MySQL常见问题总结

001 数据库应用系统设计

1.规划
2.需求分析
3.概念模型设计
4.逻辑设计
5.物理设计
6.程序编制及调试
7.运行及维护。

002 创建数据库

CREATE DATEBASE database_name

003 查看数据库

SHOW DATABASE

004 选择数据库

USE database_name

005 删除数据库

DORP DATABASE database_name

006 查看支持的引擎

SHOW ENGINES;

007 查看默认支持的存储引擎

SHOW VARIABLES LIKE ‘storage_engine%’

008 创建表

```
CRATE TABLE table_name(
    Field_name data_type,
    Field_name data_type,
    ...
    Field_name data_type
)
```

009 查看表定义

DESC[RIBE] table_name

010 删除表

DROP TABLE table_name

011 修改表

ALTER TABLE old_table_name RENAME [TO] new_table_name

012 增加字段

ALTER TABLE table_name ADD field_name data_type

013 查看表详详细定义

SHOW CREATE TABLE table_name

014 表的第一个位置增加字段

ALTER TABLE table_name ADD field_name data_type FIRST

015 表的指定位置之后增加字段

ALTER TABLE table_name ADD field_name data_type AFTER field_name

016 删除字段

ALTER TABLE table_name DORP field_name

017 修改字段

ALTER TABLE table_name MODIFY field_name data_type

018 修改字段名字

ALTER TABLE table_name CHANGE old_field_name new_field_name old_data_type

019 同时修改字段的名字和属性

ALTER TABLE table_name CHANGE old_field_name new_field_name new_data_type

020 修改字段的顺序

ALTER TABLE table_name MODIFY field_name*1 data_type [FIRST]|[AFTER field_name*2]

021 字段非空约束

Field_name data_type NOT NULL

022 字段默认值

Field_name data_type DEFAULT default_value

023 设置唯一约束

1、Field_name data_type UNIQUE
2、CONSTRAINT constraint_name UNIQUE(field_name)

024 主键约束

Field_name data_type PRIMARY KEY

025 多字段主键

CONSTRAINT constraint_name PRIMARY KEY (field_name*1, field_name*2, ...)

026 字段值自动增加

Field_name data_type AUTO_INCREMENT

027 设置外键约束

CONSTRAINT constraint_name FOREIGN KEY(filed_name)
REFERENCES other_table_name(other_field_name)

028 创建表时创建普通索引

[表示可先项]
|表示选择
table_name(
column_name
INDEX|KEY index_name
)

029 在已经存在的表上创建索引

1、CREATE INDEX index_name ON table_name (
field_name [(index_length)][ASC|DESC])
2、ALTER TABLE table_name ADD INDEX|KEY
index_name(field_name [(index_length)][ASC|DESC])

030 创建表时创建唯一索引

```
table_name(
    column_name
    UNIQUE INDEX|KEY [index_name](
        field_name [(index_length)][ASC|DESC])
)
```

031 在已经存在的表上创建唯一索引

```
1、CREATE UNIQUE INDEX index_name ON table_name (
    field_name [(index_length)][ASC|DESC])
2、ALTER TABLE table_name ADD UNIQUE INDEX|KEY
    index_name(field_name [(index_length)][ASC|DESC])
```

032 创建表时创建全文索引

```
table_name(
    column_name
    FULLTEXT INDEX|KEY [index_name](
        field_name [(index_length)][ASC|DESC])
)
```

033 在已经存在的表上创建全文索引

```
1、CREATE FULLTEXT INDEX index_name ON table_name (
    field_name [(index_length)][ASC|DESC])
2、ALTER TABLE table_name ADD FULLTEXT INDEX|KEY
    index_name(field_name [(index_length)][ASC|DESC])
```

034 创建多列索引

```
和上面的方法类似
    index_name(field_name_1 [(index_length)][ASC|DESC],
    ... ,
    field_name_n [(index_length)][ASC|DESC])
```

035 删除索引

```
DROP INDEX index_name ON table_name
```

036 创建视图

```
CREATE VIEW view_name AS select_query
```

037 查看视图信息

```
SHOW TABLE STATUS [FROM database_name][LIKE 'pattern']
```

038 查看视图设计信息

DESC[RIBE] view_name

039 删除视图

DROP VIEW view_name[, view_name]

040 修改视图

1、CREATE OR REPLACE VIEW view_name AS select_query
2、ALTER VIEW view_name AS select_query

041 创建触发器

CREATE TIRGGER trigger_name
BEFORE|AFTER DELETE|INSERT|UPDATE
ON table_name FOR EACH ROW
Triggle_statement
Triggle_statement:触发器被触发要执行的语句（增、删、改、查等等）

042 查看触发器

SHOW TRIGGERS

043 删除触发器

DROP TRIGGER trigger_name

044 插入数据

INSERT INTO table_name (field*1, field*2, ...) VALUES (value*1, value*2, vaule_3, ...)

045 查看Mysql表结构的命令，如下:

desc 表名;
show columns from 表名;
describe 表名;
show create table 表名;
use information_schema
select * from columns where table_name='表名';

046 数据库分页查询

select * from userdetail where userid limit 0,20

047 MySQL数据库引擎种类

（MySQL-5.5.5开始,InnoDB作为默认存储引擎）之前是MyISAM，更早是ISAM你能用的数据库引擎取决于mysql在安装的时候是如何被编译的。要添加一个新的引擎，就必须重新编译MYSQL。在缺省情况下，MYSQL支持三个引擎：ISAM、MYISAM和HEAP。另外两种类型INNODB和BERKLEYDB（BDB）， 也常常可以使用。

ISAM是一个定义明确且历经时间考验的数据表格管理方法，它在设计之时就考虑到数据库被查询的次数要远大于更新的次数。因此，ISAM执行读取操作的速度很快，而且不占用大量的内存和存储资源。ISAM的两个主要不足之处在于，它不支持事务处理，也不能够容错：如果你的硬盘崩溃了，那么数据文件就无法恢复了。如果你正在把ISAM用在关键任务应用程序里，那就必须经常备份你所有的实时数据，通过其复制特性，MYSQL能够支持这样的备份应用 程序。

MYISAM是MYSQL的ISAM扩展格式和缺省的数据库引擎(5.5之前)。除了提供ISAM里所没有的索引和字段管理的大 量功能，MYISAM还使用一种表格锁定的机制，来优化多个并发的读写操作。其代价是你需要经常运行OPTIMIZE TABLE命令，来恢复被更新机制所浪费的空间。MYISAM还有一些有用的扩展，例如用来修复数据库文件的 MYISAMCHK工具和用来恢复浪费空间的MYISAMPACK工具。

MYISAM强调了快速读取操作，这可能就是为什么MYSQL受到了WEB开发如此青睐的主要原因：在WEB开发中你所进行的大量数据操作都是读取操作。所以，大多数虚拟主机提供商和INTERNET平台提供商只允许使用MYISAM格式。

HEAP允许只驻留在内存里的临时表格。驻留在内存使得HEAP比ISAM和MYISAM的速度都快，但是它所管理的数据是不稳定的，而且如果在关机之前没有进行保存，那么所有的数据都会丢失。在数据行被删除的时候，HEAP也不会浪费大量的空间，HEAP表格在你需要使用SELECT表达式来选择和操控数据的时候非常有用。要记住，用完表格后要删除表格。

INNODB和**BERKLEYDB（BDB）**数据库引擎都是造就MYSQL灵活性的技术的直接产品，这项技术就是MySql++ API。在使用MySql的时候，你所面对的每一个挑战几乎都源于ISAM和MYIASM数据库引擎不支持事务处理也不支持外来键。尽管要比ISAM和MYISAM引擎慢很多，但是INNODB和BDB包括了对事务处理和外来键的支持，这两点都是前两个引擎所没有的。如前所述，如果你的设计需要这些特性中的一者或者两者，那你就要被迫使用后两个引擎中的一个了。

048 MySQL锁类型

根据**锁的类型**分，可以分为**共享锁，排他锁，意向共享锁**和**意向排他锁**。

根据**锁的粒度**分，又可以分为**行锁，表锁**。

对于mysql而言，事务机制更多是靠底层的存储引擎来实现，因此，mysql层面只有表锁，而支持事务的innodb存 储引擎则实现了行锁(记录锁（在行相应的索引记录上的锁）)，gap锁（是在索引记录间歇上的锁），next-key锁（是记录锁和在此索引记录之前的gap上的锁的结合）。Mysql的记录锁实质是索引记录的锁，因为innodb是索引组织表；gap锁是索引记录间隙的锁，这种锁只在RR隔离级别下有效；next-key锁是记录锁加上记录之前gap锁的组合。mysql通过gap锁和next-key锁实现RR隔离级别。

说明：对于更新操作(读不上锁)，只有走索引才可能上行锁；否则会对聚簇索引的每一行上写锁，实际等同于对表上写锁。

若多个物理记录对应同一个索引，若同时访问，也会出现锁冲突；

当表有多个索引时，不同事务可以用不同的索引锁住不同的行，另外innodb会同时用行锁对数据记录(聚簇索引)加 锁。

MVCC(多版本并发控制)并发控制机制下，任何操作都不会阻塞读操作，读操作也不会阻塞任何操作，只因为读不上锁。

共享锁：由读表操作加上的锁，加锁后其他用户只能获取该表或行的共享锁，不能获取排它锁，也就是说只能读不能写

排它锁：由写表操作加上的锁，加锁后其他用户不能获取该表或行的任何锁，典型是mysql事务中的更新操作

意向共享锁（IS）：事务打算给数据行加行共享锁，事务在给一个数据行加共享锁前必须先取得该表的IS锁。

意向排他锁（IX）：事务打算给数据行加行排他锁，事务在给一个数据行加排他锁前必须先取得该表的IX锁。

049 MYSQL支持事务吗？

在缺省模式下，MYSQL是autocommit模式的，所有的数据库更新操作都会即时提交，所以在缺省情况下，mysql是不支持事务的。 但是如果你的MYSQL表类型是使用InnoDB Tables 或 BDB tables的话，你的MYSQL就可以使用事务处理,使用SET AUTOCOMMIT=0就可以使MYSQL允许在非autocommit模式， 在非autocommit模式下，你必须使用COMMIT来提交你的更改，或者用ROLLBACK来回滚你的更改。

示例如下：

```
START TRANSACTION;
    SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
    UPDATE table2 SET summmary=@A WHERE type=1;
COMMIT;
```

050 MYSQL相比于其他数据库有哪些特点？

1、可以处理拥有上千万条记录的大型数据
2、支持常见的SQL语句规范
3、可移植行高，安装简单小巧
4、良好的运行效率，有丰富信息的网络支持
5、调试、管理，优化简单（相对其他大型数据库）

051 如何解决MYSQL数据库中文乱码问题？

1、在数据库安装的时候指定字符集
2、如果在安完了以后可以更改以配置文件
3、建立数据库时候：指定字符集类型
4、建表的时候也指定字符集

052 如何提高MySql的安全性？

1.如果MYSQL客户端和服务器端的连接需要跨越并通过不可信任的网络，那么需要使用ssh隧道来加密该连接的通信。

2.使用set password语句来修改用户的密码，先“mysql -u root”登陆数据库系统，然后mysql> update mysql.user set password=password(‘newpwd’)，最后执行flush privileges就可以了。

3.Mysql需要提防的攻击有，防偷听、篡改、回放、拒绝服务等，不涉及可用性和容错方面。对所有的连接、查询、其他操作使用基于acl即访问控制列表的安全措施来完成。也有一些对ssl连接的支持。

4.设置除了root用户外的其他任何用户不允许访问mysql主数据库中的user表; 加密后存放在user表中的加密 后的用户密码一旦泄露，其他人可以随意用该用户名/密码相应的数据库;

5.使用grant和revoke语句来进行用户访问控制的工作;

6.不要使用明文密码，而是使用md5()和sha1()等单向的哈系函数来设置密码;

7.不要选用字典中的字来做密码;

8.采用防火墙可以去掉50%的外部危险，让数据库系统躲在防火墙后面工作，或放置在dmz区域中;

9.从因特网上用nmap来扫描3306端口，也可用telnet server_host 3306的方法测试，不允许从非信任网络中访问数据库服务器的3306号tcp端口，需要在防火墙或路由器上做设定;

10.为了防止被恶意传入非法参数，例如where id=234，别人却输入where id=234 or 1=1导致全部显示，所以在web的表单中使用”或””来用字符串，在动态url中加入%22代表双引号、%23代表井号、%27代表单引号;传递未检查过的值给mysql数据库是非常危险的;

11.在传递数据给mysql时检查一下大小;

12.应用程序需要连接到数据库应该使用一般的用户帐号，开放少数必要的权限给该用户; pagedevide

13.在各编程接口(c c++ PHP perl Java jdbc等)中使用特定‘逃脱字符’函数; 在因特网上使用mysql数据 库时一定少用传输明文的数据，而用ssl和ssh的加密方式数据来传输;

14.学会使用tcpdump和strings工具来查看传输数据的安全性，例如tcpdump -l -i eth0 -w -src or dst port 3306 strings。以普通用户来启动mysql数据库服务;

15.不使用到表的联结符号，选用的参数 –skip-symbolic-links;

16.确信在mysql目录中只有启动数据库服务的用户才可以对文件有读和写的权限;

17.不许将process或super权限付给非管理用户，该mysqladmin processlist可以列举出当前执行的查询 文本;super权限可用于切断客户端连接、改变服务器运行参数状态、控制拷贝复制数据库的服务器;

18.file权限不付给管理员以外的用户，防止出现load data '/etc/passwd'到表中再用select 显示出来的问题;

19.如果不相信dns服务公司的服务，可以在主机名称允许表中只设置ip数字地址;

20.使用max_user_connections变量来使mysqld服务进程，对一个指定帐户限定连接数;

21.grant语句也支持资源控制选项;

22.–local-infile=0或1 若是0则客户端程序就无法使用local load data了，赋权的一个例子grant insert(user) on mysql.user to 'user_name'@'host_name';若使用–skip-grant-tables系统将对任何用户的访问不做任何访问控制，但可以用 mysqladmin flush-privileges或mysqladmin reload来开启访问控制;默认情况是show databases语句对所有用户开放，可以用–skip-show-databases来关闭掉。

23.碰到error 1045(28000) access denied for user 'root'@'localhost' (using password:no)错误时，你需要重新设置密码，具体方法是:先用–skip-grant-tables参数启动mysqld，然后执行 mysql -u root mysql,mysql>update user set password=password('newpassword') where user='root';mysql>flush privileges;，最后重新启动mysql就可以了。

053 MySQL取得当前时间的函数是?，格式化日期的函数是

取得当前时间用 now() 就行。在数据库中格式化时间 用DATE_FORMAT(date, format)。根据格式串 format 格式化日期或日期时间值date，返回结果串。

054 你如何确定 MySQL 是否处于运行状态？

答案： Debian 上运行命令 service mysql status，在RedHat 上运行命令 service mysqld status。然后看看输出即可。

055 如何开启或停止 MySQL 服务？

答案：运行命令 service mysqld start 开启服务；运行命令 service mysqld stop 停止服务。

056 如何通过 Shell 登入 MySQL？

答案：运行命令 mysql -u root –p

057 如何列出所有数据库？

答案：运行命令 show databases;

058 如何切换到某个数据库并在上面工作？

答案：运行命令 use database_name; 进入名为 database_name 的数据库。

059 如何列出某个数据库内所有表？

答案：在当前数据库运行命令 show tables;

060 如何获取表内所有 Field 对象的名称和类型？

答案：运行命令 describe table_name;

061 如何删除表？

答案：运行命令 drop table table_name;

062 创建索引

对于查询占主要的应用来说，索引显得尤为重要。很多时候性能问题很简单的就是因为我们忘了添加索引而造成的，或者说没有添加更为有效的索引导致。如果不加索引的话，那么查找任何哪怕只是一条特定的数据都会进行一次全表扫描，如果一张表的数据量很大而符合条件的结果又很少，那么不加索引会引起致命的性能下降。但是也不是什么情 况都非得建索引不可，比如性别可能就只有两个值，建索引不仅没什么优势，还会影响到更新速度，这被称为过度索引。

063 复合索引

比如有一条语句是这样的：select * from users where area='beijing' and age=22;

如果我们是在area和age上分别创建单个索引的话，由于MySQL查询每次只能使用一个索引，所以虽然这样已经相对不做索引时全表扫描提高了很多效率，但是如果在area、age两列上创建复合索引的话将带来更高的效率。如果我们创建了(area, age, salary)的复合索引，那么其实相当于创建了(area,age,salary)、 (area,age)、(area)三个索引，这被称为最佳左前缀特性。因此我们在创建复合索引时应该将最常用作限制条件的列放在最左边，依次递减。

064 索引不会包含有NULL值的列

只要列中包含有NULL值都将不会被包含在索引中，复合索引中只要有一列含有NULL值，那么这一列对于此复合索引就是无效的。所以我们在数据库设计时不要让字段的默认值为NULL。

065 使用短索引

对串列进行索引，如果可能应该指定一个前缀长度。例如，如果有一个CHAR(255)的列，如果在前10个或20个字符内，多数值是惟一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作。

066 排序的索引问题

mysql查询只使用一个索引，因此如果where子句中已经使用了索引的话，那么order by中的列是不会使用索引的。因此数据库默认排序可以符合要求的情况下不要使用排序操作；尽量不要包含多个列的排序，如果需要最好给这些列创建复合索引。

067 like语句操作

一般情况下不鼓励使用like操作，如果非使用不可，如何使用也是一个问题。like “%aaa%” 不会使用索引而like “aaa%”可以使用索引。

068 MYSQL数据库设计数据类型选择需要注意哪些地方?

VARCHAR和CHAR类型，varchar是变长的，需要额外的1-2个字节存储，能节约空间，可能会对性能有帮助。但由于是变长，可能发生碎片，如更新数据；

使用ENUM（MySQL的枚举类）代替字符串类型，数据实际存储为整型。

字符串类型

要尽可能地避免使用字符串来做标识符，因为它们占用了很多空间并且通常比整数类型要慢。特别注意不要在MYISAM表上使用字符串标识符。MYISAM默认情况下为字符串使用了压缩索引（Packed Index），这使查找更为缓慢。据测试，使用了压缩索引的MYISAM表性能要慢6倍。

还要特别注意完全'随机'的字符串，例如由MD5（）、SHA1（）、UUID（）产生的。它们产生的每一个新值都会被任意地保存在很大的空间范围内，这会减慢INSERT及一些SELECT查询。1）它们会减慢INSERT查询，因为插入的值会被随机地放入索引中。这会导致分页、随机磁盘访问及聚集存储引擎上的聚集索引碎片。2)它们会减慢SELECT查询，因为逻辑上相邻的行会分布在磁盘和内存中的各个地方。3）随机值导致缓存对所有类型的查询性能都很差，因为它们会使缓存赖以工作的访问局部性失效。如果整个数据集都变得同样"热"的时候，那么把特定部分的数据缓存到内存中就没有任何的优势了。并且如果工作集不能被装入内存中，缓存就会进行很多刷写的工作，并且会导致很多缓存未命中。

如果保存UUID值，就应该移除其中的短横线，更好的办法是使用UHEX（）把UUID值转化为16字节的数字，并把它保存在BINARY（16）列中。

069 不要在列上进行运算

select * from users where YEAR(adddate)<2007;

将在每个行上进行运算，这将导致索引失效而进行全表扫描，因此我们可以改成

select * from users where adddate<'2007-01-01';

不使用NOT IN和操作

NOT IN和操作都不会使用索引将进行全表扫描。NOT IN可以NOT EXISTS代替，id != 3则可使用id>3 or id<3来代替。

070 IS NULL 与 IS NOT NULL

不能用null作索引，任何包含null值的列都将不会被包含在索引中。即使索引有多列这样的情况下，只要这些列中有一列含有null，该列就会从索引中排除。也就是说如果某列存在空值，即使对该列建索引也不会提高性能。

任何在where子句中使用is null或is not null的语句优化器是不允许使用索引的。

071 联接列

对于有联接的列，即使最后的联接值为一个静态值，优化器是不会使用索引的。

072 MySQL几种备份方式（重点）

1、**逻辑备份**：使用mysql自带的mysqldump工具进行备份。备份成sql文件形式。

优点：最大好处是能够与正在运行的mysql自动协同工作，在运行期间可以确保备份是当时的点，它会自动将对应操作的表锁定，不允许其他用户修改(只能访问)。可能会阻止修改操作。sql文件通用方便移植。

缺点：备份的速度比较慢。如果是数据量很多的时候。就很耗时间。如果数据库服务器处在提供给用户服务状态，在这段长时间操作过程中，意味着要锁定表(一般是读锁定，只能读不能写入数据)。那么服务就会影响的。

2、**物理备份**：直接拷贝mysql的数据目录。

直接拷贝只适用于myisam类型的表。这种类型的表是与机器独立的。但实际情况是，你设计数据库的时候不可能全部使用myisam类型表。你也不可能因为myisam类型表与机器独立，方便移植，于是就选择这种表，这并不是选择它的理由。

缺点：你不能去操作正在运行的mysql服务器(在拷贝的过程中有用户通过应用程序访问更新数据，这样就无法备份当时的数据)可能无法移植到其他机器上去。

3、双机热备份。

mysql数据库没有增量备份的机制。当数据量太大的时候备份是一个很大的问题。还好mysql数据库提供了一种主从备份的机制(也就是双机热备)

优点：适合数据量大的时候。现在明白了。大的互联网公司对于mysql数据备份，都是采用热机备份。搭建多台数据库服务器，进行主从复制。

073 想知道一个查询用到了哪个index,如何查看?

explain显示了mysql如何使用索引来处理select语句以及连接表。可以帮助选择更好的索引和写出更优化的查询语句。使用方法，在select语句前加上explain就可以了。所以使用explain可以查看。

074 数据库不能停机，请问如何备份? 如何进行全备份和增量备份?

可以使用逻辑备份和双机热备份。

完全备份：完整备份一般一段时间进行一次，且在网站访问量最小的时候，这样常借助批处理文件定时备份。主要是写一个批处理文件在里面写上处理程序的绝对路径然后把要处理的东西写在后面，即完全备份数据库。

增量备份：对ddl和dml语句进行二进制备份。且5.0无法增量备份，5.1后可以。如果要实现增量备份需要在my.ini文件中配置备份路径即可，重启mysql服务器，增量备份就启动了。

075 MySQL添加索引

普通索引 添加INDEX

ALTER TABLE ‘table_name’ ADD INDEX index_name (‘column’);

主键索引 添加PRIMARY KEY

ALTER TABLE ‘table_name’ ADD PRIMARY KEY (‘column’);

唯一索引 添加UNIQUE

ALTER TABLE ‘table_name’ ADD UNIQUE (‘column’);

全文索引 添加FULLTEXT

ALTER TABLE ‘table_name’ ADD FULLTEXT (‘column’);

多列索引

ALTER TABLE ‘table_name’ ADD INDEX index_name (‘column1’, ‘column2’, ‘column3’)

076 什么情况下使用索引？

表的主关键字

自动建立唯一索引

如zl_yhjbqk（用户基本情况）中的hbs_bh（户标识编号）

表的字段唯一约束

Oracle利用索引来保证数据的完整性

如lc_hj（流程环节）中的lc_bh+hj_sx（流程编号+环节顺序）

直接条件查询的字段

在SQL中用于条件约束的字段

如zl_yhjbqk（用户基本情况）中的qc_bh（区册编号）

select * from zl_yhjbqk where qc_bh='7001'

查询中与其它表关联的字段

字段常常建立了外键关系

如zl_ydcf（用电成份）中的jldb_bh（计量点表编号）

select * from zl_ydcf a,zl_yhdb b where a.jldb_bh=b.jldb_bh and b.jldb_bh='540100214511'

查询中排序的字段

排序的字段如果通过索引去访问那将大大提高排序速度

select * from zl_yhjbqk order by qc_bh（建立qc_bh索引）

select * from zl_yhjbqk where qc_bh='7001' order by cb_sx（建立qc_bh+cb_sx索引，注：只是一个索引，其中包括qc_bh和cb_sx字段）

查询中统计或分组统计的字段

select max(hbs_bh) from zl_yhjbqk

select qc_bh,count(*) from zl_yhjbqk group by qc_bh

077 什么情况下应不建或少建索引

表记录太少

如果一个表只有5条记录，采用索引去访问记录的话，那首先需访问索引表，再通过索引表访问数据表，一般索引表与数据表不在同一个数据块，这种情况下ORACLE至少要往返读取数据块两次。而不用索引的情况下ORACLE会将所有的数据一次读出，处理速度显然会比用索引快。

如表zl_sybm（使用部门）一般只有几条记录，除了主关键字外对任何一个字段建索引都不会产生性能优化，实际上如果对这个表进行了统计分析后ORACLE也不会用你建的索引，而是自动执行全表访问。如：select * from zl_sybm where sydw_bh='5401'（对sydw_bh建立索引不会产生性能优化）

经常插入、删除、修改的表

对一些经常处理的业务表应在查询允许的情况下尽量减少索引，如zl_yhbm，gc_dfss，gc_dfys，gc_fpdy等业务表。

数据重复且分布平均的表字段

假如一个表有10万行记录，有一个字段A只有T和F两种值，且每个值的分布概率大约为50%，那么对这种表A字段建索引一般不会提高数据库的查询速度。

经常和主字段一块查询但主字段索引值比较多的表字段

如gc_dfss（电费实收）表经常按收费序号、户标识编号、抄表日期、电费发生年月、操作 标志来具体查询某一笔收款的情况，如果将所有的字段都建在一个索引里那将会增加数据的修改、插入、删除时间，从实际上分析一笔收款如果按收费序号索引就已 经将记录减少到只有几条，如果再按后面的几个字段索引查询将对性能不产生太大的影 响。

078 千万级MySQL数据库建立索引的事项及提高性能的手段

1.对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

2.应尽量避免在 **where** 子句中对字段进行 **null** 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num is null可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：select id from t where num=0

3.应尽量避免在 **where** 子句中使用**!=**或**<>**操作符，否则引擎将放弃使用索引而进行全表扫描。

4.应尽量避免在 **where** 子句中使用**or** 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num=10 or num=20可以这样查询：select id from t where num=10 union all select id from t where num=20

5.in 和 **not in** 也要慎用，否则会导致全表扫描，如：select id from t where num in(1,2,3) 对于连续的数值，能用 between 就不要用 in 了：select id from t where num between 1 and 3

6.避免使用通配符。下面的查询也将导致全表扫描：select id from t where name like ‘李%’若要提高效率，可以考虑全文检索。

7.如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：select id from t where num=@num 可以改为强制查询使用索引：select id from t with(index(索引名)) where num=@num

8.应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。如：select id from t where num/2=100应改为:select id from t where num=100*2

9.应尽量避免在where子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描。如：select id from t where substring(name,1,3)=’abc’ ，name以abc开头的id应改为:select id from t where name like ‘abc%’

10.不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

11.在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

12.不要写一些没有意义的查询，如需要生成一个空表结构：select col1,col2 into #t from t where 1=0 这类代码不会返回任何结果集，但是会消耗系统资源的，应改成这样：create table #t(...)

13.很多时候用 exists 代替 in 是一个好的选择：select num from a where num in(select num from b)用下面的语句替换：select num from a where exists(select 1 from b where num=a.num)

14.并不是所有索引对查询都有效，SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。

15.索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了insert 及 update 的 效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有 必要。

16.应尽可能的避免更新 clustered 索引数据列，因为 clustered 索引数据列的顺序就是表记录的物理存储 顺序，一旦该列值改变将导致整个表记录的顺序的调整，会耗费相当大的资源。若应用系统需要频繁更新 clustered 索引数据列，那么需要考虑是否应将该索引建为 clustered 索引。

17.尽量使用数字型字段，若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

18.尽可能的使用 varchar/nvarchar 代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

19.任何地方都不要使用 select * from t ，用具体的字段列表代替“*”，不要返回用不到的任何字段。

20.尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

21.避免频繁创建和删除临时表，以减少系统表资源的消耗。

22.临时表并不是不可使用，适当地使用它们可以使某些例程更有效，例如，当需要重复引用大型表或常用表中的某个数据集时。但是，对于一次性事件，最好使用导出表。

23.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

24.如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

25.尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

26.使用基于游标的方法或临时表方法之前，应先寻找基于集的解决方案来解决问题，基于集的方法通常更有效。

27.与临时表一样，游标并不是不可使用。对小型数据集使用 FAST_FORWARD 游标通常要优于其他逐行处理方法，尤其是在必须引用几个表才能获得所需的数据时。在结果集中包括“合计”的例程通常要比使用游标执行的速度快。如果开发时间允许，基于游标的方法和基于集的方法都可以尝试一下，看哪一种方法的效果更好。

28.在所有的存储过程和触发器的开始处设置 **SET NOCOUNT ON** ，在结束时设置 **SET NOCOUNT OFF**。无需在执行存储过程和触发器的每个语句后向客户端发送DONE_IN_PROC 消息。

29.尽量避免大事务操作，提高系统并发能力。

30.尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

079 MySql在建立索引优化时需要注意的问题

1，创建索引

对于查询占主要的应用来说，索引显得尤为重要。很多时候性能问题很简单的就是因为我们忘了添加索引而造成的，或者说没有添加更为有效的索引导致。如果不加索引的话，那么查找任何哪怕只是一条特定的数据都会进行一次全表扫描，如果一张表的数据量很大而符合条件的结果又很少，那么不加索引会引起致命的性能下降。但是也不是什么情况都非得建索引不可，比如性别可能就只有两个值，建索引不仅没什么优势，还会影响到更新速度，这被称为过度索引。

2，复合索引

比如有一条语句是这样的：select * from users where area='beijing' and age=22;

如果我们是在area和age上分别创建单个索引的话，由于mysql查询每次只能使用一个索引，所以虽然这样已经相对不做索引时全表扫描提高了很多效率，但是如果在area、age两列上创建复合索引的话将带来更高的效率。如果我们创建了(area, age,salary)的复合索引，那么其实相当于创建了(area,age,salary)、(area,age)、(area)三个索引，这被称为最佳左前缀特性。因此我们在创建复合索引时应该将最常用作限制条件的列放在最左边，依次递减。

3，索引不会包含有**NULL**值的列

只要列中包含有NULL值都将不会被包含在索引中，复合索引中只要有一列含有NULL值，那么这一列对于此复合索引就是无效的。所以我们在数据库设计时不要让字段的默认值为NULL。

4，使用短索引

对串列进行索引，如果可能应该指定一个前缀长度。例如，如果有一个CHAR(255)的 列，如果在前10 个或20 个字符内，多数值是惟一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作。

5，排序的索引问题

mysql查询只使用一个索引，因此如果where子句中已经使用了索引的话，那么order by中的列是不会使用索引的。因此数据库默认排序可以符合要求的情况下不要使用排序操作；尽量不要包含多个列的排序，如果需要最好给这些列创建复合索引。

6，like语句操作

一般情况下不鼓励使用like操作，如果非使用不可，如何使用也是一个问题。like "%aaa%" 不会使用索引而like "aaa%"可以使用索引。

7，不要在列上进行运算

select * from users where YEAR(adddate)

8，不使用**NOT IN**和操作

NOT IN和操作都不会使用索引将进行全表扫描。NOT IN可以NOT EXISTS代替，id != 3则可使用id>3 or id < 3

080 数据库性能下降，想找到哪些sql耗时较长，应该如何操作？my.cnf里如何配置？

1、show processlist;

2、select * from information_schema.processlist ;

3、可以在[mysqld]中添加如下：

log =/var/log/mysql.log

如果需要监控慢查询可以添加如下内容：

log-slow-queries = /var/log/slowquery.log

long_query_time = 1

081 聚集索引

术语“聚集”指实际的数据行和相关的键值都保存在一起。每个表只能有一个聚集索引。但是，覆盖索引可以模拟多个聚集索引。存储引擎负责实现索引，因此不是所有的存储索引都支持聚集索引。当前，SolidDB和InnoDB是唯一支持聚集索引的存储引擎。

优点：

可以把相关数据保存在一起。这样从磁盘上提取几个页面的数据就能把某个用户的数据全部抓取出来。如果没有使用聚集，读取每个数据都会访问磁盘。

数据访问快。聚集索引把索引和数据都保存到了同一棵B-TREE中，因此从聚集索引中取得数据通常比在非聚集索引进行查找要快。

缺点：

聚集能最大限度地提升I/O密集负载的性能。如果数据能装入内存，那么其顺序也就无所谓了。这样聚集就没有什么用处。

插入速度严重依赖于插入顺序。更新聚集索引列是昂贵的，因为强制InnoDB把每个更新的行移到新的位置。

建立在聚集索引上的表在插入新行，或者在行的主键被更新，该行必须被移动的时候会进行分页。

聚集表可会比全表扫描慢，尤其在表存储得比较稀疏或因为分页而没有顺序存储的时候。

第二（非聚集）索引可能会比预想的大，因为它们的叶子节点包含了被引用行的主键列。第二索引访问需要两次索引查找，而不是一次。 InnoDB的第二索引叶子节点包含了主键值作为指向行的“指针”，而不是“行指针”。 这种策略减少了在移动行或数据分页的时候索引的维护工作。使用行的主键值作为指针使得索引变得更大，但是这意味着InnoDB可以移动行，而无须更新指针。

082 索引类型

索引类型: B-TREE索引，哈希索引

B-TREE索引(默认的索引类型)加速了数据访问，因为存储引擎不会扫描整个表得到需要的数据。相反，它从根节点开始。根节点保存了指向子节点的指针，并且存储引擎会根据指针寻找数据。它通过查找节点页中的值找到正确的指针，节点页包含子节点的指针，并且存储引擎会根据指针寻找数据。它通过查找节点页中的值找到正确的指针，节点页包含子节点中值的上界和下界。最后，存储引擎可能无法找到需要的数据，也可能成功地找到包含数据的叶子页面。

例：B-TREE索引 对于以下类型查询有用。匹配全名、匹配最左前缀、匹配列前缀、匹配范围值、精确匹配一部分并且匹配某个范围中的另一部分；

B-TREE索引的局限：如果查找没有从索引列的最左边开始，它就没什么用处。不能跳过索引中的列，存储引擎不能优先访问任何在第一个范围条件右边的列。例：如果查询是where last_name=‘Smith’ AND first_name LIKE ‘J%’ AND dob=‘1976-12-23’;访问就只能使用索引的头两列，因为LIKE是范围条件。

哈希索引建立在哈希表的基础上，它只对使用了索引中的每一列的精确查找有用。对于每一行，存储引擎计算出了被索引列的哈希码，它是一个较小的值，并且有可能和其他行的哈希码不同。它把哈希码保存在索引中，并且保存了一个指向哈希表中每一行的指针。

因为索引只包含了哈希码和行指针，而不是值自身，MYSQL不能使用索引中的值来避免读取行。

MYSQL不能使用哈希索引进行排序，因为它们不会按序保存行。

哈希索引不支持部分键匹配，因为它们是由被索引的全部值计算出来的。也就是说，如果在（A，B）两列上有索引，并且WHERE子句中只使用了A，那么索引就不会起作用。

哈希索引只支持使用了= IN（）和<=>的相等比较。它们不能加快范围查询。例如WHERE price > 100;

访问哈希索引中的数据非常快，除非碰撞率很高。当发生碰撞的时候，存储引擎必须访问链表中的每一个行指针，然后逐行进行数据比较，以确定正确的数据。如果有很多碰撞，一些索引维护操作就有可能会变慢。

083 FULLTEXT全文索引

即为全文索引，目前只有MyISAM引擎支持。其可以在CREATE TABLE ，ALTER TABLE ，CREATE INDEX 使用，不过目前只有 CHAR、VARCHAR ，TEXT 列上可以创建全文索引。值得一提的是，在数据量较大时候，现将数据放入一个没有全局索引的表中，然后再用CREATE INDEX创建FULLTEXT索引，要比先为一张表建立FULLTEXT然后再将数据写入的速度快很多。FULLTEXT索引也是按照分词原理建立索引的。西文中，大部分为字母文字，分词可以很方便的按照空格进行分割。中文不能按照这种方式进行分词。Mysql的中文分词插件Mysqlcft，有了它，就可以对中文进行分词。

084 介绍一下如何优化MySql

一、在编译时优化**MySQL**

如果你从源代码分发安装MySQL，要注意，编译过程对以后的目标程序性能有重要的影响，不同的编译方式可能得到类似的目标文件，但性能可能相差很大，因此，在编译安装MySQL适应仔细根据你的应用类型选择最可能好的编译选项。这种定制的MySQL可以为你的应用提供最佳性能。

技巧：选用较好的编译器和较好的编译器选项，这样应用可提高性能10-30%。（MySQL文档如是说）

1.1、使用pgcc（Pentium GCC)编译器->（使用合适的编译器）

该编译器（http://www.goof.com/pcg/）针对运行在奔腾处理器系统上的程序进行优化，用pgcc编译MySQL源代码，总体性能可提高10%。当然如果你的服务器不是用奔腾处理器，就不必用它了，因为它是专为奔腾系统设计的。

1.2、仅使用你想使用的字符集编译MySQL

MySQL目前提供多达24种不同的字符集，为全球用户以他们自己的语言插入或查看表中的数据。却省情况下，MySQL安装所有者这些字符集，热然而，最好的选择是指选择一种你需要的。如，禁止除Latin1字符集以外的所有其它字符集：

1.3、将mysqld编译成静态执行文件

将mysqld编译成静态执行文件而无需共享库也能获得更好的性能。通过在配置时指定下列选项，可静态编译mysqld。

1.4、配置样本

二、调整服务器

三、表类型（**MySQL**中表的类型）

很多MySQL用户可能很惊讶，**MySQL确实为用户提供5种不同的表类型，称为DBD、HEAP、ISAM、MERGE和MyIASM。DBD归为事务安全类，而其他为非事务安全类。**

3.1、事务安全

DBD

Berkeley DB(DBD)表是支持事务处理的表，它提供MySQL用户期待已久的功能-事务控制。事务控制在任何数据库系统中都是一个极有价值的功能，因为它们确保一组命令能成功地执行。

3.2、非事务安全

HEAP

HEAP表是MySQL中存取数据最快的表。这是因为他们使用存储在动态内存中的一个哈希索引。另一个要点是如果MySQL或服务器崩溃，数据将丢失。

ISAM

ISAM表是早期MySQL版本的缺省表类型，直到MyIASM开发出来。建议不要再使用它。

MERGE

MERGE是一个有趣的新类型，在3.23.25之后出现。一个MERGE表实际上是一个相同MyISAM表的集合，合并成一个表，主要是为了效率原因。这样可以提高速度、搜索效率、修复效率并节省磁盘空间。

MyIASM

这是MySQL的缺省表类型(5.5.5之前)。它基于IASM代码，但有很多有用的扩展。MyIASM比较好的原因：

MyIASM表小于IASM表，所以使用较少资源。

MyIASM表在不同的平台上二进制层可移植。

更大的键码尺寸，更大的键码上限。

3.3、指定表类型

四、优化工具

MySQL服务器本身提供了几条内置命令用于帮助优化。

4.1、**SHOW**

SHOW还能做更多的事情。它可以显示关于日志文件、特定数据库、表、索引、进程和权限表中有价值的信息。

4.2、**EXPLAIN**

当你面对SELECT语句时，EXPLAIN解释SELECT命令如何被处理。这不仅对决定是否应该增加一个索引，而且对决定一个复杂的Join如何被MySQL处理都是有帮助的。

4.3、**OPTIMIZE**

OPTIMIZE语句允许你恢复空间和合并数据文件碎片，对包含变长行的表进行了大量更新和删除后，这样做特别重要。OPTIMIZE目前只工作于MyIASM和BDB表。

085 MySQL你都修改了那些配置文件来进行优化(问配置文件中具体修改的内容)?

innodb_buffer_pool_size:这是你安装完InnoDB后第一个应该设置的选项。缓冲池是数据和索引缓存的地方：这个值越大越好，这能保证你在大多数的读取操作时使用的是内存而不是硬盘。典型的值是5-6GB(8GB内存)，20-25GB(32GB内存)，100-120GB(128GB内存)。

innodb_log_file_size：这是redo日志的大小。redo日志被用于确保写操作快速而可靠并且在崩溃时恢复。一直到MySQL 5.1，它都难于调整，因为一方面你想让它更大来提高性能，另一方面你想让它更小来使得崩溃后更快恢复。幸运的是从MySQL 5.5之后，崩溃恢复的性能的到了很大提升，这样你就可以同时拥有较高的写入性能和崩溃恢复性能了。一直到MySQL 5.5，redo日志的总尺寸被限定在4GB(默认可以有2个log文件)。这在MySQL 5.6里被提高。

一开始就把innodb_log_file_size设置成512M(这样有1GB的redo日志)会使你有充裕的写操作空间。如果你知道你的应用程序需要频繁的写入数据并且你使用的时MySQL 5.6，你可以一开始就把它这是成4G。

max_connections:如果你经常看到'Too many connections'错误，是因为max_connections的值太低了。这非常常见因为应用程序没有正确的关闭数据库连接，你需要比默认的151连接数更大的值。max_connection值被设高了(例如1000或更高)之后一个主要缺陷是当服务器运行1000个或更高的活动事务时会变的没有响应。在应用程序里使用连接池或者在MySQL里使用进程池有助于解决这一问题。

InnoDB配置

从MySQL 5.5版本开始，InnoDB就是默认的存储引擎并且它比任何其他存储引擎的使用都要多得多。那也是为什么它需要小心配置的原因。

innodb_file_per_table：这项设置告知InnoDB是否需要将所有表的数据和索引存放在共享表空间里（innodb_file_per_table = OFF）或者为每张表的数据单独放在一个.ibd文件（innodb_file_per_table = ON）。每张表一个文件允许你在drop、truncate或者rebuild表时回收磁盘空间。这对于一些高级特性也是有必要的，比如数据压缩。但是它不会带来任何性能收益。你不想让每张表一个文件的主要场景是：有非常多的表（比如10k+）。

MySQL 5.6中，这个属性默认值是ON，因此大部分情况下你什么都不需要做。对于之前的版本你必需在加载数据之前将这个属性设置为ON，因为它只对新创建的表有影响。

innodb_flush_log_at_trx_commit：默认值为1，表示InnoDB完全支持ACID特性。当你的主要关注点是数据安全的时候这个值是最合适的，比如在一个主节点上。但是对于磁盘（读写）速度较慢的系统，它会带来很巨大的开销，因为每次将改变flush到redo日志都需要额外的fsyncs。将它的值设置为2会导致不太可靠（reliable）因为提交的事务仅仅每秒才flush一次到redo日志，但对于一些场景是可以接受的，比如对于主节点的备份节点这个值是可以接受的。如果值为0速度就更快了，但在系统崩溃时可能丢失一些数据：只适用于备份节点。

innodb_flush_method: 这项配置决定了数据和日志写入硬盘的方式。一般来说，如果你有硬件RAID控制器，并且其独立缓存采用write-back机制，并有着电池断电保护，那么应该设置配置为O_DIRECT；否则，大多数情况下应将其设为fdatasync（默认值）。sysbench是一个可以帮助你决定这个选项的好工具。

innodb_log_buffer_size: 这项配置决定了为尚未执行的事务分配的缓存。其默认值（1MB）一般来说已经够用

了，但是如果你的事务中包含有二进制大对象或者大文本字段的话，这点缓存很快就会被填满并触发额外的I/O操作。看看Innodb_log_waits状态变量，如果它不是0，增加innodb_log_buffer_size。

其他设置

query_cache_size: query cache（查询缓存）是一个众所周知的瓶颈，甚至在并发并不多的时候也是如此。最佳选项是将其从一开始就停用，设置query_cache_size = 0（现在MySQL 5.6的默认值）并利用其他方法加速查询：优化索引、增加拷贝分散负载或者启用额外的缓存（比如memcache或Redis）。如果你已经为你的应用启用了query cache并且还没有发现任何问题，query cache可能对你有用。这是如果你想停用它，那就得小心了。

log_bin：如果你想让数据库服务器充当主节点的备份节点，那么开启二进制日志是必须的。如果这么做了之后，还别忘了设置server_id为一个唯一的值。就算只有一个服务器，如果你想做基于时间点的数据恢复，这（开启二进制日志）也是很有用的：从你最近的备份中恢复（全量备份），并应用二进制日志中的修改（增量备份）。二进制日志一旦创建就将永久保存。所以如果你不想让磁盘空间耗尽，你可以用 PURGE BINARY LOGS 来清除旧文件，或者设置 expire_logs_days 来指定过多少天日志将被自动清除。

记录二进制日志不是没有开销的，所以如果你在一个非主节点的复制节点上不需要它的话，那么建议关闭这个选项。

skip_name_resolve：当客户端连接数据库服务器时，服务器会进行主机名解析，并且当DNS很慢时，建立连接也会很慢。因此建议在启动服务器时关闭skip_name_resolve选项而不进行DNS查找。唯一的局限是之后GRANT语句中只能使用IP地址了，因此在添加这项设置到一个已有系统中必须格外小心。

086 MySQL调优?（重点）

I 硬件配置优化

Ø CPU选择：多核的CPU，主频高的CPU

Ø 内存：更大的内存

Ø 磁盘选择：更快的转速、RAID、阵列卡，

Ø 网络环境选择：尽量部署在局域网、SCI、光缆、千兆网、双网线提供冗余、0.0.0.0多端口绑定监听

II操作系统级优化

Ø 使用64位的操作系统，更好的使用大内存。

Ø 设置noatime,nodiratime

Ø 优化内核参数

Ø 加大文件描述符限制

Ø 文件系统选择 xfs

III Mysql设计优化

III.1 存储引擎的选择

Ø Myisam：数据库并发不大，读多写少,而且都能很好的用到索引，sql语句比较简单的应用，TB数据仓库

Ø Innodb：并发访问大，写操作比较多，有外键、事务等需求的应用，系统内存较大。

III.2 命名规则

Ø 多数开发语言命名规则：比如MyAdress

Ø 多数开源思想命名规则：my_address

Ø 避免随便命名

III.3 字段类型选择

字段类型的选择的一般原则：

Ø 根据需求选择合适的字段类型，在满足需求的情况下字段类型尽可能小。

Ø 只分配满足需求的最小字符数，不要太慷慨。 原因：更小的字段类型更小的字符数占用更少的内存，占用更少的磁盘空间，占用更少的磁盘IO，以及占用更少的带宽。

对于varchar和char的选择要根据引擎和具体情况的不同来选择，主要依据如下原则:

1.如果列数据项的大小一致或者相差不大，则使用char。

2.如果列数据项的大小差异相当大，则使用varchar。

3.对于MyISAM表，尽量使用Char，对于那些经常需要修改而容易形成碎片的myisam和isam数据表就更是如此，它的缺点就是占用磁盘空间。

4.对于InnoDB表，因为它的数据行内部存储格式对固定长度的数据行和可变长度的数据行不加区分（所有数据行共用一个表头部分，这个标头部分存放着指向各有关数据列的指针），所以使用char类型不见得会比使用varchar类型好。事实上，因为char类型通常要比varchar类型占用更多的空间，所以从减少空间占用量和减少磁盘i/o的角度，使用varchar类型反而更有利。

5.表中只要存在一个varchar类型的字段，那么所有的char字段都会自动变成varchar类型，因此建议定长和变长的数据分开。

III.4 编码选择

单字节 latin1

多字节 utf8(汉字占3个字节，英文字母占用一个字节)如果含有中文字符的话最好都统一采用utf8类型，避免乱码的情况发生。

III.5 主键选择原则

注：这里说的主键设计主要是针对INNODB引擎

1.能唯一的表示行。

2.显式的定义一个数值类型自增字段的主键，这个字段可以仅用于做主键，不做其他用途。

3.MySQL主键应该是单列的，以便提高连接和筛选操作的效率。

4.主键字段类型尽可能小，能用SMALLINT就不用INT，能用INT就不用BIGINT。

5.尽量保证不对主键字段进行更新修改，防止主键字段发生变化，引发数据存储碎片，降低IO性能。

6.MySQL主键不应包含动态变化的数据，如时间戳、创建时间列、修改时间列等。

7.MySQL主键应当有计算机自动生成。

8.主键字段放在数据表的第一顺序。

推荐采用数值类型做主键并采用auto_increment属性让其自动增长。

III.6 其他需要注意的地方

Ø NULL OR NOT NULL

尽可能设置每个字段为NOT NULL，除非有特殊的需求，原因如下：

1.使用含有NULL列做索引的话会占用更多的磁盘空间，因为索引NULL列需要而外的空间来保存。

2.进行比较的时候，程序会更复杂。

3.含有NULL的列比较特殊，SQL难优化，如果是一个组合索引，那么这个NULL 类型的字段会极大影响整个索引的效率。

Ø 索引

索引的优点：极大地加速了查询，减少扫描和锁定的数据行数。

索引的缺点：占用磁盘空间，减慢了数据更新速度，增加了磁盘IO。

添加索引有如下原则：

1 选择唯一性索引。

2.为经常需要排序、分组和联合操作的字段建立索引。

3.为常作为查询条件的字段建立索引。

4.限制索引的数据，索引不是越多越好。

5.尽量使用数据量少的索引，对于大字段可以考虑前缀索引。

6.删除不再使用或者很少使用的索引。

7.结合核心SQL优先考虑覆盖索引。

8.忌用字符串做主键。

Ø 反范式设计

适当的使用冗余的反范式设计，以空间换时间有的时候会很高效。

IV Mysql软件优化

Ø 开启mysql复制，实现读写分离、负载均衡，将读的负载分摊到多个从服务器上，提高服务器的处理能力。

Ø 使用推荐的GA版本，提升性能

Ø 利用分区新功能进行大数据的数据拆分

V Mysql配置优化

注意：全局参数一经设置，随服务器启动预占用资源。

Ø key_buffer_size参数

mysql索引缓冲，如果是采用myisam的话要重点设置这个参数，根据（key_reads/key_read_requests）判断

Ø innodb_buffer_pool_size参数

INNODB 数据、索引、日志缓冲最重要的引擎参数，根据（hit riatos和FILE I/O）判断

Ø wait_time_out参数

线程连接的超时时间，尽量不要设置很大，推荐10s

Ø max_connections参数

服务器允许的最大连接数，尽量不要设置太大，因为设置太大的话容易导致内存溢出

Ø thread_concurrency参数

线程并发利用数量，(cpu+disk)*2,根据(os中显示的请求队列和tickets)判断

Ø sort_buffer_size参数

获得更快的–ORDER BY,GROUP BY,SELECT DISTINCT,UNION DISTINCT

Ø read_rnd_buffer_size参数

当根据键进行分类操作时获得更快的–ORDER BY

Ø join_buffer_size参数

join连接使用全表扫描连接的缓冲大小，根据select_full_join判断

Ø read_buffer_size参数

全表扫描时为查询预留的缓冲大小，根据select_scan判断

Ø tmp_table_size参数

临时内存表的设置，如果超过设置就会转化成磁盘表，根据参数(created_tmp_disk_tables)判断

Ø innodb_log_file_size参数(默认5M)

记录INNODB引擎的redo log文件，设置较大的值意味着较长的恢复时间。

Ø innodb_flush_method参数(默认fdatasync)

Linux系统可以使用O_DIRECT处理数据文件，避免OS级别的cache，O_DIRECT模式提高数据文件和日志文件的IO提交性能

Ø innodb_flush_log_at_trx_commit(默认1)

1.0表示每秒进行一次log写入cache，并flush log到磁盘。

2.1表示在每次事务提交后执行log写入cache，并flush log到磁盘。

3.2表示在每次事务提交后，执行log数据写入到cache，每秒执行一次flush log到磁盘。

VI Mysql语句级优化

1.性能查的读语句，在innodb中统计行数,建议另外弄一张统计表，采用myisam，定期做统计.一般的对统计的数据不会要求太精准的情况下适用。

2.尽量不要在数据库中做运算。

3.避免负向查询和%前缀模糊查询。

4.不在索引列做运算或者使用函数。

5.不要在生产环境程序中使用select * from 的形式查询数据。只查询需要使用的列。

6.查询尽可能使用limit减少返回的行数，减少数据传输时间和带宽浪费。

7.where子句尽可能对查询列使用函数，因为对查询列使用函数用不到索引。

8.避免隐式类型转换，例如字符型一定要用”，数字型一定不要使用”。

9.所有的SQL关键词用大写，养成良好的习惯，避免SQL语句重复编译造成系统资源的浪费。

10.联表查询的时候，记得把小结果集放在前面，遵循小结果集驱动大结果集的原则。

11.开启慢查询，定期用explain优化慢查询中的SQL语句。

087 mysql是怎么备份的（重点）

一、备份的目的

做灾难恢复：对损坏的数据进行恢复和还原

需求改变：因需求改变而需要把数据还原到改变以前

测试：测试新功能是否可用

二、备份需要考虑的问题

可以容忍丢失多长时间的数据；

恢复数据要在多长时间内完；

恢复的时候是否需要持续提供服务；

恢复的对象，是整个库，多个表，还是单个库，单个表。

三、备份的类型

1、根据是否需要数据库离线

冷备（**cold backup**）：需要关mysql服务，读写请求均不允许状态下进行；

温备（**warm backup**）： 服务在线，但仅支持读请求，不允许写请求；

热备（**hot backup**）：备份的同时，业务不受影响。

注：

1、这种类型的备份，取决于业务的需求，而不是备份工具

2、MyISAM不支持热备，InnoDB支持热备，但是需要专门的工具

2、根据要备份的数据集合的范围

完全备份：full backup，备份全部字符集。

增量备份: incremental backup 上次完全备份或增量备份以来改变了的数据，不能单独使用，要借助完全备份，备份的频率取决于数据的更新频率。

差异备份：differential backup 上次完全备份以来改变了的数据。

建议的恢复策略：

完全+增量+二进制日志

完全+差异+二进制日志

3、根据备份数据或文件

物理备份：直接备份数据文件

优点：备份和恢复操作都比较简单，能够跨mysql的版本，恢复速度快，属于文件系统级别的

建议：不要假设备份一定可用，要测试mysql>check tables；检测表是否可用

逻辑备份: 备份表中的数据和代码

优点：恢复简单、备份的结果为ASCII文件，可以编辑与存储引擎无关可以通过网络备份和恢复

缺点：备份或恢复都需要mysql服务器进程参与备份结果占据更多的空间，浮点数可能会丢失精度 还原之后，缩影需要重建

四：备份的对象

1、 数据；

2、配置文件；

3、代码：存储过程、存储函数、触发器

4、os相关的配置文件

5、复制相关的配置

6、二进制日志

五、备份和恢复的实现

1、利用select into outfile实现数据的备份与还原。

2、利用mysqldump工具对数据进行备份和还原

3、利用lvm快照实现几乎热备的数据备份与恢复

4、基于Xtrabackup做备份恢复。

优势：

1、快速可靠的进行完全备份
2、在备份的过程中不会影响到事务
3、支持数据流、网络传输、压缩，所以它可以有效的节约磁盘资源和网络带宽。
4、可以自动备份校验数据的可用性。

088 mysql 简单的 怎么登入 怎么创建数据库bbb创建 用户 密码 授权

一、创建用户:

CREATE USER用于创建新的MySQL账户。要使用CREATE USER，您必须拥有mysql数据库的全局CREATE USER权限，或拥有INSERT权限。对于每个账户，CREATE USER会在没有权限的mysql.user表中创建一个新记录。如果账户已经存在，则出现错误。

使用自选的IDENTIFIED BY子句，可以为账户给定一个密码。user值和 密码的给定方法和GRANT语句一 样。特别是，要在纯文本中指定密码，需忽略PASSWORD关键词。要把 密码指定为由PASSWORD()函数返回的混编值，需包含关键字PASSWORD

The create user command:mysql> CREATE USER yy IDENTIFIED BY '123';

面建立的用户可以在任何地方登陆。

mysql> CREATE USER yy@localhost IDENTIFIED BY '123';

二、授权:

命令:GRANT privileges ON databasename.tablename TO 'username'@'host'

说明: privileges - 用户的操作权限,如SELECT , INSERT , UPDATE 等.如果要授予所的权限则使用 ALL.;databasename - 数据库名,tablename-表名,如果要授予该用户对所有数据库和表的相应操作权限则可用*表示, 如*.*.

GRANT SELECT, INSERT ON test.user TO 'pig'@'%';

GRANT ALL ON . TO 'pig'@'%';

注意:用以上命令授权的用户不能给其它用户授权,如果想让该用户可以授权,用以下命令:

GRANT privileges ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION;

刷新系统权限表

flush privileges;

三、设置与更改用户密码

命令:SET PASSWORD FOR 'username'@'host' = PASSWORD('newpassword');如果是当前登陆用户用SET PASSWORD = PASSWORD("newpassword");

例子:SET PASSWORD FOR 'pig'@'%' = PASSWORD("123456");

或：update mysql.user set password=password('新密码') where User="phplamp" and Host="localhost";

四、撤销用户权限

命令: REVOKE privilege ON databasename.tablename FROM 'username'@'host';

说明: privilege, databasename, tablename - 同授权部分.

例子: REVOKE SELECT ON . FROM 'pig'@'%';

注意: 假如你在给用户'pig'@'%'授权的时候是这样的(或类似的):GRANT SELECT ON test.user TO 'pig'@'%', 则在使用 REVOKE SELECT ON . FROM 'pig'@'%';命令并不能撤销该用户对test数据库中user表的SELECT 操作.相反,如果授权使用的是GRANT SELECT ON . TO 'pig'@'%';则REVOKE SELECT ON test.user FROM 'pig'@'%';命令也不能撤销该用户对test数据库中user表的Select 权限.具体信息可以用命令SHOW GRANTS FOR 'pig'@'%'; 查看.

五、删除用户

命令: DROP USER 'username'@'host';

或：DELETE FROM user WHERE User="phplamp" and Host="localhost";

//删除用户的数据库

mysql>drop database phplampDB;

089 MySQL数据库同步怎样实现

1、安装配置，两台服务器，分别安装好MySQL。采用单向同步的方式，就是Master的数据是主的数据，然后slave主动去Master哪儿同步数据回来。两台服务器的配置一样，把关键的配置文件拷贝一下，两台服务器做相同的拷贝配置文件操作。

2、配置Master服务器，要考虑我们需要同步那个数据库，使用那个用户同步，我们这里为了简单起见，就使用root用户进行同步，并且只需要同步数据库abc。

3、配置Slave服务器，我们的slave服务器主要是主动去Master服务器同步数据回来。

4、测试安装，首先查看一下slave的主机日志：检查是否连接正常， 在Master查看信息，查看Master状态：查看Master下MySQL进程信息：在slave上查看信息：查看slave状态：查看slave下MySQL进程信息：再在Master的abc库里建立表结构并且插入数据，然后检查slave有没有同步这些数据，就能够检查出是否设置成功。

090 查询mysql数据库中用户，密码，权限的命令

查看MYSQL数据库中所有用户

SELECT DISTINCT CONCAT('User: ''',user,'''@''',host,''';') AS query FROM mysql.user;

查看数据库中具体某个用户的权限

show grants for 'cactiuser'@'%';

091 数据库死锁概念

多数情况下，可以认为如果一个资源被锁定，它总会在以后某个时间被释放。而死锁发生在当多个进程访问同一数据库时，其中每个进程拥有的锁都是其他进程所需的，由此造成每个进程都无法继续下去。简单的说，进程A等待进程B释放他的资源，B又等待A释放他的资源，这样就互相等待就形成死锁。

虽然进程在运行过程中，可能发生死锁，但死锁的发生也必须具备一定的条件，死锁的发生必须具备以下四个必要条件。

1）互斥条件：指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只由一个进程占用。如果此时还有其它进程请求资源，则请求者只能等待，直至占有资源的进程用毕释放。

2）请求和保持条件：指进程已经保持至少一个资源，但又提出了新的资源请求，而该资源已被其它进程占有，此时请求进程阻塞，但又对自己已获得的其它资源保持不放。

3）不剥夺条件：指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。

4）环路等待条件：指在发生死锁时，必然存在一个进程——资源的环形链，即进程集合{P0，P1，P2，•••，Pn}中的P0正在等待一个P1占用的资源；P1正在等待P2占用的资源，……，Pn正在等待已被P0占用的资源。

下列方法有助于最大限度地降低死锁：

（1）按同一顺序访问对象。

（2）避免事务中的用户交互。

（3）保持事务简短并在一个批处理中。

（4）使用低隔离级别。

（5）使用绑定连接。

092 数据库有几种数据保护方式（AAA）

实现数据库安全性控制的常用方法和技术有：**用户标识和鉴别；存取控制；视图机制；审计；数据加密；**

093 union和union all 的区别以及使用

Union因为要进行重复值扫描，所以效率低。如果合并没有刻意要删除重复行，那么就使用Union All两个要联合的SQL语句 字段个数必须一样，而且字段类型要“相容”（一致）；

union和union all的区别是,union会自动压缩多个结果集合中的重复结果，而union all则将所有的结果全部显示出来，不管是不是重复。

Union：对两个结果集进行并集操作，不包括重复行，同时进行默认规则的排序；

Union All：对两个结果集进行并集操作，包括重复行，不进行排序；

Intersect：对两个结果集进行交集操作，不包括重复行，同时进行默认规则的排序；

Minus：对两个结果集进行差操作，不包括重复行，同时进行默认规则的排序。

可以在最后一个结果集中指定Order by子句改变排序方式。

094 mysql的备份命令是什么

mysqldump -hhostname -uusername -ppassword databasename > backupfile.sql

备份MySQL数据库为带删除表的格式

备份MySQL数据库为带删除表的格式，能够让该备份覆盖已有数据库而不需要手动删除原有数据库。

mysqldump -–add-drop-table -uusername -ppassword databasename > backupfile.sql

直接将MySQL数据库压缩备份

mysqldump -hhostname -uusername -ppassword databasename | gzip > backupfile.sql.gz

备份MySQL数据库某个(些)表

mysqldump -hhostname -uusername -ppassword databasename specific_table1 specific_table2 > backupfile.sql

同时备份多个MySQL数据库

mysqldump -hhostname -uusername -ppassword –databases databasename1 databasename2 databasename3 > multibackupfile.sql

仅仅备份数据库结构

mysqldump –no-data –databases databasename1 databasename2 databasename3 > structurebackupfile.sql

备份服务器上所有数据库

mysqldump –all-databases > allbackupfile.sql

还原MySQL数据库的命令

mysql -hhostname -uusername -ppassword databasename < backupfile.sql

还原压缩的MySQL数据库

gunzip < backupfile.sql.gz | mysql -uusername -ppassword databasename

将数据库转移到新服务器

mysqldump -uusername -ppassword databasename | mysql –host=… -C databasename

095 在mysql服务器运行缓慢的情况下输入什么命令能缓解服务器压力

第一步 检查系统的状态

通过操作系统的一些工具检查系统的状态，比如CPU、内存、交换、磁盘的利用率，根据经验或与系统正常时的状态相比对，有时系统表面上看起来看空闲，这也可能不是一个正常的状态，因为cpu可能正等待IO的完成。除此之外，还应观注那些占用系统资源(cpu、内存)的进程。

1.1 使用sar来检查操作系统是否存在IO问题

1.2 使用vmstat监控内存 cpu资源

1.3 磁盘IO问题，处理方式：做raid10提高性能

1.4 网络问题，telnet一下MySQL对外开放的端口，如果不通的话，看看防火墙是否正确设置了。另外，看看MySQL是不是开启了skip-networking的选项，如果开启请关闭。

第二步 检查**mysql**参数

2.1 max_connect_errors

2.2 connect_timeout

2.3 skip-name-resolve

2.4 slave-net-timeout=seconds

2.5 master-connect-retry

第三步 检查**mysql** 相关状态值

3.1 关注连接数

3.2 关注下系统锁情况

3.3 关注慢查询（slow query）日志

096 怎么导出表结构?

1.导出整个数据库

mysqldump -u用户名 -p密码 数据库名 > 导出的文件名

C:\Users\jack> mysqldump -uroot -pmysql sva_rec > e:\sva_rec.sql

2.导出一个表，包括表结构和数据

mysqldump -u用户名 -p 密码 数据库名 表名> 导出的文件名

C:\Users\jack> mysqldump -uroot -pmysql sva_rec date_rec_drv> e:\date_rec_drv.sql

3.导出一个数据库结构

C:\Users\jack> mysqldump -uroot -pmysql -d sva_rec > e:\sva_rec.sql

4.导出一个表，只有表结构

mysqldump -u用户名 -p 密码 -d数据库名 表名> 导出的文件名

C:\Users\jack> mysqldump -uroot -pmysql -d sva_rec date_rec_drv> e:\date_rec_drv.sql

5.导入数据库

常用source 命令

进入mysql数据库控制台，

如mysql -u root -p

mysql>use 数据库

然后使用source命令，后面参数为脚本文件(如这里用到的.sql)

mysql>source d:wcnc_db.sql

097 正常登入MYSQL后使用什么命令查看其进程是否正常

输入show processlist;

如果有SUPER权限，则可以看到全部的线程，否则，只能看到自己发起的线程（这是指，当前对应的MySQL帐户运行的线程）。

098 mysql远程连接命令

一、MySQL 连接本地数据库，用户名为“root”，密码“123”（注意：“-p”和“123” 之间不能有空格）

C:>mysql -h localhost -u root -p123

二、MySQL 连接远程数据库（192.168.0.201），端口“3306”，用户名为“root”，密码“123”

C:>mysql -h 192.168.0.201 -P 3306 -u root -p123

099 mysql主从用什么方式传输日志

MySQL 复制基于主服务器在二进制日志中跟踪所有对数据库的更改(更新、删除等等)。每个从服务器从主服务器接收主服务器已经记录到其二进制日志的保存的更新，以便从服务器可以对其数据拷贝执行相同的更新。

100 数据库的备份方式

1、完全备份，这是大多数人常用的方式，它可以备份整个数据库，包含用户表、系统表、索引、视图和存储过程等所有数据库对象。但它需要花费更多的时间和空间，所以，一般推荐一周做一次完全备份。

2、事务日志备份，事务日志是一个单独的文件，它记录数据库的改变，备份的时候只需要复制自上次备份以来对数据库所做的改变，所以只需要很少的时间。为了使数据库具有鲁棒性，推荐每小时甚至更频繁的备份事务日志。

3、差异备份也叫增量备份。它是只备份数据库一部分的另一种方法，它不使用事务日志，相反，它使用整个数据库的一种新映象。它比最初的完全备份小，因为它只包含自上次完全备份以来所改变的数据库。它的优点是存储和恢复速度快。推荐每天做一次差异备份。

4、文件备份，数据库可以由硬盘上的许多文件构成。如果这个数据库非常大，并且一个晚上也不能将它备份完，那么可以使用文件备份每晚备份数据库的一部分。由于一般情况下数据库不会大到必须使用多个文件存储，所以这种备份不是很常用。

101 查看mysql数据库是否支持innodb

查看mysql的存储引擎：**show plugins;**

如何在mysql某个表中随机抽取10条记录

1.通过MYSQL内置的函数来操作，具体SQL代码如下：

```
SELECT * FROM tablename ORDER BY RAND() LIMIT 10
```

2.不要将大量的工作给数据库去做，这样会导致数据库在某一集中并发时间内锁死并阻塞。建议通过PHP随机生成一下1-X(总行数)之间的数字，然后将这10个随机数字作为查询条件，具体语句如：

```
SELECT * FROM tablename where ID in (2,8,4,11,12,9,3,1,33)
```

可能你还要进行重复排除，并且需要在程序中将10个值串联并连接进入SQL语句中。

102 如何查看连接mysql的当前用户。

show full processlist，在user字段中查看有哪些用户

103 写出mysql怎么修改密码?

方法一: (适用于管理员或者有全局权限的用户重设其它用户的密码)

```
进入命令行模式
mysql -u root -p
mysql>use mysql;
mysql> UPDATE user SET password=PASSWORD("new password") WHERE user='username';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

方法二:

```
mysql -u root -p
mysql>use mysql;
mysql> SET PASSWORD FOR username=PASSWORD('new password');
mysql> QUIT
```

方法三:

mysqladmin -u root “old password” “new password”

注：new password请输入你想要设置的密码。

104 MySQL怎么修复损坏的表?

有两种方法，一种方法使用mysql的check table和repair table 的sql语句，另一种方法是使用MySQL提供的多个myisamchk, isamchk数据检测恢复工具。

105 简单叙述一下MYSQL的优化（重点）

1.数据库的设计：尽量把数据库设计的更小的占磁盘空间.

1) 尽可能使用更小的整数类型.(mediumint就比int更合适).

2) 尽可能的定义字段为not null,除非这个字段需要null.

3) 如果没有用到变长字段的话比如varchar,那就采用固定大小的纪录格式比如char.

4) 表的主索引应该尽可能的短.这样的话每条纪录都有名字标志且更高效.

5) 只创建确实需要的索引。索引有利于检索记录，但是不利于快速保存记录。如果总是要在表的组合字段上做搜索，那么就在这些字段上创建索引。索引的第一部分必须是最常使用的字段.如果总是需要用到很多字段，首先就应该多复制这些字段，使索引更好的压缩。

6) 所有数据都得在保存到数据库前进行处理。

7) 所有字段都得有默认值。

8) 在某些情况下,把一个频繁扫描的表分成两个速度会快好多。在对动态格式表扫描以取得相关记录时，它可能使用更小的静态格式表的情况下更是如此。

2.系统的用途

1) 尽量使用长连接.

2) explain复杂的SQL语句。

3) 如果两个关联表要做比较话，做比较的字段必须类型和长度都一致.

4) LIMIT语句尽量要跟order by或者 distinct.这样可以避免做一次full table scan.

5) 如果想要清空表的所有纪录,建议用truncate table tablename而不是delete from tablename.

6) 能使用STORE PROCEDURE 或者 USER FUNCTION的时候.

7) 在一条insert语句中采用多重纪录插入格式.而且使用load data infile来导入大量数据，这比单纯的insert快好多.

8) 经常OPTIMIZE TABLE 来整理碎片.

9) 还有就是date 类型的数据如果频繁要做比较的话尽量保存在unsigned int 类型比较快。

3.系统的瓶颈

1) 磁盘搜索。并行搜索,把数据分开存放到多个磁盘中，这样能加快搜索时间.

2) 磁盘读写(IO)。可以从多个媒介中并行的读取数据。

3) CPU周期。数据存放在主内存中.这样就得增加CPU的个数来处理这些数据。

4) 内存带宽。当CPU要将更多的数据存放到CPU的缓存中来的话,内存的带宽就成了瓶颈.

106 如何确定有哪些存储引擎可用？

mysql> show engines; 显示了可用的数据库引擎的全部名单以及在当前的数据库服务器中是否支持这些引擎。

107 MYSQL数据库设计数据类型选择需要注意哪些地方?（重点）

VARCHAR和CHAR类型，varchar是变长的，需要额外的1-2个字节存储，能节约空间，可能会对性能有帮助。但由于是变长，可能发生碎片，如更新数据；
使用ENUM代替字符串类型，数据实际存储为整型。

字符串类型

要尽可能地避免使用字符串来做标识符，因为它们占用了很多空间并且通常比整数类型要慢。特别注意不要在MYISAM表上使用字符串标识符。MYISAM默认情况下为字符串使用了压缩索引（Packed Index），这使查找更为缓慢。据测试，使用了压缩索引的MYISAM表性能要慢6倍。

还要特别注意完全'随机'的字符串，例如由MD5（）、SHA1（）、UUID（）产生的。它们产生的每一个新值都会被任意地保存在很大的空间范围内，这会减慢INSERT及一些SELECT查询。

1）它们会减慢INSERT查询，因为插入的值会被随机地放入索引中。这会导致分页、随机磁盘访问及聚集存储引擎上的聚集索引碎片。

2)它们会减慢SELECT查询，因为逻辑上相邻的行会分布在磁盘和内存中的各个地方。

3）随机值导致缓存对所有类型的查询性能都很差，因为它们会使缓存赖以工作的访问局部性失效。如果整个数据集都变得同样"热"的时候，那么把特定部分的数据缓存到内存中就没有任何的优势了。并且如果工作集不能被装入内存中，缓存就会进行很多刷写的工作，并且会导致很多缓存未命中。

如果保存UUID值，就应该移除其中的短横线，更好的办法是使用UHEX（）把UUID值转化为16字节的数字，并把它保存在BINARY（16）列中。

108 mysql、oracle默认端口号

3306、1521

109 innodb的事务与日志的实现方式。

（1）有多少种日志

错误日志：记录出错信息，也记录一些警告信息或者正确的信息

慢查询日志：设置一个阈值，将运行时间超过该值的所有SQL语句都记录到慢查询的日志文件中。

二进制日志：记录对数据库执行更改的所有操作

查询日志：记录所有对数据库请求的信息，不论这些请求是否得到了正确的执行。

（2）日志的存放形式

（3）事务是如何通过日志来实现的，说得越深入越好。

在Innodb存储引擎中，事务日志是通过redo和innodb的存储引擎日志缓冲（Innodb log buffer）来实现 的，当开始一个事务的时候，会记录该事务的lsn(log sequence number)号; 当事务执行时，会往InnoDB存 储引擎的日志的日志缓存里面插入事务日志；当事务提交时，必须将存储引擎的日志缓冲写入磁盘（通过innodb_flush_log_at_trx_commit来控制），也就是写数据前，需要先写日志。这种方式称为"预写日志方 式"，innodb通过此方式来保证事务的完整性。也就意味着磁盘上存储的数据页和内存缓冲池上面的页是不同步 的，是先写入redo log，然后写入data file，因此是一种异步的方式。

隔离性： 通过锁实现

原子性、一致性和**持久性**是通过redo和undo来完成的。