1.对查询进行优化，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.尽量避免在 where 子句中使用!=或<>操作符，否则使查询放弃使用索引而进行全表扫描。

3.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致查询放弃使用索引而进行全表扫描，如：

select id from t where age is null

可以在age上设置默认值0，确保表中age列没有null值，然后这样查询：

select id from t where age=0

4.应尽量避免在 where 子句中使用 or 来连接条件，否则将导致查询放弃使用索引而进行全表扫描，如：

select id from t where age=10 or age=20

可以这样查询：

select id from t where age=10

union all

select id from t where age=20

5.下面的查询也将导致全表扫描：

select id from t where name like '%abc%'

若要提高效率，可以考虑全文检索。

6.in 和 not in 也要慎用，否则会导致全表扫描，如：

select id from t where age 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 age between 1 and 3

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 age=@age

可以改为强制查询使用索引：

select id from t with(index(索引名)) where age=@age

8.应尽量避免在 where 子句中对字段进行表达式操作，这将导致查询放弃使用索引而进行全表扫描。如：

select id from t where age/2=100

应改为:

select id from t where age=100\*2

9.应尽量避免在where子句中对字段进行函数操作，这将导致查询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

select id from t where datediff(day,createdate,'2005-11-30')=0--'2005-11-30'生成的id

应改为:

select id from t where name like 'abc%'

select id from t where createdate>='2005-11-30' and createdate<'2005-12-1'

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.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(...)

12.很多时候用 exists 代替 in 是一个好的选择：

select age from a where age in(select age from b)

用下面的语句替换：

select age from a where exists(select name from b where age=a.age)

13.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。

14.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。

15.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。

16.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。

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

18.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或是分页显示数据。

19.尽量避免大事务操作，提高系统并发能力。