目录：

[1、网络基础 2](#_Toc418499242)

2、nginx服务19

3、MySQL数据库44

4、FastCGI介绍53

5、安装PHP 54

6、安装wordpress博客61

# 1、MySQL插入中文数据乱码问题

mysql> show variables like "character\_set%"; #查看字符集命令 \*\*\*\*\*

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

| Variable\_name | Value |

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

| character\_set\_client | utf8 #客户端字符集（set names） |

| character\_set\_connection | utf8 #客户端字符集（set names） |

| character\_set\_database | utf8 #数据库端字符集 |

| character\_set\_filesystem | binary #文件系统字符集 |

| character\_set\_results | utf8 #客户端返回结果字符集 set names |

| character\_set\_server | utf8 #服务器端字符集 |

| character\_set\_system | utf8 #系统字符集 |

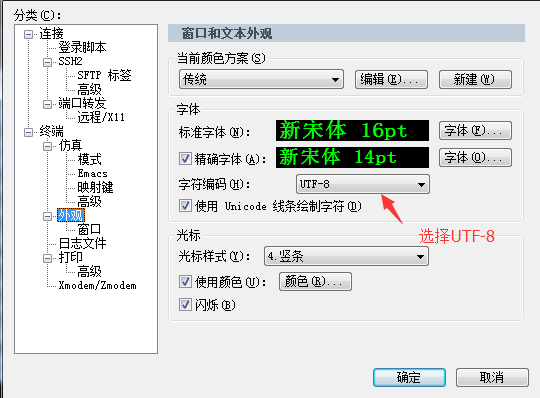
| character\_sets\_dir | /application/mysql-5.5.32/share/charsets/ |

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

8 rows in set (0.01 sec)

## 1.1 数据库中文不乱码：

### 1.1.1将CRT字符编码修改为UTF-8



### 1.1.2将Linux/etc/sysconfig/i18n字符集修改为中文模式：

sed -i 's#LANG="en\_US.UTF-8"#LANG="zh\_CN.UTF-8"#g' /etc/sysconfig/i18n

grep "LANG" /etc/sysconfig/i18n

LANG="zh\_CN.UTF-8"

在表中添加信息的命令：

insert into test values(1,'李芬');

临时改变MySQL字符集的命令：

set names gbk;

## 1.2 乱码情况：

1、环境本身不对，插入了错误数据后，很难解决。

2、环境对的时候插入了正确的数据，客户端环境破坏了，查看乱码了（易解决）。\*\*\*\*\*

数据库中文不乱码（推荐使用UTF-8）：

1、Linux客户端字符集。（UTF8）

2、Linux服务端字符集。（LANG="zh\_CN.UTF-8"）

3、数据库客户端字符集。（UTF8）

4、数据库服务端字符集。

5、具体数据库的字符集。

6、表的字符集。

7、PHP/Java程序字符集。

## 1.3 编辑客户端字符集：

1、使用 set names 指定字符集（临时生效）。

set names gbk 相当于下面三条命令：

SET character\_set\_client = gbk;

SET character\_set\_results = gbk;

SET character\_set\_connection = gbk;

查看更改过的字符集命令：

show variables like “character\_set%”

2、在登录MySQL的时候，可以指定字符集。

mysql –uroot –poldboy3306 –S /data/3306/mysql.sock –default-character-set=gbk

3、在[mysqld]下添加： （此法不适用，请忽略）

init\_connect = ‘SET NAMES gbk’

4、在/etc/my.cnf下添加：（多实例的MySQL客户端都在读取/etc/my.cnf）

[client]

default-character-set=gbk

## 1.4 编辑数据库 服务端字符集

1、MySQL5.1和5.5版本不同，命令也不同，（在/data/3306/my.cnf中添加。）

[mysqld] 对应的是database与server

default-character-set=gbk #适合5.1及以前版本。

character-set-server=gbk #适合5.5

MySQL数据库中建库建表

MySQL数据库中建库是和服务端字符集是一致的，服务端为utf8，那么建立的库就是utf8，建表则是根据库想对应的，如果库字符集为gbk，则在库中建立的表就是gbk。

指定字符集建库：

create database oldboy\_utf8 default character set utf8 collate utf8\_general\_ci;

指定字符集建表：

create table `student`(

`id` int(4) not null auto\_increment,

`name` char(20) not null,

primary key (`id`)

)ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8



## 1.5 企业案例

修改表的字符集：

alter table test character set = gbk collate gbk\_chinese\_ci;

修改库的字符集：

alter database oldboy character set = Latinl collate latinl\_swedish\_ci;

以上两种修改字符集的命令，只对新数据有用，对旧的数据可能会有问题，一般企业工作中，修改已有数据的字符集，使用如下步骤：

1、导出表结构（数据量大的话，先导出表结构，然后sed替换），在导出数据备份。

mysqldump –uroot –poldboy3306 –default-character-set =latinl –d dbname >alltable.sql #数据备份

2、在表结构中把字符集调整过来，例如：sed 替换。

3、修改系统配置，调整字符集生效。

4、mysql把表结构还原，再把数据还原。

小结：

======更改字符集总的思想========

1、数据库不要更新，导出所有数据。

2、把导出的数据进行字符集替换（替换表和库）。

3、修改my.cnf，更改mysql客户端服务端字符集，重启生效。

4、导入更改过字符集的数据，包括表结构语句，提供服务。

5、开发把程序更改为对应字符集，SSH客户端。

**数据库常用字符集：**

[root@db02 ~]# mysql -uroot -p'oldboy123' -S /data/3306/mysql.sock -e "SHOW  CHARACTER SET;" |egrep "gbk|utf8|latin1|utf8mb4"  
latin1  cp1252 West European    latin1\_swedish\_ci       1  
gbk     GBK Simplified Chinese  gbk\_chinese\_ci  2  
utf8    UTF-8 Unicode   utf8\_general\_ci 3  
utf8mb4 UTF-8 Unicode   utf8mb4\_general\_ci      4AUTO\_INCREMENT=9 DEFAULT CHARSET=latin1

要想数据库不乱码：

1、客户端字符集统一

法1：

提示：set names gbk也可以用下面三个命令替代。

法2：

SET character\_set\_client = gbk;

SET character\_set\_results = gbk;

SET character\_set\_connection = gbk;

法3：

mysql –u –p –S –default-character-set=gbk;

法4：

3、通过修改my.cnf实现修改mysql客户端的字符集，配置方法如下：

[client]

default-chaacter-set=gbk

2、服务端字符集

法1：

1、按如下要求更改my.cnf参数

[mysqld]

character-set-server=latin1

法2：

其他：编译的时候指定服务端字符集：

--default\_charset=utf8 \

--default\_collation=utf8\_general\_ci \

-dextra\_charsets=gbk,gb2312,utf8,ascii \

3、建库：

指定字符集建库：

create database oldboy\_utf8 default character set utf8 collate utf8\_general\_ci;

指定字符集建表：

create table `student` (

`id` int(4) not null auto\_increment,

`name` char(20) not null,

primary key (`id`)

) engine=innodb auto\_incr

4、程序

简体UTF8

http://download.comsenz.com/DiscuzX/3.2/Discuz\_X3.2\_SC\_UTF8.zip

5、

**在数据库中执行sql语句的方法：**

a. 习惯：尽量不在mysq命令行直接插入数据（SSH库互动影响），sql语句文件形式。

b. sql文件的格式要统一“utf8没有签名” 

c. 导入文件方式

1、可在MySQL命令行中用source执行sql文件。

2、命令方式导入数据mysql –u –p oldboy<test.sql

3、sql文件里set names utf8,或mysql –uroot –poldboy123 oldboy oldboy –default-character-set=utf8 <test.sql

**修改库和表的字符集的命令：**

**库：alter database oldboy CHARACTER SET**

**表：alter table test CHARACTER SET latin1;**

# 2、MySQL数据库服务日志

## 2.1 MySQL数据库服务常用日志文件知识

### 2.1.1 错误日志（error log）介绍与调整

#### 1）错误日志（error log）介绍

MySQL的错误日志（error log）记录MySQL服务进行mysqld在启动/关闭或运行过程中遇到的错误信息；

#### 2）错误日志（error log）实践

在配置文件中调整方法，当然可以在启动时加入启动参数。cat /data/3306/my.cnf

法1：

[mydqld\_safe]

log-error=/data/3306/mysql\_oldboy3306.err

法2：

启动MySQL命令里加入：

mysqld\_safe –defaults-file=/data/3306/my.cnf --log-error=/data/3306/mysql\_oldboy.err &

show variables like ‘log\_error%’

之前遇到数据库起不来案例：

1、把/data/3306/data 目录删除后，重新建立data目录并加属主数组。

2、把原有的my.cnf mysql这两个文件也重新还原过。

3、重新用mysqld\_safe启动过。

4、改过my.cnf里的server-id = 2

5、配置文件里的端口，路径。

排查思路

1、先把日志文件备份并清空启动一下mysql服务然后在查看日志文件报有什么错误

cat mysql\_xiao3306.err

2、mysqld3306目录下面所有文件都加上属主并是递归-R

chown –R mysql \*

3、查看一下有没有这个mysql id

id mysql

4、重新启动mysql服务

/data/3306/mysql start

5、此时再查看一下服务有没有启动起来

lsof –i :3306

### 2.1.2 普通查询日志（general query log）介绍与调整

#### 1）普通查询日志（general query log）介绍

普通查询日志（general query log）：记录客户端连接信息和执行的SQL语句信息；

#### 2）普通查询日志（general query log）调整

show variables like ‘general\_log%’

不重启数据库，让修改的数据库配置生效的命令：

set global general\_log = on;

临时生效命令：

set global general\_log\_file = “/data/3306/data/mysql\_oldboy.log”;

永久生效命令：

vi /data/3306/my.cnf #添加

[mysqld]

general\_log = on

general\_log\_file = /data/3306/data/MySQL\_oldboy.log

#####提示：高并发场景，企业里普通日志一般是关闭的（默认也是关闭），主要因为IO性能问题。

### 2.1.3 慢查询日志（slow query log）介绍与调整\*\*\*\*\*

#### 1）慢查询日志（slow query log）介绍

慢查询日志（slow query log）：记录执行时间超出指定值（long\_query\_time）的SQL语句；

#### 2）慢查询日志（slow query log）调整

此参数在/data/3306/my.cnf中的[mysqld]模块下

long\_query\_time = 1

log-slow-queries = /data/3306/slow.log

log\_queries\_not\_using\_indexes

####慢查询的设置，对于数据库SQL的优化非常重要。

#### 利用慢查询优化的一套解决方案：

1、开启慢查询

long\_query\_time = 1

log-slow-queries = /data/3306/slow.log

log\_queries\_not\_using\_indexes

2、慢查询日志切割

cat cut\_slow\_log.sh

cd /data/3306 &&\

/bin/mv slow.log slow.log.$(date +%F) &&\

mysqladmin -uroot -poldboy3306 -S /data/3306/mysql.sock flush-log

cat /var/spool/cron/root

####cut slow.log###

00 00 \* \* \* /bin/sh /server/scripts/cut\_slow\_log.sh &>/dev/null

3、使用工具mysqlsla分析慢查询，定时发给相关人员信箱

使用explain优化SQL语句（select语句）的基本流程：

1、抓慢查询SQL语句方法

a. show full processlist; （登录数据库现场抓，连续执行2次，超过2秒）

mysql -u -p -S -e "show full processlist;"|egrep -vi "sleep"

b. 分析慢查询日志

配置参数记录慢查询语句

long\_query\_time = 2

log\_queries\_not\_using\_indexes

log-slow-queries = /data/3306/slow.log

2、explain语句检查索引执行情况。

explain select \* from test where name=’oldboy’\G

explain select SQL\_NO\_CACHE \* from test where name=’oldboy’\G

3、对需要建立索引的条件列建立索引

大表不能高峰期建立索引，300万条记录。

4、分析慢查询的工具mysqlsla（每天早晨发邮件）

切割慢查询日志，去重分析后发给大家

mv /data/3306/slow.log /opt/$(date +%F)\_slow.log

mysql -u -p -S flush-logs

### 2.1.4 二进制日志（binary log）介绍与调整

#### 1）二进制日志（binary log）介绍

二进制日志（binary log）：记录数据被修改的相关信息，用户主从复制及增量恢复；

#### 2）二进制日志（binary log）调整

show variables like ‘%log\_bin%’

mysql> show variables like ?%log\_bin%';

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

| Variable\_name | Value |

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

| log\_bin | ON | #要不要打开二进制开关，记录binlog

| log\_bin\_trust\_function\_creators | OFF |

| sql\_log\_bin | ON | #临时不记录binlog（增量恢复）

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

3 rows in set (0.00 sec)

[root@db02 scripts]# grep log-bin /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

临时不记录binlog（增量恢复）

set session sql\_log\_bin =OFF;

create database abcd;

mysqlbinlog mysql-bin.000032|grep abcd

#### 二进制日志log-bin 作用：

1、记录更改的二进制形式SQL语句。

2、主从复制。

3、增量数据备份。

# 3、MySQL增量备份与恢复

## 3.1 MySQL增量备份与恢复

### 3.1.1 MySQL binlog三种模式及设置方法

#### 1）row level（行模式）

日志中会记录成每一行数据被修改的情况，然后在slave端再对相同的数据进行修改。

优点：

在row level模式下，bin-log中可以不记录执行的SQL语句的上下文相关的信息，仅仅只需要记录那一条记录被修改了，修改成什么样子了，所以row level的日志内容会非常清楚的记录下每一行数据修改的细节，非常容易理解。而且不会出现某些特定情况下的存储过程或function，以及trigger的调用和触发无法被正确复制的问题。

缺点：

row level下，所有的执行的语句当记录到日志中的时候，都将以每行记录的修改来记录，这样可能会产生大量的日志内容，比如这样一条update语句：update product set owner\_member\_id=’b’ where owner\_member\_id=’a’，执行之后，日志中记录的不是这条update语句所对应的时间（MySQL以时间的形式来记录bin-log日志），而是这条语句所更新的每一条记录的变化情况，这样就记录成很多条记录被更新的很多个事件。自然，bin-log日志的量就会很大。尤其是当执行alter table之类的语句的时候，产生的日志量是惊人的。因为MySQL对于alter table之类的表结构变更语句的处理方式是整个表的每一条记录都需要变动，实际上就是重建了整个表。那么该表的每一条记录都会被记录到日志中。

#### 2）statement level（语句级别，MySQL默认的级别）

每一条被修改数据的SQL都会记录到master的bin-log中。slave在复制的时候SQL进程会解析成和原来master端执行过的相同的SQL来再次执行。

优点：

statement level下的优点首先就是解决了row level下的缺点，不需要记录每一行数据的变化，减少bin-log日志量，节约IO，提高性能。因为他只需要记录在master上所执行的语句的细节，以及执行语句时候的上下文的信息。

缺点：

由于它是记录的执行语句，所以，为了让这些语句在slave端也能正确执行，那么他还必须记录每条语句在执行的时候的一些相关信息，也就是上下文信息，以保证所有语句在slave端被执行的时候能够得到和在master端执行时候相同的结果。另外就是，由于master现在发展比较快，很多的新功能不断的加入，使MySQL的复制遇到了不小的挑战，自然复制的时候涉及到越复杂的内容，bug也就越容易出现。在statement level下，目前已经发现的就有不少情况会造成MySQL的复制出现问题，主要是修改数据的时候使用了某些特定的函数或者功能的时候回出现，比如：sleep（）函数在有些版本中就不能真确复制，在存储过程中使用了last\_insert\_id()函数，可能会使slave和master上得到不一致的id等等。由于row level是基于每一行来记录的变化，所以不会出现类似的问题。

#### 3） mixed（混合模式）

实际上就是前两种模式的结合。在mixed模式下，MySQL会根据执行的每一条具体的SQL语句来区分对待记录的日志形式，也就是在statement和row之间选择一种。新版本中的statement level还是和以前一样，仅仅记录执行的语句。而新版本的MySQL中对row level模式也被做了优化，并不是所有的修改都会以row level来记录，像遇到表结构变更的时候就会以statement模式来记录，如果SQL语句确实就是update活着delete等修改数据的语句，那么还是会记录所有行的变更。

###一般工作中，默认就够了，如果有特殊功能需求的话，选择mixed。

**企业场景如何选择binlog的模式**

1、互联网公司，使用MySQL的功能相对少（存储过程、接发器、函数）。选择默认的语句模式，statement level（默认）

2、公司如果用到使用MySQL的特殊功能（存储过程、触发器、函数）。则选择mixed模式。

3、公司如果用到使用MySQL的特殊功能（存储过程、触发器、函数），又希望数据最大化一致，此时最好ROW level模式。

### 3.3.2 设置MySQL binlog的格式

在配置文件中参数如下：

log-bin=mysql-bin

#binlog\_format=”STATEMENT”

#binlog\_format=”ROW”

binlog\_format=”MINED”

运行时在线修改：

SET SESSION binlog\_format = ‘STATEMENT‘；

SET SESSION binlog\_format = ‘ROW‘；

SET SESSION binlog\_format = ‘MIXED‘；

SET GLOBAL binlog\_format = ‘STATEMENT‘；

SET GLOBAL binlog\_format = ‘ROW‘；

SET GLOBAL binlog\_format = ‘MIXED‘；

#### 例子：

1、设置binlog模式为ROW

mysql> SHOW variables like "binlog\_format%";

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

| Variable\_name | Value |

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

| binlog\_format | STATEMENT |

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

1 row in set (0.00 sec)

mysql> SET GLOBAL binlog\_format = 'ROW';

Query OK, 0 rows affected (0.00 sec)

在配置文件中参数如下：

log-bin=mysql-bin

#binlog\_format=”ROW”

2、重启数据库查看修改效果

mysql> SHOW global variables like "binlog\_format%";

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

| Variable\_name | Value |

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

| binlog\_format | ROW |

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

1 row in set (0.00 sec)

3、批量插入及更新数据

update test set name =”test”;

4、检验ROW 模式下binlog日志记录效果

mysqlbinlog –base64-output=”decode-rows” --verbose mysql-bin.000001

# 4、MySQL数据库的备份与恢复命令基础

## 4.1 MySQL数据库的备份与恢复

### 4.1.1 备份数据的意义

运维工作简单的概括就两件事：

1、保护公司的数据。

2、网站7\*24小时提供服务。

那么，对数据丢失一部分和网站7\*24提供服务哪个更重要呢？这个具体要看业务和公司。

例如：银行，金融行业，数据时最重要的，一条都不能丢，可能宕机停机影响没那么大。

再例如：百度搜索，腾讯QQ聊天记录丢失了几万条数据，都不算啥。

对于数据来讲，数据最核心的就是数据库数据，当然也包括其他数据。对于绝大多数企业来讲，失去数据就失去商机，失去产品，失去客户，公司倒闭……，因此，数据库的备份与恢复就显得十分重要了。

### 4.1.2 备份单个数据库练习多种参数使用

mysql数据库自带了一个很好用的备份命令，就是mysqldump，它的基本使用如下：

语法：mysqldump –u –p 数据库名> 备份的文件名

**范例1：备份名字为oldboy的库**

a.查看备份前的数据

[root@db01 opt]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "use oldboy;show tables;select \* from student;"

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

| Tables\_in\_oldboy |

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

| course |

| sc |

| student |

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

b.执行备份的命令

[root@db01 opt]# mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock >/opt/mysql\_bak.sql

c.检查备份的结果

egrep -v "#|\\*|^$" oldboy.2sql

##这里的乱码，是因为导出时的格式没加字符集，一般恢复到数据库里会正常，只是系统外查看不正常而已。另外insert是批量插入的方式，这样在恢复时效率很高。

根据查看的结果，看到了已备份的表结构语句及插入的数据整合的SQL语句，但是中文数据乱码了。

**范例2：设置字符集参数备份解决乱码问题**

a.查看备份前数据库客户端及服务端的字符集设置

mysql –uroot –poldboy3306 –e “show variables” like ‘%character%’”

b.指定对应的字符集备份，这里为--default-character=latinl

mysqldump –uroot –poldboy3306 –default-character-set=latinl oldboy>/opt/mysql\_bak.sql

执行结果：

mysqldump -uroot -poldboy3306 --default-character-set=latinl -S /data/3306/mysql.sock oldboy > oldboy.2sql

egrep -v "#|\\*|^$" oldboy.2sql

#在查看中文不乱码了，其实，即使导出的中文乱码也没关系，导入到数据库里依然是正常的。

利用上述备份好的文件进行数据恢复测试：

mysql –uroot –poldboy3306 –S /data/3306/mysql.sock oldboy < oldboy.2sql

mysql –uroot –poldboy3306 –e “select \* from oldboy.student;”

**范例3：备份时加-B参数**

mysqldump -uroot -poldboy3306 --default-character-set=latinl –B -S /data/3306/mysql.sock oldboy > oldboy.2sql

和前面的备份文件对比，看看-B参数的作用

只管看，加-B参数的作用是增加创建数据库和连接数据的语句了，即如下两条语句：

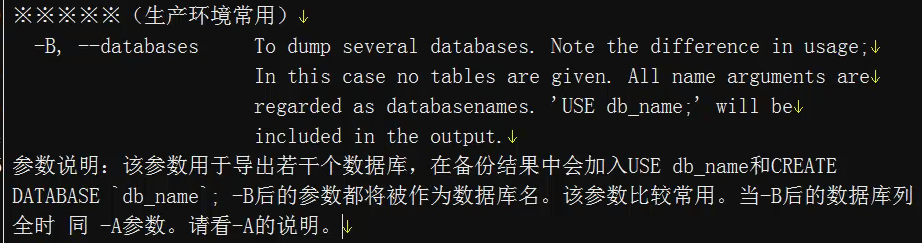
CRTEATE DATABASE /\*!32312 IF not exists\*/`oldboy` /\*!40100 default character set latinl \*/;

USE `oldboy`;

利用上述加-B的备份进行恢复测试：

mysql –uroot –poldboy3306 –S /data/3306/mysql.sock < oldboy.2sql #注意备份命令的不同

mysql –uroot –poldboy3306 –e “select \* from oldboy.student;”



**范例4：优化备份文件大小减少输出注释（debug调试）**

利用mysqldump的--compact参数优化下备份结构。

参数说明：

测试时用的比较多，也可以后话输出内容的大小，让容量更少，适合调试，该选项使得输出内容更简洁，不包括默认选项中各种注释。有如下几个参数的功能：

--skip-add-drop-table –no-set-names –skip-disable-keys –skip-add-locks



**范例5：指定压缩命令压缩备份的MySQL数据**

mysqldump –uroot –poldboy3306 –default-character-set=latinl –B oldboy|gzip >/opt/mysql\_bak\_b.sql.gz

#压缩效率将近3倍

**通过以上例子可以得出一个小结：**

1、备份数据使用-B参数，会在备份数据中增加建库及use库的语句。

2、备份数据使用-B参数，使得后面可以直接接多个库名。

3、备份数据使用gzip对备份的数据压缩。

4、debug时可以用--compact减少输出，但不用于生产。

5、指定字符集备份用—default-character-set=latinl（一般不用）。

### 4.1.3 mysqldump的工作原理

利用mysqldump命令备份数据的过程，实际上就是把数据从mysql库里逻辑的SQL语句的形式直接输出或生成备份的文件过程。可否举一反三，利用这个备份文件恢复的时候的原理？

备份的数据过滤掉注释，都是SQL语句，结果如下：

egrep -v "#|\\*|--|^$" oldboy.2sql

提示：使用mysqldump是把数据库的数据导出通过SQL语句的形式存储，这种备份方式称之为逻辑备份，效率不是很高，一般在50G以内的数据使用。其他备份方式，物理备份：cp,tar（停库），xtrabackup物理备份。

### 4.1.4 备份多个库及多个参数练习

a.操作结果：

mysqldump –uroot –poldboy3306 –B oldboy\_gbk oldboy|gzip >/opt/mysql\_bak\_b.sql.gz

b.-B参数说明

提示：-B参数是关键，标识接多个库，并且增加use db，和create database的信息。（生产环境中常用）



### 4.1.5 如何做分库备份？

分库备份实际上就是执行一个备份语句备份一个库，如果数据库里有多个库，就执行多条相同的备份单个库的备份语句就可以备份多个库了，注意：每个库都可以用对应备份的库作为库名，结尾加.sql，备份多个库的命令如下：

mysqldump –u –p –B oldboy

mysqldump –u –p –B oldboy\_gbk

#### 分库备份：

法1：

mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "show databases;"|egrep -vi "database|infor|perfor"|sed -r 's#^([a-z].\*$)#mysqldump -uroot -poldboy3306 -events -B \1|gzip >/opt/bak/\1.sql.gz#g' |bash

将上述命令放入一个脚本里就是脚本分库备份了，当然这样是很土的备份脚本了，更好的备份脚本：

mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "show databases;"|egrep -v "schema|atabase"|sed -r 's#^(.\*)#mysqldump -uroot -poldboy3306 -events -B \1|gizp >/opt/\1.sql.gz#g'|bash

分库备份的意义何在？

有时一个企业的数据库里会有多个库，例如（www,bbs,blog），但出问题时的很可能是某一个库，如果在备份时把所有的库都备份成一个数据文件的话，恢复某一个库的数据时就比较麻烦了。

### 4.1.6 备份单个表

语法：mysqldump –u –p –S 数据库名 表名 >备份的文件名

mysqldump –u –p –S oldboy student>/opt/tablesl.sql

提示：不能加-B参数，因为库oldboy后面就是oldboy表了。

### 4.1.7 备份多个表

语法：mysqldump –u –p –S 数据库名 表名1 表名2 >备份的文件名

企业需求：一个库里有大表有小表，有时可能需要只恢复某一个小表，上述的多表备份文件很难拆看，就会像没有分库那样导致恢复某一个小表很麻烦。

那么，又如何进行分表备份呢？如下，和分库的思想一样，每执行一条语句备份一个表，生成不同的数据文件即可。如下：

mysqldump –u –p oldboy test >oldboy\_test.sql

mysqldump –u –p oldboy test1 >oldboy\_test1.sql

将上述命令放入一个脚本里就是脚本分表备份了。

**分表备份缺点：文件多，很碎。**

1、备份一个完整全备份，再做一个分库表备份。

2、脚本批量恢复多个SQL文件。

见分库分表备份视频 <http://edu.51cto.com/course/course>\_id-808.html

#### 面试题：多个库或多个表备份到一块了，如何恢复单个库或表？

解答：

a. 第三方测试库，导入到库里，然后把需要的备份出来，最后恢复到正式库里。

b. 单表：grep 表名 bak.sql >表名.sql

单库：循环过滤库里所有表，grep 表名 bak.sql >表名.sql （多个表数据）

c. 事先分库表备份。

### 4.1.8 备份数据库表结构（不包含数据）

利用mysqldump –d 参数值备份表的结构，里：备份oldboy库的所有表的结构

mysqldump –u –p –S –B –d oldboy >/opt/t.sql

egrep –v “#|\\*|--|^$” /opt/t.sql

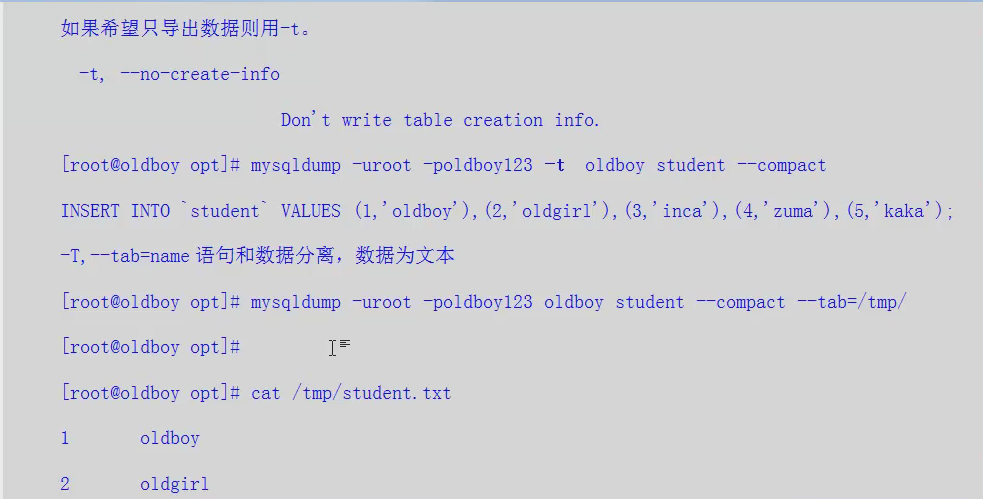
**直接备份数据 –t 参数**

mysqldump –u –p –compact –S –t oldboy student –compact

**直接把数据和表分离：-tab**

mysqldump –u –p –compact –S oldboy test –tab=/tmp/

#缺点：-T 表中的内容不再是SQL语句了，而是文本。



##### 上述例子小结：mysqldump的参数说明：

1、-B备份多个库（并添加create和use库的语句）。

2、-d 值备份库的表结构。

3、-t 只备份数据。

4、-T 分离库表和数据成不同的文件，数据是文本，非SQL语句。

### 4.1.9 刷新binlog参数

binlog是什么，记录数据库更新的SQL语句，二进制文件。

mysqldump用于定时对某一时刻的数据的全备，例如：00点进行备份bak.sql.gz。

增量备份：当有数据写入到数据库时，还会同时把更新的SQL语句写入到对应的文件里这个文件就叫做binlog文件。

10点丢失数据需要恢复数据：

1、00点时刻备份的bak.sql.gz数据还原到数据库，这个时候数据恢复到了00点。

2、00:00-10:00数据，就要从binlog里恢复。

binlog文件生效需要一个参数：log-bin

[root@db02 ~]# grep log-bin /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

[root@db02 ~]# ll /data/3306/mysql-bin.\*

-rw-rw---- 1 mysql mysql 1130 12月 16 14:51 /data/3306/mysql-bin.000001

-rw-rw---- 1 mysql mysql 126 12月 16 15:05 /data/3306/mysql-bin.000002

-rw-rw---- 1 mysql mysql 126 12月 16 15:10 /data/3306/mysql-bin.000003

-rw-rw---- 1 mysql mysql 481 12月 16 15:12 /data/3306/mysql-bin.000004

-rw-rw---- 1 mysql mysql 107 12月 16 15:13 /data/3306/mysql-bin.000005

-rw-rw---- 1 mysql mysql 126 12月 17 09:59 /data/3306/mysql-bin.000006

binlog日志切割：确定全备和增量的临界点

**-F 刷新binlog日志**，生成新文件，将来增量恢复从这个文件开始。--master-data在备份语句里添加change master语句及binlog文件及位置点信息

值为1，为可执行的change master语句，（备份文件做从库）

值为2，注释的—change master语句 --master-date除了增量恢复确定临界点外，做主从复制时作用更大。

mysqldump –u –p –S –master-data=1 --compact oldboy change master to master\_log\_file=’msyql-bin.000017’,master\_log\_pos=107;

### 4.1.10 mysqldump的关键参数说明

关键参数：mysqldump --help

1、-B指定多个库，增加建库语句和use语句。\*\*\*\*\*

2、--compact去掉注释，适合调试输出，生产不适用。

3、-A 备份所有库。

4、-F 刷新binlog日志，生成新文件，将来增量恢复从这个文件开始。

5、--master-data增加binlog日志文件名及对应的位置点（即change master语句）。

6、-x –lock-all-tables 锁表，严格的保证数据一致。

7、-l，--lock-tables lock all tables for read 只读的锁表。

8、-d 只备份库表结构，无数据。

9、-t 值备份数据，无库表结构。

10、-T 库表和数据分离不同文件，数据是文本形式。

11、--single-transaction 适合innodb事务数据库备份。

innoDB表在备份时，通常启用选项 –single-transaction来保证备份的一致性，实际的工作原理是设定本次会话的隔离级别为：repeatable read，以确保本次会话（dump）时，不会看到其他会话已经提交了的数据。###事务有隔离性。

12、-q，--quick 快速导出。

### 5.1.11 生产场景不同引擎mysqldump备份命令

myisam 引擎企业生产备份命令（适合所有引擎或混合引擎）：

mysqldump -uroot -poldboy123 -A -B -F -R --master-data=2 -x --events|gzip >/opt/all.sql.gz

提示：-F也可以不用，与--master-data有些重复。

innoDB引擎企业生产备份命令：推荐使用的

mysqldump -uroot -poldboy123 -A -B -F -R --master-data=2 --events --single-transaction |gzip >/opt/all.sql.gz

提示：-F也可以不用。与--master-data有些重复。

--master-data作用：

1、使用--master-data=2进行备份文件会增加如下内容：适合普通备份增量恢复。

--change master to master\_log\_file=’mysql-bin.000020’，master\_log\_pos=1191;

2、使用--master-data=1 进行备份文件会增加如下内容：更适合主从复制

change master to master\_log\_file=’mysql-bin.000020’，master\_log\_pos=1191;

额外补充：

一、mysqldump逻辑备份说明：

缺点：效率不是特别高；

优点：简单、方便、可靠、迁移。

二、超过50G可选方案：

1、适用与数据量不是特别大的场景，50G以内的数据。

2、xtrabackup物理备份工具：全备和增量。

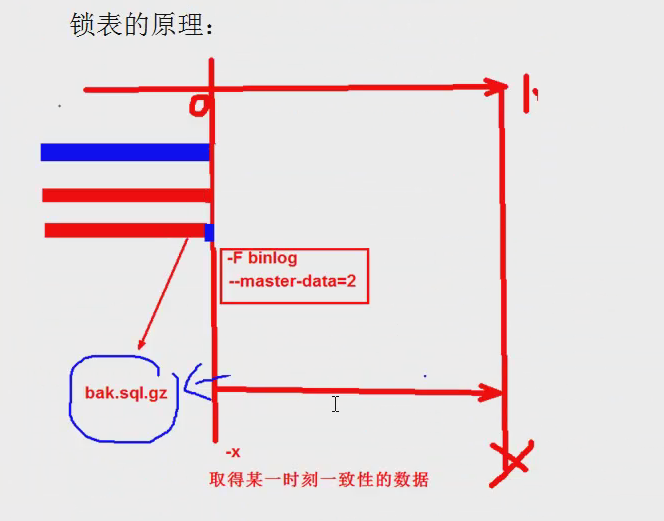
3、物理备份方案：从库停止SQL线程，打包，cp。

三、何时会使用备份的数据？nimalegebi

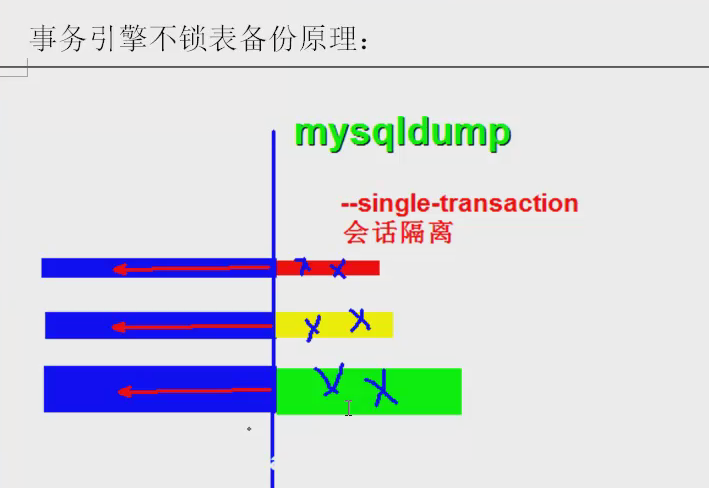
1、恢复数据到测试库的时候。

2、人为通过SQL语句将数据删除的时候。

3、做主从复制的时候。







# 6、恢复数据库实践

## 6.1数据库恢复事项

提示：

1、数据恢复和字符集关联很大，如果字符集不正确会导致恢复的数据乱码

2、mysql命令以及source命令恢复数据库的原理就是把文件的SQL语句，在数据行的过程。

利用source命令恢复数据

mysql> source /opt/oldboy\_B\_2015-12-16.sql

提示：

1、source数据恢复和字符集关联很大，如果字符集不正确会导致恢复的数据乱码。

2、UTF8数据库，那么恢复的文件格式需要为“UTF8不带签名”

针对压缩的备份方法

法1：

gzip –d /opt/mysql\_bak.sql.gz

mysql –u –p –S </opt/mysql\_bak.sql.gz

不删除备份文件：

gzip –cd 01.sql.gz >02.sql

法2：

gunzip < b\_bak.sql.gz >/opt/mysql\_bak.sql

mysql –u –p –S </opt/mysql\_bak.sql

## 6.2 实现和MySQL非交互式对话

利用MySQL –e 参数查看mysql数据

[root@db01 ~]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show databases;"

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

| Database |

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

| information\_schema |

| lidongkai\_utf8 |

| mysql |

| oldboy |

| oldboy\_gbk |

| performance\_schema |

| test |

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

企业案例：mysql sleep线程过多的问题案例：

解决方法：

1、配置文件里修改：

[mysqld]  
interactive\_timeout = 120 ç此参数设置后wait\_timeout自动生效。  
wait\_timeout = 120

2、其他方法：

1、PHP程序中，不适用持久链接，即使用mysql\_sonnect 而不是connect（Java调整连接池）

2、PHP程序执行完毕，应该显示调用mysql\_close。

3、逐步分析MySQL的SQL查询及慢查询日志，找到查询过慢的SQL优化之。

查看MySQL配置文件配置有没有在数据库中生效：

mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%server\_id%';"

mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like '%log\_bin%';"

## 6.3 不重启数据库修改数据库参数

不重启数据库修改数据库参数，但是要求重启后还能生效

mysql> show variables like '%key\_buffer%';

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

| Variable\_name | Value |

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

| key\_buffer\_size | 16777216 |

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

1 row in set (0.00 sec)

mysql> set global key\_buffer\_size = 1024\*1024\*32;

Query OK, 0 rows affected (0.25 sec)

mysql> show variables like '%key\_buffer%';

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

| Variable\_name | Value |

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

| key\_buffer\_size | 33554432 |

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

1 row in set (0.04 sec)

myisa引擎索引的缓冲区  
即时生效，重启失效：

set global key\_buffer\_size = 1024\*1024\*32;

配置文件：

/etc/my.cnf,key\_buffer\_size = 32M

不重启数据库更改数据库参数小结：

1、set global key\_buffer\_size = 1024\*1024\*32; #及时生效，重启MySQL失效。

2、配置文件也要改，编辑/etc/my.cnf，修改key\_buffer\_size = 32M

## 6.4 生产场景常用重要命令小结：

1、show processlist; #查看数据库里正在执行的SQL语句，可能无法看全完整的SQL语句。

2、show full processlist; #查看正在执行的完整SQL语句，完整显示。

3、set global key\_buffer\_size = 1024\*1024\*32; #不重启数据库调整数据库参数，直接生效，重启后失效。

4、show variables; #查看数据库的配置参数信息，例如：my.cnf里参数的生效情况。

5、kill id 杀掉线程的命令，id为线程号（mysql下）

6. show status 查看当前会话的数据库状态信息。

7. show global status 查看整个数据库的运行状态信息。很重要，要分心并要做好监控。

8. show engine innodb status 显示innodb疫情的性能状态（高级dba才会关注）

[root@db01 ~]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show global status;"|egrep "select"

Com\_insert\_select 0

Com\_replace\_select 0

Com\_select 279

计算一天之内：MySQL有多少个insert,delete语句，有没有好方法？

1、定时任务每天0点，show global status;按天取出对比。

2、按天分析binlog日志，获取数据库不同语句的频率。

MySQL命令常用参数：

-u 用户

-p 密码

-S sock

-h 远程

-P 端口

-e 在命令行执行

6.5 mysqladmin的相关命令：

**mysqladmin的相关命令**mysqladmin password oldboy123 设置密码  
mysqladmin -uroot -poldboy123 password oldboy 修改密码  
mysqladmin -uroot -poldboy123 status 查看状态  
mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock -i 1 status 一秒显示  
mysqladmin -uroot -poldboy123 flush-logs 刷新日志  
mysqladmin -uroot -poldboy123 processlist   
mysqladmin -uroot -poldboy123 extended-status #show blobal status; 查看扩展的状态  
mysqladmin -uroot -poldboy123 processlist -i 1 ç实时跟踪。  
watch mysqladmin -uroot -poldboy123 processlist  
mysqladmin -uroot -p'oldboy' -S /data/3306/mysql.sock shutdown  
mysqladmin -uroot -p'oldboy' -S /data/3306/mysql.sock variables

watch mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock processlist

## 6.6 MySQL工具mysqlbinlog

mysqlbinlog工具的作用是解析MySQL的二进制binlog日志内容，把二进制日志解析成可以在MySQL数据库里执行的SQL语句。

### 6.6.1 MySQL的binlog日志是什么？

MySQL数据目录下的如下文件就是mysql的binlog日志

mysql-bin.000001

mysql-bin.000002

mysql-bin.000003

mysql-bin.000004

提示：必须要有打开log-bin功能，才能生成上述文件。

### 6.6.2 mysql的binlog日志作用是什么？

mysql的binlog日志用来记录mysql内部**增删改**等对mysql数据库有更新的内容的记录（对数据库的改动），对数据库查询的语句如show，select开头的语句，不会被binlog日志记录，主要作用是用于数据库的增量恢复，以及主从复制。

### 6.6.3 mysqlbinlog工具解析binlog日志实践

默认情况binlog日志是二进制格式的，不能使用查看文本工具的命令查看，例如：cat、vi。

[root@db01 ~]# file /data/3306/mysql-bin.000003

/data/3306/mysql-bin.000003: MySQL replication log

1、解析指定库的binlog日志

范例：利用mysqlbinlog –d 参数解析指定库的binlog日志

[root@db01 scripts]# mysqlbinlog -d oldboy mysql-bin.000004 -r lili.sql

[root@db01 scripts]# ll lili.sql

-rw-r--r-- 1 root root 701 12月 29 22:24 lili.sql

结论：mysqlbinlog工具分库导出binlog，如果使用-d参数，那更新数据时，必须有use database，才能分出指定库的binlog，例如：

use oldboy;

insert into student values(1,’oldboy’)

下面的写法就不行

insert into oldboy.student values(2,’oldgirl’)

1） 按照位置截取：精确，时间长。

mysqlbinlog mysqlbin.000020 --start-position=365 --stop-position=456 –r pos.sql

指定开始位置，不指定结束位置，请问结束位置是？ 答：直接到文件结尾。

指定结束位置，不指定开始位置，请问开始位置是？ 答：从文件的开头到结束的位置。

2） 按照时间截取：模糊，不准，会丢数据。

mysqlbinlog mysql-bin.000020 --start-datetime=’2014-10-16 17:14:15’ --stop-datetime=’2014-10-16 17:15:15’ –r time.sql

指定开始时间，不指定结束时间，请问结束时间是？ 答：直接到文件结尾。

指定结束时间，不指定开始时间，请问结束时间是？ 答：从文件的开头到结束的位置。

### 6.6.4 mysqlbinlog命令

###### mysqlbinlog命令：

1、把binlog解析为SQL语句（包含位置和时间点）。

2、-d参数根据指定库拆分binlog（拆分单表binlog可通过SQL关键字过滤）。

3、通过位置参数截取部分binlog：位置参数：--start-position =365 –stop-position=456，精确定位取部分内容。

4、通过时间参数截取部分binlog：时间参数：--start-datetime=’2014-10-16 17:14:15’ –stop-datetime=’2014-10-16 17:15:15’，模糊取部分内容，会丢数据。

5、-r文件名，相当于重定向 “>文件名”

6、解析ROW级别binlog日志的方法：

mysqlbinlog –base64-output=decode-rows-v mysql-bin.000016

mysqlbinlog –base64-output=”decode-rows” –verbose mysql-bin.000004

# 7、MySQL生产备份实战应用指南

## 7.1 全量备份与增量备份

### 7.1.1 全量备份

全量数据就是数据库中所有的数据，全量备份就是把数据库中的所有数据进行备份。

备份所有库：

[root@db01 backup]# mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -F -B -A --events|gzip > /server/backup/mysql\_bak\_$(date +%F).sql.gz

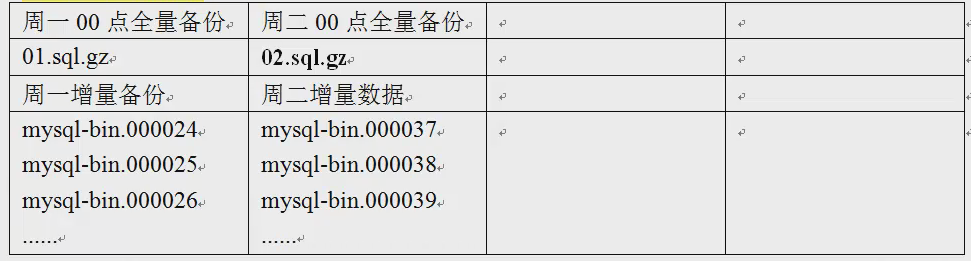
备份一个库：

[root@db01 backup]# mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -F -B oldboy|gzip > /server/backup/oldboy\_bak\_$(date +%F).sql.gz

### 7.1.2 增量备份

增量数据是从上次全量备份之后，更新的新数据，对于MySQL来说，binlog日志就是MySQL的增量数据。

1、按天全备情况





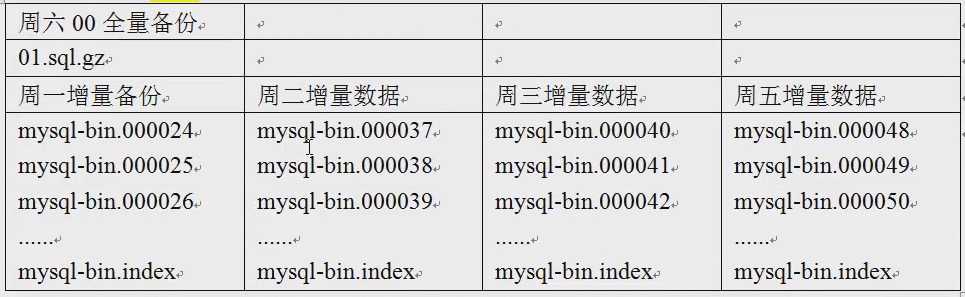
优点：

恢复时间短、维护成本低。

缺点：

占用空间多、占用系统资源多、经常锁表影响用户体验。

2、按周全备情况



优点：

占用空间小，占用系统资源少，无需锁表，用户体验好一些。

缺点：

维护成本高，恢复麻烦，时间长。

企业场景全量和增量的频率是怎么做的呢？

1）中小公司，全量一般是每天一次，业务流量低谷执行全备，备份时会锁表。

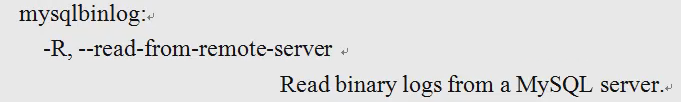
增量备：

a. 定时推binlog增量，例如每分钟推一次增量。

例：

rsync -avz /data/3306/mysql-bin.000\* rsync\_backup@10.0.0.18::backup --password-file=/etc/rsync.password

b. 再在其他远程实时读binlog。



2） 大公司周备，每周六00点一次全量，周日-下周六00点前都是增量。

优点：

节省备份时间，减小备份压力。

缺点：

增量的binlog文件副本太多，还原会很麻烦。

3） 一主多从环境，主从复制本身就是实时远程备份，可以解决服务器物理故障。

4） 一主多从环境，可采取一个从库服务器上专门用mysqldump，cp，tar，xtrabackup做备份，延迟同步。

mysql的mysqldump备份什么时候派上用场？

1、迁移或者升级数据库时。

2、增加从库的时候。

3、因为硬件或特殊异常情况，主库或从库宕机，主从可以互相切换，无需备份。

4、人为的DDL，DML语句，主从库没办法了，所有库都会执行。此时需要备份。

5、跨机房灾备，需要备份到异地。

什么情况下需要增量恢复？

生产工作中一般常用是一主多从的数据库架构，常见的备份方案是在某一个不对外的务的从库上开启binlog，然后实施定时全备份和实时增量备份。  
什么是增量备份？  
利用二进制日志和全备进行的恢复过程，被称为增量恢复。  
  
什么情况下才需要数据库增量恢复？  
1） 主或者从库宕机（硬件损坏），是否需要增量恢复呢？  
答：不需要增量恢复，主机宕机，只需要把其中一个同步最快的从库切换为主库即可。  
   主机宕机，只要选择更更新最快的从库（master.info，或5.5半同步机制），提升为主库。  
   从库宕机，直接不用就好了（一般为LVS负载均衡）。或者正常恢复  
2） 人为操作数据库SQL语句破坏主库是否需要增量恢复？  
在数据库主库内部命令行误操作，会导致所有的数据库（包括主从库）数据丢失，例如：在主库执行了drop database oldboy ；这样的删除语句，这时候的从库也会执行这个drop database oldboy语句。从而导致所有数据库上的Oldboy数据库的数据丢失，这样的场景是需要增量恢复的。  
3） 只有一个主库是否需要增量恢复？  
如果公司里只有一个主库的情况，首先应该多定时全量备份（一天每天一次）及增量备份（每隔1-10分钟对binlog日志做切割后备份到其他的服务器上，或者本地的其他硬盘里）或者写到网路文件系统（备份服务器）里。  
如果不允许数据丢失，最好的办法是做从库，通过DRBD（基于磁盘块的）同步。  
正常情况：  
主从同步：除了分担读写压力外，还可以防止物理设备损坏数据丢失的恢复。  
从库备份：在从库进行全量和增量方式的备份，可以防止人为对主库的误操作导致数据丢失。确保备份的从库实时和主库是同步状态。  
  
小结：

一般由人为误操作（或者程序）逻辑的方式在数据库执行SQL语句等误操作时，才需要增量恢复，因为此时，所有从库也都执行了误操作。

MySQL增量恢复必备条件

开启MySQL log-bin日志功能

MySQL数据库开启了log-bin参数记录binlog日志功能如下：

grep log-bin /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

提示：主库和备份的从库都要开启binlog记录功能。

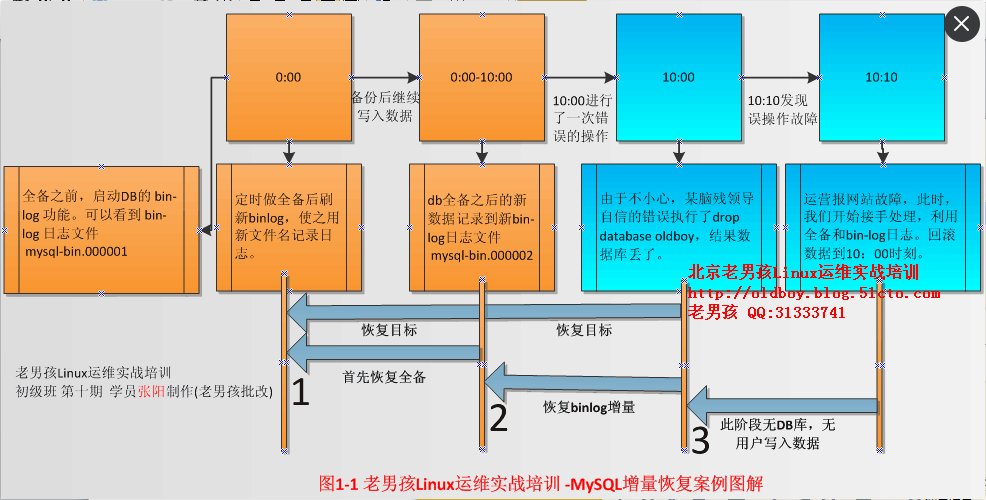
小结：

增量恢复的条件：

存在一份全备加上全备之后的时刻的所有增量binlog文件备份。

存在MySQL数据库全备

生产环境mysqldump备份命令



#### 数据库使用drop误删除后，如何恢复？

**1、在零点进行全量备份：**

date -s '2015/12/22 00:00:00'

mysqldump -uroot -poldboy3306 -S /data/3306/mysql.sock -B -F -R -x --master-data=2 oldboy|gzip >/server/backup/oldboy\_$(date +%F).sql.gz

**2、备份后，在凌晨3点时在test表中添加数据**

mysql> use oldboy;

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 test;

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

| id | name |

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

| 3 | gongli |

| 5 | kaka |

| 2 | oldgirl |

| 4 | zuma |

| 1 | 李芬 |

| 6 | 老男孩 |

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

6 rows in set (0.00 sec)

mysql> insert into test values(7,'lifen');

Query OK, 1 row affected (0.00 sec)

mysql> select \* from test;

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

| id | name |

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

| 3 | gongli |

| 5 | kaka |

| 7 | lifen |

| 2 | oldgirl |

| 4 | zuma |

| 1 | 李芬 |

| 6 | 老男孩 |

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

7 rows in set (0.00 sec)

mysql> insert into test values(8,'oldboyedu.com');

Query OK, 1 row affected (0.00 sec)

mysql> select \* from test;

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

| id | name |

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

| 3 | gongli |

| 5 | kaka |

| 7 | lifen |

| 8 | oldboyedu.com |

| 2 | oldgirl |

| 4 | zuma |

| 1 | 李芬 |

| 6 | 老男孩 |

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

8 rows in set (0.00 sec)

mysql> insert into test values(9,'hehe');

Query OK, 1 row affected (0.00 sec

**3、在早晨10的时候，误操作将库删除掉了**

mysql> drop database oldboy;

Query OK, 4 rows affected (0.02 sec)

mysql> show databases;

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

| Database |

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

| information\_schema |

| bbs |

| bingbing |

| gongli |

| mysql |

| oldboy11 |

| performance\_schema |

| xiaowan |

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

8 rows in set (0.00 sec)

**4、查找日志的分割位置，找到该日志后将删除命令删除**

[root@db02 backup]# grep CHANGE oldboy\_2015-12-22.sql

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000015', MASTER\_LOG\_POS=107;

[root@db02 backup]# cp /data/3306/mysql-bin.000015 .

[root@db02 backup]# mysqlbinlog -d oldboy mysql-bin.000015 >3.sql #解析binlog日志并重定向

[root@db02 backup]# ll

总用量 16

-rw-r--r-- 1 root root 2560 12月 23 10:08 3.sql

-rw-r----- 1 root root 787 12月 23 10:06 mysql-bin.000015

-rw-r--r-- 1 root root 5091 12月 22 00:00 oldboy\_2015-12-22.sql

[root@db02 backup]# vim 3.sql #删除drop命令

**5、首先进行全量恢复，然后进行增量恢复**

[root@db02 backup]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock < oldboy\_2015-12-22.sql

[root@db02 backup]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "select \* from oldboy.test;"

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

| id | name |

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

| 3 | gongli |

| 5 | kaka |

| 2 | oldgirl |

| 4 | zuma |

| 1 | 李芬 |

| 6 | 老男孩 |

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

[root@db02 backup]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock <3.sql

[root@db02 backup]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "select \* from oldboy.test;"

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

| id | name |

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

| 3 | gongli |

| 9 | hehe |

| 5 | kaka |

| 7 | lifen |

| 8 | oldboyedu.com |

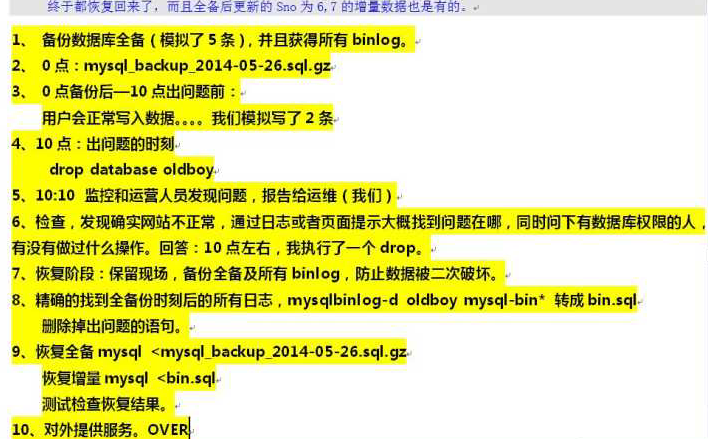
| 2 | oldgirl |

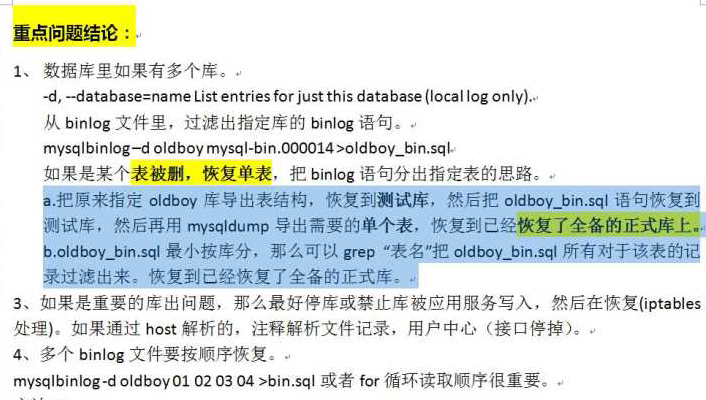
| 4 | zuma |

| 1 | 李芬 |

| 6 | 老男孩 |

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





### 3.3.3 面试题041

工作中数据库被误操作执行了一个删除的SQL语句，你如何完整恢复丢失的数据？

# 7、MySQL主从复制

## 7.1 MySQL主从复制原理介绍

MySQL的主从复制是一个异步的复制过程（虽然一般情况下感觉是实时同步的），数据库数据从一个MySQL数据库（我们称之为master）复制到另一个MySQL数据库（我们称之为slave）。在master与slave之间实现整个主从复制的过程是由三个县城参与完成的。其中有两个线程（SQL线程和IP线程）在slave端，另一个线程（IO线程）在master端（此处应边讲解边画图）。

实现MySQL的主从复制，首先必须打开master端的binlog（MySQL-bin.xxxxxx）功能，否则就无法实现主从复制。因为整个复制过程实际上就是slave从master端获取binlog日志，然后再在slave自身上以相同顺序执行获取的binlog日志中所记录的各种操作。

打开MySQL的binlog可以通过在MySQL的配置文件my.cnf中的mysqld模块（[mysqld]标识后的参数部分）增加“log-bin”参数选项。

什么是SQL？

SQL，英文全称structured query language，中文意思是结构化查询语言，它是一种关系数据库中的数据进行定义和操作的语言方法，是大多数关系数据库管理系统所支持的工业标准。

Sql server数据库

##### MySQL主从复制原理过程详细描述

下面简单描述下MySQL replication的复制过程

1、slave服务器上执行start slave，开启主从复制开关。

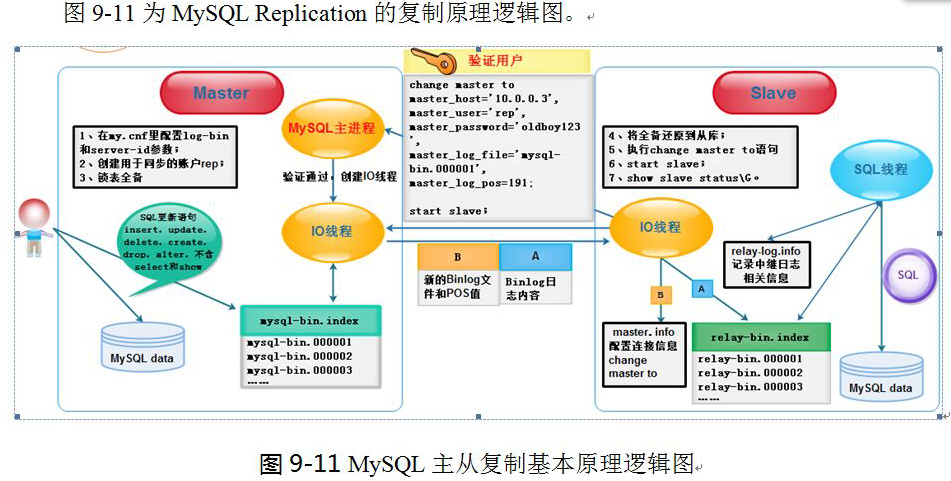
2、此时，slave服务器的IO线程会通过在master上授权的复制用户权限请求连接master服务器，并请求从指定binlog日志文件的指定位置（日志文件名和位置就是在配置主从复制服务时执行change master命令时指定的）之后发送binlog日志内容；

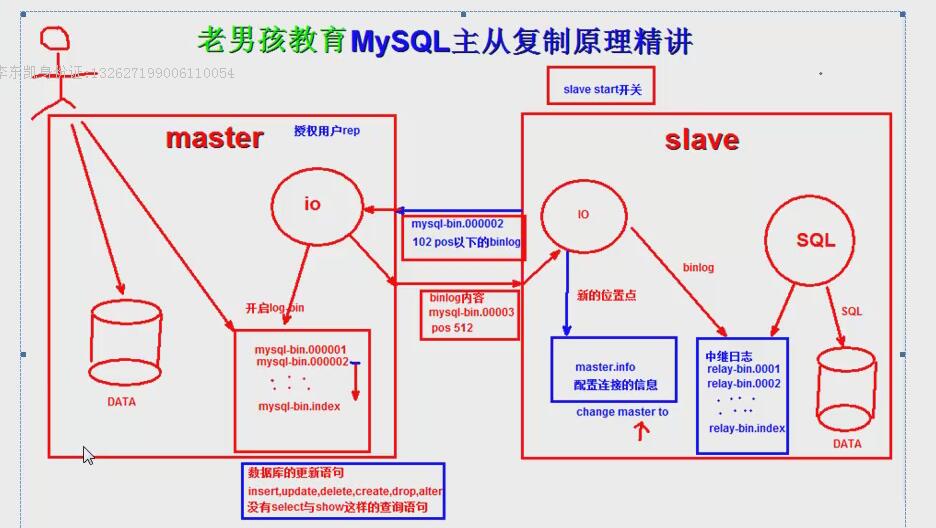
3、master服务器接受到来自slave服务器的IO线程的请求后，master服务器上负责复制的IO线程根据slave服务器的IO线程请求的信息读取指定binlog日志文件指定位置之后的binlog日志信息，然后返回给slave端的IO线程。返回的信息中除了binlog日志内容外，还有本次返回日志内容后在master服务器端的新的binlog文件名称以及在binlog中的下一个指定更新位置。

4、当slave服务器的IO线程获取到来自master服务器上IO线程发送日志内容及日志文件及位置点后，将binlog日志内容以次写入到slave端自身的relay log（即中继日志）文件（MySQL-relay-bin.xxxxxx）的最末端，并将新的binlog文件名和位置记录到master-info文件中，以便下一次读取master端新binlog日志时能够告诉master服务器需要从新binlog日志的哪个文件哪个位置开始请求信的binlog日志内容。

5、slave服务器端的SQL线程会实时的检测本地relay log中新增加的日志内容，然后及时的把log文件中的内容解析成master端曾经执行的SQL语句，应用完毕后清理应用过的日志。

6、经过了上面的过程，就可以确保在master端和slave端执行了同样的SQL语句。当复制状态正常的情况下，master端和slave端的数据时完全一样的，MySQL的同步机制是有一些特殊的情况。





##### 主从复制原理重点小结：

1、主从复制是异步的、逻辑的SQL语句级的复制。

2、同步时，主库有一个IO线程，从库有连个线程，IO和SQL线程。

3、实现主从复制的必要条件，主库要开启binlog功能。

4、binlog文件只记录对数据库有增删改的SQL语句（来自主数据库内容的变更）。

## 7.2 主从复制实践准备

### 7.2.1 定义主从复制需要的服务器角色

主库及从库IP端口：

主库及从库IP端口：

主库（mysql master）：[ip 为10.0.0.52 port 3306]

从库1（mysql slave）：[ip 为 10.0.0.52 port 3307]

从库2（mysql slave）：[ip 为 10.0.0.52 port 3308]

##### 特别提示：

1、一般常规做主从复制，主从服务器多数在不同的机器上，并且监听的端口均为默认的3306。

2、本文主从复制技术是针对前面的课程以单机数据库多实例环境来讲解的。

3、第一点虽然不在一个机器上，但是步骤和过程和2点时一样的。

### 7.1.2 主从复制数据库实战环境准备

MySQL主从复制实践，对环境要求比较简单，可以是单机单数据库多实例环境，也可以是两台服务器每个机器一个独立数据库的环境。此次实践以单机单数据库多实例的环境的环境为例，信息查看如下：

[root@db02 ~]# ss -lntup|grep 330

tcp LISTEN 0 600 \*:3306 \*:\* users:(("mysqld",5555,14))

tcp LISTEN 0 600 \*:3307 \*:\* users:(("mysqld",3497,11))

tcp LISTEN 0 600 \*:3308 \*:\* users:(("mysqld",3512,12))

提示：把3306作为主库，3307作为从库，如果根据前面的内容配置了mysql多实例环境，直接开启多实例环境使用即可。

### 7.1.3 数据库中英文名称叫法约定

MySQL主库，也可称为master，此次实例服务的端口号为3306

MySQL从库，也可称为slave，此次实例服务的端口号为3307

## 7.2 首先在主库上执行操作

### 7.2.1 设置server-id值并开启binlog参数

根据前文MySQL主从同步原理，我们知道要实现主从复制，关键因素就是开启binlog日志的功能，所以，首先要打开主库的binlog日志参数。

1. 修改主库的配置文件

执行vi /data/3306/my.cnf，编辑多实例3306的my.cnf配合文件，连个参数按如下内容修改：

[mysqld]

server-id = 1

log-bin = /data/3306/mysql-bin

提示：

1、上面两个参数要放在my.cnf中的[mysqld]模块下，否则会出错。

2、server-id的值使用服务器IP地址的最后8位如19，目的是避免不同机器或实例ID重复（不适合多实例）。

3、要先在my.cnf配置文件中查找相关参数，并按要求修改，不存在时在添加参数，切记，参数不能重复。

4、修改my.cnf配置后需重启数据库，命令：/data/3306/mysql restart，注意：要确认真正重启了。

2、检查配置参数后的结果：

[root@db02 3306]# egrep "log-bin|server-id" /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

server-id = 1

重启主库MySQL服务

[root@db02 3306]# /data/3306/mysql restart

Restarting MySQL...

Stoping MySQL...

Starting MySQL...

登录数据库检查参数的更改情况

[root@db02 3306]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock

mysql> show variables like 'server\_id';

mysql> show variables like 'log\_bin';

### 7.2.2 建立用于主从复制的账号

根据主从复制的原理，从库要想和主库同步，必须有一个可以连接主库的账号，并且这个账号的权限是主库上创建的，权限是允许主库的从库连接并同步数据。

1、登录mysql3306实例主数据库

mysql -uroot -poldboy3306 -S /data/3306/mysql.sock

2、建立用于从库复制的账号rep：

grant replication slave on \*.\* to 'rep'@'10.0.0.%' identified by 'oldboy123';

flush privileges;

# replication slave 为mysql同步的必须权限，此处不要授权all权限。

#\*.\*标识所有库所有表，也可以指定具体的库和表进行复制。oldboy.test。

#’rep’@’10.0.0.%’rep为同步账号，10.0.0.%为授权主机网段，使用了%标识允许整个10.0.0.0网段以rep用户访问。

#identified by ‘oldboy123’;oldboy123为密码，实际环境时复杂一点为好。

3、检查主库创建的rep复制账号：

show grants for 'rep'@'10.0.0.%';

### 7.2.3对主数据库锁表只读（当前窗口不要关掉）

在生产环境时，一般会每天备份一份完整数据，在备份时锁表备份并且记录下备份时的binlog对应的文件以及位置点，这样在实现主从复制的时候，就无需现锁表备份了，直接用夜里的全备即可完成主从同步的配置，另外，如果没有全备，那么最好等到夜里进行全备，也可能需要申请停机时间。因为锁表期间，会影响业务，小规模就不用了直接夜里mysqldump锁表备份即可，当然了务必要记录binlog的位置或者增加—master-data=1。

无需锁表的mysqldump命令

mysqldump –u –p –S –A –B –F –master-data=2 –x --events|gzip >/opt/all.sql.gz

innoDB引擎企业生产备份命令：推荐使用的

mysqldump –u –p –S –A –B –F –master-data=2 –x –events –single-transaction|gzip >/opt/all.sql.gz

--master-data作用：

1、使用—master-data=2进行备份文件会增加如下内容：适合普通备份增量恢复

--CHANGE MASTER TO MASTER\_LOG\_FILE=’mysql-bin.000020’，MASTER\_LOG\_POS=1191;

2、使用—master-data=1进行备份文件会增加如下内容：更适合主从复制

###### 对主数据库锁表只读（当前窗口不要关掉）

flush table with read lock; ##锁表命令，只可读，无法修改。（临时生效，退出便失效）

unlock tables; ##解锁

补充：

flush tables with read lock

flush table with read lock

提示：这个锁表命令的时间，在不同的引擎的情况，会受下面参数的控制，锁表时，如果超过设置时间不操作会自动解表。

interactive\_timeout = 60

wait\_timeout = 60

###### 默认情况下的时长为：

show variables like ‘%timeout%’;

mysql> show variables like '%timeout%';

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

| Variable\_name | Value |

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

| interactive\_timeout | 28800 |

| wait\_timeout | 28800 |

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

10 rows in set (0.00 sec)

###### 锁表后查看主库状态

查看主库状态，即当前binlog日志文件名和二进制binlog日志偏移量show master status;命令显示的信息要记录在案，后面的从库导入全备后，继续和主库复制就是要从这个位置开始复制。

show master status; ##锁表记录位置点

mysql> show master status;

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

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

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

| mysql-bin.000016 | 534 | | |

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

或新开窗口执行命令：

[root@db02 3307]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "show master status"

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

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

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

| mysql-bin.000016 | 534 | | |

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

###### 导出主数据库所有数据

单开一个新窗口，导出所有数据库数据，如果数据量很大（100G+），并且允许停机，可以停库直接打包数据文件迁移，那样更快。

[root@db02 backup]# mkdir -p /server/backup/

[root@db02 backup]# mysqldump -uroot -poldboy3306 -S /data/3306/mysql.sock -A -B --events|gzip >/server/backup/mysq\_bak\_$(date +%F).sql.gz

#-A表示备份所有库，-B表示增加use DB和drop等（导库时会直接覆盖原有的）。

为了确保导出数据期间，数据库没有数据插入，导库完毕可以再检查下主库状态信息

[root@db02 3307]# mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "show master status"

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

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

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

| mysql-bin.000016 | 534 | | |

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

#无特殊情况，binlog文件及位置点和锁表后导出数据前是一致的，即没有变化的。

导出数据完毕后，解锁主库，恢复可写，因为主库还要对外提供服务，不能一直被锁

mysql> unlock tables;

#无论主库更新多少数据，最后从库都会从上面show master status 的位置很快赶上主库的进度。

## 7.3 把主库导出的MySQL数据迁移到从库

这步常用命令有scp，rsync命令等，此次实例是单数据库多实例主从配置，因此，数据在一台机器上，查看下从库导出的数据。

[root@db02 ~]# ll bak\_2015-12-24.sql

-rw-r--r-- 1 root root 533521 12月 24 11:55 bak\_2015-12-24.sql

### 7.3.1 在MySQL从库上执行的操作过程

数据库的server-id一般在LAN内是唯一的，这里的server-id要和主库及其他从库不同，并注释掉从库的binlog参数配置，从库如果不做级联复制，并且不作为备份用，就不要开启binlog了，因为会增加磁盘IO等压力。

有两种情况需要打开log-bin，记录数据库更新的SQL语句：

a. 级联同步 A->B->C 那中间的B就要开启log-bin

b. 在从库做数据库备份，数据库备份必须要有全备和binlog日志，才是完成的备份。

1、配置从库的配置文件

执行 vi /data/3307/my.cnf，编辑my.cnf配置文件，按如下两个参数内容修改：

[mysqld]

server-id = 3

#log-bin = /data/3306/mysql-bin

提示：

1、上面两个参数要放在my.cnf中的[mysqld]模块下，否则会出错。

2、server-id的值使用服务器IP地址的最后8为如179，目的是避免不同机器或实例ID重复。

3、要先在文件中查找相关参数按要求修改，不存在时在添加参数，切记，参数不能重复。

4、修改my.cnf配置后需要重启数据库，命令为：/data/3307/mysql restart

2、检查配置参数后的结果

[root@db02 ~]# egrep "server-id|log-bin" /data/3307/my.cnf

#log-bin = /data/3307/mysql-bin

server-id = 3

3、重启数据库

[root@db02 ~]# /data/3307/mysql stop

Stoping MySQL...

[root@db02 ~]# /data/3307/mysql start

Starting MySQL...

[root@db02 ~]# ss -lntup|grep 3307

tcp LISTEN 0 600 \*:3307 \*:\* users:(("mysqld",10014,11))

4、登录数据库检查参数改变情况

[root@db02 ~]# mysql -uroot -poldboy3306 -S /data/3307/mysql.sock

mysql> show variables like 'log\_bin';

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

| Variable\_name | Value |

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

| log\_bin | OFF |

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

1 row in set (0.00 sec)

mysql> show variables like 'server\_id';

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

| Variable\_name | Value |

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

| server\_id | 3 |

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

1 row in set (0.00 sec)

7.3.2 还原主库导出的数据到从库

[root@db02 ~]# gzip -d bak\_2015-12-24.sql.gz #解压目录数据库备份

[root@db02 ~]# mysql -uroot -poldboy456 -S /data/3307/mysql.sock <bak\_2015-12-24.sql #这是恢复命令

### 7.3.4 登录从库配置同步参数

1、常规情况mysql从库连接主库的配置信息

配置信息展示：

mysql> CHANGE MASTER TO

-> MASTER\_HOST='10.0.0.52',

-> MASTER\_PORT=3306,

-> MASTER\_USER='rep'

-> MASTER\_PASSWORD='oldboy123', -> MASTER\_LOG\_FILE='mysql-bin.000016',

-> MASTER\_LOG-POS=333;

CHANGE MASTER TO

MASTER\_HOST='10.0.0.92',

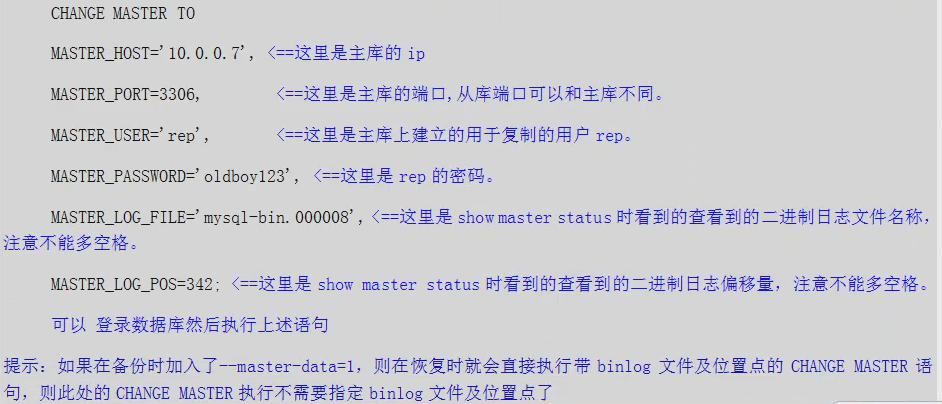
MASTER\_PORT=3306,

MASTER\_USER='rep',

MASTER\_PASSWORD='oldboy123',

MASTER\_LOG\_FILE='mysql-bin.000029',

MASTER\_LOG\_POS=107;



### 7.3.5 回顾MySQL主从同步配置步骤及原理过程

#### 一、MySQL主从同步配置步骤

1、准备两台数据库环境，或者单台多实例环境，能否正常启动和登录。

2、配置my.cnf文件，主库配置log-bin和server-id参数，从库配置server-id，不能喝主库及其他从库一样，一般不开启从库log-bin功能，注意：配置参数后要重启生效。

3、登录主库增加用于从库连接主库同步的账户例如：rep，并授权replication slave同步的权限。

4、登录主库，整库锁表flush table with read lock（窗口关闭后即失效，超时参数到了也失效）；然后show master status查看binlog的位置状态。

5、新开窗口，Linux命令行备份或导出原有的数据库数据，并拷贝到从库所在的服务器目录。

如果数据量很大，并且允许停机，可以停机打包，而不用mysqldump。

6、解锁主库，unlock tables；

7、把主库导出的原有数据恢复到从库。

8、根据主库的show master status查看binlog的位置状态，在从库执行change master to…..

9、从库开启同步开关，start slave

10、从库show slave status\G 检查同步状态，并在主库进行更新测试。

补充：

flush tables with read lock

flush table with read lock

让MySQL从库记录binlog日志方法

从库需要记录binlog的应用场景为：当前的从库还要作为其他从库的主库，例如：级联复制或双主互为主从场景的情况下，下面介绍一下从库记录binlog日志的方法。

在从库的my.cnf中加入如下参数，然后重启服务生效即可。

log-slave-updates #必须有这个参数

log-bin = /data/3307/mysql-bin

expire\_logs\_days = 7

MySQL主从复制延迟问题原因及解决方案

问题一：一个主库的从库太多，导致复制延迟。

建议从库数量3~5个位宜，要复制的从节点数量过多，会导致复制延迟。

问题二：从库硬件比主库差，导致复制延迟。

查看master和slave的系统配置，可能会因为机器配置的问题，包括磁盘IO、CPU、内存等各方面因素造成复制的延迟，一般发生在高并发大数据写入场景。

问题三：慢SQL语句过多

例如一条SQL语句，执行时间是20秒，那么从执行完毕，到从库上能查到数据也至少是20秒，这样就延迟20秒了。

问题四：主从复制的设计问题。

例如，主从复制单线程，如果主库写并发太大，来不及传送到从库就会导致延迟。更高版本的MySQL可以支持多线程复制，门户网站则会自己开发多线程同步功能。

问题五：主从库之间的网络延迟。

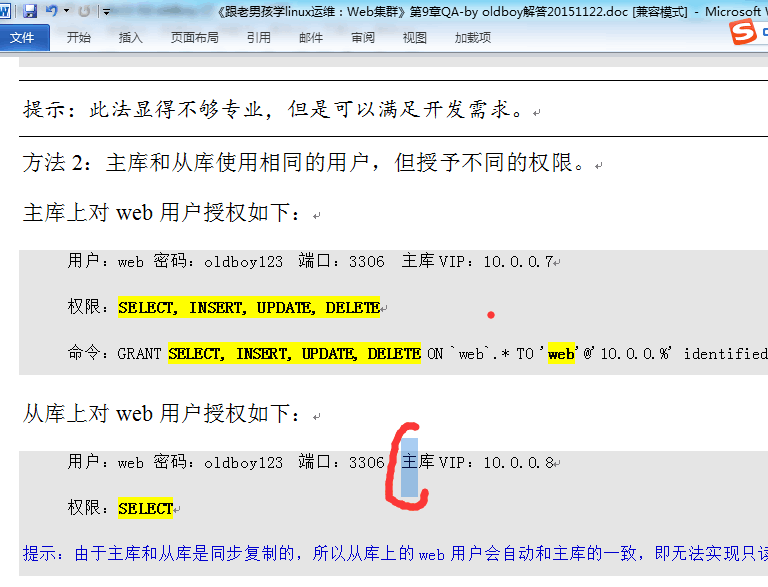
主从库的网卡、网线、连接的交换机等网络设备都可能成为复制的瓶颈，导致复制延迟，另外，跨公网主从复制很容易导致主从复制延迟。

问题六：主库读写压力大，导致复制延迟。

主库硬件要搞好一点，架构的前端加buffer以及缓存层。

通过read-only参数让从库只读访问

read-only参数选项可以让从服务器值允许来自从服务器线程或具有super权限的数据库用户进行更新。可以确保从服务器不接受来自用户端的非法用户更新。



二是忽略授权库mysql同步，主库的配置参数如下：  
binlog-ignore-db = mysql  
replicate-ignore-db = mysql

stop slave sql\_thread;

**主主同步**

关键参数：

auto\_increment\_increment = 2 #自增ID间隔，如 1 3 5

和多实例及主从同步对比：主主数据库多增加的参数：  
主从同步【主库】M需要增加的参数：  
auto\_increment\_increment        = 2  
auto\_increment\_offset           = 1  
log-slave-updates  
主从同步【从库】S需要增加的参数：  
auto\_increment\_increment        = 2  
auto\_increment\_offset           = 2  
log-bin = /data/3307/mysql-bin  
log-slave-updates  
expire\_logs\_days = 7

**指定部分库、表同步：**

DO：同步少量库  
binlog-do-db=db\_oldboy  
replicate-do-db=db\_oldboy-------------->#如需跨数据库更新并且不想复制这些更新，不应使用该选项  
replicate-do-table=db\_oldboy----------->#允许跨数据库更新。  
replicate-wild-do-table=db\_oldboy------>#用于跨数据库更新  
  
ignore：排除  
binlog-ignore-db=mysql  
replicate-ignore-db=mysql-------------->#如需跨数据库更新并且不想复制这些更新，不应使用该选项  
replicate-ignore-table=mysql----------->#该选项可以跨数据库进行更新。  
replicate-wild-ignore-table=db\_oldboy-->#该选项可以跨数据库进行更新

# 8、 MySQL数据库存储引擎知识

## 8.1 MySQL引擎概述

### 8.1.1 什么是存储引擎？

在讲清楚什么是存储引擎之前，先来打个比喻，我们都知道录制一个视频文件，可以转成不同的格式如mp4，AVI，WMV等，而存在我们电脑的磁盘上也会存在于不同类型的文件系统如Windows里常见的ntfs，fat32，存在于Linux里常见的ext3，ext4，xfs，但是，给我们或者用户看到实际视频内容都是一样的，直观区别是，占用系统的空间大小与清晰程度可能不一样。

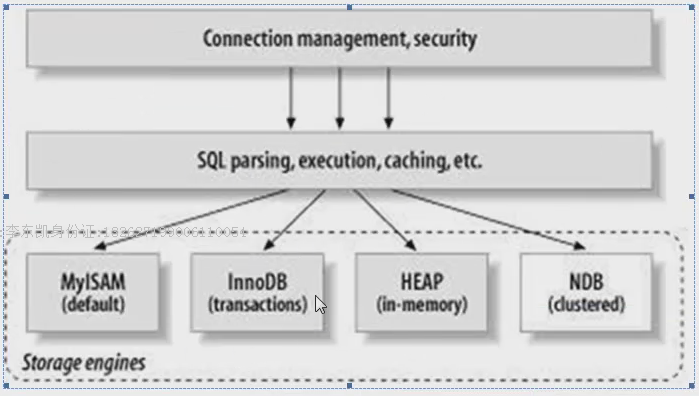
那么，数据库表里的数据存储在数据库里及磁盘上和上述的视频格式及存储磁盘文件系统格式特征类似，也有很多种存储方式。

但是，对于用户和应用程序来说同样一张表的数据，无论用什么引擎来存储，用户看到的数据都是一样的。不同的引擎存取，引擎功能，占用的空间大小，读取性能等可能有区别。

MySQL最常用存储引擎为：**MyISAM和InnoDB。**全文索引：目前5.5版本，myisam和innoDB都已经支持。

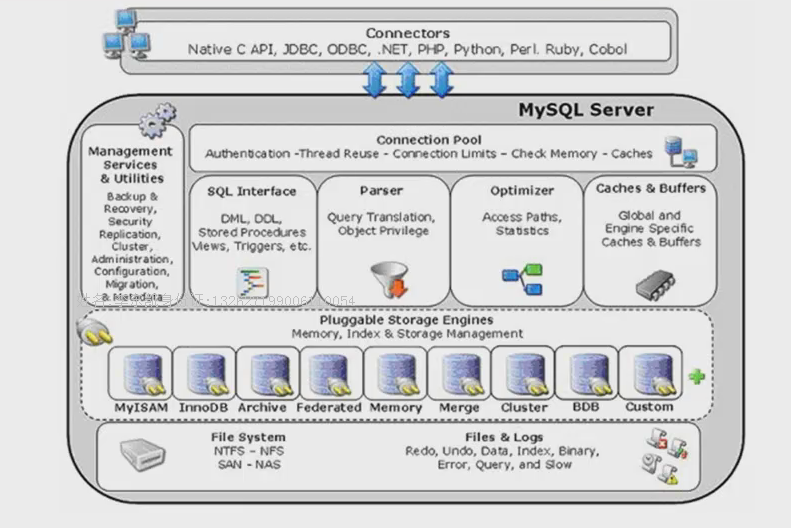
### 8.1.2 MySQL存储引擎的架构

MySQL的存储引擎是MySQL数据库的重要组成部分，MySQL常用的表的引擎为MyISAM和InnoDB两种。MySQL的每种存储引擎在MySQL里是通过插件的方式使用的，MySQL可以同时支持多种存储引擎。下面是MySQL存储引擎体系结构简图：



引擎就是组织MySQL数据的。

MySQL存储引擎体系结构简图一：



## 8.2 MyISAM引擎介绍

### 8.2.1 什么是MyISAM引擎？

MyISAM引擎是MySQL关系数据库管理系统的默认存储引擎（MySQL 5.5.5以前）这种MySQL表存储结构从旧的ISAM代码扩展出许多有用的功能。在新版本的MySQL中，InnoDB引擎由于其对事务参照完整性，以及更高的并发性等优点开始逐步的取代MyISAM引擎。

查看MySQL.5.5数据库的存储引擎：

范例：

mysql> select version();

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

| version() |

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

| 5.5.32-log |

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

1 row in set (0.00 sec)

提示：MySQL5.5数据库的默认存储引擎为MyISAM。

每一个MyISAM表都对应于硬盘上的三个文件。这三个文件有一样的文件名，但是有不同的扩展名指示其类型用途：.frm文件保存表的定义，这个文件并不是MyISAM引擎的一部分，而是服务器的一部分；.MYD保存表的数据；.MYI是表的索引文件。.MYD和.MYI是MyISAM的关键点。

### 8.2.2 MyISAM引擎特点

1、不支持事务

（事务是指逻辑上的一组操作，组成这组操作的各个单元，要么全成功，要么全失败）。

2、表级锁定

（表级锁定，数据更新时锁定整个表：其锁定机制是表级锁定，这虽然可以让锁定的实现成本很小但是也同时大大降低了其并发性能）。

3、读写互相阻塞

（读写互相阻塞：不仅会在写入的时候阻塞读取，MyISAM还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读）。

4、只会缓存索引：

（只会缓存索引：MyISAM可以通过key\_buffer\_size缓存索引，以大大提高访问性能减少磁盘IO，但是这个缓存区只会缓存索引，而不会缓存数据）。

[root@db02 3306]# grep key\_buffer my.cnf

key\_buffer\_size = 16M

5、读取速度较快，占用资源相对少。

6、不支持外键约束，但支持全文索引。

7、MyISAM引擎是mysql5.5.5前缺省的存储引擎

### 8.2.3 MyISAM引擎使用的生产业务场景

1、不需要事务支持的业务（例如转账就不行）。

2、一般为读数据比较多的应用，读写都频繁场景不适合，读多或者写多的都适合。

3、读写并发访问相对较低的业务（纯读纯写高并发也可以）（锁定机制问题）。

4、数据修改相对较少的业务（阻塞问题）。

5、以读为主的业务，例如：数据库系统表、www、blog，图片信息数据库，用户数据库，商品库等业务。

6、对数据一致性要求不是非常高的业务（不支持事务）。

7、硬件资源比较差的机器可以用MyISAM（占用资源少）。

8、单一读写分离的MySQL从库可以使用MyISAM，所谓单一就是尽量纯读，或纯写（insert，update，delete）等。

9、使用读写分离的MySQL从库可以使用MyISAM。

### 8.2.4 MyISAM引擎调优精要

1、设置合适的索引（缓存机制）。

2、调整读写优先级，根据实际需求确保重要操作更优先执行。

3、启用延迟插入改善大批量写入性能（降低写入频率，尽可能多条数据一次性写入）。

4、尽量顺序操作让insert数据都写入到尾部，减少阻塞。

5、分解大的时间长的SQL操作，降低单个操作的阻塞时间。

6、降低并发数（减少对MySQL访问），某些高并发场景通过应用进行排队队列机制Q队列。

7、对于相对静态（更改不频繁）的数据库数据，充分利用query cache或memcached缓存服务可以极大的提高访问效率，网站动态内容静态化，减少对数据库的访问。

[root@db02 3306]# grep query my.cnf

query\_cache\_size = 2M

query\_cache\_limit = 1M

query\_cache\_min\_res\_unit = 2k

8、MyISAM的count只有在全表扫描的时候特别高效，带有其他条件的count都需要进行实际的数据访问。

mysql> select count(\*) from oldboy.test;

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

| count(\*) |

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

| 10 |

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

1 row in set (0.00 sec)

9、可以把主从同步的主库使用innoDB，从库使用MyISAM引擎（不推荐）。

## 8.3 innoDB引擎

### 8.3.1 什么是innoDB引擎？

innoDB引擎是MySQL数据库的另一个重要的存储引擎，正成为目前MySQL AB所发行新版的表，被包含在所有二进制安装包里。和其他的存储引擎相比，innoDB引擎的优点是支持兼容ACID的事务（类似于postgreSQL）,以及参数完整性（即对外键的支持）。oracle公司2005年10月收购了innobase。innobase采用双认证授权，它使用GNU发行，也允许其他想将innoDB结合到行业软件的团体获得授权。

mysql> select version();

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

| version() |

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

| 5.5.32-log |

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

1 row in set (0.00 sec)

mysql> show create table oldboy.test\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: test

Create Table: CREATE TABLE `test` (

`id` int(4) NOT NULL AUTO\_INCREMENT,

`name` char(20) NOT NULL,

PRIMARY KEY (`id`),

KEY `index\_name` (`name`)

) **ENGINE=InnoDB** AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

[root@db02 data]# pwd

/data/3306/data

[root@db02 data]# du -sh ibdata1 **#innoDB引擎磁盘文件，默认128M，存放数据innodb的文件**

128M ibdata1

### 8.3.2 innoDB引擎特点

1、支持事务

（支持事务：支持4个事务隔离级别，支持多版本读）。

2、行级锁定

（行级锁定，更新时一般是锁定当前行：通过索引实现，全表扫描仍然会使表锁，注意间隙所的影响）。

3、读写阻塞与事务隔离级别相关。

4、具有非常高效的缓存特性：能缓存索引，也能缓存数据。

5、整个表和主键以cluster方式存储，组成一颗平衡树。

6、所有secondary index都会保存主键信息。

7、支持分区，表空间，类似oracle数据库。

8、支持外键约束，5.5以前不支持全文索引，以后支持了。

9、supports transactions，row-level locking，and foreign keys

10、和MyISAM引擎比，innoDB对硬件资源要求比较高。

innodb特点：

1、Row-level locking #行级锁定

2、full-text search indexes #支持全文索引

3、data caches #数据缓存

4、index caches #索引缓存

5、transactions #支持事务

6、占用资源多 #

7、读写阻塞与事务隔离级别相关。 #

8、支持外键

### 8.3.3innoDB引擎适用场景

1、需要事务支持的业务（具有较好的事务特性）。

2、行级锁定对高并发有很好的适应能力，但需要确保查询是通过索引完成的。

3、数据读写及更新都较为频繁的场景，如：BBS，SNS，微博，微信等。

4、数据一致性要求较高的业务，例如：充值转账，银行卡转账。

5、硬件设备内存较大，可以利用innoDB较好的缓存能力来提高内存利用率，尽可能减少磁盘IO。

[root@db02 3306]# grep -i innodb my.cnf

#default\_table\_type = InnoDB

innodb\_additional\_mem\_pool\_size = 4M

innodb\_buffer\_pool\_size = 32M

innodb\_data\_file\_path = ibdata1:128M:autoextend

innodb\_file\_io\_threads = 4

innodb\_thread\_concurrency = 8

innodb\_flush\_log\_at\_trx\_commit = 2

innodb\_log\_buffer\_size = 2M

innodb\_log\_file\_size = 4M

innodb\_log\_files\_in\_group = 3

innodb\_max\_dirty\_pages\_pct = 90

innodb\_lock\_wait\_timeout = 120

innodb\_file\_per\_table = 0

共享表空间对应物理数据文件：

[root@db02 3306]# ll -h /data/3306/data/ibdata1

-rw-rw---- 1 mysql mysql 128M 12月 24 13:03 /data/3306/data/ibdata1

独立表空间对应物理数据文件

innodb\_file\_per\_table

innodb\_data\_home\_dir = /data/xxx

6、相比MyISAM引擎，innodb引擎更消耗资源，速度没有MyISAM引擎快。

### 8.3.4 innodb引擎调优精要

1、主键尽可能小，避免给secondary index带来过大的空间负担。

2、建立有效索引避免全表扫描，因为会使用表锁。

3、尽可能缓存所有的索引和数据，提高相应速度，煎炒磁盘IO消耗。

4、早大批量小插入的时候，尽量自己控制事务而不要使用autocommit字段提交，有关可以控制提交方式

5、合理设置innodb\_flush\_log\_at\_trx\_commit参数值，不要过度最求安全性。

如果innodb\_flush\_log\_at\_trx\_commit的值为0，log buffer每秒就会被刷写日志文件到磁盘，提交事务的时候不做任何操作

6、避免主键更新，因为这会带来大量的数据移动。

MySQL数据库安全权限控制管理思想

1、制度与流程控制

1.1 项目开发制度流程

办公开发环境--->办公测试环境--->IDC测试环境--->IDC正式环境，通过这种较完善的项目开发制度及流程控制，尽可能的防止潜在的问题隐患发生。

1.2 数据库更新流程

开发人员提交需求--->开发主管审核--->部门领导审核--->DBA（运维）审核--->DBA（运维）执行项目开发制度及流程控制的数据库更新步骤（每个步骤都要测试），最后在IDC正式环境执行。

需要说明的是，在开发人员一开始提交需求时，就可以同时抄给以上的领导及审核人员，然后，审核人员依次审核。对于特殊紧急需求，可以根据紧急程度特殊处理，这里可以指定个紧急需求处理流程，比如：开发人员提交需求--->DBA（运维）审核，然后操作完在汇报给其他审核人员。

通过完善的数据库更新流程控制，可以防止很多潜在的数据丢失、破坏等问题发生。

1.3 DBA参与项目数据库设计

在项目开发环节上，DBA或资深运维人员最好参与数据库设计与审核工作，这样可以从源头上减少降低不良的数据库设计及不良SQL语句的发生，还可以做所有语句的审核工作，包括select，但这个需要评估工作量是否允许，一般的互联网公司实施全审核比较困难。

1.4 各种操作申请流程

1） 开发等人员权限申请流程。

2） 数据库更新执行流程。

3） 烂SQL语句计入KPI考核。

1.5 定期对内部人员培训

定期给开发及相关人员培训，目的还是从源头上降低不良数据库设计及不良SQL语句的发生，并通过培训让大家知晓数据库性能的重要性，让大家提升开发时照顾数据库性能的意识。

1） 数据库设计规范及制度。

2） SQL语句执行优化，性能优化技巧等。

3） 数据库架构设计等内容。

2、账户权限控制

2.1 内部开发等人员权限分配

1） 权限申请流程要设置规范、合理，让需求不明确者知难而退（比如老男孩老师的曾经公司开发就有上百人）。

2） 办公开发和测试环境可以放开权限，IDC测试和正式环境要严格控制数据库写权限，并且读权限和对外业务服务分离。

3） 开发人员正式环境数据库权限分配规则：给单独的不对外服务的正式从库只读权限，不能分配线上正式主库写权限。

4） 特殊人员（如领导），需要权限时，我们要问清楚他做什么，发邮件回复，注明用户名、密码、权限范围，多提醒操作注意事项，如果有可能由DBA人员代替其操作。

5） 特权账号（all privliges），由DBA控制，禁止在任何客户端上执行特权账号操作（如智能localhost或其他策略）。

2.2 web账户权限分配制度

1）写库账号默认权限为 select，insert，update

2）读库账号默认权限为select（配合mysql read-only）

4） 如果是lanmp一体在一台服务器的环境，授权localhost，

授权不要用%，尽量只用固定IP或划分VLAN子网掩码

3、数据库客户端访问控制

1、更改默认mysql client端口，如PHPmyadmin管理端口为9999，其它客户端也是一样。

2、数据库web client端统一部署在1-2台不对外服务的web上，闲置IP及9999端口只能从办公室内访问。

3、不做公网域名解析，用host实现访问（闲置任何IP直接访问）或者用内网IP访问。

5、闲置使用务必连接的账号管理数据库，根据开发人员用户角色分配指定账号访问。

6、按开发及相关人员根据职位角色分配适合的管理账号。

6、 数据库运维管理思想核心

1） 未雨绸缪 ，不要停留在制度上，想，而是，实际做出来（行动）。

2） 亡羊补牢，举一反三，切记，不能好了伤疤忘了疼。

3） 完备的架构设计及备份、恢复策略。

4） 定期思考、并实战模拟以上策略演练。

未雨绸缪永远比除了问题再处理要好的多，除了问题补救不没办法不得已的事，最差的是很多公司，没有亡羊补牢，而是好了伤疤忘了疼，没过多久问题又发生。

因此，在工作中药尽量做到未雨绸缪，从源头上减少故障的发生。其次，要做到亡羊补牢、举一反三，事情出现一次就不能再出现第二次。当然，完善的备份和恢复策略也是需要做的。只有把这些结合起来，才能把我们运维的工作做得更好。