外键约束

1 什么是外键约束

foreign key 就是表与表之间的某种约定的关系，由于这种关系的存在，我们能够让表与表之间的数据，更加的完整，关连性更强。

关于完整性，关连性我举个例子。

有二张表，一张是用户表，一张是订单表：

1》如果我删除了用户表里的用户，那么订单表里面根这个用户有关的数据，就成了无头数据了，不完整了。

2》如果我在订单表里面，随便插入了一条数据，这个订单在用户表里面，没有与之对应的用户。这样数据也不完整了。

如果有外键的话，就方便多了，可以不让用户删除数据，或者删除用户的话，通过外键同样删除订单表里面的数据，这样也能让数据完整。

创建外键约束

外键： 每次插入或更新时，都会检查数据的完整性。

方法一：通过 create table 创建外键

语法：

```
create table 数据表名称(
…,
[CONSTRAINT [约束名称]] FOREIGN KEY [外键字段]
  REFERENCES [外键表名](外键字段,外键字段2.....)
                [ON DELETE CASCADE ]
[ON UPDATE CASCADE ]
)
```

CONSTRAINT constraint [kənˈstreɪnt] 约束

REFERENCES reference [ˈrefrəns] 引用，参考

关于参数的解释：

RESTRICT: 拒绝对父表的删除或更新操作。

CASCADE: 从父表删除或更新且自动删除或更新子表中匹配的行。ON DELETE CASCADE 和 ON UPDATE CASCADE 都可用

注意：on update cascade 是级联更新的意思，on delete cascade 是级联删除的意思，意思就是说当你更新或删除主键表，那外见表也会跟随一起更新或删除。

cascade[英][kæˈskeɪd] 级联

精简化后的语法：

语法：foreign key 当前表的字段 references 外部表名 （关联的字段） type=innodb

注：创建成功，必须满足以下 4 个条件：

1、确保参照的表和字段存在。

2、组成外键的字段被索引。

3、必须使用 type 指定存储引擎为：innodb.

4、外键字段和关联字段，数据类型必须一致。

例：

用户表，表名 user

创建 user 表：

```
CREATE TABLE `user` (
`id` int(11) NOT NULL auto_increment ,
`name` varchar(50) NOT NULL default '',
`sex` int(1) NOT NULL default '0',
PRIMARY KEY (`id`)
) ENGINE=innodb;
```

#创建时，如果表名是 sql 关键字，使用时，需要使用反引号`

插入数据：

```
insert into user (name,sex)values("tank",1),("zhang",2);
mysql> select * from `user` ;
```

```
+----+-------+-----+
| id | name  | sex |
+----+-------+-----+
|  1 | tank  |   1 |
|  2 | zhang |   2 |
+----+-------+-----+
```

创建订单表 如下：

```
mysql> select * from `order`;
+----------+------+----------+-------+
| order_id | u_id | username | money |
+----------+------+----------+-------+
|        1 |    1 | tank     |  2222 |
|        2 |    2 | zhang    |  2222 |
+----------+------+----------+-------+
```

mysql> create table `order` (`order_id` int(11) auto_increment, `u_id` int(11) default '0', `username` varchar(50), `money` int(11), primary key(`order_id`), index (`u_id`) , foreign key order_f_key (u_id) references user(id) on delete cascade on update cascade) type= innodb;

注：on delete cascade on update cascade 添加级联删除和更新：

注：确保参照的表 user 中 id 字段存在。 组成外键的字段 u_id 被索引。 必须使用 type 指定存储引擎为：innodb。

“ FOREIGN KEY order_f_key (u_id)” 中 order_f_key 为外键索引名称

外键字段和关联字段，数据类型必须一致。

在 order 表中插入用户 tank 和 zhang 的数据：

mysql> INSERT INTO `order` (`u_id`, `username`, `money`) VALUES ('1', 'tank','2222');

mysql> INSERT INTO `order` (`u_id`, `username`, `money`) VALUES ('2', 'zhang','2222');

互动： select * from order; 执行不成功的，需要添加`` .

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'order' at line 1

解决：

mysql> select * from `order`;

```
mysql> select * from `order`;
+----------+------+----------+-------+
| order_id | u_id | username | money |
+----------+------+----------+-------+
|        1 |    1 | tank     |  2222 |
|        2 |    2 | zhang    |  2222 |
+----------+------+----------+-------+
2 rows in set (0.00 sec)
```

测试级联删除

mysql> delete from user where id =1; //删除用户名为 1 记录

查看 order 表中 u_id 为 1 的记录已经被删除了。

mysql> delete from user where id=1;

Query OK, 1 row affected (0.07 sec)

mysql> select * from `order`;

```
+----------+------+----------+-------+
| order_id | u_id | username | money |
+----------+------+----------+-------+
|        2 |    2 | zhang    |  2222 |
+----------+------+----------+-------+
1 row in set (0.00 sec)
```

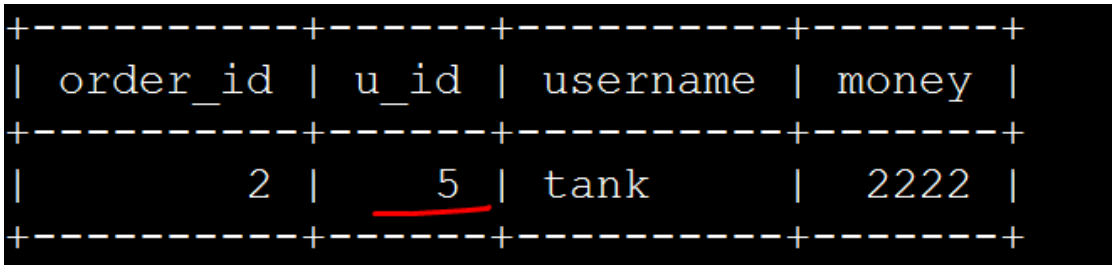
测试级联更新:

更新前的数据：

```
mysql> select * from `order`;
+----------+------+----------+-------+
| order_id | u_id | username | money |
+----------+------+----------+-------+
|        2 |    2 | tank     |  2222 |
+----------+------+----------+-------+
mysql> select * from user;
+----+-------+-----+
| id | name  | sex |
+----+-------+-----+
|  2 | zhang |   2 |
+----+-------+-----+
```

将 user 表中 zhang 的 id 改为 5.

```
mysql>  update user set id=5 where id=2;
mysql> select * from `order`;  #查看 order 表中的数据也更新为 5 了
```



```
+----------+------+----------+-------+
| order_id | u_id | username | money |
+----------+------+----------+-------+
|        2 |    5 | tank     |  2222 |
+----------+------+----------+-------+
```

测试数据完整性：

```
//下面在 order 里面插入一条数据 u_id 为 6 用户，在 user 表里面根本没有，所以插入不进去。
mysql> INSERT INTO `order` (`u_id`, `username`, `money`) VALUES ('6', 'good','123');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`bb`.`order`, CONSTRAINT `order_ibfk_1` FOREIGN KEY (`u_id`) REFERENCES `user` (`id`))

mysql> insert into user values(6,"grace",2);   #现在在 user 中添加一个 6

mysql> INSERT INTO `order` (`u_id`, `username`, `money`) VALUES ('6', 'good','2222' );

mysql> select * from `order`;
```

order_id	u_id	username	money
2	5	tank	2222
5	6	good	2222

外键约束，order 表受 user 表的约束

在 order 里面插入一条数据 u_id 为 5 用户，在 user 表里面根本没有，所以插入不进去

mysql> insert into user values(5,'Find',1);

mysql> insert into `order` (u_id,username,money)values(5,'Find',346);

```
mysql> select * from `order`;
+------+------+----------+-------+
| o_id | u_id | username | money |
+------+------+----------+-------+
|    2 |    6 | LB       |   146 |
|    3 |    3 | HPC      |   256 |
|    5 |    5 | Find     |   346 |
+------+------+----------+-------+
```

2.通过 alter table 创建外键和级联更新，级联删除

alter table 数据表名称 add

[constraint [约束名称]] foreign key (外键字段,..) references 数据表(参照字段,...)

[on update cascade|set null|no action]

[on delete cascade|set null|no action]

)

mysql> alter table order1 add foreign key(u_id) references user(id) on delete cascade on update cascade;

mysql> create table order1(o_id int(11) auto_increment, u_id int(11) default '0', username varchar(50), money int(11), primary key(o_id), index(u_id))type=innodb;

mysql> alter table order1 add foreign key(u_id) references user(id) on delete cascade on update cascade,type=innodb;

mysql> alter table order1 add constraint `bk`foreign key(u_id) references user(id) on delete cascade on update cascade,type=innodb; 指定外键名称

一定要记得带上 innodb

mysql> show create table order1;

```
-----+
| order1 | CREATE TABLE `order1` (
  `o_id` int(11) NOT NULL AUTO_INCREMENT,
  `u_id` int(11) DEFAULT '0',
  `username` varchar(50) DEFAULT NULL,
  `money` int(11) DEFAULT NULL,
  PRIMARY KEY (`o_id`),
  KEY `u_id` (`u_id`),
  CONSTRAINT `order1_ibfk_1` FOREIGN KEY (`u_id`) REFERENCES `user` (`id`)
 ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+--------+-----------------------------------------------------------------
```

删除外键

alter table 数据表名称 drop foreign key 约束（外键）名称

mysql> alter table order1 drop foreign key order1_ibfk_1;

mysql> show create table order1;

三、什么是视图

视图：是一张虚拟表，由 select 语句指定的数据结构和数据。不生成真实的文件。

语法：create view 视图名称（即虚拟的表名） as select 语句。

我们在怎样的场景使用它，为什么使用视图

如果某个查询结果出现的非常频繁，也就是，要经常拿这个查询结果来做子查询这种

视图能够简化用户的操作

视图机制用户可以将注意力集中在所关心的数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，并且可以简化用户的数据查询操作

视图是用户能以不同的角度看待同样的数据。

对于固定的一些基本表，我们可以给不同的用户建立不同的视图，这样不同的用户就可以看到自己需要的信息了。

视图对重构数据库提供了一定程度的逻辑性。

比如原来的 A 表被分割成了 B 表和 C 表，我们仍然可以在 B 表和 C 表的基础上构建一个视图 A，而使用该数据表的程序可以不变。

视图能够对机密数据提供安全保护

比如说，每门课的成绩都构成了一个基本表，但是对于每个同学只可以查看自己这门课的成绩，因此可以为每个同学建立一个视图，隐藏其他同学的数据，只显示该同学自己的

适当的利用视图可以更加清晰的表达查询数据。

有时用现有的视图进行查询可以极大的减小查询语句的复杂程度。

创建视图

语法：create view 视图名称（即虚拟的表名） as select 语句。

我们在 book 数据库中操作

mysql> create view bc as select b.bName ,b.price ,c.bTypeName from books as b left join category as c on b.bTypeId=c.bTypeId ;

试图可以按照普通表去访问。

另外视图表中的数据和原数据表中数据时同步的。

show create view bc; 查看试图的创建信息。

mysql> show create view bc;

| bc | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `bc` AS select `b`.`bName` AS `bName`,`b`.`price` AS `price`,`c`.`bTypeName` AS `bTypeName` from (`books` `b` left join `category` `c` on((`b`.`bTypeId` = `c`.`bTypeId`))) | latin1 | latin1_swedish_ci |

+------+---

例：使用视图查看数据

mysql> select * from bc \G

。。。

*************************** 44. row ***************************

bName: XML 完全探索

price: 104

bTypeName: 网站

等同于：

mysql> select b.bName ,b.price ,c.bTypeName from books as b left join category as c on b.bTypeId=c.bTypeId ;

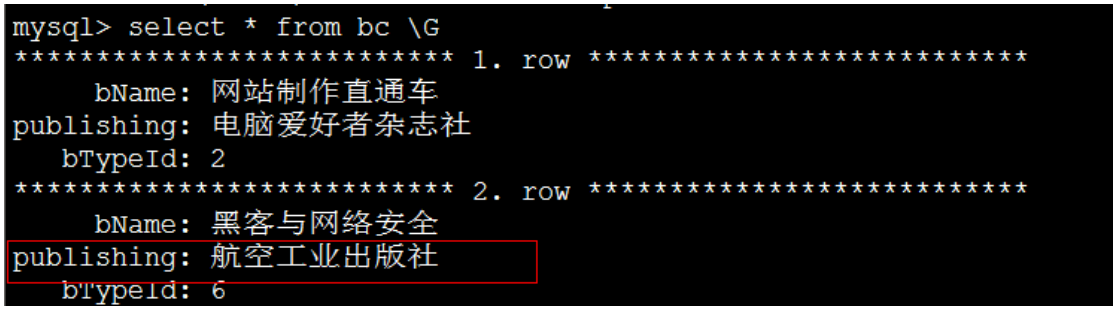
更新或修改视图

语法：

alter view 视图名称（即虚拟的表名） as select 语句。

update view 视图名称（即虚拟的表名）set

mysql> alter view bc as select b.bName ,b.publishing ,c.bTypeId from books as b left join category as c on b.bTypeId=c.bTypeId ;



更新

mysql> update bc set bName='HA' where bTypeId=34;

表中没有价格，所以不显示

删除试图：

drop view 视图名。

例：

mysql> drop view bc;

创建普通用户的三种方法：

3 删除用户：

drop user 用户名@localhost。

msyql 主从复制服务概述：

主从复制

复制解决的基本问题是让一台服务器的数据和另外的服务器保持同步。

一台主服务器可以连接多台从服务器，并且从服务器也可以反过来作主服务器。

主服务器和从服务器可以位于不同的网络拓扑中，还能对整台服务器、特定的数据库，甚至特定的表进行复制。

主从服务器的版本必须一致，即使不一致，主服务器版本可以是旧的，从服务器必须是新的版本。

原理：

MYSQL 支持单向、异步复制，复制过程中一个服务器充当主服务器，而一个或多个其它服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护日志文件的一个索引以跟踪日志循环。

当一个从服务器连接到主服务器时，它通知主服务器从服务器在日志中读取的最后一次成功更新的位置。从服务器接收从那时起发生的任何更新，然后封锁并等待主服务器通知下一次更新。
注意:进行复制时，所有对复制中的表的更新必须在主服务器上进行。以避免后期对主服务器上的表进行的更新与对从服务器上的表所进行的更新之间的冲突。

1.1 mysql 支持的复制类型：

（1）：基于语句的复制： 在主服务器上执行的 SQL 语句，在从服务器上执行同样的语句。MySQL 默认采用基于语句的复制，效率比较高。

（2）：一旦发现没法精确复制时， 会自动选着基于行的复制。基于行的复制，把改变的内容复制过去，而不是把命令在从服务器上执行一遍. 从 mysql5.0 开始支持

（3）：混合类型的复制: 默认采用基于语句的复制，一旦发现基于语句的无法精确的复制时，就会采用基于行的复制。

1.2 . 复制解决的问题

MySQL 复制技术有以下一些特点：

(1) 数据分布 (Data distribution)

(2) 负载平衡(load balancing)

(3) 数据备份(Backups) 远程容灾

(4) 高可用性和故障转移 High availability and failover

（5） 升级测试

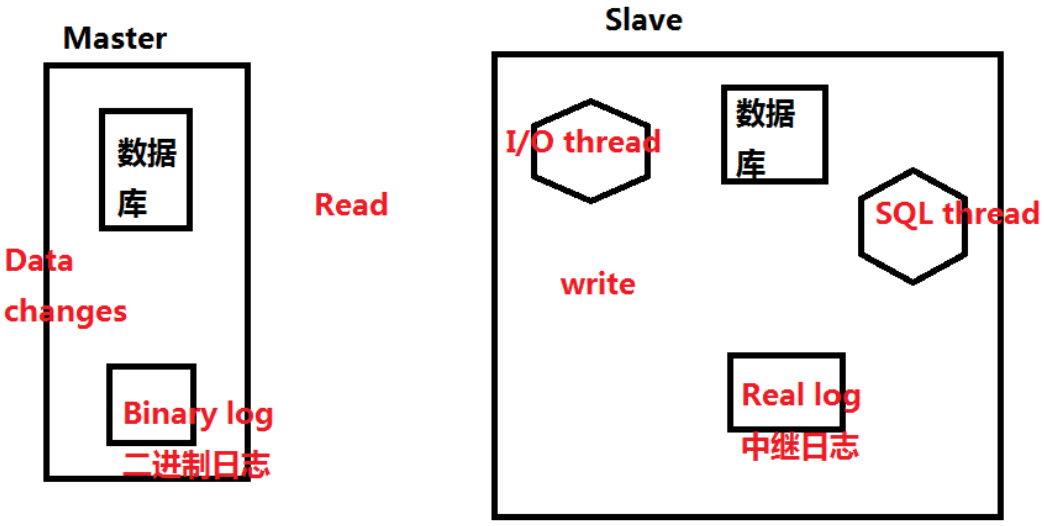
1.3 复制如何工作

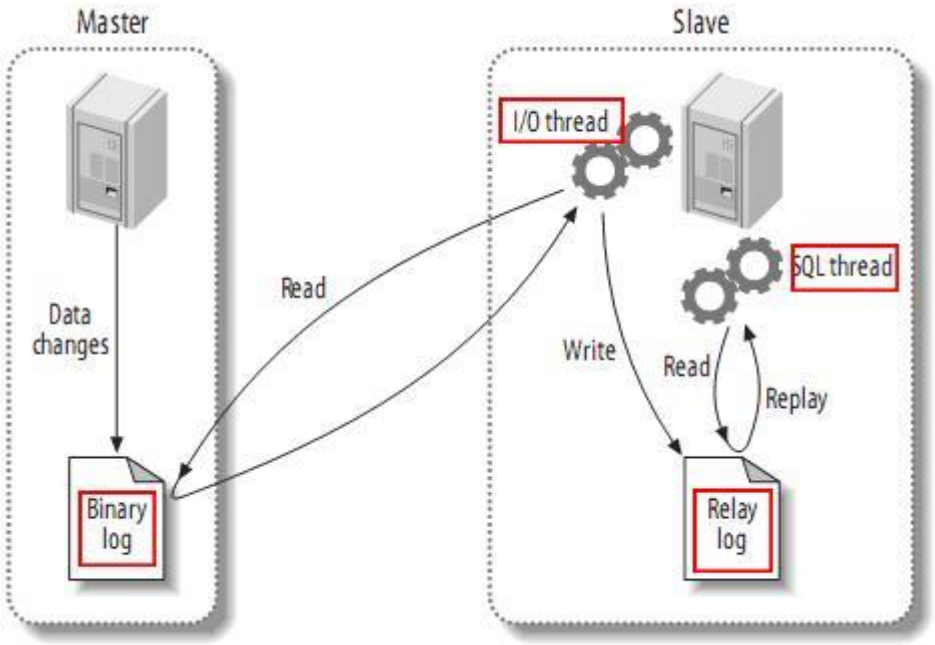
整体上来说，复制有 3 个步骤：

(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 binary log events 拷贝到它的中继日志(relay log)；

(3) slave 重做中继日志中的事件，修改 salve 上的数据。





binary log bin-log 日志
Iothread I/0 进程
relay log 中继日志
SQL thread SQL 线程，作用：重放中继日志里面的记录
Date changes 数据更改
Master：主库
Slave 备库

mysql 主从复制中：

第一步：master 记录二进制日志。在每个事务更新数据完成之前，master 在二日志记录这些改变。MySQL 将事务写入二进制日志，即使事务中的语句都是交叉执行的。在事件写入二进制日志完成后，master 通知存储引擎提交事务。

第二步：slave 将 master 的 binary log 拷贝到它自己的中继日志。首先，slave 开始一个工作线程——I/O 线程。I/O 线程在 master 上打开一个普通的连接，然后开始 binlog dump process。Binlog dump process 从 master 的二进制日志中读取事件，如果已经执行完 master 产生的所有文件，它会睡眠并等待 master 产生新的事件。I/O 线程将这些事件写入中继日志。

第三步：SQL slave thread（SQL 从线程）处理该过程的最后一步。SQL 线程从中继日志读取事件，并重新执行其中的事件而更新 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。

此外，在 master 中也有一个工作线程和其它 MySQL 的连接一样，slave 在 master 中打开一个连接也会使得 master 开始一个线程。复制过程有一个很重要的限制——复制在 slave 上是串行化的，也就是说 master 上的并行更新操作不能在 slave 上并行操作。

常见的复制拓扑
一主库多备库
主主
主库—分发主库—多个备库：采用 blackhcle 存储引擎的分发主库

模式： C/S 模式

端口：
3306

xuegod63 主 mysql 服务器配置
创建要同步的数据库：

```
mysql> create database tree;
mysql> use tree;
mysql> create table test1 (id int);
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| tree               |
| mysql              |
| test               |
```

停止
#service mysqld stop

编辑
#vim /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0 #在原配置文件中，添加以下内容：
log-bin=mysqllog #启用二进制日志，默认存在/var/lib/mysql 下面
server-id=1 #本机数据库 ID 标示。其中 master_id 必须为 1 到 232 之间的一个正整数值
binlog-do-db=tree #可以被从服务器复制的库。二进制需要同步的数据库名
binlog-ignore-db=tree2 不可以被从服务器复制的库

重新启动

#service mysqld restart

授权

mysql> grant replication slave on *.* to slave@192.168.1.64 identified by "123456";

查看状态，信息

mysql> show master status;

```
+----------------+----------+--------------+------------------+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------+----------+--------------+------------------+
| mysqllog.000001 |      242 | tree         |                  |
+----------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

查看二进制日志位置：

[root@xuegod63 ~]# ls /var/lib/mysql/

ibdata1 ib_logfile1 mysql mysqllog.index passwd

ib_logfile0 tree mysqllog.000001 mysql.sock test

mysql>show binlog events \G

复制前保证两个数据库数据一致：

mysqldump -u root -p123456 -A >all.sql

把数据传给从：

方法 1：scp all.sql 192.168.1.64:/root

方法 2：使用 nc 命令

NetCat，它短小精悍、功能实用，被设计为一个简单、可靠的网络工具，可通过 TCP 或 UDP 协议传输读写数据。同时，它还是一个网络应用 Debug 分析器，因为它可以根据需要创建各种不同类型的网络连接。

语法：

服务器端：nc 发送数据的语法： nc -l 端口 < 要传输的文件

客户端： nc 接受数据的语法： nc 远程 nc 服务器端 IP 端口 > 文件名

例：监听 9999 端口，当有客户端连接时，就把对应文件传送到来连接的客户端

开启监听：

[root@xuegod63 ~]#nc -l 9999 < all.sql

[root@xuegod63 ~]# netstat -antup | grep 9999

tcp 0 0 0.0.0.0:9999 0.0.0.0:* LISTEN 15164/nc

测试接收：
xuegod64 ：
[root@xuegod64 ~]# cd /opt/
[root@xuegod64 opt]# ls
[root@xuegod64 opt]# nc 192.168.1.63 9999 > ncall.sql
[root@xuegod64 opt]# diff ncall.sql /root/all.sql

nc 除了传输文件，也可以传输字符串
使用 nc 命令发送字符：
发送端： echo "hello hacker ! " | nc -l 5140

接受端：
[root@xuegod63 ~]# nc 192.168.1.63 5140
hello hacker !

mysql 从服务器: xuegod64
[root@xuegod64 opt]# yum install mysql-server -y
[root@xuegod64 opt]# service mysqld start
mysql>show variables like '%version%';查看版本
+-------------------------+--------------------+
| Variable_name | Value |
+-------------------------+--------------------+
protocol_version	10
version	5.1.52
version_comment	Source distribution
version_compile_machine	x86_64
version_compile_os	redhat-linux-gnu
+-------------------------+--------------------+
5 rows in set (0.03 sec)

测试连接到主服务器是否成功
[root@xuegod64 opt]# mysql -u slave -h 192.168.1.63 -p123456
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| test |
+--------------------+

看不到 tree 数据库

导入数据库，和主服务器保持一致
[root@xuegod64 ~]# mysql -u root -p < all.sql

从服务器没必要开 bin-log 日志注。
修改从服务器配置文件：
[root@xuegod64 ~]# vim /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
 # Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
#在配置文件中写入以下内容
server-id=2 #从服务器 ID 号，不要和主 ID 相同 ，如果设置多个从服务器，每个从服务器必须有一个唯一的 server-id 值，必须与主服务器的以及其它从服务器的不相同。可以认为 server-id 值类似于 IP 地址：这些 ID 值能唯一识别复制服务器群集中的每个服务器实例。

master-host=192.168.1.63 #指定主服务器 IP 地址
master-user=slave #制定在主服务器上可以进行同步的用户名
master-password=123456 #密码
####以下可以不写
master-port = 3306 #同步所用的端口
master-connect-retry=60 #断点重新连接时间
保存，重启
#service mysqld restart

如果只做为备库，就只设置 server-ID， 如果他也作为一个分发主库，开启 bin-log 和中继日志
测试：
主服务器上查看：
mysql> show master status;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 2
Current database: tree1

+----------------+----------+--------------+------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------+----------+--------------+------------------+

```
| mysqllog.000004 |      106 | tree1        |                  |
+-----------------+----------+--------------+------------------+
```

从服务器上查看：

```
mysql> show slave status \G
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: 192.168.1.63
                Master_User: slave
                Master_Port: 3306
              Connect_Retry: 60
            Master_Log_File: mysqllog.000001
        Read_Master_Log_Pos: 315
             Relay_Log_File: mysqld-relay-bin.000002
              Relay_Log_Pos: 459
      Relay_Master_Log_File: mysqllog.000001
           Slave_IO_Running: Yes   #可以看到这两个 Yes，说明从服务器安装成功。
          Slave_SQL_Running: Yes
```

Slave_IO_Running ：一个负责与主机的 io 通信

Slave_SQL_Running：负责自己的 slave mysql 进程

测试：数据同步

xuegod63 写数据：

```
mysql> use tree;
Database changed
mysql> show tables;
+--------------+
| Tables_in_tree |
+--------------+
| test1        |
+--------------+
1 row in set (0.00 sec)

mysql> insert into test1 values(1);
```

xuegod64读数据：

```
mysql> use tree;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

```
mysql> select * from test1;
+------+
| id   |
+------+
|    1 |
+------+
```

排错：

如果遇到主从不同步，看一下主从 bin-log 的位置，然后再同步。

同步之前如果怀疑主从数据不同步可以采取：上面冷备份远程拷贝法或者在从服务器上命行同步方法。

主服务器：

[root@xuegod63 mysql]# mysqldump -u root -p -B book > v4.sql

加-B，执行时，会创建 book 库：

```
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `book` /*!40100 DEFAULT CHARACTER SET lati
n1 */;

USE `book`;
```

或

[root@xuegod63 mysql]# mysqldump -u root -p --all-databases > v5.sql

[root@xuegod63 mysql]# scp v5.sql 192.168.1.64:/root

查看 binlog 状态：

```
mysql> show master status ;
+---------- -------+----------+--------------+------------------+
| File            | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------- ------+----------+--------------+------------------+
| mysqllog.000004 |      106 | tree1        |                  |
+-----------------+----------+--------------+------------------+
```

从服务器执行

[root@xuegod64 ~]# mysql -u root -p < v5.sql   #导入数据库

从服务器执行 MySQL 命令下：

mysql> slave stop；   #先停止 slave 服务

mysql> change master to master_log_file='mysqllog.000004' ,master_log_pos=106;

#根据上面主服务器的 show master status 的结果，进行从服务器的二进制数据库记录回归，达到同步的效果

mysql>slave start;   #启动从服务器同步服务

mysql> show slave status\G;

用 show slave status\G;看一下从服务器的同步情况

Slave_IO_Running: Yes

Slave_SQL_Running: Yes

如果都是 yes，那代表已经在同步

重启从服务器，再查看状态：
停止从服务器 slave stop;
开启从服务器 slave start;

如果出错：
1、二进制日志没有开启
2、IPTABLES 没有放开端口
3、对应的主机 IP 地址写错了

SQL 线程出错
1、主从服务器数据库结构不统一

出错后，数据少，可以手动解决创建插入，再更新 slave 状态。
注：如果主上误删除了。那么从上也就误删除了。 #因此主上要定期做 mysqldump 备份。

实战 2：mysql 主主 双向主从复制：
通过 mysql 主主：进行 mysql 双向同步数据库 tree1 的配置

mysql 主：服务端：xuegod63.cn IP：192.168.1.63
mysql 主：服务端：xuegod64.cn IP：192.168.1.64

配置 xuegod64 ： 身份 1： xuegod64 的主。 身份 2： xuegod63 的从。

```
[root@xuegod64 ~]# vim /etc/my.cnf
 [mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

server-id=2
master-host=192.168.1.63
master-user=slave
master-password=123456

####to master
log-bin=mysqlslave-bin-log
binlog-do-db=tree
```

binlog-ignore-db=mysql #避免同步 mysql 用户 相关配置。

授权：

[root@xuegod64 opt]# mysql -u root -p123456

mysql> grant replication slave on *.* to slave64@'192.168.1.63' identified by '123456';

[root@xuegod64 opt]# service mysqld restart

配置 xuegod63 ： 身份 1： xuegod64 的主。 身份 2： xuegod64 的从。

[root@xuegod63 mysql]# vim /etc/my.cnf

[root@xuegod63 mysql]# cat !$

cat /etc/my.cnf

[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

user=mysql

Disabling symbolic-links is recommended to prevent assorted security risks

symbolic-links=0

log-bin=mysqllog

server-id=1

binlog-do-db=tree

to xuegod64 slave mysql

master-host=192.168.1.64

master-user=slave64

master-password=123456

replicate-do-db=tree

[root@xuegod63 ~]# service mysqld restart

测试：

xuegod64

mysql> show master status ;

```
+--------------------------+----------+--------------+------------------+
| File                     | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------------+----------+--------------+------------------+
| mysqlslave-bin-log.000001 |      106 | tree         | mysql            |
+--------------------------+----------+--------------+------------------+
```

1 row in set (0.00 sec)

#说明 xuegod64 作为 mysql 主已经成功。

xuegod63

[root@xuegod63 mysql]# service mysqld restart

```
Stopping mysqld:                                           [  OK  ]
Starting mysqld:                                           [  OK  ]

mysql> show slave status \G;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    4
Current database: tree

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.64
                  Master_User: slave64
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysqlslave-bin-log.000001
          Read_Master_Log_Pos: 106
               Relay_Log_File: mysqld-relay-bin.000002
                Relay_Log_Pos: 260
        Relay_Master_Log_File: mysqlslave-bin-log.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: tree
          Replicate_Ignore_DB:
```

测试主主数据同步：

xuegod64

```
mysql> show master status ;
+--------------------------+----------+--------------+------------------+
| File                     | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------------+----------+--------------+------------------+
| mysqlslave-bin-log.000001 |      106 | tree         | mysql            |
+--------------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

#说明 xuegod64 作为 mysql 主已经成功。

xuegod63

```
[root@xuegod63 mysql]# service mysqld restart
Stopping mysqld:                                           [  OK  ]
Starting mysqld:                                           [  OK  ]
```

```
mysql> show slave status \G;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    4
Current database: tree

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.64
                  Master_User: slave64
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysqlslave-bin-log.000001
          Read_Master_Log_Pos: 106
               Relay_Log_File: mysqld-relay-bin.000002
                Relay_Log_Pos: 260
        Relay_Master_Log_File: mysqlslave-bin-log.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: tree
          Replicate_Ignore_DB:
```

测试主主数据同步：

在 xuegod63 上插入数据：

```
mysql> use tree;
Database changed
mysql> insert into test1 values(66);
Query OK, 1 row affected (0.00 sec)
```

在 xuegod64 上插入数据：

```
mysql> use tree;
Database changed
mysql> insert into test1 values(77);
```

在 xuegod63 上查看数据：

```
mysql> select * from test1;
+------+
| id   |
+------+
|    1 |
|    2 |
```

| 3 |
| 4 |
| 5 |
| 66 |
| 77 |
+------+

在 xuegod64 上查看数据：
mysql> select * from test1;
+------+
| id |
+------+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 66 |
| 77 |
+------+
总结：说明主主配置成功。

扩展:
mysql 主从配置常见参数：
root@xuegod64 ~]# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
#添加以下内容：
server-id = 2 从服务器 ID 号，不要和主 ID 相同
master-host = 192.168.1.63 指定主服务器 IP 地址
master-user = slave 制定在主服务器上可以进行同步的用户名
master-password = 123456 密码

master-port = 3306	同步所用的端口，可以不写
master-connect-retry=60	断点重新连接时间，可以不写
replicate-ignore-db=mysql	屏蔽从服务器对 mysql 库的同步
replicate-do-db=tree1	指定从服务器要同步数据库的名称。 #不写也可以。
log-bin=mysqlslave-bin-log	设定生成的二进制文件 log 文件名
binlog-do-db=db1	设置主服务器要同步数据库名。即指定 xuegod64 上要产生的二进制日志的数据库名。 #这个是主服务器上配置
binlog-ignore-db=mysql	避免同步 mysql 用户配置，以免不必要的麻烦。#这个是主服务器上配置