1. 在查询频率高的字段**建立索引和缓存**，对于经常查询的数据建立索引会提升查询速度，同时把经常用的数据进行缓存，避免对数据库的多次查询减少磁盘的IO,节约时间。

2. **在where查询子语句上尽量避免对字段的NULL值判断，否则数据库引擎将会放弃索引而使用全表扫描.** 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.**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。

4.**尽量使用数字型字段**，否则会降低查询和连接的性能，并会增加存储开销。

**5.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。

**引起引擎放弃使用索引而进行全表扫描：**

1. 在 where 子句中使用!=或<>操作符，
2. where 子句中使用 or 来连接条件，
3. in 和 not in 也要慎用，
4. select id from t where name like '%abc%'
5. .应尽量避免在 where 子句中对字段进行表达式操作，如select id from t where num/2=100 应改为: select id from t where num=100\*2
6. 应尽量避免在where子句中对字段进行函数操作，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%'
7. 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。

6.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。

7.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(...)

8.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)

9.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。

10.尽可能的使用 varchar 代替 char ，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。

11.避免频繁创建和删除临时表，以减少系统表资源的消耗。

12.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。

13.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。

14.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。

15.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。

16.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。

17.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。

18.尽量避免大事务操作，提高系统并发能力。26.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。