# 层级关系、聚合关系和对等关系：具有类似数据的组织机构

客户经理想要知道他们客户的组织层级和总公司与子公司相关联的方式，子公司与下级组织相关联的方式，如部门和分支机构。项目经理想知道如何将计划拆分成项目，再将项目拆分成活动和任务的，换句话说，某类工作计划是如何被进一步拆分成较低层次的工作计划的。投资银行想知道如何用其他金融工具构建某些金融工具，比如说：期权。制造需要知道材料单以及零配件是如何能够构成其他零配件的。以上所有的示例均有一个共同点；他们都涉及某一情形，某一类信息在某种程度上与其情形本身相关。本章将介绍一种模式，这个模式支持在一个架构中维护相类似的数据。我们将这类自关联的关系称之为递归关系。

## 这类模型的意义是什么？

大多数的企业的生死存亡取决于他们创建和分析的报告。企业中的工作人员需要总结报告中的信息或是钻研报告中数据的细节。一类数据的层级、集合和对等的关系可以让人们挖掘出报告的详细信息。例如，某个经理可能开始想知道为什么一个项目将要花费六个月的时间来完成。经理们通常想在六个月内对项目的不同层级进行研究，以能够更好地理解不同阶段、活动以及任务的细节。

另一个常见的示例，如何将一个产品分解成多个组成成分（其他产品）。工程师、营销人员和制造商创建了材料单，展示了产品的所有组成成分。这对于材料资源的管理和生产非常重要。这些通常称为配件清单，且递归关系支持这一架构。

大型金融机构的风险部门需要能够评估不同原子级的有价证券，这些有价证券构成复合产品（共同基金、期权、衍生产品），这些产品记录在他们的资产负债表上。如果他们不能够改造这些产品的构建方式，他们的财务状况将不能对冲市场变化（市场风险）或客户破产（信用风险）。

## 本章节包含哪些内容？

本章首先介绍企业所使用的数据分类方法，这些类型的数据都与相似类型的数据相关。稍后本章介绍了企业用于组织这类数据的数据模型模式。

和本书的大多数章节一样，每种模式的建模风格起初都是使用非常特定的风格（一级递归模式），随着章节进展逐渐过渡到非常灵活的风格（三级递归模式）。不同层次的泛化程度可以应用于不同的企业或不同的建模风格。

本章节包括：

* 递归关系的定义和通过递归关系组织数据的不同方法
* 支持递归关系的不同模式
* 每种模式的关联性
* 深入了解每种模式
* 何时使用以及何时不使用这些模式
* 每种模式的简介，包含模式的优缺点

## 什么是递归关系和如何通过递归关系来组织数据？

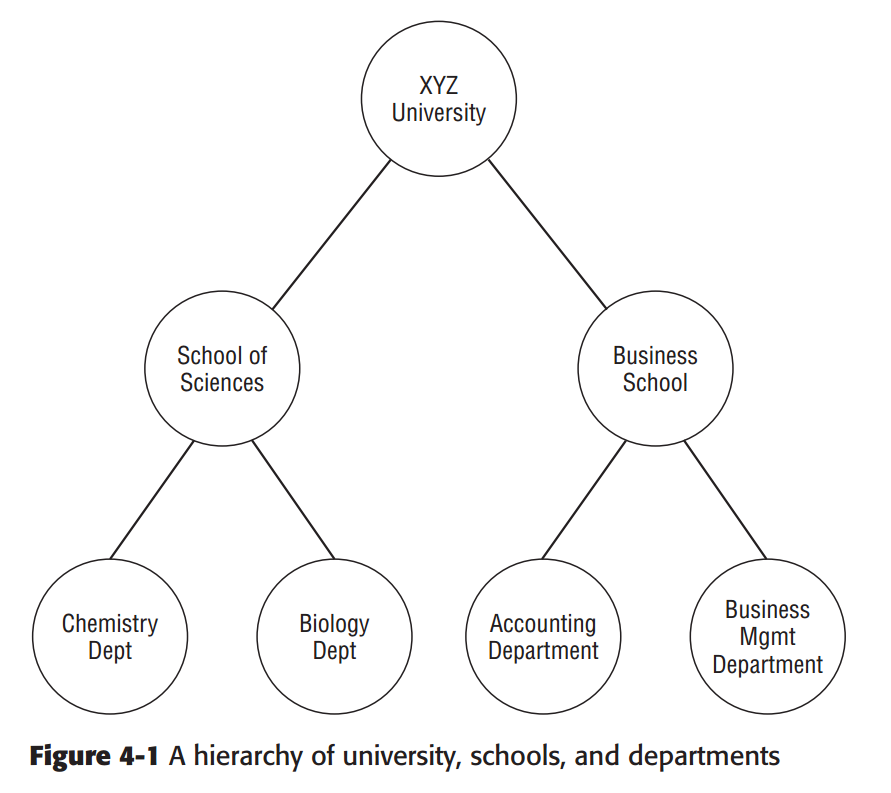
递归关系的一种定义是：“同一类别的多个对象之间的语义连接。”另一个定义是：“是复杂的递归式自关联关系；这些关系的起点和目标都是同一个实体。”每种定义都指出了重要的一点，递归关系是指实体的实例与该实体的其他实例之间的关系。这是这种自引用形成了递归关系。

**注：数据专业人士给这类关系取了很多的名称，并且交替着使用这些名称。自引用关系，复杂关系，“猪耳朵”关系，递归关联以及递归关系都是很少见的术语。递归关系似乎是其中最为盛行的术语。很可能是这个术语在语义上最精确。本章节的大多部分将使用这个术语。**

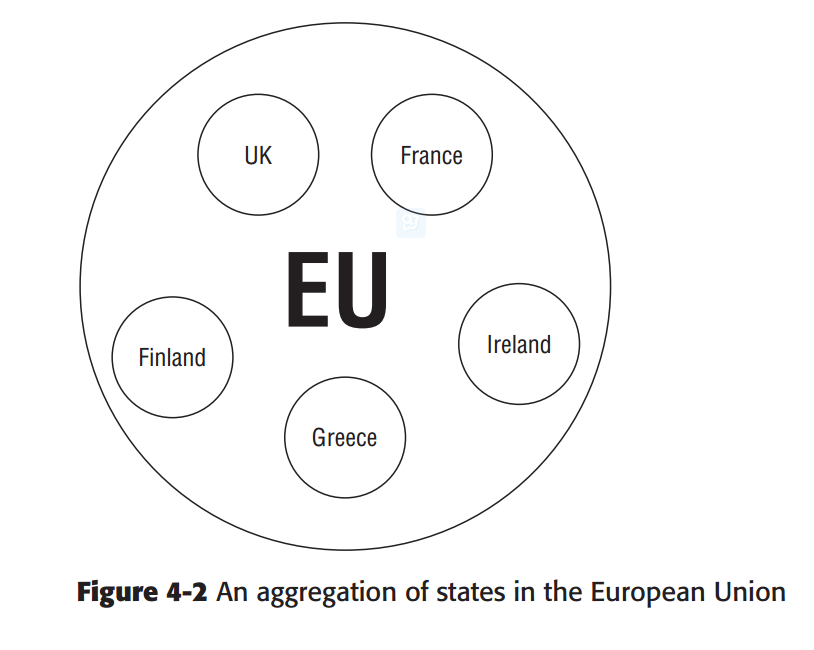
递归关系提供了收集数据并对数据进行分组的方法以及将同一类型的数据与该类型中的其他数据进行关联的各种方法。例如，材料清单是一个架构，该架构支持一个产品（或零配件）由其他产品（或零配件）构成。类似的是组织的层级架构，客户组织可能由其他的客户组织构成。三种非常重要的递归关系分别是层级关系、聚合关系和对等关系，下面我们将介绍这些关系：

* 可以将层级描述为“在一个组织的顶部有少数的或一个事物，而在每个事物之下又有几个事物，是个颠倒的树形结构。”4例如，计算机中的目录层次架构。

层级架构暗含着所有权，处于层级架构的顶端实体对象拥有其下面的所有子对象。这就意味着如果摧毁了父对象，层级架构也就被摧毁了。例如，一个大学有若干个学院（例如，商学院、理学院等等），而每个学院又拥有多个院系（例如，化学系等）。如果大学倒闭了，所有的院系将不复存在！图4-1是这个概念的图形化表示。层级架构的另一个有意思的一点是层级关系通常是同一方向上的一系列的一对多关系。换句话说，每个“子对象”只有一个“父对象”（如果是多对多的层级架构，可以有多个“父对象”），而每个 “父对象”可以有一个或多个 “子”实体对象。



**图4-1**大学、学院和科系之间的层次架构



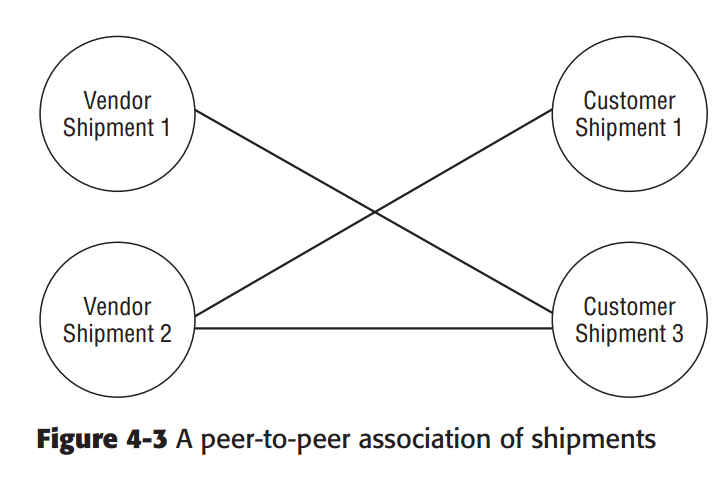
**图4-2** 欧盟的国家聚合体

* 可以将聚合架构描述成“一个总体，包含了其引用的组成部分。“一个帝国是许多国家的聚合体，且这些国家拥有一个共同的领导人 （Edmund Burke）”

这种架构与层级架构有何不同呢？聚合架构与层级架构的不同点在于，摧毁了聚合架构的所有者，并不能够摧毁该层级的其他组成部分。例如，很多欧洲的国家都是被称为EU（欧盟）的国家集合体的成员。多数EU的成员国（英国、爱尔兰、法国、芬兰等等）一般都是EEC（欧洲经济共同体）的成员。EU取代了EEC；当EEC不再存在的时候，这些成员国并未消失。他们只是变成了新的国家聚合体EU的成员。图4-2是欧盟的国家聚合体的图形表示法。

* 可以将对等关系定义为“通过某些共同的因素，人员、事物之间建立的一种连接；一种联合。5”这是当同一类别的数据项以不同的方式建立的彼此之间的关系而不是 父-子关系。以制造企业的装运为例，为了完成客户装运，需要卖方装运。

这种关系与层级架构关系和聚合架构关系又有何不同之处呢？对等关系是各要素之间的关系，他们属于同一个级别，所以不存在具有很多较低层级元素的更高级元素。他们与层级架构不同，是因为层级关系暗含了所有权，而对等关系不存在所有权。删除或摧毁对等关系中的任一元素是不会牵连或删除所有相关联的元素的。换句话说。这种关系不存在所属元素。值得一提的是对等关系支持不同方向上的多对多、一对多和一对一关系。图4-3以图解法介绍了这个概念。



**图4-3**装运的对等关联

**注：我们集中介绍了三种最常见的使用递归关系手机数据并将其分组的方式，但是递归关系还支持其他的架构，例如集合架构（作为某些过程的结果，将一组对象累积在一个地点，例如，邮票的收集或采购统计信息的收集）和组成架构（将零配件或元素组成一个整体，例如，一首乐曲是由其音符构成的）。我们还需要注意的是层级架构和聚合架构并不专属于递归关系，但他们会频繁地出现在递归关系中，这也是我们决定在此讨论他们的原因。而对等关系应当只出现在递归关系中。**

递归关系可以使用多种方式将数据组织起来，为什么区分这些方式很重呢？理解一类数据关联方式的实质有助于了解该数据的意义和重要性，并能帮助你管理这些数据。假如手机制造商决定停产某一特定型号的手机。手机的设计师一个聚合架构（材料单），该设计介绍了如何使用零配件来构建该型号的手机。因为不再生产这个型号的手机，所以此材料单也变成了废弃物。即使整个层级都被淘汰了，但是材料单中的各个零配件仍然可以存在，并且在其他产品的材料单当中可以使用这些零配件。

现在查看下图4-1中的大学、学院和科系的层级架构。可以说，“会计系”和“经济管理系”是“商学院”这一层级的组成部分。正如我们先前所述，如果大学倒闭了，各学院和各科系也将倒闭。换句话说，如果“XYZ大学”这一“主”实体被摧毁了，那么这个层级以及该层级上的所有成员都将会被摧毁。我们应当能够管理和理解不同组织的数据。否则我们又怎么会知道删除“大学”的后果是级联删除了“学院”和“科系”呢？

## 一级递归模式

有些企业需要描述非常特定的层次架构和聚合架构。例如，总部位于美国的公司，横跨了北美洲的多个国家、州和城市，它可能会要求国家，比如说美国，由一些州构成，并要将这些州进一步的分解成城市。这个聚合很重要的一点是它是一个静态架构的示例。换句话说，将国家（美国）分解成州（纽约州、新泽西州、加利福利亚洲等等），而将这些州进一步分解成市（纽约市、洛杉矶市等）

### 为何我们需要这种模式呢？

一级递归模式为层级架构（或聚合架构）的数据提供了最特定的建模方式。当企业需要支持不变更的而且对于层级间的关联方式具有严格业务规则的层级架构或聚合架构时，可以应用一级递归模式。一般情况下，这种模式不用于对等关系的建模。

使用一级递归模式的另一个原因是它可以作为层级架构或聚合架构的视图展现。这种模式清晰地展示了层级架构或聚合架构是如何被分解的。与较为通用的灵活架构相比，这种非常特定的架构可能更容易为商业人士或非技术人员所理解。

这种模式支持层级架构和聚合架构的基本方面：

* 不同层级的层级架构或聚合架构的定义或表示
* 层级之间的关系，也就是特定的业务规则，这些规则规定了每个实体在层级架构或聚合架构中的位置。

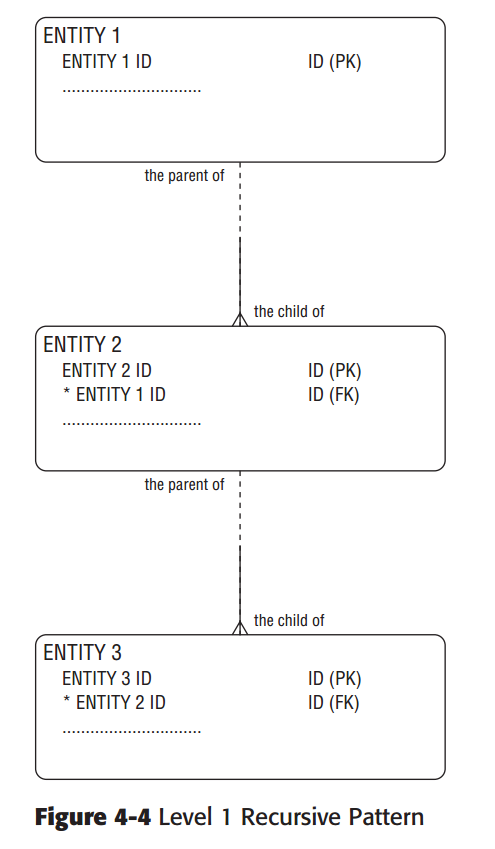
### 这种模式是如何进行工作的？

图4-4描述了一种模式，这种模式使用了非常特定的建模风格为一个三个层级的层级架构/聚合架构建模。每个实体分别代表了层级架构/聚合架构的不同层级。实体1是层级架构/聚合架构的顶端层级。实体2代表的是层级架构/聚合架构的中间层级，也就是实体1的“子”层级，实体3的“父”层级。实体3 是层级架构/聚合架构的最低层级，且其没有“子”层级。企业可以根据需求拓展这种模式的层级数量，只需要在层级架构/聚合架构的尾端（实体3）添加一个新实体（或多个新实体），或者是在层级之间或实体1之上插入新实体。

在这个模型中，你可以看到“实体1可以是一个或多个实体2的父实体”，而“实体2却必须只能是一个实体1的子实体”，“实体2 可以是一个或多个实体3 的父实体”，而“实体3却必须只能是一个实体2 的子实体”

想象这样一个场景：XYZ公司是一个大型的国际技术服务公司，该公司担心它没有以非常有条理的全面的方式来捕获和控制它所从事的项目的信息。为了更正这种状况，检查这些项目是如何构建的，该公司投资了一项研究。聘请了一个专业数据人士检验该技术服务公司所从事的所有项目。

为了启动与XYZ公司的重要项目经理讨论问题，数据专业人士创建了图4-5.图4-5所示的是具有三个层级的企业。可定义项目为“大型的或主要的事业，尤其是指那些牵涉到大笔资金、人力和设备的事业。”5“每个项目可以由一个或多个阶段构成。”

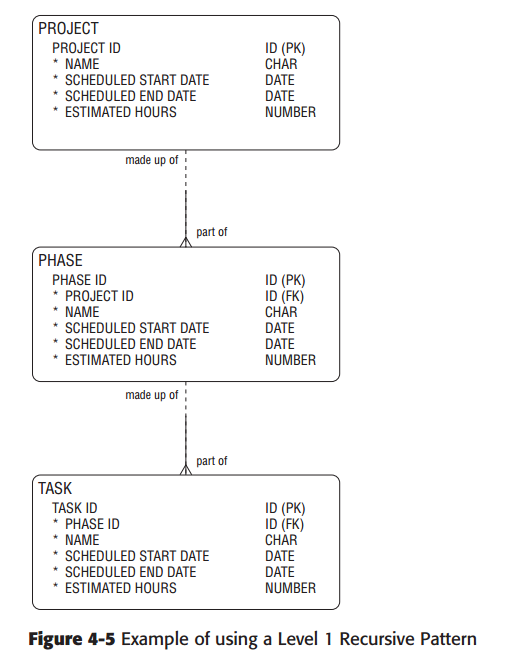


**图4-4**  一级递归模式

下一个层级是个阶段。“每个阶段必须只能是一个项目的组成部分。”阶段可以被定义为“变化或发展过程中的一个阶段。”“每个阶段有一个或多个任务构成。”

最后，在最低层的层级是任务。可以将任务定义为“指派给某个人的、某人正着手干的或期待某人完成的一项明确的工作。”5任务是XYZ公司所评估的最低层的工作。

**注：我们将项目、阶段和任务想成工作计划。数据模型资源手册1提供了这个概念的详细解释。6一般可以将工作计划描述为“计划中的，进行中的或已完成的工作活动。”它可能是指那些与工作需求履行相关的活动。在接下来的部分，这个术语将变得十分的重要。**



**图4-5** 一级递归模式应用范例

表4-1进一步介绍了如何应用这种模式。在表4-1中有一个项目，其项目名称是“企业数据仓库”。这个项目分为四个阶段，该项目的阶段名称分别是“规划”、“系统分析”、“ETL设计”、“测试”。每个阶段都有预定开始时间和预计小时数。例如，从“2009年6月3号”开始做规划，并且预计会耗费“700个”小时。每个阶段都会有任务。例如，在“规划”阶段包含两个任务，“创建规划模版”和“创建规划”。每个任务都有预定开始时间和预计小时数 — 例如，预定从“2009年7月20号”开始做“创建规划模版”的任务，预计这个任务会花费“30个”小时。

**（表）**

**表4-1** 一级递归模式范例

如果“企业数据仓库”项目被取消了，阶段的四个实例以及其所有的任务也将不复存在。这是一个典型的层级架构。正如先前所论，层级暗含着所有权，如果“主”对象被摧毁了，那么其“所属”对象或实例也将被毁灭。

看到图4-5中的架构，会发现一些有意思的问题。在该架构的三个层级上都枚举了预计小时数属性。例如，**项目**预计小时数的值是“2300个”小时，阶段总预计小时数的总数是“1410个”小时（“700”+“400”+“200”+“110”），最后，计算出任务预计小时数的总和是“1110个”小时。这是这个模式的优点和缺点。不同的预计时间彼此之间是不相互依赖的。你可以规定**项目**的预计小时数不同于**阶段**的预计小时数。例如，在项目的开始阶段，你就可以估计这个项目要花费多少时间；然后随着各阶段的进展，你可以预计各阶段要花费多少时间，以此类推。如果你想在子任务这一层级上维护时间，你从**阶段**和**项目**中将预计小时数属性删除。换句话说，以及递归模式允许你在层级/聚合架构的任何层级上维护特定的属性。这就意味着你可以维护层级架构/聚合架构的特定的业务规则。在我们的示例中，我们只在任务这一层级上包含了完成百分比属性，使用它来估计项目和阶段的完成百分比。正如我们将在本章的稍后部分所示，较泛化的模式是不会给你这样做的机会的。

在各阶层维护相同属性的这种特定架构可能会引起一些误解。如果你不谨慎，你可能将**阶段**预计小时数或**任务**预计小时数相加，以计算出项目的总预计小时数，但是你所计算出来的结果会与项目预计要花费的时间存在冲突。那么哪个总数才是正确的呢？如果这个架构被实现了，开发人员要从哪里查询项目的预计小时数总数呢—从顶层，从第二层还是从底层？“最顶层中”或“其他层级中”的小时数不存在冲突么？程序员应当查看项目的预计小时数，但是如果一个项目是由各阶段组成的，那么他们是否能够根据所有阶段的总预计小时数或所有任务的总预计小时数来算出项目的预计小时数呢？这些实体之间的关系将会让你相信这是可能的。

另一个有趣的问题出现在表4-1的最后一行。任务名称为“创建系统测试计划”的任务，其预定开始时间是“2008年11月3号”。但是在阶段层上，阶段名称为“测试”的阶段，其预定开始日期是“2008年12月1号”。怎么可能在创建阶段之前开始做该阶段的任务呢？

在某种程度上，这也是这种模式的一个优点，因为它能够指明各层级上的属性的不同意义和定义。因此，在这个架构的不同层级中的预定开始时间可以具有不同的定义。 这就表示完全有可能将任务的开始时间定义在阶段的开始时间之前。例如，可以将阶段的预定开始时间定义为“阶段的开始时间，这个日期可以在任务预定开始日期之前，当中或之后”。我们将会在本章节见到较泛化的模式，在该架构的所有层级中，不管属性属于哪个层级，都必须具有相同的定义，比如说预定开始日期。

使用这种模式的数据架构的其他业务示例：

* 在材料单的架构中，一个制成品可能由部件构成，而部件则由零配件构成
* 在地理范围架构中，可能国家由州构成，而州则由市构成，以此类推。
* 在组织机构中，总公司可以分有子公司，而子公司分有分支机构，分支机构低下又有各部门。

### 何时应当使用这种模式？

当我们决定使用这种模式时，我们需要考虑以下因素：

* **当有一个定义明确的静态的层级架构时。**如果正在研究的领域已为XYZ公司所充分理解，且预计该领域的层级关系在将来不会发生变化，那么可以应用这种模式。
* **当需要支配层级架构/聚合架构的模型体现业务规则时。**在数据模型中可能有些特定的业务规则，而展现这些规则是很重要的，例如，“每个阶段必须是且只能是一个项目的子实体”这个规则展示了“阶段只能在一个项目当中”。
* **当需要在层级架构的不同层级上定义特定的属性或关系时。**尽管在示例中并没有表现出这一点，但是这种模式允许每个层级具有不同的属性。例如，任务可以具有属性完成百分比，而这个属性是其他层级所没有的。如果企业决定项目的预计小时数必须总是与相关阶段预计小时数相等，那么你只能将这个属性从项目当中删除掉，因为应为它是可以推断出来的。还有，正如我们先前所说，虽然属性名称是相同的，但是在不同架构层次上，该属性具有不同的定义。
* **作为数据分析的工具。**在这个案例中，XYZ公司的项目经理可以运用这个模型来确定所需的层级架构的层次。第二，要意识到这种模型的问题，例如，评估和维护层级架构的不同层次中的预计小时数。最后，为层级模型或聚合模型提供了一个非常简单的架构，而这种架构有助于以一种更直接的方式启动数据建模工作
* **当数据专业人士需要业务需求更加容易理解时，并可能将其用于作为陈述说明的一部分，并提供给其他IT专业人士和/或商业人员。**本部分的模型图是个简单明了的声明范畴，展现了层级架构的三个不同层次，而这些都是企业感兴趣的层次。

### 这种模式的缺点是什么？

这种模式的弱点如下：

* **这是一个刚性的架构，不能轻易地更改它。**假如企业商定将有计划来负责项目的管理，在计划中有项目，然后是阶段，再是活动，最后是任务，会发生什么呢？这就可能需要大规模地更改数据模型和底层的数据架构。
* **这种模式不支持多对多关系。**8 这种模式展现了一对多关系，这种关系对层级关系或聚合关系而言是很常见的。当然，你可以更改这种模式，以使其支持多对多关系。
* **在相同层级结构的各个层次中维护具有相似意义和类似数据的属性可能会是人们感到很困惑，也可能会导致数据问题。**例如，任务中的预定开始时间可以在项目开始时间之前吗？计划的预计小时数可以少于其子类项目的预计小时数总数吗？值得一提的是这与捕获冗余数据信息是不同的。层级架构的不同层次中的属性可以不具有相同的定义。但是就我们的经验而言，这些属性通常具有相同的含义。也可以公平地说，如果你捕获了不同层级上的相似数据，你可以摆脱层级架构的上一层次中的“冗余”属性，而且可以根据较低层中的相同的属性，推断出该属性的值。例如，将任务中的预计小时数加起来，便能推断出该阶段的预计小时数了。
* **这种架构可以处理层级关系和聚合关系，但是完全不能处理对等关系。**

### 简介

在这一节中介绍了一级递归模式，数据专业人士可以使用该模式以特定的方式为层级关系或聚合关系建立模型。它可以帮助数据专业人士清晰地阐明数据需求和概念。这种模式也具有重大意义，因为你可以捕获同一层级架构中不同层次上的属性和关系，而这些属性和关系可以是特定于某一层级的，而至于层级架构或聚合架构还可以帮助你处理非常特定的业务规则。

层级架构或聚合架构将永远不会变化，这是一个冒险的假设。即使是看起来是十分静态的层级架构或聚合架构在将来也有可能会生变化。使用这种模式时，如果层级架构有变化，将导致数据模型失效，需要更改数据模型， 而这会造成重大的返工。数据模型可以维护模式不同层级上的相似数据。例如，在层级架构的计划、项目和任务层次上都有预计小时数这一属性。这样可能会导致误会和数据质量问题，因为在不同层级上，处理的数据可能不同，而且可能不能够一致地处理这些元素之间的关系。

对于企业而言，特定层级架构模式存在一个重大的问题，也就是该架构是一个刚性的架构。在有些情况下，可以将此点看成是一个优势，因为数据模型和随后的数据架构可以描述特定的数据需求一节执行特定的业务规则。然而，这也是个劣势，因为当信息需求发生变化时，则需要更改数据模型，这会导致该模型中的数据架构发生变化。

## 二级递归模式

在处理递归关系时，有些企业需要的是非常特定的递归关系，因此使用较为特定的模式。以及递归模式有助于满足企业这样的需求。但在有些情形中，企业需要一种更加灵活的解决方案。例如，制造公司可能有一组产品，这些产品由各配件构成，这些配件又由子配件构成，层级架构/聚合架构可以具有无数个层级。制造公司的工程师可能非常具有创造性，不断地发掘更好的方法，用来简化该公司产品的设计。

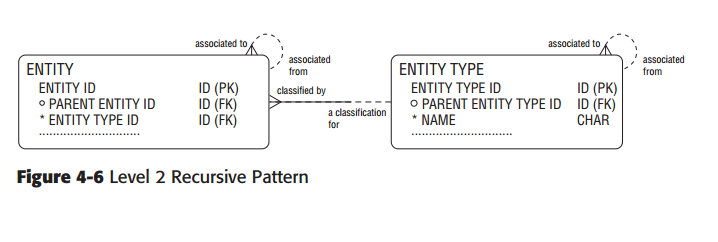
因此，随着时间的推移材料单的架构可能会逐步演变，材料单所需的层级将可能会继续发生变化。

### 为何我们需要这种模式？

这种模式为层级关系、聚合关系或对等关系的创建提供了一种灵活的解决方案。与一级模式相比，这种模式更加泛化些，并且它避免了一些在前一节中所提出来的问题，包括当发掘现的关联的时候，需要更该数据模型这样的问题。

### 这种模式是如何进行工作的？

图4-6 展示了为递归关系建模的模式，跟先前的模式相比，这种模式使用了更灵活的方法。在模型图的**实体**的旁边，你可以看到一对多的递归关系支持层级关系架构、聚合关系架构以及对等关系架构等。你还可以看到在模型图中添加了一个**实体类型**。递归关系通常也支持不同类型的实体之间的关系。实体类型有自己的递归关系。这样就需要将不同类型的实体关联到层级架构或聚合架构当中。例如，**项目**类型是**阶段**类型的父类型，而**阶段**类型又是**任务**类型的父类型。



**图4-6** 二级递归关系

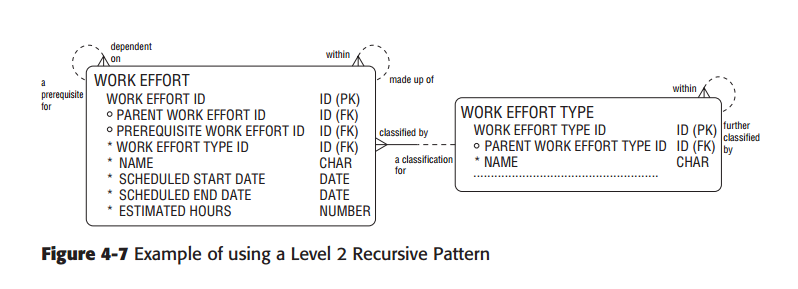
**注：具有递归关系的实体很可能没有与之相关联的类型实体。换句话说，实体的所有实例都是同一个类型。例如，可能企业有一个零配件实体，而该实体是由很多其他零配件构成的，在这里就不需要有一个零配件类型。但是通常情况下，我们发现实体中的递归关系构建了架构（层级架构、聚合架构），该架构的每个层级都有一个“标签”。这些标签就是类型。例如，“项目”、“阶段”和“任务”这些都是泛化的实体工作计划的标签，而“制成品”、“部件”和“原材料”则是泛化的实体零配件的“标签”**

如果你观察下图4-6，你会发现实体1有一个可选的外键—父实体标识，这个外键满足了需求：“每个实体的关联只能源自一个实体，而没个实体却可以关联至一个或多个实体”。这就意味着一个实体的实例可以有也可以没有父实体。

注：在诸多数据模型中我们发现了一个非常常见的一个错误，在这些模型中这些递归关系（支持层级架构）是强制性的。如果这个关系是强制性的其将不能够支持“树形结构的顶端”。试想下，拥有很多员工的大型公司。如果你的邮件从邮件收发室出发，通过经理得到雇员清单，最终邮件到达公司的CEO和/或老板手中。但是他或她没有上司，因此这个实例不能更有一个强制性的上级关系！

因为“每个实体的关联自一个实体”，在这个模式中，每个实体实例只能有一个父实体。这一点最好是使用示例来解释下，假如一个客户预订了一部X25翻盖手机（一类手机的名称），这部手机具有以下特征：比如说红色或蓝色的外壳，超大容量的存储卡或标准的存储卡等等。客户从诸多选择中挑选他期望的特征，换句话说，“具有红色外壳和超大容量存储卡的X25翻盖手机”. “X25翻盖手机”各订单项之间是相互关联的。因此在这个示例中，可以使用这种模式为订单项之间的关系建立模型，也就是订单项“X25翻盖手机”与另一个订单项“红色外壳”相关，且还与订单项“超大容量的存储卡”相关联因此，订单项“红色外壳”是只能在一个主订单项中，比如说“X25翻盖手机”订单项中，“超大容量存储卡”也同样如此。

**注：**这种模式可以包含多个递归关系。例如，一种递归关系可以支持依赖性（一个实例依赖于另一个实例），另一种递归关系支持并发性（两个实例需要在同一时间发生）等等。为了简单起见，我们只介绍了一种递归关系，但是因不能为此所限制。如图4-7所示。



**图4-7** 二级递归模式应用示例

如果我们继续前一节所介绍的XYZ公司的案例，你可以看到数据专业人士根据一级递归模式制作的图4-5，给图形展示了该公司各层级**工作计划**的初始作用域。在与股东进一步的商谈后，数据专业人士建立了**项目**、**阶段**和**任务**，这些事实上就是**工作计划**。换句话说，股东表明**项目**、**阶段**和**任务**具有相同的属性和关系，例如预定日期、预计小时数和当事人指派等等；一般情况下它，都可以认为他们是工作计划。不同的**工作计划**彼此关联的方式也不同。在此基础上，数据专业人士制作了图4-7。除了要体现工作计划可以由其他工作计划组成外，XYZ公司也想体现项目、阶段、或任务可以依赖于另一个项目、阶段或任务。

在图4-7中你可以看到“每个工作计划可以由一个或多个工作计划构成。” 例如，“项目”实例，是一类**工作计划**，存放在**工作计划类型**中，其可以由“阶段”实例构成，这个是**工作计划**的另一个类型，所以也是**工作计划类型**的实例。“阶段”的那些实例是由另一个**工作计划类型**实例“任务”实例构成。如果XYZ公司决定它需要添加额外的层级比如说“规划”或“活动”，这种模型可以在不更改的情况下适应这一需求。你所要做的事实将“活动”和“规划”添加到**工作计划类型**当中，然后将特定的项目或活动的实例添加到**工作计划**当中。

表4-2说明的是这种模型的实例。“企业数据仓库”项目的工作计划标识是“9002”，计划从“2009年1月1号”开始做此项目并预计会花费“2300个”小时。需要注意的是“企业数据仓库”**工作计划**实例，在**工作计划**中上级工作计划属性没有值或上级**工作计划**没有名称。这表明“企业数据仓库”是**工作计划**层级架构的父节点或最高节点。“企业数据仓库”有一个工作计划类型名称“项目”。XYZ公司之前表明了它要使用该架构来管理它的工作计划，而这些工作计划是由项目构成的，而项目由各阶段构成，各阶段是任务构成的。

在此还列举了另一个**工作计划**“规划”。“规划”的工作计划标识是“9004”，计划于“2009年6月3号”（也就是预定开始日期）开始该执行该计划，并且估计会耗费“700个”小时（也就是预计小时数）。“规划”是**工作计划**的一个“阶段”类型。它有一个上级工作计划标识“9002”，这个是“企业数据仓库”**工作计划**的外键。这就意味着“规划”是“企业数据仓库”**工作计划**的子计划。工作计划标识为“9006”的“系统分析”**工作计划**也是“企业数据仓库”**工作计划**的子计划。可以将这种架构看成目录树形结构，“企业数据仓库”看成是根目录，而“规划”和“系统分析”**工作计划**则是“企业数据仓库”工作计划的子目录。“规划”和“系统分析”又有其自己的**工作计划**，以大半个任务的形式作为他们的子计划。在案例“规划”中，你可以看到任务“创建规划模版”和“创建模版”，而在“系统分析”案例中，你可以看到“创建初始范畴声明”和“创建源系统库存”以及“发掘目标需求”这三个**任务**。

**表4-2** 二级递归模式的示例，工作计划。

**（表）**

启动计划后，XYZ意识到他们需要将任务进一步分解成子任务。表4-2的最后两行说明了这种模式的灵活性，能够满足这类需求，并且不需要改变数据模型。因此，可以添加一个新的工作计划类型名称“子任务”，在表的最后两行中展示的是任务“发掘目标需求”是由子任务“进行访谈”和“模型目标需求”。

有些工作计划在启动之前，可能会依赖于其他计划的完成。图4-7展示了“单个工作计划可以且只可以依赖于一个工作计划”。表4-3阐释了这类示例，“创建规划”这个任务实例依赖于“创建规划模版”的完成。此示例不是使用这种模式为层级关系或聚合关系建模，而是为对等关系建模。

**表4-3** 二级递归模式范例，依赖关系

**（表）**

图4-7有两种递归关系，一种关系表明了“每个工作计划可以由一个或多个工作计划组成”，而另一个则表明“每个工作计划会是一个或多个工作计划的前提，单个工作计划只能依赖于工作计划”（如果工作计划依赖于多个其他工作计划，那么则需要的是多对多关系，在下一个模式中将介绍这点）。这说明了一个实体可以具有多个递归关系。

以下要点表明在其他示例中，你可以将其他类型的特定关系泛化成通用的泛化实体，这是个具有递归关系的实体。

* 在材料单架构中，**制成品**是由子配件构成的，而**部件**则是由**零配件**构成的。该架构可以泛化到具有递归关系的**零配件**实体中，并且通过类型实体，将它分别归类为“制成品”、“部件”或“原材料”。
* 在地理分界架构中，一个**国家**可能由**州**构成，而**州**则由**市**构成，可以将这种架构泛化到一个通用实体中，具有递归关系的**地理分界**，以及通过类型实体将其分类为“国家”、“州”或“市”。
* 在组织架构中，**总公司**分为**子公司**，然后将**子公司**再分为各**个分支机构**，分支机构下又分为各**部门**，此架构可以泛化到存在递归关系的通用实体“**组织**”（**组织单位**）当中，并使用一个类型实体，将它分类为“总部”、“子公司”、“分支机构”或“部门”。

将可应用的泛化实体替换之前的实体，可以使用图4-6所示的模式为先前的数据模型建模。例如，为了维护层级关系或聚合关系中的各种层级，你可以具有以下实体：**分支机构**、**地理范围**和**组织**，以及**分支机构类型**、**地理范围类型**和**组织类型**。

### 何时应当使用这种模式？ 我们可以在以下时刻使用此模式：

* **需要一种灵活的模型时，能够支持所需的层级，并且不需要改变底层的实体。在先前的模式中，添加另一个层级到分层架构中，就需要添加新实体或关系。**如果需要插入新层级（比如说需要在**阶段**和**任务**之间插入活动，或者将**任务**分解成**子任务**），完成这些都需要做很多的工作。在当前这种模式中，更改层级数量或层级类型都会更加容易。例如，如果需要添加计划到工作计划层级架构中，就要添加一个新工作计划类型“计划”和一个关于计划的特定实例到工作计划当中。递归关系支持计划之间的关系以及所有其他类型的工作计划。
* **递归实体之间需要绝对的一对多关系，不存在多对多关系。**例如，有个层级架构，子关系可能有多个父关系，这种模式将不太适用这中情形（然而，在本章的下一个模式中，二级拓展递归模式将适用于这种情况）。
* **层级架构的不同层级全都具有相同或相似的属性和关系（尽管使用这种模式并没有这种要求）。**在范例中，XYZ公司表明项目、阶段和任务包含相同的属性和关系，因而这些意义根据其特征作为工作计划的不同类型。如果他们不包含相同的属性和关系，你可以添加项目、阶段和任务的子类到工作计划当中，并且维护每个子类特有的属性和关系。

**这种模式的缺点是什么？**

此模式的弱点如下：

* **与一级模式相比而言，这种模式更加的概括抽象。**所以很能与相关人士交流该模型的含义。例如，一级模式非常清晰地展示了层级架构的每个层次，而这种模式对于此类信息的阐释却很模糊。当然，信息越少拓展空间越大，因为这种模式是可以解释的，特别是使用示例图（展示实体特定的实例和属性）和/或实践范例。
* **这种模式仍然导致模型在层级架构的不同层次中重复了同类数据，因此会引起数据的不一致性。**例如，你仍然可以在不同层级上记录预计小时数。正如我们说述，这将导致父实例（例如，阶段）的预计小时数与其子实例（例如，阶段的所有任务所花费时间的总和）的预计小时数之和不一致。一种解决方案是使工作计划具有子类，比如说：项目、阶段和任务，然后将任务子类作为唯一的一个包含预计小时数属性的子类。
* **一对多递归关系可能有很多的局限性。**在层级关系中，当子实体需要关联到多个父实体时，这种模式不支持这个需求。例如，一个零配件被包含在多个父配件中，这个模型只允许在一个父配件中展示该零配件。上述情况也适用于任务，这种模式不允许一个任务是两个不同项目的组成部分。
* **这种模式对特定类型的工作计划彼此相关联的方式没执行业务规则。**例如，有可能只有项目是由阶段构成的，而阶段则是由任务构成的。这个模式允许无数的工作计划类型，无论这些类型他们是否有效，都可以关联到任何其他的工作计划类型。当使用这种模式时，对于制定那些在模型中可能并不是很明显的规则，数据专业人士不应当“偷懒”。
* **对于可选性可以具有不同的规则的架构，这种模型不能维护该架构的各层级。**例如，项目必须有一个名称；任务可以具有一个名称。这表明工作计划的任何工作计划类型都必须有一个名称。此外，除了最顶层的实体外，所有的子实体都必须要有一个父实体，而这个模型不能够执行这一规则。
* **这种模型不能够维护架构中各层实体所具有的特定关系。**例如，项目具有一个赞助商；任务和阶段则没有。或者只有任务是由某个特定的人员来实现的。
* **这种模式很难处理具有不同属性的各类实体。**例如，回叙任务具有一个实际工作时间属性，而阶段和项目却没有。然而，可以添加实体的子类来适应这一需求，例如，可以使项目、阶段和任务这些工作计划具有子类，这样可以在子类中维护特定的属性或关系。

### 简介

在这一节中，你可以看到二级递归模式是非常有用的一种模式，该模式为创建层级关系、对等关系或聚合关系提供了一种灵活的解决方案。这种模式具有一些重要的特征。第一，我们将**项目**、**阶段**和**任务**概括为**工作计划**的不同类型，如图4-7所示。我们之所能够这么做是因为**项目**、**阶段**和**任务**代表了项目的基本数据类型，并具有相同属性和关系。因为我们可以将**项目**、**阶段**和**任务**概括到**工作计划**中，因此我们可以将层级关系，如前一节的图4-5所示，概括为递归关系。这种模式真的可以帮助到那些需要更改层级或聚合架构的企业，比如说更改他们的申报架构或“材料单”。

也列举了这种模式存在的问题。例如，如果项目具有特定的属性或关系，这种模式将丢失这些属性或关系。为了解决这个问题，我们可以添加工作计划的子类，使其子类来维护特定的关系和属性。但是，概括的递归关系将不再支持层级关系（聚合关系）之间的特定的可选性：例如，一个项目必须有一个或多个阶段，但是一个阶段却是可以有一个或多个任务。

这种模式比一级模式更加抽象，但是它很容易向技术受众解释。通常情况下，这个数据模型需要更多的文档，比如说示例图，事件示例，和/或具有演示数据的电子表格和数据库，因为这种模式不像一级模式那样，无需加以说明。

这种模式另一个关键的局限性是它值提供了一对多关系。所以，如果有一个层级架构需要其子实体与多个父实体相关，这种模式不支持这一需求。可以被称为“多对多”或“矩阵”关系的架构，将会在下一节介绍。

## 二级递归拓展模式

大多数企业需要在处理层级、聚合或对等关系时需要灵活性。二级递归模式支持这种灵活性，但是以牺牲一定的可理解性为代价。这是灵活性与理解性的经典平衡。企业有时需要数据模型保留特定的商业知识，比如，特定的一对多关系，还要能够维护一定程度的灵活性，能够使用不同的方法创建层级、聚合或对等关系。

### 为何我们需要这种模式？

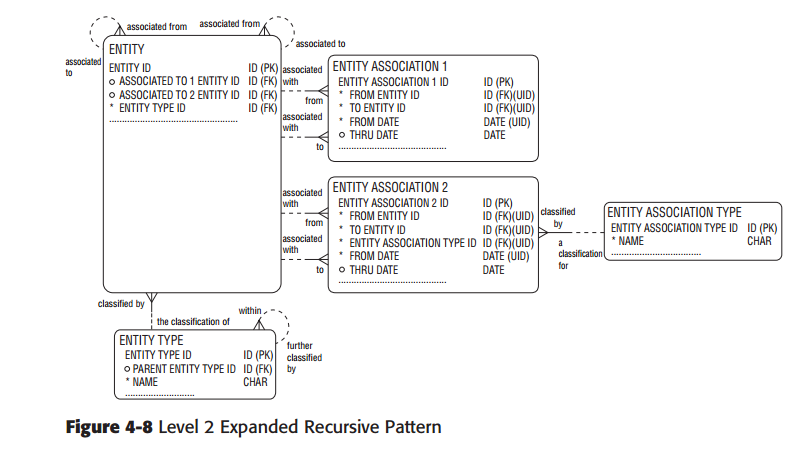
当需要多对多递归关系时，企业应当考虑使用二级拓展递归模式。9例如，电子设备制造商可能需要制造成品的不同方法，制成品中可以包括零配件，可以在不同的情形中包含同样的零配件并且可以将这些零配件与很多的父配件相关联。此外，可以有很多其他类型的多对多递归关系，例如当零配件的库存不足时，制造商可以使用什么零配件来代替所需零配件，而这些零配件可能互不兼容，也可能会相得益彰。

在这一节中我们将使用另一个示例，该示例是IT服务公司希望掌握的项目（工作计划）架构，比如说，任务（工作计划）是其他任务（工作计划）的早期版本，任务具有优先级（需要在另一个任务之前完成），以及什么是工作分解结构，要了解一个工作计划可能是多个上级工作计划的一部分。

**注：可以将优先级定义为“优先的事实、状态或权利；优先权”优先可以定义为“在一个场所中按照秩序、等级的重要性以及时间来先后执行某事”。依赖关系则可以定义为“以其他事物为先决条件或由其他事物而决定；因情况而异”。5**

**这种模式是如何进行工作的？**

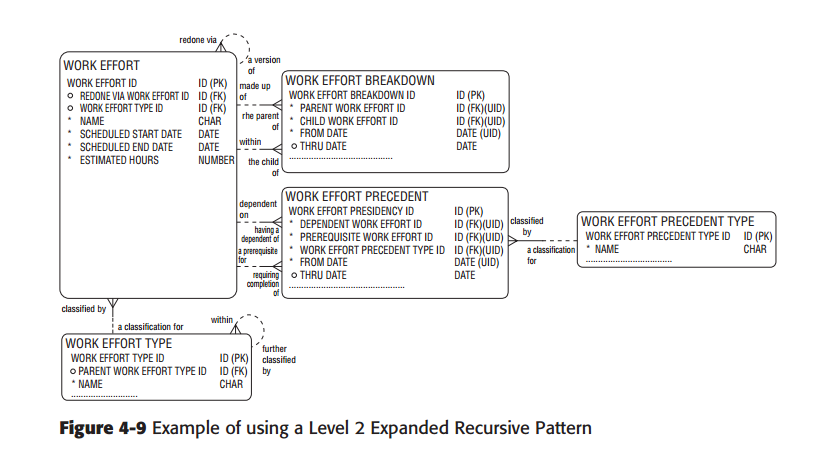
图4-8 说明了模式可以维护一对多递归关系和/或多对多递归关系。这种模式也支持上一种模式的所有递归关系；然而，此模式还进一步支持下面这种情形，即实例之间存在更复杂的多对多关系。



**图4-8** 二级拓展递归模式

你可以从图4-6中看到一对多递归关系。如果你需要多对多关系，那么你就需要有个关联实体或交叉实体，能让实体的实例以多对多的方式关联到实体的其他示例。在图4-8中，你可以看到这些关联实体，**关联实体**1和2。9特别是这些关联实体解决了实体与其自身相关联的不同方式。**关联实体**可以有也可以没有他们自己的数据属性。他们的存在可能仅仅是处理存在于递归实体中的多对多关系。

这种模式另一个有趣的添加是关联实体2有一个**关联实体类型**实体。有些递归关系可以被分类，例如，工作计划间的依赖关系要么是优先类型的依赖关系（某一工作计划需要在另一个工作计划之前完成）要么是并发类型的依赖关系。（两个工作计划需要同步完成）。子图4-9中我们将使用不同的范例来进一步地解释这一概念。



**图4-9** 二级拓展递归模式应用示例

我们将继续使用前一节所介绍的案例来解释这个模式。XYZ公司担忧有些待完成的任务实际上是两个不同项目的组成部分。例如，“创建源系统库存”可以是个工作计划，而该工作计划是数据仓库项目的系统分析模块中的一部分，但是它也是企业级配置管理项目的一部分。

因此，有些工作计划在本质上是以多对多的方式实现递归的，因而一个“父”工作计划有很多的“子”工作计划，而该“子”工作计划可以有很多的“父”工作计划。XYZ公司也想保留一对多递归关系，这种关系描述了它是如何管理工作计划的版本。最后，XYZ项目经理表明有很多不同的方法可以说明项目任务的优先级-例如，只有当优先的任务已全部完成了，才能开始启动某个任务，或一旦启动了优先任务，该任务也就可以启动了

基于XYZ公司项目经理的需求以及耳机拓展模式的应用，数据专业人数制作了图4-9.在这个图中数据专业人数维护了与图4-6相似的架构，但是此图允许满足XYZ公司的额外需求的多对多关系。

**工作计划**间的特定关系是通过右上方的**工作计划**实体的一对多关系来处理的。这个关系表示“借助于一个或多个工作计划，可以重做每个工作计划并且每个工作计划可以是且只可以是一个工作计划的一个版本”。这抓住了工作计划之间的关系，工作计划的新迭代或新版本已替换另一个工作计划。例如，试想下XYZ公司启动了任务“创建初始声明范围”，然后重新规划这个任务，并用此类工作的新版本替代了它，新版本叫做“开发工作和范围的声明”。

在表4-4中你可以看到通过使用“开发工作和范围的声明”，“创建初始声明方位”被重做了，而“开发工作和范围的声明”是较新版的工作计划。此外，通过使用工作计划的新版本“开发需求说明书”，重做了“发掘目标需求”。

**表4-4** 二级拓展递归模式示例，各种版本的工作计划

（表）

**工作计划分类**描述了在一个层级架构中各部分是如何构成工作计划的。通常工作计划本身也是一个项目计划。XYZ明确要求只要有一中类型**工作计划分类**。换句话说，它抓住了将项目分解成主要项目计划的方法，并且没有多种方法可以分解**工作计划**。这也是为什么没有**工作计划分类类型**的原因。XYZ公司还表明有些任务是多个阶段的工作计划中的一部分，而有些阶段则可以是多个项目中的一部分。

注：工作分解实际上可以有特定类型的分解架构—根据项目管理来分解工作或根据卖方来分解工作等等。这也适用于零配件，例如，工程师物料清单、市场营销物料清单和生产制造物料清单。如果是这样的话，一切就迎刃而解而解了，像我们所展示的工作计划有限类型一样，我们只需要以同样的方式添加一个工作计划分类类型就行了。

**表4-5** 二级拓展递归模式示例，工作计划分类

（表）

在表4-5中你可以看到“创建源系统库存”有一个工作计划标识9008，任务既是“企业数据仓库”项目下“系统分析”阶段中的一部分也是“配置管理”项目下“系统库存”阶段中的一部分。在不同的项目中可以为多个阶段执行相同的任务，这种情况经常出现。像这种情形，需要模式支持多对多递归关系。

工作计划优先级保持某一工作计划必须在另一个工作计划开始之前启动。XYZ公司的项目经理提出了一条规则，首要的工作计划必须被百分之百完成了才能够启动从属的工作计划。这是非常常见的情形。经过一番调查研究后，数据专业人士发掘了另一种类型的工作计划优先级，在该类型中在首要工作计划全部完成之前可以启动从属的工作计划。XYZ公司的项目经理将这两种类型分别称为“完全”优先任务和“部分”优先任务。在表4-6中你可以看到“创建规划模版”必须在“创建模版之前完成。”系统分析人员需要在任何规划开始之前百分之百地完成模版。在案例“创建初始声明范围”和“创建源系统库存”中，系统分析人员可以制作声明范围的大纲和介绍，然后收集所有的源系统文档。最后，你可以看到“创建规划”也需要等待“创建源系统库存任务”任务的启动。换句话说，“创建规划”具有两个不同的优先级任务— 一个是“创建规划模版”的任务必须全部地完成，而还有一种是“创建系统库存”任务只需要部分完成。

**表4-6** 二级拓展模式递归模式的示例，工作计划优先级

（表）

工作计划优先级的起始日期被定义为工作计划之间的这种关联的生效日期。为了能够看到何时可以启动其他的工作计划，有些建模人员可能认为我们需要添加一个属性来展示全部或部分完成“父”工作计划的结束日期。应当在**工作计划**实体中维护这个信息，可以添加一个完成率属性来表示，但是最后是使用状态模式来展示**工作计划状态**的完成状态，该实体维护完成率和状态日期时间（参看本书第6章中的状态模式）。

工作计划优先级是一种对等关系。我们的意思是：在工作计划优先级的上下文中，工作计划优先级的所有实例彼此是对等的，由于不存在父——子关系并且某一工作计划必须在其他工作计划部分或全部完成后才能启动。所以在示例“创建规划模版”中，在XYZ公司可以开始“创建规划之前”必须要全部完成“创建规划模版”。需要注意的是这两个工作计划都是任务。以下情况有可能吗？例如，“系统分析”（阶段）必须在“创建规划”（任务）启动之前完成？换而言之，也就是项目、阶段和任务彼此之间是否可以对等？答案是肯定的，因为可以将他们全都看作工作计划。在这个示例中，这是非常有意义的。它应当能够表现出必须先完成项目你才有可能启动任务、阶段或另一个项目等等。

因为工作计划优先级是对等关系，所以如果你删除了对等关系的一个成员，你不必级联删除关系链中的所有成员或将这些成员标为无效。但是这样也出现了为题。优先级的对等关系是否是可传递的？换句话说，就是“创建初始声明范围”必须优先于“创建源系统仓库”而“创建源系统仓库”必须优先于“创建规划”（如表4-6所示），这表示“创建初始声明范围”必须优选于“创建规划”？答案是肯定的，由于传递的逻辑，非常类似于关系理论中的传递依赖性10。你应当谨慎地维持“创建初始作用域声明”优先于“创建规划”。否侧，你可能会丢失首要工作计划的重要数据。这指明了当你具有对等关系时（分级关系或聚合关系），你需要规定将数据架构作为一个整体来管理的方法。例如，当你删除了一个示例或关系时，你需要考虑这个操作对其他示例或关系所造成的影响。

### 何时应当使用这种模式？

我们在以下时刻使用这种模式：

* **需要多对多递归关系时。**你需要评估每类工作计划是否要多对多关系，如果需要，那么二级递归模式将不起作用，需要的是这种模式。
* **当有同类多对多递归关系的变异性时，关联实体2和关联实体类型考虑到这些变异。**例如，工作计划优先级的变异体可以是“部分的”或“全部的”，而工作计划优先级类型可以维护这些变异体。
* **需要能够灵活地支持对等关系。**这种模式可以支持一对一、一对多或多对多对等关系。
* **层级架构的不同层级都具有相同的或非常类似的属性和关系。**在XYZ公司的案例中，XYZ公司表明项目、阶段和任务包含相同的属性和关系；所以，可以将它们按照特征作为工作计划的不同类型。

### 此模式的缺点是什么？

这种模式的缺点如下：

* **这种模式明确声明了实体的各种递归关系，但是如果你发掘了还有未捕获的递归关系会发生什么呢。**这意味着你需要为你发掘的每个递归关联添加一个新实体或新递归关系。如果是一个动态的环境，在该环境中可以发掘递归关系的新类型，对于这种环境这种模式可能不是最有效的建模方法。例如，如果也出现需求，记录那些需要在同一时间点完成的工作计划，那么则需要给模型添加另一种递归关系，也就是“并发”递归关系。
* **这种模式不能维护某些类型的业务规则。**例如，将同一项目中的相同的两个任务作为并发关系和优先级关系来维护会怎么样呢？这样做会引起困惑，应为这些关系在逻辑上没有意义。这将需要通过在数据模型之外记录和实施业务规则来解决，可以通过具有业务规则的三级递归模式或使用第8章所介绍的业务规则模式来解决。
* **对于属性的可选性层级架构的不同层级可以具有不同的规定，而这种模式不能够维护这些规则。**例如，可能有规则会规定**任务**必须有**预计小时数**，但是**项目**和**阶段**则可以有预计小时数属性，因为他们可以更具相关的任务推断出这个属性的值。这种模式表明任何**工作计划类型**的**工作计划**必须具有预计小时数。此外，对于所有的子对象除了最该层的对象都必须具有一个父对象这样的规则，这种模式也不能执行。
* **这种模式不能维护层级架每层级的特定关系。**例如，**项目**具有**赞助商**；**任务**和**阶段**则没有；或者只有任务是“由特定人员来履行的”。
* **这种模式使得处理具有不同属性的各种实体变得更加困难。**例如,或许**任务**有一个实际工作时间属性，而**阶段**和**项目**却没有。但是我们可以给实体添加子类以适应这一需求，例如，**项目**、**阶段**和**任务**可以有**工作计划**子类，可以使用这些子类来维护特定的属性（或关系）。

### 简介

在这一节我们介绍了一种模式，由于允许很多类的一对多关系或多对多关系，这种模式比上一种模式更加灵活。这种模式维护了一些商业知识（比如说指明一对多递归关系）；然而，这种模式真正的优势是它维护了一定程度的灵活性，允许使用多种不同的方法创建层级关系、聚合关系或对等关系。这种模式的重要意义在于它明确地解决了多对多递归关系。这个模式支持层级关系、聚合关系和对等关系。这此一节我们解释了为什么理解对等关系的底层规则很重要，例如，传递依赖性规则。最后，这种模式也很重要，因为我们向你介绍了如何混合多对多关系和一对多关系。

当你需要支持复杂的环境时，而在该环境需要获得实体与其自身相关联的多种方式，那么这种模式可以帮助你。除了不太容易适应可能会出现的新类型递归关系（我们将在下一个模式中解决这个问题），这个方法通常为在相当动态的环境中实现数据库提供了良好的基础。

如果你需要支持多种多对多递归关系这样的灵活性，需要了解你希望处理的各种递归关系，以及他们的属性（M:M或1：M）,并且这些关系不会发生变化，那么这种模式将会是合适的选择。

虽然这种模式取决于受众的数据建模的经验，但是在大多数情况下，用作作用域声明的一部分或数据需求的演示，这种模式有些过于抽象。在此模式中，明确介绍了每个不同的特定递归关系。这种模式有助于你楷书启动建模计划，并帮助你开始定义通用术语，用于企业当中的递归关系。

除非你在工作计划中创建特定的子类（例如，项目[PROJECT]、阶段[PHASE]和任务[TASK]），否则这种模式是不支持各种**工作计划**[WORK EFFORT]的特定属性和关系的。假如项目是唯一能够捕获预收益率[booked ratio]【注释1】与实收益率[burned ratio]【注释2】的工作计划将会怎么样呢？（对于专业的服务公司，“预收益[booked]”是指预计的收入总额而“实收益[burned]”是指实际赚取的收入金额），或是假如任务不同于其他类型的工作计划，它与**人员**[PERSON]之间有特定的关系有会怎么样呢？

## 三级递归模式

在有些情况下，递归关联模型要具有很好的灵活性。在动态的环境中，

实体存在或可能出现的各种类型的递归关系可能并不是很明显。有时，数据专业人士只是标出了核心的递归关系类型，但是我们需要知道企业将会拓展这些关系类型，为了管理那些关系，有必要添加其他的递归关系类型。

例如，在产品的递归关系案例中，数据专业人士可以确定需要有个**产品分类**[PRODUCT BREAKDOWN]（产品是有其他产品构成的）和一个**替代产品**[PRODUCT SUBSTITUTION]（产品可以被另一个产品取代）。他们可能会使用先前的二级拓展递归模式来适应这些需求。然后可能还会有其他的数据需求，而这些数据需求又牵涉到其他类型的递归关系比如说**过时产品**[PRODUCT OBSOLESCENCE]（一个产品取代了另一个产品），**互斥产品**[PRODUCT INCOMPATIBILITY]（一个产品不应当与另一个产品一起出售），或**互补产品**[PRODUCT COMPLEMENT]（是指那些被推荐作为所订产品的补充产品的产品）等等。随着时间的推移，当有其他的递归关系出现时，上一种模式可能不是最好的选择。

### 为什么我们需要这种模式？

这种模式满足了对灵活性的要求，可以添加或更新实体的递归关联[recursive association]。这种模型很适合于下面所描述的情形，数据专业人士是在一个动态的环境或是在一个新的业务领域中工作，尚不清楚需要多少不同类型的递归关联。数据专业人士该如何声明这些关联，从而能够灵活地添加新的关联，并不需要更改底层模型？

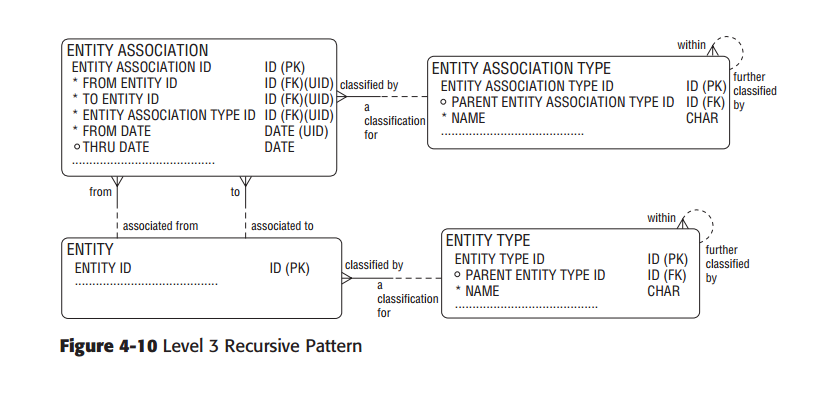
这种模式还提供了一个稳定的、共同的数据模型架构，有助于促进服务、功能、流程和规则的标准化。例如，同一模式可以应用于**当事人**[PARTY]、**产品**[PRODUCT]、**工作计划**[WORK EFFORT]、**零配件**[PART]、**订单项**[ORDER ITEM]、**库存项**[INVENTORY ITEM]或任何其他具有自关联关系的实体，所以这个模型可以保持模型的各部分的一致性。因此，通用的服务、功能、流程和规则全都适用于同类数据架构。这种模式还非常的灵活，当你发掘新类型的关联时，不用更改底层数据模型。

最后，这种模式提供了健壮的实现。我们已成功在多种不同的环境中实现了这一模式。

### 这种模式是如何进行工作的？

这种模式包含了我们在二级拓展模式[the Level 2 Expanded Pattern]中所介绍的关联，且将这些关联概括到共同的**关联实体**[ENTITY ASSOCIATION]中，并添加**关联实体类型**[ENTITY ASSOCIATION TYPE]，用这个实体来定义这种模式所能处理的不同类型的关联。添加**实体关联类型**的实例，使得添加其他的递归关系变得更加容易。换而言之，**关联实体**[ENTITY ASSOCIATION]允许实体现有的或将来可能会有的各类多对多关系，并通过**关联实体类型[**ENTITY ASSOCIATION TYPE]来将这些关系分类。

在图4-10中，“每个实体的关联可以来自[associated from]一个或多个关联实体[ENTITY ASSOCIATION(s)]”且“每个实体可以关联到[associated to]一个或多个关联实体[ENTITY ASSOCIATION(s)]”



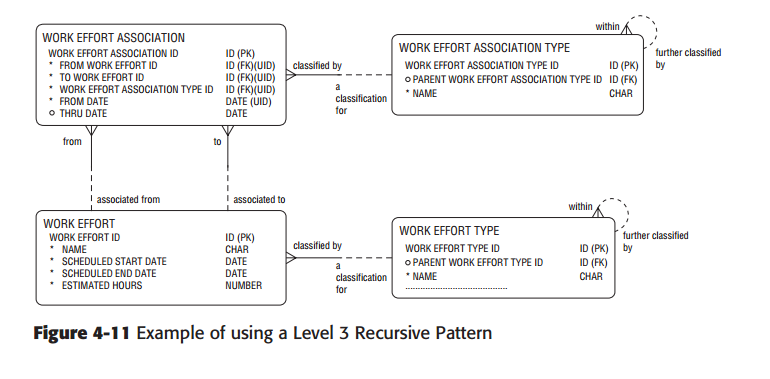
**图4-10** 三级递归模式

通过**关联实体类型**[ENTITY ASSOCIATION TYPE]，可以对**关联实体**[ENTITY ASSOCIATION]进行分类。你可以通过添加**关联实体类型**[ENTITY ASSOCIATION TYPE]的实例来创建新的关联类型。“每个**关联实体类型**[ENTITY ASSOCIATION TYPE]可以是一个或多个**关联实体**[ENTITY ASSOCIATION(s)]的分类且每个**关联实体**[ENTITY ASSOCIATION]必须只能使用一个**关联实体类型**[ENTITY ASSOCIATION TYPE]来进行分类”。

**注：此处的关联实体类型[ENTITTY ASSOCIATION TYPE]是前一种模式的关联实体类型[ENTITY ASSOCIATION TYPE]的父类。在前一种模式中，它可有像“部分的”和“全部的”这样的值，这些值只是优先递归关系实例的子集。此处，关联实体类型[ENTITY ASSOCIATION TYPE]可以维护那些值，还可以包含其他类型的递归关系，比如说“依赖关系[Dependent]”、“并发关系[Concurrent]”和“分类关系[Breakdown]”。所以，在前一种模式中，关联实体类型[ENTITY ASSOCIATION TYPE]只维护特定类型的递归关系，而此处它是更加广泛的实体。**

需要注意的是**关联实体类型**[ENTITY ASSOCIATION TYPE]具有自关联的递归关系，表明“每个**关联实体类型**[ENTITY ASSOCIATION]只能在一个**关联实体类型**[ENTITY ASSOCIATION TYPE]内部”。这种模式支持可能会出现的层级类型，例如，“优先级关系[Precedent]”和“并发关系[Concurrency]”是**关联实体类型**[ENTITY ASSOCIATION TYPE]“依赖关系[Dependency]中的两种递归关系”

图4-11展示的是这种模式的示例。在图4-9中，你可以看到**工作计划分类**[WORK EFFORT BREAKDOWN]和**工作计划优先级**[WORK EFFORT PRECEDENT]以及支持不同**工作计划**[WORK EFFORT]版本的一对多关系。在这个模式中，你可以通过添加**关联实体类型**[ENTITY ASSOCIATION TYPE]的实例（“分类[Breakdown]”、“优先级[Precedent]”和“版本[Version]”）和**工作计划分类**[WORK EFFORT BREAKDOWN]、**工作计划优先级**[WORK EFFORT PRECEDENT]所捕获的关系以及不同版本工作计划的一对多递归关系来维护这类数据,这些关系全由**工作计划关联**[WORK EFFORT ASSOCIATION]来维护。任何新的关联类型都可以使用这种方式来处理—例如，如果XYZ公司想要体现“互斥性[Incompatibility]”，就是指在同一个项目中两个工作计划不可以同时存在，或是“替代性[Superseding]”是指一个工作计划取代了其他的工作计划。最后，为了捕获分类的层级结构，比如说在关联类型“优先级[Precedent]”内的“全部[Total]”和“部分[Partial]”关联类型,你可以使用**工作计划关联类型**实体[WORK EFFORT ASSOCIATION TYPE]的递归关系。



**图4-11** 三级递归模式应用范例

在表4-7中，你可以看到很多示例，该示例展示了处理不同类型的递归关系的方法；这些示例全是工作计划关联的实例。换句话说，这种模式为处理所有的递归关系提供了一个通用的方法。

**表4-7** 工作计划关联和工作计划关联类型的示例

（表）

在表4-7的前两行，你可以看到**工作计划**关系的两个“版本”，在该表中“创建初始作用域声明 [Create Initial Scope Statement]”现已变成了一个工作计划版本，被称为“开发工作和作用域的声明[Develop Statement of Work and Scope]”，而“发掘目标需求[Discover Target Requirements]”则变成了一个被称作“开发需求说明书[Develop Requirements Specifications]”的工作计划版本。在表4-7中你还可以看到**工作计划关联**也可以有一个“优先级[Precedent]”关联类型，表明“创建规划模版[Create Mapping Template]”需要在“创建模版[Create Mapping]”之前发生。

在表4-5展示了“企业数据仓库[Enterprise Data Warehouse]”项目的工作分解架构。**工作计划关联**[WORK EFFORT ASSOCIATION]实体容纳此工作计划架构也很容易。你可以从表4-7中看出规划[Mapping]是“企业数据仓库[Enterprise Data Warehouse]”的下一级工作计划，你还回发现“规划[Mapping]”中具有两个工作计划，分别是“创建规划模版[Create Mapping Template"]”和“创建规划[Create Mapping]”。

这种模型的“泛化”特性也会导致出现一些问题。在前一节的图4-9中，你可以看到**工作计划分类**[WORK EFFORT BREAKDOWN]和**工作计划优先级**[WORK EFFORT PRECEDENT]的关系名称是十分生动的。例如，**工作计划**[WORK EFFORT]和**工作计划优先级**[WORK EFFORT PRECEDENT]之间的关系表示“每个**工作计划**[WORK EFFORT]可以依赖于一个或多个**工作计划优先级**[WORK EFFORT PRECEDENT(s)]，而每个**工作计划优先级**[WORK EFFORT PRECEDENT]依赖于**工作计划**[WORK EFFORT]；每个工作计划可以是一个或多个工作计划优先级的前提条件，而每个工作计划优先级则必须等待唯一的**工作计划**[WORK EFFROT]完结”。在这种泛化的模式中，需要这些丰富的关系名称更加的泛化，以能够包含很多不同类型的递归关系。例如，“每个**工作计划**[WORK EFFORT]的关联来自一个或多个**工作计划关联**[WORK EFFORT ASSOCIATION(s)]”。出于这个原因，本书所展示的案例都是使用了最概括的模式，使得这种模式有点难以理解。

这种模式除了可以应用到工作计划中外，还有很多其他的数据建模情境，三级递归模式也能行之有效，例如：

* 在材料单架构中，你可以将**实体**[ENTITY]、**关联实体**[ENTTITY ASSOCIATION]和**关联实体类型**[ENTITY ASSOCIATION TYPE]替换成**零配件**[PART]、**零配件关联**[PART ASSOCIATION]和**零配件关联类型**[PART ASSOCIAZTION TYPE]。可能会有以下**零配件**子类：**制成品**[FINISHED GOOD]、**子配件**[SUBASSEMBLY]和**原材料**[RAW MATERIAL]，**零配件关联类型**[PART ASSOCIAZTION TYPE]的实例:“零配件分类[part breakdown]”、“替代零配件[part substitute]”、“过时零配件[part obsolescence]”和“互补零配件[part complement]”和“不兼容零配件[part incompatibility]”以及其类型的零配件关联。
* 在地理架构中你可以将实体、关联实体和关联实体类型替换为**地理范围**[GEOGRAPHIC BOUNDARY]、**地理范围关联**[GEOGRAPHIC BOUNDARY ASSOCIATION]和**地理范围关联类型**[GEOGRAPHIC BOUNDARY ASSOCIATION TYPE]。**地理范围**[GEOGRAPHIC BOUNDARY]可以有子类：**国家**[COUNTRY]、**州**[STATE]、**地区**[TERRITORY]、**省**[PROVINCE]、城市[CITY]、乡村[COUNTY]以及更多的代表世界各个地方的子类（要想了解更多关于这方面的信息，可参看第7章中的联系机制）。**地理范围联系类型**[GEOGRAPHIC BOUNDARY ASSOCIATION TYPE]可以具有以下实例：“内部区域[within]（例如位于中州内部的城市）”、“重叠区域[overlapping]”和“边界区域[bordering]”以及地理范围关联的其他可能类型。
* 在组织架构中你可以将**实体**[ENTITY]、**关联实体**[ENTITY ASSOCIATION]和**关联实体类型**[ORGANIZATION ASSOCIATION TYPE]替换成**组织**[ORGANIZATION]、**组织关联**[ORGANIZATION ASSOCIATION]和**组织关联类型**[ORGANIZATION ASSOCIATION TYPE]。**组织关联类型**[ORGANIZATION ASSOCIATION TYPE]可以具有实例：“上级关系[parent relationship]”（用于捕获组织机构的关联，此关联记录了哪个组织从属于哪个组织）、“合并关系[merged]”（指那些合并的组织机构）、“合作伙伴[partner]”以及组织关联的诸多其他可能类型。这个架构除了可以拓展为**组织机构**[ORGANIZATION(s)]，还可以拓展成**当事人**[PARTY(s)]，可以包含人员和组织的各种关联。

**注：在数据模型资源手册卷1和卷2中所介绍模型中以及在本书第9章中企业数据模型这一节中，我们在当事人关系[PARTY RELATIONSHIP]和当事人关系类型[PARTY RELATIONSHIP TYPE]中使用了术语关系[relationship] 而非关联[association]。之所以这样是因为业内对于关系[relationship]更加熟悉，并且在语义上也很适合当事人；但是，使用这两个术语的任何一个都是可以的。如果你的企业使用了不同的方式来表述此类关联，我们鼓励你使用术语关系[relationship]。**

### 该在什么时候使用这种模式呢？

我们在以下情况中使用这种模式：

* **需要为层级关系、对等关系和聚合关系提供非常灵活的解决方案时。**新类型的关系可以添加到**关联实体类型**[ENTITY ASSOCIATION TYPE(s)]中，并且不需要更改模型。
* **需要使用统一的建模方法捕获所有的关联时。**这种模式可以保持各种关联的记录和管理的一致性。此外，围绕通用的**关联实体**[ENTITY ASSOCIATION]和**关联实体类型**[ENTITY ASSOCIATION TYPE]，可以创建一组通用的业务规则。
* **需要能够容纳任何类型的递归关系时。**这种模式会容纳层级关系、聚合关系和点对点关系。因为所有的递归关系都被规定为是多对多关系，而这种模式也可以容纳一对多关系，而且这种模式考虑到了关系的基数可能会发生改变。

### 这种模式的弱点是什么？

这种模式有如下缺点：

* **难以理解。**有些人可能会觉得这种模式更加难以理解，并且管理起来也不容易。
* **由于这种模式处理了所有的递归关系。**所以这种模式没有维护每种递归关系特定的数据和关系。或许不同类型的递归关系具有不同的属性。例如，除了不能执行一对多的规则外(例如，在层级架构中，一个下级只能有一个上级)，还有某些递归关系只需建立成一对多模型，而对于这样的递归关系如果使用多对多模式的话，未免大材小用了。
* **越是泛化，就越不清楚每个关联类型表示的是哪种递归，也就越不清楚在不同的情形中应当会发生什么？**在二级模式中，有个“分解”类型的架构，也就是层级架构。层级架构暗含了所属关系，所以如果你从层级架构中删除了“企业数据仓库[Enterprise Data Warehouse]”这个所有者，那么也就相当于删除了整个项目。即使你只是将该项目的名称改成“EDW XYZ Corp”,这个举动也对整个层级架构产生影响。在这个模式中，你正在捕获的“版本”架构，也就是对等类型的架构。这类机构并没有暗示所属关系。所以删除一个任务是不会导致你将其他任务也删除了。换句话说，你必须小心管理和说明**关联实体**[ENTITY ASSOCIATION]中的各种架构。在一个层级关系中，当你删除了最顶层的节点后，你是否相当于删除了所有的子节点了呢？而在对等关系中，你是不会删除所有相关的元素的。在聚合关系中，你是可以删除最顶层的节点的，即使这样做也不会导致子节点的删除。不同的规则适用于不同的架构。你要么使用一个“类型”实体来表示架构的类型，要么使用一个属性来标识该架构类型，比如说层级架构、对等架构或聚合架构。
* **关联实体[ENTITY ASSOCIATION]和关联实体类型[ENTITY ASSOCIATION TYPE]没有捕获递归关系的规则。**表4-7中的示例展示了实体的两个相关实例和某一类型的关系，即在“分解”架构中，“企业数据仓库”是“规划的”父层级。在有些情况下，这些信息可能是不够的。例如，在此关系中，两个实例之间可能存在一个行为。在产品分解架架构中，你可以看到一组部件可以构成产品层级架构。但是捕获产品层级架构中的组成成份，并不是因为要使用这些成分来生成最终产品，而是因为这些成分必须排除在制造产品之外。这些成分或是“互补的”或有一种“暗示意思”，即，假如有产品A，也就表示包括了产品B。在这个模式中，不用特地捕获这些支配关联行为的规则。你只需捕获现存的关联类型。
* **这种模式不能维护层级结构或聚合结构中的可选性规则。**例如，除了最顶层的实体，所有的子实体必须具有一个父实体，这种模式是不能执行这个规则的。
* **实体的各个层级对于可选性具有不同的规则，这种模式也不能维护这种情况。**例如，可能有规则规定**任务**[TASK(s)]必须有预计小时数，但是**项目**[PROJECT(s)]和**阶段**[PHASE(s)]却是可以有预计小时数，因为预计小时数可以从相关的任务[TASK(s)]中推断出来。这种模式表示任何**工作计划类型**[WORK EFFORT(s)]的**工作计划**[WORK EFFORT TYPE]都必须要有预计小时数。
* **这种模式不能维护每个层级的特定关系。**例如，**项目**[PROJECT(s)]有**赞助商**[SPONSOR]；而**任务**[TASK(s)]和**阶段**[PHASE(s)]则没有；或只有**任务**[TASK(s)]是“有特定人员实现的”。
* **不同类型的实体具有不同的特定属性，用这种模式来处理这些实体可能会更加困难。**例如，如果**任务**[TASK(s)]有个实际工作小时数，而**阶段**[PHASE(s)]和**项目**[PROJECT(s)]却没有。

### 简介

在这一节中，我们介绍了一种非常灵活的方法，用于维护和创建层级关系、聚合关系和对等关系。这种模式适合于以下企业：具有动态业务环境的企业，或是那些对于企业可能即将需要的各类递归关联没有一个良好的认识的企业。

这种模式的重要意义在于：所有的关系都是用完全一样的方法来处理的。由于这个特征，添加其他的数据模型架构也变得更加容易，比如说维护规则、构建通用的服务或常规。此外，使用一个方法处理数据社区中的所有的递归关系，这种模式有助于培养这种能力。数据社区中的那些递归关系也无需担心一对多架构。同时他们只在一个地方处理任何特定实体所需的所有递归架构。

这种模式的意义也很重大，因为它被明确地构建成“包容变化”的模式。很容易添加新类型的递归关系。

这种模式可以帮助你，因为它一旦被实现了，很少需要改变它，它以完全一样的方式支持各种递归架构—层级架构、聚合架构、对等架构以及任何你需要用到的新类型的关联，比如说组合和集合【注解3】。

这种结构更加的泛化，其概念解释起来甚至是让数据专业人士都感到困惑， 一般情况下你不会使用这种模式作为作用域声明的一部分。这种模式也可以帮助数据建模人员“捕获所有的”情形。然而，这也就意味着建模人员可以偷懒了，因为为不同的递归关系记录的业务规则，全都不用他们捕获了。

在数据模型的一个地方捕获各类递归关系，你可能还需要给每个结构“贴上标签”，以表明该结构的类型。毕竟，维护层级架构的业务规则与维护对等关系的那些规则是不同的。

除非你为**工作计划**[WORK EFFORT(s)]创建特定的子类（例如，**项目**[PROJECT]、**阶段**[PHASE]和**任务**[TASK]），否则这种模式是不能支持不同类型的**工作计划**[WORK EFFORT(s)]所具有的特定关系或属性。例如，假如**项目**[PROJECT(s)]是唯一的一个可以捕获预收益率[booked ratio]与实收益率[burned ratio]的（前一章节中提到的）工作计划类型，该怎么办呢？假如**任务**[TASK(s)]与**人员**[PERSON]之间有一种特定的关联，又该如何呢？

## 维护规则的三级递归模式

在维护规则的三级递归模式这一节中，我们将介绍另一种处理递归关联的灵活方案。我们引进了一种新的方式来审视那些不同关联之间的关系。在前一章节，我们是使用**关联实体类型**[ENTITY ASSOCIATION TYPE]来将关系进行分类的。例如，表4-7中我们捕获的关联类型有：“优先级关系[Precedent]”、“版本关系[Version]”、和“分解关系[Breakdown]”。对于大多数企业而言，这些关系类型已经足够了，但是，对有些企业来说，这些“分类”架构可能还不足以满足其需求。

### 为何我们需要这种模式？

维护规则的三级递归模式满足了企业对于灵活性方案的需求，这个方案既考虑到了递归关系的分类也考虑到了管理那些递归关系的规则的创建。在表4-7我们创建了以下工作计划关联类型[WORK EFFORT ASSOCIATION TYPE]：“优先级关系[Precedent]”、“版本关系[Version]”、和“分解关系[Breakdown]”。这些关系类型确实是在对关系进行归类么？还是它们这是在描述一种行为？或是这两方面均包括？

从某种意义上来说，“分解关系”类型描述了一个行为，也就是，“企业数据仓库[Enterprise Data Warehouse]”可以分解成“规划[Mapping]”,而从另一方面来说的话，可以将“分类关系”这种关联归类到工作计划分解架构中，使其成为其中的一员。这两种解释看是十分合理其实都存在缺陷。但是“分解关系”是一种分类还是一个规则。还是说两者都是呢？这种模式可以将递归关系的分类同管理递归关系行为的规则区分开来。

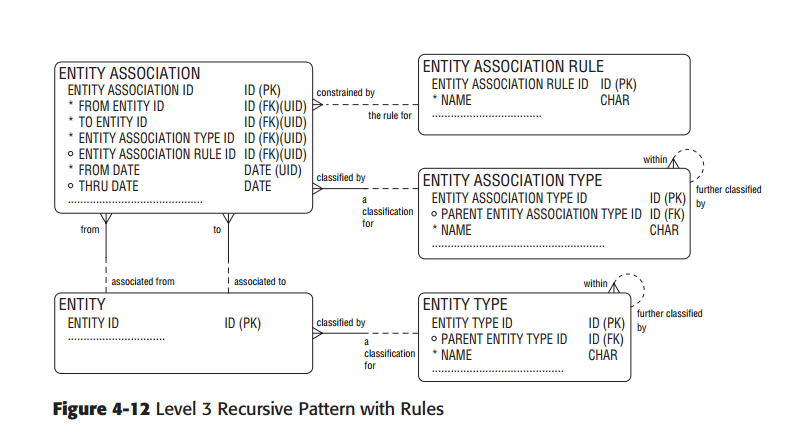
例如，两个零配件是可替换的两配件。可以具有这样一个规则：这两个另配件可以用于制造某个特定的手机。“备份零配件”关联类型可以是递归关系的一个分类。虽然这些是备份零配件，但是我们并有指定一个规则，规定这部手机的某个零配件可以取代其他零配件。某一零配件可以取代其他零配件，在制造手机的背景下，可以将这个事实当成一个规则，这个规则管理了这两个零配件之间的行为。如果企业做出了不同的选择，指定了不同规则，规定：这两个零部件可以明确地“排除”在可彼此替换零部件之外。再者，“备选零部件”关联类型对递归关系进行了分类。可以将“排除[Exclusion]”或“替换[Substitution]”看成是一种规则，在企业生产制造的过程中，这个规则规定了这两个零部件的行为。

### 这种模式是如何进行工作的？

图4-12除了添加了一个关联实体规则，基本上与图4-10一样。正如前一章节所示，关联实体类型被定义为关联实体的分类。关联实体规则可以定义成“支配某种关联行为、活动或流程的原则和规则”。

在三级关联模式中，“每个**实体**[ENTITY]的关联可以源自一个或多个**关联实体**[ENTITY ASSOCIATION(s)]或每个**实体**[ENTITY]可以关联到一个或多个**关联实体**[ENTITY ASSOCIATION(s)]，而每个**关联实体**[ENTITY ASSOCIATION]可以通过一个且只能通过一个**关联实体类型**[ENTITY ASSOCIATION TYPE]来进行分类”。在现在的图4-12中，可以看出“每个**关联实体**[ENTITY ASSOCIATION]可以为且只能为一个**关联实体**[ENTITY ASSOCIATION RULE]所约束”。

那么，**关联实体类型**[ENTITY ASSOCIATION TYPE]与**关联实体规则**[ENTITY ASSOCIATION RULE]有何不同之处呢？对于很多企业而言，这两者并无差别。在前一节中，你见识了XYZ公司是如何将工作计划关联[WORK EFFORT ASSOCIATION]分类成“分解关系[Breakdown]”、“优先级关系[Precedent]”和“版本关系[Version]”的。“分解关系[Breakdown]”按照事物的类型对工作计划[WORK EFFORT]进行分类的，也就是一个分解架构。“优先级关系[Precedent]”是按照行为进行分类的，即每个工作计划必须优先于另一个工作计划。有些企业可能会选择将类型同规则区分开来建模。这么做的好处是：企业可以精确地了解一种关系的不同实例彼此之间是如何相互作用的，还可以捕获到那些相关实体的类型。

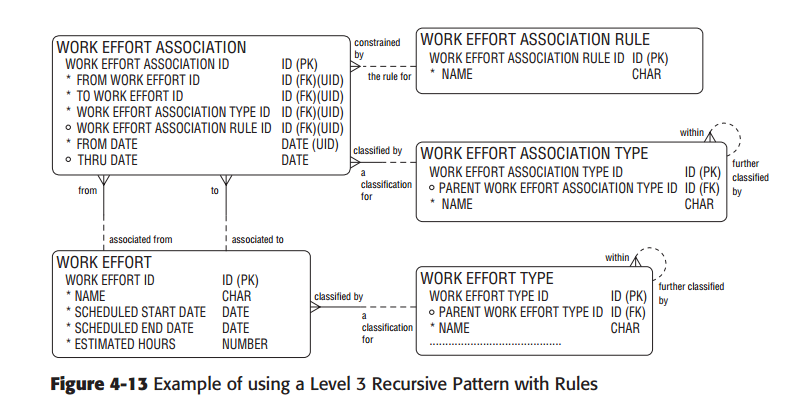


**图4-12** 维护规则的三级递归模式

为了展示XYZ公司管理**工作计划**[WORK EFFORT(s)]的另一种方式，数据专业人士制作了图4-13。这个图与图4-11非常相似，然而，工作**计划关联规则**[WORK EFFORT ASSOCIATION RULE]的添加，使得我们可以定义工作计划彼此相互关联的方式和现存的规则类型，而这些类型的规则支配了这些实例的行为。

通过工作计划关联类型，你可以捕获各种关联。关于这一点，比在前一节就已经见识到了。以下示例是工作计划关联类型的实例：

* **工作分解架构**：当较高层级上的工作计划被进一步细分成各个小的工作计划时。例如，项目可以分解成阶段，而阶段可以分解成任务，以此类推。
* **项目聚合架构**：一个计划下有一个项目集合。
* 对等工作计划架构：对等关系的各个工作计划位于工作计划架构中的同一层级上，比如说一组数据仓库工作计划。例如，在预算和业绩方面，对等工作计划可以作为彼此相互衡量比较的一个基准。



**图4-13** 维护规则的三级递归模式的应用范例

关联实体的规则规定了关联的两边是如何关联到其他实体的。所以在图4-13中，你可以看到工作计划关联规则概括了存在于两个工作计划之间的各种的行为，例如：

* **替换**：例如，一个任务可以替代另一个任务
* **排除**：例如，某个特定项目不能够称为某个特定计划的一部分，或某个特定任务被“排除”在特定阶段或项目之外。
* **并发**：两个工作计划必须同时进行
* **优先**：某一工作计划炫耀在另一个工作计划启动之前完成（部分或全部）。
* **互补**：两个工作计划在一起相得益彰，例如，两个项目便可以互助，比如说元数据项目有助于数据管理项目，而反之亦然。在这个示例中，若元数据项目取得更大的进展，数据管理项目便也会取得更大的进展，反之亦然。

这个工作计划关联是否能够帮助XYZ公司更好的管理业务呢？它对于业务的管理起到相当大的作用。如果观察表4-8，你可以看到图4-13架构的实例。这些实例演示了维护工作计划关联的分类和规则的非常全面的方案，该方案就是XYZ公司构建工作计划关联[WORK EFFORT ASSOCIATION(s)]所采用的方法。

**表4-8**工作计划关联类型和工作计划关联规则

**（表）**

首先，你可以看到在“工作分解架构[Work Breakdown Structure]（**工作计划关联类型**）”中有个名称为“系统分析[Systems Analysis]”的工作计划,而该工作计划包含以下几个部分：“创建初始声明范围[Create Initial Scope Statement]”、“创建源系统”[Create Source System Inventory]和“发掘目标需求[Discover Target Requirements]”。**工作计划关联规则**没有捕获这三个角色的特定的行为。换句话说，“系统分析”有三个子工作计划，分别是“创建初始声明范围[Create Initial Scope Statement]”、“创建源系统库存[Create Source System Inventory]”和“发掘目标需求[Discover Target Requirements]”。这个架构可以归类为“工作分解结构[Work Breakdown Structure]”。我们没有给他们之间的关系分配任何的规则。**工作计划关联规则**[WORK EFFORT ASSOCIATION RULE]不是强制性的，所以不是每个关联都必须要有一个规则。

表的第四行中，“创建源系统库存[Create Source System Inventory]”和“发掘目标需求[Discover Target Requirements]”属于 “对等工作计划[Peer Work Efforts]”关联类型,并且这些工作计划有一个“并发”规则，来表明这两个工作计划需要同时执行。我们并不是随口说说将彼此相关的工作计划“创建源系统库存[Create Source System Inventory]”和“发掘目标需求[Discover Target Requirements]”归类为“对等工作计划[Peer Work Efforts]”，我们还存储了名称为“并发[concurrent]”的**工作计划关联规则**[WORK EFFORT ASSOCIATION RULE]，用于表明这两个工作计划必须一起进行。

你或许会问：不能够在前一种模式中这样做吗？不能分为两行么：一种为“对等工作计划[Peer Work Efforts]”关联类型，另一种为“并发关系[Concurrency]”， 但是“并发[Concurrency]”与“对等工作计划[Peer Work Efforts]”是同一个概念吗？对于“对等工作计划”我们可以将其考虑成关联类型的一个类别。但是“并发关系[Concurrency]”是关系类型的一个类别码？还是说它更接近于一个规则，用于约束两个任务之间的行为？有些企业认为将关系的分类与影响实例行为的交互方式区分开来具有重要的意义。

表的第五行也展示了这两个相互关联的工作计划，其工作计划关联类型是“对等工作计划”，这意味着在工作分解架构中这两个工作计划属于同一层次。你还可以从这行看到他们是“互补关系”。这点表示在某种程度上，这两个工作计划可以相互帮助。这表明了作为对等体的这两个工作计划是彼此相互关联的，还有一种说法是在某种程度是这两个工作计划可以互补的，因为我们可以从有助于其他工作计划的工作计划中学习到一些东西。虽然两个工作计划可以是对等的实体，但是不以任何形式相互补充。

**注：图4-12的替换方案考虑到了很多的规则，因此，使用了多对多关系替代了关联实体规则与关联实体之间的一对多关系。例如，可以将“创建源系统库存[Create Source System Inventory]”和“发掘目标需求[Discover Target Requirements]”归类到“对等工作计划”关联类型中，然后该类型可以有两个规则：“并发”和“互补”规则。因此，相同的工作计划关联可以拥有很多的规则。**

在表4-8中，你可以以看到一个新的**工作计划**[WORK EFFORT]，其上级标识是“9012”，名称是“创建源系统需求[Create Source System Requirements] ”。这个**工作计划**与“ 发掘目标需求[Discover Target Requirements]”相关联。此关联属于名称为“对等工作计划”这一**工作计划关联类型**[WORK EFFORT ASSOCIATION TYPE]。这表示这两个工作计划是两个可以组合在一起进行的任务，因为这两个工作计划位于同一个层级；换句话说，他们是对等的实体。他们也有**工作计划关联规则**[WORK EFFORT ASSOCIATION RULE(s)]“替代”。XYZ公司决定将“创建源系统库存[creating a source system inventory]”作为一种替代方案，认为使用此工作计划替代“发掘源系统需求[Discover Source System Requirements]”是可以接受的。他们创建这个规则的起始日期是“2009年1月1号”。

在表4-8中你还可以看到工作就计划“数据管理计划[Data Management Program]”低下有两个相关的工作计划：“企业数据仓库[Enterprise Data Warehouse]”和“客户主数据项目[Customer Master Project]”。

表4-8的最后一行展示了“硬件购买[Hardware Purchase]”，在其“工作分解结构[Work Breakdown Structure]”元素的“企业级数据仓库[Enterprise Data Warehouse]”的背景下，具有一个“排除[]”的工作计划关联规则[]。这表示XYZ公司明确表明工作计划[]不应当是“企业数据仓库[]”项目中的一部分。因为该公司已决定不会为数据仓库项目购买任何硬件。不包含“硬件购买”或不将“硬件购买”关联到“企业数据仓库”项目，这样做是否会更好些？通过将“硬件购买”标识为排除在外，XYZ公司明确表明了该公司将不会包括购买硬件这一工作计划。这与根本不捕获关联是不同的。

**注：当创建产品或零部件的设计时，这种模式是很有用 。关于产品各个组件之间的交互有很多的规则。比如说替代规则、不兼容规则、互补规则和过时规则。**

**注：关联规则不只是局限于自关联（递归关系）。一个、两个或多个实体之间的任何关联都可以具有规则，这些规则不仅仅是将这些关联进行了分类，还支配了这实体的行为。但是当处理递归关系时，这些规则似乎出现的更加频繁。**

### 该在何时使用这种模式？

我们在以下情形中使用这种模式：

* **需要定义影响递归关系的行为。**正如上一个模式，三级递归模式所示，这个需求也可以通过关联类型来完成，但是这个模式具有不同的语义观点，它明确地将关联的类型与关联的规则区分开来了。
* **需要根据此模式所使用的数据创建和驱动应用。**例如，应用可以检查程序，以确保具有“优先级”规则的工作计划有序执行，当相关的首要工作计划完成的时候才能够启动其他的工作计划，或该应用可以确保任何运算均不包含工作计划的数量，这些已经通过工作计划关联规则被“排除”在外。
* **为了对帝业有更好的了解，需要捕获企业的规则。**当管理数据时，如果恰好是一个规则驱动型的企业，那么这种模式将可以支持这一特质。

### 这种模式有哪些缺点？

这种模式的缺点如下：

* **这种模式是一种更加概括形式的模型。**越是概括，其架构的使用方式就越令人困惑。需要仔细确定所有股东都能对该架构有一个透彻的理解。
* **这是一个十分崭新的概念，要获得企业的认可尚需一些时日。**很多企业认为这是种矫枉过正的模式。然而，我们以成功在规则驱动型企业中实现了这一模式。

**注：这种模式的优缺点类似于前一种模式。最后一部分只将重点放在了添加工作计划关联规则[WORK EFFORT ASSOCIATION RULE]的优缺点上。**

### 简介

在这一节，介绍了另一种看待关系的方式，这些关系存在于递归关系中的不同关联实体之间。介绍了我们是如何定义不同关联实体之间行为的。维护规则的三级递归模式满足了对解决方案灵活性的需求，这种方案可以对关系进行分类，还能创建规则，用于支配关系实体的行为。

**关联实体类型**[ENTITY ASSOCIATION TYPE]被定义为**关联实体**[ENTITY ASSOCIATION]的类型，比如说“工作分解架构[Work Breakdown Structure]”、“计划集合[Program Aggregation]”和“对等工作计划[Peer Work Efforts]”。可以将**关联实体规则**[The ENTITY ASSOCIATION RULE]定义为“支配关联的行为、活动或流程的原则或规则”例如，“替换规则[Substitution]”、“排除规则[Exclusion]”、“并发规则[Concurrent]”以及“优先规则[Precedent]”。

这一特定模式特别适用于那些业务规则驱动型企业 。然而，这是一种更抽象的模式，有很多新的概念。可能对于很多企业而言，这种模式过犹不及或是太过复杂。

## 模式的总结

**表4-9**包含了本章节所提到的模式的简介

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模式** | **说明** | **何时使用** | **何时不使用** | **优点** | **缺点** |
| 一级递归模式 | 为递归关系建模的一种特定方法。不同的实体代表层级架构中的不同层级。实体之间的关系展示了一个架构中的不同层级是如何相互关联的。 | 作为声明范围的一部分。  帮助非技术业务代表的开发和验证数据需求。  当需要一个静态的层级或聚合架构时  使得数据建模工作的启动更加简单，有助于阐释所用的术语以及所出现的数据问题  当一个架构的不同层级具有不同的数据和关系时 | 需要设计很灵活时，例如，需要给机构添加其他的层级。  为对等关系建模时。 | 利于理解和使用。作为声明范围的一部分。  给层级架构中的各层级之间的关系提供了业务规则。  更加解决了特定的业务需求。  层级或聚合架构的各层级可以具有特定的属性和关系。  在聚合架构或层级架构的不同层级中，可以使用不同的可选性。 | 不是灵活的模型。不同的层级可能会包含类似的信息（例如，在项目、阶段和任务层级上的都有一个预计小时数属性），如果数据定义不一致，就可能会引起混乱。  这种模式不支持对等关系。  这种模式不支持多对多关系，也不执行一对一关系。 |
| **模式** | **说明** | **何时使用** | **何时不使用** | **优点** | **缺点** |
| 二级递归模式 | 这种模式支持任何类型的一对多关系，并具有一定程度的灵活性，允许给层级架构或聚合架构添加新的层级。 | 当你需要一种灵活性 的解决方案，在没有不用改变模型的情况下，可以为分层架构或聚合架构添加新的层级。  当分层架构或聚合架构的各层级具有相同或类似的属性和关系时。  当你只需要维护一对多关系时。 | 当你需要维护分层机构或聚合架构各层级上的特定属性和关系时。（然而，有些特定属性和关系可以使用此模式中的子类来维护。）  当分层结构或聚合架构的各层级之间具有不同的可选性时。  当需要维护多对多递归关系时。 | 是一种维护递归关系的灵活方法，允许添加任意数量的其他层级。  可以将其当作递归关系建模的通用方法，因此，企业可以制作通用的规则并能够以一致的方式处理递归关系。  因为这种模式捕获了一对多关系，所以它可以维护一定的规则， | 由于比一级模式更加抽象，所以使用这个模式开发或验证数据需求更加困难。  不是十分的灵活，不能添加其他的可能会出现的递归关系类型。例如，如果需要一个新类型的递归关系，比如说“不兼容关系”，我们将需要添加另一个递归关系。  可以让分层架构的各层级维护冗余数据。例如，预计小时数可以更具较低层级的属性值推断出来，但是这种模式的所有层级都包含这个属性。  不能够维护各层级的特定数据需求（各层级上不同的关系的可选性，不同层级所具有的不同的属性和关系）。  强迫架构的所有层级具有相同的属性，即使各层级的属性存在不同之处（虽然这个为题可以通过使用子类来调解）。  不能为业务规则建模，这些规则管理什么是被允许的，例如什么关联是相互排斥的（等等）。  不维护层级间关系的特定业务规则。（例如，不显示特定的关系和规则，比如说项目是由一个或多个阶段构成的）。 |

**（续）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模式** | **说明** | **何时使用** | **何时不使用** | **优点** | **缺点** |
| 三级递归模式 | 三级递归模式提供了非常灵活的解决方案，随着时间的推移会出现各种递归关系，这种模式允许动态地添加这些递归关系。 | 当你需要非常灵活的解决方案时，因为随着时间的推移，需要新类型的递归关联或发掘了新类型的递归关系。  当需要支持任何类型的递归关系，可以通过将递归关系抽象成多对多关系。  当需要以非常一致的方式捕获各种递归关系时。这个模式有助于为所有的角色创建一组通用的规则，可以围绕通用的实体关联和实体关联类型来创建。  作为实现的基础，易于适应变化，例如，可以在没有更改模型的情况下，能够添加新的关联类型。  当层级架构、聚合架构或对等架构的各个层级具有相同或类似的属性和关系时。 | 作为非技术业务代表声明范围的一部分。  作为一个模型的基础，帮助非技术业务代表开发或验证数据需求。  需要捕获架构中各层级上的特定业务规则。  当层级架构或聚合架构的不同层级具有特定的属性和关系时。（这个为问题也可以使用子类来解决。）  当分层架构的各层级之间具有不同的可选性时。 | 提供了灵活性，可以维护任意数量的递归关系类型。  当新类型的递归关系出现时，能够在不更改模型的情况下添加这些关系。  可以将其作为为递归关系建模的一种通用的方式，因此企业可以创建通用的规则和以一致的方式来处理这些递归关系。  在变幻多端的动态环境中这种模式可以充当有效的实现基础。 | 这种模式很难为非数据专业人士所理解，它的抽象程度增加一个另一个层次。  越是概括的模式，就会越容易导致困惑，例如，可能会导致曲解每个关联类型所代表的内容。  这种模式不维护关联的业务规则比如说那些实例是互相排斥的（等等）。  在层级架构的不同曾加上可能会产生冗余数据。例如，预计小时数可以更下一层级的属性值推断出来，但是这种模式的所有层级都维护了该属性。  不维护各层级特定的数据需求（例如，在不同的层级上需要不同的关系可选性、不同实体中的不同属性和不同的关系）。 |

**（续）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模式** | **说明** | **何时使用** | **何时不使用** | **优点** | **缺点** |
| 维护规则的三级递归模式 | 这种模式支持以概括的方式维护三级递归模式中的递归关系，但是引进了维护关联的规则这一概念，这种模式将递归关系的分类与支配递归关系行为的规则进行了区分。 | 当企业希望灵活地捕获不同的递归关联时，当新的关联出现时，在不需要更改模型的情况下，可以适应其他类型的关联。  当企业希望捕获支配架构中各元素行为的规则时。  当企业希望走向基于规则的数据架构时 | 当企业不赞同或不需要关联之间的规则时，当企业想将这下规则当作关联类型来处理的话，如同前一个模式的处理方式。当层级架构或聚合架构的各层级上具有特定的属性和关系时。（这个问题也可以使用子类来解决）。  当层级架构或聚合架构的各个层级之间存在不同的可选性时。 | 对于递归关系实例，这种模式可以为维护规则。  促进数据驱动的应用程序，可以参考这些数据架构，而不是在应用中硬编码这些规则。  有助于更好地理解企业中的规则。  可以帮助你捕获递归架构中不同关联的丰富的信息和规则。  在不断变化的动态环境中这是一个很好的实现，例如，随着时间的推移，会出现其他类型的递归关系和规则。 | 这种模式是本章节中最复杂和最抽象的模式，因此会更加地使人困惑，并可能更难获得认可。可能这种模式的应用会过犹不及或者不需要这种模式，这得视具体情况而定。  不维护各层级的特定的数据需求（例，在不同的层级上需要不同的关系可选性，不同的属性以及不同的关系）。可能会在层级架构的各层级上产生冗余数据。例如，预计小时数可以根据较低层级的属性推断出来，而这种模式的每个层级都包含了这个属性。 |

**参考文献**

1 想了解更多的组织数据请参看本书的第五章节

2 This definition was taken from Ananda Amatya（March 1999). Available at http://www.dcs.warwick.ac.uk／~ananda/umlNotes /node15 5.html.

3 这个定义源自Alf A. Pederson's article “Entity Relationship Modeling” (April 2005). Available at http://www.devarticle.com/c/a/Development-Cycles /Entity-Relationship -Modeling.

4 此定义源自在线免费计算机专业词典(1993 -2008 Denis Howe) Available at http:// foldoc.org/index.cg? query =erarchy & action: Search.

5 Definition taken from Dictionary.com.

6 Please refer to Chapter 6 0f *The Data Model Resource Book, Revised Edition, Volume l, A Library of Universal Data Models* for all Enterprises, by Len Silverston(Wiley. 2001) for a discussion of this specific example.

7 这个定义的出自于the Universal Data Models Repository. Universal Data Models 2001-2008.

8 You can find a very interesting discussion on many-to-many recursive relationships at the Steve Hoberman & Assoaates web site At http://[www.stevehoberman.c](http://www.stevehoberman.c)om/DesignChallenge/challengellresponse.htm.

9 You can find good guides for the creation and resolution of many-to-many relationships in CASE\*METHOD: *Entity Relationship* Modeling by Richard Barker (Addison Wesley, 1990) or *Data Modeling Made Simple* by Steve Hoberman (Technics Publications, 2008).

10 See *The Relational Database* by John Carter (International Thomas Computer Press, 1995) and *Database Systems*: *A Practical Approach to Design, Implementation, and Management,* 3rd Edition, by Thomas M. Connolly and Carolyn .Begg Addison-Wesley, 2004).

【注解1】booked ratio 暂译为“预收益”

【注解2】burned ratio 暂译为“实收益”

【注解 3】compositions and collections 暂译“组合和集合”