http://www.ihref.com/read-16422.html

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，

```
Sql 代码 : 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：

```
Sql 代码 : select id from t where num=0;
```

3.应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎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，

```
Sql 代码 : select id from t where num=10 or num=20;
```

可以这样查询：

```
Sql 代码 : select id from t where num=10 union all select id from t where num=2
0;
```

5.in 和 not in 也要慎用，否则会导致全表扫描，如：

```
Sql 代码 : 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：

```
Sql 代码 : select id from t where num between 1 and 3;
```

6.下面的查询也将导致全表扫描：

```
Sql 代码 : select id from t where name like '%c%';
```

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：

```
Sql 代码 : select id from t where num=@num ;
```

可以改为强制查询使用索引：

```
Sql 代码 : 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。

```
Sql 代码 : select id from t where num/2=100;
```

可以这样查询：

```
Sql 代码 : 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：

```
Sql 代码 : select id from t where substring(name,1,3)='abc';#name 以 abc 开头的
id
```

应改为：

```
Sql 代码 : 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：

```
Sql 代码 : 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：

```
Sql 代码 : create table #t(…);
```

13.很多时候用 exists 代替 in 是一个好的选择：

```
Sql 代码 : select num from a where num in(select num from b);
```

用下面的语句替换：

```
Sql 代码 : 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 ***,male、female 几乎各一半，那么即使在 *** 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。 sql 优化方法使用索引来更快地遍历表。 缺省情况下建立的索引是非群集索引，但有时它并不是最佳的。在非群集索引下，数据在物理上随机存放在数据页上。合理的索引设计要建立在对各种查询的分析和预测上。一般来说：

a.有大量重复值、且经常有范围查询(> ,< ,> =,< =)和 order by、group by 发生的列，可考虑建立集群索引;

b.经常同时存取多列，且每列都含有重复值可考虑建立组合索引;

c.组合索引要尽量使关键查询形成索引覆盖，其前导列一定是使用最频繁的列。索引虽有助于提高性能但 不是索引越多越好，恰好相反过多的索引会导致系统低效。用户在表中每加进一个索引，维护索引集合就 要做相应的更新工作。

30.定期分析表和检查表。

```
分析表的语法：ANALYZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name[, tbl_name]...
```

以上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 能够生成正确的执行计划。如果用户感觉实际执行计划并不是预期的执行计划，执行一次分析表可能会解决问题。在分析期间，使用一个读取锁定对表进行锁定。这对于 MyISAM，DBD 和 InnoDB 表有作用。

```
例如分析一个数据表：analyze table table_name

检查表的语法：CHECK TABLE tbl_name[,tbl_name]...[option]...option = {QUICK | FAS
T | MEDIUM | EXTENDED | CHANGED}
```

检查表的作用是检查一个或多个表是否有错误，CHECK TABLE 对 MyISAM 和 InnoDB 表有作用，对于 MyISAM 表，关键字统计数据被更新

CHECK TABLE 也可以检查视图是否有错误，比如在视图定义中被引用的表不存在。

31.定期优化表。

```
优化表的语法:OPTIMIZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name [,tbl_name]...
```

如果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(含有 VARCHAR、BLOB 或 TEXT 列的表)进行更多更改，则应使用 OPTIMIZE TABLE 命令来进行表优化。这个命

令可以将表中的空间碎片进行合并，并且可以消除由于删除或者更新造成的空间浪费，但 OPTIMIZE TABLE 命令只对 MyISAM、 BDB 和 InnoDB 表起作用。

例如： `optimize table table_name`

注意： analyze、check、optimize 执行期间将对表进行锁定，因此一定注意要在 MySQL 数据库不繁忙的时候执行相关的操作。

补充：

1、在海量查询时尽量少用格式转换。

2、ORDER BY 和 GROPU BY:使用 ORDER BY 和 GROUP BY 短语，任何一种索引都有助于 SELECT 的性能提高。

3、任何对列的操作都将导致表扫描，它包括数据库教程函数、计算表达式等等，查询时要尽可能将操作移 至等号右边。

4、IN、OR 子句常会使用工作表，使索引失效。如果不产生大量重复值，可以考虑把子句拆开。拆开的子 句中应该包含索引。

5、只要能满足你的需求，应尽可能使用更小的数据类型：例如使用 MEDIUMINT 代替 INT

6、尽量把所有的列设置为 NOT NULL,如果你要保存 NULL,手动去设置它，而不是把它设为默认值。

7、尽量少用 VARCHAR、TEXT、BLOB 类型

8、如果你的数据只有你所知的少量的几个。最好使用 ENUM 类型

9、正如 graymice 所讲的那样，建立索引。

10、合理用运分表与分区表提高数据存放和提取速度。