**第一章 介绍**

**为何我们需要这本书？**

根据我们的咨询经历，我们发现目前很多公司依旧关起门来发展自己的数据模型，几乎没有外部参考材料。这样一来，无论是聘请经验丰富的顾问还是动用内部员工来开发系统设计的重要组件，都会增加一大笔的成本费用。拥有更加客观的参考材料是非常有必要的，因为一个机构可以用这些参考资料来检测它的数据模型和数据库设计，同时还可以为数据模型或者数据库结构提供备选方案。这本书充分的扩展了目前的《数据模型资源手册》第1卷和第2卷（Wiley出版社，2001年）中的工具，为企业高效开发高质量的数据模型提供了更全面的指导。

《数据模型资源手册》第一卷回答了这样一个问题“我们在哪里可以找到这样一本书，它可以给我们介绍一个模拟常见的数据模型结构的标准方法？”该书第一卷为我们公用数据区域提供了一个广泛的模板数据模型库，公用数据区包括个人，组织，产品，订单，装运，发票，会计及预算，人力资源，等等。同时它也提供模板模型，专门服务于数据仓库模型，销售分析，人力资源和库存管理分析，等等。

该书第二卷，延续着第一卷的思路，继续拓展这些模板模型，与此同时，也添加了一些新的数据模型结构，其目的在于更好服务于某些特定行业，比如说制造业，通讯，医疗保健，保险，金融服务，专业服务，旅游，电子商务零售行业。

尽管通过这个系列前两本书的学习，个人和组织已经大大提高了他们数据模型的质量，也节省了大量的人力物力和宝贵的时间，可是当我们已经完成了这些模型的时候，问题还是一如既往地出现了。 “我们怎样才能快速扩展，并按客户具体要求制定这些模型使得其可以快速开发出任何高质量的数据模型，以满足于我们的组织，我们的需要，即使它对企业而言是特别的。此外，许多组织都希望坚持用一个标准的方式来创建通用的的数据结构。他们经常说，“我们成为不了第一个问如何扩展我们的数据模型或者学会以相同的思想来构建新的模型类型的人。当然，这些别人都已经做过了。模型资源手册的第3卷，阐述了这些问题并提出了一些担忧。这本书完全继承了先前出版书籍的精华，并探讨了适用于所有的数据模型的共同的底层结构。

我们有一个很有用的拇指规则，似乎适用于大部分数据模型：一个数据模型的三分之一通常包含适用于大多数机构的通用结构，还有三分之一通常适用于具体行业，最后三分之一只适用于某个特定机构。大部分情况下，《数据模型资源》的第一和第二册都把注意力放在了这个规则的前两个“三分之一”上。在数十年的数据建模经验中我们还发现，有些很平常的模型适用程度大大超过大部分数据模型结构的50%，而且还可以进行重复使用。比如说，一个订单的状态可以以同样的方式作为一个人或组织的状态。产品和人的分类遵循同样地模式，这里先不管一种分类是关于产品，还是关于人。

这本书的一个好处是，它不但解释和增强了《数据模型资源手册》一二卷所用的基本模式。且远远不止这些，因为本书中所列举的数据模型模式适用于通用结构，这些通用结构是可适用于所用企业、特定行业的数据模型结构，同样适用于某些企业的特定数据模型结构。通过重复使用这些通用的数据模型模式，这本书提供了用来快速一致地为多种数据请求建模的模版。那么，这对于帮助整合数据、分享数据、将数据变作一种有价值的战略资产具有巨大而积极的影响。

通用数据模型和数据模型通用模式的差别在于，通用数据模型适用于非常常见的模型，而通用模式几乎可用来拓展和开发任何类型的数据模型。为了把问题阐释清楚，我们用家具来打个比方吧。让我们先来想一下鸠尾榫接头的设计。鸠尾榫接头是一种连锁装置，可以用来制造各种家具（这与通用模式很像）。现在再让我们想一下一整套桌椅的设计（桌椅的设计和通用数据模型很像）。现在你就知道这两个概念的关系了。很多通用数据模型都是以通用模式为基础的。前两本书为非常常见的数据模型提供了具体的例子。那些常见数据模型可以重复利用，比如说装运，订单，发票等模型。与此相反，数据模型的通用模式则提供的是底层结构的构建模块，以使建模人员可以重重复利用这些构建模块来建立任何模型，甚至可以使用这些构建模块构建非常特定的模型。

模式可以用于快速开发和/或修改通用模型和行业模型或者用于开发全新的模型。每个模式都有关于如何实现该模式的现实生活中的例子。任何组织都可以使用这本书中出现的通用模式作为指导方针和一套标准，这样建模人员能够坚守提高数据模型的一致性，并节约大量开发和维护的时间以及提高模型的质量。任何企业的专业人士均可以使用第1卷和第2卷中的模版模型快速启动数据建模，然后使用第3卷中的模式以一致的方式建立这些常见的模型，前提是你确信这些模式是准确可靠的，是适用于现实生活的。我们的许多客户已经以不同的方式使用了这些模式，例如：

* 为IT专业人员在构建数据模型时提供一个可遵循的标准。正是这个标准让他们保持了数据模型的一致风格，并使得后续的数据结构能够在他们的组织中跨越不同的数据库。
* 可以作为标准的工具箱帮助数据专业人员建立/拓展他们的数据模型。该模式涵盖了数据建模人员所要解决的诸多典型问题。当模式已经为我们提供了不同风格的解决方案时，也就是提供了各有利弊的有效替换选择方案，为什么还要再度解决这个问题？
* 可以为一个标准，该标准能够用作通用数据结构的基础，它允许开发人员或编程人员为这些基于模式的共同结构创建标准接口或根据这些结构创建标准接口。这意味着编程人员可以“编写接口”并且不用过多担心很多底层数据语义和数据结构的处理。
* 当客户购买软件或其他标准数据模型时，能作为他们的有用工具。模式可以设置数据需求，供应商的数据模型和数据库必须上升到很常用的数据需求。例如：模式可以指定一个解决方案，该方案需要支持非常灵活的分类法，这种分类法在不改变数据模型和数据结构的情况下允许新的分类类型。您正在寻找的产品在维护分类方面是否适应这种类型的灵活性。它使用了灵活的模式吗？如果没有的话，当你需要时它又如何提供适当水平的灵活性？
* 作为一个客观的来源，该来源针对一个企业之前的系统开发工作，能够评估和检查企业的数据模型，所以它能够评估备选方案。
* 作为企业的数据专业人士和一般IT职员的培训教材。模式涵盖了在不同泛化层级的各种结构。使用了实例来详细的解释模式，在使用过程中这些范例能够指导数据建模人员和其他IT专业人士。

我们的很多客户已成功使用这些模式为各种各样的数据建模工作节约了时间并提高了质量。这些工作的范围从为原型创建数据模型，到开发用于使他们的模型全球化的企业级的数据模型。

**拓展数据建模学科：**

数据建模一直是一门学科，它作为学科得到首次认可是在博士peter chen 1976年的一篇文章中，该文章阐释了描述数据结构的方法，该方法被称作实体关系建模(Entity-Relationship Modeling)。从那时开始该方法便成为数据库建模和设计的标准方法。数据库设计人员可以通过正确构建组织机构的数据模型来消除数据冗余问题，而冗余数据也正是导致不准确信息和低效率系统出现的一个关键原因。

有很多关于设计模式的书籍和文章，但是却很少写到关于实体关系建模（正如我们这本书中所描述的）的基本模式。据说模式之父或者说之母是Christopher Alexander、Sarah Ishikawa 和Murray Silverstein，他们写了《模式语言》：城镇、建筑、施工2。这是一本关于架构的书，里面收集了很多模式，这些模式被作为制作城市建设规划问题解决方案的基础。很多程序员喜欢这本书当中的概念以及他们简化创建可重用代码的过程。另一个开创性的工作称为设计模式：“四人帮”（ Erich Gamma, Richard Helm, Ralph Johnson, 和 John Vlissides）写的《可复用的面向对象软件的原理》通过使用通用接口为解决编程问题提出了通用解决方案。

《数据模型资源手册》4的前两卷和David Hay的杰作《数据模型模式》5均包含可重用数据模型，这些模型可以满足常见的数据建模需求，例如：如何为当事人、产品、订单或者合同等建立数据模型。很多人认为这些书在数据建模的上下文中为建模提供模式。然而，这些书所讨论的东西与本书所讨论的内容是截然不同的。在这本书中，我们所使用的术语“模式”具有十分不同的含义，我们将某个特定应用的可重用数据模型（包含其他书籍中的可重用的模型）和不依赖于任何一种应用的底层核心模版做了区分，核心模版我们称之为数据建模的通用模式，是这本书的重点。虽然数据库建模方面已经存在许多标准，但我们仍需推动数据库建模向下一步前进：以方便的格式提供带有范例的通用数据模式库，以使得他们能够被重用。重用这些模式库也是这本书的目的。

**何谓模式？何谓通用模式？**

一般而言，模式是指能够作为制造其他事物的指南的事物。6 数据建模当中的模式可以说是一个模版，能够作为开发数据模型的一个指南。例如。在第6章中，状态模式提供了为任何类型的实体建模的指南或标准。因此，状态模式能够适用于当事人（PARTY）、产品（PRODUCT）、订单（ORDER）、发票（INVOICE）或各种状态的其他实体。当事人实体可能具有激活或未激活等状态。产品实体可能具有“引进的”、“断货”等状态。我们可以使用相同的“模式”来为以上这些实体的状态建模。

单词*通用*（*universal*）在韦伯词典中的定义是“适用于各种不同的用途”：渗透、影响或拓展到整个领域。数据建模的通用模式是可复用的指南，它为数据建模工作中出现的非常盛行或通用的主题提供了数据模型模版。这些模式的用途是他们能够应用于不同应用中，且所提供的全面的（也就是通用的）观点为大多数企业节省了时间和精力。

基于数十年的数据建模经验我们发现同样类型的模式会不断地出现在数据建模的工作中。在这本书中我选择了那些我们认在数据建模中是最常用的“通用的”模式。我们还发现在大多数组织机构中大部分数据模型与角色（当事人所扮演的）、分层和递归、分类、状态、联系机制以及商务规则相关联。因此，我们为每个类型的数据都提供了模式。虽然还有其他类型的数据建模模式，但是我们选择了这些模式，因为我们相信这些模式是最为常用的，自然而然能够带来最大的益处。

我们在每一章节介绍一种模式并为该模式提供了备选方案以及将该模式应用到具体数据需求的范例。每个备选方案都为同类型数据模型提供了不同泛化层级的模式。例如：在状态这一章，我们提供了非常特定的模式来将状态作为属性建模，在另一个不是十分特定的模型中有一个*状态类型（*STATUS TYPE*）*实体，一个较为通用的模型允许一个实体具有任何数量的状态类型。一个更为泛化的模型提供了一个*状态应用*（STATUS APPLICATION）实体，该实体允许所有需要状态的实体与这个通用的数据模型结构相关联。

**模式的意义是什么？**

数据建模的通用模式类似于工程师建设桥梁的工程图纸。工程师建造任何类型的吊桥都会有基本的工程蓝图。每当需要建造独特的吊桥时，工程师并不试图想出全新的解决方案；他或她使用现有的设计模式。桥的外形可能各有迥异，但是很多的底层结构是相同的。例如，日本东京的密佑高架桥和纽约的布鲁克林大桥两者均是使用了相同基本设计模式的高架桥。

数据建模的通用模式为建立非常通用类型的数据模型提供了有效的实践和备选方案。底层数据模型展示了当事人*（PARTY）*如何与发票*（INVOICE）*关联，其关联方式和当事人*PARTY(s)*与装运*SHIPMENT(s)*关联的方式相类似，也和当事人与支付*PAYMENT(s****)***、协议*AGREEMENT(s)*、或其他实体相关联的方式相类似。例如：在特殊事务的上下文中当事人（人员或组织机构）可能具有某些角色或者是关于另一个实体的角色的某些角色，现有非常通用的方法为该类型的数据需求建模。我们称这种特例为上下文角色模式*(Contextual Role Pattern)*。另一个示例是项目*（*PROJECT*）*的状态或金融交易（TRADE）的状态基本模式是相同的，只是这两个模式分别适用于不同类别的数据。我们称这种模式为状态模式*（Status Pattern）*。当你遇见一个状态或者一个“上下文角色”时，在设计模式现存的情况下为何还要去尝试创建一个新的数据结构？

正如我们前面所说，本书的前两卷专注于提供通用行业的模版数据模型结构以帮助大多数的数据建模人员和企业启动他们的工作。然而，在我们做咨询业务时，很多客户都问如何拓展这些数据模型以及如何将这些数据模型应用于其他行业及如何创建可复用的模型示例等等诸如此类的问题。在我们研究这个问题的解决方案时，我们认为通用数据模型的实际延伸是提供通用模式，这些通用模式构成可复用的底层模块，这些模式有助于快速启动建模工作并为数据建模解决方案提供了备选方案，提高了数据模型工作的质量保证了数据模型的一致性 。

**本书的方法**

多数数据模型建模的书都将焦点集中在如何构建数据模型的技术上。这本书假设读者懂得数据建模的基本知识。数据模型建模已经存在很长时间了，因此多数信息系统专业人员对这个概念相当熟悉，因此也能理解这本书。通过阅读这本书，数据专业人士可以通过建立，定制及完善本书中现有的数据模型模式来开发他们组织机构的数据模型，如此以便能够开发高效高质量的新数据模型。从本质上来说，它为专业人士提供了基本工具和可复用的构建模块。由于我们所提供的是数据建模的初步基础，这样数据专业人员或者任何参与数据建模的人可以更加高效的完成建模工作。

正如我们所提到的，本书的每个章节包含了同种模式的不同的变形或层级，并从最为特定版本的模式逐渐过渡到最为泛化版本的模式。不同版本或层级的模式能满足不同组织机构的各种信息需求。这些模式横跨了各种各样的学科领域，是可复用的模版。例如，分类模式可用于支持**当事人**（PARTY）、**贸易**（TRADE）、**发票**（INVOICE）、**工作计划**（WORK EFFORT）、**装运**（SHIPMENT）等分类实体。然后，我们采取每个版本或层级的模式演示它如何应用于特定的场景中的。这些场景通常是基于我们现实生活的经历。在第一章节中，我们介绍了不同泛化程度的产品分类方式，例如，将虚拟的计算机硬件和软件零售商称为欧洲电子零售商。尽管本书中所有的人员、组织机构、样本数据和场景是虚构的，但是这些数据模型的示例确实源自我们在过去已开发实践过的数据模型。

在每一小节中，我们试图保持各个图表的基本格局一致以使得图标中所包含的实体位于同一位置。这样有助于展现不同模式由于经历每一泛化层级而产生的演变。但这中方法也并不总是可行，尤其是在第九章，在这一章节中我们将多种模式一起加入到模型中以满足不同的信息要求。

**不同级别的模式**

本书的每一章节介绍了不同泛化层级的模式，每一种模式都是从特定的模式逐渐发展成较为泛化的模式。在每一章节中，每种模式模拟同类型的数据，只是使用了不同的数据模型的结构和样式。例如，在第七章中，每一个联系机制的模式处理不同类型的联系信息数据（如：电话号码、邮政地址、电子邮件地址等等）。最初，以一种特殊的方法来处理联系机制，该方法是将联系机制信息作为某个特定角色的属性，比如**客户**（CUSTOMER）这个实体的角色具有国家代码、地区代码和电话号码等属性。我们称这种较特定的模式为一级联系机制模式。随后，由于使用越来越泛化的数据模型结构为同类型的联系信息建模的方法，模式变得越来越灵活。为此，我们随后展现了较为泛化的二级联系机制模式，之后便是更加泛化的三级联系机制模式，最后是最为泛化的四级联系机制模式。你可以根据企业建模的需求和建模任务中所涉及到的情境选择模式的级别和样式。

你如何回答是使用特定模式还是通用模式这样的问题呢？首先你要问下自己这个问题：“这个数据模型的用途是什么？”我们相信建立一个数据模型有两个主要目的：

1. 为了说明和沟通信息的需求。
2. 为数据库设计提供一个良好的基础。

这些需求可以使相互冲突的。如果目的是为了说明和沟通信息的需求，建模人员最有可能会开发一个更为具体的模型来展示业务代表的具体需求。例如，为了定义联系信息的信息需求是什么，建模人员可能会展示特定实体如客户（CUSTOMER）、供货商（SUPPLER）、雇员（ENPLOYEE）的一些属性如：国家代码、地区代码、电话号码和电子邮件地址。因此，可以认为这种模式是特定风格的模式，是一级联系机制模式。

**注：在此我们想强调下要谨慎使用一级模式，因为这些模式一般不是优秀数据库设计的有效基础。正如我们所说，建立数据模型一般有两个目的：数据模型能作为帮助理解数据要求的工具，也能作为数据库设计的启动基础。一级模式对于前一个目的起到很好的作用，但是对于后一个目的通常不是很有效。**

相反，如果目的是为数据库设计打下良好的基础，建模人员可能需要使用更加灵活地的模式，比如说三级联系机制模式或者四级联系机制模式，在这些模式中，任何当事人（人员或组织机构）可以具有任意数量的联系机制，这些不同类型、不同目的、不同用途、不同优先级的联系机制具有很多的分类方式。因此，这种模型非常的稳定，即使将来有增加联系机制的类型或类别的需求，也不需要改变它。这类型模型常常很难理解并且其不包含模型中很多具体的规则。例如，就泛化形式的联系机制而言，任何当事人都可能具有多数的联系机制，但是这有一条规则表明某个特定人员应当只有一个有效的呼机号。所以泛化数据模型模式并不遵循这条规则，特定数据模型模式则相反。

因此，我们建议你记录相关的业务规则，特别是那些泛化数据模型模式的业务规则。针对如何记录这些规则，业务规则引擎或元数据存储库提供了很多强大的解决方案。然而，我们发现很多企业并没有可用的这类解决方案，或者他们正在创建解决方案的过程中。因此你可以考虑用一个简单的方法来记录这些业务规则，将它们记录在文件当中并将文件作为数据模型的附件。此外，本书中的一些模式特别是在第八章节当中提到的那些模式提供了包含各种商务规则的数据结构。除了记录商务规则，我们认为所有数据模型的示例说明也都很重要，尤其是泛化数据模型的范例/实例。

图1-1说明了，当你从一级模式朝三级或四级模式移动时，你就是从特定风格的建模移向较为泛化风格的建模。同时也展示了当数据是“静态的”应使用一级模式，模式的级别越高，对模式灵活性要求就越高。因此，如果数据的性质是静态的不会改变的（例如，你只需要一个电话号码），那么特定的建模方式可能更适合。因为每个类型的模式都有特定和泛化之选，所以我们将在本书中讨论每个特定模式的优点和缺点。表1-1泛化了使用特定风格建模和泛化风格建模的好处。

**图 1-1**

**Table 1-1 特定风格建模和泛化风格建模的优点**

|  |  |
| --- | --- |
| 特定风格模型的优点 | 泛化风格模型的优点 |
| 更容易理解的模型 | 更加灵活。可以轻松的容纳更多的数据，也可以在不改变数据模型的情况下更改数据。能够满足当前和未来需求。 |
| 该方法使得和非技术受众沟通，验证和收集需求以及定义数据要求的范围变得更加容易。 | 更多的一致性。越高级别的模式常常导致数据模型具有相同类型结构，并促进了同一数据模型之间和不同数据模型之间的一致性和标准化。 |
| 是一种很好的方式，有助于在使用泛化数据模型之前理解数据需求。 | 以泛化数据模型作为物理数据库的基础，可以更多地使用通用例程来管理访问数据，因为数据结构都是比较相似的。 |
| 可以直接通过数据模型指定和执行更多的业务规则。 | 将各种数据类型融合到同一数据模型结构中，有时可以提供更大的能量和更好的性能。例如，当在同一个实体中维护所有的产品分类时，数据模型可以提供强大的分析能力。 |
| 更容易实现原型 | 在将这些模式作为物理数据库设计时，特别是使用他们开发强健、高质量的数据库模型时，这些模式为数据库设计提供了坚实稳定的基础。 |

**注：我们这里使用了单词*泛化（generalization）*[注解1]而非更为常见的术语*抽象（abstraction）。*因为很多的专业人士认为抽象意味着细节的丢失。例如，地图上的道路其实是一个抽象的概念，因为它限制细节到一定程度以达到将注意力集中在道路上的目的。泛化是指为了更灵活的支持数据需求，将很特定的数据模型结构转化为较为泛化的数据模型结构。通过使用不太特定的数据结构和通过添加、更改、删除数据实例以适应当前和未来的需求，泛化通过这两种方式提供了这种灵活性。使用术语泛化的另一个原因是面向对象社区以不同的方式使用了术语抽象的不同含义。还应当指出，动态的环境需要灵活的解决方案。然而，灵活的解决方案就其性质而言是泛化的，而泛化的解决方案是很难理解的。此外，*泛化*不能与*规范化*（normalization）[注解2]相混淆，他们是完全不同的概念。泛化是指使用更灵活的数据模型结构而规范化则是根据“键，组合键，唯一键”来对数据进行分组以消除数据冗余。**

如果你熟悉Zachman 框架，你就可能认识到数据模型的受众（audience）是不同的，这就导致我们需要不同类型和不同风格的数据模型。例如，一个专为所有者/业务代表而设计用来验证信息需求的模型可能看起来与专为设计师/建筑师而设计的模型不同，不同的模型将成为数据库设计的基础。为所有者或业务代表而开发的模型最有可能是特定模型比如说一级或二级模型，如此便能使用具体模式来说明和沟通数据需求。适用于设计师或工程师的模型将更有可能被设计成具有灵活性和适应性的模型，因此便可以减少了维护成本，三级或四级模式可能更适用于这种模型。

**注：John Zachman 的框架展示了各不相同的六行和六列，六列对应的是IT开发中不同类型的模型，而六行则对应的是不同的受众。在本书中我们将重点集中于第一列（数据源模型），我们使用不同的视图展示不同级别的类型。例如，Zachman框架的第二行对应的是专为“所有者”而设计的模型，这种模型通常使用像一级和二级这样较为特定的模式会更有效。考虑到Zachman 框架中的模型的受众是设计师和工程师，所以对应于Zachman 框架的第三行的模型使用三级或四级的模式将可能最合适，你打算如何使用这些模式取决于你如何理解Zachman框架中的行，你或许也能操作有些模式的参数，比如说适用于Zachman框架第一行（“规划者”视图）的一级模式和适用于第四行（“建造者”视图）的一些较为泛化的模式。我们要弄明白的关键点是不同的受众使用的是不同级别的设计模式。**

有些数据建模人员更喜欢对于不同的受众使用不同的数据模型。但是，维护、映射和交叉引用 “业务数据模型”和“架构数据模型”，的工作量非常大。模式在这方面有很大的帮助。当为业务代表开发一个用来收集和验证数据需求的模型时，我们一般使用一级或二级模式。然后当我们开发“设计师”或“架构设计”视图时，我们可以使用三级或四级模式来替换一级或二级模式。因此，这些模式具有“即插即用”的性质。这种“即插即用”的特质能够节省大量的时间且有助于这类模型的同步。例如，展示订单状态的一级状态模式能够迅速地被三级状态模式所替换，这样便在同一模型中展现了更加灵活的方法。可以认为模式是可以相互替换的组件。

**注：第9章是说明如何在不同类型的数据建模工作中使用这些即插即用的模式。**

除了针对两个不同受众使用两个不同数据模型的方案外，还有另一个解决方案即将具体的和泛化的这两种模式合并使用到同一模型。（将在第九章的讨论如何使用这些模式开发企业数据模型中介绍这一解决方案。）通常情况下，同种数据需求的相同数据模型可以使用具体和泛化这两种模式。然后创建用来展现模型的具体方面和泛化方面的视图。例如，在一个项目中，你需要为各种当事人的角色建模，为了验证需求，你可以先为不同角色的具体关系建模，即，赞助商，工人，项目经理，和项目负责人。然后你可以在模型中添加其他实体，展现出模型的架构视图，在视图中一个工作计划（这可能是项目、活动、任务或任何其他工作项）可以与任意个数当事人相关联，这些当事人可以具有多数角色。

“混合型”建模解决方案可用于本书中的任何模式，记住这点很重要。在第3章节中，我们提供了一个混合模式的范例，即混合上下文角色模式。混合模式是使用了特定和泛化这两种模式来为具体类型的数据需求建模。例如，混合上下文角色模式提供了单例模式，该模式包含了具体的二级上下文角色模式（该模式为特定的角色建模，比如说为某**项目**的**项目负责人**建立模型）和泛化的三级情境角色模式（此模式是为泛化角色建模，比如说为某**项目**的**项目角色关系**建模），我们可以以同样的思路开发一个“混合角色模式”，“混合分类模式”或本书中的任何其他模式。

**注：“混合”方法为同种类型的数据建模提供了可选择的方法：一种是使用特定方式建模，一种是使用泛化方式建模；这并不等于说这种方式赞成为相同数据构建冗余模型，我们不认为这样的模型是冗余的数据模型，因为我们并不主张你使用特定和泛化这两种数据模型结构捕获相同实例数据。我们将在第九章中更加详细的介绍这个方法。**

那么，为什么我们使用层级概念来描述这些模式，而不是使用与这些模式相关联的概念数据模型、逻辑数据模型、物理数据模型来描它们。首先，与概念数据模型、逻辑数据模型和物理数据模型的概念相比而言，这些模式与模型的泛化级别关联更多。在*数据模型理论和实践中*Graeme Simsion博士指出：通过大量的研究表明不同的建模人员在相同情境中为同一信息需求建立的模型也很有可能差异甚大。此外，他还指出这些基于同样信息需求而建立的模型存在三个主要的差异。差异之一便是泛化的层次，反映了数据建模的变形程度。因此，本书中的层级强调了泛化的程度，一级模式泛化程度较小，而四级模式泛化程度较高。（如图1-1）。

**注：在Simsion 博士的“*数据模型理论与实践*”一书中，他从*J. Verelst10*中引用了一个分类方案展示了造成数据模型的可变性的三个主要原因。这些原因分别是*“结构变异性”*（使用不同的数据结构比如说属性或实体代表现实世界中的相同概念）、*“垂直变异性”*（指使用不同级别的泛化）和*“水平变异性”*（指在泛化程度一样的情况下数据分类不同）。虽然我们专注于不同程度的泛化模式，但是我们所展示的模式也向你展现了“变异性结构”的备选方案。**

在很大程度上，对于不能将这些模式同概念模型、逻辑模型和数据模型相混淆，我们还有另外一个原因，因为数据管理行业对于概念数据模型、逻辑数据模型和物理数据模型的准确概念、所包含的内容以及其模型之间的不同之处存在很大的争议。Karen Lopez 一个公认的杰出的数据管理行业的领导者，主办了一个称为“数据模型争议点”的研讨会。她主办这个课程已有数十年，提倡投票参与，提出问题，如：“什么是概念数据模型？”并一直收到很多关于概念数据模型、业务数据模型和物理数据模型的定义的不同观点。正如她所指出，数据建模人员经常会为各种数据模型争议的议题争得面红耳赤例如，某个数据模型应当具有什么程度的泛化，属性是否应当作为概念数据模型或业务数据模型中的一部分，模型是否应当使用“当事人”这一概念等等。Simsion博士还提出了一个相同点即：从新手到专家，所有的数据建模人员缺乏共同的认识。他分享了大量关于这个问题和正在争论的议题的研究。这些研究甚至涉及他书中数据建模的本质和目的。在数据建模行业，目前对于概念数据模型、业务数据模型、逻辑数据模型和物理数据模型的目的和定义还没有出现一种通用唯一的解释。该定义既要足够广泛，要能大多数人接受又要特定以满足定义严谨性的要求。由于很难想出这样的通用定义，所以如果我们希望围绕这些概念讨论下模式，就使用一个经典的“第22条军规（Catch-22）”解决方案。我们认为如果我们表明了对概念数据模型、逻辑数据模型、和/或物理数据模型定义的立场或者争论，这些模型定义将会干扰我们想在本书中所表达的内容。我们认为泛化层级是对数据模型进行分类的另一个方法，这样的方法更助于达到我们的目标。

作为数据建模人员，通常会要被要求创建满足特定业务需求的数据模型。例如，通过使用像实体、关系和属性这样的对象来创建显示业务数据的模型。需要数据模型的企业希望我们创建能够支持特定功能的模型，那么数据建模人员试图将这些模型分成不同的类别，这些类别对于数据建模人员有很重要的意义（概念模型、业务模型、逻辑模型等等）。所以，我们决定将数据模型进行分类，但是不是使用这些分类，而是根据模型的泛化程度来分类。反过来，泛化的级别意味着应用于特定目的或功能的模型的适应性。正如我们所说，越特定的模型一般用于同业务代表沟通信息需求，而越泛化的模型通常作为一个基础，该基础能够使数据库设计的基础更加灵活。

本书的另一个好处是它将根据泛化的不同级别为每一个模式提供备选方案并指出了每种方案的优点和缺点，以帮助建模人员在拓展模型或创建新模型时做出明智的选择。例如，对于上下文角色建模（从一个实体到一个当事人的关系），我们提出了五种备选方案（一级模式、二级模式、三级模式、四级模式和混合模式）。数据模型资源手册卷1和卷2在各种通用数据模型中使用了这些不同的模式。但是，此时当你在应用每个备选方案时你能够理解这些基本原理，并且能够使用类似于前两册书中所提出的指导来了解何时以及如何拓展、定制、创建新数据模型。

**谁是这本书的潜在读者？**

这本书适用于以下人员：数据建模人员、数据架构师、数据分析师、数据库管理员、数据库设计人员、数据管理员、计算机科学教师、计算机科学专业的学生、企业数据集成人员以及涉及数据建模方面的任何人。本书的内容适合专业人士使用在以下领域：数据管理领域、数据质量管理领域、元数据管理领域、主数据管理领域、数据仓库领域、数据管控领域以及任何使用数据模型的其他领域。任何与以上所提到的角色或专业领域相关联的人都可以使用本书中的数据模型结构，这些数据模型的使用不但可以提高他们的工作效率，为他们鉴定潜在缺陷提供了一个关卡，而且通过理解备选方案而后应用那些模式还可以提高模型的质量。

这本书除了能作为专注于该领域的系统专业人士的无价工具箱外，还能作为一些组织机构的教科书，比如说企业或大学。

大多数人喜欢通过范例来学习，所以这本书展示了很多经过深思熟虑的数据模型模式和如何使用这些模式的范例，它既是经验丰富从业者的强力助手，又可以作为初学者的入门指南。

**本书包括哪些内容**

本书中的大多数章节讲的是我们认为最常见也是最有用的通用模式。第2章到第8章中讲了可复用模式和数据模型的备选方案。这些章节的内容包括模式的解释、每个模式的范例，样本数据以及每个建模选择方案的优缺点。第九章讲的是如何在不同类型的数据建模工作中应用这些模式。为了获得认同，在第十章中对于这些模式的使用和标准化提出了一些想法。具体内容如下：

**第二章节，“设置角色：当事人做什么”**这一部分详述了什么是声明角色，然后提供了数据模型模式，这些模式用来为人员和组织机构所做的事情建模，换句话说，就是在整个企业的大背景中，他们扮演了什么样的角色。例如，一个人或组织可能是客户、供应商、和/或雇员。本章介绍了每一种模式是如何支持数据关联至每个角色的以及数据（除了当事人角色）是如何关联至当事人的（人员或组织）。

**第三章，“使用角色：当事人如何被关联，”**这部分详述了什么是上下文角色，提供了数据模型模式和备选方案，这些模式可以用来在特定业务活动或其他实体的上下文中为人员和组织所做之事建立模型。例如,在订单的上下文情境中包括了客户的角色（如，他们可能是某个具体订单的 “付款”客户、“运达”客户或是“最终用户”客户）。

**第四章，“层级，聚合和点对点关系[注解3]：组织机构的类似数据，”**这部分详述了数据与相似类型的数据相关联的不同方式，例如，工作计划是如何同其他工作计划相关联的，部分是如何同其部分相关联的，或当事人是如何与其他当事人相关联的。这一章节为递归关系建模提供了模式和备选方案。

**第五章，“类型和类别：数据的分类”，**这部分详述了分类学[注解4]，类型和分类，然后提供了用于给任何类型的数据分类的数据模型模式和备选方案。例如，这些模式可用于为当事人、产品、订单、工作计划、固定资产或任何其他有分类的实体建立分类模型。

**第六章，“状态：数据的状态，”**详述了什么是状态并提供了模式和备选方案，这些模式和备选方案给出了为某个特定交易或实体状态建模的不同方法。例如，订单的状态可能是“已收到”“信用度待审核”“已录入”“取消”或“已处理”产品的状态可能是“构思阶段”[注释5],“已引进”，“中断销售”或“断货”。每个模式都支持三个基本方面的状态，也就是实体的许可状态，实体的当亲状态和历史状态，包括每个开始是有效后来不再有效的状态。

**第七章，“联系机制：如何获得联系，”**详述了什么是联系机制并提供了数据模型模式和其备选方案。当企业希望维护电话号码、电子邮件地址、邮政地址换或其他类型的联系信息的数据时，使用这些模式和备选方案能够满足企业的需求。这一章节提供了多种模式来维护联系机制和联系机制的类型、目的、用途、地点、优先级以及其他分类。

**第八章，“业务规则：事情应当如何进行，”**详述了数据模型模式和其备选方案，使用这些模式和备选方案创建以数据为中心的方法来维护支配企业运行的规则。模式维护规则的三个方面数据：关于规则本身的数据，规则所牵涉到的因素数据，规则的效果数据。例如，定价规则可能有规则数据（某个规则的名称和/或规则的声明），因素数据（与地理范围、缺货[注解6]等实体之间的关系）和效果数据（应用于特定因素的价格和折扣）。

**第九章，“使用模式”**阐释了怎样将不同的模式应用于不同的工作和情形。这些类型的工作包括应用这些模式收集需求，开发原型，开发特定的应用数据模型，开发企业数据模型，开发关系型数据仓库的数据模型，开发基于星型模式数据仓库的数据模型和开发主数据管理的数据模型。通过根据数据建模工作所涉及的类型和其对泛化层级的需求来替换不同级别的模式，我们展现了如何即插即用的使用这些模式。

**第十章，“社会化模式”**根据我们在各种企业实现这些模式的经验，在这部分描述了个人因素、文化因素和政治因素以及人类动力学因素，其涉及到使用这些通用模式以获得认可。本章为模式的社会化提供了“通用原则”，无论你是在创建企业级的标准和指导方针还是试图使用这些模式帮助某个特定工作启动数据模型以获得认可。具体而言，本章还介绍了如何用这些模式来获得认可的方法：理解人们为何会或不会使用这些模式的动机；为模式创建一个共同的，明确的引人注目的目标和愿景；建立信任以使人们知道他们能够依靠这些模式；当冲突出现时能够有效的管理它。

**注：为了提高读者的体验，本书中的所有图均可以在该网站（****[www.wiley.com/go/datamodelresourcebookvol3](http://www.wiley.com/go/datamodelresourcebookvol3).）查看。**

**其他的数据建模模式**

本书的用意是介绍那些与最常用数据模型结构相关联的模式。还有很多其他的数据建模模式存在，但是我们没有将这些模式写到这本书中，比如说以下模式（仅列了一些）：

**名称：**这种模式为维护实体的名称提供了备选方案。这种模式可维护实体的多个名称，历史名称，除了这些它还是提供了灵活维护任意数目的名称和名称部分的方法。这些模式能够应用于给人员实体、组织机构实体、产品实体或任何具有很多不同类型名称的实体命名， **标识符：**此模式为以多种方式维护标示符的实体提供了一个通用的结构。例如，如果打开你的皮夹，你可以看到有很多识别你身份的方法：驾驶执照、医保卡、社保号等等。产品也能有很多不同类型的标识符来识别其不同的来源。我们也可能使用不同的方式来识别投资工具，比如说股票代码、CUSIP号码和ISIN号码。标识符模式支持各种识别数据的方法，包括那些可能来自不同源头的方法。

**事务和事件：**不同类型的事务，例如，订单、装运、发票、会计事务(仅举几例)都具有一些很通用的属性或类似的方法，或者换句话说，你可以使用模式来为这些事务建模。例如，很多事务都是通过事件来初始化的（比如说客户订购产品。客户投诉等等）。事务通常都具有详细项（例如，订单项、装运项、发票项等等）。这些详细项彼此之间通常是相互关联的（例如，订单项与装运项关联，而装运项又以差不多的方式与发票项相关联。最后，事务和事件通常都具有相似的角色类型、状态、分类、递归关系和业务规则。

**授权：**这是一种够解决替换方式问题的常见模式，访问不同的数据类型需要什么权限、什么能够被分享而什么人能够访问什么类型的数据，这个模式能够用来为这些权限建模。例如，登录网站时、从 ATM提取现金时、使用电话号码识别码时或者是其他任何情形时，对于谁能够访问什么样数据，有必要为其提供创建、查询、更新或删除数据的授权和/或权限。

**本书中所采用的约定和标准**

本节对本书给出的模型的命名标准和图表约定进行描述。我们对Richard Barker*案例\*方法：实体关系模型*这本书中所提倡的数据建模表示法稍作修改，然后将这些修改版的表示法应用在本书中。 以下各节给出了本书在实体、超类/子类、属性、关系和范例数据（使用说明表来提供能够理解的数据样例）建模时所采用的约定和标准。然后我们给出了一些常见类型的数据建模表示法范例并简单解释了本书选择这些表示法的原因。

**实体**

实体是一个重要的概念，企业希望建立和存储的信息都是关于实体的信息。在本书的整个篇幅中，涉及实体的时候都用大写字母表示。例如，订单（ORDER）表示一个实体，其中存储了当事人之间购买货物的承诺信息。当在一个句子中使用实体的名称表示概念和业务规则时，可能没有用特殊字体表示，例如，单词“*order(订单)*”在后面这个句子中并没有大写：“许多企业都具有一些机制，例如用销售订单表记录销售订单信息。”实体的命名习惯是尽可能使用能够很好地反映出所要维护信息意义的单数名词。在本书的不同章节中，我们也为多数核心实体提供了很多定义。

实体用圆角方框表示，如图1-2展示了实体**订单**的例子。

**图1-2**

**子类和超类**

子类，有时也称为子实类，是一类具有与更加一般实体（也就是超类）相同的特征的实体，如具有共同的属性和关系。同时子类可能还具有超类所没有的属性和关系。例如，**法定组织**和**非正式组织**是实体**组织**的子类。

在数据模型图中，子类表示在其他实体之中。子类间的共同属性和关系在实体之外表示，外面的实体被称为超类。这样，子类将继承超类的属性和关系。图1-3展示的是一个超类**组织**和它的子类**法定组织**和**非正式组织**。要注意，应用于超类组织的属性**名称**，**税务标识**仅适用于子类**法定组织**，并且因为税务标识只适用于子类法定组织所以它只能在**法定组织**这个子类的层次上表示。因为法定组织和非正式组织继承了超类的属性，所以它们都具有**名称**这个属性。

**图1-3**

父类可能具有很多层。如图1-3所示，**企业**和**政府机构**是法定组织的子类，但同时也是**组织**的子类。组织的另一个子类非正式组织，非正式组织可能具有一个**团体**子类或**家庭**子类**。**因此，圆角框有可能会层层嵌套到任意数目的层级，以表示子类继承了父超类（外层框）的属性和关系。

每个子类彼此之间应该是互不相交的，并且它们应当在分类的不同层次上代表一个完整的分类集合。例如，组织可能要么是非正式组织（一个子类）要么是法定组织（另一个子类）。这个完整的分类集可能是在较高层次上的分类，有可能一些更加详细的子类没有被明确的包括到数据模型当中；相反，这些子类却有可能包含在一个**类型**实体中，正如图1-3中所示的**组织类型**。所以有些时候，实体的分类会出现在模型的两个地方：作为一个子类出现和作为一个类型实体的实例出现，类型实体表示了该实体所允许的类型的范围。以这种方式建立模型子类的原因是：某个具体子类可能有属性和关系，这样，它本身被当作一个实体来建模，也可能**类型**实体有其他的属性和关系。例如，**当事人角色**的子类可能是**客户**，而这客户有其自身的属性和关系。你也有可能具有一个**角色类型**，该角色类型有一个**客户**的实例（所有其他的角色类型也一样），而这个角色类型可能与其他实体相关联比如说**授权，**显示了什么角色被授予什么权限。因此，你即需要子类又需要泛化类型的数据模型结构。在本书中，当模型是具有子类的父超类结构时，我们一般展示一个**类型**实体。**类型**实体至少包含了对应于各子类的实例。我们将在第五章节进一步详述此内容。

**注：有些数据模型需要可互相包含的子类。虽然在卷1和卷2中我们为这种类型提供了种表示法，但是因为在本书中我们并没有使用相互包含的子类，所以并没有给它们提供一个约定。**

**属性**

属性保存着关于一个实体的特定信息条，比如一个订单的订购日期。属性在这本书中用粗体的小写字母来标识，就如此前的**订单日期**范例。在图表中，属性则用大写字母来表示。

**图1-4**

属性显示在实体中，它具有以下几个部分：

* 对于非主键属性使用任选的（o）或强制的（\*）指示符。在图1-4中订单角色有一个强制的起始日期属性和一个非强制的终止日期属性（也就是可选属性）。主键属性，如当事人中的当事人标识，订单中的订单标识，订单角色中的订单角色标识以及角色类型中的角色类型标识，其前面没有可选/强制指示符，因为主键总是强制的。
* 正文部分代表属性名称，例如，**订单角色**中的**起始日期**。
* 数据类型代表的是属性维护的数据性质。例如，数据类型为“ID”（标识符）的标识符，比如说订单实体中的订单标识或订单角色实体中的起始日期都有“DATE”（存储一个日期的或日期和时间的值）数据类型。表1-2列出了本书所使用的所有数据类型。
* 主键和外键分别使用（PK）和（FK）表示。你能从图1-4看到订单角色有三个外键：一个来自订单（**订单标识**），一个来自当事人（**当事人标识**），一个来自角色类型（**角色类型标识**）。所有的这些属性后面都有（FK）。你也能够看到每一个实体均有它们自己的主键：**订单标识**、**订单角色标识**、**角色类型标识**和**当事人标识**。按照惯例，本书中所使用的所有主键名称均是实体的名称后面跟上id;例如，订单的主键是订单标识（order id）。每个实体主键的值是无意义的唯一的序列号，这些序列号被称为代理键。我们之所以用无意义序列号作为我们的主键，是因为数据可能会发生变化（与有意义的主键截然相反比如说识别人员的社保号），我们可以开发一个模型，当数据发生变化（例如，社保号的变更）时，不会出现任何问题。

**注：对于数据模型是否应当使用代理键（没有任何内在含义只是用来关联实体）或自然键，不同的建模人员持不同的观点。我们承认关于这个问题两个派别的想法都有一个有效的论证，但是我们觉得涉及到模式时使用代理键（没有双关意图）更加自然，遵守使用无意义键的做法，那么即使某个特定实体实例的键发生变化，也不会造成数据出现问题或异常。（这些情况是可能出现的，因为键是将实体联系起来的机制，所以一个主键的值发生变化能够产生连锁反应，可能导致数据不一致等问题的出现）。**

* 除了主键标识实体的唯一性外，还可以使用唯一标志符号“UID”作为其替换标识符。例如，对于实体订单角色，当事人标识，订单标识，角色类型标识和起始日期用（UID）来指示所有键的组合可以使用组合键唯一地标识某个特定实例。这种方式有助于体现键的性质，例如，在以下情形中，单独使用当事人标识、订单标识和角色类型标识是不足以标识这个实体的唯一性，原因是相同当事人(John Smith)的同一订单(订单#123)在两个不同的时刻能有一样的角色（bill-to customer）,假设这个当事人首次被标识为付款客户，然后另一个不同当事人被标识付款客户，此后第一个当事人再一次被标识为付款客户。这样就需要用起始日期来标识实例的唯一性。表1-2展示的是本书所使用的各种数据类型和每种数据类型值的解释。属性名中包含的某些字符串所具有的含义见表1-3中所示的约定。

**关系**

*关系*定义了两个实体之间相互关联的方式。当在本文中用到关系的时候，通常使用小写字母表示，作为正文的普通部分。在某些情况下，需要特别地突出表示关系的地方，则用粗体的小写字母来表示。例如，由……制造（manufactured by）这种形式就有可能是本书正文中表示关系的一种形式。

**表1-2** 本书所采用的数据类型

|  |  |
| --- | --- |
| **数据类型** | **数据类型所包含的值** |
| ID | 用于指定主键的无意义标识符。这一般是个递增的（正数）序列号。例如，1、2、3、4、5等等。 |
| DATE | 指年、月、日，例如，2003年9月5日。  我们不为日期指定任何的格式。但是，每一章节中表中的范例，我们使用Mon,DD,YYYY这种格式来表示日期。例如，Feb.15,2006 |
| DATETIME | 日期和由时钟所提供的一天中的时间，例如，3/4/10 4:13 p.m。我们没有为日期指定特定的格式，如12小时制或24小时制。 |
| CHAR | 字符集，换句话说，就是文本字符串。 |
| DESC | 实体特性的说明信息。通常是比CHAR数据类型大的文本字符串。 |
| IND | 一个标识符或标记，表示某个属性只可能有两个值。例如，“是Y(es)”或“否N(o)”，“男M（ale）”或“女F(emale)”。 |
| NUMBER | 正数值或负数值，包括浮点值，例如，125, 1.0, 9.25,-45 等等。 |
| MONEY | 一个数额或一笔钱，例如，1,000,000美元、100英镑或10,000港币。一个属性是资金数据类  型，数据模型将维护该数据类型的值比如说“1,000,000”，“100”，或“10,000”，如果需要维护不同的国际货币金额，可能需要与**货币类型**实体相关联，该实体可以维护“美元”“英镑”或“港币”及其相应的货币符号。 |

**关系任选性**

关系可以是任选的，也可以是强制的。从一个实体关联至下一个实体的虚关系线条表示从这个实体到另一个实体的关系是可选的，而实线则表示关系是强制的（对于每个实体的所有实例都必须具有这个关系）。图1-5展示了一种关系，该关系是“每个**装运**必须只能与一个**邮政地址**关联。”因此，有可能存在还没有用于装运的邮政地址。

**表1-3**属性命名所采用的约定

|  |  |
| --- | --- |
| 属性名称中的字符串 | 意义 |
| ID(标识) | 系统生成的连续的唯一的数字标识（例如，1,2,3,4等等） |
| NAME(名称) | 用来指代某人或某物的术语。例如，角色类**型**名称，**产品**名称和**当事人分类**名称所表示的名称是用来指代某个角色类型（例如，“客户”），产品（例如，“A123号小工具”）或当事人分类（例如，“收入水平”）。 |
| DISCRIPTION(描述) | 用于表达关于实体信息的文本，可以帮助描述该实例的性质。例如，产品的产品说明会描述该产品的特性，该说明可能是“本产品是一款六英寸的不锈钢设备，它能帮助人们轻松挖出苹果核” |
| INDICATOR(标识) | 一个二元值（例如，是/否,男/女） |
| FROM DATE(起始日期) | 该属性指定了一个日期范围的起始日期，该日期范围内实例是合法有效的，该日期范围包括指定的起始日期。例如，**订单角色**的起始日期指的是当事人第一次开始（或将开始）扮演关于该订单的一个角色的起始日期值。 |
| THRU DATE(终止日期) | 该属性指定的是一个日期范围内的终止日期，包括指定的终止日期。（没用to date来表示该含义是因为thru date 更能清晰的表示出终止日期包括在内的含义）。 |
| …………………… | 这些小点表示属性的省略，这些被省略的属性可能会被包含在模式的实体中，但是这些属性与该模式的含义关联性不大。 |

**图1-5** 强制关系vs任选关系

**关系的基数**

关系可以是一对一、一对多、或者是多对多的。人们一般将此称为关系的基数。*鸦脚印*（一个三叉线，看起来像乌鸦的脚）表明了一个实体实例是否与另一个实体的多个实例相对应。图1-6表明，“每个**订单**必须由一个或多个**订单条目**构成”，因为三叉线位于**订单条目**那一端。关系的另一端表明，“每个订单条目都必须是一个也只是一个订单的一个部分”。一个一对一关系的表示中没有三叉线，而多对多关系则在两端都有一个三叉线。有时，我们把一对多关系称为父-子关系。

数据模型图没有展示多对多关系，因为多对多的关系可以分解成*关联*（*associative*）实体（也就是交叉实体）。打破多对多关系是一项常见的数据建模实践，因为可能需要维护关联实体的信息。

**图1-6** 一对多关系

有时在关系语句中需要加入术语*随时间变化*以表明该关系是否是一对多关系。例如，一个订单可能看上去只有一个订单状态；然而，如果需要状态历史，则每个订单必须随着时间变化可能具有一个或多个订单状态。

**外键关系**

外键定义为另外一个实体的主键出现在实体中（或表中）。例如，图1-6中，源于**订单**实体的**订单标识**是**订单条目**实体的一个外键属性。任何一个一对多的关系都表明在关系那一端的实体的主键将被指向到关系*多*（鸦脚印）的那一端的实体中。你可以看到作为外键的属性，在它的数据类型旁边会有一个（FK），这个是用来标识它是一个源自于相关实体的外键。有些数据建模人员并不将这样的外键作为实体的一个属性表示出来，因为这是多余的，通过关系便能推断出外键。事实上，为了页面更加的简洁且能够展示更多的数据模型，我们在本书的卷1和卷2系列中，就是这样做的。因为本书中所涉及的模式关联的实体较少，且所占空间也较小，还有我们收到一些反馈说将外键作为属性显示可能会对读者有用，所以在这一卷中，我们选择了将外键作为属性显示出来。例如，图1-6，在订单条目实体中，订单标识就被当作一个属性。

**用于处理多对多关系的关联实体**

*关联实体*也被称为*交叉实体*或*交叉引用实体*。这些实体通过实体间的交叉引用来解决多对多关系。它们常常具有附加属性，用以进一步地对关系进行描述。图1-7展示了当事人和联系机制之间的多对多关系，是通过当事人联系机制这一关联实体来解决的。每个当事人可能与多个联系机制相关联，比如说当事人的邮政地址、电信号码或电子邮件，因为人员和组织机构通常可以通过多种方式与他们联系。反过来，每个联系机制可能与多个当事人相关联。例如，许多人都可能有相同的工作地址或工作电话号码。这种多对多的关系可以通过交叉实体当事人联系机制来解决。

**图1-7**通过关联实体解决多对多关系

每个关联实体继承相关实体的键，并将其作为关联实体的外键。例如，当事人标识（继承于当事人）和联系机制标识（继承于联系机制）都是当事人联系机制的外键属性。这些外键属性也是关联实体的唯一键（UID）的一部分,可能还有其他属性，这些属性是唯一键的一部分。例如，**当事人联系机制**的外键属性：**联系机制标识**、**当事人标识**和**起始日期**一起组成了唯一键（UID）。之所以需要起始日期作为UID的一部分是因为一个当事人可能在两个不同的时刻点拥有相同的联系机制（相同的电子邮件地址）。

需要注意的是，在所有给出的例子中，每个关系都有两个相关的关系名称，从两个方面对关系进行描述。应当使用这些关系名称，如此它们便可以读成一个完整的句子，如以下格式所示：“每一个实体【必须是/可能是】关系名称【有且只有一个/一个或多个】实体，随时间变化，”在方括号中填入相应的选项即可：例如，“每个当事人随着时间的变化可以通过一个或多个的联系机制来联系”。

**互斥弧**

互斥用于表示一个实体与两个或多个其他实体相关联的关系，但是，在这些关系中，对于一个实体的特定实例，只能允许一种关系存在。互斥弧用一条贯穿于两个或多个关系线的弧线表示。在弧线上用一个“XOR”符号表示将关系连接起来。图1-8给出了一个互斥弧的范例。这个关系可以理解为：“每个库存要么必须只能位于一个设施内，要么必须只能位于一个容器内，但是这两者不能同时成立。”这表明库存项目存储在这两个层级类型的某一个类型中：它们要么位于诸如仓库这样的设施中，要么位于诸如设施内的箱柜这样的容器之中。

**说明表中的示例数据**

数据模型的许多部分是通过表阐释的，该表包含了属性的可能值。每个图例表通常被用来展示特定的实践示例。或者换而言之，我们通过展示实体和属性的实例来阐释该模式，从而当某个模式形成时，该实例给出了该模式看起来如何的范例。通常一张说明表是不够的，模式的阐释至少需要两张表。

表1-4所示的是订单项目实体的阐释。为了阐释某个实体的细节，该表展示了直接来自于相关实体的信息。例如，虽然图示表主要被用来阐释**订单项**实体内的实例，但是表1-4引入了**订单**实体的一些属性信息。

需要注意的是实体名后跟随“.”,然后它的后面再跟上属性名，所以ORDER.ORDERID是**订单**实体中的**订单标识**属性列。当在本书的正文中引用每个说明表中的数据时，数据会放在双引号中。例如，文本可能指的是某一个具体的订单“12930”，订单项标识“1”，该订单项的数量是“120”，单价是“200（美元）”。

**表1-4** 订单项目

有时，在说明表的某一列中使用括号，用来表示数据是紧密相关的并为该数据值添加上下文的值。最后一列是 ORDER TEM.UIT PRICE(CURRENCY TYPE),第一行单价的值是“200”，而这个值的货币类型是“美元”（因此，单价就是$200 US）。括号有时也被用来解释说明表中列的特性。例如，在阐释某个递归关系的实例时，**零配件[part]**（PART）实体的递归关系可能是有父**零配件**和子**零配件**（说明一个部分是由多数部分组成的）。因此，可能在说明表中有称为“PART.PERTID（PARENT）”的列，如此表明这列是高层次部分的列。而“PART.PARTID(CHILD)”则用来说明这一列表示的是较低层次的部分或者说是子成分。

**数据建模表示法**

正如我们之前所说，我们决定在本书中使用Barker的表示法。从图1-2到图1-8都是使用这种表示法的范例。Barker的表示法指的是由Richard Barker、Ian Palmer和Harry Ellis等人开发的实体关系图（ERD）表示法，它的名称源自于使其流行的 Richard Barker。这种表示法有时也指Oracle设计工具表示法，它被Oracle设计师的作为数据建模工具来使用。

根据前两卷的反馈和本书中所要说明的模型类型，为了帮助到读者，我们对这种表示法稍作修改。因此，我们遵循Barker的表示法，但是除了以下这些补充和变化的部分。

* 为了您的方便，数据模型在属性旁边用符号（PK）和（FK）明确的标明了主外键。
* 数据模型展示了属性的数据类型以供您参考，例如，char、number、等等
* “…………”符号用来表示可能还用其他的属性没有在模式中显示，因为这些属性被认为与模式的理解没有密切的关系。
* 我们不遵循以下这条规则：图表中的鸦脚印必须总是从右指向左，从下指向上。因为我们发现以一种使模型最具有意义的方式来画关系，人们阅读这些图表会更加容易。
* 有很多不同的数据建模表示法，所以我们花了相当多的时间和精力来考虑哪种表示法将最适合这本书和我们的读者，例如，除了我们所采用的数据建模表示法，这还有更流行的数据建模表示法，包括：
  + 信息工程表示法
  + IDEF1X表示法
  + 统一建模语言（UML）

此外，还有很多其他的数据建模表示法，包括陈表示法、对象定义语言（ODL）和对象-角色建模语言（ORL）,但是在我们经验中，写这本书时，这些都是不经常使用的表示法。

为了让你对数据建模表示法有一个更好的认识和证明相同的数据建模结构可能会因不同的数据建模工具而看起来不同这一点，在接下来这节展示了相同的数据建模结构使用了不同的表示法的范例，即图1-7所示的数据模型。

图1-9展示的是如何使用特定的数据建模工具（计算机协会的Erwin工具）和信息工程表示法来为图1-7中的数据模型建模。

图1-10展示的是同样的数据模型，也使用了信息工程表示法，然而，这次使用了不同的工具，即Embarcadero的ER/Studio数据建模工具。

**图1-9** 采用计算机协会的Erwin工具所绘的联系机制模式。

**图1-10**采用Embarcadero的ER/Studio工具所绘制的联系机制模式

IDEF1X(集成定义信息模型)，作为联邦信息处理的标准，于1993年推出并被广泛用于联邦部门。IDEF2X是一种用来设计关系数据库的具有语法的方法，在开发一个概念模式时，可设计其用来支持必要的语义结构。图1-11展示的是使用了IDEF1X表示法的同一数据模型。

图1-12展的是同一模式，但是使用了统一建模语言（UML）。

我们认为这些数据建模表示法都有优点和缺点，而且没有一种表示法是比另一种表示法更加高级。如果你想了解更多关于这些不同数据建模技术的优缺点，你可以参考David C.Hay的论文“数据建模技术的比较”，它逐个解释了这些数据建模表示法。我们选择使用Barker的表示法有以下几点原因：

* 该表示法允许子类框嵌套框。如此便能更加简洁优雅的将这些子类展示在页面中（在我们看来），它使用了较少的线条，却给人更多直观的理解（在我们看来）。
* 对于关系名称直接翻译成句子的约定能够使读者更好的理解关系的意义（在我们看来）。
* 这种表示法使用了鸦脚印符号，这是一个很常用的、大多数建模人员都理解的符号。
* 在数据模型资源手册卷1和卷2中使用了这种表示法，从而，我们保持了与前两卷的一致性。

**图 1-11** 展示使用IDEF1X表示法的联系机制模式

**图 1-12** 展示使用UML的联系机制模式

我们在本节中强调了有多种表示法和我们选择某一特定表示法的原因。然而，模式是可使用所有上述所提到的表示法（或任何其他表示法）。

在本书中所使用的模式与使用了不同数据建模表示法的顾客接触后，我们发现将本书中的数据建模表示法转化为任何其他的数据建模表示法是很简单容易的。因此，根据我们的经验，数据建模人员和数据专家能够很容易的遵循和使用本书中的模式，不管他们目前正在使用何种数据建模表示法。

**总结**

我们这篇介绍的目的是处理本书中的基础概念问题。这些问题包括这本书的需求，这些模式如何增强数据建模学科，模式和通用模式的定义，本书的方法，用于提供备选的泛化层次，本书的受众，本书内容总结，本书所使用的数据建模约定。

是的这本书独特而重要的是我们分享了基本模式，而这些模式是可以作为大多数数据模型的关键构建模块。我们认为本书所提供的那些模式和备选方案是为短时间内创建高质量数据模型提供了一个宝贵工具。

**参考文献**

1 P.P-S. Chen, ‘*The Entity Relationship Model-Towards a Unified View of Data*’, ACM Transactions on Database Systems, Vol. 1, No. 1, March1976, pp. 9-36. To see information on this article refer to Dr. Peter Chen’sweb page at Louisiana State University at http://www.csc.lsu.edu/∼chen/chen.html.

2 *A Pattern Language: Towns, Buildings, Construction* by Christopher Alexander, Sarah Ishikawa, and Murray Silverstein (Center For Environmental Structure Series, Oxford University Press, 1977).

3 *Design Patterns: Elements of Reusable Object Orientated Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Addison Wesley, 1994).

4 *The Data Model Resource Book*, Volumes 1 and 2, Revised Edition, by Len Silverston (Wiley, 2001).

5 *Data Model Patterns: Conventions of Thought* by David Hay (Dorset House Publishing, 1995).

6 A definition of *pattern* taken from Wordweb. Accessed at www.wordwebonline.com.

7 The idea regarding usage of the term *generalization* versus *abstraction* came from comments and suggestions made to us by Karen Lopez,karenlopez@infoadvisors.com.

8 “A Framework for Information Systems Architecture”by John A. Zachman, *IBM Systems Journal*, Vol. 26, No. 3 (1987). Also see http://www.zifa.com/ for diagrams and explanations of the Zachman Architecture.

9 *Data Modeling Theory and Practice* by Dr. Graeme Simsion (Technics Publications, 2007).

10 Verelst, J. (2004) *Variability in Conceptual Modeling*, Universityof Antwerp.

11 At the time of this writing this presentation was available at <http://www.infoadvisors.com/ArticlesVideos/> InfoAdvisorsPresentations.aspx.

12 For an alternative evaluation method see ‘What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models’ by Daniel L. Moody and Graeme G. Shanks in *Entity-Relationship Approach —ER’94: Business Modeling and Re-Engineering*, edited by P. Loucopoulos

(Berlin/ Heidelberg: Springer, 1994).

13 *Case \* Method: Entity Relationship Modeling* by Richard Barker (Addison- Wesley Professional, 1990).

14 From the IDEF Data Modeling Method Overview at http://www.idef.com/IDEF1X.html.

15 “A Comparison of Data Modeling Techniques” by David C. Hay, published in The Database Newsletter, Volume 23, Number 3, May/June, 1995 and updated August, 1999.

【注解1】：generalization 暂译“泛化”，与abstract“抽象”这个词的意思差不多但是又有所不同。至于不同点参考译文。

【注解2】：normalization 暂译“规范化”。

【注解3】：Hierarchies Relationship “分层关系”。

【注解4】：Aggregation Relationship “聚合关系”。

【注解5】：Peer-to-peer Relationship “点对点关系”。

【注解4】：taxonomies “分类学、分类”与classification“分类”这个词的意思类似。但是应该有些差别，至于有何差别原文没提到。Classification 和category这两个词翻译过来都是分类的意思，但是根据原文这两个词所表示的含义是不同的，为了区分它们之间的差别，所以在译文中将Classification翻译为“分类”，而将category这个词翻译为“类别”。

【注解5】conceived 这个此是“设想、构思”的意思，在这个它用来表示产品的一个状态，本人理解可能是指一个产品尚未成型还处于设想构思阶段的意思，所以在这里暂译为“构思阶段”。

【注解6】QUANTITY BREAK 直译过来是指（产品的）数量分段的意思，卷1中将其译为“缺货”。