**Relational DataBase Study Note**

2014-4-14

目录

[2014-4-14 3](#_Toc385283796)

[Introduction to database design 3](#_Toc385283797)

[第一步：确定实体（Identifying Entities）。 3](#_Toc385283798)

[第二步：确定关系（Identifying Relationships） 4](#_Toc385283799)

[第三步：标识属性（Identifying Attributes） 4](#_Toc385283800)

[第四步：创建实体关系图（ERD） 4](#_Toc385283801)

[第五步：确定键值（Assigning Key） 5](#_Toc385283802)

[第六步：找到合适的数据类型（Defining the Attribute's Data Type） 5](#_Toc385283803)

[第七步：规范化（Normalization） 5](#_Toc385283804)

[第八步：数据词典（Glossary） 5](#_Toc385283805)

# 2014-4-14

## Introduction to database design

**Writing before**:When you design database, must know “What do I need to know”. If you can’t do it,congratulation！！！，you dead meat,because if you find out later that you forgot something, usually you need to start all over. *Adding things to your database is mostly a lot of work*.

注：写英文还是太勉强了。这段就当做练习。

数据库设计要遵循“八荣八耻”，首先我们在这里说说“八荣”，所谓的“八荣”就是一个合理化数据库设计的八个步骤。

### 第一步：确定实体（Identifying Entities）。

我们需要知道什么是实体，实体就是每一种数据库保存“对象”，个人理解，数据库实体并不一定是一张表，可能是两个或者更多个表所表达的一中数据类型，一类事物等。

以下是<http://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html>举得一个例子，我认为很好的说明了这个事情“当一个网店设计的时候，有顾客，商品，卖家，商店等实体，但是数据库可能会存在很多张数据表”



### 第二步：确定关系（Identifying Relationships）

确定实体与实体之间的关系，根据实际情况（业务形态和流程），建立数据之间的“流通”关系。

（1）、建立实体与实体（每一个实体）之间的关系，一对多，一对一，多对一，多对多。

（2）、调转关系，整理清理，使用1，N，M三个符号表示。

注：有些实体可能存在着自关联的情况，也就是“递归关系”。

（3）、消除冗余关系，也就是消除数据库关系闭环，消除间接实体关系关联。

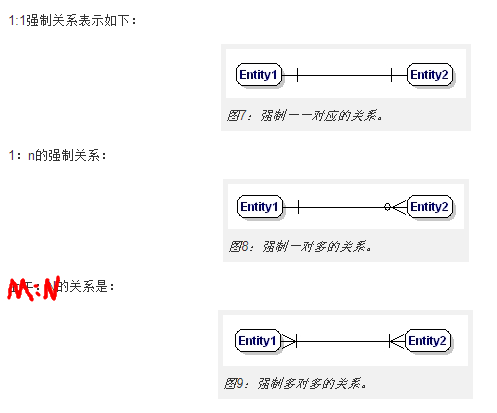
（4）、解除M:N关系，可以通过建立新的实体（虚拟实体）来解除M:N的关系。

### 第三步：标识属性（Identifying Attributes）

也就是建立实体属性，换做面向对象思想就是抽象对象所具有的一些属性（我晕，还是属性）。也可以简单的理解为创建column。

### 第四步：创建实体关系图（ERD）

符号ER图：实体使用矩形，实体间关系使用连线表示，如下图



### 第五步：确定键值（Assigning Key）

主键：唯一标识。

外键：建立关系的重要手段。

### 第六步：找到合适的数据类型（Defining the Attribute's Data Type）

这个在此不多说，基本原则：固定长度为已知长度属性服务，非固定长度为可伸缩属性服务，根据数据库分配的数据文件有略微差别，锁定标准文件大小为60K/个，所以在分配字段长度是应该加以考虑。

### 第七步：规范化（Normalization）

关于规范化的东西在此不作记录，以后会专有时间进行研习。

### 第八步：数据词典（Glossary）

数据库设计中的每一个实体，属性，数据类型的介绍。

# 2014-4-15

## 数据库命名规则

1. 使用英文名字：例如FirstName，HomeZIPCode等。
2. 在名称中包含对象类型：例如tblEmployee（这个是网站上的，我个人建议都统一使用两个字母表示对象类型，tb表、sp存储过程、vw视图、fn函数），spGetSalary等。
3. 不要使用空格：在数据库名称，表名以及字段名中都应该保持没有空格，可以使用“-”或者“\_”代替，但是要保持风格一致。
4. 如果不是短暂性的字段就不要用时间标识：如字段Name2011，除非这个字段仅仅在这一年使用，否则请不要使用时间标识。
5. 简约之美：字段，表，数据库都一样，在使用少的单词能够表达时，则不需要添加任何前缀或者后缀。
6. 不要使用数字后缀：除非特殊要求，否则不要再字段中存在数字后缀，此规则与规则4有几分相似。就是说尽量不要用数字进行一些标志。
7. 寻求一下场外求助：如果你不确定这个命名的正确与否，可以问问别人的意见。

以下是我个人添加的命名规则意见：

1. 不要使用关键字进行命名，比如name字段，users表等。虽然不会有错，但是你会制造很多麻烦。

# 2014-4-16

## 标准化数据库

第一范式：数据表中的每一个数据项都为不可再分的基础项。

注：“姓名”=“姓”+“名字”？？，这样是否正确呢？所谓的每一个数据项都为不可再分的数据项，实际表现为，在开发过程总，每一个数据项都是独立表达一个数据的意义，而不会出现在开发过程中需要经常把一个数据项进行split使用。

第二范式：首先要满足第一范式，其次，每一个非主属性完全函数依赖主键。

注：第二范式就是要求要有主键。关于主键的特性在此不多说，如果认为“性别”能成为主键，那就去尝试一下，数据库开发工具这一关都过不去。

第三范式：在满足第二范式的同时，保证非主属性不传递依赖于主属性。

比如某表有“学号”，“姓名”，“所在系”，“系主任名称”。

学号🡪所在系—>系主任名称。

这个表存在不合理主键关系，学号🡪系主任名称。因此要做表的分离。

**BC范式和第四范式在此不作说明**

以上都是方法论，当然如果你有一些好的理由可以打破这些规则。以下说说这些合理的理由。

**理由一**：超耗能的连接，在开发过程中有些时候为了显示某些数据，不得不进行多表连接，而这样做的同时不存在任何数据运算和逻辑运算。如果经常这样做，可以考虑一下是否可以创建一点数据冗余，前提是这些冗余不会造成脏数据。

**理由二**：简化设计难度，如果你在设计或者开发过程中，尝尝发现想要读取一个数据或者进行数据运算组合时候总是一个坑，或者是非常难，难于上青天，好吧，不要为了一点点规则而大大加剧你的成本，要知道，在你设计的时候已经如此困难了，以后维护或者其他人来看你的数据库的时候，估计只会有一堆脏话留给你了。

**理由三**：仅仅是为了做一个快速原型产品，好吧，既然如此就别去想太多了，怎样方便怎么做，但是如果你想做成一个产品，就不要忘记回头看看你的设计，适时做一些整理，否则你就只能在做产品的时候重新设计数据库了。

## 标准化基础知识

定义：数据库标准化包括消除冗余（同样的数据存在多个表中）和数据完整性。

不得不说的第四范式：看了课本和网上的一些资料，是我文学功底不够还是他们说的不是正常话，个人理解就是通过分表的方式去掉组合主键情况。

# 2014-4-18

## 选择一个优秀的主键：

规则一：唯一标识。

规则二：非隐秘性字段或者不是具有重要意义的字段。

建议：可以交给数据库管理，数据库都可以生成主键字段，除了用来标识没有任何意义。

# 2014-4-21

数据库设计十项通病：

1、简单草率的设计/计划

2、无视标准化规则

3、没有标准或者poor的命名规则

4、缺乏文档

5、一个表来保存所有域值

6、使用 guid列作为你唯一的key

7、不使用数据库来保护数据的完整性

8、不使用存储过程来访问数据

9试图建立通用的对象

10、缺乏测试