数据库设计文档

[数据库](http://baike.baidu.com/view/1088.htm)设计(Database Design)是指对于一个给定的应用环境，构造最优的数据库[模式](http://baike.baidu.com/view/37878.htm)，建立[数据库及其应用](http://baike.baidu.com/view/2233662.htm)系统，使之能够有效地存储数据，满足各种用户的应用[需求](http://baike.baidu.com/view/195818.htm)（信息要求和处理要求）。在数据库领域内，常常把使用数据库的各类系统统称为[数据库应用系统](http://baike.baidu.com/view/1353688.htm)。

# 定义

数据库设计

　　(Database Design)是指根据用户的[需求](http://baike.baidu.com/view/195818.htm)，在某一具体的[数据库管理系统](http://baike.baidu.com/view/68446.htm)上，设计数我据库的结构和建立数据库的过程。数据库系统需要操作系统的支持。

　　数据库设计是建立[数据库及其应用](http://baike.baidu.com/view/2233662.htm)系统的技术，是信息系统开发和建议中的核心技术。由于[数据库应用系统](http://baike.baidu.com/view/1353688.htm)的复杂性，为了支持相关程序运行，数据库设计就变得异常复杂，因此最佳设计不可能一蹴而就，而只能是一种“反复探寻，逐步求精”的过程，也就是规划和结构化数据库中的[数据对象](http://baike.baidu.com/view/178571.htm)以及这些数据对象之间关系的过程。

## 设计步骤

**需求分析**

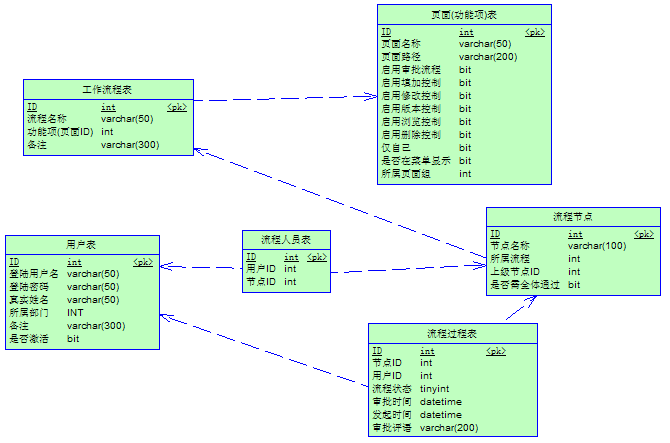
　　调查和分析用户的业务活动和数据的使用情况，弄清所用数据的种类、范围、数量以及它们在业务活动中交流的情况，确定用户对[数据库系统](http://baike.baidu.com/view/7809.htm)的使用要求和各种约束条件等，形成用户需求规约。

**概念设计**

对用户要求描述的现实世界(应用范围，如：学校，商场等)，通过对其中诸处的分类、聚集和概括，建立抽象的[概念数据模型](http://baike.baidu.com/view/8137917.htm)。这个[概念模型](http://baike.baidu.com/view/176360.htm)应反映现实世界各部门的[信息结构](http://baike.baidu.com/view/988371.htm)、信息流动情况、信息间的互相制约关系以及各部门对[信息储存](http://baike.baidu.com/view/3465313.htm)、查询和加工的要求等。所建立的模型应避开数据库在计算机上的具体实现细节，用一种抽象的形式表示出来。以扩充的实体—（[E-R模型](http://baike.baidu.com/view/1282929.htm)）联系模型方法为例，第一步先明确现实世界各部门所含的各种实体及其属性、实体间的联系以及对信息的制约条件等，从而给出各部门内所用信息的局部描述(在数据库中称为用户的局部视图)。第二步再将前面得到的多个用户的局部视图集成为一个全局视图，即用户要描述的现实世界的概念数据模型。

**逻辑设计**

　　主要工作是将现实世界的概念[数据模型](http://baike.baidu.com/view/72142.htm)设计成数据库的一种逻辑[模式](http://baike.baidu.com/view/37878.htm)，即适应于某种特定数据库管理系统所支持的逻辑[数据模式](http://baike.baidu.com/view/3353516.htm)。与此同时，可能还需为各种[数据处理](http://baike.baidu.com/view/51860.htm)应用领域产生相应的逻辑子模式。这一步设计的结果就是所谓“逻辑数据库”。



oa工作流数据库设计

**物理设计**

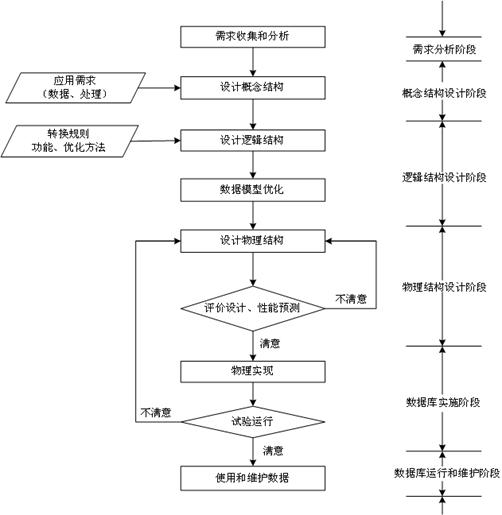
根据特定[数据库管理系统](http://baike.baidu.com/view/68446.htm)所提供的多种存储结构和存取方法等依赖于具体计算机结构的各项物理设计措施，对具体的应用任务选定最合适的物理存储结构(包括文件类型、[索引](http://baike.baidu.com/view/262241.htm)结构和数据的存放次序与位逻辑等)、存取方法和存取路径等。这一步设计的结果就是所谓“物理数据库”。

**验证设计**

　　在上述设计的基础上，收集数据并具体建立一个数据库，运行一些典型的应用任务来验证数据库设计的正确性和合理性。一般，一个大型数据库的设计过程往往需要经过多次循环反复。当设计的某步发现问题时，可能就需要返回到前面去进行修改。因此，在做上述数据库设计时就应考虑到今后修改设计的可能性和方便性。

**运行与维护设计**

　　在数据库系统正式投入运行的过程中，必须不断地对其进行



数据库设计步骤

　　评调整与修改。

　　至今，数据库设计的很多工作仍需要人工来做，除了[关系型数据库](http://baike.baidu.com/view/495393.htm)已有一套较完整的数据[范式理论](http://baike.baidu.com/view/2131801.htm)可用来部分地指导数据库设计之外，尚缺乏一套完善的数据库设计理论、方法和工具，以实现数据库设计的自动化或交互式的半自动化设计。所以数据库设计今后的研究发展方向是研究数据库设计理论，寻求能够更有效地表达语义关系的数据模型，为各阶段的设计提供自动或半自动的设计工具和集成化的[开发环境](http://baike.baidu.com/view/4831305.htm)，使数据库的设计更加工程化、更加规范化和更加方便易行，使得在数据库的设计中充分体现[软件工程](http://baike.baidu.com/view/1659.htm)的先进思想和方法。

## 设计方法

**手工试凑法**

　　设计质量与设计人员的经验和水平有直接关系

　　缺乏[科学理论](http://baike.baidu.com/view/2120462.htm)和工程方法的支持，工程的质量难以保证

　　数据库运行一段时间后常常又不同程度地发现各种问题，增加了维护代价

**规范设计法**

　　基本思想：过程迭代和逐步求精

　　典型方法：

　　(1)[新奥尔良](http://baike.baidu.com/view/52996.htm)（New Orleans）方法：将数据库设计分为四个阶段

　　S.B.Yao方法：将数据库设计分为五个步骤

　　I.R.Palmer方法：把数据库设计当成一步接一步的过程

　　(2)[计算机辅助设计](http://baike.baidu.com/view/76137.htm)

[ORACLE](http://baike.baidu.com/view/15020.htm) Designer 2000

[SYBASE](http://baike.baidu.com/view/118488.htm) PowerDesigner

## 各级模式的形成过程

　　1.需求分析阶段：综合各个用户的应用需求

　　2.概念设计阶段：形成独立于机器特点，独立于各个DBMS产品的[概念模式](http://baike.baidu.com/view/175360.htm)(E-R图)

　　3.逻辑设计阶段：首先将E-R图转换成具体的数据库产品支持的数据模型，如[关系模型](http://baike.baidu.com/view/176484.htm)，形成数据库逻辑[模式](http://baike.baidu.com/view/37878.htm)；然后根据用户处理的要求、安全性的考虑，在基本表的基础上再建立必要的视图(View)，形成数据的[外模式](http://baike.baidu.com/view/175373.htm)

　　4.物理设计阶段：根据DBMS特点和处理的需要，进行物理存储安排，建立索引，形成数据库[内模式](http://baike.baidu.com/view/176434.htm)。

## 数据库设计技巧

### 设计数据库之前

　　（需求分析阶段）

　　1) 理解[客户](http://baike.baidu.com/view/88584.htm)需求，询问用户如何看待未来需求变化。让客户解释其需求，而且随着开发的继续，还要经常询问客户保证其需求仍然在开发的目的之中。

　　2) 了解企业业务可以在以后的开发阶段节约大量的时间。

　　3) 重视输入输出。

　　在定义数据库表和字段需求（输入）时，首先应检查现有的或者已经设计出的报表、查询和视图（输出）以决定为了支持这些输出哪些是必要的表和字段。

　　举例：假如客户需要一个报表按照邮政编码排序、分段和求和，你要保证其中包括了单独的邮政编码字段而不要把邮政编码糅进地址字段里。

　　4) 创建数据字典和ER 图表

　　ER 图表和数据字典可以让任何了解数据库的人都明确如何从数据库中获得数据。ER图对表明表之间关系很有用，而数据字典则说明了每个字段的用途以及任何可能存在的别名。对SQL [表达式](http://baike.baidu.com/view/420676.htm)的文档化来说这是完全必要的。

　　5) 定义标准的[对象](http://baike.baidu.com/view/2387.htm)命名规范

　　数据库各种对象的命名必须规范。

### 表和字段的设计

　　（[数据库逻辑设计](http://baike.baidu.com/view/3158762.htm)）

　　表设计原则

　　1) 标准化和规范化

　　数据的标准化有助于消除数据库中的[数据冗余](http://baike.baidu.com/view/575274.htm)。标准化有好几种形式，但Third Normal Form（3NF）通常被认为在性能、扩展性和[数据完整性](http://baike.baidu.com/view/702953.htm)方面达到了最好平衡。简单来说，遵守3NF 标准的数据库的表设计原则是：“One Fact in One Place”即某个表只包括其本身基本的属性，当不是它们本身所具有的属性时需进行分解。表之间的关系通过外键相连接。它具有以下特点：有一组表专门存放通过键连接起来的关联数据。

　　举例：某个存放客户及其有关定单的3NF 数据库就可能有两个表：Customer 和Order。Order 表不包含定单关联客户的任何信息，但表内会存放一个键值，该键指向Customer 表里包含该[客户信息](http://baike.baidu.com/view/4289271.htm)的那一行。

　　事实上，为了[效率](http://baike.baidu.com/view/47610.htm)的缘故，对表不进行标准化有时也是必要的。

举例：在网页等游戏数据库中，为了效率会把数据库设计成一张很大的表。当然为了使数据库效率更高，会做数据库集群等配置。

2) 数据驱动

　　采用数据驱动而非[硬编码](http://baike.baidu.com/view/2024903.htm)的方式，许多策略变更和维护都会方便得多，大大增强系统的灵活性和扩展性。

　　举例，假如用户界面要访问外部数据源（文件、XML 文档、其他数据库等），不妨把相应的连接和路径信息存储在用户界面支持表里。还有，如果用户界面执行[工作流](http://baike.baidu.com/view/60285.htm)之类的任务（发送邮件、打印信笺、修改记录状态等），那么产生工作流的数据也可以存放在数据库里。角色权限管理也可以通过数据驱动来完成。事实上，如果过程是数据驱动的，你就可以把相当大的责任推给用户，由用户来维护自己的工作流过程。

　　3) 考虑各种变化

　　在设计数据库的时候考虑到哪些数据字段将来可能会发生变更。

　　举例，姓氏就是如此（注意是西方人的姓氏，比如女性结婚后从夫姓等）。所以，在建立系统存储客户信息时，在单独的一个数据表里存储姓氏字段，而且还附加起始日和终止日等字段，这样就可以跟踪这一数据条目的变化。

　　4) 每个表中都应该添加的3 个有用的字段

　　dRecordCreationDate，在VB 下默认是Now()，而在SQL Server · 下默认为GETDATE()

　　sRecordCreator，在SQL Server 下默认为NOT NULL DEFAULT · USER

　　nRecordVersion，记录的版本标记；有助于准确说明记录中出现null 数据或者丢失数据的原因 ·

　　5) 对地址和电话采用多个字段

　　描述街道地址就短短一行记录是不够的。 Address\_Line1、Address\_Line2 和Address\_Line3 可以提供更大的灵活性。还有，电话号码和邮件地址最好拥有自己的数据表，其间具有自身的类型和标记类别。

　　6) 使用角色实体定义属于某类别的列

　　在需要对属于特定类别或者具有特定角色的事物做定义时，可以用角色实体来创建特定的时间关联关系，从而可以实现自我文档化。

　　举例：用PERSON 实体和PERSON\_TYPE 实体来描述人员。比方说，当John Smith, Engineer 提升为John Smith, Director 乃至最后爬到John Smith, CIO 的高位，而所有你要做的不过是改变两个表PERSON 和PERSON\_TYPE 之间关系的键值，同时增加一个日期/时间字段来知道变化是何时发生的。这样，你的PERSON\_TYPE 表就包含了所有PERSON 的可能类型，比如Associate、Engineer、Director、CIO 或者CEO 等。还有个替代办法就是改变PERSON 记录来反映新头衔的变化，不过这样一来在时间上无法跟踪个人所处位置的具体时间。

　　7) 选择数字类型和文本类型尽量充足

　　在SQL 中使用smallint 和tinyint 类型要特别小心。比如，假如想看看月销售总额，总额字段类型是smallint，那么，如果总额超过了$32,767 就不能进行计算操作了。

　　而ID 类型的文本字段，比如客户ID 或定单号等等都应该设置得比一般想象更大。假设客户ID 为10 位数长。那你应该把数据库表字段的长度设为12 或者13 个字符长。但这额外占据的空间却无需将来重构整个数据库就可以实现数据库规模的增长了。

　　8) 增加删除标记字段

　　在表中包含一个“删除标记”字段，这样就可以把行标记为删除。在[关系数据库](http://baike.baidu.com/view/68348.htm)里不要单独删除某一行；最好采用清除数据[程序](http://baike.baidu.com/view/17674.htm)而且要仔细维护索引整体性。

### 选择键和索引

　　（数据库逻辑设计）

　　键选择原则：

　　1) 键设计4 原则为

　　关联字段创建外键。

　　所有的键都必须唯一。

　　避免使用复合键。

　　外键总是关联唯一的键字段。

　　2) 使用系统生成的主键

　　设计数据库的时候采用系统生成的键作为主键，那么实际控制了数据库的索引完整性。这样，数据库和非人工机制就有效地控制了对存储数据中每一行的访问。采用系统生成键作为主键还有一个优点：当拥有一致的键结构时，(不让主键具有可更新性)

　　在确定采用什么字段作为表的键的时候，可一定要小心用户将要编辑的字段。通常的情况下不要选择用户可编辑的字段作为键。

　　4) 可选键有时可做主键

　　把可选键进一步用做主键，可以拥有建立强大索引的能力。

　　索引使用原则：

　　索引是从数据库中获取数据的最高效方式之一。95%的数据库性能问题都可以采用索引技术得到解决。

　　1) 逻辑主键使用唯一的成组索引，对系统键（作为[存储过程](http://baike.baidu.com/view/68525.htm)）采用唯一的非成组[索引](http://baike.baidu.com/view/262241.htm)，对任何外键列采用非成组索引。考虑数据库的空间有多大，表如何进行访问，还有这些访问是否主要用作读写。

　　2) 大多数数据库都索引自动创建的主键字段，但是可别忘了索引外键，它们也是经常使用的键，比如运行查询显示主表和所有关联表的某条记录就用得上。

　　3) 不要索引memo/note 字段，不要索引大型字段（有很多字符），这样作会让索引占用太多的存储空间。

　　4) 不要索引常用的小型表

　　不要为小型数据表设置任何键，假如它们经常有插入和删除操作就更别这样作了。对这些插入和删除操作的索引维护可能比扫描表空间消耗更多的时间。

### 数据完整性设计

　　（数据库逻辑设计）

　　1) 完整性实现机制：

[实体完整性](http://baike.baidu.com/view/68269.htm)：主键

　　参照完整性：

　　父表中删除数据：级联删除；受限删除；置空值

　　父表中插入数据：受限插入；递归插入

　　父表中更新数据：[级联更新](http://baike.baidu.com/view/690054.htm)；受限更新；置空值

　　DBMS对参照完整性可以有两种方法实现：外键实现机制（约束规则）和[触发器](http://baike.baidu.com/view/71792.htm)实现机制

　　用户定义完整性：

　　NOT NULL；CHECK；触发器

　　2) 用约束而非商务规则强制数据完整性

　　采用[数据库系统实现](http://baike.baidu.com/view/3080937.htm)数据的完整性。这不但包括通过标准化实现的完整性而且还包括数据的功能性。在写数据的时候还可以增加触发器来保证数据的正确性。不要依赖于商务层保证数据完整性；它不能保证表之间（外键）的完整性所以不能强加于其他完整性规则之上。

　　3) 强制指示完整性

　　在有害数据进入数据库之前将其剔除。激活[数据库系统](http://baike.baidu.com/view/7809.htm)的指示完整性特性。这样可以保持数据的清洁而能迫使开发人员投入更多的时间处理错误条件。

　　4) 使用查找控制数据完整性

　　控制数据完整性的最佳方式就是限制用户的选择。只要有可能都应该提供给用户一个清晰的价值列表供其选择。这样将减少键入代码的错误和误解同时提供数据的一致性。某些公共数据特别适合查找：国家代码、状态代码等。

　　5) 采用视图

　　为了在数据库和应用程序代码之间提供另一层抽象，可以为应用程序建立专门的视图而不必非要应用程序直接访问数据表。这样做还等于在处理数据库变更时给你提供了更多的自由。

### 其他设计技巧

　　1) 避免使用触发器

　　触发器的功能通常可以用其他方式实现。在[调试程序](http://baike.baidu.com/view/499639.htm)时触发器可能成为干扰。假如你确实需要采用触发器，你最好集中对它文档化。

　　2) 使用常用英语（或者其他任何语言）而不要使用[编码](http://baike.baidu.com/view/237708.htm)

　　在创建下拉菜单、列表、报表时最好按照英语名排序。假如需要编码，可以在编码旁附上用户知道的英语。

　　3) 保存常用信息

　　让一个表专门存放一般数据库信息非常有用。在这个表里存放数据库当前版本、最近检查/修复（对 Access）、关联设计文档的名称、客户等信息。这样可以实现一种简单机制跟踪数据库，当客户抱怨他们的数据库没有达到希望的要求而与你联系时，这样做对非客户机/[服务器](http://baike.baidu.com/view/899.htm)环境特别有用。

　　4) 包含版本机制

　　在数据库中引入版本控制机制来确定使用中的数据库的版本。时间一长，用户的[需求](http://baike.baidu.com/view/195818.htm)总是会改变的。最终可能会要求修改数据库结构。把版本信息直接存放到数据库中更为方便。

　　5) 编制文档

　　对所有的快捷方式、命名规范、限制和[函数](http://baike.baidu.com/view/15061.htm)都要编制文档。

　　采用给表、列、触发器等加注释的数据库工具。对开发、支持和跟踪修改非常有用。

对数据库文档化，或者在数据库自身的内部或者单独建立文档。这样，当过了一年多时间后再回过头来做第2 个版本，犯错的机会将大大减少。

（动态项目就没做相应的文档，导致别人看数据代码的时候一头雾水。）

　　6) 测试、测试、反复测试

　　建立或者修订数据库之后，必须用用户新输入的[数据测试](http://baike.baidu.com/view/106296.htm)数据字段。最重要的是，让用户进行测试并且同用户一道保证选择的数据类型满足商业要求。测试需要在把新[数据库](http://baike.baidu.com/view/1088.htm)投入实际服务之前完成。

　　7) 检查设计

　　在开发期间检查数据库设计的常用技术是通过其所支持的[应用程序](http://baike.baidu.com/view/330120.htm)原型检查数据库。换句话说，针对每一种最终表达数据的原型应用，保证你检查了[数据模型](http://baike.baidu.com/view/72142.htm)并且查看如何取出数据。