**数据库设计解决方案**

**入门经典**

**日期：2017年11月22日**

**作者：高志超**

**第一部分 数据库和数据库设计介绍**

# 第1章：有效数据库设计的目标

## 1.1 理解数据库设计的重要性

应用程序设计中数据库设计是最为关键的一项任务

数据库是信息的存储库，供应用程序的其他部分进行管理并显示给用户。

无用输入无用输出(GIGO)原则完全适用

## 1.2 信息容器

数据库是什么？

数据库是一种存储数据的工具，允许以某种方式创建、阅读、更新和删除数据。

名片、笔记本、档案柜和大脑都是信息容器。

## 1.3 信息容器的优缺点

## 1.4 理想数据库特性

### 1.4.1 CURD

创建(Create)、读取(Read)、更新(Update)和删除(Delete)

### 1.4.2 检索

检索(Retrieval)是读取的另一个名词，既CRUD中的R。

数据库应该允许查找每条信息

数据库应该允许组织数据以便可以找到特定的数据段

### 1.4.3 一致性

CRUD中的R的另一个方面是一致性，既提供一致的结果。

### 1.4.4 有效性(验证)

### 1.4.5 轻松的纠错

### 1.4.6 速度

良好的设计数据库能够快速创建、读取、更新和删除记录

### 1.4.7 原子事物处理

事务处理要么其所属操作全部完成要们什么都不发生，而不能只进行一部分操作

### 1.4.8 ACID

Atomicity(原子性) Consistency(一致性) Isolation(隔离性) Durability(持久性)

原子性是指事物处理是原子的，事物处理中的操作要么全部完成，要么都不执行

### 1.4.9 持久性和备份

数据必须是持久的

良好的数据库设计有助于更为容易地创建备份

### 1.4.10 低成本和可扩充性

# 第2章：数据库类型

## 2.3 关系数据库

关系数据库包含保存行和列的表

每一行均保存了有关特定实体的相关数据，每一列均表示有关该实体的一条信息。

特性：

数据类型

基本约束

参照完整性

级联删除和更新

连接

复杂的查询

适用于如下情况：

需要在不同的表之间执行复杂的查询和连接

需要执行数据有效性验证

需要为特定的数据段考虑任意数量的值

希望能灵活地构建在开始设计项目时本没有计划的新查询。

不适用于如下情况：

需要使用一种特殊的数据拓扑来执行应用程序的主要功能

## 2.4 电子数据表

## 2.5 分级数据库

## 2.6 XML

扩展标记语言是一种用于存储分级数据的语言。

## 2.8对象数据库

## 2.9 对象关系数据库

# 第3章：关系数据库基础知识

## 3.1 关系视角

关系数据库非常强大，是当今计算机领域最常用的数据库类型

## 3.2 表、行和列

可以将关系数据库视为表的集合，每个表又包含行和列。

列的数据类型是指该列能保存的数据类型

## 3.3 关系、属性和元组

列的正式术语是“属性”或“数据元素”

行的正式术语是“元组”，英文为tuple。包含3个属性的关系视为保存值的三元组

## 3.4 键

关系数据库术语包括大量不同形式的键。

键是用来查找表中行的一列或多列的组合。

超键是表中一个或多个列的特定组合

主键是一种实际上用来在表中唯一标识或查找行的超键

超键：没有两行具有完全相同值的一组列

主键：数据库用来快速定位记录的唯一键

## 3.5 索引

索引是一种数据库结构，可以更快且更容易地基于一个或多个字段中的值查找记录。

索引不等同于键

创建和维护索引会花费数据库一些额外的时间，因此不应该随意地创建索引

请索引最可能需要搜索的字段而不要费事索引那些不太可能需要搜索的字段

## 3.6 约束

对表中允许的数据施加限制

基本约束：字段必须(非空约束)、特殊值Null、阻止一个字段保存类型不匹配的值

检查约束：age>0

主键约束：不存在完全相同的两个记录，并且构成主键的所有字段都有非空值

唯一性约束：

## 3.7 数据库操作

关系数据库的8种最初操作：

选择(selection)：选择表中的一些或者全部记录

投射(projection)：从一个表中删除列

联合(union)：联合具有相似列的表并删除副本

相交(intersection):查找两个表中相同的记录

差值(difference):在一个表中选择另一个表中没有的记录

迪卡尔积(cartsian product):创建一个新的表

连接(join):类似于迪卡尔积，不同之处是在他们满足某一个条件的情况下，一个 表中的记录才与另一个表中的那些记录配对。

相除(divide):与迪卡尔积恰好相反。

关系数据库的主要操作是连接，并常常同选择和投射操作结合使用。

**第二部分 数据库设计过程和技术**

# 第4章：了解用户需求

## 4.1 制定计划

常见次序：

拿出一个问题清单

与客户会面

了解重要的人物

向客户请教

设身处地为用户考虑

调研现行业务

集体讨论

考虑未来

搞清楚客户的思维过程

了解客户真正需要什么

区分优先次序

证实了解的情况

撰写需求文档

编制用例

确定可行性

## 4.2 拿出一个问题清单

4.2.1 功能

期望系统实现什么功能，此外还要涉及如何实现这些功能

记录客户提出的系统应该如何工作的想法

系统应该实现什么功能

构建此系统的原因是什么，你希望它完成那些任务？

系统应该呈现什么样的外观？请大致绘出用户界面

系统的不同部分分别需要多少响应时间

### 4.2.2 数据需求

用户界面需要什么数据

应该从哪里获取数据

这些数据字段是如何关联的

如何处理这些任务？

数据来自何处？

### 4.2.3 数据完整性--完整性约束

各个字段允许获取那些值？

需要那些字段？

各个字段的有效域(允许的值)是什么，格式？多长？是否需要处理？

在下订单前是否需要客户记录？

若客户取消一个账户，是希望删除对应的记录？还是仅将他们标记为非活动的？

系统需要何种级别的可靠性？

是否需要全天候访问系统？

需要采用什么样的备份频率？

如果系统崩溃会带来多大的灾难？

需要多快的备份和运行速度？

如果发生崩溃丢失一些数据将会带来何种后果？

### 4.2.4 安全性

有助于确定哪种数据库产品和使用何种体系结构

每个用户是否需要单独的口令？

不同的用户是否需要使用不同的数据段？

数据库内的数据是否需要加密？

是否需要提供审计线索来记录执行的每项操作和执行人？

有那些不同类别的用户？职员、主管和超级用户

### 4.2.5 环境

该系统是增强还是替代现有的系统

是否存在描述现有系统的文档资料

现有的系统是否有可供调研的纸质表单

需要现有系统中的那些特性？不需要那些？

现有的系统使用何种类型的数据？存储方式如何，以及不同数据段的关联方式

是否存在说明现有系统的文档？

是否存在现有系统必须与之交互的其他系统

它如何与这些系统正确交互？

新项目是否向现有的系统发送/接收数据？如何发送/接收？

是否存在说明这些系统的文档

业务的运作方式如何

### 4.3 与客户会面

了解客户需求

### 4.4 了解重要人物

并非所有人都对产品有相同的想法，你需要搞清楚那些客户是领导，那些是项目参与者，

那些人对项目没有发言权

执行负责人--推动项目的最高等级的客户

客户负责人--充分了解客户的需求

客户代表--指派向你回答问题并帮助定义项目的某个人

利益相关者

助手/信息提供者

### 4.5 向客户请教

与客户负责人交谈并了解客户内心需要的东西

### 4.6 设身处地为用户着想

### 4.7 调研现行业务

### 4.8 集体讨论--直到确定项目需求

### 4.9 考虑未来--考虑未来的需求

### 4.10 搞清楚客户的思维过程

有些客户可能有这方面的了解，听听他们的想法

记住客户看问题的角度与你不同，对于客户提的意见不要立即否决。

### 4.11 了解客户真正的需求

### 4.12 区分优先次序--不必告诉客户

第一优先级特性是那些必须出现在项目中的内容，必不可少的基本要素

第二优先级特性是第一个版本使用并且有时间开发下一个版本之前允许确实那些特性

第三优先级特性是客户认为很美妙但是没有版本1和2特性重要的那些特性

### 4.13 证实了解的情况

### 4.14 撰写需求文档

需求文档描述了你准备构建的系统，此文档有时也成为产品需求文档(PRD)、需求规格说明、说明书。

需求文档需要阐明你计划构建什么东西以及它将实现那些功能。

该文档需要说明将解决的问题并且应该描述客户如何使用它来解决问题。

在后期设计和开发过程中，需求文档可以帮助所有人顺利工作，避免用户的指责。

### 4.15 编制用列

### 4.16 确定可行性

# 第5章：将用户需求转换成数据模型

5.1 数据模型的概念

模型自身不能做任何事情，它并不构建数据，是一种方案，是用于构建某种事物(数据库)的蓝图

它提供了考虑将要构建的数据库的一种具体方法。

## 5.2 用户界面模型

首先应该大致绘制出可视界面的草图

## 5.3 语义对象模型(SOM)

概念有些技术性，但是它们仍与人们思考实物的方式紧密关联，因此语义对象模型对用户而言相对容易理解

### 5.3.1 类和对象

语义类是由能力之别某个特定实体的一个已命名的属性集合。PERSON类

语义对象（SO）是某个语义类的一个实例.它是一种实体实例，具有通过填写的类定义的所有属性。

属性表现为3种方式：简单、组和对象

简单属性保持简单值，如字符串、数字或日期

组属性保存合成值——由其他多个值构成的值。例如，Address属性可以包含Street、Suite、City、State和ZipCode。

对象属性表示于其他某个语义对象的关系。

### 5.3.2 基数

一个属性的基数是指对象可以具有的该属性的值的范围。

可以在属性的旁边写下由一个句点分割的属性值上限和下限。

通常来说，如果值是可选的，则最小的基数是0；如果值是必需的，则最小基数是1；如果最多只运行一个值，那么最大的基数是1；如果运行任何数量的值，那么最大基数是N。

最常见的基数最可能是以下几种：

1.1：恰好只要求一个值

1.N: 可以取任意数量的值但至少必须有一个值

0.1：一个可选的值

0.N：任意数量的可选值

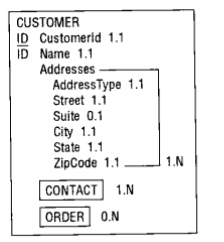
### 5.3.3 标识符

对象标识符是用户用来标识类中对象的一个或多个属性的组合，可以包括单一属性

可以通过在其属性左边下下文字ID来指示标识符

可以通过对其左边的ID加下划线来表示一个唯一的标识符

### 5.3.4 组合在一起



图中用一个大的方框围住整个类的定义，顶部是类名称。

CustomerID是一个用于标识客户的简单属性，因此它采用ID符号表示法。CustomerID值是唯一的，因此ID下加下划线。该值是必需的并且一个客户只能有一个ID，因此它的基数是1.1。

用户有时希望通过名称来搜索客户，因此Name属性也是一个标识符。但是两个客户可能具有相同的Name，因此这里的ID不加下划线

Addresses属性中除了Suite属性外，所有属性都是必需的并且只能保存一个值，而Suite是可选的。图中的线条给出了包含在Addresses值内的属性。属性右下角1.N指示CUSTOMER对象必须有一个或多个Addresses值。

最后，该类具有两个分别名为CONTACT和ORDER的对象属性。CONTACT属性表示客户的一个或多个联系人。该属性周围的方框指明这是一个对象属性，它的技术1.N指示CUSTOMER必须至少有一个联系人。

客户首次创建时它不会有订单记录，所以ORDER的基数是0.N。

### 5.3.5 语义视图

有时定义用于查看相同数据的不同视图是有用的，例如根据权限不同查看到不同信息。

### 5.3.6 类的类型

简单对象、组合对象、复合对象、混合对象、关联对象和继承对象

## 5.4 实体关系模型(entity-relationship diagram,ERD)

是另一种形式的对象模型

类似语义对象模型，它还允许你表示对象及其关系

ER图的关注点不同，它更强调关系而对类结构的关注稍微弱一些

### 5.4.1 实体、属性和标识符

实体和语义对象类似，标识希望在对象模型中跟踪的某种事物的具体实例

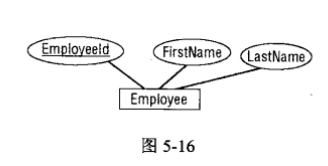
实体可以是物理对象（员工、订单）或逻辑抽象（约会、讨论）

相似的实体组成实体类或实体集合，员工实体Bowb\gnick属于Employee实体集合

实体包括属性来描述他们表示的对象

绘制实体集合：

* 第一种方法：一个实体集合包含在一个矩形框中，它的属性绘制在椭圆形内并使用线条连接到集合。如果其中一个属性是标识符(键或主键)，则在其名称下加下划线。



这种表示法的一个问题是它占用过多的空间。

* 第二种方法：与语义对象模型使用的方法类似，只需要将集合的名称放在ER图中。线条和其他简略描述的符号通过连接实体集合来说明它们的关系。

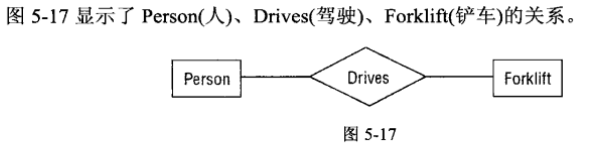
### 5.4.2 关系

ER图用一个含有关系名称的菱形来指示关系。

名称通常是极富有描述性的词语，如果名称不够说明问题，则可以给关系添加属性

就像给实体添加属性一样：将属性放在一个椭圆形中并用一个直线将它连接到关系。

通常实体是名词，关系是动词



### 5.4.3 基数

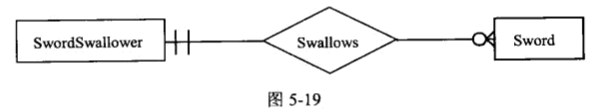
为了添加基数信息，ER图可以向进入和离开实体集合的直线添加三种符号中的一种或多种

三种符号说明如下：

环：环(圈或椭圆)表示0

线条：一条短线(虚线或竖线)表示1

鱼尾纹：鱼尾纹(圆锥形帐篷或其他叫法)表示许多



ER图只能使用3中符号来表示3中基数：0、1和许多。

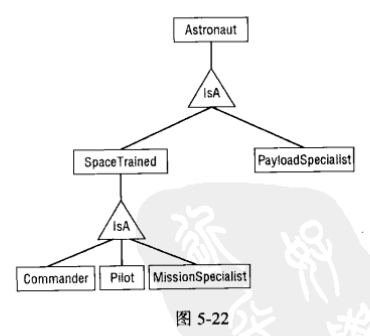
这些模型的目的是为了对系统有所了解，而不是严格遵循规则而得到可笑的结论，因此我认为应该结合这两种方法的优点，并按照语义对象模型的方式使用ER图来制定基数。



### 5.4.4 继承

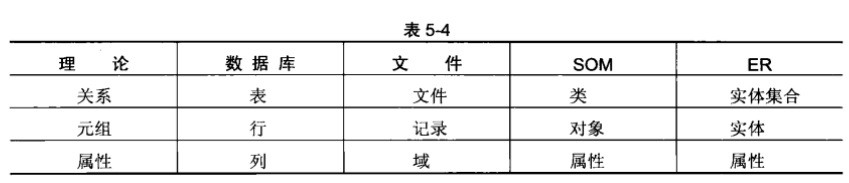
和语义对象模型类似，ER图可以表示继承。

ER图将继承性表示为一种特殊的关系，成为IsA（读作“is a”）,画在一个三角形内部，三角形的一个顶点指向父类，其他引入三角形的线条连接到三角形的侧边。



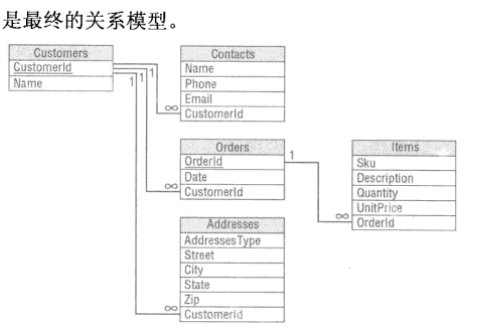
## 5.5 关系模型

理解第3章中介绍的概念和本章目前为止介绍的概念之间的映射方式后，将语义对象模型和ER图转换成关系形式就不太困难了。



为了将语义对象模型和ER图转换成关系模型，只需将类或实体集映射到表。然后找出表中的哪些列构成表之间的外键关系。

为每个实体构建一个表，使用弗雷的主键作为子类的外键以便按照一对一的关系联系这两个类。



# 第6章：提取业务规则

## 6.1 业务规则的概念

优化数据库的第一步是提取业务规则。

业务规则描述业务认为重要且值得记录的对象、关系和操作。

它们包括用来定义企业操作和处理日常业务的方式的规则和策略。

所有的这些规则都是以某种形式作为约束现实的。

## 6.2 识别重要的业务规则

P108

## 6.3 提取重要的业务规则

## 6.4 多层应用程序

# 第7章：规范化数据

7.1 规范化的概念

7.2 第一范式(1NF)

7.3 第二范式(2NF)

7.4 第三范式(3NF)

7.5 至步于第三范式

7.6 Boyce-Codd范式(BCNF)

7.7 第四范式

7.8 第五范式

7.9 域/键范式(DKNF)

7.10 必要的冗余

7.11 最佳的规范化级别

第8章：设计数据库以支持软件应用程序

3.1

3.2

3.3

3.4

第9章：常见的设计模式

3.1

3.2

3.3

3.4

第10章：常见的设计陷阱

3.1

3.2

3.3

3.4

第三部分 详尽的案例分析

第11章：用户需要和要求

3.1

3.2

3.3

3.4

第12章：构建数据模型

3.1

3.2

3.3

3.4

第13章：提取业务规则

3.1

3.2

3.3

3.4

第14章：规范化和优化

3.1

3.2

3.3

3.4

第四部分 实现数据库

第16章：MySQL

3.1

3.2

3.3

3.4

第五部分 高级主题

第17章：SQL简介

3.1

3.2

3.3

3.4

第18章：使用SQL脚步来构建数据库

3.1

3.2

3.3

3.4

第17章：数据库维护

3.1

3.2

3.3

3.4

第17章：数据库安全

3.1

3.2

3.3

3.4