

Mycat的安装与数据库分片

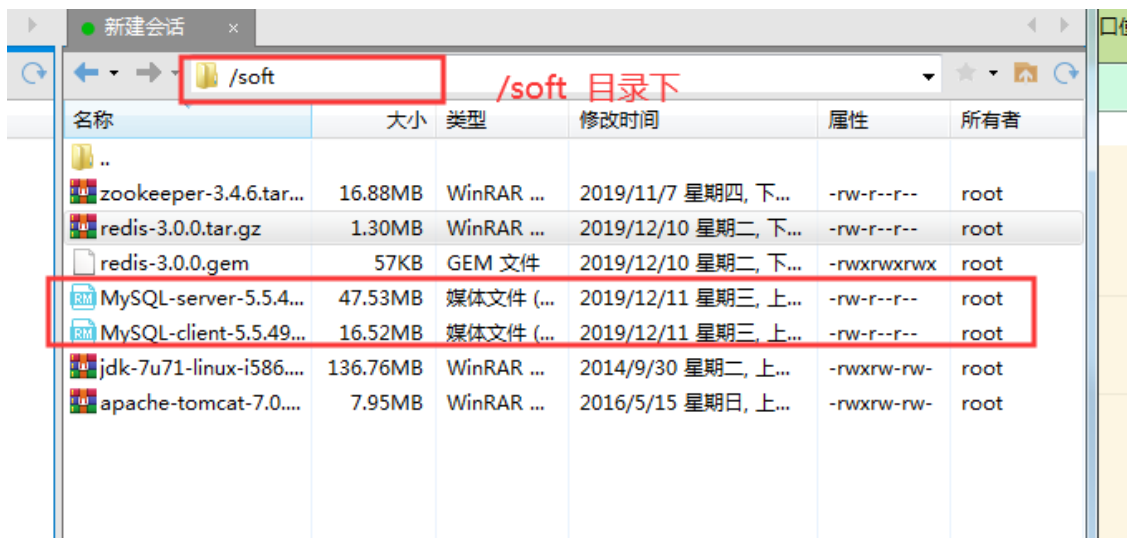
1. MySQL安装与启动

JDK: 要求jdk必须是1.7及以上版本

MySQL: 推荐mysql是5.5以上版本

1.1 MySQL安装

1. 将MySQL的服务端和客户端安装包（RPM）上传到服务器



2. 查询之前是否安装过MySQL

```
rpm -qa|grep -i mysql
```

注意 如果有了上面提示的版本的高版本则不用进行mysql的卸载安装

3. 卸载旧版本

```
rpm -e --nodeps 软件名称
```

4. 安装mysql的服务器

```
rpm -ivh MySQL-server-5.5.49-1.linux2.6.i386.rpm
```

5. 安装客户端

```
rpm -ivh MySQL-client-5.5.49-1.linux2.6.i386.rpm
```

6. 启动MySQL服务

```
service mysql start
```

7. 登陆MySQL

```
mysql -u root
```

8. 设置远程登陆权限

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123' WITH GRANT OPTION;  
flush privileges;           //注意一定要带上‘；’要不然没作用
```

第一行命令 的意思是 给登陆密码为 123 用户名为root 的用户所有的 数据库 所有的表的所有的权限

第二行命令 是刷新权限使其立即生效

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123' WITH GRANT OPTION;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> flush privileges;  
Query OK, 0 rows affected (0.00 sec)
```

9. 本地SQLyog连接远程MySQL进行测试

2.mycat安装与启动

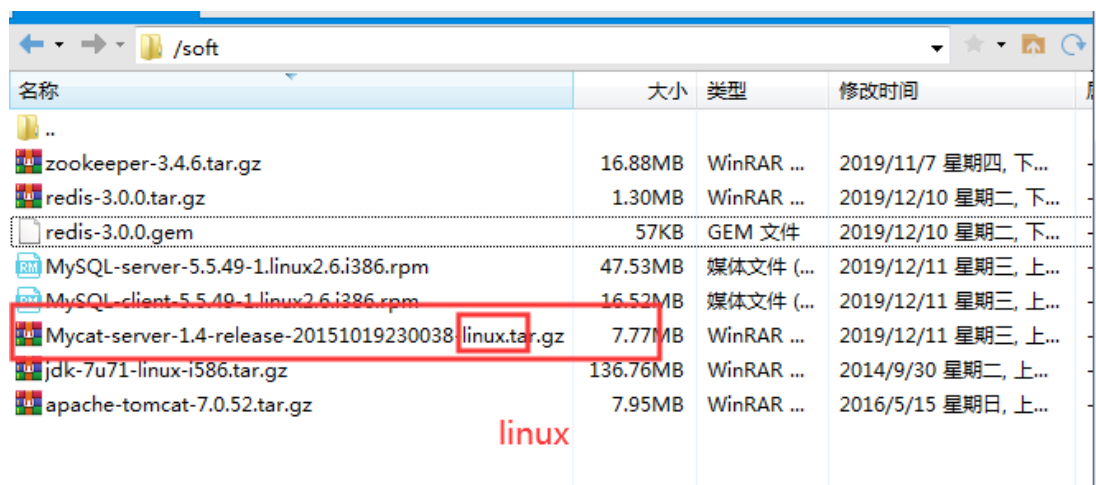
1. 官网:

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

2. 下载地址

<https://github.com/MyCAT/apache/Mycat-download>

3. 将Mycat-server-1.4-release-20151019230038-linux.tar.gz上传至服务器



名称	大小	类型	修改时间
..			
zookeeper-3.4.6.tar.gz	16.88MB	WinRAR ...	2019/11/7 星期四, 下...
redis-3.0.0.tar.gz	1.30MB	WinRAR ...	2019/12/10 星期二, 下...
redis-3.0.0.gem	57KB	GEM 文件	2019/12/10 星期二, 下...
MySQL-server-5.5.49-1.linux2.6.i386.rpm	47.53MB	媒体文件 (...)	2019/12/11 星期三, 上...
MySQL-client-5.5.49-1.linux2.6.i386.rpm	16.52MB	媒体文件 (...)	2019/12/11 星期三, 上...
Mycat-server-1.4-release-20151019230038-linux.tar.gz	7.77MB	WinRAR ...	2019/12/11 星期三, 上...
jdk-7u71-linux-i586.tar.gz	136.76MB	WinRAR ...	2014/9/30 星期二, 上...
apache-tomcat-7.0.52.tar.gz	7.95MB	WinRAR ...	2016/5/15 星期日, 上...

linux

4. 解压然后将解压后的文件移动到 /usr/local/

```
tar xzf Mycat-server-1.4-release-20151019230038-linux.tar.gz
```

```
mv 解压后的文件名 /usr/local/
```

5. 测试mycat,进入mycat 的bin 目录

```
./mycat start           //启动
```

```
./mycat stop            //停止
```

3.MyCat分片-海量数据存储解决方案

3.1 什么是分片？

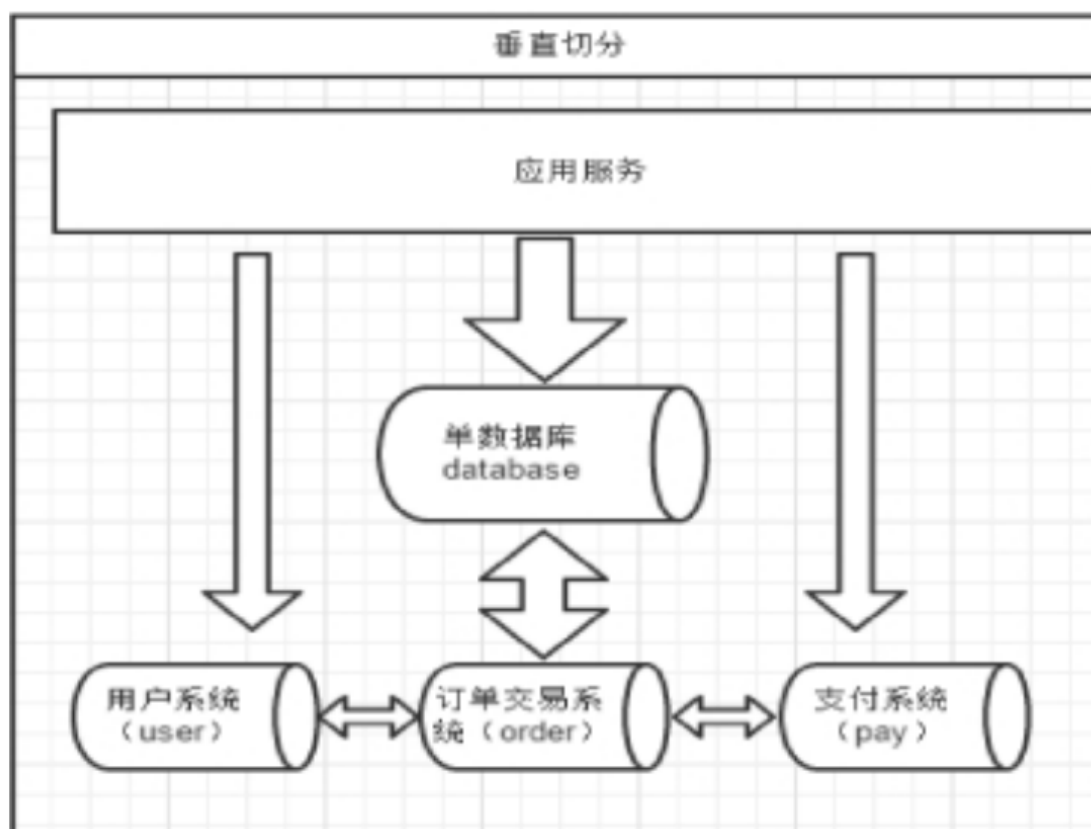
简单来说，就是指通过某种特定的条件，将我们存放在同一个数据库中的数据分散存放到多个数据库（主机）上

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。

3.1.1 垂直分片

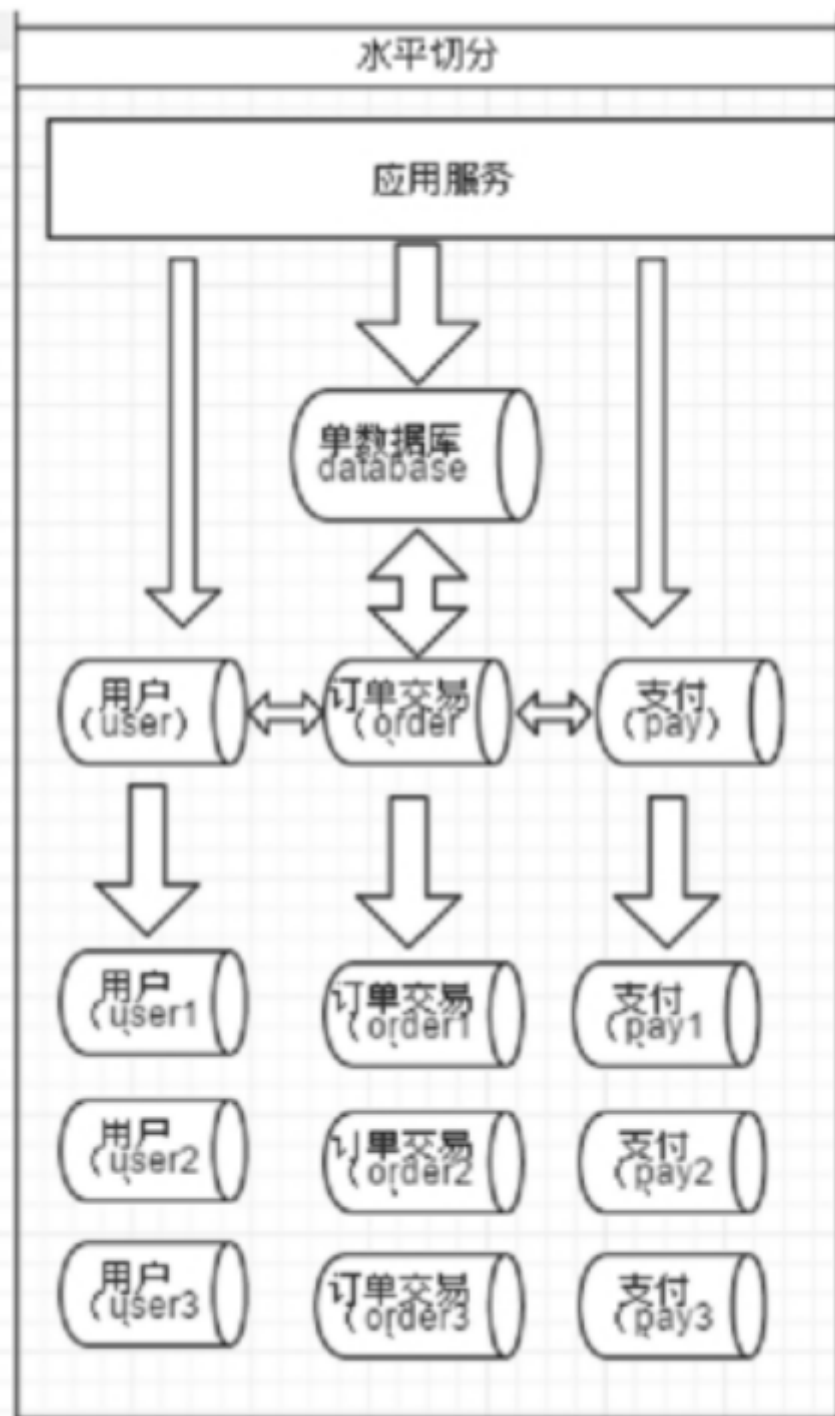
一种是按照不同的表（或者Schema）来切分到不同的数据库（主机）之上，这种切分可以称之为数据的垂直（纵向）切分



3.1.2 水平分片

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。



3.2.3 分片相关的概念

逻辑库

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

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

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

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。

3.2 分片的配置

3.2.1 配置schema.xml

schema.xml作为MyCat中重要的配置文件之一，管理着MyCat的逻辑库、逻辑表以及对应的分片规则、**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 **rule**用于指定分片规则，**auto-sharding-long**的分片规则是按ID值的范围

进行分片 1-5000000 为第1片 5000001-10000000 为第2片....

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。

在mysql服务器上创建3个数据库，分别是db1 db2 db3

修改schema.xml如下：

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://org.opencloudb/">
  <schema name="PINYOUGOUDB" checkSQLSchema="false" sqlMaxLimit="100">
    <table name="tb_test" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />
  </schema>
  <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="0" writeType="0"
dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <writeHost host="hostM1" url="192.168.25.142:3306" user="root" password="123">
  </writeHost>
  </dataHost>
</mycat:schema>
```

3.2.1 配置server.xml

```
<property name="charset">utf8</property>
```

修改user的设置，我们这里为 PINYOUGOUDB设置了两个用户：

```
<user name="test">
  <property name="password">test</property>
  <property name="schemas">PINYOUGOUDB</property>
</user>
<user name="root">
  <property name="password">123</property>
  <property name="schemas">PINYOUGOUDB</property>
</user>
```

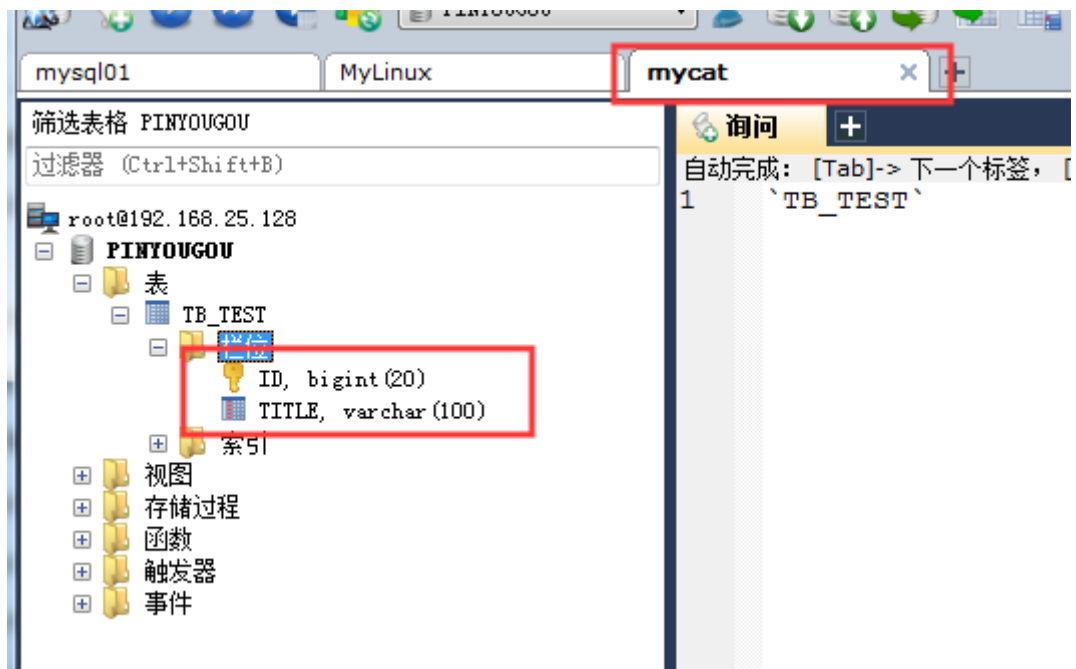
3.3 测试MyCat分片

重要：在测试前除了进行前面的配置及启动mysql与mycat外，还要在mysql中创建上面提到的三个数据库db1,db2及db3，不然报：找不到datasource错误。

进入mycat，执行一个创建表语句：

```
CREATE TABLE tb_test (
  id BIGINT(20) NOT NULL,
  title VARCHAR(100) NOT NULL ,
  PRIMARY KEY (id)
) ENGINE=INNODB DEFAULT CHARSET=utf8
```

创建后你会发现，MyCat会自动将你的表转换为大写，这一点与Oracle有些类似



我们再查看MySQL的3个库，发现表都自动创建好啦。好神奇。

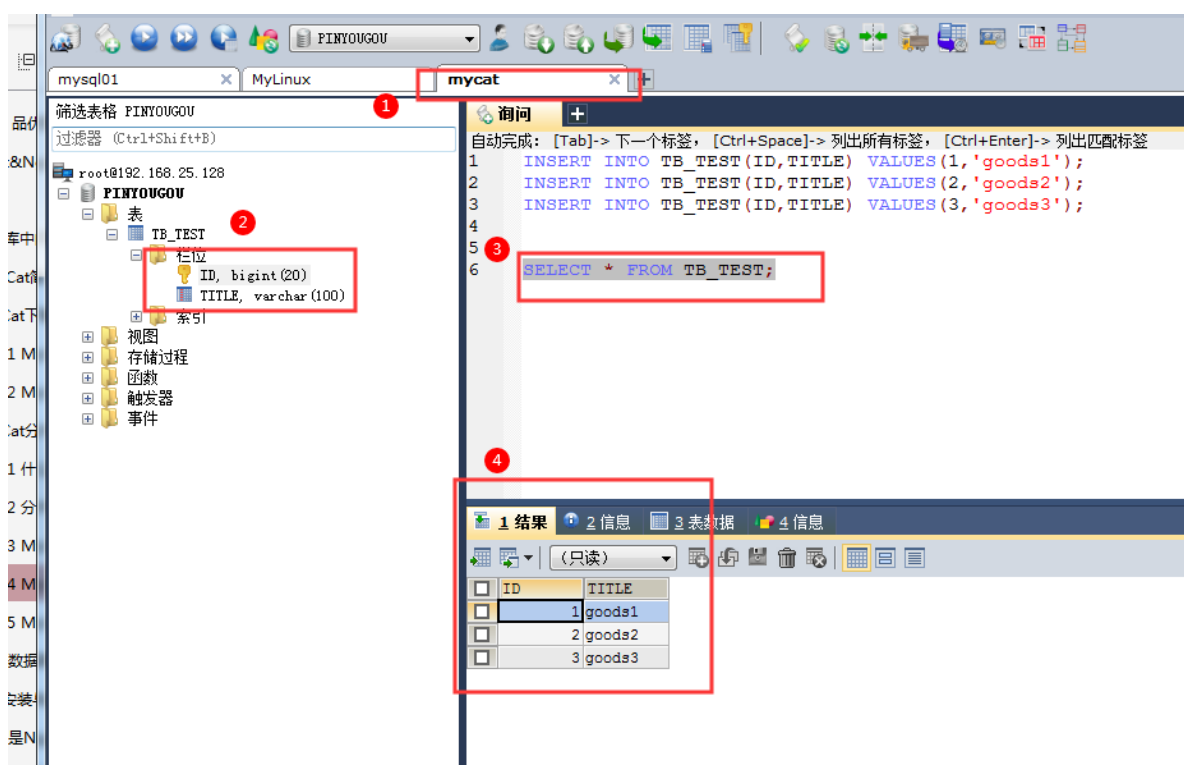
接下来是插入表数据，注意，在写INSERT语句时一定要写把字段列表写出来，否则会出现下列错误提示：

错误代码： 1064

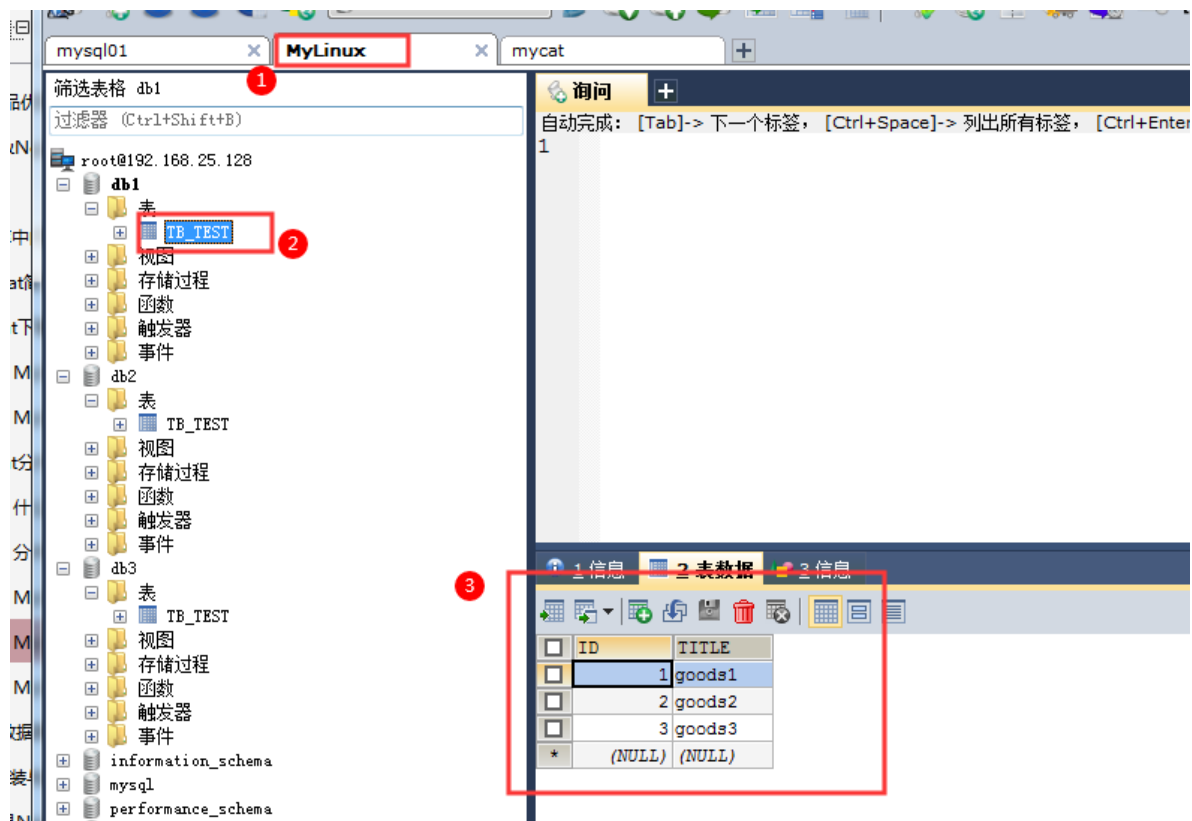
partition table, insert must provide ColumnList

我们试着插入一些数据：

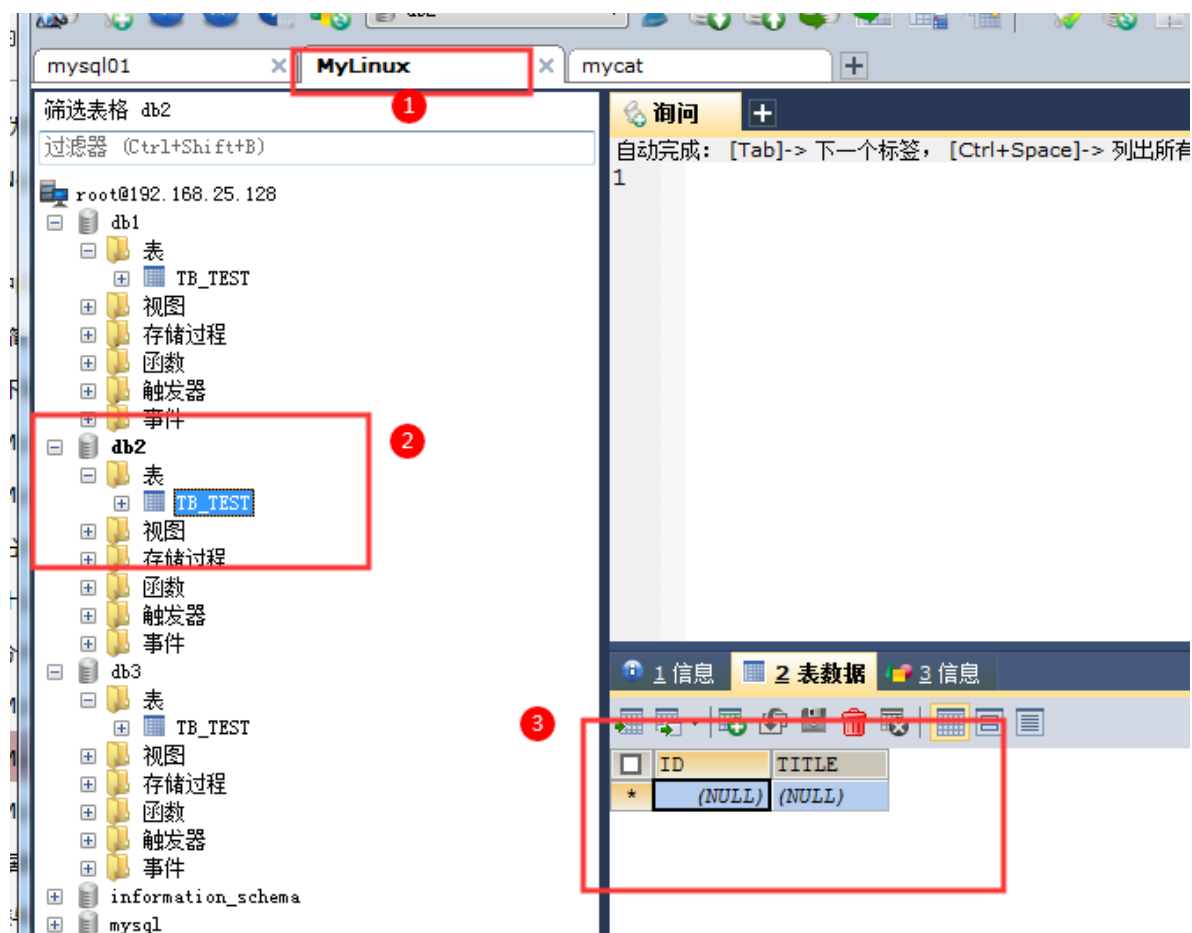
```
INSERT INTO TB_TEST(ID,TITLE) VALUES(1,'goods1');
INSERT INTO TB_TEST(ID,TITLE) VALUES(2,'goods2');
INSERT INTO TB_TEST(ID,TITLE) VALUES(3,'goods3');
```

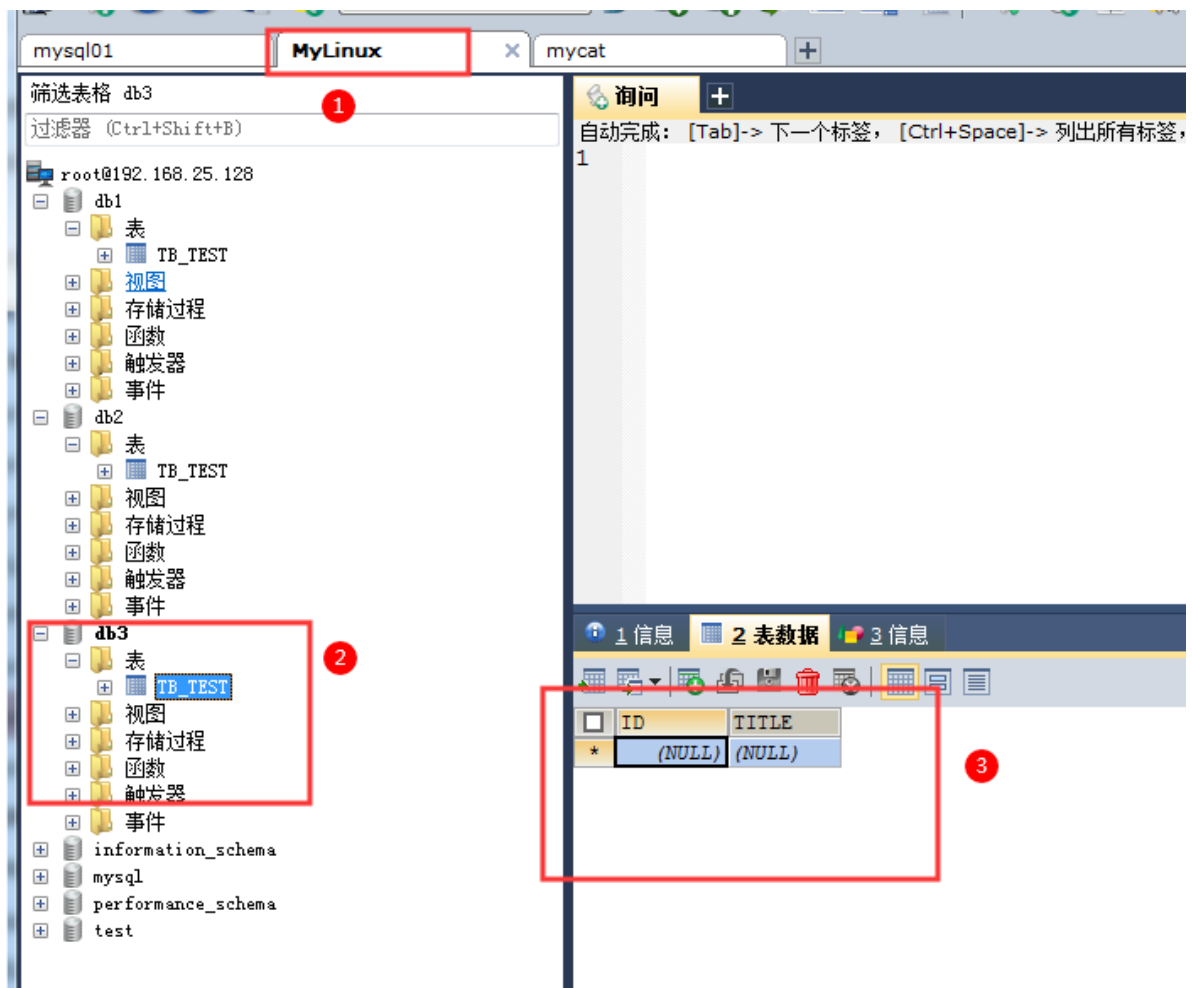


然后我们mycat管理的数据库分片中： 第一个分片也插入了上面的字段值



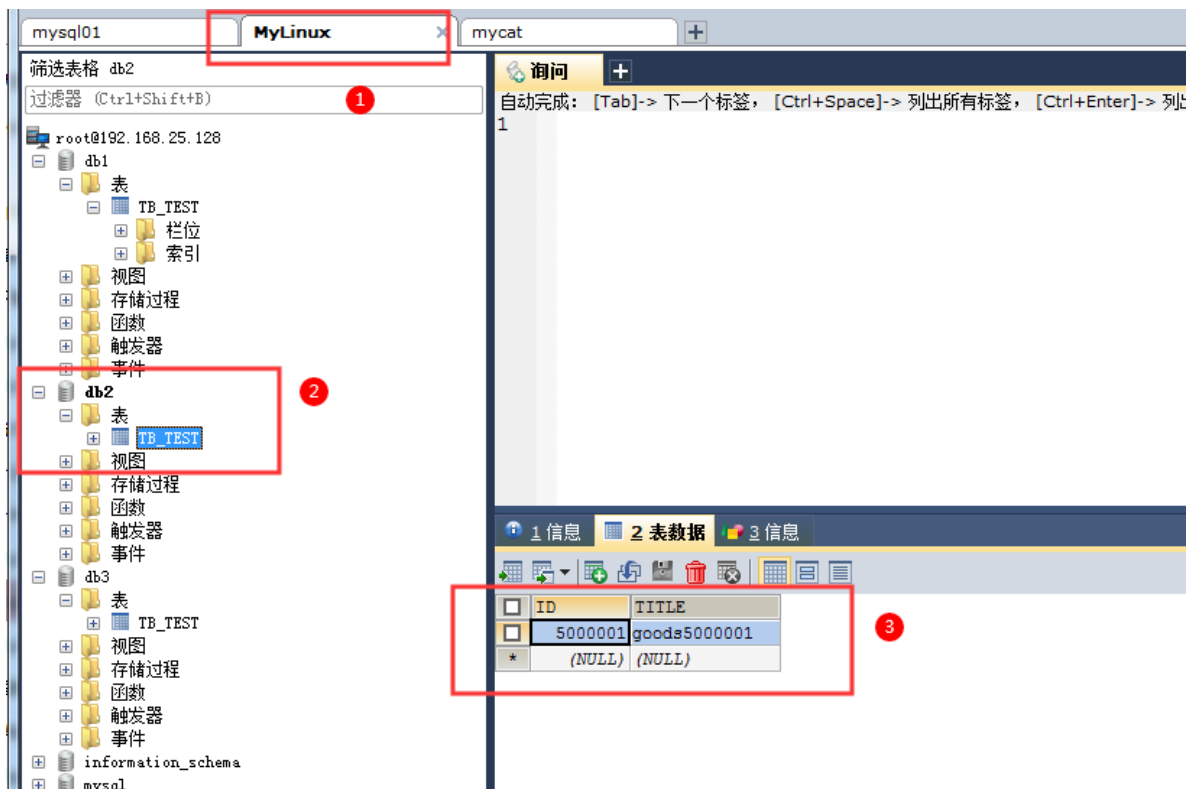
但是其他两个分片却没有插入数据：





什么时候插入第二个节点或者第三个节点呢？

```
inset into TB_TEST(ID,TITLE) VALUES(5000001,'goods5000001');
```



因为我们采用的分片规则是每节点存储500万条数据，所以当ID大于5000000则会存储到第二个节点

上。

目前只设置了两个节点，如果数据大于1000万条，会怎么样呢？执行下列语句测试一下

```
INSERT INTO TB_TEST(ID,TITLE) VALUES(10000001,'goods10000001');
```

3.4 MyCat的分片规则

`rule.xml` 用于定义分片规则，我们这里讲解两种最常见的分片规则

3.4.1 按主键范围分片rang-long

在配置文件中我们找到`rules.xml`

```
<tableRule name="auto-sharding-long">
  <rule>
    <columns>id</columns>
    <algorithm>rang-long</algorithm>
  </rule>
</tableRule>
```

1. `tableRule` 是定义具体某个表或某一类表的分片规则名称
2. `columns` 用于定义分片的列
3. `algorithm` 代表算法名称

我们接着找rang-long的定义

```
<function name="rang-long"
  class="org.opencloudb.route.function.AutoPartitionByLong">
  <property name="mapFile">autopartition-long.txt</property>
</function>
```

1. `Function` 用于定义算法
2. `mapFile` 用于定义算法需要的数据

我们打开`autopartition-long.txt`

```
# range start-end ,data node index
# K=1000,M=10000.
0-500M=0
500M-1000M=1
1000M-1500M=2
```

对于上面的代码的解释是：

```
id 在0-500万 落在分片一
id 在500万-1000万 落在分片二
id 在1000万-1500万 落在分片三
```

3.4.2 一致性哈希murmur

当我们需要将数据平均分在几个分区中，需要使用一致性hash规则(修改的是`rule.xml`文件)我们找到`function`

的name为murmur 的定义，将count属性改为3，因为我要将数据分成3片

```

<function name="murmur"
    class="org.opencldb.route.function.PartitionByMurmurHash">
    <property name="seed">0</property><!-- 默认是0 -->
    <property name="count">2</property><!-- 要分片的数据库节点数量，必须指定，否则没法分片 -->
    <property name="virtualBucketTimes">160</property><!-- 一个实际的数据库节点被映射为这么多虚拟节点，默认是160倍，也就是虚拟节点数是物理节点数的160倍 -->
    <!-- <property name="weightMapFile">weightMapFile</property> 节点的权重，没有指定权重的节点默认是1。以properties文件的格式填写，以从0开始到count-1的整数值也就是节点索引为key，以节点权重值为值。所有权重值必须是正整数，否则以1代替 -->
    <!-- <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>
        用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的murmur hash值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西
    -->
</function>

```

我们再配置文件中可以找到表规则定义（rule.xml）

```

<tableRule name="sharding-by-murmur">
    <rule>
        <columns>id</columns>
        <algorithm>murmur</algorithm>
    </rule>
</tableRule>

```

但是这个规则指定的列是id，如果我们的表主键不是id，而是order_id，那么我们应该重新定义一个tableRule: (rule.xml)

```

<!-- 自定义的分片规则 根据我们表的字段主键进行一致性哈希 均衡分片 -->
<tableRule name="sharding-by-murmur-orderId">
    <rule>
        <columns>order_id</columns>
        <algorithm>murmur</algorithm>
    </rule>
</tableRule>

```

在schema.xml中配置逻辑表时，指定规则为sharding-by-murmur-orderId(schema.xml)

```

<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://org.opencldb/">

    <schema name="PINYOUYOU" checkSQLschema="false" sqlMaxLimit="100">
        <!-- auto sharding by id (long) -->
        <table name="TB_TEST" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />
        <!-- 自定义的分片规则 对表 TB_ORDER 中的字段 进行一致性哈希均衡分片 -->
        <table name="TB_ORDER" dataNode="dn1,dn2,dn3" rule="sharding-by-murmur-orderId" />
    </schema>

    <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="0">

```

```

        writeType="0" dbType="mysql" dbDriver="native" switchType="1"
slaveThreshold="100">
    <heartbeat>select user()</heartbeat>

    <writeHost host="hostM1" url="192.168.25.128:3306" user="root"
        password="123">
    </writeHost>
</dataHost>
</mycat:schema>

```

最后我们导入数据测试一下：

1. 在mycat的PINYOUYOU中创建表tb_order

```

USE PINYOUYOUDB; CREATE TABLE `tb_order` (
    `order_id` BIGINT(20) NOT NULL COMMENT '订单id',
    `payment` DECIMAL(20,2) DEFAULT NULL COMMENT '实付金额。精确到2位小数;单位:元。
    如:200.07, 表示:200元7分',
    `payment_type` VARCHAR(1) COLLATE utf8_bin DEFAULT NULL COMMENT '支付类型, 1、
    在线支付, 2、货到付款',
    `post_fee` VARCHAR(50) COLLATE utf8_bin DEFAULT NULL COMMENT '邮费。精确到2位小
    数;单位:元。如:200.07, 表示:200元7分',
    `status` VARCHAR(1) COLLATE utf8_bin DEFAULT NULL COMMENT '状态: 1、未付款, 2、
    已付款, 3、未发货, 4、已发货, 5、交易成功, 6、交易关闭,7、待评价',
    `create_time` DATETIME DEFAULT NULL COMMENT '订单创建时间',
    `update_time` DATETIME DEFAULT NULL COMMENT '订单更新时间',
    `payment_time` DATETIME DEFAULT NULL COMMENT '付款时间',
    `consign_time` DATETIME DEFAULT NULL COMMENT '发货时间',
    `end_time` DATETIME DEFAULT NULL COMMENT '交易完成时间',
    `close_time` DATETIME DEFAULT NULL COMMENT '交易关闭时间',
    `shipping_name` VARCHAR(20) COLLATE utf8_bin DEFAULT NULL COMMENT '物流名称',
    `shipping_code` VARCHAR(20) COLLATE utf8_bin DEFAULT NULL COMMENT '物流单号',
    `user_id` VARCHAR(50) COLLATE utf8_bin DEFAULT NULL COMMENT '用户id',
    `buyer_message` VARCHAR(100) COLLATE utf8_bin DEFAULT NULL COMMENT '买家留言',
    `buyer_nick` VARCHAR(50) COLLATE utf8_bin DEFAULT NULL COMMENT '买家昵称',
    `buyer_rate` VARCHAR(2) COLLATE utf8_bin DEFAULT NULL COMMENT '买家是否已经评
    价',
    `receiver_area_name` VARCHAR(100) COLLATE utf8_bin DEFAULT NULL COMMENT '收货
    人 地区名称(省, 市, 县)街道',
    `receiver_mobile` VARCHAR(12) COLLATE utf8_bin DEFAULT NULL COMMENT '收货人手
    机',
    `receiver_zip_code` VARCHAR(15) COLLATE utf8_bin DEFAULT NULL COMMENT '收货人
    邮 编',
    `receiver` VARCHAR(50) COLLATE utf8_bin DEFAULT NULL COMMENT '收货人',
    `expire` DATETIME DEFAULT NULL COMMENT '过期时间, 定期清理',
    `invoice_type` VARCHAR(1) COLLATE utf8_bin DEFAULT NULL COMMENT '发票类型(普通
    发 票, 电子发票, 增值税发票)',
    `source_type` VARCHAR(1) COLLATE utf8_bin DEFAULT NULL COMMENT '订单来源:
    1:app 端, 2: pc端, 3: M端, 4: 微信端, 5: 手机qq端',
    `seller_id` VARCHAR(100) COLLATE utf8_bin DEFAULT NULL COMMENT '商家ID',
    PRIMARY KEY (`order_id`), KEY `create_time` (`create_time`),
    KEY `buyer_nick` (`buyer_nick`), KEY `status` (`status`),
    KEY `payment_type` (`payment_type`)
) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

```

2. 在表中添加如下内容：


```

INSERT INTO
`TB_ORDER`(`order_id`,`payment`,`payment_type`,`post_fee`,`status`,`create_time`
`,`update_time`,`payment_time`,`consign_time`,`end_time`,`close_time`,`shipping_n
ame`,`shipping_code`,`user_id`,`buyer_message`,`buyer_nick`,`buyer_rate`,`receiv
er_area_name`,`receiver_mobile`,`receiver_zip_code`,`receiver`,`expire`,`invoice
_type`,`source_type`,`seller_id`) VALUES (1,NULL,'1',NULL,'0','2017-08-24
20:42:25','2017-08-24
20:42:25',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(2,NULL,'1',NULL,'0','2017-
08-24 20:44:03','2017-08-24
20:44:03',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(3,3.00,'1',NULL,'0','2017-
08-24 20:46:10','2017-08-24
20:46:10',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(4,266.64,'1',NULL,'0','2017-08-24 20:46:11','2017-08-24
20:46:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(5,299.97,'1',NULL,'0','2017-08-24 20:46:11','2017-08-24
20:46:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(6,3.00,'1',NULL,'0','2017-
08-24 20:46:40','2017-08-24
20:46:40',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(7,266.64,'1',NULL,'0','2017-08-24 20:46:40','2017-08-24
20:46:40',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(8,299.97,'1',NULL,'0','2017-08-24 20:46:40','2017-08-24
20:46:40',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(9,3.00,'1',NULL,'0','2017-
08-24 21:01:10','2017-08-24
21:01:10',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(10,266.64,'1',NULL,'0','2017-08-24 21:01:11','2017-08-24
21:01:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(11,299.97,'1',NULL,'0','2017-08-24 21:01:11','2017-08-24
21:01:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(12,3.00,'1',NULL,'0','2017-
08-24 21:05:56','2017-08-24
21:05:56',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(13,266.64,'1',NULL,'0','2017-08-24 21:05:56','2017-08-24
21:05:56',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(14,299.97,'1',NULL,'0','2017-08-24 21:05:56','2017-08-24
21:05:56',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),(15,3.00,'1',NULL,'0','2017-
08-24 23:07:38','2017-08-24
23:07:38',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(16,266.64,'1',NULL,'0','2017-08-24 23:07:38','2017-08-24
23:07:38',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(17,299.97,'1',NULL,'0','2017-08-24 23:07:38','2017-08-24
23:07:38',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,NULL,NULL),
(18,177.76,'1',NULL,'1','2017-08-25 11:59:03','2017-08-25

```

11:59:03',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'yijia'), (19,2.00,'1',NULL,'1','2017-08-25 11:59:03','2017-
08-25
11:59:03',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'qiandu'), (20,177.76,'1',NULL,'1','2017-08-25
11:59:03','2017-08-25
11:59:03',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'yijia'), (21,177.76,'1',NULL,'1','2017-08-25
23:26:10','2017-08-25
23:26:10',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (22,2.00,'1',NULL,'1','2017-
08-25 23:26:11','2017-08-25
23:26:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL),
(23,177.76,'1',NULL,'1','2017-08-25 23:26:11','2017-08-25
23:26:11',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (24,2.00,'1',NULL,'1','2017-
08-25 23:28:10','2017-08-25
23:28:10',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (25,1.00,'1',NULL,'1','2017-
08-25 23:49:18','2017-08-25
23:49:18',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (26,0.01,'1',NULL,'1','2017-
08-26 00:06:31','2017-08-26
00:06:31',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'修正大
厦','13700221122',NULL,'李佳红',NULL,NULL,NULL,NULL), (27,0.01,'1',NULL,'1','2017-
08-26 00:10:13','2017-08-26
00:10:13',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (28,0.01,'1',NULL,'1','2017-
08-26 00:17:53','2017-08-26
00:17:53',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (29,0.01,'1',NULL,'1','2017-
08-26 00:19:56','2017-08-26
00:19:56',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (30,0.01,'1',NULL,'1','2017-
08-26 00:37:47','2017-08-26
00:37:47',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (31,0.02,'1',NULL,'1','2017-
08-26 00:37:47','2017-08-26
00:37:47',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (32,0.02,'1',NULL,'2','2017-
08-26 00:41:13','2017-08-26
00:41:13',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (33,0.01,'1',NULL,'2','2017-
08-26 00:41:14','2017-08-26
00:41:14',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中騰大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,NULL), (34,0.01,'1',NULL,'1','2017-
08-26 11:57:26','2017-08-26
11:57:26',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'qiandu'), (35,0.01,'1',NULL,'1','2017-08-26
12:21:39','2017-08-26
12:21:39',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'qiandu'), (36,0.01,'1',NULL,'1','2017-08-26
12:34:46','2017-08-26
12:34:46',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'qiandu'),(37,0.01,'1',NULL,'2','2017-08-26 12:47:44','2017-
08-26
12:47:44',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL

L,NULL,NULL,NULL,NULL,'qiandu'), (38,0.02,'1',NULL,'2','2017-08-26
12:47:44','2017-08-26
12:47:44',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,'yijia'), (918159799198212096,400.00,'1',NULL,'1','2017-10-
12 01:02:09','2017-10-12
01:02:09',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'咏春武馆总
部','999111',NULL,'叶问',NULL,NULL,NULL,'qiandu'),
(918334996291301376,2004.00,'2',NULL,'1','2017-10-12 12:38:19','2017-10-12
12:38:19',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','qiandu'),
(918334996698148864,1798.00,'2',NULL,'1','2017-10-12 12:38:19','2017-10-12
12:38:19',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','yijia'),
(918773289399160832,200.00,'1',NULL,'1','2017-10-13 17:39:56','2017-10-13
17:39:56',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'qiandu'),
(918780408353546240,0.01,'1',NULL,'2','2017-10-13 18:08:14','2017-10-13
18:08:14',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'qiandu'),
(918806410454654976,0.01,'1',NULL,'2','2017-10-13 19:51:33','2017-10-13
19:51:33',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中腾大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,'qiandu'),
(918833485639081984,0.01,'1',NULL,'1','2017-10-13 21:39:08','2017-10-13
21:39:08',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中腾大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,'qiandu'),
(918835712441212928,0.01,'1',NULL,'2','2017-10-13 21:47:59','2017-10-13
21:47:59',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中腾大
厦','13301212233',NULL,'李佳星',NULL,NULL,NULL,'qiandu'),
(919055624854081536,0.01,'1',NULL,'1','2017-10-14 12:21:50','2017-10-14
12:21:50',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'中腾大
厦','13301212233',NULL,'李佳星',NULL,NULL,'2','qiandu'),
(919059760869863424,0.02,'1',NULL,'2','2017-10-14 12:38:16','2017-10-14
12:38:16',NULL,NULL,NULL,NULL,NULL,NULL,'lijialong',NULL,NULL,NULL,'金燕龙办公
楼','13900112222',NULL,'李嘉诚',NULL,NULL,'2','qiandu'),
(1055670104882151424,0.46,'',NULL,'1','2018-10-26 11:59:01','2018-10-26
11:59:01',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','qiandu'),
(1055670108459892736,80.00,'',NULL,'1','2018-10-26 11:59:01','2018-10-26
11:59:01',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','sina'),
(1055670108506030080,0.46,'',NULL,'1','2018-10-26 11:59:01','2018-10-26
11:59:01',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','qiandu'),
(1055670108963209216,80.00,'',NULL,'1','2018-10-26 11:59:02','2018-10-26
11:59:02',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','sina'),
(1055670737236393984,0.04,'1',NULL,'1','2018-10-26 12:01:31','2018-10-26
12:01:31',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','qiandu'),
(10556707376795904,50.00,'1',NULL,'1','2018-10-26 12:01:31','2018-10-26
12:01:31',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','sina'),
(1077785650679250944,10451.00,'1',NULL,'1','2018-12-26 12:38:18','2018-12-26
12:38:18',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','1'),
(1077785654370238464,10451.00,'1',NULL,'1','2018-12-26 12:38:18','2018-12-26
12:38:18',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','1'),

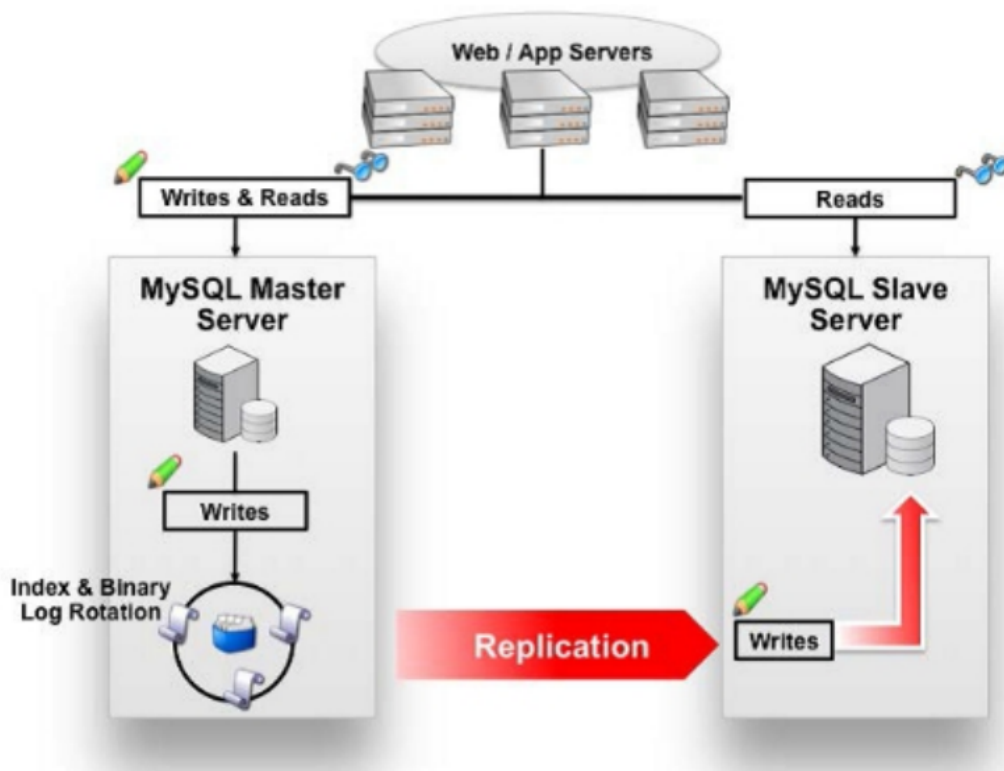

```
(1077785658354827264,10451.00,'1',NULL,'1','2018-12-26 12:38:19','2018-12-26
12:38:19',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,'2','1'),
(1106054512382050304,7512.00,'1',NULL,'1','2019-03-14 12:48:39','2019-03-14
12:48:39',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'),
(1106054517457158144,7512.00,'1',NULL,'1','2019-03-14 12:48:41','2019-03-14
12:48:41',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'),
(1106054520011489280,7512.00,'1',NULL,'1','2019-03-14 12:48:41','2019-03-14
12:48:41',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'),
(1106392066859991040,7512.00,'1',NULL,'1','2019-03-15 11:09:59','2019-03-15
11:09:59',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'),
(1106392071922515968,7512.00,'1',NULL,'1','2019-03-15 11:10:00','2019-03-15
11:10:00',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'),
(1106392074422321152,7512.00,'1',NULL,'1','2019-03-15 11:10:00','2019-03-15
11:10:00',NULL,NULL,NULL,NULL,NULL,NULL,'xiaohong',NULL,NULL,NULL,'永春武
馆','11011011',NULL,'李小龙',NULL,NULL,NULL,'1'));
```

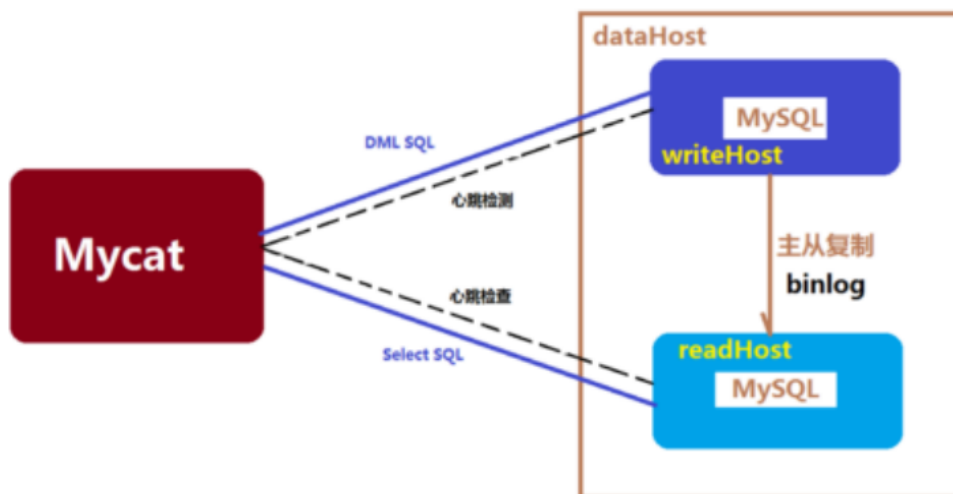
4. 了解数据库的读写分离

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。

Nginx的安装与启动

1.什么是Nginx

Nginx 是一款高性能的 http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。由俄罗斯的程序设计师伊戈尔·西索夫（Igor Sysoev）所开发，官方测试 nginx 能够支撑 5 万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

1.1 Nginx的应用场景

1. **http服务器**。Nginx 是一个 http 服务可以独立提供 http 服务。可以做网页静态服务器。
2. **虚拟主机**。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。
3. **反向代理，负载均衡**。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务

器集群可以使用 nginx 做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而

某台服务器闲置的情况。

1.2 在linux下安装Nginx

1.2.1 环境准备

1. 安装gcc（如果已经安装，省略此步）

```
yum install gcc-c++
```

2. 第三方开发包

```
n PCRE
```

PCRE(Perl Compatible Regular Expressions)是一个 Perl 库，包括 perl 兼容的正则表达式库。nginx的 http 模块使用 pcre 来解析正则表达式，所以需要在 linux 上安装 pcre 库。

```
yum install -y pcre pcre-devel
```

注：pcre-devel 是使用 pcre 开发的一个二次开发库。nginx 也需要此库。

n zlib

zlib 库提供了很多种压缩和解压缩的方式，nginx 使用 zlib 对 http 包的内容进行 gzip，所以需要在 linux 上安装 zlib 库。

```
yum install -y zlib zlib-devel
```

n OpenSSL

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能

及 SSL 协议，并提供丰富的应用程序供测试或其它目的使用。nginx 不仅支持 http 协议，还支持

https（即在 ssl 协议上传输 http），所以需要在 linux 安装 openssl 库。

```
yum install -y openssl openssl-devel
```

1.2.2 Nginx下载

官方网站下载 nginx: <http://nginx.org/>

我们使用的是1.8.0版本

1.2.3 Nginx安装

1. 把 nginx 的源码包nginx-1.8.0.tar.gz上传到 linux 系统
2. 解压 文件到 /usr/local/ 目录下

```
tar zxvf 压缩包 -C /usr/local/
```

3. 进入解压完的nginx-1.8.0目录下使用一下命令来创建 makeFile 文件

```

./configure \
--prefix=/usr/local/nginx \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--with-http_gzip_static_module \
--http-client-body-temp-path=/var/temp/nginx/client \
--http-proxy-temp-path=/var/temp/nginx/proxy \
--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
--http-scgi-temp-path=/var/temp/nginx/scgi

```

执行成功后可以看到makeFile 文件

```

[root@localhost nginx-1.8.0]# ll
total 660
drwxr-xr-x 6 1001 1001  4096 Dec 11 03:42 auto
-rw-r--r-- 1 1001 1001 249124 Apr 21  2015 CHANGES
-rw-r--r-- 1 1001 1001 379021 Apr 21  2015 CHANGES.ru
drwxr-xr-x 2 1001 1001  4096 Dec 11 03:42 conf
-rwxr-xr-x 1 1001 1001  2478 Apr 21  2015 configure
drwxr-xr-x 4 1001 1001  4096 Dec 11 03:42 contrib
drwxr-xr-x 2 1001 1001  4096 Dec 11 03:42 html
-rw-r--r-- 1 1001 1001  1397 Apr 21  2015 LICENSE
-rw-r--r-- 1 root root    345 Dec 11 03:46 Makefile
drwxr-xr-x 2 1001 1001  4096 Dec 11 03:42 man
drwxr-xr-x 3 root root  4096 Dec 11 03:46 objs
-rw-r--r-- 1 1001 1001    49 Apr 21  2015 README
drwxr-xr-x 8 1001 1001  4096 Dec 11 03:42 src
[root@localhost nginx-1.8.0]#

```

---- 知识点小贴士 ----

Makefile是一种配置文件， Makefile 一个工程中的源文件不计数，其按类型、功能、模块分别放在若干 个目录中，makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为 makefile就像一个Shell脚本一样，其中也可以执行操作 系统的命令。

---- 知识点小贴士 ----

configure参数

```

./configure \
--prefix=/usr \ 指向安装目录
--sbin-path=/usr/sbin/nginx \ 指向（执行）程序文件（nginx）
--conf-path=/etc/nginx/nginx.conf \ 指向配置文件
--error-log-path=/var/log/nginx/error.log \ 指向log
--http-log-path=/var/log/nginx/access.log \ 指向http-log
--pid-path=/var/run/nginx/nginx.pid \ 指向pid
--lock-path=/var/lock/nginx.lock \ （安装文件锁定，防止安装 文件被别人利用，或自己误操作。）
--user=nginx \
--group=nginx \
--with-http_ssl_module \ 启用ngx_http_ssl_module支持（使支持 https请求，需已安装 openssl）
--with-http_flv_module \ 启用ngx_http_flv_module支持（提供寻 求内存使用基于时间的偏移量文件）
--with-http_stub_status_module \ 启用ngx_http_stub_status_module支持（获取 nginx自上次启动以来的工作状态）
--with-http_gzip_static_module \ 启用ngx_http_gzip_static_module支持（在线实时压缩输出数据流）

```

```
--http-client-body-temp-path=/var/tmp/nginx/client/ \ 设定http客户端请求临时文件路径
--http-proxy-temp-path=/var/tmp/nginx/proxy/ \ 设定http代理临时文件路径
--http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ \ 设定http fastcgi临时文件路径
--http-uwsgi-temp-path=/var/tmp/nginx/uwsgi \ 设定http uwsgi临时文件路径
--http-scgi-temp-path=/var/tmp/nginx/scgi \ 设定http scgi临时文件路径
--with-pcre 启用pcre库
```

4. 编译

```
make
```

成功后有如下界面：

```
objs/src/http/modules/nginx_http_ssi_filter_module.o \
objs/src/http/modules/nginx_http_charset_filter_module.o \
objs/src/http/modules/nginx_http_userid_filter_module.o \
objs/src/http/modules/nginx_http_gzip_static_module.o \
objs/src/http/modules/nginx_http_autoindex_module.o \
objs/src/http/modules/nginx_http_auth_basic_module.o \
objs/src/http/modules/nginx_http_access_module.o \
objs/src/http/modules/nginx_http_limit_conn_module.o \
objs/src/http/modules/nginx_http_limit_req_module.o \
objs/src/http/modules/nginx_http_geo_module.o \
objs/src/http/modules/nginx_http_map_module.o \
objs/src/http/modules/nginx_http_split_clients_module.o \
objs/src/http/modules/nginx_http_referer_module.o \
objs/src/http/modules/nginx_http_rewrite_module.o \
objs/src/http/modules/nginx_http_proxy_module.o \
objs/src/http/modules/nginx_http_fastcgi_module.o \
objs/src/http/modules/nginx_http_uwsgi_module.o \
objs/src/http/modules/nginx_http_scgi_module.o \
objs/src/http/modules/nginx_http_memcached_module.o \
objs/src/http/modules/nginx_http_empty_gif_module.o \
objs/src/http/modules/nginx_http_browser_module.o \
objs/src/http/modules/nginx_http_upstream_hash_module.o \
objs/src/http/modules/nginx_http_upstream_ip_hash_module.o \
objs/src/http/modules/nginx_http_upstream_least_conn_module.o \
objs/src/http/modules/nginx_http_upstream_keepalive_module.o \
objs/nginx_modules.o \
-lpthread -lcrypt -lpcre -lcrypto -lcrypto -lz
make[1]: Leaving directory `/usr/local/nginx-1.8.0'
make -f objs/Makefile manpage
make[1]: Entering directory `/usr/local/nginx-1.8.0'
sed -e "s|%%PREFIX%%|usr/local/nginx|" \
    -e "s|%%PID_PATH%%|/var/run/nginx/nginx.pid|" \
    -e "s|%%CONF_PATH%%|usr/local/nginx/conf/nginx.conf|" \
    -e "s|%%ERROR_LOG_PATH%%|var/log/nginx/error.log|" \
    < man/nginx.8 > objs/nginx.8
make[1]: Leaving directory `/usr/local/nginx-1.8.0'
[root@localhost nginx-1.8.0]#
```

5. 安装

```
make install
```

成后如下界面：

```
sed -e "s|%%PREFIX%%|usr/local/nginx|" \
-e "s|%%PID_PATH%%|var/run/nginx/nginx.pid|" \
-e "s|%%CONF_PATH%%|usr/local/nginx/conf/nginx.conf|" \
-e "s|%%ERROR_LOG_PATH%%|var/log/nginx/error.log|" \
< man/nginx.8 > objs/nginx.8
make[1]: Leaving directory `usr/local/nginx-1.8.0'
[root@localhost nginx-1.8.0]# make install
make -f objs/Makefile install
make[1]: Entering directory `usr/local/nginx-1.8.0'
test -d '/usr/local/nginx' || mkdir -p '/usr/local/nginx'
test -d '/usr/local/nginx/sbin' || mkdir -p '/usr/local/nginx/sbin'
test ! -f '/usr/local/nginx/sbin/nginx' || mv '/usr/local/nginx/sbin/nginx'
'/usr/local/nginx/sbin/nginx.old'
cp objs/nginx '/usr/local/nginx/sbin/nginx'
test -d '/usr/local/nginx/conf' || mkdir -p '/usr/local/nginx/conf'
cp conf/koi-win '/usr/local/nginx/conf'
cp conf/koi-utf '/usr/local/nginx/conf'
cp conf/win-utf '/usr/local/nginx/conf'
test -f '/usr/local/nginx/conf/mime.types' || cp conf/mime.types '/usr/local/nginx/conf'
cp conf/mime.types '/usr/local/nginx/conf/mime.types.default'
test -f '/usr/local/nginx/conf/fastcgi_params' || cp conf/fastcgi_params '/usr/local/nginx/conf'
cp conf/fastcgi_params '/usr/local/nginx/conf/fastcgi_params.default'
test -f '/usr/local/nginx/conf/fastcgi.conf' || cp conf/fastcgi.conf '/usr/local/nginx/conf'
cp conf/fastcgi.conf '/usr/local/nginx/conf/fastcgi.conf.default'
test -f '/usr/local/nginx/conf/uwsgi_params' || cp conf/uwsgi_params '/usr/local/nginx/conf'
cp conf/uwsgi_params '/usr/local/nginx/conf/uwsgi_params.default'
test -f '/usr/local/nginx/conf/scgi_params' || cp conf/scgi_params '/usr/local/nginx/conf'
cp conf/scgi_params '/usr/local/nginx/conf/scgi_params.default'
test -f '/usr/local/nginx/conf/nginx.conf' || cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf'
cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/var/run/nginx' || mkdir -p '/var/run/nginx'
test -d '/var/log/nginx' || mkdir -p '/var/log/nginx'
test -d '/usr/local/nginx/html' || cp -R html '/usr/local/nginx'
test -d '/var/log/nginx' || mkdir -p '/var/log/nginx'
make[1]: Leaving directory `usr/local/nginx-1.8.0'
[root@localhost nginx-1.8.0]#
```

2.Nginx的启动与访问

注意：启动nginx 之前，上边将临时文件目录指定为/var/temp/nginx/client， 需要在/var下创建此 目

录

```
mkdir /var/temp/nginx/client -p
```

进入到Nginx目录下的sbin目录

```
cd /usr/local/nginx/sbin
```

输入命令启动Nginx

```
./nginx
```

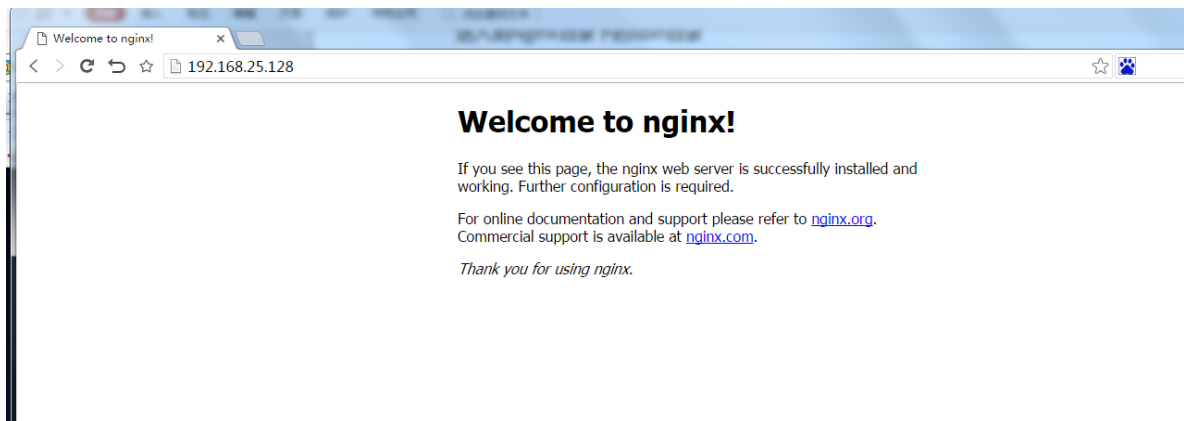
启动后查看进程

```
ps aux|grep nginx
```

```
[root@localhost sbin]# ps aux|grep nginx
root      2141  0.0  0.0  5408  576 ?        Ss   04:00   0:00 nginx: master process ./nginx
nobody    2142  0.0  0.0  5612  916 ?        S    04:00   0:00 nginx: worker process
root      2145  0.0  0.0  4356  720 pts/7    S+   04:00   0:00 grep nginx
[root@localhost sbin]#
```

地址栏输入虚拟机的IP即可访问（默认为80端口）

注意打开虚拟机80端口的防火墙



关闭Nginx:

```
./nginx -s stop
```

或者:

```
./nginx -s quit
```

重启 nginx:

1、先关闭后启动。

2、刷新配置文件:

```
./nginx -s reload
```

3.Nginx静态网站的部署

3.1 静态网站的部署

将我们之前生成的静态页（d:\static）上传到服务器的/usr/local/nginx/html下即可访问

nginx的配置文件 `nginx.conf`

```
server {  
    listen      80;  
    server_name localhost;  
    location / {  
        root    html;  
        index  index.html ;  
    }  
}
```

```
./nginx -s reload
```

//重新加载配置文件然后访问 192.168.25.128 就可以访问了

3.2 配置虚拟主机

虚拟主机，也叫“网站空间”，就是把一台运行在互联网上的物理服务器划分成多个“虚拟”服务器。虚拟主机技术极大的促进了网络技术的应用和普及。同时虚拟主机的租用服务也成了网络时代的一种新型经济形式

3.2.1 端口绑定

```
#端口绑定
server {
    listen      80;
    server_name localhost;
    location / {
        root    html;
        index   index.html ;
    }
}

#端口绑定
server {
    listen      81;
    server_name localhost;
    location / {
        root    html;
        index   index.html ;
    }
}
```

访问192.168.25.128:80

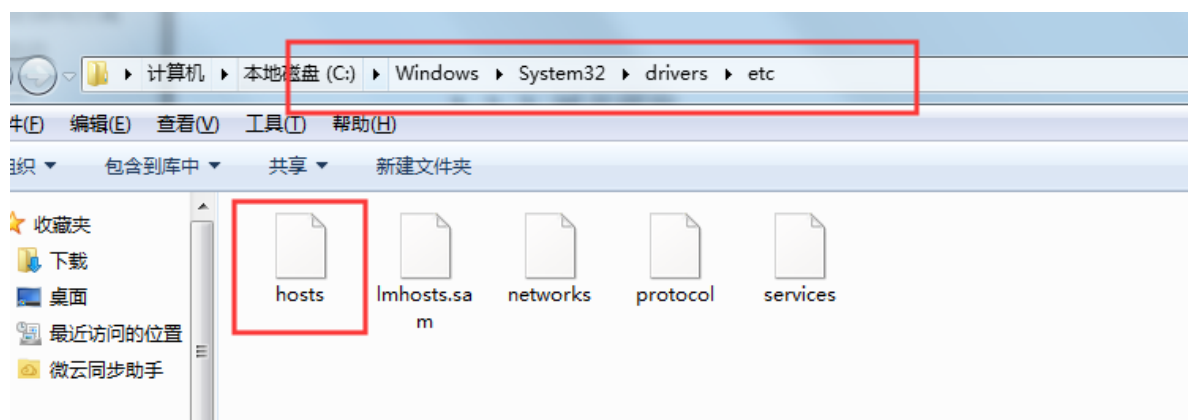
192.168.25.128:81 都是到指定的资源

3.2.2 域名绑定

win+R 运行下面的代码进入到**windows**的本地域名解析文件：**hosts**

C:\Windows\System32\drivers\etc

如果没有就创建一个：





用ip映射域名

4.Nginx的反向代理与负载均衡

4.1 反向代理

nigin.cnf 配置文件:

```
#反向代理并且负载均衡:
upstream tomcat-protal{
    server 192.168.25.128:8180 weight=2;    //tomcat各个端口对应的权重
    server 192.168.25.128:8280 weight=3;
    server 192.168.25.128:8380 weight=1;
    server 192.168.25.128:8480 weight=4;
}
server {
    listen      80;
    server_name www.pinyougou.com;
    location / {
        proxy_pass http://tomcat-protal;    //这里的地址 与上面的upstream后面的
        完全一致
        index index.html;                    //通过这个地址nginx找到tomcat让
        tomcat访问对应                        //的页面
    }
}
```