表格关联查询就不需要考虑解决跨分片的问题
  + 工具字典表:解释业务中一些数据的数据,特点是数据量稳定,变化不到,数据量不大.
  + 比如:
    - 电信业务中,有记录日志字段111119980707982556,这是业务表格的一个字段值,要想解释这个字段必须从2张工具字典表关联查询比如
      * 1111:地市代号-->江苏-南京
      * 982556:员工代号-->山东籍-济南电信分公司-56号员工-刘首付

### ER分片表

#### 应用场景

* + tb\_order:每一行数据都表示一个用户的某个订单
    - 保存了所有用户的所有订单数据
    - 订单数据量非常大--分片表格
  + tb\_order\_item:每一行数据都表示一个订单中某个商品
    - 保存了电商项目的所有订单商品数据
    - 数据量更大--分片表格
  + 两张表格的关系
    - 订单一行数据对应多个订单商品行数据(1对多的关系)

tb\_order

|  |  |  |
| --- | --- | --- |
| id | money | user\_id |
| 500w | 800 | a |
| 800w | 1500 | b |

tb\_order\_item

|  |  |  |
| --- | --- | --- |
| id | o\_id | product\_name |
| 1 | 800w | 海尔电视 |
| 2 | 800w | 乐事薯片 |
| 3 | 800w | 田格本(10个) |
| 601w | 500w | 面包 |
| 602w | 500w | 烤箱 |

* 关联查询2个表格

select \* from tb\_order left join tb\_order\_item on

tb\_order.id=tb\_order\_item.o\_id where tb\_order.id=500w

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | money | user\_id | id | o\_id | product\_name |
| 500w | 800 | a | 601w | 500w | 面包 |
| 500w | 800 | a | 602w | 500w | 烤箱 |

#### 存在的问题

假如mycat分片计算逻辑为id为1-500存储到分片dn1，最后二个关联表最后相关联的数据存放的数据分片不一致，导致了查询关联数据查询不到的问题。

见下图：mycat计算时tb\_orderid为1-500W存储到dn1，同时tb\_order\_item的id为1-3也存储到dn1，但是外键是800W，对应tb\_orderid存储到了dn2，导致关联查询问题。

|  |
| --- |
| **D:\untitled.png** |

#### 解决办法

上面的二个表为一对多关系，一个订单（order）包括多个商品（order\_item），在mycat中可认为是order为主表，order\_item为子表。

既然有关联关系的表记录被数据分片到了不同的dn，那我想办法让他们在同一个dn不就行了吗？

主表用id计算分片，而子表用外键值计算分片，有关联关系的记录id值和外键一定相等，数据分片的位置也一定一样

|  |
| --- |
| D:\untitled.png |

* 关联表格的主子表关系
  + 由主表来定义分片计算规则,包括:分片个数,分片是谁,分片计算规则rule
  + 子表跟随进行不再自定义这些内容

主子表区分:

tb\_order主表(父表)

tb\_order\_item从表(子表)

父表的数据单独存在可以有意义

子表的数据单独存在没有意义

选择的计算分片的字段:订单id(子表外键字段)

#### ER分片表配置

以上例中的t\_order表和t\_order\_item表为例，t\_order主键为order\_id。t\_order\_item表主键为id，外键为t\_order的order\_id，名字也叫order\_id。

* 定义计算分片字段：order\_id
* 定义计算分片的方法：一致性hash

##### 配置schema.xml

在schema.xml中添加一个table作为主表的配置即可实现ER分片表的使用,table下需要包含一个子表的childTable标签.

|  |
| --- |
| <table name="t\_order" primaryKey="order\_id" dataNode="dn3,dn4"  rule="easymall-order-hash">          <!--t\_order\_item childTable-->          <childTable name="t\_order\_item" primaryKey="id" joinKey="order\_id"          parentKey="order\_id"/>  </table> |

属性**joinKey**：属于子表的外键关联字段(t\_order\_item 外键order\_id)

属性**parentKey**：属于子表外键关联字段对应主表的字段名称(t\_order order\_id)

ER分片表个配置中2个属性在childTable必须成对出现 ，二个必须设置的意思？

添加2个dataNode dn3 dn4

|  |
| --- |
| <dataNode name="dn3" dataHost="localhost1" database="easydb"/>  <dataNode name="dn4" dataHost="localhost2" database="easydb"/> |

##### 配置rule.xml

* + 添加一个专门为订单表格设计的一致性hash计算的tableRule

|  |
| --- |
| <tableRule name="easymall-order-hash">  <rule>  <columns>order\_id</columns>  <algorithm>murmur</algorithm>  </rule>  </tableRule> |

以order\_id计算

murmur:一致性hash

### 局限性

购物车表格,用户表格,商品表格(订单,订单商品,商品,用户表格);

这些表格的总体结构都数据多对多,mycat不能解决海量数据的多对多的关系.

以购物车,用户,商品表格为例;

2个主表用户,商品

一个子表购物车(多对多外键关联的表格一样,同时存在2个主表的字段关联)

如果数据是海量,多对多在mycat中无法直接处理跨分片.

解决办法:

* 可以使用大数据管理数据关系,做数据清洗
* 从业务逻辑多次进行查询

# 附录A--sql黑名单含义

<blacklist check="true">selelctAllow</blacklist>

配置项        缺省值        描述

selelctAllow        true        是否允许执行 SELECT 语句

selectAllColumnAllow        true        是否允许执行 SELECT \* FROM T 这样的语句。如果设置为 false，不允许执行 select \* from t，但可以select \* from (select id, name from t) a。这个选项是防御程序通过调用 select \* 获得数据表的结构信息。

selectIntoAllow        true        SELECT 查询中是否允许 INTO 字句

deleteAllow        true        是否允许执行 DELETE 语句

updateAllow        true        是否允许执行 UPDATE 语句

insertAllow        true        是否允许执行 INSERT 语句

replaceAllow        true        是否允许执行 REPLACE 语句

mergeAllow        true        是否允许执行 MERGE 语句，这个只在 Oracle 中有用

callAllow        true        是否允许通过 jdbc 的 call 语法调用存储过程

setAllow        true        是否允许使用 SET 语法

truncateAllow        true        truncate 语句是危险，缺省打开，若需要自行关闭

createTableAllow        true        是否允许创建表

alterTableAllow        true        是否允许执行 Alter Table 语句

dropTableAllow        true        是否允许修改表

commentAllow        false        是否允许语句中存在注释，Oracle 的用户不用担心，Wall 能够识别 hints和注释的区别

noneBaseStatementAllow        false        是否允许非以上基本语句的其他语句，缺省关闭，通过这个选项就能够屏蔽 DDL。

multiStatementAllow        false        是否允许一次执行多条语句，缺省关闭

useAllow        true        是否允许执行 mysql 的 use 语句，缺省打开

describeAllow        true        是否允许执行 mysql 的 describe 语句，缺省打开

showAllow        true        是否允许执行 mysql 的 show 语句，缺省打开

commitAllow        true        是否允许执行 commit 操作

rollbackAllow        true        是否允许执行 roll back 操作

##如果把 selectIntoAllow、deleteAllow、updateAllow、insertAllow、mergeAllow 都设置为 false，这就是一个只读数据源了。##

拦截配置－永真条件

selectWhereAlwayTrueCheck        true        检查 SELECT 语句的 WHERE 子句是否是一个永真条件

selectHavingAlwayTrueCheck        true        检查 SELECT 语句的 HAVING 子句是否是一个永真条件

deleteWhereAlwayTrueCheck        true        检查 DELETE 语句的 WHERE 子句是否是一个永真条件

deleteWhereNoneCheck        false        检查 DELETE 语句是否无 where 条件，这是有风险的，但不是 SQL 注入类型的风险

updateWhereAlayTrueCheck        true        检查 UPDATE 语句的 WHERE 子句是否是一个永真条件

updateWhereNoneCheck        false        检查 UPDATE 语句是否无 where 条件，这是有风险的，但不是SQL 注入类型的风险

conditionAndAlwayTrueAllow        false        检查查询条件(WHERE/HAVING 子句)中是否包含 AND 永真条件

conditionAndAlwayFalseAllow        false        检查查询条件(WHERE/HAVING 子句)中是否包含 AND 永假条件

conditionLikeTrueAllow        true        检查查询条件(WHERE/HAVING 子句)中是否包含 LIKE 永真条件

其他拦截配置

selectIntoOutfileAllow        false        SELECT ... INTO OUTFILE 是否允许，这个是 mysql 注入攻击的常见手段，缺省是禁止的

selectUnionCheck        true        检测 SELECT UNION

selectMinusCheck        true        检测 SELECT MINUS

selectExceptCheck        true        检测 SELECT EXCEPT

selectIntersectCheck        true        检测 SELECT INTERSECT

mustParameterized        false        是否必须参数化，如果为 True，则不允许类似 WHERE ID = 1 这种不参数化的 SQL

strictSyntaxCheck        true        是否进行严格的语法检测，Druid SQL Parser 在某些场景不能覆盖所有的SQL 语法，出现解析 SQL 出错，可以临时把这个选项设置为 false，同时把 SQL 反馈给 Druid 的开发者。

conditionOpXorAllow        false        查询条件中是否允许有 XOR 条件。XOR 不常用，很难判断永真或者永假，缺省不允许。

conditionOpBitwseAllow        true        查询条件中是否允许有"&"、"~"、"|"、"^"运算符。

conditionDoubleConstAllow        false        查询条件中是否允许连续两个常量运算表达式

minusAllow        true        是否允许 SELECT \* FROM A MINUS SELECT \* FROM B 这样的语句

intersectAllow        true        是否允许 SELECT \* FROM A INTERSECT SELECT \* FROM B 这样的语句

constArithmeticAllow        true        拦截常量运算的条件，比如说 WHERE FID = 3 - 1，其中"3 - 1"是常量运算表达式。

limitZeroAllow        false        是否允许 limit 0 这样的语句

禁用对象检测配置

tableCheck        true        检测是否使用了禁用的表

schemaCheck        true        检测是否使用了禁用的 Schema

functionCheck        true         检测是否使用了禁用的函数

objectCheck        true        检测是否使用了“禁用对对象”

variantCheck        true        检测是否使用了“禁用的变量”

readOnlyTables        空        指定的表只读，不能够在 SELECT INTO、DELETE、UPDATE、INSERT、MERGE 中作