# 数据库高可用实现(二)

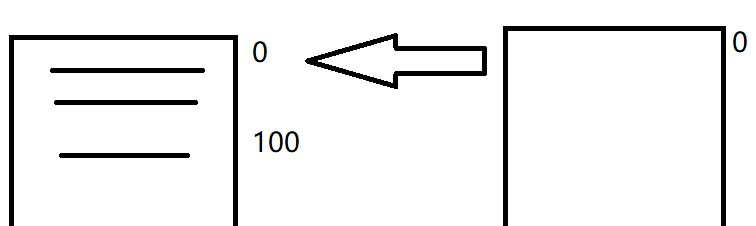
## 数据库挂载注意事项

### 需求

问题:数据库主从挂载时,必须保证数据一样吗??? 不一定

前提条件:同步的数据库和数据表的结构必须一致.

说明:如果数据同步,主从数据库的结构相同,那么同步数据量可以不相同



## 数据库读写分离技术

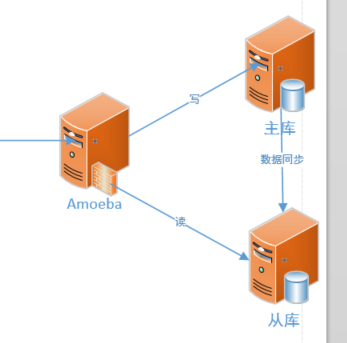
说明:

因为现在的架构设计全部的访问的压力都数据库的服务器端,如果用户持续高并发,那么后台的数据库服务器有宕机的风险.一旦数据库服务器宕机,则整个服务都将陷入瘫痪.

### Amoeba介绍(入门级)

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）。

Amoeba相当于一个SQL请求的[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8" \t "_blank)，目的是为负载均衡、读写分离、高可用性提供机制，而不是完全实现它们。用户需要结合使用MySQL的 Replication等机制来实现副本同步等功能。amoeba对底层数据库连接管理和路由实现也采用了可插拨的机制，第三方可以开发更高级的策略类来替代作者的实现。这个程序总体上比较符合[KISS原则](https://baike.baidu.com/item/KISS%E5%8E%9F%E5%88%99)的思想。

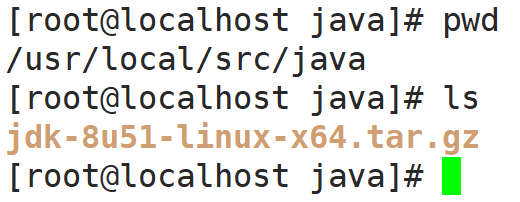


## 安装Amoeba

### 配置JDK

说明:编辑/etc/profile,注意修改其中的JDK路径

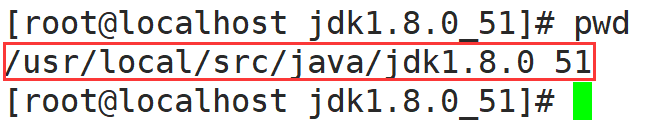
1.上传文件



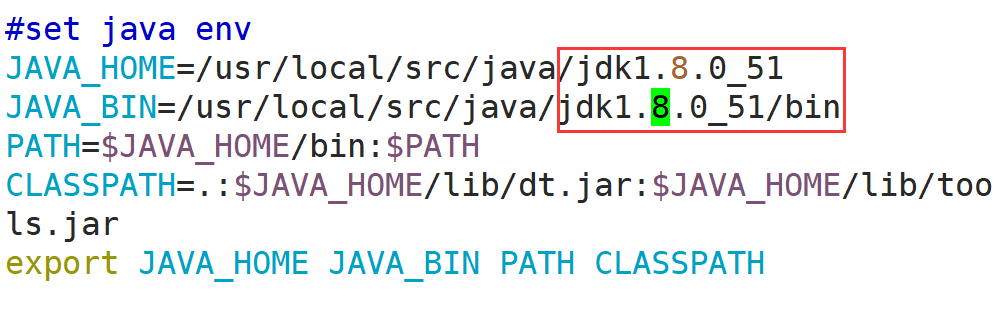
2.解压JDK

tar -xvf jdk-8u51-linux-x64.tar.gz

3.找到JDK根目录



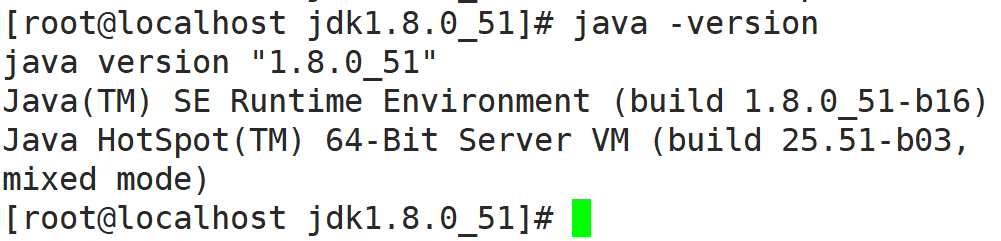
4.编辑profile文件



5.让JDK配置生效

source /etc/profile

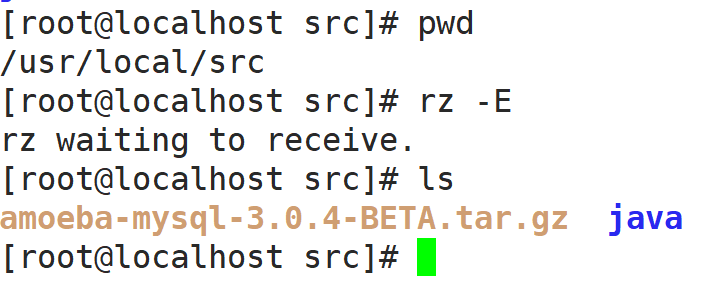
6.检测JDK文件



## 安装Amoeba

### 上传/解压文件

1.上传文件

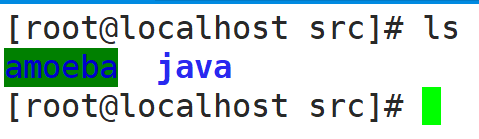


2.解压文件

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

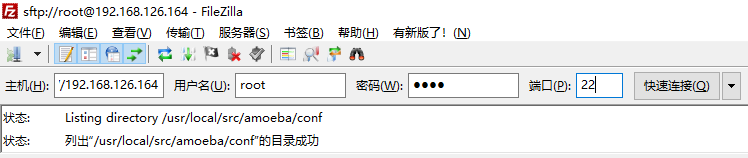
3.修改文件名称

mv amoeba-mysql-3.0.4-BETA amoeba

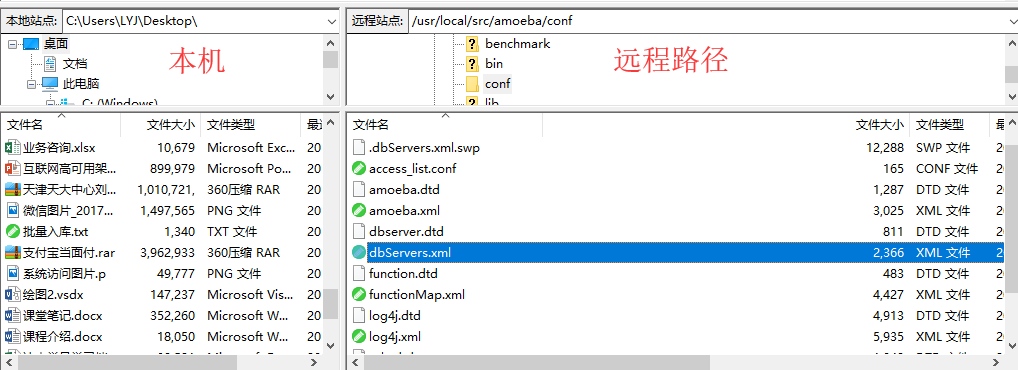


### FZ工具使用

说明:使用IP和用户名密码和端口链接远程Linux系统



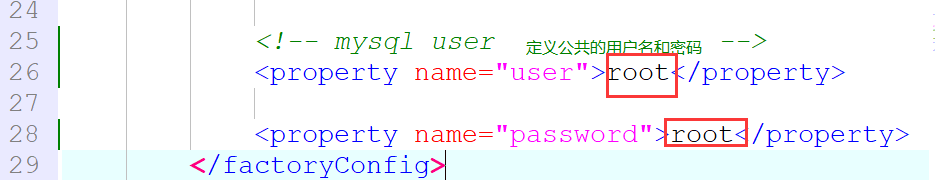
2.将文件下载都本机后编辑



### 编辑db-server.xml

说明:该配置文件是定义代理服务器中连接数据库的配置项.

1. 修改数据库链接的用户名和密码



1. 配置数据库链接

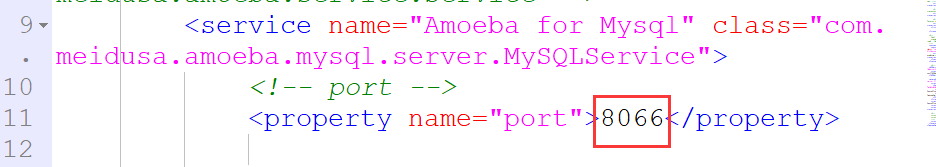


1. 配置读操作连接池

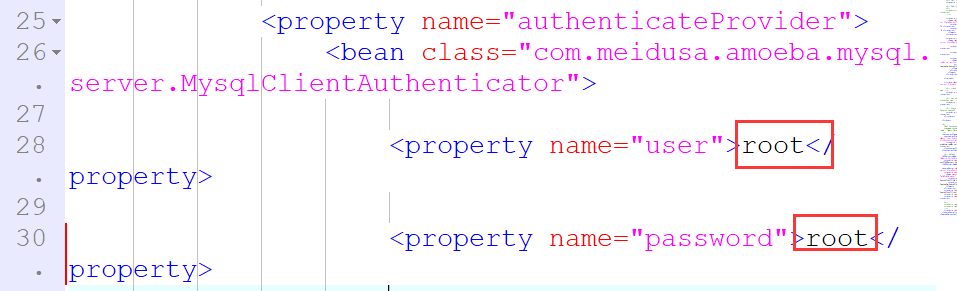


### 配置amoeba.xml

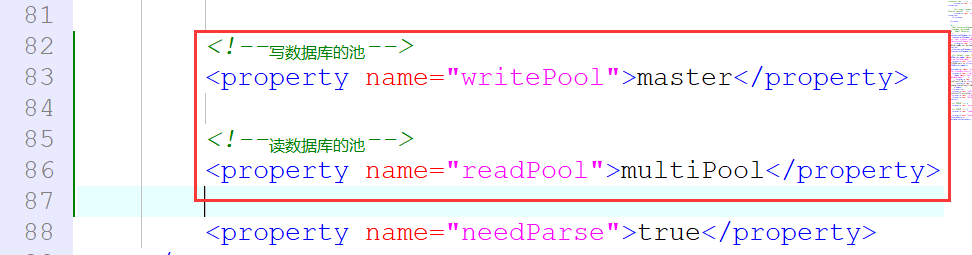
1. 配置端口号



1. 设定链接amoeba的用户名和密码



1. 实现读和写分离



### 修改JVM内存

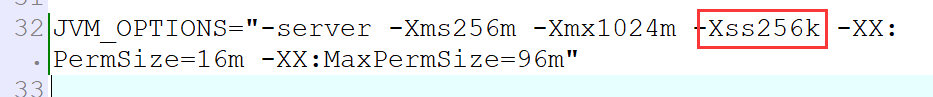
JVM\_OPTIONS="-server -Xms256m -Xmx1024m -Xss196k -XX:PermSize=16m -XX:MaxPermSize=96m"

-Xms256m 当程序启动时,初始化内存大小

-Xmx1024m 内存最大设定

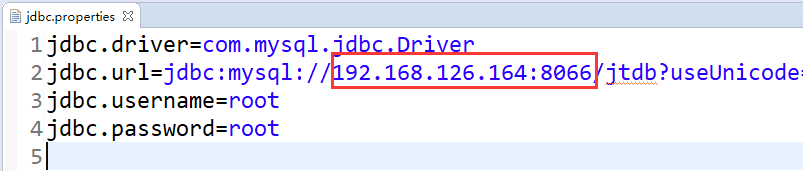
-Xss196k jvm中,线程占用空间的大小

修改线程大小



### Amoeba读写分离测试

1. 修改数据库链接



1. 前提关闭防火墙

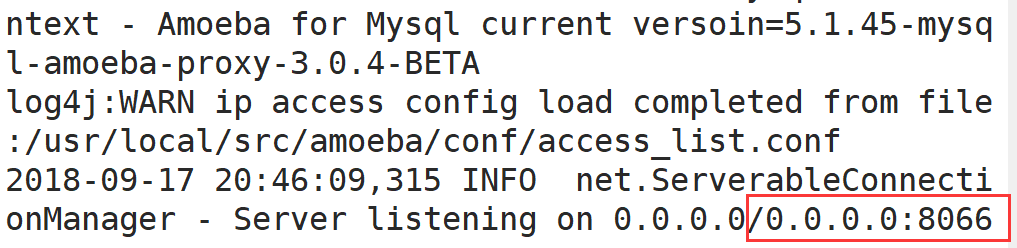
service iptables stop

1. 启动Amoeba

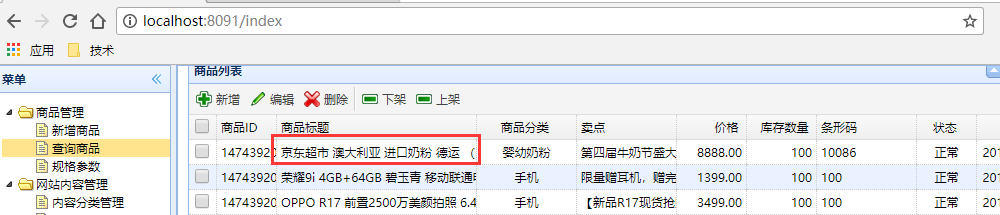
./launcher

./shutdown

Kill -9 PID



1. Amoeba测试



修改从库中的数据,不要修改主键,之后测试,测试时记得重新刷新.

## 实现数据库的高可用

### 高可用回顾

当程序由于异常/断电/其他因素程序不同正常访问,这时应该自动的实现故障迁移.使得用户访问不受影响.

### 双机热备

说明:如果只配置主从结构,这时如果发生主库宕机的现象.当主库重启启动时,因为利用高可用的机制实现故障的迁移.用户将大量的数据发往了从库.导致从库中的数据比主库多,当主库重启时主动的备份将失效.导致主从数据库的数据不一致.

解决方案:

实现双机热备.

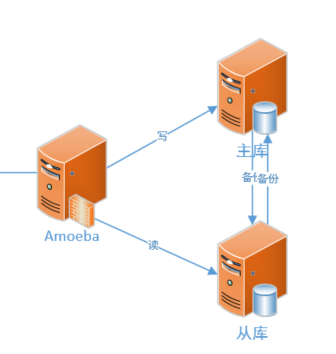
主库 -------A库

从库 -------B库

备份策略:

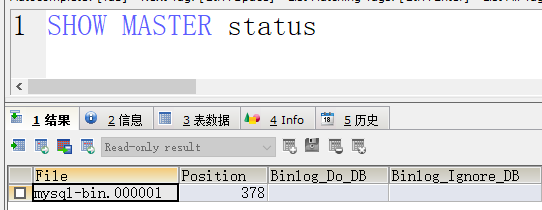
B库同步A库中的数据.

A库同步B库中的数据.



### 双机热备配置

1. 检测B库主机状态



1. 挂载主从

#实现主从挂载

change MASTER to

MASTER\_HOST="192.168.126.163",

MASTER\_PORT=3306,

MASTER\_USER="root",

MASTER\_PASSWORD="root",

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

MASTER\_LOG\_POS=378

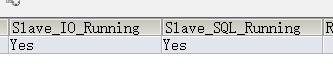
#启动主从服务

start slave

#检测状态是否正常

show SLAVE status

1. 检查状态



## Mycat

### Mycat介绍



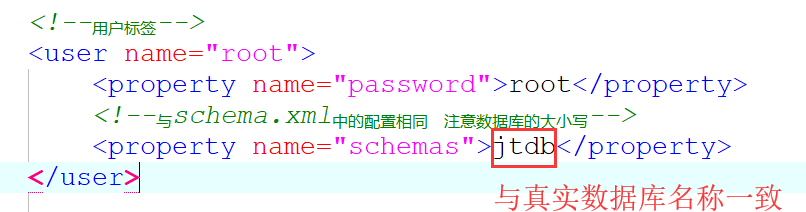
### 安装Mycat

前提:必须安装JDK1.7及以上版本.

1.上传并且解压文件



### 编辑Server.xml配置文件



### 编辑schemas.xml文件

<?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">

<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="192.168.126.162:3306" user="root" password="root">

<!--读数据库-->

<readHost host="hostS1" url="192.168.126.163:3306" 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="192.168.126.163:3306" user="root" password="root">

<readHost host="hostS1" url="192.168.126.162:3306" user="root" password="root" />

</writeHost>

</dataHost>

</mycat:schema>

### Mycat测试

1. 将修改好的配置文件上传



1. mycat命令

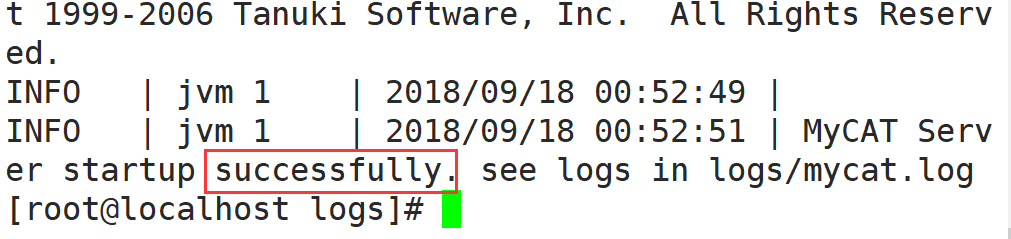
./mycat start

./mycat stop

./mycat restart

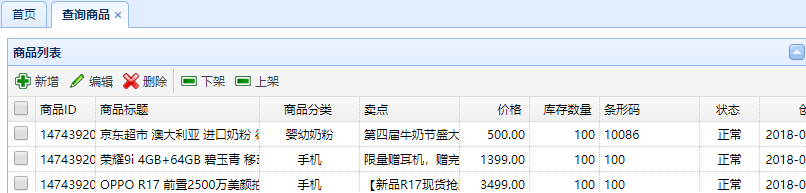
./mycat status **检测mycat状态信息**

3.检测mycat启动



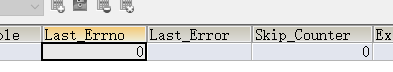
4.高可用测试

将主库关机,之后查询服务,检查是否正确.之后修改数据,检查主库数据是否同步.

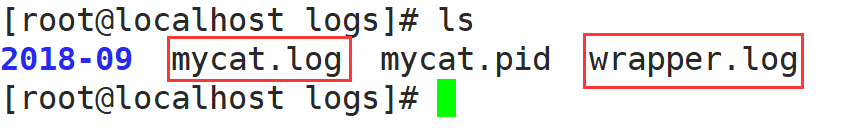


### 关于配置报错问题

1. 关闭amoeba.
2. 数据配置文件 no/yes



1. 数据库配置Connecting yes
2. IP问题
3. 防火墙
4. Mycat日志文件



## 分库分表

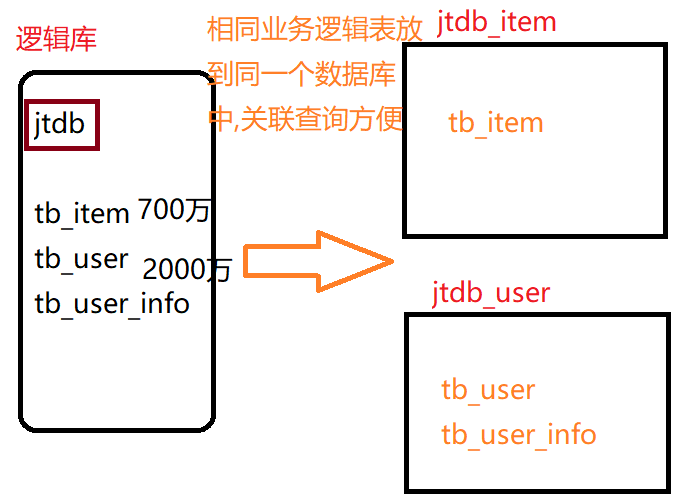
需求:由于公司内部的业务升级,表中的数据量太多,如何优化数据库.

使用一切技术技术手段,都不能满足查询要求----数据库瓶颈问题(量大)

终极策略:分库分表

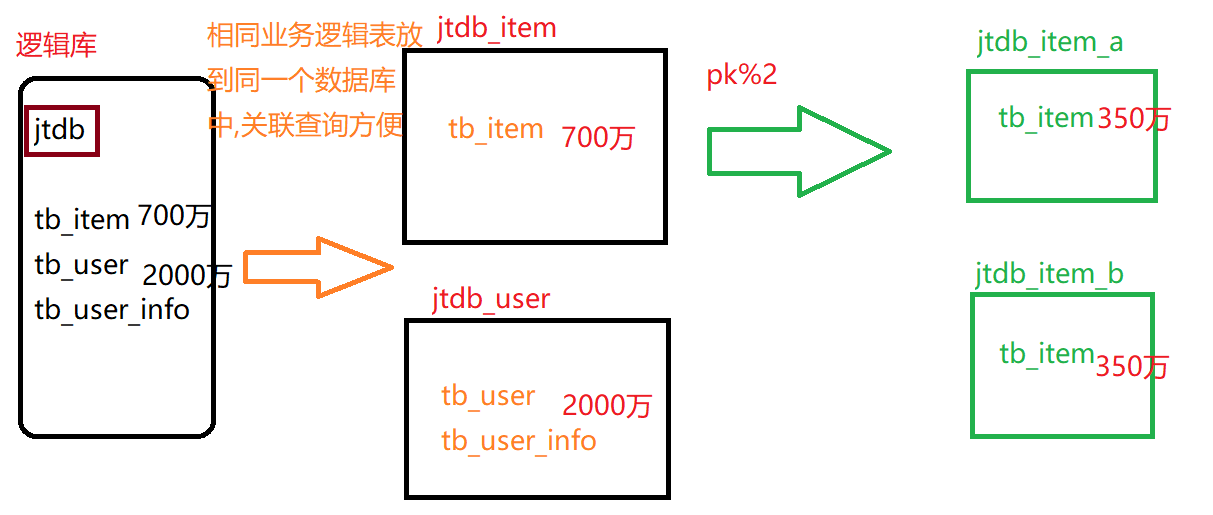
### 数据库垂直拆分

说明:根据业务逻辑进行拆分,叫做垂直拆分



### 数据库水平拆分

说明:将一张数据表拆分为若干张表,共同维护表数据.



### 数据库优化策略

1. 优化sql语句

原则:尽量使用主键查询/减少关联查询(表设计)

1. 创建数据库索引
2. 使用缓存策略(内存),mamerchche/Redis
3. 定期数据转储,将旧数据保存到历史表

Tb\_order tb\_history\_order

1. 分库分表(2-4服务器)