辛星笔记之MySQL进阶视频

----------即李强强视频----------

----------于2014年12月----------

**********寄语**********

1.百度搜索“辛星 笔记”可以找到更多更全更优秀的资料奥。

2.去除废话，提炼精华，辛星笔记，始终如一。

**********目录**********

**********辛星笔记**********

传播知识，传递温情。 我心永恒，始终如一。

前进的路上，辛星陪伴您。 只要星哥在，编程充满爱。

我的博客：blog.csdn.net/xinguimeng

我的邮箱：xinguimeng@163.com

**********致敬原著**********

1.这是兄弟连的李强强讲师的特级课视频的MySQL部分，在此祝愿工作顺利，万事如意。

2.敬人者，人恒敬之，爱人者，人恒爱之。

第一节：用户授权

**********淘宝数据库发展三个阶段**********

1.第一阶段：整个网站使用 lamp 架构

数据库采用几台 MySQL

应用系统分为前台、后台两大系统

2.第二阶段：MySQL 迁移到 Oracle

PC server 升级到 IBM 小型机

低端存储升级到高端存储

3.第三阶段：核心业务从 Oracle 逐步迁移到分布式 MySQL 集群

大量采用 PC server，采用本地硬盘

**********基本语句**********

1.给从服务器设置授权用户：

mysql>grant all on *.* to user@192.168.10.2 indentified by "pass"

mysql>grant replication slave on *.* to user@192.168.10.2 indentified by "pass"

2.查看用户授权表：

select suer,host,password form mysql.user;

**********binl-log 日志**********

1.开启 MySQL 的 bin-log 日志：

vi /etc/my.cnf

2.修改地方如下：

[mysqld]

port=3306

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

log-slow-queries=mysql-slow.log

log-error=mysql.err

log=mysql.log

log-bin=mysql-bin

#这里给出的是文件名的前缀

3.查看 bin-log 日志是否开启，在 MySQL 中可以用：

show variables like "%log%";

**********与 bin-log 有关的操作**********

1.mysql>flush logs；会多出一个最新的 bin-log 日志。

2.mysql>show master status；查看最后一个 bin 日志。

3.mysql>reset master；清空所有的 bin-log 日志。

4.使用 mysqlbinlog --no-defaults mysql-00001.bin 来查看二进制日志信息。

第二节：主从复制

**********操作**********

1.备份数据：

mysqldump -u 用户名 -p 密码 数据库 -l -F >'文件名'

2.其中的-F 即 flush logs，可以重新生成新的日志文件，当然包括 log-bin 日志。

3.此处的-l 是 lock 的简写，表示读锁，此时无法进行写操作，但是可以进行读操作。

4.查看 binlog 日志用 mysql>show master status。

**********数据恢复**********

1.mysql 恢复数据：

mysql -u 用户名 -p 密码 数据库 -v -f < sql 文件

2.其中-v 是查看导入的详细信息，其中-f 是当中间遇到错误时，可以 skip 过去，继续执行下面的语句，这里的 f 是 force 的简写。

3.恢复二进制日志文件：

mysqlbinlog --no-defaults 文件名|mysql -u 用户名 -p 密码

********完整的 bin-log 日志恢复**********

1.创建一张表 t1。

2.添加若干数据。

3.使用 mysqldump 备份现有数据。

4.再次添加一些数据。

5.删除数据库中的数据。

6.用已经备份的 sql 文件来恢复数据。

7.使用 bin-log 来恢复其他数据。

8.这里有个问题就是在删除数据之前必须得先生成 bin-log 日志。

9.而 mysqlbinlog 命令可以使用--stop-position="533"来指定恢复到指定 position 的位置，还可以通过指定--start-position，--stop-date，--start-date 等等。

**********主从复制优点**********

1.优点一：如果主服务器出现问题，可以快速切换到从服务器提供的服务。

2.优点二：可以在从服务器上执行查询操作，降低主服务器的访问压力。

3.优点三：可以在从服务器上执行备份，以避免备份期间影响主服务器的服务。

4.一般只有更新不频繁的数据或者对实时性要求不高的数据可以通过从服务器查询，实时性要求高的数据依然需要从主数据库获得。

**********主服务器配置**********

1.登录 mysql 数据库：

mysql -u 用户名 -p 密码

2.给从服务器设置授权用户：

grant all slave on *.* to user@192.168.10.2 indentified by "pass"

或者

grant replication slave on *.* to user@192.168.10.2 indentified by "pass"

3.修改主数据库服务器的配置文件 my.cnf，开启 binlog 日志，并且设置 server-id 的值：

log-bin=mysql-bin

server-id=1

4.(选做)在主服务器上设置读锁定有效，确保没有数据库操作，以便获得一个一致性的快照：

mysql>flush tables with read lock;

5.查看主服务器上当前的二进制名和偏移量值：

mysql>show master status；

6.目前主数据库服务器已经停止了更新操作，生成主数据库的备份，备份的方式有两种：

(1)cp 全部的数据。

(2)mysqldump 备份数据。

7.如果主数据库的服务可以停止，那么直接 cp 数据文件应该是最快的生成快照的方法。

8.主数据库备份完毕之后，主服务器可以恢复写操作，剩下的操作只需要在从服务器上去执行：

mysql>unlock tables；

9.把数据库的一致性备份恢复到从数据库上，把以上的压缩包解压后放到相应的目录即可。

10.可以使用 show slave status 来查看从服务器状态。

**********从服务器配置**********

1.修改从数据库的 server-id，注意 server-id 的值必须是唯一的，不能和数据库的配置相同，如果有多个从服务器，每个从服务器必须有自己唯一的 server-id 值。

2.在从服务器的配置文件中修改如下配置项：

master-host、master-user、master-password、master-port、log-bin、replicate-do-db、replicate-do-table

3.重新启动 mysql 即可。

**********查看状态**********

1.查看相应的主从复制进程列表有两种。

2.第一种是使用 show processlist 的方式

如果出现：state：waiting for master to send event，说明连接主数据库为成功，且成功获取 bin-log

如果出现 state：has read all ready log；waiting for the slave i/o thread to update it 说明成功执行 bin-log 日志，正在等待着去再次连接主数据库，并更新获取 bin-log 日志。

3.第二种方式即使用 show slave status 的方式：

Slave_IO_Running:yes 此进程负责从服务器从主服务器上读取 binlog 日志，并写入从服务器上的中继日志中。

Slave_SQL_Running：yes 此进程负责读取并且执行中继日志中的 binlog 日志。

注意以上两个都为 yes 则表明成功，只要其中一个进程的状态是 no，则表示复制进程停止，错误原因可以从 last error 字段的值中看到。

**********数据库常用命令**********

1.启动复制线程：start slave

2.停止复制线程：stop slave

3.查看从数据库状态：show slave status

4.查看主数据库 bin-log 日志：show master logs

5.动态改变到主服务器的配置：change master to

6.查看从数据库运行进程：show processlist

7.一般从服务器启动时会默认执行 start slave。

**********常见问题**********

1.有时候数据库无法从不，比如 show slave status 显示 slave_sql_running 为 no，而 seconds_behind_master 为 null。

2.原因可能如下：

(1)程序可能在 slave 上进行了写操作。

(2)slave 机器重启后，事务回滚

3.解决办法一：首先 slave stop，然后 set GLOBAL SQL_SLAVE_SKIP_COUNTER=1，然后 slave start。

4.解决方法二：对于 slave 执行 slave stop，对于 master 来执行 show master status 来得到主服务器上当前的二进制日志名和偏移量，然后到 slave 服务器上执行手动同步，即 change master to 来修改 master_log_file 和 master_log_pos 的值，然后启动 slave 服务即可。

5.通过 show slave status 查看 Slave_SQL_Running 为 yes，而 Seconds_Behind_Master 为 0 即为正常。

第三节：分区理论

**********大数据存储**********

1.当 mysql 中一个表的总记录数超过了 1000 万，会出现性能的大幅度下降吗？答案是肯定的。

2.但性能下降的比率由系统的架构、应用程序、数据库索引、服务器硬件等多种因素而定。

3.数据库多达上亿的数据量，分表之后的单个表也已经突破千万，那么单个表的更新等均影响着系统的运行效率。

4.甚至是一条简单的 SQL 都有可能压垮整个数据库，比如整个表对某个字段的排序操作等等。

**********解决方案**********

1.针对海量数据的优化主要有两种方法：大表拆成小表和 SQL 语句的优化。

2.对于 SQL 语句的优化，可以通过增加索引等来调整，但是数据量的增大将会导致索引的维护代价增大。

3.对于大表拆成小表，可以分为垂直分表(竖切)和水平分表(横切)。

**********水平分表**********

1.水平分表技术就是将一个表拆分成多个表，比较常用的方式是将表中的记录按照某种 hash 算法进行拆分，简单的拆分方式比如取模方式。这种分区方法也必须对前端的应用程序中的 SQL 进行修改方可使用。

2.比如一个 SQL，它可能会修改两个表，那么我们必须得写成两个 SQL 语句从而可以完成一个逻辑的事务。它会使得程序的判断逻辑越来越复杂，这样也会导致程序的维护代价高，也就失去了采用数据库的优势。

3.因此，分区技术可以有力地避免以上的弊端，成为解决海量数据存储的有力方法。

**********mysql 分区介绍**********

1.mysql 的分区技术不同于之前的分表技术，它与之前水平分表有点类似，但是它是在逻辑层进行的水平分表，对于应用程序而言它还是一张表。

2.MySQL5.1 有 4 种分区类型：

(1)RANGE 分区：基于属于一个给定连续区间的列值，把多行分配给分区。

(2)LIST 分区：类似于按 RANGE 分区，区别是在于 LIST 分区是基于列值匹配一个离散值集合中的某个值来进行选择。

(3)HASH 分区：基于用户定义的表达式的返回值来进行选择的分区，该表达式使用将要插入到表中的这些行的列值进行计算，这个函数可以包含 MySQL 中有效的、产生非负整数值的任何表达式。

(4)KEY 分区：类似于按 HASH 分区，区别在于 KEY 分区只支持计算一列或多列，且 MySQL 服务器提供其自身的 HASH 函数。

**********RANGE 分区**********

1.使用分区建表语句范例：

```
create table 表名(各个列定义)
partition by range(某个字段名)(
  partition p0 values less than(6),
  partition p1 values less than (11),
  partition p2 values less than (16),
  partition p3 values less than(21)
);
```

2.上面例子中我们就看到它可以对于某个字段进行大小判断，通过分区的值来判断应该放置到哪个表中。

**********LIST 分区**********

1.list 分区与 range 分区有很多相似的地方，它通常用于值属于枚举类型的情况。

2.范例如下：

```
create table 表名(各列)
partition by list(字段名)(
 partition pn values in(3,5,6,9),
 partition pe values in (1,2,10)
....
);
```

**********hash 分区**********

1.hash 分区主要用于确保数据在预先确定数目的分区中平均分布。

2.它可以基于用户定义的表达式的返回值来进行选择的分区，该表达式使用将要插入到表中的这些行的列值进行计算。

3.范例：

```
create table 表名(各个字段)
partition by hash(字段名)
partitions 4;
```

4.哈希分区多用于测试。

**********key 分区**********

1.与 hash 分区类似，但是它的 key 可以不是整数类型，比如用字符串等类型的字段。

2.MySQL 簇(Cluster)使用函数 md5()来实现 key 分区，对于使用其他存储引擎的表，服务器使用自己内部的哈希函数，这些函数是基于与 password()一样的运算法则。

**********不同分区技术对比**********

1.range 分区的优点在于适合与日期类型，支持复合分区，缺点是有限的分区数目，它一般只针对某一列。

2.list 分区适合与有固定取值的列，支持复合分区，缺点是有限的分区，插入记录在这一列的值不在 list 中，则数据丢失，它一般只针对某一列。

3.hash 分区的优点是线性的 hash 使得增加、删除和合并分区更加快捷，缺点是线性 hash 的数据分布不均匀，而一般 hash 的数据分布较为均匀，一般只针对某一列。

4.key 分区的优点在于列可以为字符型等其他非 int 类型，缺点是效率较之前的低，因为函数为复杂的函数，比如 md5 或者 sha 函数，一般只针对某一列。

**********操作**********

1.我们可以用 show plugins 来查看插件。

2.分区插件就是 partition。

3.在 MyISAM 引擎下，每个分区都会存储为两个文件，还有一个 frm 文件和一个 par 文件。

第四节：分区实验

**********插入大量数据**********

1.在 mysql 中插入大量数据的方式可以使用这种方式：

```
insert into t1 select * from t1;
```

2.当然用存储过程也相当不错。

**********存储过程**********

1.可以用\d //来把分隔符换为//，当然用 delimiter //也可行。

2.下面是一个向 t3 表插入数据的存储过程：

```
create procedure p3()
begin
set@i =1;
while @i < 10000 do
insert into t3 values(@i);
set @i = @i +1;
end while;
end //
```

3.查看存储过程的语句：show procedure status

4.调用该存储过程：call p3()；

5.查看关于存储过程的帮助用? procedure 即可。

**********实验测试**********

1.这里李强强讲师使用三个字段，使用 MyISAM 引擎插入八百万行数据，然后使用分区和未分区两种建表方式来进行测试。

2.对于 select 其中的某个时间段内，使用分区的使用时间为 0.55s，未使用分区为 4.69s，则分区表比未分区表的执行时间少百分之九十九。

3.我们使用 desc　select 语句的时候，发现它的 rows 参数会小很多。

4.当然用 explain　select 语句也可以。

5.当在搜索的该列创建索引之后，两者差别不大，即使用分区和未分区在这种情况下几乎无分别。

6.如果对使用分区的字段进行区间判断，并且增加一个不使用分区的字段的值且匹配一个不存在的值的时候，发现未使用分区的时候会远远慢于使用索引的情况。

**********InnoDB**********

1.对于 InnoDB 的数据结构，首先要解决两个概念性的问题：共享表空间和独占表空间。

2.共享表空间以及独占表空间都是针对数据的存储方式而言的。

3.共享表空间：某一个数据库的所有的表数据，索引文件全部放在一个文件中，默认这个共享表空间的文件路径在 data 目录下，默认的文件名为 ibdata1，初始化为 10M。

4.独占表空间：每一个表都将会生成以独立的文件方式来进行存储，每一个表都有一个.frm 表描述文件，还有一个.ibd 文件。其中这个文件包括了单独一个表的数据内容以及索引内容，默认情况下它的存储位置也是在表的位置之中。

*********InnoDB 共享表空间**********

1.优点：可以放表空间分成多个文件存放到磁盘上，表空间文件大小不受表大小的限制，比如一个表可以分布在不同的文件上。数据和文件放在一起方便管理。

2.缺点：所有的数据和索引放到一个文件中意味着将有一个很大的文件，虽然可以把一个大文件分成多个小文件，但是多个表以及索引在表空间中混合存储，这样对于一个表做了大量的删除操作后表空间中将会有大量的空隙，特别是对于统计分析，日志系统这类应用最不适合用共享表空间。

**********相关配置**********

1.示例配置：

innodb_data_home_dir="C:\mysql\data\"

innodb_log_group_home_dir="C:\mysql\data\"

innodb_data_file_path=ibdata1:10M:autoextend

innodb_file_per_table=1

2.参数说明：这个设置配置一个可扩展的大小尺寸为 10M 的单独文件，名为 ibdata1，没有给出文件的位置，所以默认的是在 MySQL 的数据目录内。

3.其中 innodb_data_home_dir 代表为数据库文件锁存放的目录，其中 innodb_log_group_home_dir 为日志存放目录，其中 innodb_file_per_table 是否使用共享以及独占表空间。

4.其中以上几个参数必须在一起加入。

**********独占表空间**********

1.在使用 InnoDB 引擎的时候，只有在开启了独占表空间之后，才能创建成功 InnoDB 表引擎的表分区。

2.开启了独占表分区，也就是配置 innodb_file_per_table 为 1，此时每个表会有单独的 ibd 文件。

3.我们可以使用 show variables like "%per_table%"来查看是否开启了独占表空间。

第五节：基础操作

**********数据库优化包含知识点**********

1.mysql 基础操作

2.常用的 sql 技巧

3.sql 语句优化

4.mysql 数据库优化

5.MyISAM 表锁

6.MySQL 服务器优化

**********mysql 基础操作**********

1.这里所说的 mysql 基础操作分为：

表复制、索引、视图、内置函数、预处理语句、事务处理、存储过程、触发器、重排 auto_increment 值

2.对于 mysql 的函数，主要侧重于字符串函数和数学函数。

**********mysql 表复制**********

1.复制表结构+复制表数据：

```
create table t3 like t1;
insert into t3 select * from t1;
```

2.此时的 t3 表会拥有和 t1 一样的结构和数据。

**********mysql 索引**********

1.可以使用 alter table 来创建普通索引、unique 索引和 primary key 索引，分别如下：

```
alter table 表名 add index 索引名(字段名)
alter table 表名 add unique (字段名)
alter table 表名 add primary key(字段名)
```

2.当然也可以使用 create index 系列的语句：

create index 索引名 on 表名(字段名)

create unique index 索引名 on 表名(字段名)

3.删除索引使用 drop index：

drop index 索引名 on 表名

4.使用 alter table 来删除索引：

alter table 表名 drop index 索引名

alter table 表名 drop primary key

5.因为 create index 无法创建主键索引，所以通用的还是使用 alter table 的方式。

6.查看某个表 t1 的索引：

show index from t1；

7.当我们对某个字段设置了 auto_increment 的时候，是不能够用 alter table 表名 drop primary key 来删除的，因为它会报错：Incorrect table definition；there can be only one auto column and it must be defined as a key，也就是说，我们必须得先移除该属性，才能取消它的主键。

**********视图**********

1.创建视图范例：

create view 视图名 as select 语句；

2.view 视图的帮助信息使用? view 即可。

3.注意查看视图使用 show views。

4.删除视图使用：drop view 视图名。

**********mysql 内置函数**********

1.字符串函数：

concat(string2 [,....]) 连接字符串

lcase(string) 转换成小写

ucase (string)转换成大写

length(string) 字符串长度

ltrim(string) 去除前端空格

rtrim(string) 去除后端空格

repeat(string,count) 重复 count 次

replace(str,search_str,replace_str) 在 str 中用 replace_str 替换 search_str

substring(str,position [,length]) 在 str 的 position 开始，取 length 个字符，注意这里的第一个位置是 1，而不是 0

space(count) 生成 count 个空格

2.数学函数：

bin(decimal_number) 十进制转二进制

ceiling(number) 向上取整

floor(number) 向下取整

max(num1,num2) 取最大值

min(num1,num2) 取最小值

sqrt(number) 开平方

rand() 返回 0-1 内的随机值

3.日期函数：

curdate() 返回当前日期

curtime() 返回当前时间

now() 返回当前的日期时间

unix_timestamp(date) 但会当前 date 的 Unix 时间戳

from_unixtime(timestamp) 返回 Unix 时间戳的日期值

week(date) 返回日期 date 为一年中的第几周

year(date)　返回日期 date 的年份

datediff(expr,expr2)　返回其实时间和结束时间 expr2 天数

**********预处理语句**********

1.设置 stmt1 预处理，传递一个数据作为一个 where 判断条件：

prepare stmt1 form 'select * from t1 where id >?';

2.设置一个变量：

set @i = 1

3.执行 stmt1 预处理：

execute stmt1 using @i;

4.设置@i 为 5：

set @i = 5

5.再次执行 stmt1：

execute stmt1 using @i;

6.删除预处理 stmt1：

drop prepare stmt1;

**********事务处理**********

1.关闭自动提交功能：

set autocommit =0;

2.从表 t1 中删除了一条记录：

delete from t1 where id=11;

3.此时做一个 p1 还原点：

savepoint p1;

4.再次从表 t1 中删除一条记录：

delete from t1 where id = 10;

5.再次做一个 p2 还原点：

savepoint p2；

6.此时恢复到 p1 还原点，当然后面的 p2 这些还原点会自动失效：

rollback to p1；

7.退回到最原始的还原点：

rollback ；

8.注意 MyISAM 是不支持事务机制的。

9.修改表为 InnoDB 的表引擎：

alter table m engine = innodb；

**********存储**********

1.创建一个存储 p1()：

```
mysql>\d //
mysql> create procedure p1()
    ->begin
    ->set @i =0;
    ->while @i <10 do
    -> select @i;
    ->set @i = @i +1;
    ->end while;
    ->end ;
    ->//
```

2.执行存储 p1();

```
mysql>\d ;
mysql> call p1();
```

3.查看 procedure p1()的 status 信息：

```
mysql>show  procedure status \G
```

4.查看 procedure p1()的具体信息：

```
mysql> show  create  procedure p1\G
```

**********触发器**********

1.修改 delimiter 为//，范例：

```
mysql>\d //
```

2.创建一个名字为 tg1 的触发器，当向 t1 表中插入数据时，就向 t2 表中插入一条数据：

```
mysql> create trigger  tg1 before insert on t1 for each row
    > begin
    >insert  into t2(id)  values(new.id);
    >end //
```

3.准备两个空表 t1 和 t2.

4.向 t1 表中插入多条数据，之后看 t2 表的数据。

5.删除表 t1 后 t2 表中的数据也会跟着删除：

```
mysql >\d //
mysql> create trigger tg2 before delete on t1 for each row
    >begin
    > delete from t2 where id = old.id;
    >end //
mysql>\d ;
```

6.更改表 t1 后 t2 表中的记录跟着更改 范例：

```
mysql>\d //
```

```
mysql>create trigger tg3 before update on t1  for  each row
    >begin
    >update t2 set  id = new.id where id = old.id;
    >end  //
mysql>\d ;
```

7.查看触发器：

```
show  triggers
```

**********重排 auto_increment 值**********

1.当我们清空表的时候，不能用 delete from tablename，而是使用 truncate table tablename 或者 truncate tablename，这样 auto_increment 就恢复成 1 了。

2.或者清空内容后使用 alter 命令修改如下：

```
alter  table  tablename  auto_increment = 1;
```

**********常用 SQL 技巧**********

1.正则表达式的使用

2.使用 rand()提取随机行

3.利用 group by 的 with rollup 子句做统计

4.用 bit group functions 做统计

5.使用外键需要注意的问题

6.mysql 中 help 的使用

**********正则表达式的使用**********

1.MySQL 利用 regexp 命令提供给用户扩展的正则表达式功能，具体模式序列如下。

2.其中^表示在字符串的开始处进行匹配。

3.其中$表示在字符串的末尾处进行匹配。

4.其中.表示匹配任意单个字符，包括换行符。

5.其中[...]表示匹配出括号内的任意字符。

6.其中[^...]表示匹配不出现括号内的任意字符。

7.其中 a*表示匹配零个或者多个 a(包括空格)。

8.其中 a+表示匹配 1 个或者多个 a(不包括空格)。

9.其中 a?表示匹配 1 个或者零个 a。

10.其中 a1|a2 表示匹配 a1 或者 a2。

11.其中 a(m)表示匹配 m 个 a。

12.其中 a(m,)表示匹配至少 m 个 a。

13.其中 a(m,n)表示匹配 m 到 n 个 a。

14.其中 a(,n)表示匹配 0 到 n 个 a。

15.(.....)表示将模式元素组成单一元素。

**********操作**********

1.使用正则表达式$进行匹配：

```
mysql>select name,email from t where email  regexp '@163.com$';
```

2.使用 like 方式查询：

```
mysql>select name,email from t where email like '%@163.com';
```

**********使用 rand()提取随机行**********

1.MySQL 数据库中有一个随机函数 rand()，它能够获取一个 0-1 之间的数。

2.我们可以利用这个函数一起和 order by 能够把数据随机排序。

3.范例：select * from stu order by rand();

**********with rollup**********

1.使用 group by 的 with rollup 子句可以检索出更多的分组聚合信息。

2.不过 with rollup 不可以和 order by 同时使用。

3.范例：select cname，pname，count(pname) from demo group by cname,pname with rollup;

*********用 bit group functions 做统计**********

1.在使用 group by 语句时可以同时使用 bit_and、bit_or 函数来完成统计工作。

2.这两个函数的作用主要时做数值之间的逻辑位运算。

3.对 order_rab 表中 id 分组时对 kind 做位与运算：

select id，bit_or(kind) from order_rab group by id;

4.这些函数只有在使用 group by 进行聚合的时候才有意义。

**********外键**********

1.InnoDB 类型的表支持外键，MyISAM 类型的表虽然创建外键可以成功，但是不起作用，主要原因还是不支持外键。

2.创建外键的范例如下：

create table temp(id int,name char(20),foreign key(id) references outtable(id) on delete cascade on update cascade)；

**********help 使用**********

1.在 mysql 中那么多的命令如何才能记得住是个问题，这里有一个特别好的获得帮助的好方法。

2.在 mysql>的提示下的操作？ %可以获得所有的 mysql>里的命令，这个是最多的，那么这里的东西如何去进一步获得帮助呢？可以使用？create 这种方式。

3.如果我们记不住 optimize 这个单词，可以使用？opti%来操作。

4.使用？reg%可以获得记不住的 regexp 用法。

5.查看所有使用? contents 可以得到所有的帮助大纲，通过这个目录再用?继续往下细查。

**********SQL 语句优化包含知识点**********

1.优化 SQL 语句

2.索引问题

3.两个简单使用的优化方法

4.常用 SQL 的优化

**********show status**********

1.通过 show status 命令了解各种 SQL 的执行频率。

2.格式：show [session|global] status;

3.其中 session 是默认值，表示当前连接，而 global 表示自数据库启动至今。

4.范例：

mysql>show status;

mysql>show global status;

mysql>show status like 'com_%';

mysql>show global status like 'com_%';

5.其中 show status like 'com_insert%'和 show status like 'com_select'等增删改查是比较重要的。

6.其中 com_xxx 表示每个 xxx 语句执行的次数，比如：

com_select 表示执行 select 操作的次数，一次查询次数只累计加 1

com_update 执行 update 操作的次数

com_insert 执行 insert 操作的次数，对批量插入只算一次

com_delete 执行 delete 操作的次数

7.其中对于 InnoDB 存储引擎来说：

InnoDB_rows_read 执行 select 操作的次数

InnoDB_rows_updated 执行 update 操作的次数

InnoDB_rows_inserted 执行 insert 操作的次数

InnoDB_rows_deleted 执行 delete 操作的次数

8.其他：

connections 表示连接 mysql 的数量

uptime 服务器已经工作的秒数

slow_queries 表示慢查询的次数

9.由于 show status 只是查看状态，因此它的值都无法被修改，只能被查看。

**********执行效率较低的 SQL 语句**********

1.解析执行效率较低的 SQL 语句，可以使用 explain 或者 desc。

2.通过 explain 来分析执行低效的 SQL 语句的时候，比较关注的就是索引的使用情况和影响行数。

3.对于 select_type 表示查询的类型，simple 表示简单查询，即不使用表连接或者子查询，primary 表示主查询，即外层的查询，union 表示 union 中的第二个或者后面的查询语句，subquery 表示子查询的第一个 select 等等。

4.其中的 type 表示表的检索性能，性能从好到差分别是：system(表仅一行)、const(只一行匹配)、eq_ref(对于前面的每一行使用主键和唯一)、ref(同 eq_ref，但是没有使用主键和唯一)、ref_or_null(同前面对 null 查询)、index_merge(索引合并优化)、unique_subquery(主键子查询)、index_subquery(非主键子查询)、range(表单中的范围查询)、index(都通过查询索引来得到数据)、all(通过全表扫描得到的数据)。

5.而 possible_keys 表示查询时可能使用的索引，key 表示实际使用到的索引，key_len 表示索引字段的长度。rows 表示扫描行的数量，而 extra 表示执行情况的说明和描述。

第六节：索引优化

**********索引**********

1.索引是数据库优化中最常见也是最重要的手段之一，通过索引通常可以帮助用户解决大多数的 SQL 性能问题。

2》MyISAM 存储引擎的表的数据和索引是自动分开存储的，各自是独立的一个文件。而 InnoDB 存储引擎的表的数据和索引是存储在同一个表空间里面的，但是可以有多个文件组成。

3.MySQL 目前不支持函数索引，但是能对列的前面某一部分进行索引，比如 name 字段，可以只取 name 的前四个字符进行索引，这个特性可以大大缩小索引文件的大小，用户也可以设计表结构的时候也可以对文本列根据此特性进行灵活设计。

4.范例：

```
create index ind_name on t1(name(4));
```

其中 t1 是表名，而 ind_name 是索引名。

**********使用索引**********

1.索引用于快速找出在某个列中有一特定值的行。

2.对相关列使用索引是提高 select 操作性能的最佳途径。

3.查看某个表的索引范例：show index from t1;

**********创建索引**********

1.对于创建的多列索引，只要查询的条件中用到最左边的列，索引一般就会被用到。

2.比如创建一个复合索引，范例：

```
create index ind_co_mon on t1(company,money);
```

3.当我们按 company 进行查询的时候，发现用到了复合索引：

```
explain select * from t1 where company = 2006 \G
```

4.当我们按 money 进行查询的时候就没有使用索引：

```
explain select * from t1 where money=1\G
```

**********小提示**********

1.使用 like 的查询，后面如果是常量并且只有%号不在第一个字符，索引才可能会被使用。

2.范例：

explain select * from t1 where name like '%3'\G

3.如果对大的文本进行搜索，使用全文索引而不使用 like '%...%'

4.如果列名是索引，使用 column_name is null 将使用索引，is not null 也会使用索引。

**********何时使用索引**********

1.如果 MySQL 估计使用索引比全表扫描更慢，则不使用索引。

2.比如如果列 key_part1 均匀分布在 1 到 100 之间，如下查询使用索引就不是很好：

select * from t1 where key_part1 >1 and key_part <90;

3.如果使用 memory/heap 表并且 where 条件中不使用“=”进行索引列，那么不会用到索引。

4.heap 表只有在"="的条件下会使用索引。

5.用 or 分隔开的条件，如果 or 前的条件中的列有索引，而后面的列中没有索引，那么设计的索引都不会被用到。

6.通过 5 我们知道也就是我们的 or 里面最好同时加索引。

7.如果不是索引列的第一部分，那么在查询的时候是不会使用索引的。

8.如果 like 是以%开始，可见虽然在 name 上面建有索引，但是由于提交呢中 like 的值的%在第一位，那么 MySQL 也不会采用这个索引。

9.如果列类型是字符串，但是在查询时把一个数值型常量赋值给了一个字符型的列名 name，那么虽然在 name 列上有索引，但是也不会用到。范例：explain select * from t1 where name=14\G

**********查看索引使用情况**********

1.如果索引正在工作，handler_read_key 的值将很高，这个值代表一个行被索引读的次数。

2.而 handler_read_rnd_next 的值高则意味着查询运行低效，并且应该建立索引补救。

3.查看操作的范例：

```
show status like 'handler_read%';
```

**********优化方法**********

1.对于大多数开发人员来说，可能只希望掌握一些简单使用的优化方法，对于更多更高效的优化，更倾向于交给专业 dba 来做。

2.定期分析表和检查表：

```
analyze [local|no_write_to_binlog] table 表名 1,表名 2....
```

3.上述语句用于分析和存储表的关键字分布，分析的结果将可以使得系统得到准确的统计信息，使得 SQL 能够生成正确的行计划。

4.分析表的语法如下(检查一个或多个表是否有错误):

```
check table 表名 1,表名 2.... [option] ... option
={quick|fast|medium|extended|changed}
```

**********实例操作**********

1.首先我们创建一个表 t1.

2.然后我们创建一个视图：

```
create view t2 as select * from t1;
```

3.然后我们用 t1 复制得到 t3：

```
create table t3 like t1;
insert into t3 select * from t1;
```

4.然后我们 drop table t1 删掉 t1 表，之后我们用 check table t2；发现此时报错。

5.然后我们 rename table t3 to t1；之后，再次 check table t2 就没问题了。

**********定期优化表**********

1.优化表的语法格式：

optimize [local|no_write_to_binlog] table 表名 1，表名 2....

2.如果已经删除了表的一大部分，或者如果已经对含有可变长度行的表进行了很多的改动，则需要做定期优化。

3.这个命令可以将表中的控件碎片进行合并，但是此命令只对 MyISAM、BDB、InnoDB 表起作用。

4.范例：optimize table t1;

**********大量导入导出数据**********

1.可以在 mysql 命令行中使用? outfile 来查看帮助手册。

2.导出文件范例：

select * from t1 into outfile '/tmp/t1.txt';

3.在 win 下范例：

select * form t1 into outfile 'D:/xin.txt';

4.导入文件范例：

load data infile "/tmp/xin.txt' into table t1(name);

5.如果不记得格式可以用? infile 来操作。

**********对于 MyISAM **********

1.对于大批量插入数据，使用 load 命令导入数据的时候，适当设置可以提高导入的速度。

2.对于 MyISAM 存储引擎的表，可以通过以下方式快速的导入大量的数据：

alter table 表名 disable keys

loading the data

alter table 表名 enable keys

3.其中 disable keys 和 enable keys 用来打开或关闭 MyISAM 表非唯一索引的更新，可以提高速度，不过，它对 InnoDB 表无效。

4.如果有唯一索引的时候，导入数据不建议关闭该功能，因为之后再恢复索引的时候可能会很麻烦。

**********对于 InnoDB**********

1.因为 InnoDB 表是按照主键顺序保存的，所以将导入的数据主键的顺序排列，可以有效的提高导入数据的效率。

2.比如对于 load data infile 'xxx.xxx' into table t1;如果该文件是按照 t1 的主键存储顺序保存的，那么导入速度会高一些。

3.在导入数据前线执行 set autocommit=0 来关闭自动提交事务，在导入结束后执行 set autocommit=1 来恢复自动提交，可以提高导入效率。

4. 关闭唯一性校验可以提高导入效率，在导入数据前线执行 set unique_checks=0，关闭唯一性校验，在导入数据后执行 set unique_checks=1 来恢复唯一性校验，可以提高导入效率，此时我们一定要注意原来的数据中不会冲突。

**********优化 insert 语句**********

1.尽量使用多个值表的 insert 语句，这样可以大大缩短客户与数据库的连接、关闭等损耗。

2.可以使用 insert delayed(延迟执行)语句来得到更高的效率。

3.将索引文件和数据文件分别存放到不同的磁盘上。

4.可以增加 bulk_insert_buffer_size 变量值的方法来提高速度，但是只对 MyISAM 表使用。

5.当从一个文件中装载一个表是，使用 load data infile 通常比使用很多 insert 语句要快 20 倍。

第七节：数据库与服务器优化

**********优化 group by 语句**********

1.如果查询包含 group by 但用户想要避免排序结果的损耗，则可以使用 order by null 来禁止排序。

2.因为默认使用 group by 的时候会进行升序排序，这点可以从 explain 的 extra 里面看到，它会多一个 using filesort。

**********优化多表查询**********

1.比如我们在 t1 表中查询 id 在 t2 表中的 id 的数据：

```
select * from t1 where id in (select uid from t2);
```

2.我们假设 t1 的 id 和 t2 的 uid 都建立了索引，但是上面查询语句的问题在于使用 explain 可以看到外层的查询不会使用索引，type 类型为 all。

3.此时我们可以优化为：

```
select t1.* from t1,t2 where t1.id = t2.uid;
```

4.此时上面的语句就会使用两个表的索引了。

5.因此一般使用连接查询来优化嵌套查询是常规的优化思路，因为嵌套查询的外层查询是无法使用索引的。

6.使用连接来优化子查询可以看出查询扫描的记录范围和使用索引的情况都有了很大的改善，连接之所以会更有效率一些，是因为 mysql 不需要在内存中创建临时表来完成这个逻辑上需要两个步骤的查询工作。

**********数据库优化**********

1.优化表的类型

2.通过拆分提高表的访问效率

3.使用中间表提高统计查询速度

**********说明**********

一般来说，中间表建议用视图

**********大存储量解决**********

1.通常解决方案：分库分表和分区。

2.主要目的：

(1)减少表的记录数。

(2)减少对操作系统的负担压力。

**********分析**********

1.在 MySQL 中，可以使用函数 procedure analyse()来对当前应用的表进行分析。

2.范例：

```
select * from t1 procedure analyse()\G
```

**********读锁定**********

1.加读锁的范例语句：

```
locak table t1 read
```

2.开启另一个 mysql 连接终端，接着去尝试：

```
select * from t1
```

3.再 insert、update 和 delete t1 这张表，你会发现所有的数据都停留在终端上，并没有真正的去操作。

4.读锁定对我们在做备份大量数据的时候非常有用，比如：

```
mysqldump -uroot -p111 test >test.sql
```

5.读锁是其他终端可以读，但是不可以增删改，写锁是其他终端也不可以读，也不可以增删改。

**********写锁定**********

1.加写锁的范例：

```
lock table t1 write
```

2.打开另一个 mysql 中断，尝试去 select、insert、update 和 delete 这张表 t1，发现都不能操作，都会停留在终端上。

3.只有等待第一个终端操作完成，第二个终端才能真正执行。

4.表的写锁定比读锁定更加严格。

5.一般情况下哦我们很少显式的去对表进行 read 和 write 锁定的，MyISAM 会自动进行锁定。

**********解锁**********

1.可以看到写锁是比读锁更加严格的锁定。

2.我们使用 unlock tables 来解锁。

**********服务器优化**********

1.四种字符集问题

2.binary log 日志问题

3.slow log 慢查询日志问题

4.socket 问题

5.root 密码丢失

**********字符集问题**********

1.这四种字符集为：

服务器字符集、数据库字符集、客户端字符集、连接字符集

2.要设置它们，在 mysql 的配置文件中，在 client 段中，default-character-set 控制客户端字符集和连接字符集，可以设置为 utf8，在 mysqld 段中，character-set-server 控制服务器字符集和数据库字符集以及继承下来的表字符集，collation-server 是校验字符集，可以设置为 utf8_general_ci，它通常用于排序。

3.可以在 mysql 终端使用 show character set 来查看所有的字符集。

**********binlog 日志**********

1.可以使用 show variables like '%bin%'来查看二进制日志的状况。

2.它默认是开启的。

3.可以在配置文件中进行设置，即 log-bin=文件名前缀

**********慢查询**********

1.可以用 show variables like '%slow%'来查看慢查询日志的状况。

2.可以用 show variables like '%long%'来查看慢查询时间。

3.开启慢查询日志，在 my.cnf 中的 mysqld 段中配置即可：

log_slow_queries = slow.log

long_query_time = 5

**********socket 问题**********

1.这个 socket 默认保存在/tmp/mysql.sock 下。

2.有时候登录 mysql 的时候提示不能用 socket 登录，此时可以换成 tcp 方式去登录，但是可以测试时可以这样用，但是必须要在 php 去用之前把这个事情解决了。

3.操作如下：

mysql -uroot -p111 --protocol tcp -hlocalhost

4.这样就可以登录，这样就不用 mysql.sock 来登录，而 mysql.sock 是启动 mysqld 服务时产生的。

5.这只是临时救济用的，我们重新启动 mysql 就会重新拥有该 sock 文件。

**********root 密码丢失**********

1.首先 service mysqld stop。

2.然后 mysqld_safe --skip-grant-tables --user=mysql &

也就是跳过授权表 mysql.user 和 mysql.db 这些表

3.然后 mysql -uroot

4.如果使用 set password =password("wei")；是会报错的，因为我们使用了--skip-grant-tables。

5.我们使用 update user set password=password('wei') where user = 'root' and host='localhost';

6.或者使用 set password for root@localhost = password('wei');