# 实现数据库高可用

## 数据库备份

### 数据库冷备份

说明:将数据库中的数据定期转化为sql文件进行保存.

缺点:1.数据备份不及时

2.容易丢失数据.

行业特点:虽然数据库冷备份方式容易造成数据的丢失.但是一般最为恢复数据最后有效的手段一直使用.

### 数据库热备份

特点:可以实现数据库的实时的同步!



备份步骤:

1. 当主库发生数据修改时,会将修改数据的sql语句记录到二进制日志文件中.这时主库的任务完成.
2. 从库利用IO线程读取主库的二进制日志文件信息.将数据写入从库的中继日志中.
3. 从库中开启sql线程读取中继日志,将数据最终同步到从库.

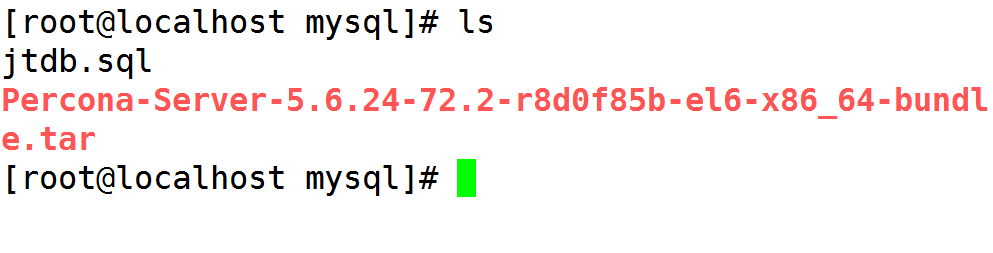
## 数据库搭建

### 准备文件夹

说明:在/usr/local/src中创建mysql文件夹

mkdir mysql

### 上传安装文件



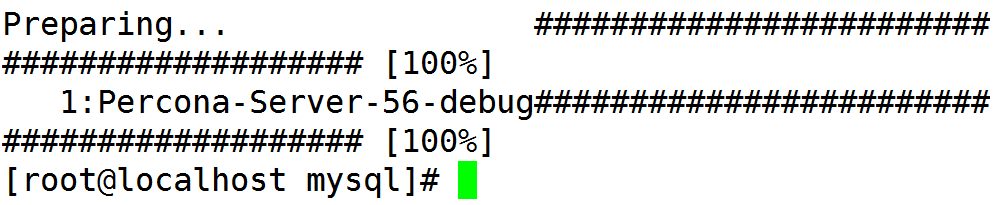
### 安装mysql数据库

1. 解压文件

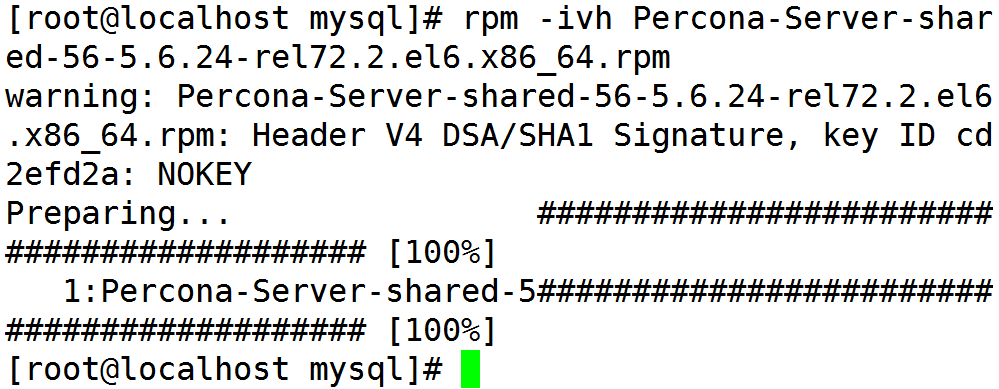
tar -xvf Percona-Server-5.6.24-72.2-r8d0f85b-el6-x86\_64-bundle.tar

1. 安装顺序
2. debuginfo----shards------client-----server
3. 安装命令

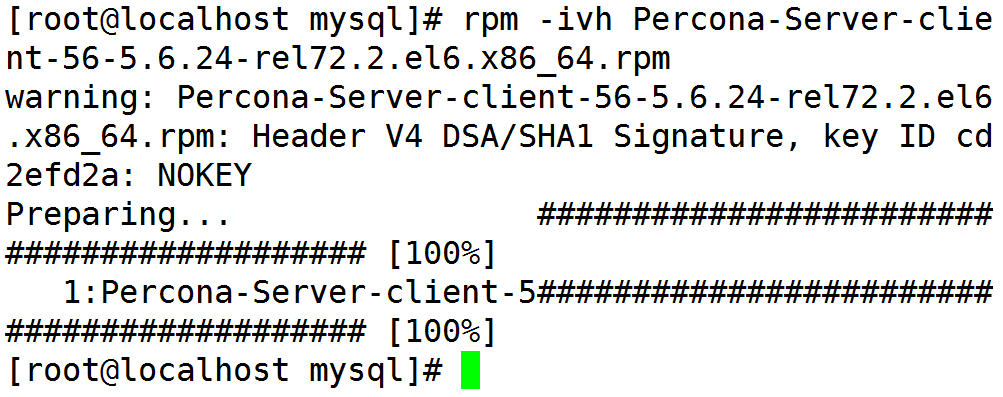
rpm -ivh Percona-Server-56-debuginfo-5.6.24-rel72.2.el6.x86\_64.rpm



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



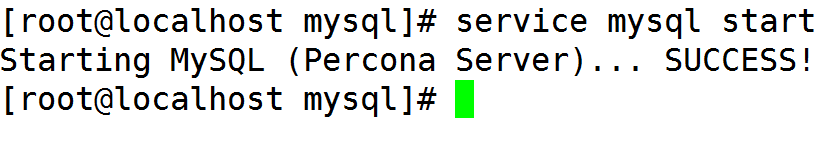
rpm -ivh Percona-Server-server-56-5.6.24-rel72.2.el6.x86\_64.rpm

### 启动mysql

启动mysql service mysql start

重启mysql service mysql restart

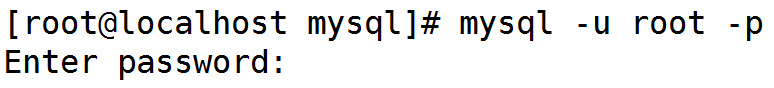
停止mysql service mysql stop



### 设定用户名和密码

执行指令:

mysqladmin -u root password root

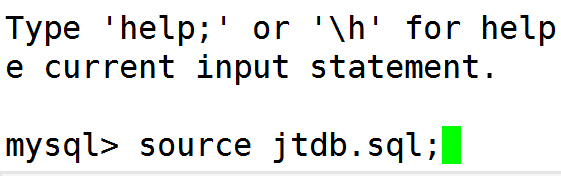
输入用户名和密码之后登陆数据库

### 导入jt数据库

在mysql的**客户端中执行指令**

命令: source jtdb.sql;

Source /usr/local/src/mysql/jtdb.sql;



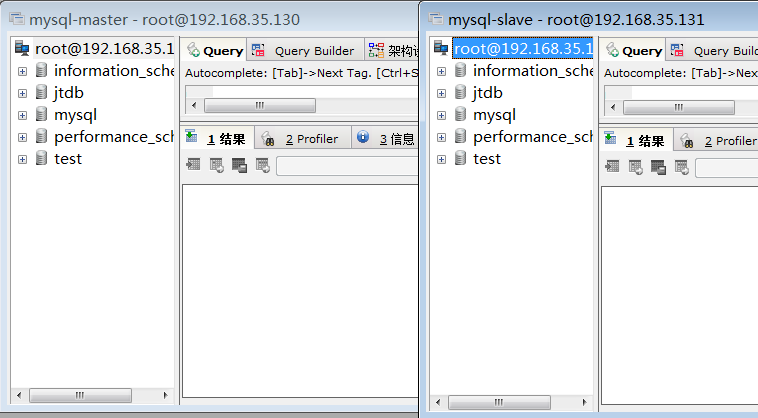
导入完成后,检查jtdb导入是否正确.

### 远程数据库链接

1. 关闭Linux防火墙
2. 开放mysql对外访问权限

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

开放权限之后.通过sqlyog工具链接

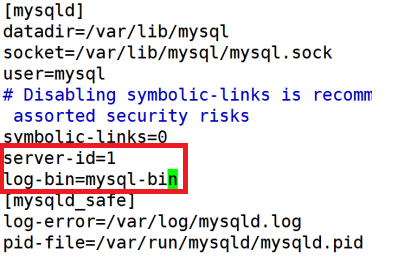


### 配置数据库二进制日志文件

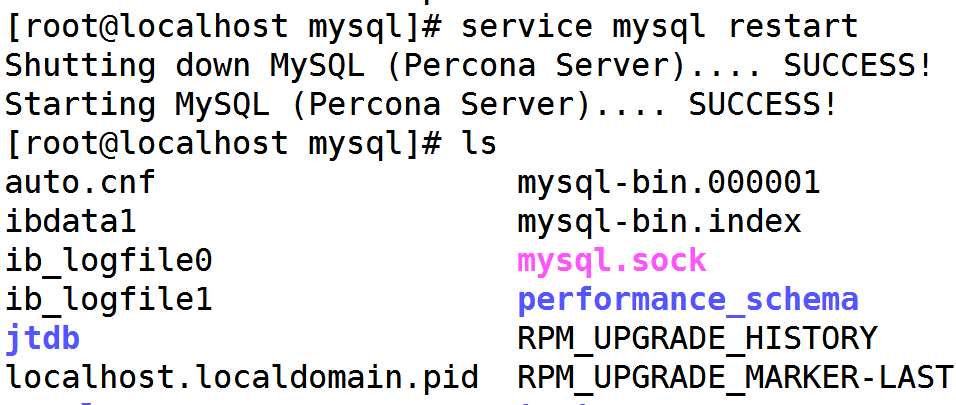
说明:需要在数据库主库和从库中分别执行该操作.

步骤:

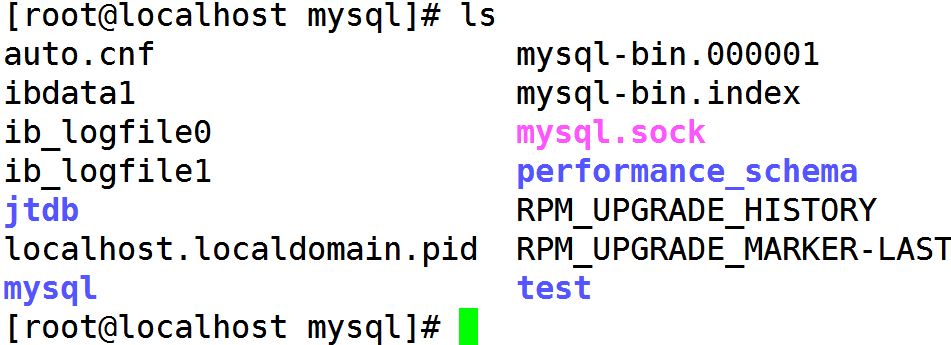
1. 修改/etc/my.cnf



1. 重启mysql数据库.



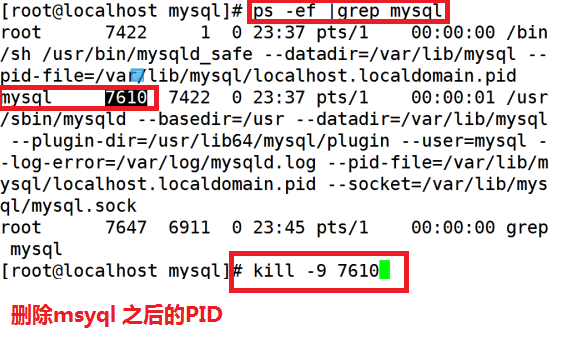
3.检查数据库二进制日志文件 如果包含mysql-bin.000001则说明数据库二进制文件启动成功!!!



### 数据库中报错检查

如果mysql数据库启动不正确.则需要手动的将mysql数据库的服务进程号关闭.之后重启mysql

ps -ef |grep mysql



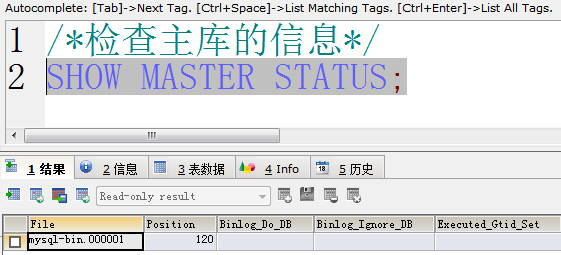
### 主从数据库挂载

主库:192.168.35.130

从库:192.168.35.131

说明:如果需要进行数据库同步,则应该在**从库中执行挂载主库的指令**

#### 获取主库二进制日志文件信息



#### 在从库中执行指令

/\*这是mysql 从库

IP/端口/用户名/密码/二进制文件名称/位置

\*/

CHANGE MASTER TO

MASTER\_HOST="192.168.35.130",

MASTER\_PORT=3306,

MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

MASTER\_LOG\_POS=120

关于位置pos的说明

1. 如果主库发生了数据的变化,则pos位置从120变化到N.这时从库读取二进制日志文件从120到N的记录之后进行同步.
2. 如果主从库需要从开始同步(pos=0),则必须**保证库与表必须一致**(数据可以没有).

#### 启动主从服务

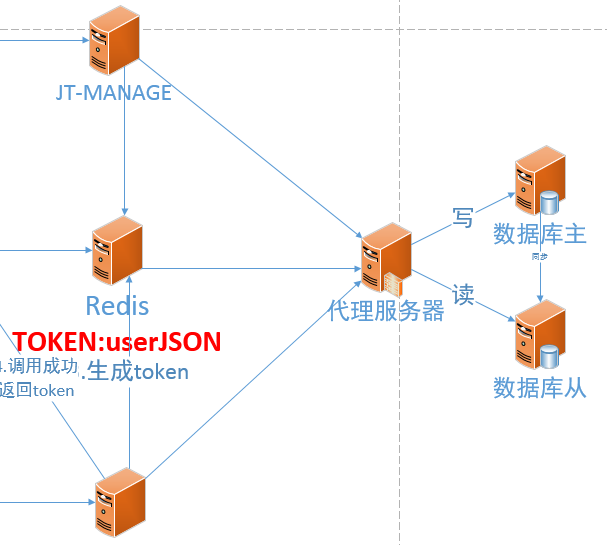


#### 主从测试

当修改主库数据时,检查从库的数据是否自动的修改.

## 实现数据库读写分离/负载均衡

### 原理



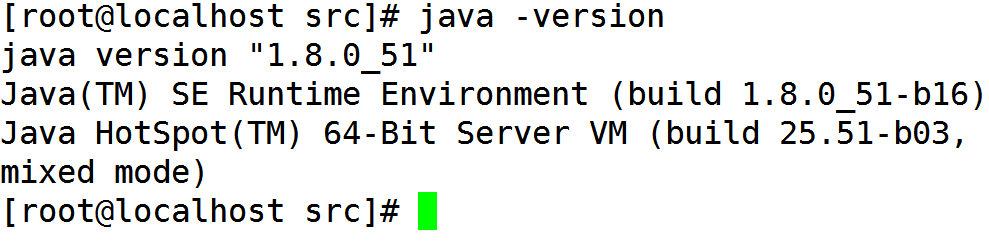
### Amoeba实现读写分离/负载均衡

Amoeba是一个以MySQL为底层数据存储，并对应用提供MySQL协议接口的proxy。它集中地响应应用的请求，依据用户事先设置的规则，将SQL请求发送到特定的数据库上执行。**基于此可以实现负载均衡、**[**读写分离**](https://baike.baidu.com/item/%E8%AF%BB%E5%86%99%E5%88%86%E7%A6%BB)**、高可用性**等需求。与MySQL官方的MySQL Proxy相比，作者强调的是amoeba配置的方便（基于XML的配置文件，用SQLJEP语法书写规则，比基于lua脚本的MySQL Proxy简单）。

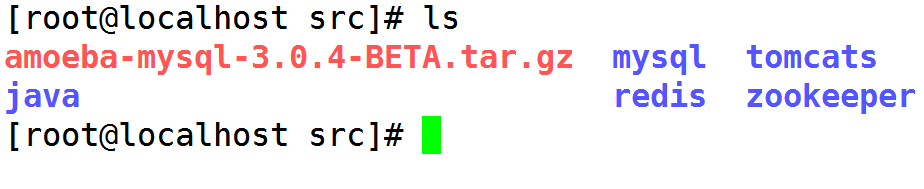
### 配置amoeba

1. 检查JDK安装和使用

Java -version



1. 上传amoeba的安装包



1. 解压安装文件

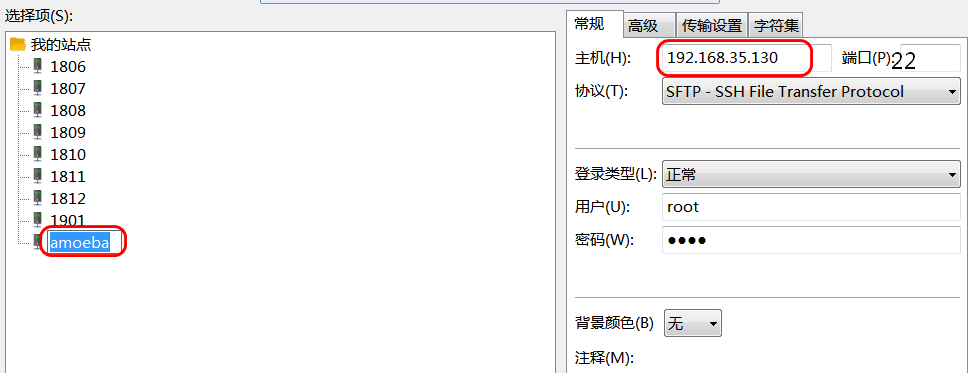
tar -xvf amoeba-mysql-3.0.4-BETA.tar.gz

1. 修改文件名称

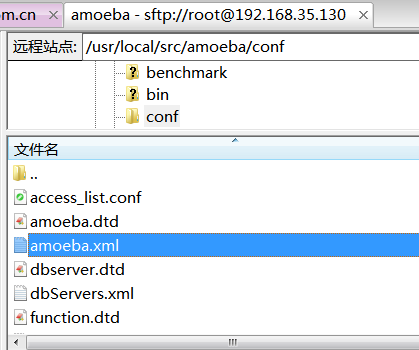


### 远程链接文件

1.链接amoeba



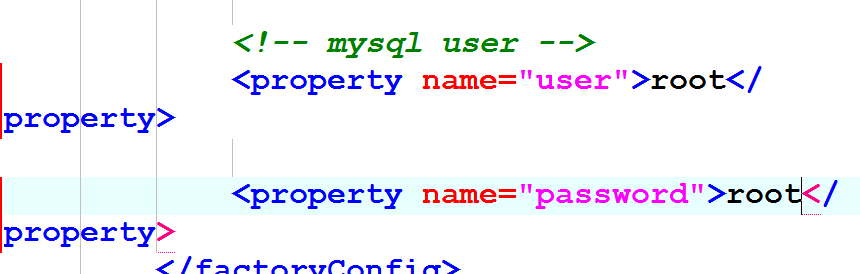
2.查看Amoeba文件信息



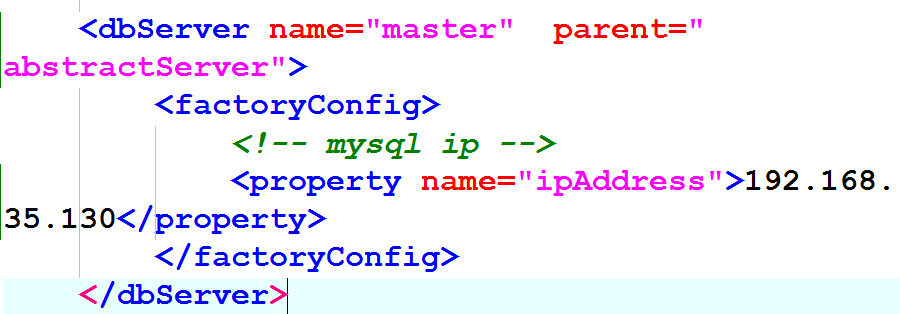
### 修改dbServers.xml

作用:amoeba链接远程mysql服务器.实现负载均衡配置

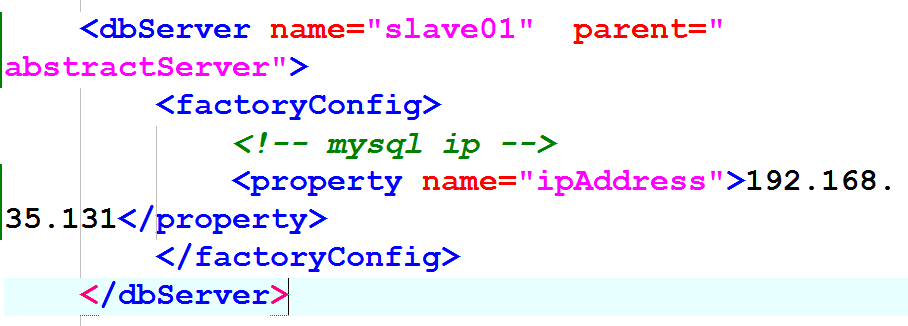
1.修改用户名和密码



2.修改主库IP信息



3.修改从库IP地址

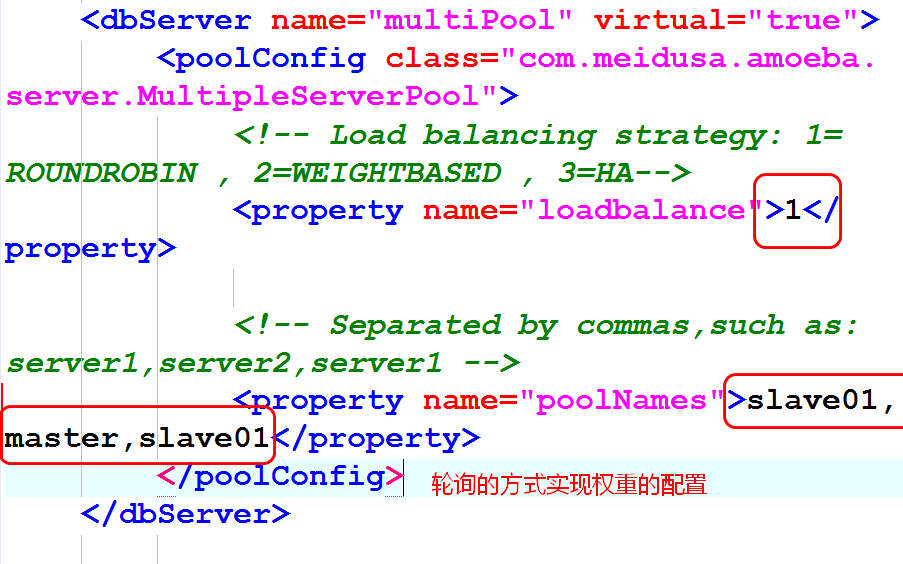


### 修改负载均衡策略

说明:在amoeba中只有**读操作需要使用负载均衡**.写库操作时不能添加该策略.因为数据库从库不能写操作.

策略:

1. 轮询 根据配置文件的顺序实现负载均衡配置.
2. 权重 如果mysql数据库响应快速则优先使用该数据库.
3. Hash 根据链接的IP地址绑定mysql数据库.



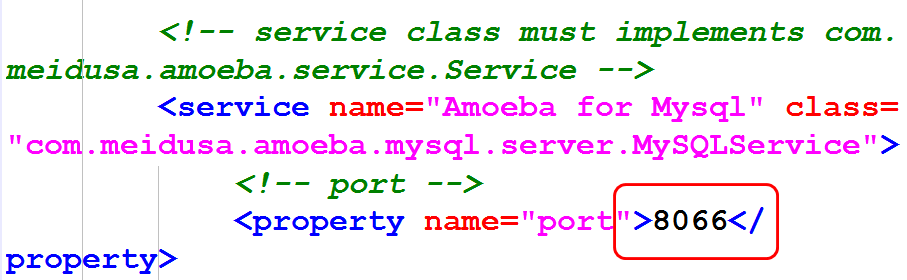
注意事项:

1. 额外注意中英文逗号问题.
2. 当文件修改完成后,将数据上传到远程服务器中.

## 编辑amoeba.xml

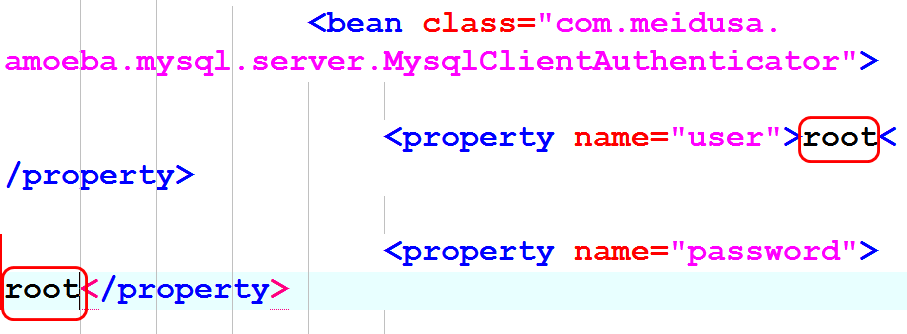
作用:用户链接代理服务器配置,完成读写分离操作

### 检查端口号

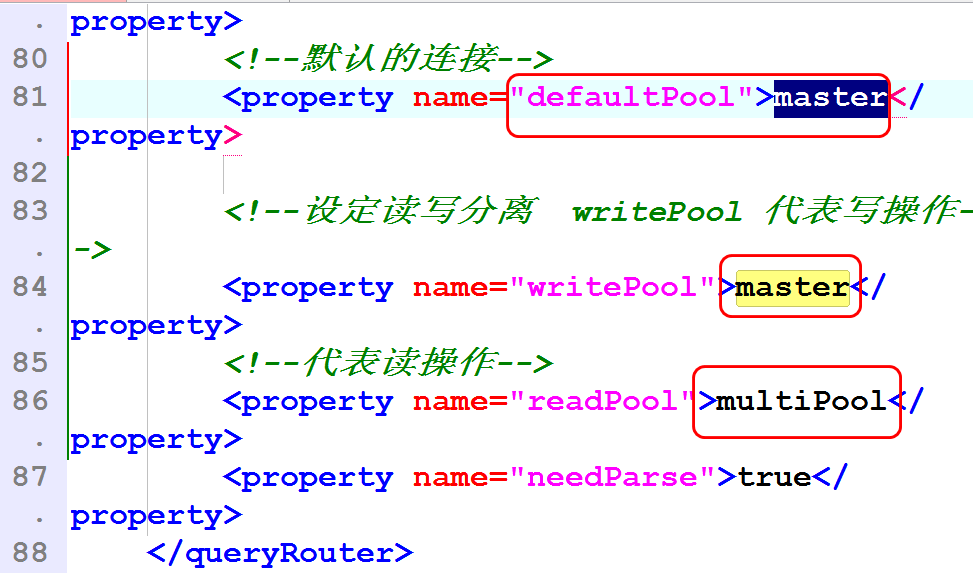


### 设定用户名和密码

说明:在amoeba.xml文件中设定用户链接的用户名和密码



设定读写分离策略



### 修改jvm.properties文件

说明:因为amoeba启动时对线程大小有特殊的要求.必须大于217kb才能执行.

JVM\_OPTIONS="-server -Xms256m -Xmx1024m -Xss256k -XX:PermSize=16m -XX:MaxPermSize=96m"

Xms256m: 代表初始化虚拟机内存大小.

Xmx1024m: jvm中运行的最大内存 1g

Xss196k jvm中单个线程大小

### Amoeba启动测试

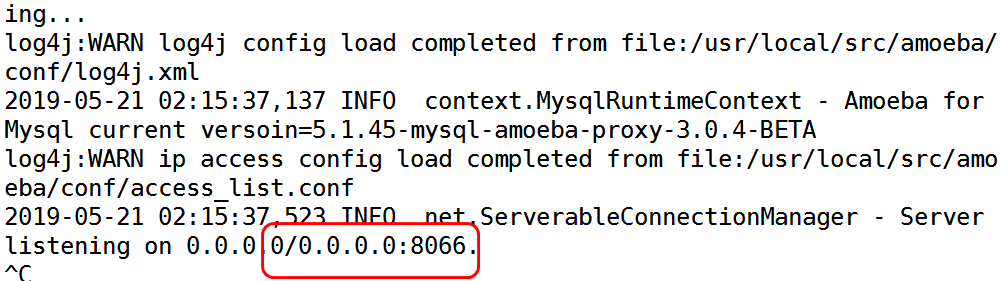
#### 命令:

./launcher 启动

./shutdown 关闭

#### 启动

如图所示,表示amoeba启动正确



#### 数据库链接

spring:

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://192.168.35.130:8066/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

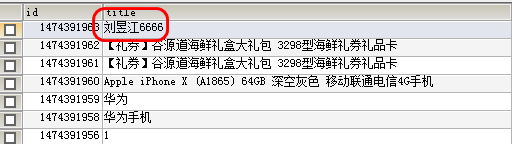
#### 数据库读写分离/负载均衡测试

测试说明:

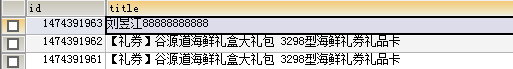
1. 将数据库进行修改,检查数据库从库数据是否与主库一致.
2. 测试负载均衡.

规则:理论上不能修改从库数据.否则IO线程会中断.为了完成测试,需要修改从库中不为主键的字段.

主库



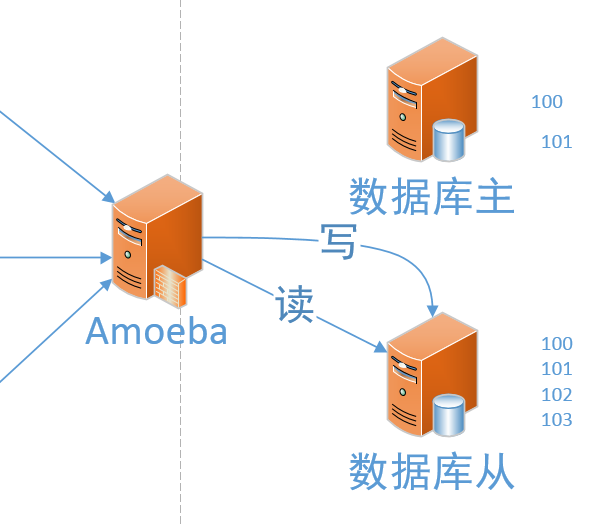
从库



**补充说明:当测试通过之后,切记将数据库数据保持一致!!!!!**

## 实现高可用存在问题说明

### 问题说明



当数据库主库宕机时,利用软件/程序可以实现故障迁移.有数据库从库代替主库提供服务.

在主库宕机的期间之内,用户全部操作的都是从库. 这时从库的数据一定比主库的数据多.

当主库有运维人员修复后继续提供服务.用户写操作继续写到主库中.

当主从同步时.主库中的数据与从库中的数据可能会出现主键冲突的现象,则主从同步的状态会立即终止.

长时间同步不能进行时会造成严重的数据不一致的问题.

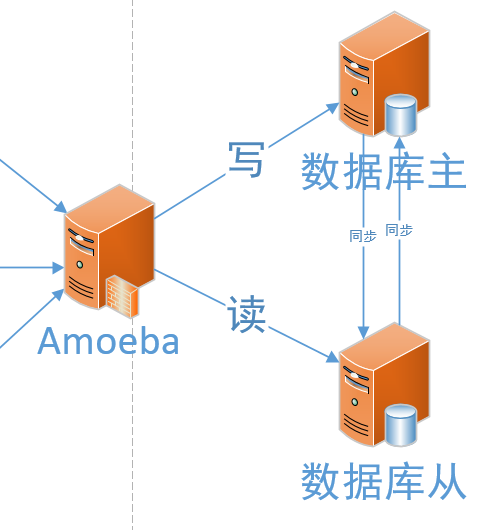
### 同步二进制文件规则

规则:只有用户操作的数据库,该数据才会写入二进制日志文件中.

同步时:从库同步主库的数据,不会写入从库的二进制文件.

## 数据库双机热备

### 概念说明



说明:在数据配置时完成双向的配置.

### 双机热备的配置

配置策略:

A:192.168.35.130 主机

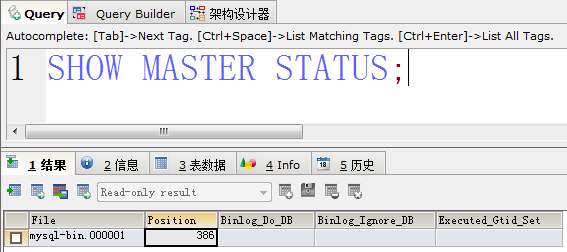
B:192.168.35.131 从机

B数据库挂载A.

A数据库挂载B.

### 查询主机的状态

说明:查询B数据库主机的状态信息.



### 实现主从挂载

说明:在A数据库中执行以下命令

/\*我是主库 现在当做从库使用\*/

CHANGE MASTER TO

MASTER\_HOST="192.168.35.131",

MASTER\_PORT=3306,

MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

MASTER\_LOG\_POS=386;

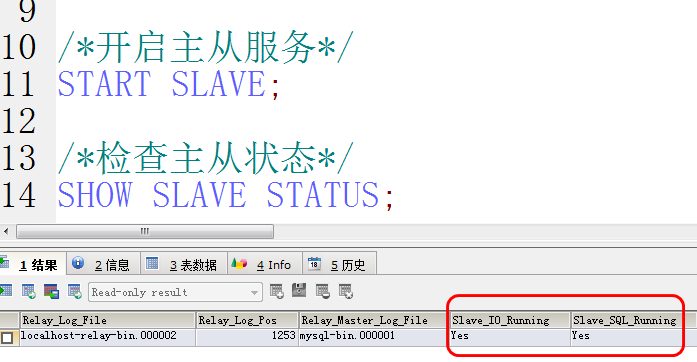
### 启动主从服务

/\*开启主从服务\*/

START SLAVE;

/\*检查主从状态\*/

SHOW SLAVE STATUS;



出现2个yes表示搭建成功.双机热备配置成功!!!

## Mycat介绍

### 说明



· 支持SQL92标准 insert/update/delte

· 支持MySQL、Oracle、DB2、SQL Server、PostgreSQL等DB的常见SQL语法

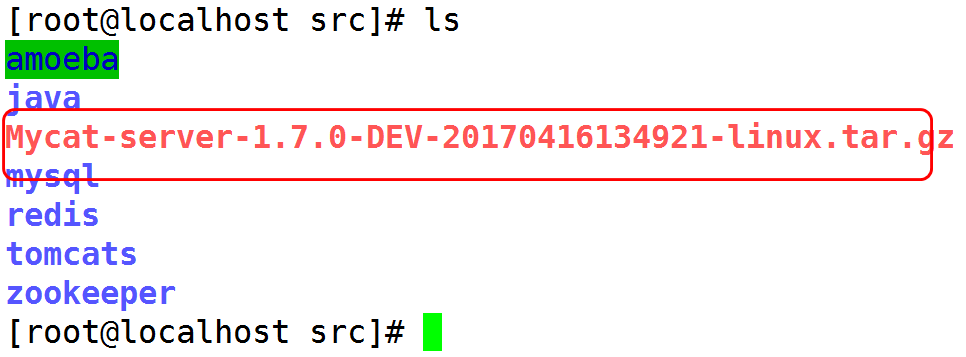
· 遵守Mysql原生协议，跨语言，跨平台，跨数据库的通用中间件代理。

· **基于心跳的自动故障切换，支持读写分离，支持MySQL主从，以及galera cluster集群**。

### Mycat安装和使用

#### 上传安装文件

说明:将mycat安装文件上传到Linux服务器中.

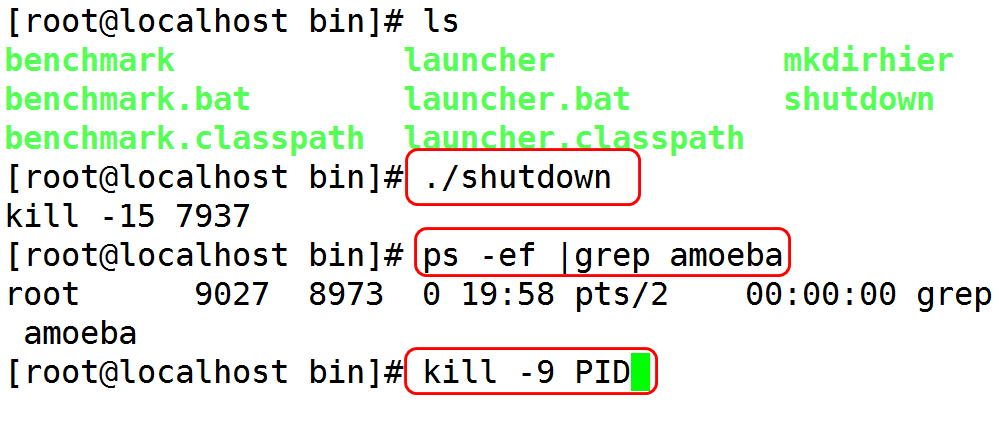


#### 解压mycat

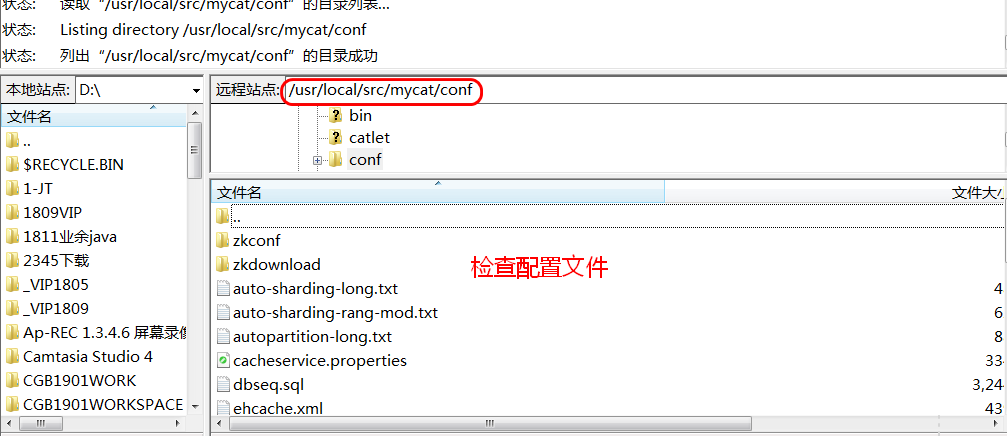
tar -xvf Mycat-server-1.7.0-DEV-20170416134921-linux.tar.gz

#### 关闭Amoeba服务

说明:因为mycat启动时会占用8066端口,所以应该将amoeba服务关闭,否则会造成端口被占用的报错.

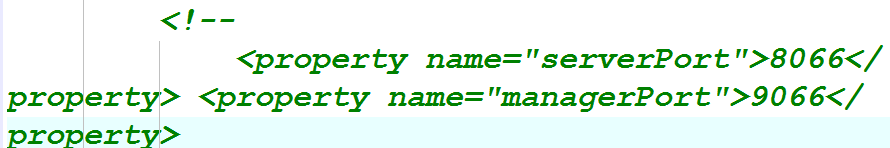


### 利用FZ工具远程链接



### 编辑Server.xml文件

1. 默认端口号8066



1. 定义用户信息

<!--用户标签-->

<user name="**root**">

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

<!--与schema.xml中的配置相同 注意数据库的大小写-->

<property name="schemas">**jtdb**</property>

</user>

<user name="user">

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

<property name="schemas">jtdb</property>

<property name="readOnly">true</property>

</user>

### 编辑schema.xml

编辑schema进行配置时.通过2个**writeHost**实现数据库高可用的配置.

当第一个writehost通过心跳检测发现宕机时,会自动的切换到第二台writeHost中.

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<!--name属性是自定义的 dataNode表示数据库的节点信息-->

<schema name="jtdb" checkSQLschema="false" sqlMaxLimit="100" dataNode="jtdb"/>

<!--定义节点名称/节点主机/数据名称-->

<dataNode name="jtdb" dataHost="localhost1" database="jtdb" />

<!--参数介绍-->

<!--balance 0表示所有的读操作都会发往writeHost主机 -->

<!--1表示所有的读操作发往readHost和闲置的主节点中-->

<!--writeType=0 所有的写操作都发往第一个writeHost主机-->

<!--writeType=1 所有的写操作随机发往writeHost中-->

<!--dbType 表示数据库类型 mysql/oracle-->

<!--dbDriver="native" 固定参数 不变-->

<!--switchType=-1 表示不自动切换, 主机宕机后不会自动切换从节点-->

<!--switchType=1 表示会自动切换(默认值)如果第一个主节点宕机后,Mycat会进行3次心跳检测,如果3次都没有响应,则会自动切换到第二个主节点-->

<!--并且会更新/conf/dnindex.properties文件的主节点信息 localhost1=0 表示第一个节点.该文件不要随意修改否则会出现大问题-->

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"

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

<!--心跳检测执行的sql语句-->

<heartbeat>select 1</heartbeat>

<!--配置第一台主机主要进行写库操作,在默认的条件下Mycat主要操作第一台主机在第一台主机中已经实现了读写分离.因为默认写操作会发往137的数据库.读的操作默认发往141.如果从节点比较忙,则主节点分担部分压力.

-->

**<writeHost host="hostM1" url="192.168.35.130:3306" user="root" password="root">**

**<!--读数据库-->**

**<readHost host="hostS1" url="192.168.35.131:3306" user="root" password="root" />**

**</writeHost>**

<!--定义第二台主机 由于数据库内部已经实现了双机热备.-->

<!--Mycat实现高可用.当第一个主机137宕机后.mycat会自动发出心跳检测.检测3次.-->

<!--如果主机137没有给Mycat响应则判断主机死亡.则回启东第二台主机继续为用户提供服务.-->

<!--如果137主机恢复之后则处于等待状态.如果141宕机则137再次持续为用户提供服务.-->

<!--前提:实现双机热备.-->

**<writeHost host="hostM2" url="192.168.35.131:3306" user="root" password="root">**

**<readHost host="hostS1" url="192.168.35.130:3306" user="root" password="root" />**

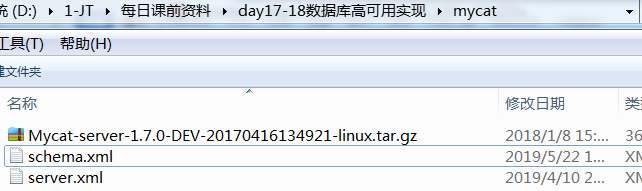
**</writeHost>**

</dataHost>

</mycat:schema>

### 上传配置文件

将mycat中原有的文件删除,将课前资料中的2个文件上传即可.



### Mycat命令

./mycat console 控制台显示

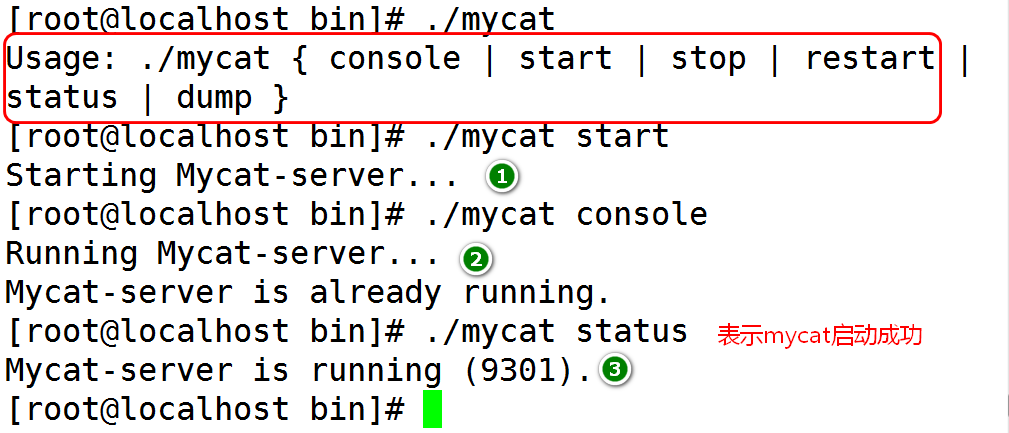
start 启动mycat

stop 停止

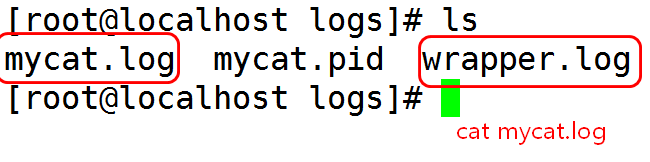
restart 重启

status 状态信息

Dump 持久化配置



### 如果程序报错



### Mycat高可用测试

测试说明:

1. 用户链接mycat检查是否正确获取数据.
2. 将数据库中的主库宕机.检查用户访问是否受限.
3. 用户操作数据,修改数据.之后重启数据库主库检查数据是否同步!!

## 数据库分库分表

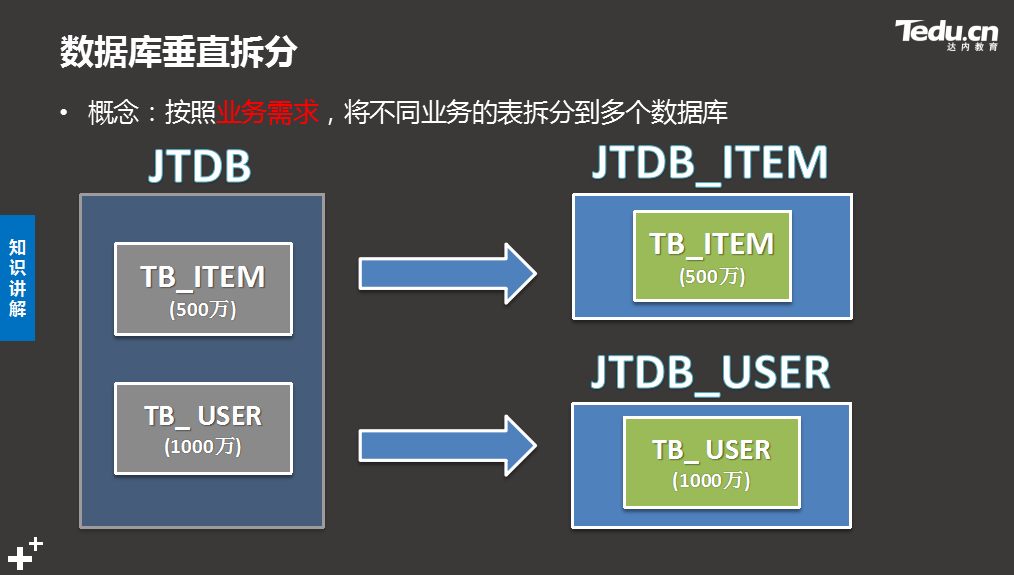
### 说明

由于数据库中保存了海量的数据.常规查询操作时速度较慢,这时采用数据库优化策略明显不能提升查询的速度时,使用分库和分表.

以上的问题称之为数据库瓶颈.

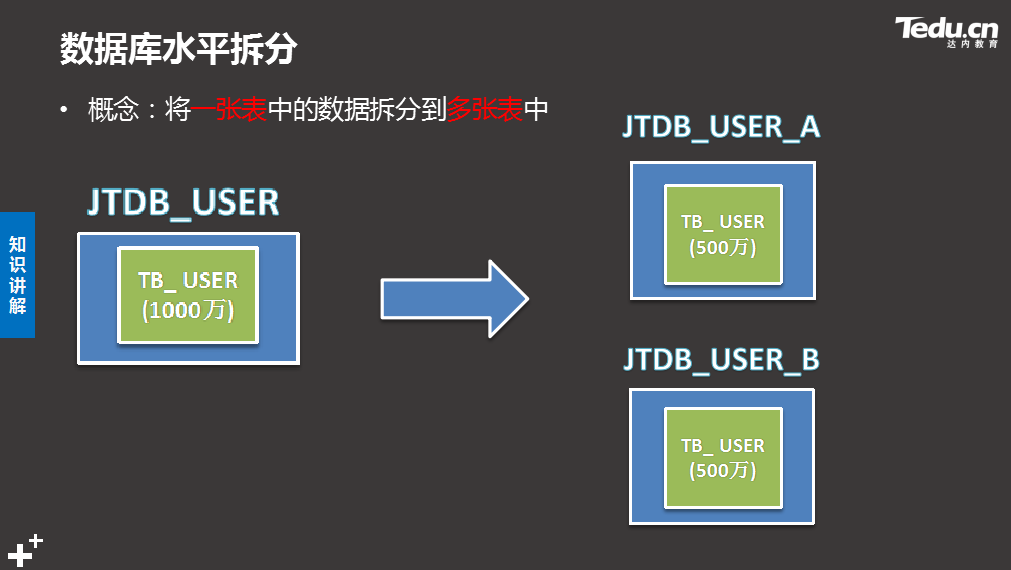
### 垂直拆分

说明:根据数据库中业务将不同业务的表保存到多个数据库中.



### 水平拆分

说明:将一张表中大量的数据拆分保存到多张数据表中.

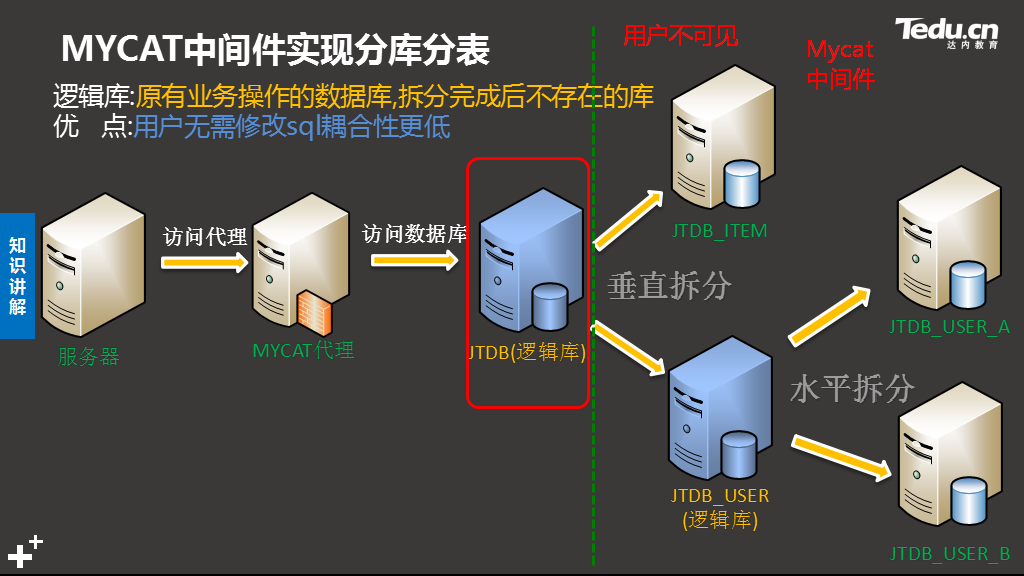


### Mycat调用原理图

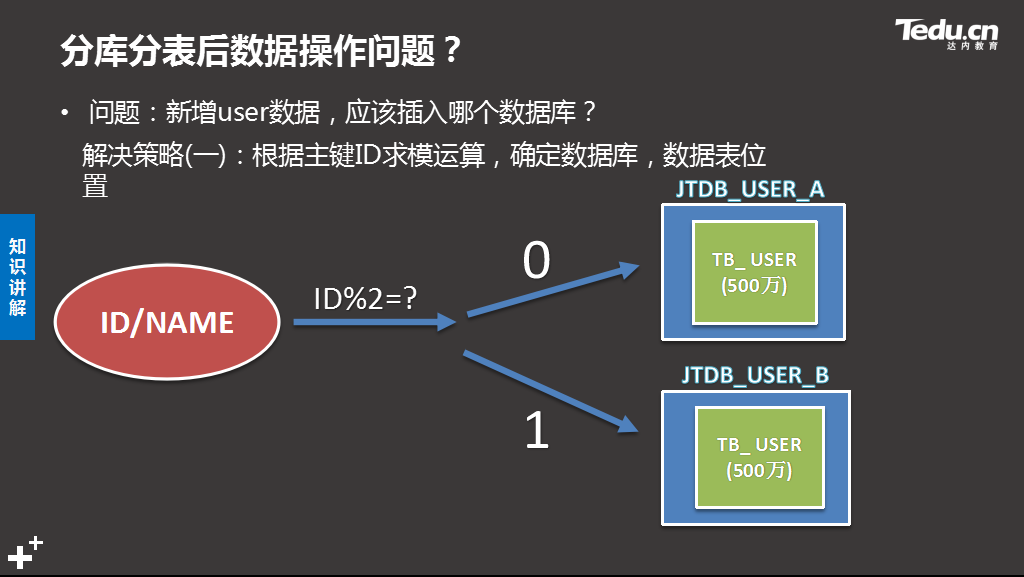
中间件:中间件其实就是简化开发使用的工具.数据库中用户操作的sql在不修改的条件下依然可以操作分库分表以后的数据,这需要中间件的支持.

中间件对用户:中间件封装了用户所操作的数据/库/表

中间件对于数据库:利用逻辑库的概念,通过具体配置将逻辑库与真实的数据库进行链接.同时用户不需要了解.



### Mycat中入库策略(一)



### 数据库优化手段

1. ”优化”sql语句.

1.1查询时精确字段.

1.2尽可能根据主键查询.

1.3如果做多表查询时,尽可能提前确定数据,减少笛卡尔积.

1.4禁止使用notin

1. 灵活使用左连接/有链接/内连接 减少sql语句.
2. 建立数据库索引. 视图(虚表)不会提高查询效率.
3. 利用redis提高查询的效率.
4. 利用历史表定期进行数据的转储.减少表中数据的量.
5. 如果用户查询依然很慢,则建议分库分表.(主要将核心业务进行分库分表)否则运维成本变高.