Why use a DBMS

data independence and efficient access

reduced application development time

data integrity and security

uniform data administration

concurrent access, recovery from crashes

File vs. Database

application must stage large datasets between main memory and secondary storage

special code for different queries

must protect data from inconsistence due to multiple concurrent users

crash recovery

security and access control

视情况而定，索引有好处也有坏处。为什么要创建索引？这是因为创建索引可以大大提高系统的性能。  
第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。  
第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。  
第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。  
第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。  
第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

也许会有人要问：增加索引有如此多的优点，为什么不对表中的每一个列创建一个索引呢？这种想法固然有其合理性，然而也有其片面性。虽然，索引有许多优点，但是为表中的每一个列都增加索引是非常不明智的。这是因为增加索引也有许多不利的方面。  
第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。  
第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。  
第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

**因此，在创建索引的时候，应该仔细考虑在哪些列上可以创建索引，在哪些列上不能创建索引。**

一般来说，应该在这些列上创建索引：  
1、在经常需要搜索的列上建立索引可以加快检索速度；  
2、在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；  
3、在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；  
4、在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；  
5、在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；  
6、在经常使用在WHERE子句中的列上创建索引，加快条件的判断速度。

同样，对于有些列不应该创建索引。一般来说，不应该创建索引的的这些列具有下列特点：  
第一，对于那些在查询中很少使用或参考的列不应创建索引。这是因为既然这些列很少使用到，因此有索引或者无索引并不能提高查询速度。相反，由于增加了索引反而降低了系统的维护速度和增大了空间需求。  
第二，对于那些只有很少数据值的列也不应增加索引。这是因为由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引并不能明显加快检索速度。  
第三，对于那些定义为text, image和bit数据类型的列不应增加索引。这是因为这些列的数据量要么相当大，要么取值很少。  
第四，当修改性能远远大于检索性能时，不应创建索引。这是因为修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能但是会降低修改性能。当减少索引时，会提高修改性能但会降低检索性能。

数据独立性

数据的独立性是数据库系统的最基本的特征之一 。数据独立性是指应用程序和数据结构之间相互独立, 互不影响。在三层模式体系结构中数据独立性是指数据库系统在某一层次模式上的改变不会使它的上一层模式也发生改变的能力。正是三级模式间的两层映像保证了数据库系统中的数据具有较高的数据独立性。

数据独立性包括数据逻辑独立性和数据物理独立性。

逻辑独立性

用户的应用程序与数据库的逻辑结构是相互独立的。即，当数据的逻辑结构改变时，用户程序也可以不变。

有了外模式/模式映像，模式改变时，如增加新的属性、修改属性的类型，只要对外模式/模式的映像做相应的改变，可使外模式保持不变，则以外模式为依据编写的应用程序就不受影响，从而应用程序不必修改，保证了数据与程序之间的逻辑独立性，也就是逻辑数据独立性。

物理独立性

用户的应用程序与存储在磁盘上的数据库中数据是相互独立的。即，数据在磁盘上怎样存储由DBMS管理，用户程序不需要了解，应用程序要处理的只是数据的逻辑结构，这样当数据的物理存储改变了，应用程序不用改变。

有了模式/内模式映像，当内模式改变时，如存储设备或存储方式有所改变，只要对模式/内模式映像做相应的改变，使模式保持不变，则应用程序就不受影响，从而保证了数据与程序之间的物理独立性，称为存储数据独立性

二、笛卡尔积

例如， 给出3个域：

D1=导师集合SUPERVISOR=｛张清玫，刘逸｝

D2=专业集合SPECIALITY=｛计算机专业，信息专业｝

D3=研究生集合POSTGRADUATE=｛李勇，刘晨，王敏｝

D1，D2，D3的笛卡尔积为

D1×D2×D3＝｛

(张清玫，计算机专业，李勇)，(张清玫，计算机专业，刘晨)，

(张清玫，计算机专业，王敏)，(张清玫，信息专业，李勇)，

(张清玫，信息专业，刘晨)，(张清玫，信息专业，王敏)，

(刘逸，计算机专业，李勇)，(刘逸，计算机专业，刘晨)，

(刘逸，计算机专业，王敏)，(刘逸，信息专业，李勇)，

(刘逸，信息专业，刘晨)，(刘逸，信息专业，王敏) ｝

基数为2×2×3＝12

1NF是指数据库表的每一列都是不可分割的基本数据项，即实体中的某个属性不能有多个值或者不能有重复的属性。

2NF要求属性完全依赖于主键，不能存在仅依赖主关键字一部分的属性。

3NF要求每一个非主属性既不部分依赖于码也不传递依赖于码。

BCNF消除了主属性对候选码的部分和传递函数依赖。

注：1.相对于BCNF，3NF允许存在主属性对候选码的传递依赖和部分依赖。

2.BCNF比较抽象，略作解释：在学生信息表里，学号是一个候选码，学号可确定学生姓名；(班级,学生姓名）也是一组候选码，有(班级,学生姓名）->学号，因此在主属性间形成了传递依赖。

3NF分解：

口诀：

保函依赖分解题，先求最小依赖集。

依赖两侧未出现，分成子集放一边，剩余依赖变子集。

若要连接成无损，再添候选做子集。

下面通过几道例题讲解口诀：

例1.已知R(ABCDE), F={A ->D,E->D,D->B,BC->D,DC->A}求保持函数依赖的3NF分解，和具有无损连接性及保持函数依赖的3NF分解

第一步：保函依赖分解题，先求最小依赖集。先求出R的最小依赖集，可得F={A ->D,E->D,D->B,BC->D,DC->A}

第二步：依赖两侧未出现，分成子集放一边。首先可以发现没有不出现在两侧的元素不用单独分出一个子集，“剩余依赖变子集”然后我们将各依赖分别划分为子集得到：{AD} {ED} {DB} {BCD} {DCA}，即为所求保持函数依赖的3NF分解

第三步：若要连接成无损，再添候选做子集。

(1)候选码的求解：所谓候选码即能决定整个关系的，我们通过找未出现在依赖右边的和两侧均未出现的元素即可求得，

(2)可以发现C E未出现在右边，因此候选码为{CE}。故所求具有无损连接性及保持函数依赖的3NF分解为{AD} {ED} {DB} {BCD} {DCA} {CE}

例2.关系模式R,有U={A,B,C,D,E,G},F={B->G,CE->B,C->A,CE->G,B->D,C->D}，将关系模式分解为3NF且保持函数依赖

将关系模式分解为3NF且保持函数依赖：

第一步：保函依赖分解题，先求最小依赖集。先求出R的最小依赖集，

假设B->G冗余，则(B)+=BD,没有G故不冗余。

假设CE->B冗余，则（CE）+=CEGDA,没有B故不冗余。

假设C->A冗余，则（C）+=CD,故不冗余。

一次可以得到最小函数依赖集Fm={B->G,CE->B,C->A,B->D,C->D}

第二步：依赖两侧未出现，分成子集放一边，剩余依赖变子集。首先可以发现没有不出现在两侧的元素，然后我们将各依赖分别划分为子集得{BG} {CEB} {CA} {BD} {CD}，即为所求保持函数依赖的3NF分解

第三步：若要连接成无损，再添候选做子集。找到R的一个候选码为{CE}。故所求具有无损连接性及保持函数依赖的3NF分解为{BG} {CEB} {CA} {BD} {CD} {CE} (注：范式分解并不唯一，正确即可)

二、BCNF分解：

将关系模式R<U,F>分解为一个BCNF的基本步骤是

1）先求最小依赖集，候码非码成子集

3）余下左侧全候码，完成BCNF题。

例.关系模式R,有U={A,B,C,D,E,G},F={B->G,CE->B,C->A,CE->G,B->D,C->D}，将关系模式分解为3NF且保持函数依赖

将关系模式分解为3NF且保持函数依赖：

第一步：先求最小依赖集。可以发现CE->G多余，因此最小依赖集为F={B->G,CE->B,C->A,B->D,C->D}。

第二步：候码非码成子集。由于候选码为(CE)因此将CE->B划分出子集（BCE），而B->G，B->D左侧均不含主属性(C、E)中的任何一个故划分出(BG),(BD）

第三步：此时剩余依赖F={C->A,C->D}剩余元素{A,C,D}检查发现函数依赖左侧都是候选码即完成BCNF分解，如果不满足则继续分解余下的。

于是BCNF分解的最后结果为{（BG）,(BD),(ACD),(BCE)}。