数据库高可用分布式集群

2018年9月14日 14:30

单机数据库策略

数据库的设计模式

早期系统的特点,硬件非常昂贵,磁盘空间有限. 考虑数据存储的磁盘空间占用率

三范式的设计模式

极大的节省磁盘空间;系统的数据库设计模式遵循三范式的话, 系统的访问效率变低了;利用时间换空间;

随着访问并发不断增加,三范式已经慢慢退出舞台,在单机策略 下需要一种类似于三范式的东西达到一个目的,利用空间换时

间.违反三范式就可以做到这个结果,反范式;

索引优化

概念:数据库具有索引的数据结构,对索引进行优化来实现查询 或者写操作的时间效率的提升---索引的优化;

索引:按照一定规则的数据结构设计,实现的存储在磁盘上的一批文件,这些文件可以快速定位查询数据;这样的一批文件称之 为索引文件.

词典查询:

词典查询一个字(在数据库搜索一个表格某条数据记录) 需求:差"李"字

第一步:确定查询的方法(拼音法,偏旁部首,难检字,笔画)

索引的数据结构是不一样的(根据不同条件查询时,选择的索引结构可能有变化),对应某个数据库,就有具体的某种索引数据结

分区 day04all 的第 36页

结构可能有变化),对应某个数据库,就有具体的某种索引数据结 构

确定使用拼音

第二步:打开字典,从拼音结构的内容从a-l,a,b,c这些内容是写 在字典页上的数据(磁盘索引文件中的某些数据)

肉眼观察a,b,c相当于磁头在读取索引文件的数据---第一次磁 盘I/O操作;

第三步:找la,le,lan,li ----磁盘的第二次I/O

第四步:找里,理,李 ----第三次磁盘I/O

李 65页

第五步:跳转页面直接反倒65查看李的所有数据----指针跳转+ 第四次磁盘I/O

mysql数据库中的索引文件到底是什么结构?

分区 day04all 的第 37页

mysql的索引使用到的数据结构B+TREE,B-tree,二叉树,红黑树 三阶的B+tree

每个长方形是一个数据单元(I/O一次读取,大小根据文件系 统定义的,windows的单元大小在ntfs中4KB,最小512B)

根节点(查询的起始节点,也是枝节点的一种),枝节点,叶节

点;

随着索引文件的数据量增加,B+TREE会变得比途中高,宽

利用索引文件中的数据可以快速定位到某一条数据记录

例子:查45

第一次磁盘I/O读取根节点 5,28,65;45与3个数对

比,28<45<65,获取到对应的指针(第二个指针,[28,65))

第二次磁盘I/O读取枝节点28,35,56,35<45<56,第二个指 针([35,56))

第三次磁盘I/O读取叶节点 35,38,45,45=45找到了索引的 最终数据(从45这个索引的数据可以获取卫星指针,直接寻

分区 day04all 的第 38页

最终数据(从45这个索引的数据可以获取卫星指针,直接寻 址到45对应的数据记录(username,id,age,school))

单机策略当中

解决单机性能问题

1. 数据库的设计
2. 索引的数据优化

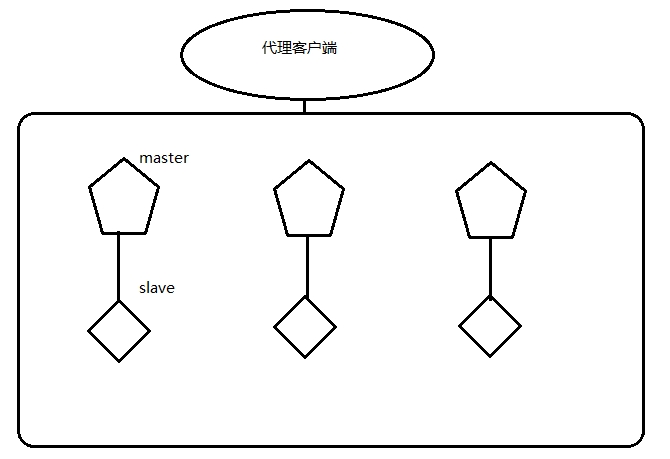
单机的数据库总会碰到单机处理数据的瓶颈

mysql单表数据量千万级

oracle单表数据量 亿级

数据库集群结构

分布式的数据库集群完成海量数据记录的存储,跨域单个数据 库的表格上限瓶颈;

mysql数据库集群阶段实现一个分布式的高可用的集群. mysql的主从复制

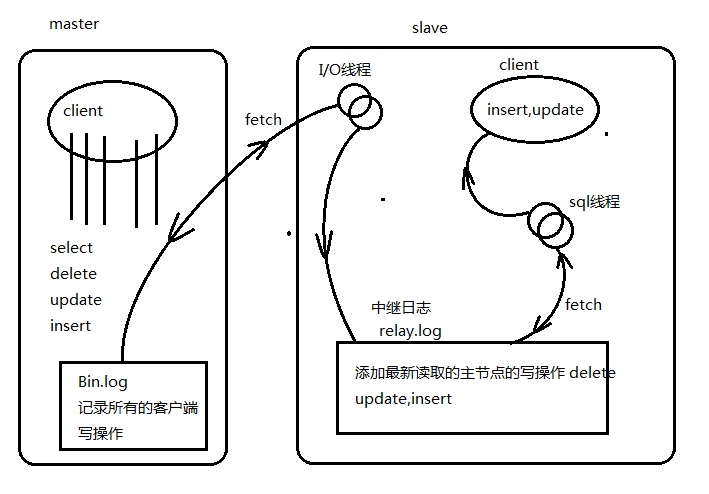
分区 day04all 的第 39页

mysql的主从复制

类似于redis主从复制结构,mysql也可以提供多级的,一主多从的主从结构;

利用2台数据库,搭建一个简单的主从结构,主节点数据,会实时同步到从节点;

mysql数据库的主从复制原理;

master-salve结构一旦搭建,需要主数据库在本机上开启记录日志操作信息的二进制文件.所有的master写操作都会把命令做二进制处理写入文件中Binary.log

挂接到当前主节点的从节点,开启一个I/O线程,和sql线程

I/O线程连接登录主节点实时的获取Binary.log的更新数据(如何知道哪些是更新的哪些是已经抓取过的.根据position指针的值),例如上次更新时,positin=900,主节点一旦写操

作,position1355,I/O上次记录是900,变成了1355,从900之后的内容一值获取到1355的更新数据;

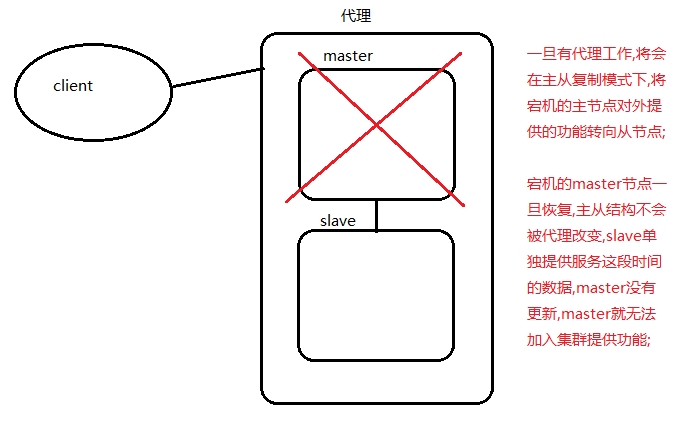
所有的I/O获取的写操作都会写到从节点上的一个中继日志Relay.log,

sql线程监听逻辑和I/O一样,监听中继日志,将更新数据启动一个从节点的客户端直接执行, 从而达到与主节点的数据同步的效果;

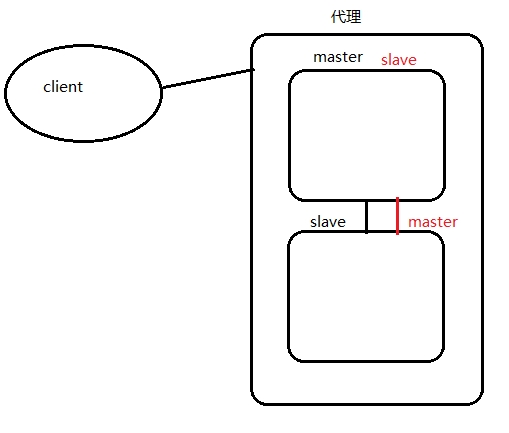
单机热备

热备:数据库中,热是动态,备是备份,一个主从结构在一个高可用替换的逻辑下,就称之为单机热备;

分区 day04all 的第 40页

基于单方向的主从复制结构的热备关系,单机热备;

双机热备

为了解决单机热备存在的问题,将单向的主从关系在2个数据库 节点中做成双向的,互为主从,无论哪个宕机,在重启后,从节点

的功能都会从另外一台来更新数据,继续提供数据库的高可用,

可靠的数据访问;

分区 day04all 的第 41页

多台mysql安装

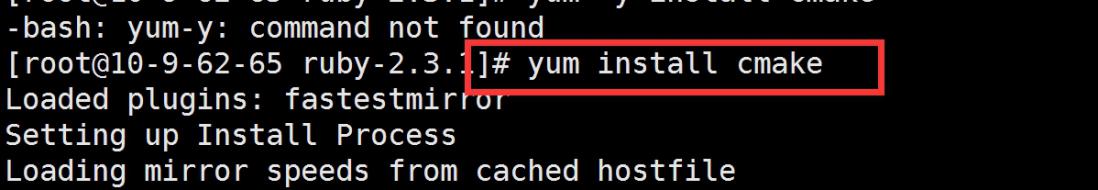
2017年12月13日 18:54

主从结构 一主一从，需要2个服务器完成数据库的安装

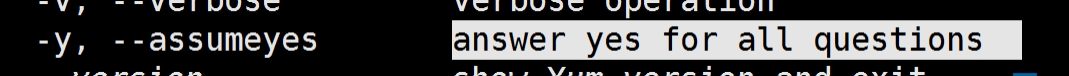
虚拟机使用克隆功能,先克隆在安装.

**安装cmake(无需安装)**

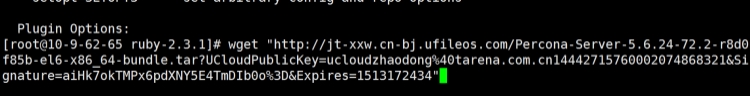
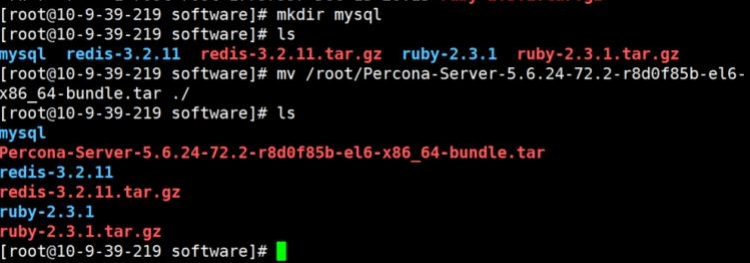
#yum -y install cmake

截图和命令不符,可以查看yum --help

发现-y选项是所有回答yes/no的都回答yes比较省事

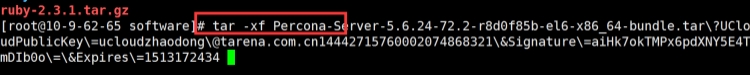
**安装包解压**

**如果使用自己的虚拟机先上传在移动在解压**

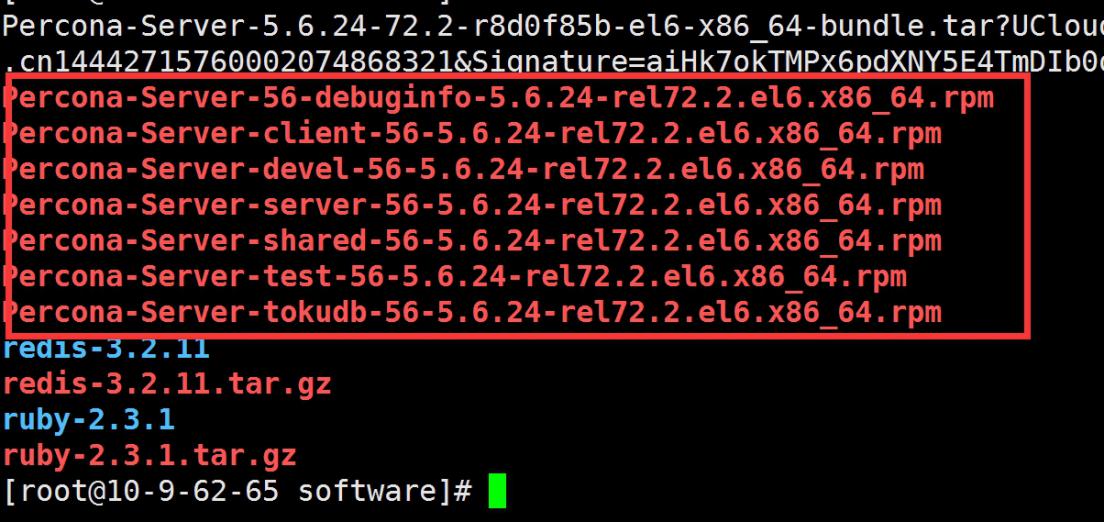
将安装包移动到software(根据自己的习惯管理目录)

mv Per\*\*\*\*\*\* /home/software

到software目录中执行tar命令

ls发现众多的rpm文件

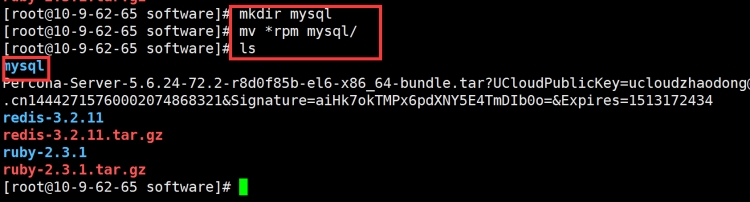
分区 day04all 的第 43页

生成一个管理文件mysql

mkdir mysql

将所有文件移动到mysql目录中

mv \*rpm mysql

在mysql目录中执行rpm安装命令

rpm命令可以理解为专门对rpm格式的文件进行解压安装的功能命令

这里我们需要rpm安装4个文件,顺序不能错乱

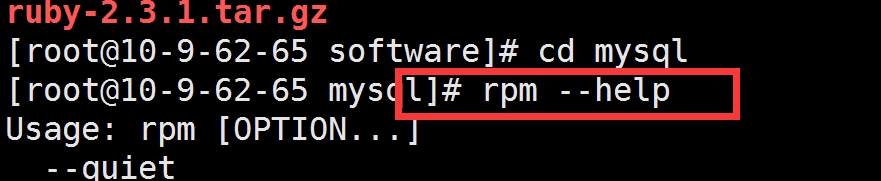
rpm -ivh Percona-Server-56-debuginfo-5.6.24-rel72.2.el6.x86\_64.rpm

rpm -ivh Percona-Server-shared-56-5.6.24-rel72.2.el6.x86\_64.rpm

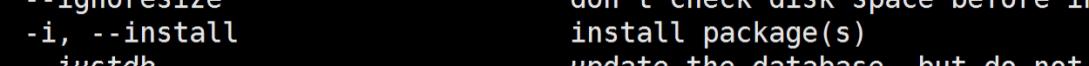
rpm -ivh Percona-Server-client-56-5.6.24-rel72.2.el6.x86\_64.rpm

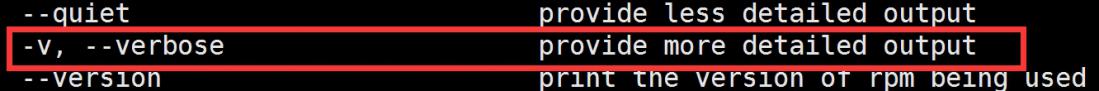
rpm –ivh Percona-Server-server-56-5.6.24-rel72.2.el6.x86\_64.rpm

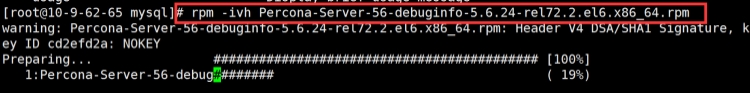
ivh的含义可以调用rpm --help查看

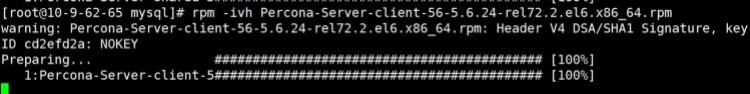
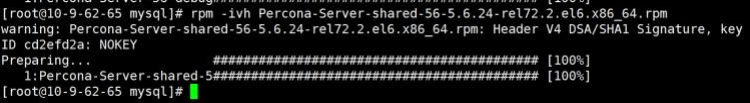
-h, --hash print hash marks as package installs (good with -v)

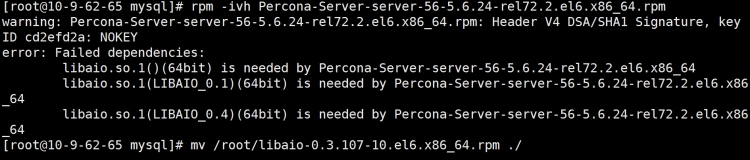
-i, --install install package(s)

-v, --verbose provide more detailed output

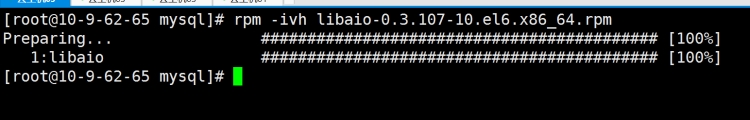
按照上面的顺序依次安装rpm文件

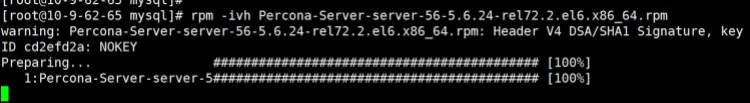
分区 day04all 的第 44页

第四个rpm文件如果出现以下问题

需要安装libaio

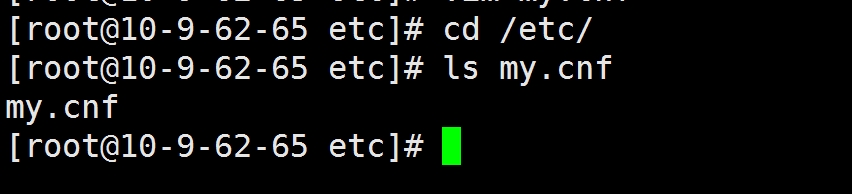
在软件地址中有对应文件的rpm,使用wget就可以获取

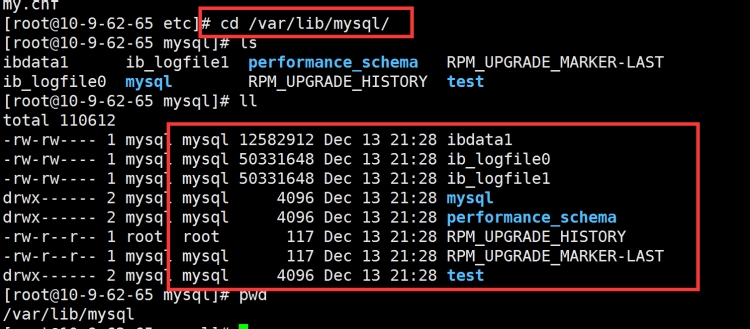
重新安装percona的server

需要用到的rpm就安装完成了

安装完成后检查两点

一个是/etc/my.cnf存在不存在

一个是/var/lib/mysql目录中有没有mysql数据文件

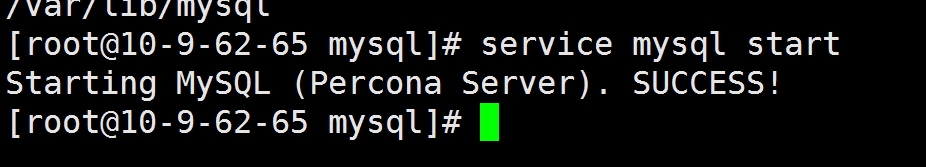
如果有说明安装基本成功

**启动mysql服务**

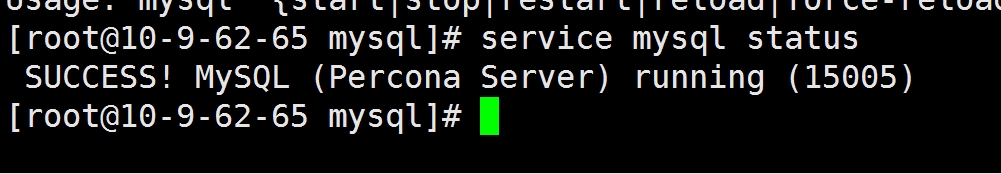
分区 day04all 的第 45页

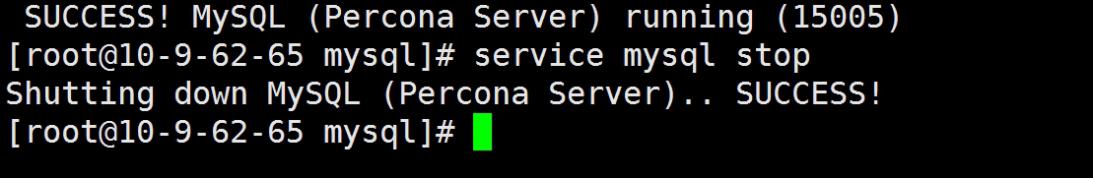
**启动mysql服务**

service mysql start //自动安装到它自己制定的目录下，注册服务为mysql 这是就会发现/var/lib/mysql目录下的文件多了一些内容

然后可以调用另外两个命令

service mysql status 查看服务状态

service mysql stop 停止服务

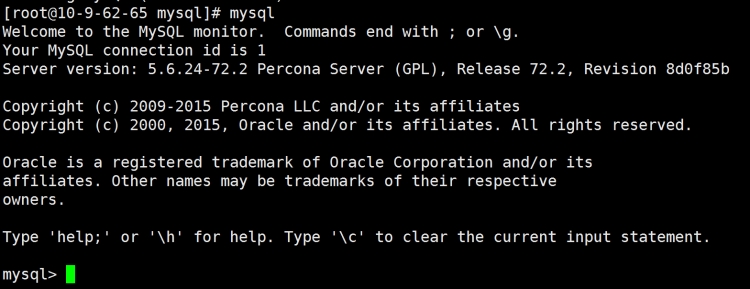
service mysql restart

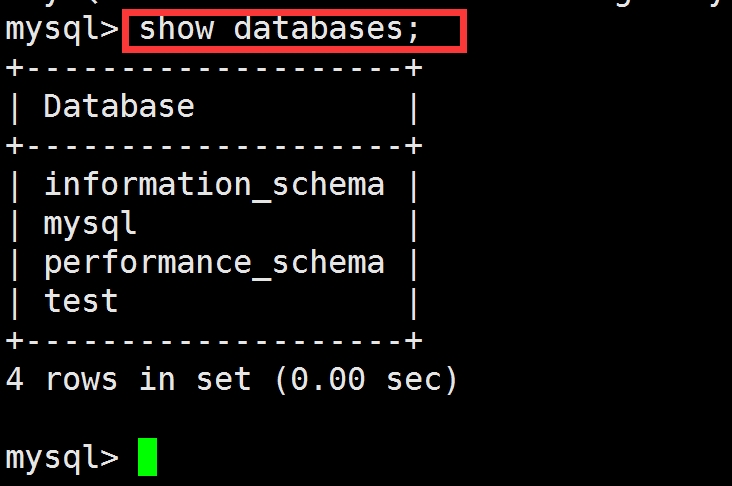
然后在开启服务运行下面的操作

**修改密码**

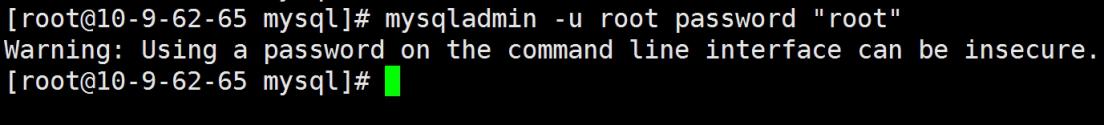
默认安装的mysql没有密码

直接输入命令mysql就可以进入查看

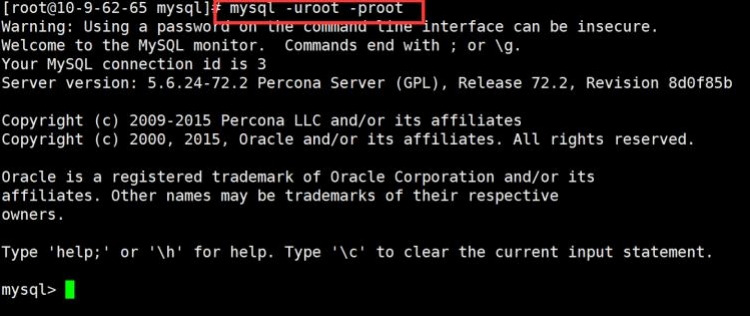
分区 day04all 的第 46页

所以需要我们配置访问密码,用户名root,密码root

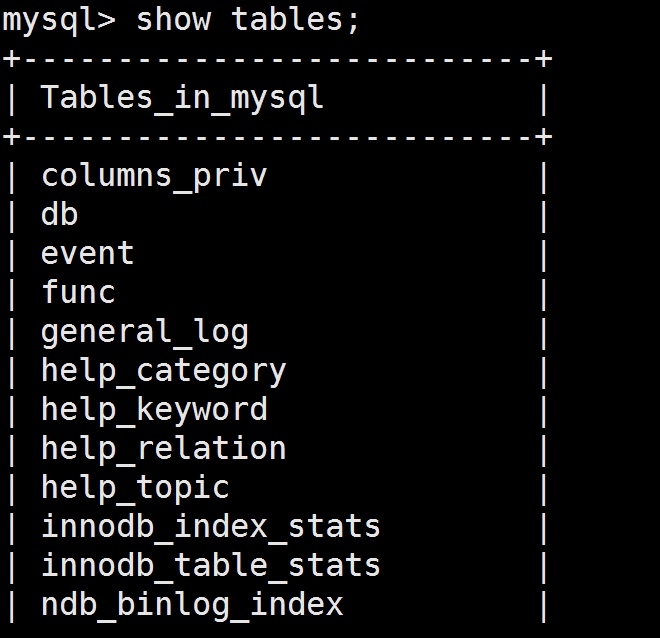
#mysqladmin -u root password "root"

warning不用管

使用有密码的命令进入mysql

验证mysql命令是否有效

分区 day04all 的第 47页

开启3306端口或者直接关闭防火墙

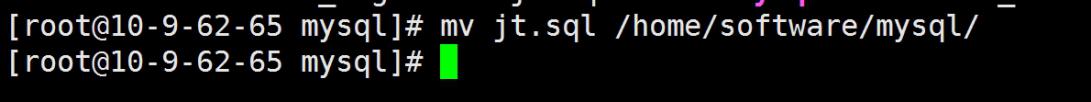
/sbin/iptables -I INPUT -p tcp --dport 3306 -j ACCEPT

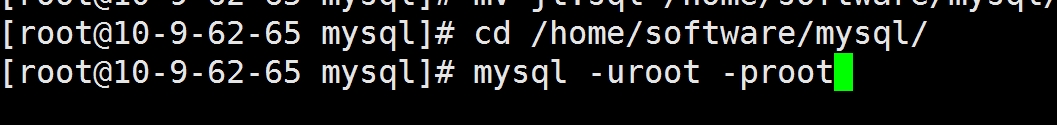
查看防火墙端口开启状态

service iptables status

开启完成

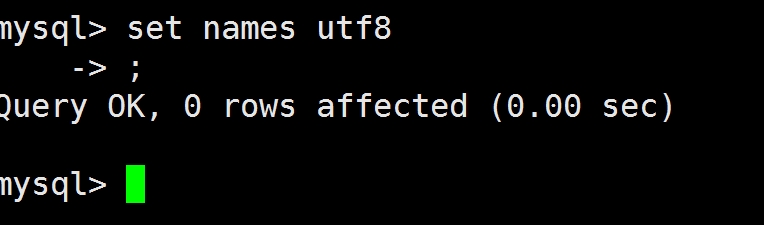
**创建数据库表格和数据(导入jt.sql文件)(数据库暂时不导入)** 上传文件到mysql目录下,jt.sql文件

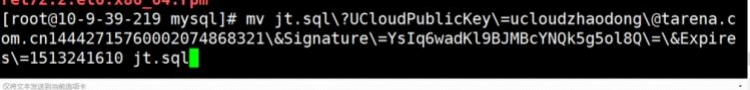
在/home/softwre/mysql目录中登录mysql客户端

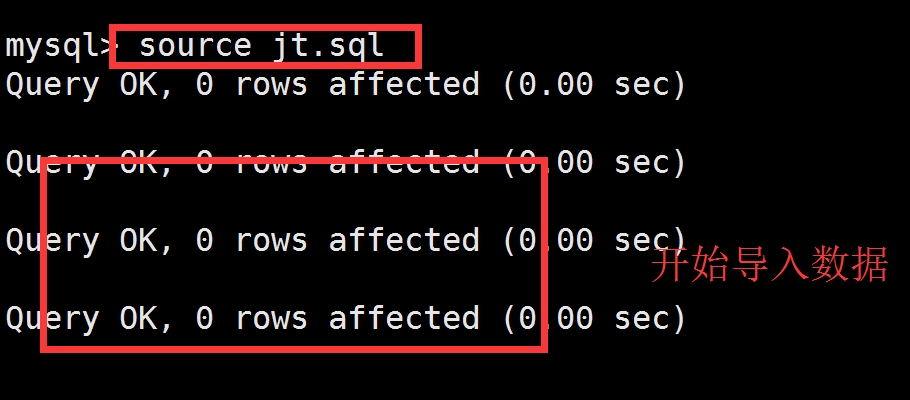
在客户端中执行命令

mysql>set names utf8; #防止乱码

分区 day04all 的第 48页

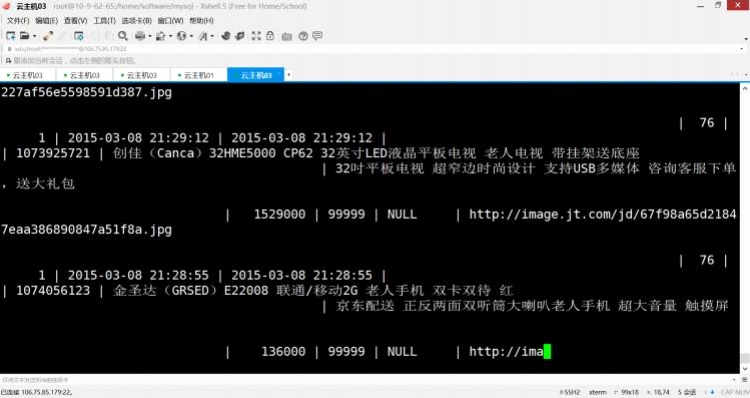
由于wget过来的文件名字不一样,需要修改成jt.sql

mysql>source jt.sql;

检查数据是否正确

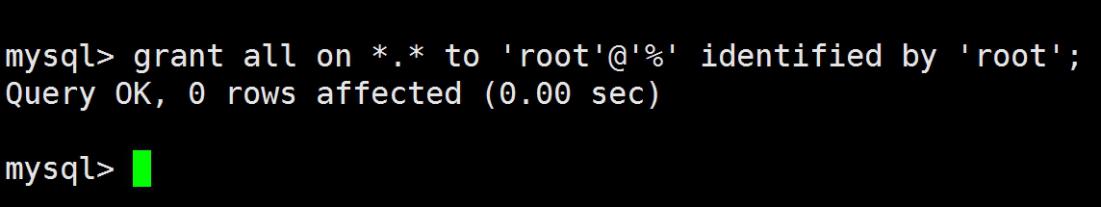
mysql>use jtdb;

mysql>select \* from tb\_item;

默认安装的mysql没有外部访问权限,所以需要开启访问权限

grant [权限] on [数据库名].[表名] to ['用户名']@['web服务器的ip地址'] identified by ['密码']; "%"表示所有10.0.0.1这样格式的ip地址 10-0-0-1

mysql>grant all on \*.\* to 'root'@'%' identified by 'root';

然后根据以上的过程安装mysql到第二和第三台云主机中

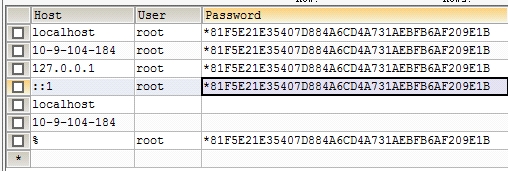
分区 day04all 的第 49页

然后根据以上的过程安装mysql到第二和第三台云主机中

mycat登录时，使用的登录命令类似如下：

mysql -h 10-9-19-3 -P 3306 -u root -p root

在数据库的mysql中的user表格如果没有对应的等权限，mycat登录失败

将登陆的客户端中mysql.user表格所有对应root用户的数据记录的password全部覆盖成'root'的加密文本

配置主从关系

2017年12月13日 21:54

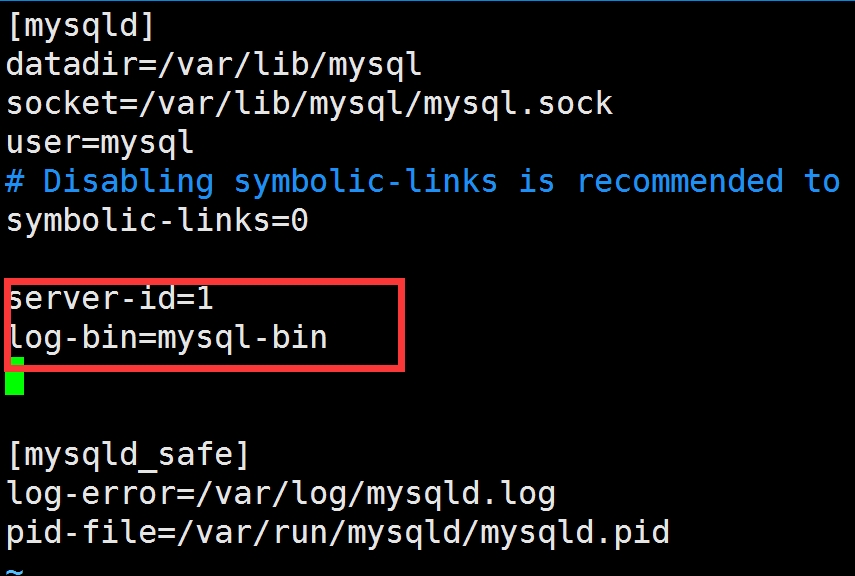
**配置主服务器（开启二进制日志文件）**

编辑主master服务器配置文件/etc/my.cnf

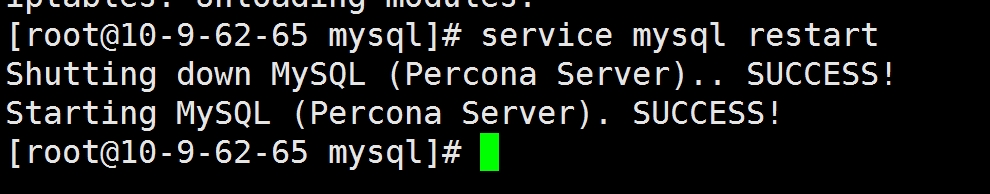
在[mysqld]节点下加入两句话

server-id=1

log-bin=mysql-bin #启用二进制日志,二进制的日志文件名mysql-bin；

重启服务：service mysql restart

在启动或者重启mysql服务的过程中会有一些常见问题,可以参考附录3--mysql常见问题解决

登录mysql：mysql –uroot -proot

mysql>flush tables with read lock; #数据库锁表，不让写数据;这步骤可不做

对于当前环境的mysql无需使用lock命令,因为没有人操作,但是生产环境中必须这样做

mysql>show master status; #查看MASTER状态（这两个值File和Position）其中的file 就是二进制文件,position记录当前操作sql的步骤数(注意一条sql包含多步,所以不是sql语句的条数)

mysql>unlock tables; #从启动好后，记得要解除锁定

主数据库到此配置完毕

**其中file:二进制的文件名称,以我们定义的mysql-bin命名二进制文件,序号表示会滚动,重启服务器**

分区 day04all 的第 51页

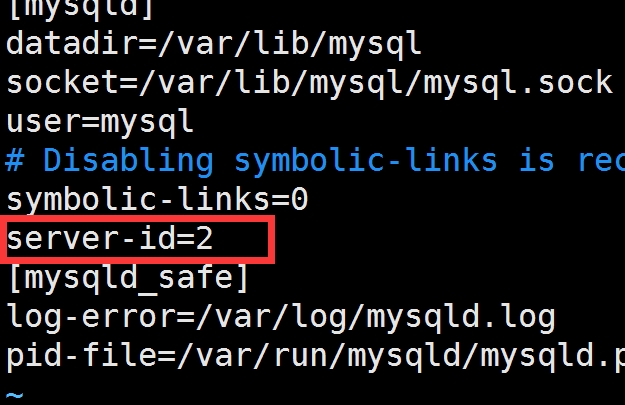
**其中file:二进制的文件名称,以我们定义的mysql-bin命名二进制文件,序号表示会滚动,重启服务器 时文件序号+1;**

**position:120,表示初始化状态的当前主节点的写操作的最终指针位置;**

**配置从服务器**

修改/etc/my.cnf增加一行

server-id=2

重启服务

service mysql restart

**通过mysql命令配置同步日志的指向：（类似于redis的slaveof挂接主从）**

mysql>change master to master\_host='106.75.74.254', master\_port=3306,

master\_user='root',master\_password='root',

master\_log\_file='mysql-bin.000001',

master\_log\_pos=120;

master\_host 主服务器的IP地址

master\_port 主服务器的PORT端口

master\_log\_file 和主服务器show master status中的File字段值相同

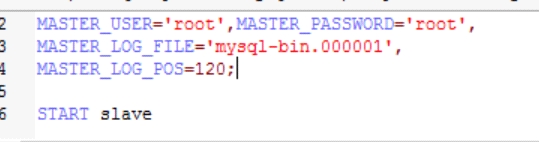
master\_log\_pos 和主服务器show master status中的Position字段值相同

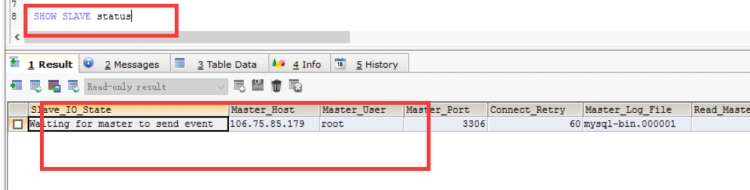
mysql>start slave; #stop slave;停止服务，出错时先停止，再重新配置

mysql>show slave status\G; #查看SLAVE状态，\G结果纵向显示。必须大写,这个命

令无法再sqlyog中使用

service mysql restart #重启服务



注意：如果出错，可以看后面的错误信息。观察Slave\_SQL\_Running\_State字段，它会记录详细的错误信息

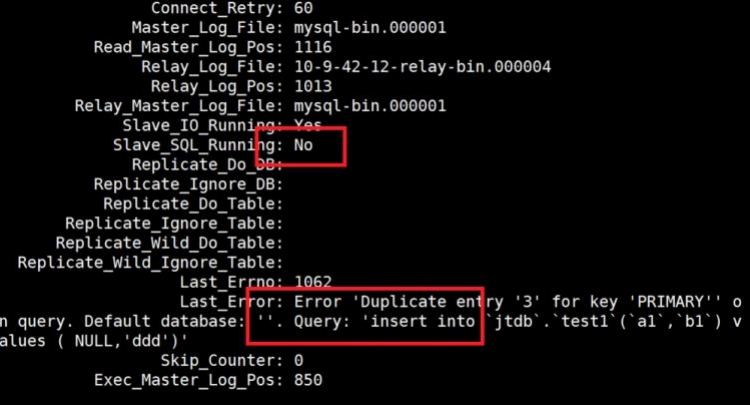
**测试同步状态**

案例1:在主中创建表格,插入数据

观察从

案例2:将从节点中插入数据,然后在主里继续添加数据

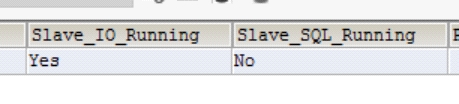
观察主,从状态

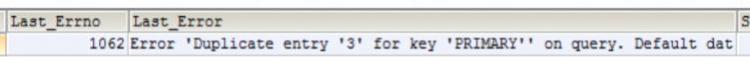
案例3:对第二个案例的数据在主中进行变更,

观察从

由于第二步操作主从结构失效

这是发现没法同步,调用show slave status 发现已经报错

sql线程已经不工作了

id为3的重复,在从中有数据了

重新挂接（不能够轻易的在单机热备的从节点中操作写）

错误数据必须清除或者调整正确否则继续主从失效

查看主节点中的二进制文件名称 pos

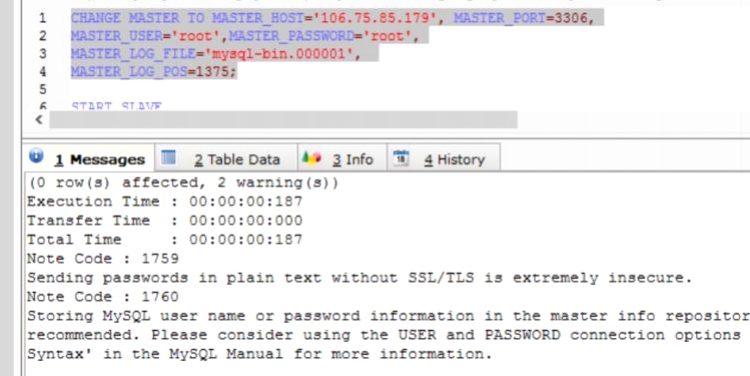
停止从节点的从状态

stop slave

分区 day04all 的第 53页

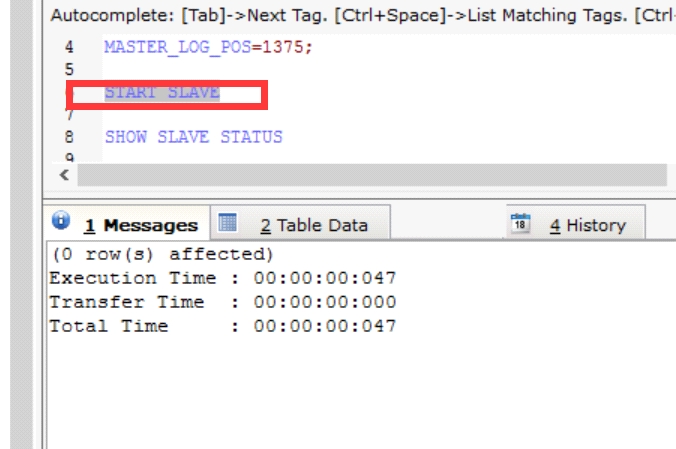
show master status;

在从节点中把查询出来的最新数据放到命令里挂接主节点

启动从节点的slave

start slave

分区 day04all 的第 54页

但是这个时候发现id为3的对应b1字段的值没有改

所以mysql虽然支持主从关系但是并没有维护读写分离的状态

作业:双向主从的配置,

测试宕机其中一台,在另一台写数据,观察重启后的恢复数据库有没有新的数据;

mycat数据库中间件

2018年9月15日 9:15

Mycat介绍

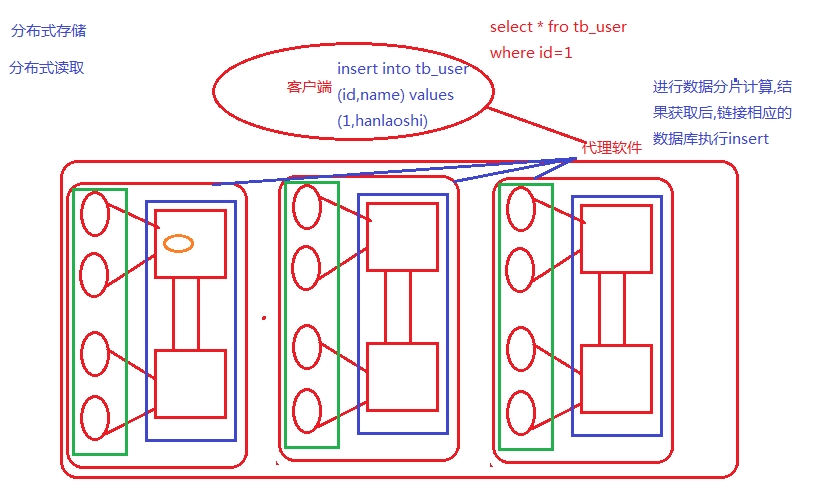
彻底的开源的面向企业应用开发的大数据库集群,性能极高,京东,淘宝,双11.

Mycat的原理

代码客户端链接mycat,mycat中配置数据库集群(复杂的可以支持高可用的主从集群),客户端的sql语句传给了mycat,中间件就会根据配

置的分布式读写分离的高可用集群来讲sql语句进行拼接拦截,重新

整理,然后发送给对应的数据库去执行sql语句;

mycat数据库中间件的特性

1 高性能的读写分离

每个数据分片都 对应有2个标签writeHost,readHost

前者只能指向写操作的数据库,readHost可以指向读操作的数据库

1. 100亿级别的大表水平分片,集群并行计算,性能不会有丝毫影响

分区 day05all的第 2页

1. 整合资源的能力

数据来源,种类繁多

mycat的安装和测试

2018年9月15日 9:35

Mycat的安装和测试启动

java编写的，需要依赖jdk环境，需要mysql的lib包支持

命令是通过mysql进行操作；可以在使用mycat的环境安装 mysql，还可以在mycat的环境配置好mysql的内容

上传mycat包，解压运行

# tar -xf Mcat\*\*\*\*.tar.gz

目录结构：

bin： mycat就是命令文件

catlet：三方插件引入位置（plugin)

lib：依赖的jar包

conf ：server.xml,schema.xml

logs：wrappers.log 记录了启动的信息

启动测试

bin目录中，执行mycat console（successfully） mycat start日志不需要打印到控制台,会写入一个

logs/wrapper.log(报错时,常用,报错信息非常明确)

分区 day05all 的第 4页

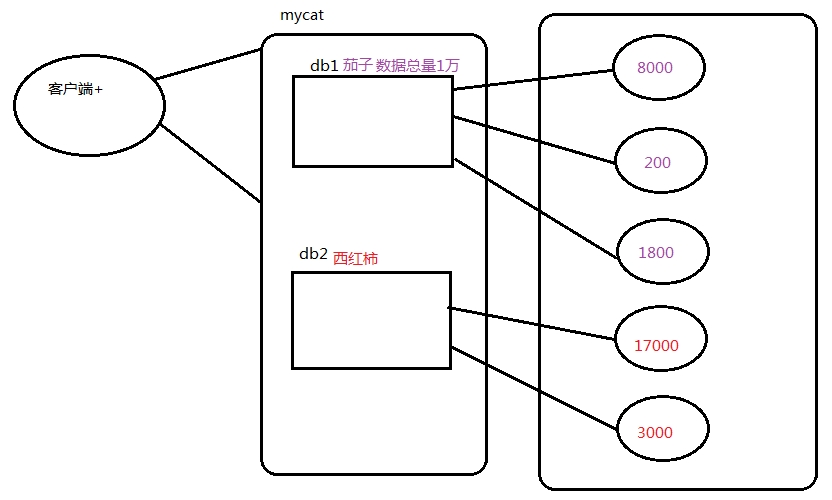
mycat的核心概念与配置文件

2018年9月15日 9:39

mycat的核心概念

逻辑库: 基于分布式读写分离的数据库集群,数据可以来源于不同的数据库节点,客户端链接mycat,看到的结构只有一

个逻辑库;

逻辑库,在mycat配置文件中,<schema>的标签标示

逻辑表(逻辑的数据分片的细化)

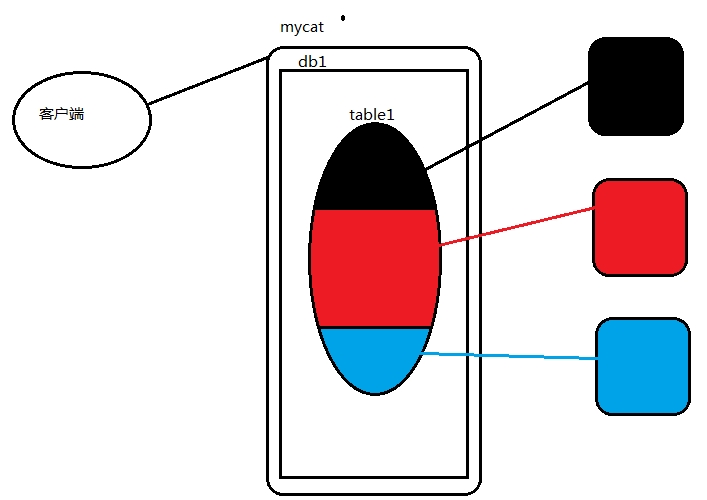
根据逻辑表的特性和数据来源

分片表: 当数据量非常大时候,当前逻辑表(10亿)大量数据会 被切分(数据分片),到多个后台的数据库中的真实表格存储,

所有的分片集合到mycat对客户端进行展示时,能展示一张

特别大的表格----分片表.

分区 day05all 的第 5页

企业中存储业务数据的表格基本都需要分片表(因为大),比 如京东淘宝,的商品信息,物流的用户信息;

非分片表

某些固定数据量(变动不大)的表格,不需要后台多个数据库分片存储.mycat可以设计一种非分片表格,只需要数据库结

构保证当前这些数据可靠(双机热备);

企业中非常多的使用非分片表示在一些:

工具表格

字典表格

信息表格

工具字典表:

系统中非常多的数字代表含义的情况:

http协议500:异常 404:找不到页面

在工具字典表中记录这些信息,一旦需要可以利用数据查询

数字 含义

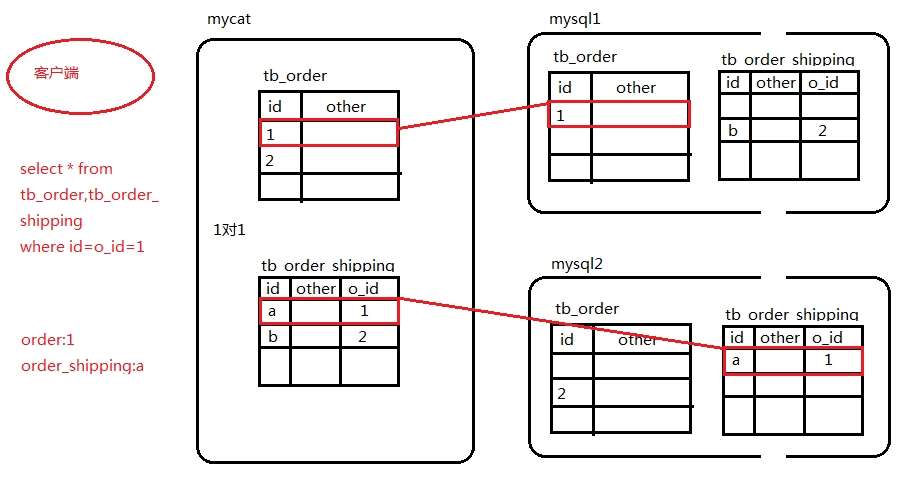
1292712763 明天下午不开会

分区 day05all 的第 6页

ER表:

分片表格的一种,涉及到表格的关系体现时一种逻辑表格的 结构.为了在多张表格关联查询时,在底层mycat尽可能少做

跨库的查询而出现的一种结构.

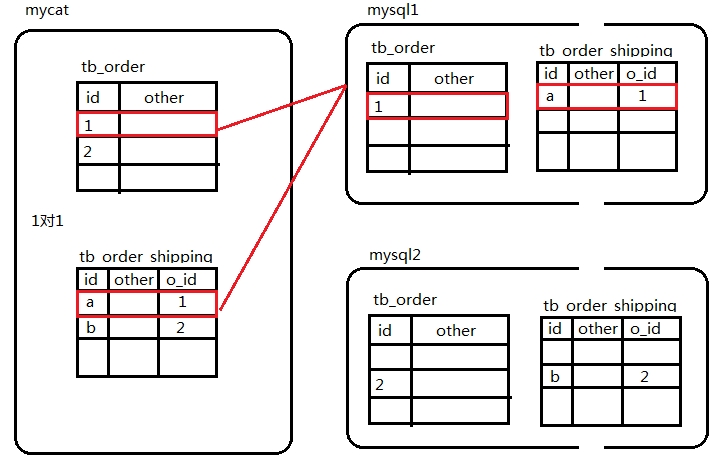
图中的关系,a表示的物流信息,在关联查询时需要同1号订单同时查询出来,由于分片表涉及导致分片相关信息存储在了 不同的数据库内容,mycat底层就要在拦截sql语句后做跨库操作,如果这种关联查询非常频繁,mycat虽然具备翻译跨库操作的能力,但是尽量还是要避免;mycat对于此种情况就设 计了ER分片表

ER分片表,设计了table(主表)和childtable(子表从表),根据配置的关系,将相关联的2个条数据在切分时,尽最大可能存放

到同一个数据库,下次的关联查询会直接从相同的一个数据

库直接后去2个相关数据记录,从而避免跨库操作;

分区 day05a



mycat中的配置文件

2018年8月15日 16:54

在conf中有两个核心的配置文件，配置内容实现：

读写分离，高可用替换，分布式存储，ER分片表的逻辑。

server.xml：配置当前mcyat启动的资源设定（端口占用等）

用户登录信息的设置

schema.xml：逻辑库，逻辑表，分片表，ER表的配置，底层链接 的实际数据库信息。

server.xml

包括2个重要标签

system:配置mycat启东时占用的系统的一些资源属性（注释掉的不 需要动，保持默认）

user标签：指定当前登录的用户信息；

**属性：name表示登录当前的mycat的用户名**

**password表示登录密码**

**schemas表示当前这个用户可以查看的逻辑库，以“,” 分割,这里的名称必须和schema.xml中的schema标签名 称对应**

readOnly 表示当前用户只读，不能操作写quarantine标签：安全措施可以设置白名单黑名单

子标签

whitehost：白名单列表

blacklist：黑名单

修改为如下内容：

<user name="root">

<property name="password">root</property>

分区 day05all的第 9页

<property name="password">root</property> <property name="schemas">mytest</property>

</user>

schema.xml:当前mycat代理的数据库集群的设定

逻辑库，分片/非分片表，ER表，读写分离，高可用替换

标签结构

**schema**：逻辑库

**table**：逻辑表

childetable：逻辑表的从逻辑表

**datanode**：节点信息的收集，是逻辑表**/**逻辑库对应真实节点的过 度**(**一个数据分片的来源**)**

**datahost**：管理一个数据分片的具体标签**,**对于连接数据库**,**操作特

性**,**管理权限**,**都是在这个标签控制的

**heartbeat**:心跳检测(select user())

**writeHost**:写操作的数据库节点（也可以读）

**read**Host:与当前writeHost相关的（主从）读节点（只

读）

每个标签的属性

**schema:**

name="TESTDB" checkSQLschema="false"

sqlMaxLimit="100"

name：逻辑库的名称(需要和实际的数据库同名)

checkSQLschema: 查询sql语句的逻辑库名称携带.例如sql语句

“select \* from tb\_item”,mycat拦截，将其表格名称根据当前属 性的设定进行修改拼接逻辑库名字\*.tb\_item，false不会自动检查

逻辑库名称，不拼接逻辑库名字“select \* from tb\_item”,可能会 导致无法找到逻辑库执行后续逻辑，true自动检测，会拼接逻辑库 名字“select \* from TESTDB.tb\_item”

sqlMaxLimit：执行查询语句时，自动配置的分页数据，只查前

100条，在没有limit条件时,拼接会根据方言不同使用不同的拼接方

分区 day05all的第 10页

100条，在没有limit条件时,拼接会根据方言不同使用不同的拼接方 式,mysql limit 起始位置,条数.

oracle rows

**table:逻辑表**

<table name="travelrecord" dataNode="dn1,dn2,dn3" primaryKey="id",rule="auto-sharding-long" />

name:逻辑表名称，配置和真实数据库的表名最好相同

dataNode:对应一个，多个数据库节点的标签，多个值表示分片， 单个值表示非分片

rule:数据分片的计算规则，auto-sharding-long，利用long整型

的id值进行范围划分（0-500万，500万-1000万）

primaryKey:指向表格的主键，最好与真实字段一致

**childTable:从表(ER分片表相关)**

<childTable name="orders" primaryKey="ID"

joinKey="order\_id"

parentKey="id">

name：从表名称，保持和数据库真实表格一致

primaryKey：主键名称

jointKey:从表中的外键字段；必须和真实数据库的从表外键字段名 一致

parentKey:主表的字段名，必须和主表与从表的外键相关，对应主

表的字段名称

**datanode标签：数据节点**

name:表示名称，逻辑库，逻辑表指向的依据

dataHost: 指向真实管理数据库的标签名称dataHost

database: 真实数据库名称，逻辑库和所有真实数据库名称一致

分区 day05all的第 11页

database: 真实数据库名称，逻辑库和所有真实数据库名称一致

datahost：管理数据分片的真实数据库的标签，内部包含真实数据 库的信息

<dataHost name="localhost1" maxCon="1000"

minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

name:名称，与datanode指向对应

maxCon:底层数据库的链接最大数

minCon：相反

**balance:值是0,1,2,3,分别表示对当前datahost中维护的数据库门的读写操作逻辑**

**0:不开启读写分离，所有的读写操作都在最小的索引号的**

**writeHost(第一个writeHost标签)**

**1：全部的readHost和备用writeHost都参与读数据的平衡，如果 读的请求过多，负责写的第一个writeHost也分担一部分**

1. **：所有的读操作，都随机的在所有的writeHost和readHost**

**中进行**

1. **：所有的读操作，都到writeHost对应的readHost上进行（writeHost不参加了）**

**writeType:控制当前datahost维护的数据库集群的写操作**

**0：所有的写操作都在第一个writeHost标签的数据库进行 1：所有的写操作，都随机分配到所有的**

**writeHost（mycat1.5完全不建议配置了）**

dbtype：数据库类型（不同数据库配置不同名称，mysql,oracle） dbDriver:数据库驱动，native,动态获取

分区 day05all的第 12页

**switchType：切换的逻辑**

**0：故障不切换**

**1：故障切换，当前写操作的writeHost故障，进行切换，切换到下一个writeHost；**

**slaveThreshold：判断主从间的数据是否同步；100同步的**

**gap(缝隙).读的时候判断Last\_IO\_Error\_Timestamp时间戳,值越大,表示从节点与主节点的延迟越高**

配置读写分离和故障切换

2018年9月15日 10:25

明显观察出读写分离的效果,需要将数据库个数多一点,数据不能同步

主要配置的是scheme.xml文件

<mycat:schema xmlns:mycat="http://org.opencloudb/" >

<schema name="mytest" checkSQLschema="true" sqlMaxLimit="100" dataNode="dn1">

</schema>

<dataNode name="dn1" dataHost="localhost1" database="mytest" />

<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="106.75.120.140:3306" user="root"

password="root">

<!-- can have multi read hosts --> <!--<readHost host="hostS2"

url="192.168.1.200:3306" user="root" password="xxx" />-->

</writeHost>

<writeHost host="hostM2" url="106.75.120.155:3306" user="root" password="root" />

<!-- <writeHost host="hostM2" url="localhost:3316"

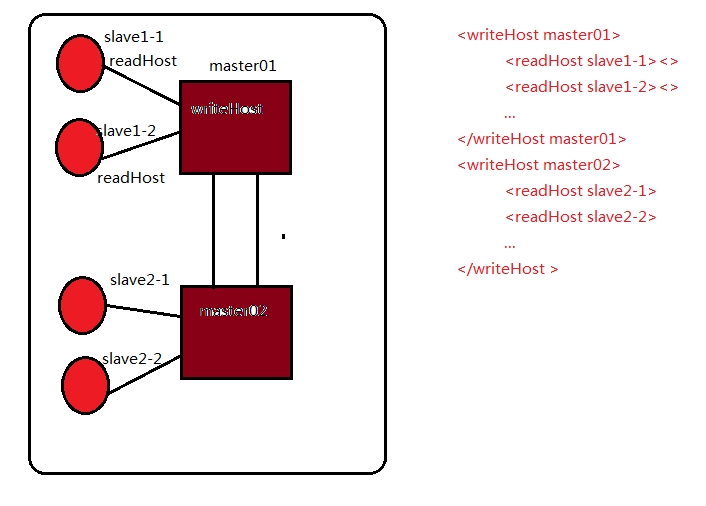
user="root" password="123456"/> -->

</dataHost>

</mycat:schema>

分区 day05all 的第 14页

相继修改balance的值,查看读数据的变化

分区 day05all 的第 15页

配置分布式分片表

2018年9月15日 14:25

**table:逻辑表**

<table name="travelrecord" dataNode="dn1,dn2,dn3" primaryKey="id",rule="auto-sharding-long" />

name:逻辑表名称，配置和真实数据库的表名最好相同

dataNode:对应一个，多个数据库节点的标签，多个值表示分片， 单个值表示非分片

rule:数据分片的计算规则，auto-sharding-long，利用long整型

的id值进行范围划分（0-500万，500万-1000万） primaryKey:指向表格的主键，最好与真实字段一致

利用2个独立的数据库中同名的表格,student,完成范文分片计算的数 据分片表格的配置

<schema name="mytest" checkSQLschema="true"

sqlMaxLimit="100" dataNode="dn1">

<table name="student" primaryKey="id"

dataNode="dn1,dn2" rule="auto-sharding-long"/> </schema>

<dataNode name="dn1" dataHost="localhost1" database="mytest" />

<dataNode name="dn2" dataHost="localhost2"

database="mytest" />

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native"

switchType="1" slaveThreshold="100"> <heartbeat>select user()</heartbeat>

分区 day05all 的第 16页

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1"

url="106.75.120.140:3306" user="root"

password="root">

</writeHost>

</dataHost>

<dataHost name="localhost2" maxCon="1000"

minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM1"

url="106.75.120.155:3306" user="root"

password="root">

</writeHost>

</dataHost>

对应范围分片

范围的计算结果在autopartition-long.txt文件中保存可以指定范 围的数字

还有取模范围分片(字符串取余运算),按日期分片(与范围分片一

致)

在id的取值类型不同时,mycat支持各种分片逻辑

峰会某个题目的一部分内容;

发现mycat提供的分片计算逻辑都不适用,可以自定编写计算代 码

分区 day05all 的第 17页

码

在rule.xml中配置计算分片逻辑与计算方法的对应关系,自行编写代码,封装jar包,根据class全路径,指定你自定义的class来计算

分片

<function name="murmur"

class="org.opencloudb.route.function.PartitionByMurmur Hash">

<property name="seed">0</property> <property name="count">2</property>

<property name="virtualBucketTimes">160</property>

ER分片表

2018年9月15日 14:46

**childTable:从表(ER分片表相关)**

<childTable name="orders" primaryKey="ID"

joinKey="order\_id"

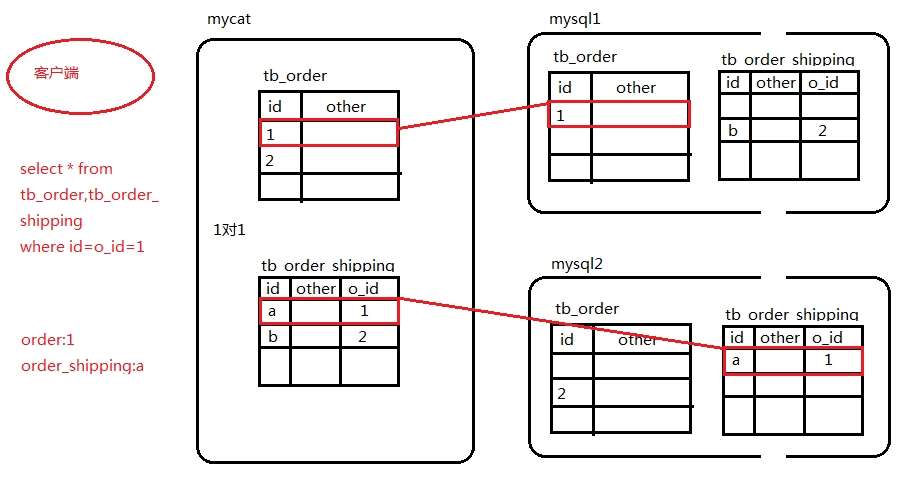
parentKey="id">

name：从表名称，保持和数据库真实表格一致 primaryKey：主键名称

jointKey:从表中的外键字段；必须和真实数据库的从表外键字段名 一致

parentKey:主表的字段名，必须和主表与从表的外键相关，对应主

表的字段名称

存在相关性的2个表格订单表,一个是订单关联的物流表,1对1的关

系,单从每张表格的特点,都是大数据表,需要切分,每张表都是分片表; <schema name="mytest" checkSQLschema="true"

sqlMaxLimit="100" dataNode="dn1">

<table name="student" primaryKey="id"

dataNode="dn1,dn2" rule="auto-sharding-long"/> <table name="tb\_order" primaryKey="id"

dataNode="dn1,dn2" rule="auto-sharding-long">

<childTable name="tb\_order\_shipping"

分区 day05all的第 19页

dataNode="dn1,dn2" rule="auto-sharding-long">

<childTable name="tb\_order\_shipping" primaryKey="id" joinKey="order\_id"

parentKey="id"/>

</table>

</schema>

<dataNode name="dn1" dataHost="localhost1"

database="mytest" />

<dataNode name="dn2" dataHost="localhost2"

database="mytest" />

<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="106.75.120.140:3306" user="root"

password="root">

<!-- can have multi read hosts -->

<!--<readHost host="hostS2"

url="192.168.1.200:3306" user="root" password="xxx" />-->

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="localhost2" maxCon="1000"

minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native"

分区 day05all的第 20页

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1"

url="106.75.120.155:3306" user="root"

password="root">

<!-- can have multi read hosts -->

<!--<readHost host="hostS2"

url="192.168.1.200:3306" user="root" password="xxx" />-->

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

Mycat后续内容

2018年9月15日 15:21

工程与mycat整合

修改配置文件中的链接数据库的地址ip:8066/mytest

mycat高性能的数据库中间件存在的问题

数据库集群高可用,分布式的.管理数据库集群的mycat中间件是单节点;不存在mycat程序的高可用

能否在启动另外一个mycat管理同一个数据库集群?

为了完成mycat端的高可用

分区 day05all 的第 22 页