**一、数据库设计模式：**

**主扩展模式**，通常用来将几个相似的对象的共有属性抽取出来，形成一个“公共属性表”；其余属性则分别形成“专有属性表”，且“公共属性表”与“专有属性表”都是“一对一”的关系。“专有属性表”可以看作是对“公共属性表”的扩展，两者合在一起就是对一个特定对象的完整描述，故此得名“主扩展模式”。

**适用情况：**对象的个数不多；各个对象之间的属性有一定差别；各个对象的属性在数据库设计阶段能够完全确定；各个扩展对象有独立的、相对比较复杂的业务处理需求，此时用“主扩展模式”。将各个对象的共有属性抽取出来设计为“主表”，将各个对象的剩余属性分别设计为相应的“扩展表”，“主表”与各个“扩展表”分别建立一对一的关系。

**主从模式**，是数据库设计模式中最常见、也是大家日常设计工作中用的最多的一种模式，它描述了两个表之间的主从关系，是典型的“一对多”关系。如论坛表对应多张发帖表，发帖表对应多张回复表。

**适用情况**：对象的个数较多且不固定；各个对象之间的属性几乎没有差异；对象的属性在数据库设计阶段能够完全确定；各个对象没有独立的业务处理需求，此时用“主从模式”。将各个对象设计为“从表”的记录，与“主表”对象建立一对多的关系。

**名值模式**，通常用来描述在系统设计阶段不能完全确定属性的对象，这些对象的属性在系统运行时会有很大的变更，或者是多个对象之间的属性存在很大的差异。

**适用情况：**对象的个数极多；各个对象之间的属性有较大差异；对象属性在数据库设计阶段不能确定，或者在系统运行时有较大变更；各个对象没有相互独立的业务处理需求，此时用“名值模式”。

**多对多模式**，也是比较常见的一种数据库设计模式，它所描述的两个对象不分主次、地位对等、互为一对多的关系。对于A表来说，一条记录对应着B表的多条记录，反过来对于B表来说，一条记录也对应着A表的多条记录，这种情况就是“多对多模式”。“多对多模式”需要在A表和B表之间有一个关联表，这个关联表也是“多对多模式”的核心所在。根据关联表是否有独立的业务处理需求，可将其划分为两种细分情况：关联表有独立的业务处理需求以及关联表没有独立的业务需求。

**适用情况：**两个对象之间互为一对多关系，则使用“多对多模式”。

**二、模糊查询**

在SQL中查询中常用的模糊查询有三个关键字：LIKE、IS NULL、BETWEEN、IN。

select \* from stuInfo where name like '张%'

select \* from stuInfo where SAddress is null

select stuName,Score from Score where Score between 60 and 80

select \* from stuInfo where SAddress in('北京','广州','上海')

**三、关系数据库**

关系数据库是支持关系模型的数据库系统。关系模型由关系数据结构、关系操作集合和完整性约束三部分组成。**关系操作**分为关系代数、关系演算、具有关系代数和关系演算双重特点的语言（SQL语言）。**完整性约束**提供了丰富的完整性：实体完整性、参照完整性和用户定义的完整性。  
**特点：**

1. 操作方便：通过应用程序和后台联结，方便了用户的对数据的操作，特别是没有编程基础的人
2. 易于维护:丰富的完整性：实体完整性、参照完整性和用户定义的完整性，大大降低了数据的冗余，和数据不一致的概率
3. 便于访问数据: 提供了诸如视图，存储过程，触发器，索引等对象，
4. 更安全，更快捷:权限的分配，使其较以往的数据库在安全性上要高的多

**四、数据库范式**

设计关系数据库时，遵从不同的规范要求，设计出合理的关系型数据库，这些不同的规范要求被称为不同的范式，各种范式呈递次规范，越高的范式数据库冗余越小。

目前[关系数据库](http://baike.baidu.com/view/68348.htm" \t "_blank)有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、[第四范式](http://baike.baidu.com/view/176748.htm" \t "_blank)(4NF）和[第五范式](http://baike.baidu.com/view/7170192.htm" \t "_blank)（5NF，又称完美范式）。满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多规范要求的称为第二范式（2NF），其余范式以次类推。一般说来，数据库只需满足第三范式(3NF）就行了。

**第一范式（1NF）:**所谓第一范式（1NF）是指在[关系模型](http://baike.baidu.com/view/176484.htm" \t "_blank)中，对域添加的一个规范要求，所有的域都应该是原子性的，即数据库表的每一列都是不可分割的原子数据项，而不能是集合，数组，记录等非原子数据项。即实体中的某个属性有多个值时，必须拆分为不同的属性。在符合第一范式（1NF）表中的每个域值只能是实体的一个属性或一个属性的一部分。第一范式就是无重复的域。是否必须满足1NF的最低要求，主要依赖于所使用的[关系模型](http://baike.baidu.com/view/176484.htm" \t "_blank)。

**第二范式（2NF）**在1NF的基础上，非码属性必须完全依赖于码[在1NF基础上消除非主属性对主码的部分函数依赖]**.**满足2NF必须先满足1NF。2NF要求数据库表中的每个实例或记录必须可以被唯一地区分。选取一个能区分每个实体的属性或属性组，作为实体的唯一标识。例如在员工表中的身份证号码即可实现每个一员工的区分，该身份证号码即为候选键，任何一个候选键都可以被选作主键。在找不到候选键时，可额外增加属性以实现区分，如果在员工关系中，没有对其身份证号进行存储，而姓名可能会在数据库运行的某个时间重复，无法区分出实体时，设计辟如ID等不重复的编号以实现区分，被添加的编号或ID选作主键。

第二范式（2NF）要求实体的属性完全依赖于主关键字。简而言之，第二范式就是在第一范式的基础上属性完全依赖于主键。

**第三范式（3NF）**在1NF基础上，任何非主[属性](http://baike.baidu.com/view/77730.htm)不依赖于其它非主属性[在2NF基础上消除传递依赖]**.**第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个关系中不包含已在其它关系已包含的非主关键字信息。例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性，也就是在满足2NF的基础上，任何非主属性不得传递依赖于主属性。

**巴斯-科德范式（BCNF）:**在1NF基础上，任何非主[属性](http://baike.baidu.com/view/77730.htm)不能对主键子集依赖[在3NF基础上消除对主码子集的依赖]巴德斯科范式（BCNF）是第三范式（3NF）的一个子集，即满足巴斯-科德范式（BCNF）必须满足第三范式（3NF）。通常情况下，巴斯-科德范式被认为没有新的设计规范加入，只是对第二范式与第三范式中设计规范要求更强，因而被认为是修正第三范式，也就是说，它事实上是对第三范式的修正，使数据库冗余度更小。这也是BCNF不被称为第四范式的原因。某些书上，根据范式要求的递增性将其称之为第四范式是不规范，也是更让人不容易理解的地方。而真正的第四范式，则是在设计规范中添加了对多值及依赖的要求。对于BCNF，在主码的任何一个[真子集](http://baike.baidu.com/view/1205.htm)都不能决定于非主属性。关系中U主码，若U中的任何一个[真子集](http://baike.baidu.com/view/1205.htm)X都不能决定于非主属性Y，则该设计规范属性BCNF。例如：在关系R中，U为主码，A属性是主码中的一个属性，若存在A->Y,Y为非主属性，则该关系不属性BCNF。

一般关系型数据库设计中，达到BCNF就可以了！

**五、数据库索引**

[索引](http://baike.baidu.com/view/262241.htm)是对数据库表中一列或多列的值进行排序的一种结构。索引分为[聚簇索引](http://baike.baidu.com/view/1028053.htm)和[非聚簇索引](http://baike.baidu.com/view/1615249.htm)两种，聚簇索引 是按照数据存放的物理位置为顺序的，而非聚簇索引就不一样了；聚簇索引能提高多行检索的速度，而非聚簇索引对于单行的检索很快。建立[索引](http://baike.baidu.com/view/262241.htm)的目的是加快对表中记录的查找或排序。根据数据库的功能，可以在[数据库设计](http://baike.baidu.com/view/8268.htm)器中创建三种[索引](http://baike.baidu.com/view/262241.htm)：[唯一索引](http://baike.baidu.com/view/709651.htm)、主键索引和[聚集索引](http://baike.baidu.com/view/692530.htm)。（也有分类是唯一索引 主键索引 普通索引 组合索引）

**优点：**1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。 2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。3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。4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。5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。

**缺点**：1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。2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。 3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。

创建索引有多种方法，这些方法包括直接创建索引的方法和间接创建索引的方法。直接创建索引，例如使用CREATE INDEX语句或者使用创建索引向导，间接创建索引，例如在表中定义主键约束或者唯一性键约束时，同时也创建了索引。虽然，这两种方法都可以创建索引，但 是，它们创建索引的具体内容是有区别的。   
使用CREATE INDEX语句或者使用创建索引向导来创建索引，这是最基本的索引创建方式，并且这种方法最具有柔性，可以定制创建出符合自己需要的索引。在使用这种方式 创建索引时，可以使用许多选项，例如指定数据页的充满度、进行排序、整理统计信息等，这样可以优化索引。使用这种方法，可以指定索引的类型、唯一性和复合 性，也就是说，既可以创建聚簇索引，也可以创建非聚簇索引，既可以在一个列上创建索引，也可以在两个或者两个以上的列上创建索引。

通过定义主键约束或者唯一性键约束，也可以间接创建索引。主键约束是一种保持数据完整性的逻辑，它限制表中的记录有相同的主键记录。在创建主键约束时，系统自动创建了一个唯一性的聚簇索引。虽然，在逻辑上，主键约束是一种重要的结构，但是，在物理结构上，与主键约束相对应的结构是唯一性的聚簇索引。换句话 说，在物理实现上，不存在主键约束，而只存在唯一性的聚簇索引。同样，在创建唯一性键约束时，也同时创建了索引，这种索引则是唯一性的非聚簇索引。因此， 当使用约束创建索引时，索引的类型和特征基本上都已经确定了，由用户定制的余地比较小。

当在表上定义主键或者唯一性键约束时，如果表中已经有了使用CREATE INDEX语句创建的标准索引时，那么主键约束或者唯一性键约束创建的索引覆盖以前创建的标准索引。也就是说，主键约束或者唯一性键约束创建的索引的优先 级高于使用CREATE INDEX语句创建的索引。

**索引的特征** 索引有两个特征，即唯一性索引和复合索引。 唯一性索引保证在索引列中的全部数据是唯一的，不会包含冗余数据。如果表中已经有一个主键约束或者唯一性键约束，那么当创建表或者修改表时，SQL Server自动创建一个唯一性索引。然而，如果必须保证唯一性，那么应该创建主键约束或者唯一性键约束，而不是创建一个唯一性索引。当创建唯一性索引 时，应该认真考虑这些规则：当在表中创建主键约束或者唯一性键约束时，SQL Server自动创建一个唯一性索引；如果表中已经包含有数据，那么当创建索引时，SQL Server检查表中已有数据的冗余性；每当使用插入语句插入数据或者使用修改语句修改数据时，SQL Server检查数据的冗余性：如果有冗余值，那么SQL Server取消该语句的执行，并且返回一个错误消息；确保表中的每一行数据都有一个唯一值，这样可以确保每一个实体都可以唯一确认；只能在可以保证实体 完整性的列上创建唯一性索引，例如，不能在人事表中的姓名列上创建唯一性索引，因为人们可以有相同的姓名。 复合索引就是一个索引创建在两个列或者多个列上。在搜索时，当两个或者多个列作为一个关键值时，最好在这些列上创建复合索引。当创建复合索引时，应该考虑 这些规则：最多可以把16个列合并成一个单独的复合索引，构成复合索引的列的总长度不能超过900字节，也就是说复合列的长度不能太长；在复合索引中，所 有的列必须来自同一个表中，不能跨表建立复合列；在复合索引中，列的排列顺序是非常重要的，因此要认真排列列的顺序，原则上，应该首先定义最唯一的列，例 如在（COL1，COL2）上的索引与在（COL2，COL1）上的索引是不相同的，因为两个索引的列的顺序不同；为了使查询优化器使用复合索引，查询语 句中的WHERE子句必须参考复合索引中第一个列；当表中有多个关键列时，复合索引是非常有用的；使用复合索引可以提高查询性能，减少在一个表中所创建的 索引数量。

**一般来说，应该在这些列上创建索引**，例如：在经常需要搜索的列上，可以加快搜索的速度； 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；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；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；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；在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。

**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者参考的列不应该创建索引。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该增加索引。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数据类型的列不应该增加索引。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，不应该创建索 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，降低检索性能。因 此，当修改性能远远大于检索性能时，不应该创建索引。