Mycat

# 海量数据的存储问题

如今随着互联网的发展，数据的量级也是撑指数的增长，从GB到TB到PB。对数据的各种操作也是愈加的困难，传统的关系性数据库已经无法满足快速查询与插入数据的需求。这个时候NoSQL的出现暂时解决了这一危机。它通过降低数据的安全性，减少对事务的支持，减少对复杂查询的支持，来获取性能上的提升。

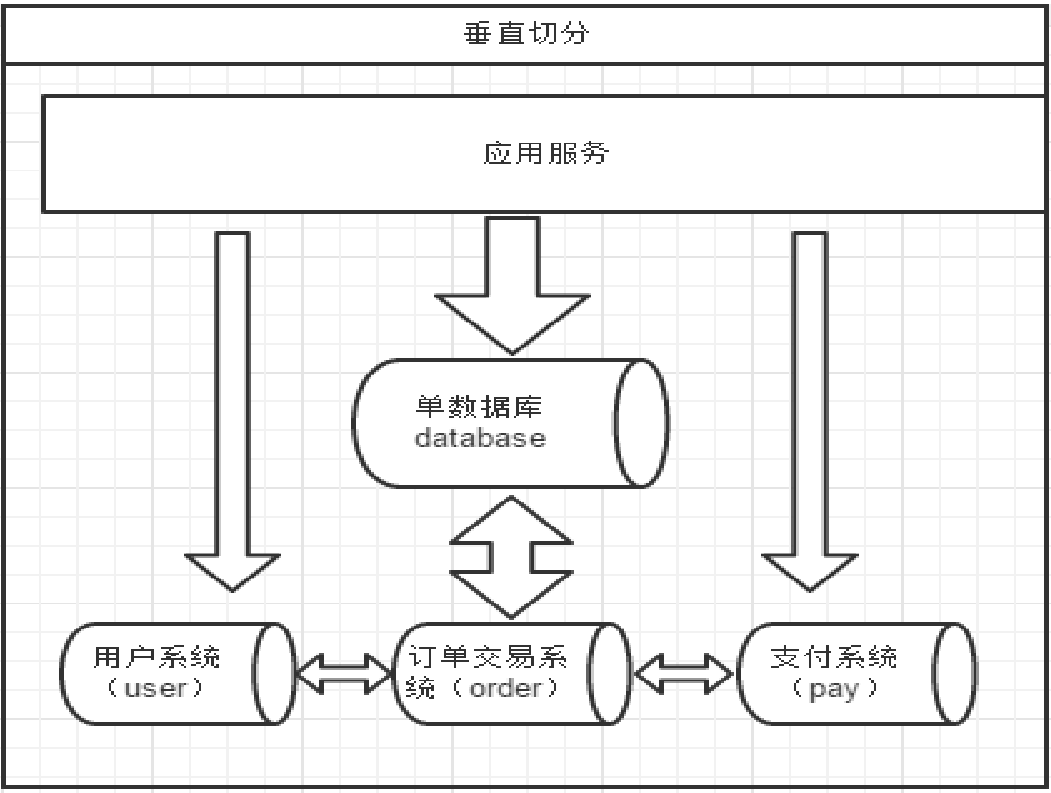
但是，在有些场合NoSQL一些折衷是无法满足使用场景的，就比如有些使用场景是绝对要有事务与安全指标的。这个时候NoSQL肯定是无法满足的，所以还是需要使用关系性数据库。如果使用关系型数据库解决海量存储的问题呢？此时就需要做数据库集群，为了提高查询性能将一个数据库的数据分散到不同的数据库中存储。

## 什么是数据库分片

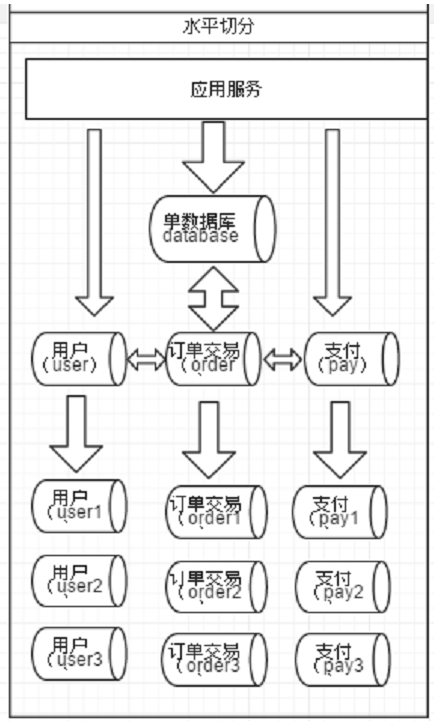
简单来说，就是指通过某种特定的条件，将我们存放在同一个数据库中的数据分散存放到多个数据库（主机）上面，以达到分散单台设备负载的效果。

数据的切分（Sharding）根据其切分规则的类型，可以分为两种切分模式。

1. 一种是按照不同的表（或者Schema）来切分到不同的数据库（主机）之上，这种切可以称之为数据的垂直（纵向）切分



1. 另外一种则是根据表中的数据的逻辑关系，将同一个表中的数据按照某种条件拆分到多台数据库（主机）上面，这种切分称之为数据的水平（横向）切分。



## 如何实现数据库分片

当数据库分片后，数据由一个数据库分散到多个数据库中。此时系统要查询时需要切换不同的数据库进行查询，那么系统如何知道要查询的数据在哪个数据库中？当添加一条记录时要向哪个数据库中插入呢？这些问题处理起来都是非常的麻烦。

这种情况下可以使用一个数据库中间件mycat来解决相关的问题。接下来了解一下什么是mycat。

# Mycat介绍

## 什么是Mycat？

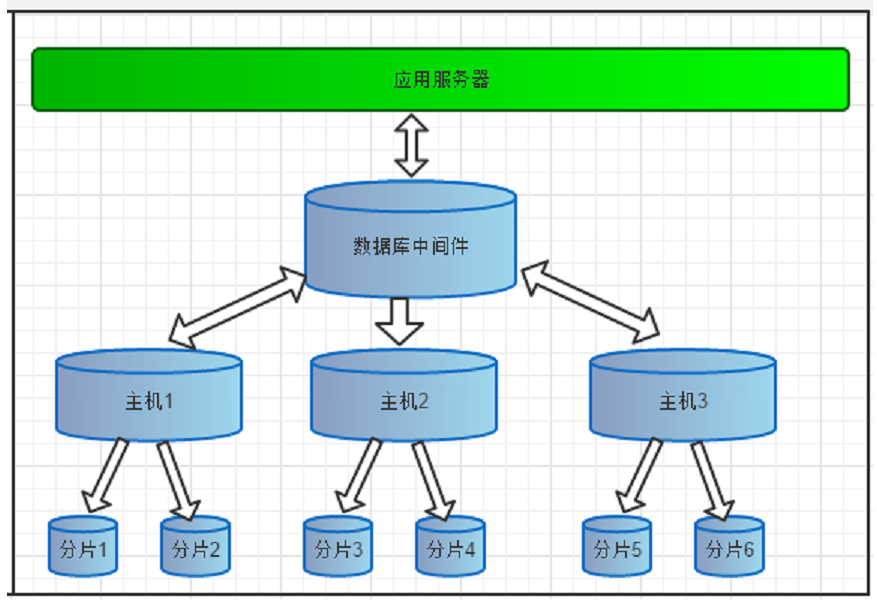
Mycat 背后是阿里曾经开源的知名产品——Cobar。Cobar 的核心功能和优势是 MySQL 数据库分片，此产品曾经广为流传，据说最早的发起者对 Mysql 很精通，后来从阿里跳槽了，阿里随后开源的 Cobar，并维持到 2013 年年初，然后，就没有然后了。

Cobar 的思路和实现路径的确不错。基于 Java 开发的，实现了 MySQL 公开的二进制传输协议，巧妙地将自己伪装成一个 MySQL Server，目前市面上绝大多数 MySQL 客户端工具和应用都能兼容。比自己实现一个新的数据库协议要明智的多，因为生态环境在哪里摆着。

Mycat 是基于 cobar 演变而来，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，同时兼容绝大多数数据库成为通用的数据库中间件。

简单的说，MyCAT就是：

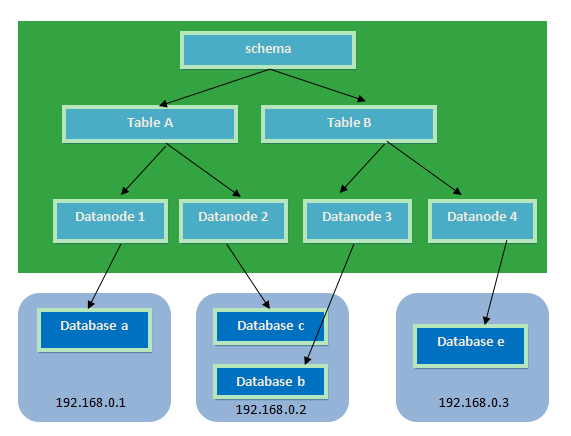
一个新颖的数据库中间件产品支持mysql集群，或者mariadb cluster，**提供高可用性数据分片集群。你可以像使用mysql一样使用mycat。对于开发人员来说根本感觉不到mycat的存在。**



## Mycat支持的数据库



## Mycat的分片策略



## 概念说明

### 逻辑库(schema) ：

前面一节讲了数据库中间件，通常对实际应用来说，并不需要知道中间件的存在，业务开发人员只需要知道数据库的概念，所以数据库中间件可以被看做是一个或多个数据库集群构成的逻辑库。

### 逻辑表（table）：

既然有逻辑库，那么就会有逻辑表，分布式数据库中，对应用来说，读写数据的表就是逻辑表。逻辑表，可以是数据切分后，分布在一个或多个分片库中，也可以不做数据切分，不分片，只有一个表构成。

分片表：是指那些原有的很大数据的表，需要切分到多个数据库的表，这样，每个分片都有一部分数据，所有分片构成了完整的数据。 总而言之就是需要进行分片的表。

非分片表：一个数据库中并不是所有的表都很大，某些表是可以不用进行切分的，非分片是相对分片表来说的，就是那些不需要进行数据切分的表。

### 分片节点(dataNode)

数据切分后，一个大表被分到不同的分片数据库上面，每个表分片所在的数据库就是分片节点（dataNode）。

### 节点主机(dataHost)

数据切分后，每个分片节点（dataNode）不一定都会独占一台机器，同一机器上面可以有多个分片数据库，这样一个或多个分片节点（dataNode）所在的机器就是节点主机（dataHost）,为了规避单节点主机并发数限制，尽量将读写压力高的分片节点（dataNode）均衡的放在不同的节点主机（dataHost）。

### 分片规则(rule)

前面讲了数据切分，一个大表被分成若干个分片表，就需要一定的规则，这样按照某种业务规则把数据分到某个分片的规则就是分片规则，数据切分选择合适的分片规则非常重要，将极大的避免后续数据处理的难度。

# Mycat的下载及安装

## 安装环境

1. jdk：要求jdk必须是1.7及以上版本
2. Mysql：推荐mysql是5.5以上版本
3. Mycat：

Mycat的官方网站：

<http://www.mycat.org.cn/>

下载地址：

<https://github.com/MyCATApache/Mycat-download>

## 安装步骤

Mycat有windows、linux多种版本。本教程为linux安装步骤，windows基本相同。

第一步：下载Mycat-server-xxxx-linux.tar.gz

第二步：将压缩包解压缩。建议将mycat放到/usr/local/mycat目录下。

第三步：进入mycat目录，启动mycat

./mycat start

停止：

./mycat stop

mycat 支持的命令{ console | start | stop | restart | status | dump }

Mycat的默认端口号为：8066

# Mycat的分片

## 需求

把商品表分片存储到三个数据节点上。

## 安装环境分析

两台mysql数据库服务器：

Host1：192.168.25.134

Host2：192.168.25.166

host1环境

操作系统版本 : centos6.4

数据库版本 : mysql-5.6

mycat版本 ：1.4 release

数据库名 : db1、db3

mysql节点2环境

操作系统版本 : centos6.4

数据库版本 : mysql-5.6

mycat版本 ：1.4 release

数据库名 : db2

MyCat安装到节点1上（需要安装jdk）

## 配置schema.xml

### Schema.xml介绍

Schema.xml作为MyCat中重要的配置文件之一，管理着MyCat的逻辑库、表、分片规则、DataNode以及DataSource。弄懂这些配置，是正确使用MyCat的前提。这里就一层层对该文件进行解析。

schema 标签用于定义MyCat实例中的逻辑库

Table 标签定义了MyCat中的逻辑表

dataNode 标签定义了MyCat中的数据节点，也就是我们通常说所的数据分片。

dataHost标签在mycat逻辑库中也是作为最底层的标签存在，直接定义了具体的数据库实例、读写分离配置和心跳语句。

|  |
| --- |
| 注意：若是LINUX版本的MYSQL，则需要设置为Mysql大小写不敏感，否则可能会发生表找不到的问题。  在MySQL的配置文件中/etc/my.cnf [mysqld] 中增加一行  　　lower\_case\_table\_names=1 |

### Schema.xml配置

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://org.opencloudb/">  <schema name="e3mall" checkSQLschema="false" sqlMaxLimit="100">  <!-- auto sharding by id (long) -->  <table name="tb\_item" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />  </schema>  <dataNode name="dn1" dataHost="localhost1" database="db1" />  <dataNode name="dn2" dataHost="localhost2" database="db2" />  <dataNode name="dn3" dataHost="localhost1" database="db3" />  <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="192.168.25.134:3306" user="root"  password="root">  <!-- can have multi read hosts -->  </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>  <!-- can have multi write hosts -->  <writeHost host="hostM1" url="192.168.25.166:3306" user="root"  password="root">  <!-- can have multi read hosts -->  </writeHost>  </dataHost>  </mycat:schema> |

## 配置server.xml

### Server.xml介绍

server.xml几乎保存了所有mycat需要的系统配置信息。最常用的是在此配置用户名、密码及权限。

### Server.xml配置

|  |
| --- |
| <user name="test">  <property name="password">test</property>  <property name="schemas">e3mall</property>  <property name="readOnly">false</property>  </user> |

## 配置rule.xml

rule.xml里面就定义了我们对表进行拆分所涉及到的规则定义。我们可以灵活的对表使用不同的分片算法，或者对表使用相同的算法但具体的参数不同。这个文件里面主要有tableRule和function这两个标签。在具体使用过程中可以按照需求添加tableRule

和function。

此配置文件可以不用修改，使用默认即可。

## 测试分片

### 创建表

配置完毕后，重新启动mycat。使用mysql客户端连接mycat，创建表。

|  |
| --- |
| -- ----------------------------  -- Table structure for tb\_item  -- ----------------------------  DROP TABLE IF EXISTS `tb\_item`;  CREATE TABLE `tb\_item` (  `id` bigint(20) NOT NULL COMMENT '商品id，同时也是商品编号',  `title` varchar(100) NOT NULL COMMENT '商品标题',  `sell\_point` varchar(500) DEFAULT NULL COMMENT '商品卖点',  `price` bigint(20) NOT NULL COMMENT '商品价格，单位为：分',  `num` int(10) NOT NULL COMMENT '库存数量',  `barcode` varchar(30) DEFAULT NULL COMMENT '商品条形码',  `image` varchar(500) DEFAULT NULL COMMENT '商品图片',  `cid` bigint(10) NOT NULL COMMENT '所属类目，叶子类目',  `status` tinyint(4) NOT NULL DEFAULT '1' COMMENT '商品状态，1-正常，2-下架，3-删除',  `created` datetime NOT NULL COMMENT '创建时间',  `updated` datetime NOT NULL COMMENT '更新时间',  PRIMARY KEY (`id`),  KEY `cid` (`cid`),  KEY `status` (`status`),  KEY `updated` (`updated`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品表'; |

### 插入数据

将此文件中的数据插入到数据库：



### 分片测试

由于配置的分片规则为“auto-sharding-long”，所以mycat会根据此规则自动分片。

每个datanode中保存一定数量的数据。根据id进行分片

经测试id范围为：

Datanode1：1~5000000

Datanode2：5000000~10000000

Datanode3：10000001~15000000

当15000000以上的id插入时报错：

[Err] 1064 - can't find any valid datanode :TB\_ITEM -> ID -> 15000001

此时需要添加节点了。

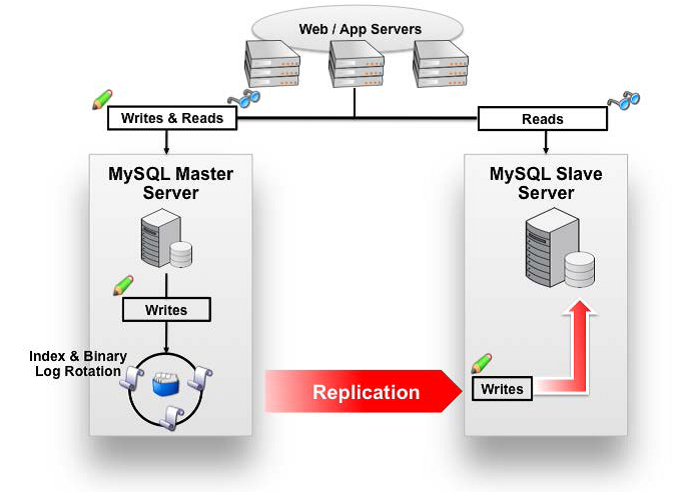
# Mycat读写分离

数据库读写分离对于大型系统或者访问量很高的互联网应用来说，是必不可少的一个重要功能。对于MySQL来说，标准的读写分离是主从模式，一个写节点Master后面跟着多个读节点，读节点的数量取决于系统的压力，通常是1-3个读节点的配置



Mycat读写分离和自动切换机制，需要mysql的主从复制机制配合。

## Mysql的主从复制



主从配置需要注意的地方

1、主DB server和从DB server数据库的版本一致

2、主DB server和从DB server数据库数据名称一致

3、主DB server开启二进制日志,主DB server和从DB server的server\_id都必须唯一

## Mysql主服务器配置

第一步：修改my.cnf文件：

在[mysqld]段下添加：

|  |
| --- |
| binlog-do-db=db1  binlog-ignore-db=mysql  #启用二进制日志  log-bin=mysql-bin  #服务器唯一ID，一般取IP最后一段  server-id=134 |

第二步：重启mysql服务

service mysqld restart

第三步：建立帐户并授权slave

mysql>GRANT FILE ON \*.\* TO 'backup'@'%' IDENTIFIED BY '123456';

mysql>GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* to 'backup'@'%' identified by '123456';

#一般不用root帐号，“%”表示所有客户端都可能连，只要帐号，密码正确，此处可用具体客户端IP代替，如192.168.145.226，加强安全。

刷新权限

mysql> FLUSH PRIVILEGES;

查看mysql现在有哪些用户

mysql>select user,host from mysql.user;

第四步：查询master的状态

mysql> show master status;

+------------------+----------+--------------+------------------+-------------------+

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB | Executed\_Gtid\_Set |

+------------------+----------+--------------+------------------+-------------------+

| mysql-bin.000001 | 120 | db1 | mysql | |

+------------------+----------+--------------+------------------+-------------------+

1 row in set

## Mysql从服务器配置

第一步：修改my.cnf文件

[mysqld]

server-id=166

第二步：配置从服务器

mysql>change master to master\_host='192.168.25.134',master\_port=3306,master\_user='backup',master\_password='123456',master\_log\_file='mysql-bin.000001',master\_log\_pos=120

注意语句中间不要断开，master\_port为mysql服务器端口号(无引号)，master\_user为执行同步操作的数据库账户，“120”无单引号(此处的120就是show master status 中看到的position的值，这里的mysql-bin.000001就是file对应的值)。

第二步：启动从服务器复制功能

Mysql>start slave;

第三步：检查从服务器复制功能状态：

mysql> show slave status

……………………(省略部分)

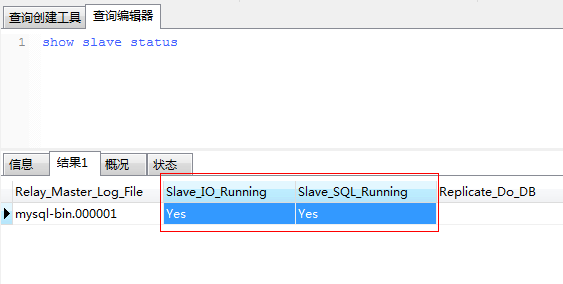
Slave\_IO\_Running: Yes //此状态必须YES

Slave\_SQL\_Running: Yes //此状态必须YES

……………………(省略部分)

注：Slave\_IO及Slave\_SQL进程必须正常运行，即YES状态，否则都是错误的状态(如：其中一个NO均属错误)。

|  |
| --- |
| 错误处理：  如果出现此错误：  Fatal error: The slave I/O thread stops because master and slave have equal MySQL server UUIDs; these UUIDs must be different for replication to work.  因为是mysql是克隆的系统所以mysql的uuid是一样的，所以需要修改。 |
| 解决方法：  删除/var/lib/mysql/auto.cnf文件，重新启动服务。 |



以上操作过程，从服务器配置完成。

## Mycat配置

Mycat 1.4 支持MySQL主从复制状态绑定的读写分离机制，让读更加安全可靠，配置如下：

|  |
| --- |
| <dataNode name=*"dn1"* dataHost=*"localhost1"* database=*"db1"* />  <dataNode name=*"dn2"* dataHost=*"localhost1"* database=*"db2"* />  <dataNode name=*"dn3"* dataHost=*"localhost1"* database=*"db3"* />  <dataHost name=*"localhost1"* maxCon=*"1000"* minCon=*"10"* **balance=*"1"***  **writeType=*"0"*** dbType=*"mysql"* dbDriver=*"native"* **switchType=*"2"* slaveThreshold=*"100"***>  <heartbeat>**show slave status**</heartbeat>  <writeHost host=*"hostM"* url=*"192.168.25.134:3306"* user=*"root"*  password=*"root"*>  <readHost host=*"hostS"* url=*"192.168.25.166:3306"* user=*"root"*  password=*"root"* />  </writeHost>  </dataHost> |

1. **设置 balance="1"与writeType="0"**

Balance参数设置：

1. balance=“0”, 所有读操作都发送到当前可用的writeHost上。

2. balance=“1”，所有读操作都随机的发送到readHost。

3. balance=“2”，所有读操作都随机的在writeHost、readhost上分发

WriteType参数设置：

1. writeType=“0”, 所有写操作都发送到可用的writeHost上。

2. writeType=“1”，所有写操作都随机的发送到readHost。

3. writeType=“2”，所有写操作都随机的在writeHost、readhost分上发。

“readHost是从属于writeHost的，即意味着它从那个writeHost获取同步数据，因此，当它所属的writeHost宕机了，则它也不会再参与到读写分离中来，即“不工作了”，这是因为此时，它的数据已经“不可靠”了。基于这个考虑，目前mycat 1.3和1.4版本中，若想支持MySQL一主一从的标准配置，并且在主节点宕机的情况下，从节点还能读取数据，则需要在Mycat里配置为两个writeHost并设置banlance=1。”

1. **设置 switchType="2" 与slaveThreshold="100"**

**switchType 目前有三种选择：**

**-1：表示不自动切换**

**1 ：默认值，自动切换**

**2 ：基于MySQL主从同步的状态决定是否切换**

“Mycat心跳检查语句配置为 show slave status ，dataHost 上定义两个新属性： switchType="2" 与slaveThreshold="100"，此时意味着开启MySQL主从复制状态绑定的读写分离与切换机制。Mycat心跳机制通过检测 show slave status 中的 "Seconds\_Behind\_Master", "Slave\_IO\_Running", "Slave\_SQL\_Running" 三个字段来确定当前主从同步的状态以及Seconds\_Behind\_Master主从复制时延。“

# 附：Centos6.5下安装mysql

第一步：查看mysql是否安装。

rpm -qa|grep mysql

第二步：如果mysql的版本不是想要的版本。需要把mysql卸载。

yum remove mysql mysql-server mysql-libs mysql-common

rm -rf /var/lib/mysql

rm /etc/my.cnf

第三步：安装mysql。需要使用yum命令安装。在安装mysql之前需要安装mysql的下载源。需要从oracle的官方网站下载。

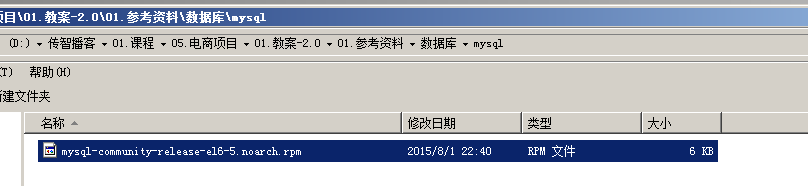
1. 下载mysql的源包。

我们是centos6.4对应的rpm包为：mysql-community-release-el6-5.noarch.rpm

1. 安装mysql下载源：

yum localinstall mysql-community-release-el6-5.noarch.rpm

()此附件可保存



1. 在线安装mysql：

yum install mysql-community-server

第四步：启动mysql

service mysqld start

第五步：需要给root用户设置密码。

/usr/bin/mysqladmin -u root password 'new-password'　　// 为root账号设置密码

第六步：远程连接授权。

GRANT ALL PRIVILEGES ON \*.\* TO 'myuser'@'%' IDENTIFIED BY 'mypassword' WITH GRANT OPTION;

注意：'myuser'、'mypassword' 需要替换成实际的用户名和密码。