 之前做过一次试验，当所建立的索引没有在where条件中应用时，查询结果需要40多分钟；而建对索引时，不到30秒，很神奇吧，这就是索引的作用，他就像目录一样，可以轻松的找到你想要的数据，就像字典，如果没有目录，可想其查询的难度。

    在总结索引设计的原则时，我们有必要来阐述SQLSERVER中索引的四种类型：

1.聚集索引（Cluster Index）  
    聚集索引中索引存储的值的顺序和表中的数据的物理存储顺序是完全一致的。建立索引时，系统将对表的物理数据页中的数据按列进行排序，然后再重新存储到磁盘上，即聚集索引与数据是混为一体的，它的叶节点中存储的是实际的数据。特点如下：

* 表的数据按照索引的数据顺序排列；
* 每个数据表只能建立一个聚集索引，并且会在第一个建立，往往会在主码所在的列或者最常查询的列上建立聚集索引；
* 索引将占用用户数据库的空间；
* 适合范围查询。

聚集索引通常提供更快的数据访问速度。

    建立聚集索引后，更新索引列数据，往往会导致表中物理记录的存储顺序的变化，维护的代价会比较大，对于需要经常更新的列，不宜建立聚集索引。

2.非聚集索引（Non-Cluster Index）  
    非聚集索引存储的数据顺序一般和表的物理数据的存储不同。尽管查询速度慢一些，但是维护的代价小。而且表中最多可以建立249个非聚集索引以满足多种查询的需要。

3.惟一索引（Unique Index）  
    惟一索引是指索引存储的值必须是惟一的，不允许两行具有相同的索引值（包括NULL）。主码索引时当然的惟一索引。

4.复合索引  
    复合索引是指利用表中的多个列值的组合来构建索引值。SQLServer 2000规定复合索引最多使用16个列的值进行组合，索引列值最大长度不能超过900字节，而且这些列必须在同一个数据表中。

    而合理的索引设计是要建立在对各种查询的分析和预测上，我们可以按照如下原则建立索引，希望对软件开发相关人员有所帮助：

* 索引中的数据尽可能少，即窄索引更容易被选择；
* 聚集索引码要被包含在表的所有的非聚集索引中，所以聚集索引码要尽可能短；
* 建立高选择性的非聚集索引；
* 频繁请求的列上不能建立聚集索引，应该建立非聚集索引，并且要尽可能使值惟一；
* 尽可能减少热点数据。如果频繁地对表中的某些数据进行读和写，这些数据就是热点数据，要想办法将热点数据分散；
* 监控磁盘的数据流量。如果利用率太高，就要考虑索引列并在多个磁盘上分布数据以减少I/O；
* 在至少有一个索引的表中，应该有一个聚集索引。包括的不同值的个数有限，返回一定范围内值的列，查询时返回大量结果的列考虑建立聚集索引；
* 分析经常使用的SQL语句的Where子句，得出经常取值的数据，考虑对这些数据列根据常见的查询类型建立索引；
* 主码如果涉及多个数据列，要将显著变化的数据列放在首位。如果数据列的变化程度相当，将经常访问的数据列放在首位；
* 有大量重复值、且经常有范围查询。如（between，>,<,>=,<=）、order by、group by发生的列，可考虑建立聚集索引；
* SQL查询语句同时存取多列的数据，且每列都含有重复值，可以考虑建立覆合索引，覆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；
* 索引值较短的索引具有比较高的效率，因为每个索引页上能存放较多的索引行，而且索引的级别也比较少。所以，缓存中能防止更多的索引列，这样也减少了I/O操作；
* 表上的索引过多会影响UPDATE、INSERT和DELETE的性能，因为所有的索引必须做响应的调整。另外，所有的分页操作都被记录在日志中，这样也会增加I/O操作；
* 一般不对经常被更新的列建立聚集索引，这样会引起整行的移动，严重影响性能；
* 查询很少或着数据很少的数据表一般不用建立索引；
* 与ORDER BY或GROUP BY一起使用的列一般使用建立聚集索引。如果ORDER BY 命令中用到的列上有聚集索引，就不会生成1个临时表，因为行已经排序。GROUP BY命令则一定产生1个临时表；
* 当有大量的行正在被插入表中时，要避免在本表一个自然增长（例如Identity列）的列上建立聚集索引。如果建立了聚集索引，那么INSERT的性能就会大大降低，因为每个插入的行必须到表的最后一个数据页面。

五、索引分类  
    1.直接创建索引和间接创建索引  
    直接创建索引：： CREATE INDEX mycolumn\_index ON mytable (myclumn)

    间接创建索引：定义主键约束或者唯一性键约束，可以间接创建索引  
    2.普通索引和唯一性索引  
    普通索引：： CREATE INDEX mycolumn\_index ON mytable (myclumn)

    唯一性索引：保证在索引列中的全部数据是唯一的，对聚簇索引和非聚簇索引都可以使用  
    CREATE UNIQUE clustered INDEX myclumn\_cindex ON mytable(mycolumn)

3.单个索引和复合索引  
    单个索引：即非复合索引  
    复合索引：又叫组合索引，在索引建立语句中同时包含多个字段名，最多16个字段

CREATE INDEX name\_index ON username(firstname,lastname)

    4.聚簇索引和非聚簇索引(聚集索引，群集索引)  
   聚簇索引：物理索引，与基表的物理顺序相同，数据值的顺序总是按照顺序排列  
 CREATE CLUSTERED INDEX IX\_TestTable\_TestCol1 ON dbo.TestTable (TestCol1);

(允许有重复记录的聚簇索引)  
   非聚簇索引：

CREATE NONCLUSTERED INDEX IX\_ProductVendor\_VendorID ON Purchasing.ProductVendor (BusinessEntityID);

操作案例

最普通的情况，是为出现在where子句的[字段](http://baike.baidu.com/view/159839.htm" \t "_blank)建一个[索引](http://baike.baidu.com/view/262241.htm)。为方便讲述，先建立一个如下的表。

**CREATE TABLE mytable(**

idserial primary key,

category\_id int not null default0,

user\_id int not null default0,

adddate int not null default0

);

如果在查询时常用类似以下的语句：

SELECT \* FROM mytable WHERE category\_id=1;

最直接的应对之道，是为category\_id建立一个简单的[索引](http://baike.baidu.com/view/262241.htm" \t "_blank)：

**CREATE INDEX mytable\_categoryid ON mytable (category\_id);**

OK.如果有不止一个选择条件呢？例如：

**SELECT \* FROM mytable WHERE category\_id=1 AND user\_id=2;**

第一反应可能是，再给user\_id建立一个索引。不好，这不是一个最佳的方法。可以建立多重的[索引](http://baike.baidu.com/view/262241.htm" \t "_blank)。

**CREATE INDEX mytable\_categoryid\_userid ON mytable(category\_id,user\_id);**

注意到在命名时的习惯了吗？使用"表名\_[字段](http://baike.baidu.com/view/159839.htm)1名\_字段2名"的方式。很快就会知道为什么这样做了。