1.

MySQL的日志大概可以分成三种，错误日志(error\_log)，查询日志(query\_log)，二进制日志(binary\_log)：

查询日志(query log)：一般的，查询日志可以分为两种，通用查询日志(general query log)和慢查询日志(slow query log)；其中，通用查询日志可以用来各客户端连接时的相关信息和在数据库上执行的SQL语句；慢查询日志记录了SQL语句时间超过了预设的long\_query\_time的语句，在数据量较大的情况下，可以看看慢查询日志中有哪些语句需要进行优化。

二进制日志(binary\_log)：简单来说，二进制日志记录了对MySQL更新的操作，主要目的是尽可能的将数据库恢复到数据库故障点，因为二进制日志包含备份后进行的所有更新。

2.

我们通过对my.cnf进行配置以后，启动mysql。我们可以使用以下语句进行验证是否配置被读取并且生效。

例如通过对数据库查询可以查看当前的log设置：

mysql> show variables like 'log\_error'; #展示log\_error日志的地址

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

| Variable\_name | Value |

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

| log\_error | /var/log/mysqld.log |

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

mysql> show variables like "datadir"; #展示数据库文件存放的地址

mysql> show variables like "socket"; #展示socket文件存放的地址

3.

mysql的配置从my.cnf中读取。里面可以分为2个区域[mysqld]以及[mysqld\_safe] 。 [mysqld]的配置信息将用于mysql服务器通过mysqld启动的时候读取，mysqld\_safe用于mysql服务器通过mysqld\_safe启动的时候读取。

mysqld\_safe启动比mysqld启动会多一些安全特性，例如当mysql崩溃的时候自动重启等。

例如配置log error:

1、找到配置文件/etc/my.cnf，如果找不到find / -type f -name 'my.cnf'全局查找即可

2、将错误日志参数写入配置文件

[mysqld]

log-error=/var/lib/mysql/mysql.err

4.

Mysql数据库文件：

mysql数据库在系统上是以一个文件夹的形式出现的，假设创建了Test数据库，那么在磁盘上就会出现Test的目录，而Test数据库的全局配置则是Test文件夹中的db.opt文件来管理。mysql的数据文件在datadir下，你在数据库中执行 show variables like '%datadir%';就会显示[数据库文件](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BA%93%E6%96%87%E4%BB%B6&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)的路径，你可以找到Test文件夹。

5.

MySQL储存数据，处理数据的依靠为数据引擎 (Storage Engine) 。在MySQL中，不同的数据引擎会带有不一样的支持形成，例如是否支持事务处理，是否支持并行处理，最大容纳多少数据量, 数据库锁的颗粒度是表格锁还是行锁, 是否支持外键限制等等。

查看本MySQL版本所支持的数据引擎以及所选用的数据引擎。

登陆进mySql控制台show engines

MySQL自20多年前成立以来一直支持可插拔存储引擎，但在一段相当长的时间里MyISAM一直是默认的存储引擎.MySQL刚开始是为小型网站的小型数据库设计的，许多应用已经习惯使用MyISAM存储引擎。

　　刚开始没什么问题，一切正常，但现在的问题是：MyISAM没有考虑到应用到高并发高负载，多核CPU和RAID阵列的场景，也不能弹性扩展。所以网站流量越来越多后，他们不能扩展，因为MySQL查询会在表级锁上等待数秒 (MyISAM只支持这种锁机制) 。他们不想每次MySQL崩溃时损坏他们的业务数据。

许多人并不知道，自MySQL存在以来MyISAM存储引擎就有一个兄弟叫InnoDB。并且高并发负载，性能和弹性(也包括原子性，一致性和隔离)正是它的特长。  
当然，在InnoDB发展过程中也有过一些问题(尤其是2006年5.0.30之前的版本的性能问题)，但在这之后的10年时间里，InnoDB已经在你能想到的领域(或者没有)得到了证明，而MyISAM已经很少被关注了。  
🡪因此，从MySQL 5.5.5开始，InnoDB成为默认的存储引擎，现在你几乎找不到大型MySQL数据库的安装使用MyISAM而不是InnoDB。

🡪MyISAM存储引擎不支持外键限制。在MyISAM存储引擎下设置[foreign\_key\_checks](https://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html" \l "sysvar_foreign_key_checks) 系统变量是不起作用的， 在InnoDB下才会有效果。

🡪原本的数据表为InnoDB, 如果在这些表中有设置外键域， 即使先设置foreign\_key\_checks = 0然后将表改为MyISAM类型也不可以。必须先将所以外键域取消才可以改变。

6.

查看mySql的版本mysql -V

7.

在mysql 5.1中，Alert操作具体后台流程

如果对表中的某个域进行alert操作，并且该域需要影响表格中的数据时（例如，修改某个域的类型并且需要所有的数据行上的该域数据都变化），则mysql实际上会根据目标表格首先复制一个临时表，然后在临时表中进行操作。并且在此期间内对原表的读操作不会有影响，对原表的写作操会锁住，并且等修改临时表操作完成后让暂停的写操作应用在新表当中。所以数据库在更改时所占用的空间会变大。如果这种临时表所需要的空间不够，则会修改失败。

如果对表中的某个域进行alert操作，并且该域不需要影响表格中的数据时（例如，修改某个域的名称，不需要每个数据行进行操作），则mysql不会对原表进行复制。Mysql只修改元数据马上完成。

在mysql 8.x中，Alert操作可以选择不同的算法。可以选择复制临时表处理方式， 可以选择直接对原表进行处理，可以执行只修改元数据不对具体数据行进行任何处理等方式。Mysql 8.0的默认行为是只修改元数据而不改变具体数据行。

8.

3306端口是MySQL的默认端口。33060是MySQL 扩展插口的默认端口。

9.

Mysql修改配置文件/etc/my.cnf以后需要重启才会生效。

10.

Mysql的配置文件分成不同的group例如[client], [mysqld], [mysql]等等，不同的区域将在不同的情况下读取生效。

[mysqld]对应SQL后台程序(也就是MySQL服务器) ，它是关于服务器端的一个程序，mysqld意思是mysql daemon，在后台运行，监听3306端口，如果你想要使用客户端程序，这个程序必须运行，因为客户端是通过连接服务器来访问数据库的。你只有启动了mysqld.exe，你的mysql数据库才能工作。

[mysql]对应是一个客户端软件，可以对任何主机的mysql服务（即后台运行的mysqld）发起连接，mysql自带的客户端程序一般都在cmd或者终端下进行操作

11.

字符集， 字符序

字符集（character set）：定义了字符以及字符的编码。

字符序（collation）：定义了字符的比较规则。

举个例子：

有四个字符：A、B、a、b，这四个字符的编码分别是A = 0, B = 1, a = 2, b = 3。这里的字符 + 编码就构成了字符集（character set）。

如果我们想比较两个字符的大小呢？比如A、B，或者a、b，最直观的比较方式是采用它们的编码，比如因为0 < 1，所以 A < B。

另外，对于A、a，虽然它们编码不同，但我们觉得大小写字符应该是相等的，也就是说 A == a。

这上面定义了两条比较规则，这些比较规则的集合就是collation。

同样是大写字符、小写字符，则比较他们的编码大小；

如果两个字符为大小写关系，则它们相等。

在数据的存储上，MySQL提供了不同的字符集支持。而在数据的对比操作上，则提供了不同的字符序支持。

MySQL提供了不同级别的设置，包括server级、database级、table级、column级，可以提供非常精准的设置。

**MySQL对于字符集字符序规定：**

一个字符集对应至少一种字符序（一般是1对多，也就是同一个字符集可以有不同的排序方式）。

两个不同的字符集不能有相同的字符序。

每个字符集在不指定的情况下，有默认的字符序。

查看支持的字符集：SHOW CHARACTER SET;

查看支持的字符序：SHOW COLLATION;

查看server字符集，字符序：

mysql> SHOW VARIABLES LIKE "character\_set\_server";

mysql> SHOW VARIABLES LIKE "collation\_server";

在/etc/my.cnf中进行配置：

[mysqld]

collation-server=utf8\_general\_ci

character-set-server=utf8

12. MySQL中的模式 (strict mode) ：

模式也就是MySQL中的sql\_mode,在数据库中可以使用show variables like “%sql\_mode%”;来查看现在MySql中启动的数据库模式

数据库模式有以下几种：

**ONLY\_FULL\_GROUP\_BY**  
在严格模式下，不要让GROUP BY部分中的查询指向未选择的列，否则报错。

**NO\_ZERO\_DATE**  
在严格模式，不要将’0000-00-00’做为合法日期。你仍然可以用IGNORE选项插入零日期。在非严格模式，可以接受该日期，但会生成警告。

**NO\_ZERO\_IN\_DATE**  
在严格模式，不接受月或日部分为0的日期，对年不限制。如果使用IGNORE选项，我们为类似的日期插入’0000-00-00’。在非严格模式，可以接受该日期，但会生成警告。

**ERROR\_FOR\_DIVISION\_BY\_ZERO**  
在严格模式，在INSERT或UPDATE过程中，如果被零除(或MOD(X，0))，则产生错误(否则为警告)。如果未给出该模式，被零除时MySQL返回NULL。如果用到INSERT IGNORE或UPDATE IGNORE中，MySQL生成被零除警告，但操作结果为NULL。

**NO\_AUTO\_CREATE\_USER**  
在严格模式下，防止GRANT自动创建新用户，除非还指定了密码。

**NO\_ENGINE\_SUBSTITUTION**  
如果需要的存储引擎被禁用或未编译，可以防止自动替换存储引擎。

**STRICT\_TRANS\_TABLES 以及 STRICT\_ALL\_TABLES (当启动时，数据库运行下严格模式下)**  
为事务存储引擎启用严格模式，也可能为非事务存储引擎启用严格模式，非法数据值被拒绝，下面有详细说明。

严格模式控制MySQL如何处理非法或丢失的输入值。有几种原因可以使一个值为非法。例如，数据类型错误，不适合列，或超出范围。当新插入的行不包含某列的没有显示定义DEFAULT子句的值，则该值被丢失。

对于事务表，当启用STRICT\_ALL\_TABLES或STRICT\_TRANS\_TABLES模式时，如果语句中有非法或丢失值，则会出现错误。语句被放弃并滚动。

对于非事务表，如果插入或更新的第1行出现坏值，两种模式的行为相同。语句被放弃，表保持不变。如果语句插入或修改多行，并且坏值出现在第2或后面的行，结果取决于启用了哪个严格选项：

对于STRICT\_ALL\_TABLES，MySQL返回错误并忽视剩余的行。但是，在这种情况下，前面的行已经被插入或更新。这说明你可以部分更新，这可能不是你想要的。要避免这点，最好使用单行语句，因为这样可以不更改表即可以放弃。

对于STRICT\_TRANS\_TABLES，MySQL将非法值转换为最接近该列的合法值并插入调整后的值。如果值丢失，MySQL在列中插入隐式 默认值。在任何情况下，MySQL都会生成警告而不是给出错误并继续执行语句。

严格模式不允许非法日期，例如’2004-04-31’。它允许日期使用“零”部分，例如’2004-04-00’或”零”日期。要想禁止，应在严格模式基础上，启用NO\_ZERO\_IN\_DATE和NO\_ZERO\_DATE SQL模式。

如果你不使用严格模式(即不启用STRICT\_TRANS\_TABLES或STRICT\_ALL\_TABLES模式)，对于非法或丢失的值，MySQL将插入调整后的值并给出警告。在

如果在/etc/my.cnf配置中放入：

[mysqld]

sql\_mode=IGNORE\_SPACE, NO\_ZERO\_IN\_DATE, NO\_ZERO\_DATE, ERROR\_FOR\_DIVISION\_BY\_ZERO, NO\_AUTO\_CREATE\_USER, NO\_ENGINE\_SUBSTITUTION

也就是不启动**STRICT\_TRANS\_TABLES以及STRICT\_ALL\_TABLES,则为取消严格模式。**

13.