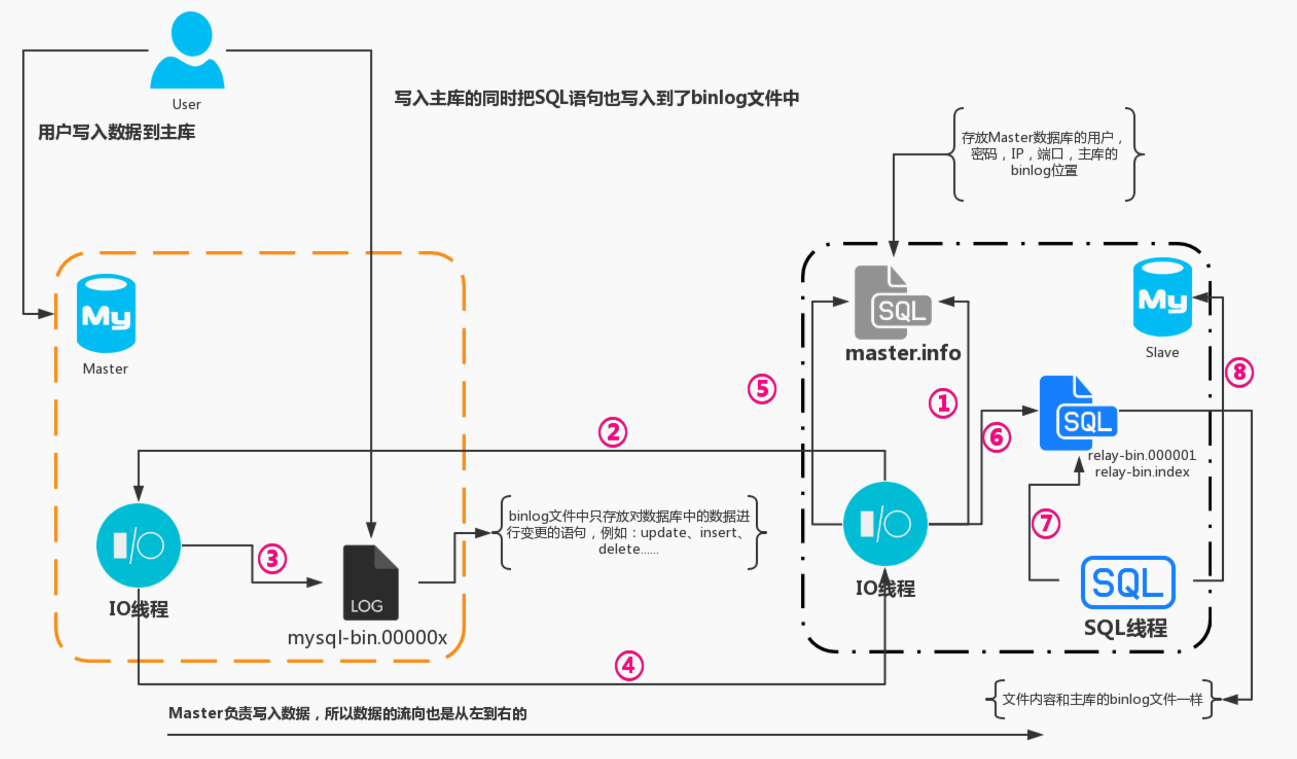
# **[搭建MySQL主从复制经典架构](https://www.cnblogs.com/edisonchou/p/4133148.html)**

MySQL自身提供的一种高可用解决方案，数据同步方法采用MySQL replication技术

(**从服务器到主服务器上拉取二进制binlog日志文件，然后将日志文件解析成相应的sql在从库执行**)



流程说明：

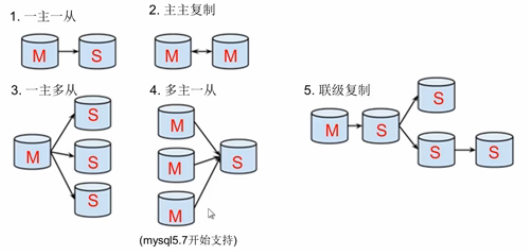
MySQL主从复制之前我们需要先启动Master数据库然后再启动Salve数据库，然后在Salve数据库中执行start slave;，执行完成之后，流程就如下了：

1. Salve的IO线程会读取mastr.info文件中配置好的主库信息，比如说存放的有：Master数据库的用户名、密码、端口、还有Master的binlog索引位置；
2. 拿到信息之后就带着信息去连接Master的主库IO线程
3. 当主库的IO线程先检查Slave传过来的配置信息是否正确，如果正确，就拿着Slave传过来的binlog索引位置和Master库的binlog文件中最后一个索引位置进行对比，如果一致就陷入等待状态，等待Master的binlog索引位置更新；
4. 如果不一致就把Slave传过来的binlog索引位置往后的所有SQL语句包括最后一条SQL语句的索引位置发送个给Slave的IO线程；
5. Slave的IO线程拿到信息之后，先把Master传过来的binlog索引在Slave的master.info文件中进行更新；
6. 然后再把Master传过来的SQL语句写入到relay文件中，然后继续循环执行第二个步骤；
7. Slave的SQL线程会一直持续的观察relay日志文件中是否有改动，如果没有就继续监听；
8. 如果发现relay中有变动，那么就获取变动的内容转换为SQL语句，并且把SQL语句在Salve的数据库中进行执行。

## 主从形式

### mysql主从复制

* 一主一从
* 主主复制
* 一主多从---扩展系统读取的性能，因为读是在从库读取的；
* 多主一从---5.7开始支持
* 联级复制---



### ****用途及条件****

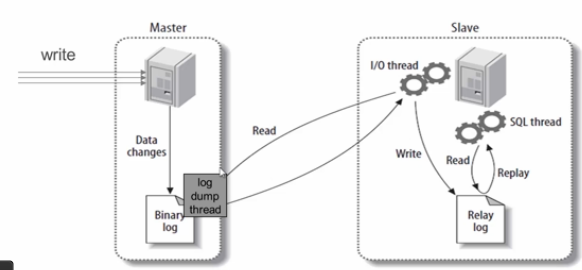
mysql主从复制用途

* 实时灾备，用于故障切换
* 读写分离，提供查询服务
* 备份，避免影响业务

主从部署必要条件：

* 主库开启binlog日志（设置log-bin参数）
* 主从server-id不同
* 从库服务器能连通主库

### ****主从原理****



从库生成两个线程，一个I/O线程，一个SQL线程；

i/o线程去请求主库的binlog，并将得到的binlog日志写到relay log（中继日志）文件中；

主库会生成一个 log dump 线程，用来给从库 i/o线程传binlog；

SQL 线程，会读取relay log文件中的日志，并解析成具体操作，来实现主从的操作一致，而最终数据一致；

### ****问题及解决方法****

mysql主从复制存在的问题：

* 主库宕机后，数据可能丢失
* 从库只有一个sql Thread，主库写压力大，复制很可能延时

解决方法：

* 半同步复制---解决数据丢失的问题
* 并行复制----解决从库复制延迟的问题

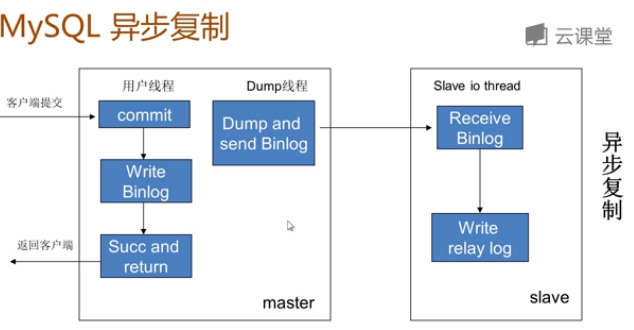
### ****半同步复制****

mysql semi-sync（半同步复制）

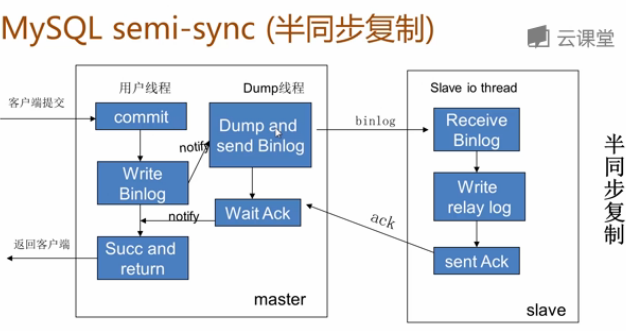
半同步复制：

* 5.5集成到mysql，以插件的形式存在，需要单独安装
* 确保事务提交后binlog至少传输到一个从库
* 不保证从库应用完这个事务的binlog
* 性能有一定的降低，响应时间会更长
* 网络异常或从库宕机，卡主主库，直到超时或从库恢复

主从复制--异步复制原理



半同步复制原理：



### ****并行复制****

mysql并行复制

* 社区版5.6中新增
* 并行是指从库多线程apply binlog
* 库级别并行应用binlog，同一个库数据更改还是串行的(5.7版并行复制基于事务组)

设置

set global slave\_parallel\_workers=10;

设置sql线程数为10

### ****其他****

部分数据复制

主库添加参数：

binlog\_do\_db=db1

binlog\_ignore\_db=db1

binlog\_ignore\_db=db2

或从库添加参数

replicate\_do\_db=db1

replicate\_ignore\_db=db1

replicate\_do\_table=db1.t1

replicate\_wild\_do\_table=db%.%

replicate\_wild\_ignore\_table=db1.%

联级复制（常用）

A->B->C

B中添加参数：

log\_slave\_updates

B将把A的binlog记录到自己的binlog日志中

复制的监控：

show slave status \G

复制出错处理

常见：1062（主键冲突），1032（记录不存在）

解决：

手动处理

跳过复制错误：set global sql\_slave\_skip\_counter=1

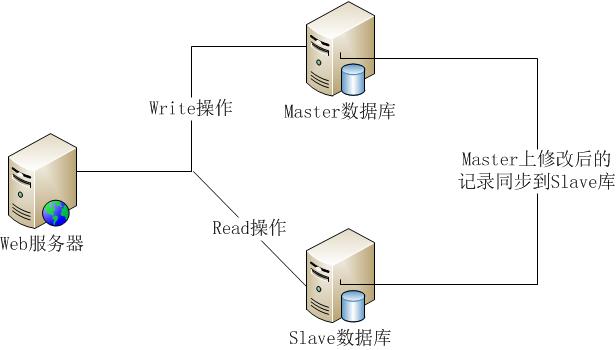
### ****总结****

* mysql主从复制是mysql高可用性，高性能（负载均衡）的基础
* 简单，灵活，部署方式多样，可以根据不同业务场景部署不同复制结构
* 复制过程中应该时刻监控复制状态，复制出错或延时可能给系统造成影响
* mysql主从复制目前也存在一些问题，可以根据需要部署复制增强功能

# **一、业务发展驱动数据发展**

　　随着网站业务的不断发展，用户量的不断增加，数据量成倍地增长，数据库的访问量也呈线性地增长。特别是在用户访问高峰期间，并发访问量突然增大，数据库的负载压力也会增大，如果架构方案不够健壮，那么数据库服务器很有可能在高并发访问负载压力下宕机，造成数据访问服务的失效，从而导致网站的业务中断，给公司和用户造成双重损失。那么，有木有一种方案能够解决此问题，使得数据库不再因为负载压力过高而成为网站的瓶颈呢？答案肯定是有的。

目前，大部分的主流关系型数据库都提供了主从热备功能，通过配置两台（或多台）数据库的主从关系，可以将一台数据库服务器的数据更新同步到另一台服务器上。网站可以利用数据库的这一功能，**实现数据库的读写分离，从而改善数据库的负载压力**。



利用数据库的读写分离，Web服务器在写数据的时候，访问主数据库（Master），主数据库通过**主从复制机制**将数据更新同步到从数据库（Slave），这样当Web服务器读数据的时候，就可以通过从数据库获得数据。这一方案使得在大量读操作的Web应用可以轻松地读取数据，而主数据库也只会承受少量的写入操作，还可以实现数据热备份，可谓是一举两得的方案。

# **二、MySQL数据复制原理**

　　刚刚我们了解了关系型数据库的读写分离能够实现数据库的主从架构，那么主从架构中最重要的数据复制又是怎么一回事呢？MySQL作为最流行的关系型数据库之一，通过了解MySQL的数据复制流程，会使得我们对主从复制的认知会有一定的帮助。



从上图来看，整体上有如下三个步凑：

　　（1）Master将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；

　　（2）Slave将Master的二进制日志事件(binary log events)拷贝到它的中继日志(relay log)；

**PS：**从图中可以看出，Slave服务器中有一个I/O线程(I/O Thread)在不停地监听Master的二进制日志(Binary Log)是否有更新：如果没有它会睡眠等待Master产生新的日志事件；如果有新的日志事件(Log Events)，则会将其拷贝至Slave服务器中的中继日志(Relay Log)。

1. Slave重做中继日志(Relay Log)中的事件，将Master上的改变反映到它自己的数据库中。

**PS：**从图中可以看出，Slave服务器中有一个SQL线程(SQL Thread)从中继日志读取事件，并重做其中的事件从而更新Slave的数据，使其与Master中的数据一致。只要该线程与I/O线程保持一致，中继日志通常会位于OS的缓存中，所以中继日志的开销很小。

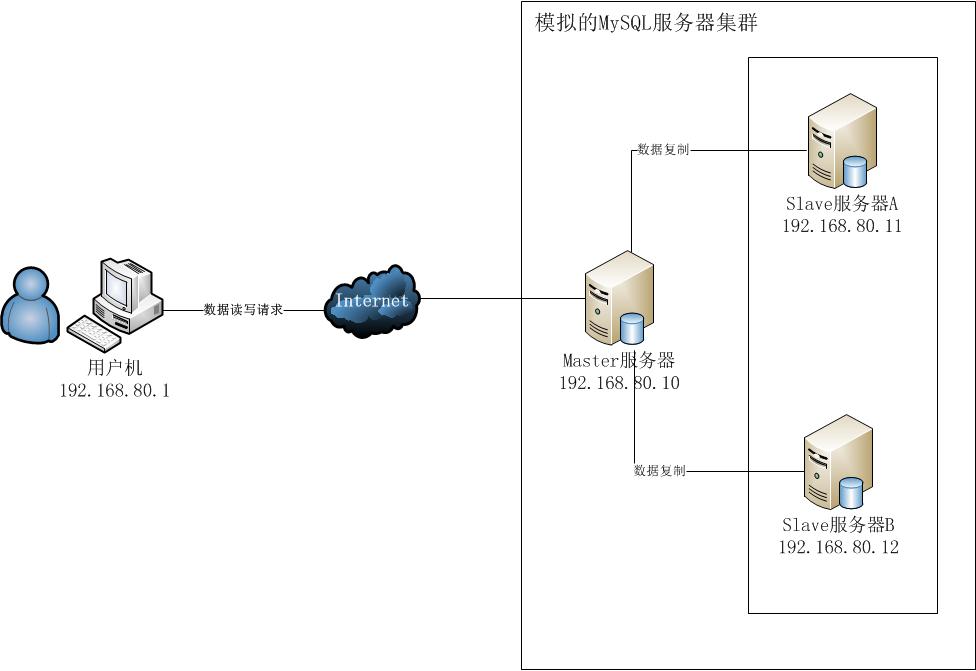
# **三、MySQL主从复制实战**

## **3.1 实验环境总览与准备工作**

　　（1）实验环境

　　①服务器环境：本次我们主要借助VMware Workstation搭建一个三台Windows Server 2003组成的MySQL服务器集群，其中一台作为Master服务器（IP：192.168.80.10），其余两台均作为Slave服务器（IP：192.168.80.11,192.168.80.12）。

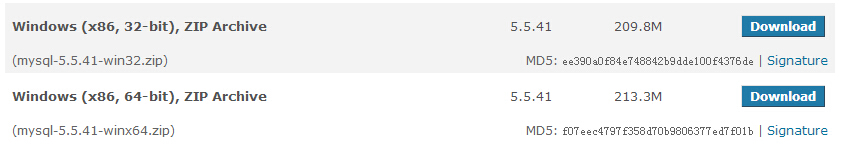
　　②客户机环境：本次我们在Windows 7宿主机（IP：192.168.80.1）编写一个C#控制台程序，对MySQL服务器进行基本的CRUD访问测试。



1. 准备工作

下载MySQL文件：[http://dev.mysql.com/downloads/mysql/5.5.html#downloads](http://dev.mysql.com/downloads/mysql/5.5.html" \l "downloads" \t "https://www.cnblogs.com/edisonchou/p/_blank)

　　　　这里我们选择5.5版本，为了节省时间，直接选择了Archive免安装版本。又由于虚拟机中的Windows Server 2003是32位，所以选择了32-bit的Archive版本进行使用。



　　　　下载完成后，将三个压缩包分别拷贝至Master（IP：192.168.80.10）、Slave1（IP：192.168.80.11）及Slave2（IP：192.168.80.12）中。

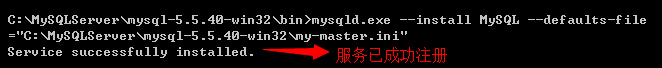
## **3.2 配置MySQL主服务器**

　　（1）将MySQL文件拷贝到Master服务器，并解压到一个指定文件夹。这里我放在了：C:\MySQLServer\mysql-5.5.40-win32

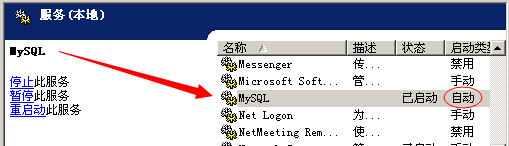
　　（2）新建一个配置文件，取名为：my-master.ini，添加以下内容：

|  |
| --- |
| [client]  port=3306  default-character-set=utf8  [mysqld]  port=3306  #character\_set\_server=utf8 一定要这样写;  character\_set\_server=utf8  #解压目录  basedir=C:\MySQLServer\mysql-5.5.40-win32  #解压目录下data目录,必须为data目录  datadir=C:\MySQLServer\mysql-5.5.40-win32\data  #sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES 这个有问题，在创建完新用户登录时报错  sql\_mode=NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION  #主服务器的配置  #01.开启二进制日志  log-bin=master-bin  #02.使用二进制日志的索引文件  log-bin-index=master.bin.index  #03.为服务器添加唯一的编号  server-id=1 |

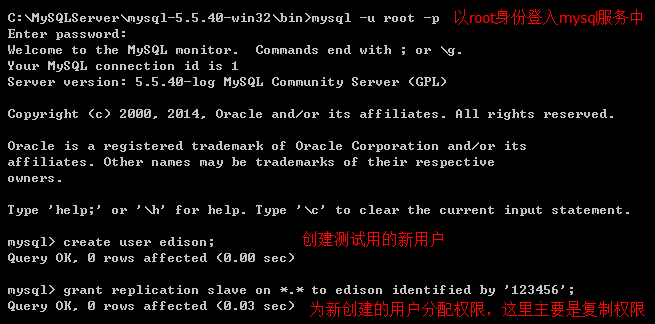
3）将my-master.ini传送到Master服务器中mysql所在的文件夹中，并在命令行中将其注册为Windows服务：（这里要转到mysql的bin文件夹中进行操作，因为没有设置环境变量）



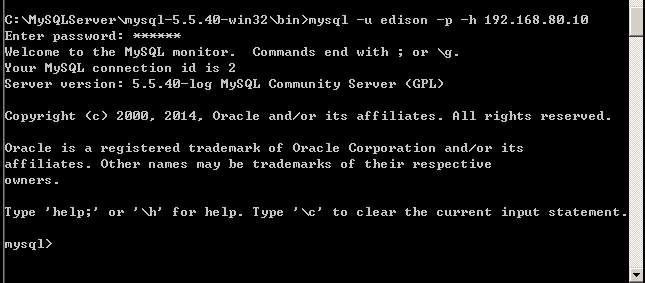
　　（4）启动mysql服务，并设为自启动类型；



　　（5）使用root账号登陆mysql，创建一个具有**复制权限**的用户；（此时root是没有密码的，直接回车即可）



　　（6）在Slave1或Slave2上通过远程登录Master上的mysql测试新建用户是否可以登录；



## **3.3 配置MySQL从服务器**

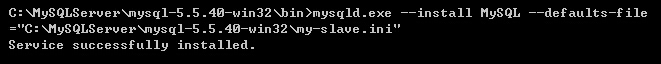
　　（1）同Master服务器，将MySQL文件拷贝解压到指定文件夹下；

　　（2）新建一个配置文件，取名为：my-slave.ini，添加以下内容：

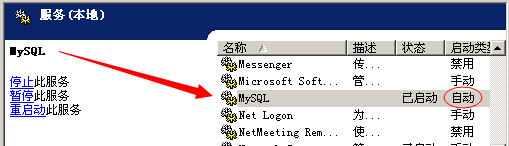
|  |
| --- |
| [client]  port=3306  default-character-set=utf8  [mysqld]  port=3306  #character\_set\_server=utf8 一定要这样写;  character\_set\_server=utf8  #解压目录  basedir=C:\MySQLServer\mysql-5.5.40-win32  #解压目录下data目录,必须为data目录  datadir=C:\MySQLServer\mysql-5.5.40-win32\data  #sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES 这个有问题，在创建完新用户登录时报错  sql\_mode=NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION  #从服务器的配置  #01.为服务器添加唯一的编号  server-id=2  #02.开启中继日志  relay-log=slave-relay-log-bin  #03.使用中继日志的索引文件  relay-log-index=slave-relay-log-bin.index |

**PS：**这里server-id要确保唯一，我们这里Master（192.168.80.10）的server-id=1，那么Slave1（192.168.80.11）就设置其server-id=2，Slave2（192.168.80.12）则设置其server-id=3。

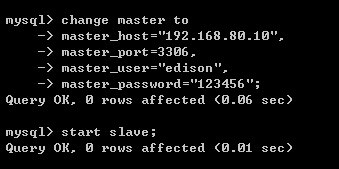
（3）将my-slave.ini传送到Slave1和Slave2服务器中mysql所在的文件夹中，并在命令行中将其注册为Windows服务：（这里要转到mysql的bin文件夹中进行操作，因为没有设置环境变量）



　　（4）分别启动两台Slave的mysql服务，步骤同master所述；当然，也可以在cmd中输入命令：net start MySQL

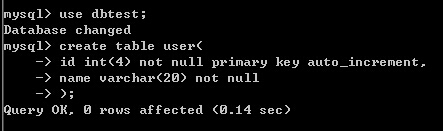


　　（5）分别使用两台Slave的root账号登陆mysql，通过指定的语句配置主从关系设置；

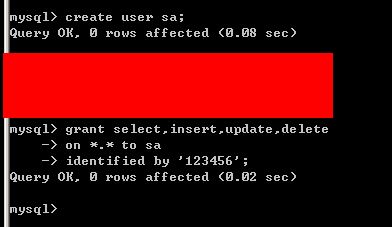


　　（6） 为了方便后面的测试，这里我们在Master上通过root进入mysql，创建一个测试用的数据库和数据表；

IMG_259

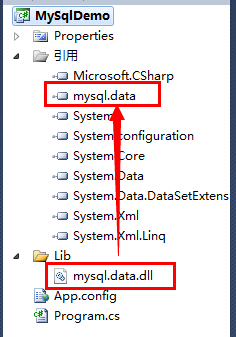


　　（7）还要创建一个用户，这个用户具有对所有数据库的增删查改的权限，以便用来进行测试；



## **3.4 编写C#程序测试主从复制结构**

　　（1）下载mysql for .net开发包，添加对mysql.data.dll的引用



　　（2）在控制台程序中写代码访问Master服务器，并查看程序运行结果；

　　①数据库连接部分：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <configuration>  <connectionStrings>  <add name="mysqlmaster"  connectionString="server=192.168.80.10;database=dbtest;uid=sa;password=123456"/>  </connectionStrings>  </configuration> |

　　②程序代码部分：在程序中首先显示user表内容（这时表是空的），然后会添加5条user信息，其中会修改第3条user信息的name为Edison Chou，最后会删除第5条user信息；

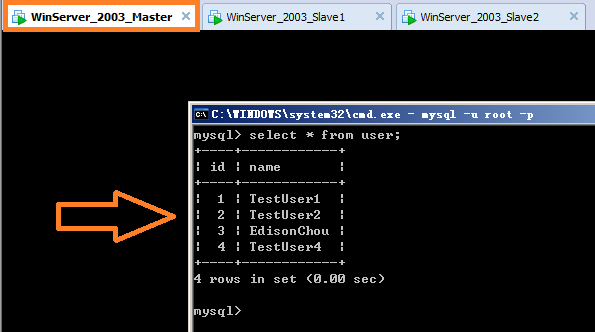
|  |
| --- |
| static void Main(string[] args)  {  string connStr = ConfigurationManager.ConnectionStrings["mysqlmaster"]  .ConnectionString;  // 01.Query ShowUserData(connStr);  // 02.Add a user to table  for (int i = 0; i < 5; i++)  {  AddUserData(connStr, "TestUser" + (i + 1).ToString());  }  ShowUserData(connStr);  // 03.Update a user on table  UpdateUserData(connStr, 3, "EdisonChou");  ShowUserData(connStr);  // 04.Delete a user from table  DeleteUserData(connStr, 5);  ShowUserData(connStr);  Console.ReadKey();  }  #region 01.Func:ShowUserData  private static void ShowUserData(string connStr)  {  using (MySqlConnection con = new MySqlConnection(connStr))  {  con.Open();  using (MySqlCommand cmd = con.CreateCommand())  {  cmd.CommandText = "select \* from user";  using (MySqlDataReader reader = cmd.ExecuteReader())  {  if (reader.HasRows)  {  Console.WriteLine("------------table:user------------");  while (reader.Read())  {  Console.WriteLine(reader[0] + "-" + reader[1]);  }  Console.WriteLine("------------table:user------------");  }  }  }  }  }  #endregion  #region 02.Func:AddUserData  private static void AddUserData(string connStr, string userName)  {  using (MySqlConnection con = new MySqlConnection(connStr))  {  con.Open();  using (MySqlCommand cmd = con.CreateCommand())  {  cmd.CommandText = "insert into user(name) values('" + userName + "')";  int result = cmd.ExecuteNonQuery();  if (result > 0)  {  Console.WriteLine("Add User Successfully.");  }  }  }  }  #endregion  #region 03.Func:UpdateUserData  private static void UpdateUserData(string connStr, int userId,  string userName)  {  using (MySqlConnection con = new MySqlConnection(connStr))  {  con.Open();  using (MySqlCommand cmd = con.CreateCommand())  {  cmd.CommandText = "update user set name='" + userName  + "' where id=" + userId;  int result = cmd.ExecuteNonQuery();  if (result > 0)  {  Console.WriteLine("Update User Successfully.");  }  }  }  }  #endregion  #region 04.Func:DeleteUserData  private static void DeleteUserData(string connStr, int userId)  {  using (MySqlConnection con = new MySqlConnection(connStr))  {  con.Open();  using (MySqlCommand cmd = con.CreateCommand())  {  cmd.CommandText = "delete from user where id=" + userId;  int result = cmd.ExecuteNonQuery();  if (result > 0)  {  Console.WriteLine("Delete User Successfully.");  }  }  }  }  #endregion |

③程序运行结果：

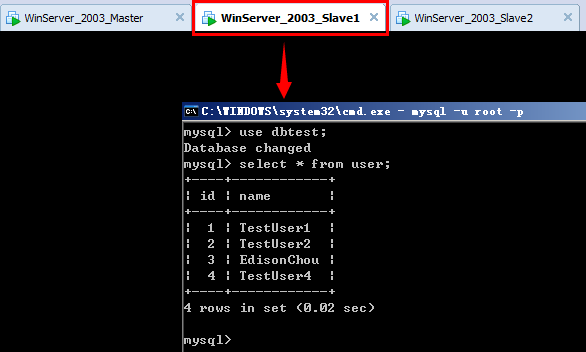


　　（3）在Slave1（192.168.80.11）和Slave2（192.168.80.12）上查看user表是否自动进行了数据同步；

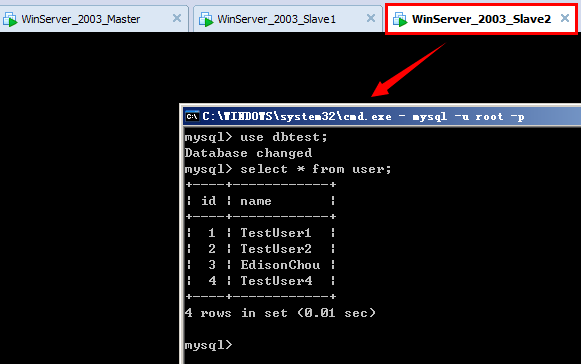
　　①首先在Master上查看user表还剩哪些信息？



　　②其次在Slave1上查看user表是否进行了同步：



　　③最后在Slave2上查看user表是否进行了同步：



　　（4）初步尝试读写分离：**一主一从**模式的一个最简单的实现方式

　①在Slave1上新建一个只具有读（select）权限的用户，这里取名为reader：

　　　　create user reader;

　　　　grant select on \*.\* to reader identified by '123456';

　②新增一个mysqlslave的数据库连接字符串：

|  |
| --- |
| <connectionStrings>  <add name="mysqlmaster"  connectionString="server=192.168.80.10;database=dbtest;uid=sa;password=123456"/>  <add name="mysqlslave"  connectionString="server=192.168.80.11;database=dbtest;uid=reader;password=123456"/>  </connectionStrings> |

③新增一个枚举DbCommandType来记录读操作和写操作：

public enum DbCommandType

{

Read,

Write

}

④修改读取数据表的代码判断是读操作还是写操作：

|  |
| --- |
| private static void ShowUserData(DbCommandType commandType)  {  string connStr = null;  if (commandType == DbCommandType.Write)  {  connStr = ConfigurationManager.ConnectionStrings["mysqlmaster"]  .ConnectionString;  }  else  {  connStr = ConfigurationManager.ConnectionStrings["mysqlslave"]  .ConnectionString;  }  using (MySqlConnection con = new MySqlConnection(connStr))  {  con.Open();  using (MySqlCommand cmd = con.CreateCommand())  {  cmd.CommandText = "select \* from user";  using (MySqlDataReader reader = cmd.ExecuteReader())  {  if (reader.HasRows)  {  Console.WriteLine("------------table:user------------");  while (reader.Read())  {  Console.WriteLine(reader[0] + "-" + reader[1]);  }  Console.WriteLine("------------table:user------------");  }  }  }  }  } |

**PS：**关于MySQL的读写分离实现，主要有以下几种方式：

一种是基于MySQL-Proxy做调度服务器模式，另一种是借助阿里巴巴开源项目**Amoeba**(变形虫)项目实现（这种方式貌似用的比较多），另外呢就是自己写一个类似于哈希算法的程序库来选择目标数据库；

# **学习小结**

此次我们主要简单地学习了主从复制的一些相关概念，了解了MySQL在Windows下搭建主从复制架构的过程，最后通过改变程序方式使得一主一从模式下实现读写分离（虽然是很简单很粗陋的实现）。

<https://www.cnblogs.com/edisonchou/p/4133148.html>