# 状态：数据的状态

随着商业世界复杂性的增长，企业的流程也随之变得更加复杂。提供服务的计划或项目变得更加棘手复杂，需要我们更加详细地管理这些工作计划的状态。财务信息，比如说期权价格或利率的变更，是受各种信息影响的，而这些信息都具有不同的状态。当将状态添加到企业的计算机系统中时，捕获数据结构和管理数据状态在当今的现代企业中变得尤为重要。

当数据自然地历经各业务流程时，可能会通过很多不同的方式创建、更新或删除大量的数据状态，例如，一个大型电子零售商接到一个1,000部手机的订单，在接收这个订单后，可以将其提交到订单录入系统。然后主管可以确认订单。如果提交订单无误，订单录入系统将会查看提供给该客户的库存是否充足；如果库存充足，订单将会触发物流系统，设置装运。订单发货后，计费系统生成付款请求，或换句话说，生成发票。在此案例中，手机订单经历了如下几中状态：“接收”、“录入”、“确认”、“待定（等待库存检查）。”“已发货”和“已出发票”是不是订单的状态呢？还是说他们是关联的装运和发票的状态？订单可能还有很多其他的状态，比如说‘取消’、‘信用检查失败’等等。随着流程的变更，订单还可能需要其他的状态，比如说“可能存在欺诈”或“等待主管批准”。

在前一个案例中，生成了很多的事务，当订单的事务从一种状态到另一种状态时，订单的状态随着不同事件的出现而发生变更。所有流程中的不同的相对时间里，各实体可以具有一个或多个状态。这些状态表明一些业务情况或状况，比如说“录入”、“取消”等状态。对于任何一个企业而言，具有清晰的管理状态策略是至关重要的。否则又如何得知在其业务内发生了什么呢？

数据专业人士在采访主题专家时，遇到一个常见的问题是：随着各种事情或事件的发生，很难阐释和理解各种状态。例如，操作人员可能会说他们的工作是完成“未交付订单”。对于操作人员而言，订单要么是“未交付订单”要么就是“已交付订单”。他们所见到只是整个生命周期当中的小部分。所以，当他们进行工作时，订单总是有待付运，而当他们完成工作时，订单将变成已交付，剩下的则是其他人的问题了。财务部可以认为订单有两种状态，有偿或无偿。换句话说，对于财务部门而言，订单有两种利息状态，“有偿”或“无偿”。对于数据专业人士而言，当订单经历完整的订单履行生命周期时，它可以处于我们所提及的所有状态，且还要添加更多的状态。

在数据建模任务中，当我们审视现存的数据库设计时，我们发现有很大比例的字段都是日期[date]和日期时间[datetime]，而在这些字段中有很多是表示状态的。例如，可能会有以下字段：产品引进日期、人员雇用日期、发票日期、手机呼叫开始日期时间以及很多其他的日期和时间字段。我们还发现通常这些时间字段的建模方式十分不同，没有考虑到：在企业内，以一致的方式管理很常见且很重要的数据，是非常有利的。在本章节，我们将采取一致的方式为这些状态数据建模，以解决这样的需求。

## 此类模式的意义是什么？

我的订单发生了什么事？加快订单需要花多长时间？有多少发票过期了？某一产品的当前状态是什么；例如，是已引进，还是即将断货【注释1】等等？这些都是关于状态的常见问题，是企业可能会问其自身的问题。了解重要业务数据的状态和了解该状态的日期和/或时间是唯一的一种可以准确地回答这类业务问题的方法。了解状态何时被创建、改变、或删除也并不总是适用；在有些情形中，仅了解一条业务信息的状态就足够了。例如，你只想知道状态为“损坏而无法修复”的库存项，而至于该库存项是何时损坏的，可能并不重要，尽管根据给情形而言，其可能是很重要的数据。通常，一个重要的时间字段，要么维护日期、日期和时间，要么维护起始日期和终止日期，该时间字段通常与状态相关联。因此，关于状态，需要回答的基本问题有：

* 实体可以有哪些状态，也就是，状态有哪些可行的值？例如，当事人、产品、库存项、订单、装运、发票、支付、账户交易、预算以及数模型中的诸多其他实体都有哪些潜在的状态？
* 某一特定实体的现在的状态是什么？过去的状态是什么？例如，一个订单是“未交付”、“已交付”、“待定”、“即将发货”？对于这个问题，你可能想从可用的状态中选择一个状态，然后将此特定状态指定到该实体。
* 你需要捕获状态的哪些特定时间？你是否需要捕获状态出现的时间（或日期与时间），或预期出现的时间？例如，订单是在何时“未交付”、“已交付”、“待定”的，以及此订单“预计发货”时间是什么时候？你是否需要捕获状态的起始和结束时间（开始到结束所经历的时间）？你是否只要捕获某一事件发生时的特定日期与时间？换而言之，你是否需要知道该状态何时开始存在，而在什么时候又会停止，或你是否需要知道状态出现或改变的时间？例如，你可能需要知道订单在“2009年3月1日下午2点”时是“未交付”状态，而在“2009年3月4日下午3点”该订单的状态时“已交付”，还有其他的订单状态，比如说“已接收”，你可能只需要知道接收订单的具体日期与时间。

## 本章节有哪些内容？

本章节先定义了什么是状态。然后介绍了不同层级的数据建模模式，这些模式支持以一致的方式为状态建模。每个模式都支持状态的三个基本方面。

* 特定实体所允许的状态
* 实体的状态是什么？（或假如该实体在同一时间具有多个状态，那么这些状态是什么）
* 该状态什么时候开始生效，何时发生变更或在何时不复存在（如果需要的话）。

本章节从最特定的风格开始（一级状态模式），逐渐过渡到非常泛化的风格（四级状态模式）。

在本章节的最后提供了两种模式，“状态类别模式[Status Category Pattern]”和“维护多重隶属关系和规则的状态类型模式[Status Type with Multi Rollup and Rules Pattern]”，这些模式是二级、三级或四级模式的加强版。这些模式通过引进第5章节和第4章节中的模式来强化状态模式。“状态类别模式”支持将状态分类成各种集合这样的需求，当一个实体具有多组状态时，这种模式是很有用的。例如，订单可以具有一组订单处理状态（“接收”、“录入”和“确认”），还可以有一组计划进度状态（“按时完成”、“进度落后”和“过期”）。“维护多重隶属关系和规则的状态类型模式”支持维护规则的需求，这些规则是关于在什么样的条件下状态可以（或不可以）相互关联。换句话说，你可能会想维护这样的规则：在“接收”订单之前，是不可以“录入”订单的。

**注：在“状态类别模式”和“维护多重隶属关系和规则的状态类型模式”中使用了相同的概念，即使用其他模式增强某一模式，其中特别是添加了灵活的自关联（递归）模式和分类模式，在本书中也可以将这种概念作为一种技术，用以强化其他模式。**

大多数企业可以使用本章节中的数据模型模式建立一致的数据模型，这些数据模型支持维护状态数据的各种需求。

总之，本章节包括以下几项内容：

* 状态这一概念的定义和介绍
* 每种模式的关联性和重要意义
* 不同层级的状态模式
* 该在何时使用，何时不使用这些状态模式
* 深入分析每种模式

所有模式的简介，优缺点以及何时该使用它们和何时不该使用它们

**注：在统一建模语言（UML）中，使用单词“state”来标示状态。在其他的上下文中，状态是一个核心概念，比如说在有限状态机、有向图以及其他类型的模型中。触发事务在统一建模语言中被称为转移。1**

## 什么是状态？

状态可以定义为“某一特定时间的状态；例如，‘失修的条件（或状态）’”2。状态是指可应用于数据的某些事态或情形。例如，一个账户可以是“开户”或“销户”状态。状态可以是指数据必须遵照的某些法定条件，比如说资产负债表科目必须处于“满足Sarbanes-Oxley法案条约”这种状态，或由于正在进行刑事调查，金融账户处于“冻结”状态。计算机程序通常使用状态来决定遵循某一特定路径。首席执行官们想知道在一段时间内生成了多少“新账户”客户或有多少项目是“过期的”。在数据模型模式方面这意味着什么呢？这表示实体实例可以处于（或已处于）某类状态，并且处于某种状态往往具有非常重要的业务影响。

可以将状态类型分组成多个特别的分类。例如，一个账户可能会有“账户开立”状态类型，比如说“请求”、“录入”和“开户”。账户还可以具有“账户维护”状态类型，比如说“休眠”、“激活”、“未激活”或“调查中”。有趣的是，这表示一个账户一次可以具有多个状态。换句话说，一个账户可以是“已开户”账户和“激活”账户（或“已开户”和“未激活”）。那么，在数据模型方面。这又意味着什么呢？这表示可以将每种类型的状态分成一个或多个状态类别。

实体实例在一个单一分类中是否可以具有多个状态类型？例如，账户在同一时间是否可以处于“录入”和“开户”这两种状态类型（这些都属于“账户开立”状态）？出人意料的是答案居然是肯定的；这种情况是可能的。那是因为实体的当前状态时主观性的。实体处于哪个状态取决于正在查看该状态的人。例如，一个被“请求”的账户，然后被“录入”，最后是“开户”，似乎该账户只有一种当前状态即“开户”。但是对于账户录入管理人员而言，该账户已被“录入”，这表示他或她的职员已完成这项工作，他或她可以报告那天“录入”了多少账户。对于账户管理人员，该账户已经被“开户”，这表明他或她可以报告这个账户的收益。换而言之，账户录入管理人员对处于“录入”状态的账户感兴趣，而账户管理人员则对处于“开户”状态的账户感兴趣。因此，该账户有时可以具有两个状态！不同的管理人员而言，会有不同的视角，这两种状态都是‘当前’状态。因此，你必须要注意，要认识到对于企业中的特定的人或组织而言，一个实体可以具有多个特定的“当前状态”（和历史状态），而横跨整个企业，你应当认识到实体在同一时间可以具有多个‘当前’状态。如此的话，就这些模式而言，这意味着什么呢？这表示你需要支持这样的事实：在同一时间或者是随着时间推移，一个实体可以与一个或多个状态相关联。

**注：在有些情况下，可能只需维护实体的一个‘当前状态’，对企业的所有组成成分而言，该状态都是‘当前’的。我们在二级状态模式，当前状态中支持这一点。**

有些情况中，当同种状态分类中（或在不同的状态分类中）有不同的状态类型时，可能需要规则来约束这些状态之间的行为。例如，一个客户可以处以激活状态或未激活状态，3但是该顾客不可以在同一时间处于这两种状态！换而言之，有些状态类型是互斥的。其他的一些状态类型可以相互替换。例如，订单处于“录入”状态和处于“完成录入”状态，实际上这些是源自不同应用的订单的两个状态类型，但是它们表示的是同一件事。这表示状态类型可以相互关联，捕获状态之间的关系特性是很重要的（这些状态时彼此的同义词吗？他们是相互排斥的吗？等等）。因此，模式可能需要支持某些规则，规定各状态之间的行为。我们在本章节的最后一部分，维护多重隶属关系和规则的状态类型模式中将解决这一问题。

很多状态是推导出来的。例如，典型的是具有多条订单项的订单，这些订单项可以处于各种状态（“缺货订单”、“已分配库存”等等）。你可能会认为订单或订单项也具有“已发货”状态。但是，订单（和订单项）一般是与装运（装运项）相关联的，所以“已发货”这一状态可以根据关联的装运状态推导出来。我们建议应用推导出订单状态这种方式（在此案例中，我们可以通过查看关联的装运，推断出“已发货”状态），而不要将这些推断的值作为实例来维护。可能会需要更改另一个实体的状态，但是重要的是不要为的冗余状态属性建模。例如，当将装运状态更改成“已交付”时，你不想在更改这个状态时还必须将订单状态更改成“已装运”，因为你可以推断出此状态，并且如果你在多个地方维护了这一状态，这些状态很可能会不同步。由于我们尽量不去冗余地维护数据，它可能意味着可以不直接在任何地方为某些状态建模，但是必须可以推断出这些状态。4

查看状态的另一种方法是假设可以将**订单**、**装运**、**支付**或其他实体看成支持重要业务处理的实体。在此事务内，可能会有很多的事件，这些时间改变、更新或完善该事务。例如，装运可能要支持“装运计划”事件，“装运包装”事件和“装运交付”事件。你可以使用数据模型资源手册卷1修订版（Wiley, 2011）第5章节中的装运数据模型，为数据模型中的装运建模。这有助于捕获完整的事务。但是你可能不是特别想在模型中使用一个实体处理所有的更新、改变或完善装运的事件。在你的“装运”数据模型中，是否有一个**装运计划**实体或一个**装运交付**实体？你或许有，或许没有。如果你特地将这些事件作为实体来建模，你可以根据数据模型中的**装运计划**或**装运交付**实体推断出装运的状态。若果你不将这些事件建模成实体，你可以使用状态模式捕获这些事件，然后捕获**装运**的状态。

在本章节我们所指的状态是在将来可能会发生的。例如，可能会有个装运计划日期[**shipment scheduled date**]，这个日期是指预计的发货日期，或订单预期支付日期[**order expected payment date**]，此日期指的是你预期收到销售订单付款的日期。关于将这些预期日期作为属性来建模，一些数据建模人做了适当的保留。这些建模人员可能会认为只有当这些状态发生时，才可能捕获他们。换句话说，只有当货运事件发生时，而不是你预计某事发生时，装运才可能有状态。类似的，只有当发票的事件发生时，发票才会有个状态，比如说当你从客户那收到支票（并且已结清）时，且非你预期发生的事情。这种观点可能认为装运计划日期[**shipment scheduled date**]和订单预期支付日期[**order expected payment**]不是状态；他们是装运和发票实体的信息应当将这些作为实体的属性来建模。这种观点的依据是事件发生的预期日期与它确切发生的日期通常是不同的。例如，装运的“发货”日期可能要比装运预计日期要迟些（或早些）。

另一种观点（预期日期实际上是一种状态）时依据下面这个理念的：记录预期日期可以改变状态。例如，当装运的预期计划日期从2010年3月1日变成2010年1月1日时，有些建模人员表示这是个非常重要的装运状态，它改变了装运的‘状态’，因为预期装运的时间更早了。

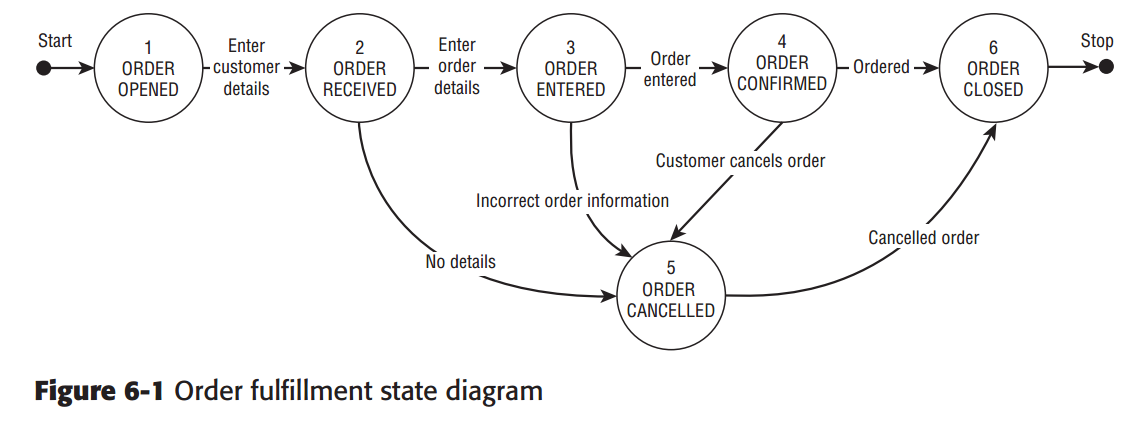
无论你是否赞同预期日期是状态这一观点，在本章节的状态模式中这种观点是同样有效的。如果你同意预期日期不是一种状态，那么你可以使用状态模式只去捕获已发生的状态将“将来日期”作为属性来建模。另外，如果企业想要维护“将来日期”状态，那它可以将这些状态和已发生的状态一起包含到模式中。以我们之见，这些观点都有道理，我们不可以偏向于某一种观点 — 只要在整个建模计划中你使用的是一致的观点。在本章节中，我们决定包含“将来日期”，因为我们想介绍这种模式所能支持的最广泛的状态。

状态模式可以维护以下信息：

* **数据所能容许的状态。**状态模式允许一组属性作为状态类型，比如说名称、描述、有效起始日期、有效终止日期等等。
* **应用于某一实体的状态。**例如，被“录入”的订单。
* **状态的时间信息，如果存在该方面的话。**何时“开立”订单，何时装运何时处于“已发货”状态的？
* **状态类型的分类。**此模式允许将状态分成几组类似的分类，比如说订单可以有一组订单处理状态，还可以有一组计划进度状态。
* **不同类型的状态之间可能存在规则。**例如，可能有规则表明在订单处于“已关闭”状态之前，该订单必须事先是处于“已开立”状态。

## 一级状态模式

以一种非常特别的方式应用一级状态模式为状态建模。这种模式使用实体的“事件”属性维护每个状态，例如，订单接收日期、订单录入日期、订单确认日期等等。随着实体经历一些预先定义的步骤，这些步骤构成了业务流程，状态不被更新，例如，在订单履行流程中，当它经历了整个生命周期，一个订单便可有如下状态：“开立”、“接收”、“录入”、“确认”和“取消”或“已关闭”。图6-1中的状态模型图说明了这一点。基本状态模式必须支持所列出的所有状态，数据可以具有覆盖其整个生命周期的状态。

**图 6-1** 订单履行状态模型图

注：为了说明如何使用这种模式，我们使用了相对比较简单的订单状态的流程模型。我们认识到在一个订单流程中可能有很多其他的状态，他们的流程可能各不相同。例如，可能状态“需要修改订单数据[Order data needs correction]”会取代下一步的“取消订单[Order Cancellation]”，在有些情况下，“取消订单[Order Cancellation]”可能会取代其后面的流程“订单关闭[Order Closed]”而成为最后的状态。

一级状态模式支持前部分所介绍的三个主要特征。

首先，可以将状态记录成潜在的属性。例如，订单可以具有以下状态属性：**已接收订单**[order received]、**已录入订单**[order entered]、**已确认订单**[order confirmation]、**取消订单**[order cancelled]和**订单开立起始日期**[order opened from date]、**订单关闭终止日期**[order closed thru date]。

其二，通过更新属性值，实体处于某种状态会记录下来。例如，当属性订单开立起始日期有值时，可以认为订单处于“开立”状态。

第三，状态可以具有关联的时间或时间范围。例如，订单的“接收”时间是2010年1月4日下午2点，可以在订单接收日期时间属性中维护该日期与时间。

**注：有些事件发生在某一个时刻点，而有些事件则是发生在一个时间段内。例如，订单可以于1月5日上午12:32“接收”的，而同一订单的“开立”时间却是从“”1月5号一直到“1月10号”。这表示有些状态时使用一个单一日期（或日期时间）来维护的，而有些是通过日期范围来维护的。**

**注：有时你可能不需要捕获状态的时间因素。例如，企业决定只要表示出装运已制定“计划[Planned]”而不需要知道制定装运计划的具体日期。**

### 为什么我们需要这种模式？

特定的一级状态模式的目标是明确地维护实体的所有状态，为了创建非常简单、特定和易于理解的模型，此模式使用该实体的“事件”属性来维护状态。这个类型的模型在展示业务代表和主题专家（SME）所定义的状态范围方面是很有效的。如果在订单实体中每个状态都有相应的特定属性来表示，那么抓提庄家可以看到订单可以是开立的、关闭的以及确认过的。此外，如果适用的话，此模式可以捕获状态激活（或预计）的日期（或日期时间）；例如，订单开立的起始日期、订单关闭的终止日期和订单的确认日期。此模式为主题专家提供了事件的清单，从所列的事件中他们可以得知实体的状态以及与状态相关联的重要的日期（或日期和时间）。

### 这种模式是如何工作的？

图6-2 说明的是如何使用属性维护实体的基本状态。实体代表的是相关数据、事务或事件，数据专业人士有兴趣维护实体的状态。例如，这个实体可以是**当事人**[PARTY]、**产品**[PRODUCT]、**贸易**[TRADE]、**预订**[RESERVATION]【注释2】、**订单**[ORDER]、**装运**[SHIPMENT]等等（在很多数据模型中，有很多的维护状态数据的实体）

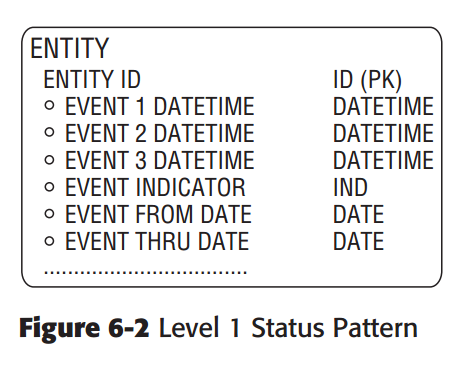


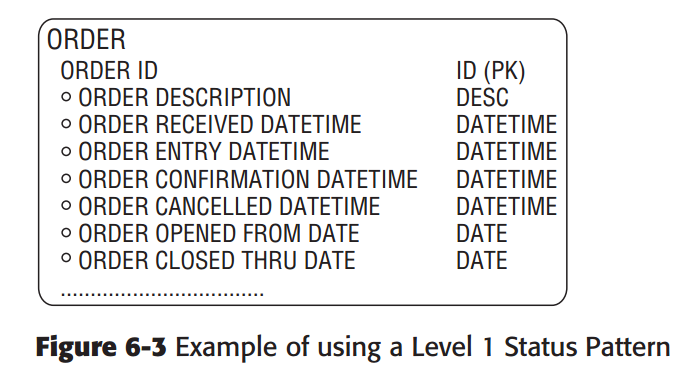
图6-2一级状态模式

由于在实体的上下文中可能会出现一个或多个事件，所以实体可能会有一个或多个属性。例如，订单可能处于“开立”、“确认”或“关闭”等状态。每个状态属性（**事件1的日期时间**[event 1 datetime]、**事件2的日期属性**[event 2 datetime]、**事件的起始日期**[event from date]、**事件的终止日期**[event thru date]）表示事件发生时相应的状态。实体同时可以具有多个属性；例如，**事件1的日期时间**[event 1 datetime]、**事件2的日期时间**[event 2 datetime]和**时间3的日期时间**[event 3 datetime]可能同时都有值。如果一个事件属性有值，你可以认为该事件已发生，如果它是个计划事件的话，可以认为该事件即将发生，比如说**装运预计日期**[shipment scheduled date]。如果时间没有值，则表示该事件没有发生（或是未安排或未预计其发生），没有设置该事件的状态。

在这个模式中，当你给时间1日期时间、事件2日期时间、事件3日期时间等属性添加值时，你就在维护两件事。你记录了事件发生的时间，也就记录了实体的状态（或状况）。我们通常不鼓励数据建模人员让一个属性具有两个用途，因为有些人认为这是“超负荷”属性，是不好的建模方式。然而，在这个状态模式的案例中，这是合理的，因为事件发生的日期暗示了实体的状态。除了事件状态日期属性[event status datetime]，还有创建事件指示属性[event indicator]，此属性通常是冗余的。例如，你可以为订单接收指示建模，用来记录以接收的订单，然后使用另外一个状态属性，订单接收日期时间[order received datetime]，用以记录接收订单的日期时间。订单接收指示属性[order received indicator]没有时间值；仅有一个标志表示某事件已经发生或已经有计划安排了。并不是有两个属性，一个是捕获已存在或即将出现的状态，而一个是捕获该状态出现的时间（例如，状态日期时间属性[status datetime]），我们认为使用同一属性来捕获伴随着时间而出现的实体状态，这种做法会更加有优势，因为添加时间的时间久意味着该事件状态确实已经存在了（或预计即将出现）。

有些状态可能完全没有时间成分。例如，如果装运有个名称为**装运过期指示**[shipment overdue indicator]的属性，你变可以知道装运的状态是“过期”，但是它可能是个不会有‘时间’元素的状态（尽管这得视情况而定，比如这个属性也需要时间元素）。最后，要注意不是所有的属性都会涉及到一个时刻点；而有些状态是发生在一个时间段内的。例如，时间起始日期[event from date]和事件终止日期[event thru date]捕获的是存在于一个日期范围内的状态。在图6-3订单开立起始日期[order opened from date]和订单关闭终结日期[order closed thru date]是状态的示例，这些状态持续了一段时间，例如订单持续一周都处于开立状态，然后当订单履行后才会被关闭。出于这个原因，有些状态时‘时刻点’状态，而有些状态则是‘时间段’状态，通常情况下，‘时间段’状态只关注日期，对日期与时间没兴趣；在此模式中的数据类型是DATE，而非DATETIME。然而，如果需要一定程度的时间粒度，这些状态也可以使用datetime这一数据类型。为状态捕获恰当的‘日期型’数据类型是很重要的，因为在捕获状态时，通常业务需要特定程度的粒度。例如，一个公司可能关心的是在中午之前需要录入大部分订单，或者它想要知道每月订单开立的平均小时数。

注：尽管图6-2所示的模式没有表现出数据类型为DATETIME的‘时刻点’属性以及数据类型为DATE的‘时间段’属性，可分别将DATE 或DATETIME数据类型用于状态，至于使用什么数据类型视具体情况而言。



**图6-3** 使用一级状态模式的范例

**注：有些数据建模人员可能偏向于使用同一事件类型名称来表示‘时间范围’属性。例如，指明订单从某个日期开立到另一个日期终止（包含终止日期），没有使用订单开立起始日期和订单关闭终止日期属性，但是可以使用其他的属性名称：订单开立起始日期、订单开立终止日期。另一种方式是将这些事件看作‘时刻点’状态，每个方式都有效，但是它在很大程度上规定了语义的一致性。**

图6-3进一步阐释了使用这种模式的方法。情境如下：试想一家大型手机制造商想要查看订单所经历的各种状态，随着订单所经历的订单履行工作流不同，其状态也不同。雇佣了数据专业人士检查订单的各种状态，然后向运营领导和IT人士报告订单可能存在的所有状态。

对图6-1所示的订单执行工作流做了详细的分析并与公司的各位专家面谈后，数据专业人士基于一级状态模式制作了图6-3.图6-3中有个订单实体，订单是对交付的货物或服务的一种付款承诺5。在经过一些分析后，数据专业人士将订单所经历的事件表示成订单的属性。当某些事件发生时，便会为订单实例更新状态。

在这种情况下，数据专业人士发现多有关于订单的数据都被当作是“开立”的。这中情况可以发生在下面所属地情况：当销售人员已联系客户并对订购的货物或服务做出了口头承诺时。销售人员在此刻将一些非常基本的数据输入到订单录入系统当中并获得一个订单号。然后客户可以通过电子邮件、传真传送他们货物和/或服务的细节，在这个时候，认为订单“已接收”。所以订单有可能在“接收”之前就“开立”了。例如，在表6-1中订单标识为“12560”的订单实例：“高级手机[Deluxe Mobile Phone]”，这一订单的接收日期时间是“2010年2月2号下午2点”，而该订单开立的起始日期是“2010年2月1日”。换句话说，当该公司意识到一个订单在订单执行工作流中时，订单的开立起始日期是日期（可能是通过某些口头承诺），而订单接收日期时间是日期与时间，该订单的信息是由客户正式发送的。

数据专业人士还发现有个时间戳，是订单“录入”订单录入计算机系统时的时间，并将此时间戳作为订单的属性：订单录入时间日期[the order entry datetime]。在表6-1中，订单标识为“32999”的“手机配件[Mobile Phone Accessories]”订单，其订单录入日期时间是“2010年1月3日下午5点”。数据专业人士发现了三个状态，这表明一个订单存在三个潜在的创建日期。如果高级管理人员想知道一个订单从创建多完成平均花费了多长时间，他们可能会得到三种不同的答案。为了阐释这点，可思考表6-1第三行所示的订单号为“32999”的订单。

销售主题专家认为（SME）应在获得承诺时创建订单，比如说在客户来电许下口头承诺。在订单号为“32999”的订单案例中，订单状态订单接收日期时间[order received datetime]是“2010年1月3日下午3点”。

从数据录入职员的立场来看，订单创建时间是订单录入时间日期[order entry datetime]，也就是“2010年1月3日下午5点”，比销售人员所认为的创建时间迟两个小时。

从物流职员的角度来说，订单开始其生命周期是在订单进入订单执行工作流程的时候（如图6-1所示），订单开始起始日期[order opened from date]是“2010年1月3日”，所以在此案例中，他们根据日期而非小时来衡量生命周期的。

会计人员是在订单状态处于确认的情况下，才登记订单的收益百分比。

订单号为“329999”的“普通手机[mobile Phone Standard]”，此订单的订单确认日期时间[order confirmation datetime]是“2010年1月12日上午4点”。从会计部的角度来说，这个日期是订单生命周期的起点。

从手机公司的角度来说，这个是非常有价值的信息。此模式可以指明不同部门之间存在的误区，表明可以统一企业的重叠状态，或指出公司所想维护的各种有效立场。

**表6-1** 一级状态模式应用范例

在表6-1中，订单标识为“23000”的“普通手机[Mobile Phone Standard]”订单是于“2010年2月4日”取消的。虽然订单是在2010年2月4日取消的，但是如果查看下订单关闭终止日期[order closed thru date]，你会发现订单处于开立状态的最后一天是“2010年2月14号”。这似乎表明订单可以被取消，但同时该订单仍然是开立的。这种情况是很常见的。支持取消订单并将其变成关闭订单，还需要完成很多的事务。在订单号为“23000”此案例中，已登记了该订单的收益，因为该订单已被确认过了，确认时间是“2010年2月2日下午2点”。而流程现在必须反转。必要的反转与流程完成时，订单就被关闭了，订单关闭终止日期[order closed thru date]是“2010年2月14日”。这个范例阐释了一些非常有用的方面。第一，数据专业人士捕获了订单事件，还维护了跨距整个订单生命周期的订单状态。这些状态不局限于公司的某一个特定的专家组，因此，可以捕获了各种观点。关于订单状态的各种观点几乎是集中在一个单一的架构中。最后，此模式表明状态的时间成分可以是时刻点或时间段。

**注：很多数据建模人员不喜欢在属性名称中使用类单词。例如，可以使用event 1来表示event 1 datetime。属性名中不使用类单词是有一些非常合理的缘由的。例如，在属性名称event 1 datetime中个单词表示该属性的数据类型，但是在实体中已经捕获了数据类型datetime，所以这样表示会被认为捕获了冗余信息。如果数据类型变成date会发生什么呢？属性名称将失效。在本书中的属性名称中，我们使用了类单词是有用意的，因为这样做有助于模式解释的阅读，并且通常对于确定属性的性质是很有用的，不用查看数据类型便可确认。这是个常见的数据建模问题；建模人员通常对此两种命名方案中的某一个非常中意，但是这两种观点都有各自的优缺点。我们没有特别的偏向某一方案，只要你能将方案一致的应用于企业。**

### 该在何时使用这种模式？

我们在以下几种情形中使用这个数据模型模式：

* **当有一组定义明确的静态状态，并且预期不需添加其他的状态。根据调查，认为该领域是静态的，不需要添加新的状态类型。**在先前的案例中，指定了特定的状态，比如说订单接收日期时间[order received datetime]、订单录入日期时间[order entry datetime] 、订单确日期时间[order confirmation datetime]、订单取消日期时间[order cancelled datetime]、订单开立起始日期[order opened from date]和订单关闭终止日期[order closed thru date]，可以认为这些状态都是静态的，预计在将来是不会发生变化的。
* **当有很少的状态时。**例如，在某些应用中，实体**人员**只可以有两种状态：“健在[Alive]”或“已故[Deceased]”。这种模式可能适合这种应用，因为它不需要能够容纳很多状态的灵活性（由本章节的其他模式提供），在此应用使用具有这种灵活性的模式，未免大财小用了。另一种选择是对于关键状态的我们使用这种模式。而对于其他状态使用二级、三级或四级模式。例如，你可以选择使用此种模式来维护订单实体中的订单录入时间，而其他状态可以使用更加灵活的模式来维护。
* **如果状态是针对某一特定实体的，你可能希望将此状态作为该实体的属性。**对于将此类实体特定的状态作为属性来维护这样的观点存在较为强烈的争议，因为这样做的话，这个转改不能为数据模型中的其他实体所复用。例如，装运具有装运遗失日期[shipment lost date]的状态。这个状态只用于装运，它不能为其他实体所使用。在这种情况下，甚至可以使用这种模式作为原型的实现。
* **当数据专业人士需要一种简单的方法，用以表达业务数据需求，或是用来作为非技术受众或管理人员的基本作用域声明。**模型图提供了主要的操作和业务代表用来作为订单状态的简单声明或者是用来作为交流验证需求的一种方式。模型图展示**订单**的状态（源自于事件属性）。由于此模式使用的是特定风格的建模方式，所以很容易理解。这种模型介绍了企业中各个小组所使用的术语。可以使用这个模型快速开展状态的讨论。
* **当阐释说明不同立场很重要时。**关于订单实际创建时间，企业不同的组具有不同的观点。在先前的情境中，各相关人士对于重要的订单状态皆有各自的独到理念。

### 此模式的弱点是什么？

缺点如下：

* **这种模式不适合用于动态环境，在动态的环境中，随着时间的推移会出现或需要新的状态类型。**例如，一个新的流程就可能会需要新的状态，比如说“有待信用检查[Pending credit check]”、“主管批准[Approved by supervisor]”或“合法[Accepted by legal]”等状态。那么必须更新实体添加新属性，使用这种模式的话，有可能需要更改模型，更改模型是非常费时费力的。
* **当一个实体具有很多的状态时，那么使用这种模式很难一致的管理这些状态。**假如使用具体风格的建模方式，但是很多时候，订单实体所具有的状态远不止图6-3所示的六种状态，很多属性是日期和指示符。此模式可能不清楚在这些属性中，哪些属性是与状态相关的，而哪些又与状态无关（例如，可以认为“过期指示符”是状态字段，而“优先指示符”则不是）。这种模式不能提供分类状态。如我们先前所说，有一组状态；例如，订单可以有一组订单处理状态（“接收”、“录入”和“确认”），还有一组订单进度状态（“按时”、“延期”和“过期”），并且这种模式没有提供区分状态类型集合的方法。
* **这种模式不维护状态的规则。**这种模式将所有的状态集合在一起，然后关联到订单，但是没有体现这些状态的规则，比如说为了记录其他的状态而需要某些状态。例如，有个规则表明：如果没有为订单录入“接收”状态，该订单是不能有“取消”状态的。这种模式支持这些规则。

### 简介

一级状态模式是非常特定的方法，通过维护“事件”属性来维护状态，这些属性表示了特定实体可能会经历的状态以及可能会出现或预计会出现的状态。在订单案例中，你可以看到订单接收日期时间[order received datetime]、订单录入日期时间[order entered datetime]、订单确认日期时间[order confirmation datetime]、订单取消日期时间[order cancelled datetime]、订单开立起始日期[order opened from date]和订单关闭终止日期[order closed thru date]，各属性的时刻点可能各不相同。指明模式的状态源自哪个日期（日期和时间）事件是很重要的，并且不能冗余的捕获它们。例如，我们不是使用指示符来维护出现事件，然后使用状态日期时间属性来记录事件发生的时间（或预计发生的时间），我们可以使用一个单一属性来维护事件发生或即将发生的时间。

在此模式中，你还可以看到订单开立起始日期[order opened from date]和订单关闭终止日期[order closed thru date]，这两个属性展现了具有时间跨度的属性。一个事件可以发生在一个时刻点，比如说接收订单的日期与时间，而有些时间发生在一个时间段内，比如说一直处于开立状态的订单，随后会在某一时刻点被关闭。因此，有些状态是‘时刻点’，比如说订单接收日期时间[order received datetime]，而有些状态在一段‘时间范围’内都是有效的。还可能某一状态没有时间成分；例如，装运过期标识符[shipment overdue indicator]可以指明装运过期了，但是并没有时间与此属性相关联。

在一个单一的地方维护实体的所有相关状态，企业中的相关当事人可以查看各组对实体状态的不同看法。对于类似的概念这种模式借鉴了企业所使用的不同词汇。本章节这部分的示例，物流部、销售部、数据录入部和会计部等部门根据各自的观点使用了不同的状态来指明订单的生命周期的起点。因此，对于收集和验证相关状态的数据需求，这种模式是非常有效的。

总之，由于这种模式提供了一种开始收集状态需求的初始方式，因此，可以使用它定义针对状态的计划范畴。这种模式易于理解和实现。但是需要注意— 如果环境是静态的，需要更改、添加或删除状态，那么使用这种模式最大的弊端就是需要更改数据模型，一般情况下而言，更改数据模型的代价是很高昂的。

## 二级状态模式，当前模式

我们遇到过一种情形，企业只关注实体的当前状态，并且实体也只有一个状态，因为随着实体参与到生命周期中，一种状态会为另一种状态所替代：我的订单现在的状态是什么？是处于“接收”状态、“开立”状态还是“关闭”状态？我的装运现今的状态是什么？有些时候，企业不关注实体所处的多重状态，也不关注那些历史状态。例如，信用卡清空业务只关注事务的当前状态。企业不需要维护信用卡事务整个生命周期内的各种状态。对于只需要捕获当前状态的情况，我们使用二级状态模式，当前状态。

当数据专业人士希望制作一个特定的当前状态模式解决方案时，这种模式为他们提供了灵活的策略。解决方案的本质是在实体只有一个状态的时候使用这种模式。这表明企业的所有部门将只能看到一个实体状态。例如，处于“开户”状态的账户对于数据录入人员、市场人员和销售人员而言该账户的状态都是“开户”。

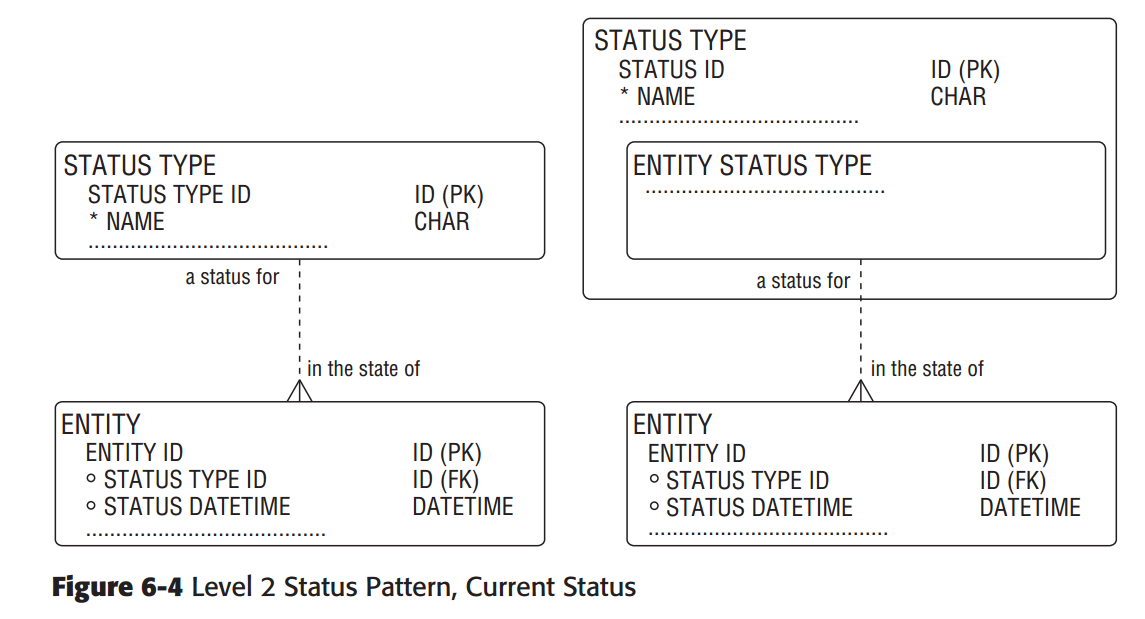
这种模式与前一种模式的不同之处在于它提供了更加灵活的解决方案，因为随着时间的推移，流程发生变化，需要添加状态时，这种模式可以很容易地将状态类型添加为实例。尽管这种模式支持实体所具有的或即将具有的状态类型，但是该实体只能具有一个当前状态。例如，**订单**在某一特定时刻点，只能具有一个状态。

### 为何我们需要这种模式？

企业所需的这种模式可能会随着时间推移需要其他的有效状态；然而，在任何特定的时间点只可以给相关实体指定一种状态。在前一节中，一级状态模式明确将实体的不同状态类型定义成该实体的属性。例如，如图6-3所示订单实体具有订单接收日期时间[order received datetime]、订单录入日期时间[order entry datetime]等属性。如果企业只想捕获当前状态的话，除了当前状态其他的状态属性都将是空白的。这种模式为只捕获当前状态提供了更加优雅的解决方案。此外，如果将来需要新的状态，这种模式可以在不更改模型的情况下满足这一需求。例如，由于其他流程的添加，可能会出此案其他需求，比如说需要 “订单批准”这一状态，以确保在确定订单之前，此订单是可接受的，只需添加一个**状态类型**实例“订单批准[Order Approved]”，此模式便能适应这个需求。

### 这种模式是如何工作的？

图6-4阐释的是添加了**状态类型**实体的模式。可以定义**状态类型**实体为一组状态或分享一些、全部或若干组通用特征的条件事务。数据专业人士可以捕获已知的实体状态类型，当发现新状态类型时，还可以添加新类型。通过将状态类型“概括”到其自身的状态类型实体中，可以做到这一点。正如前部分所述，实体代表数据专业人士感兴趣的一组数据、事务或事件，比如说产品[PRODUCT]、发票[INVOICE]、订购单[PURCHASE ORDER]、支付[PAYMENT]、票[TICKET]以及其他潜在实体。



**图6-4** 二级状态模式，当前状态。

从图6-4中你可以看到该模式的两个变异体。

第一，企业的各种状态类型（例如，当事人状态类型、订单状态类型、发票状态类型等）可以统一到一个单一的实体**状态类型**当中。在同一实体中管理所有的状态类型可以简化模型。此外，当企业中所有实体的各种状态类型具有相同的属性时（尽管在模型中没有展示，但是状态类型可以包含其他属性，比如说描述[description]、简称[short name]、全称[long name]、有效起始日期[effective from date]等等），使用这种模式是很重要的。**状态类型**包含各状态类型的名称，比如说“已开立[Opened]”、“已接收[Received]”等等。

第二，特定的实体状态类型可以作为状态类型实体的子类。使用子类是一种方法，可以将特定实体的状态聚集到其自身的结构中，这样便可以在数据模型中执行因业务规则，一个实体只能具有实体状态子类中的状态（实体状态类型）。例如，建模人员可能想要一个订单状态类型，该实体可以维护订单的特定状态。此外，在有些情形中，各种状态类型的数据是不一样的，也就是，有些类型可能具有不同的属性和/或特定的关系，这种变形适用于会遇到这样情形的企业。

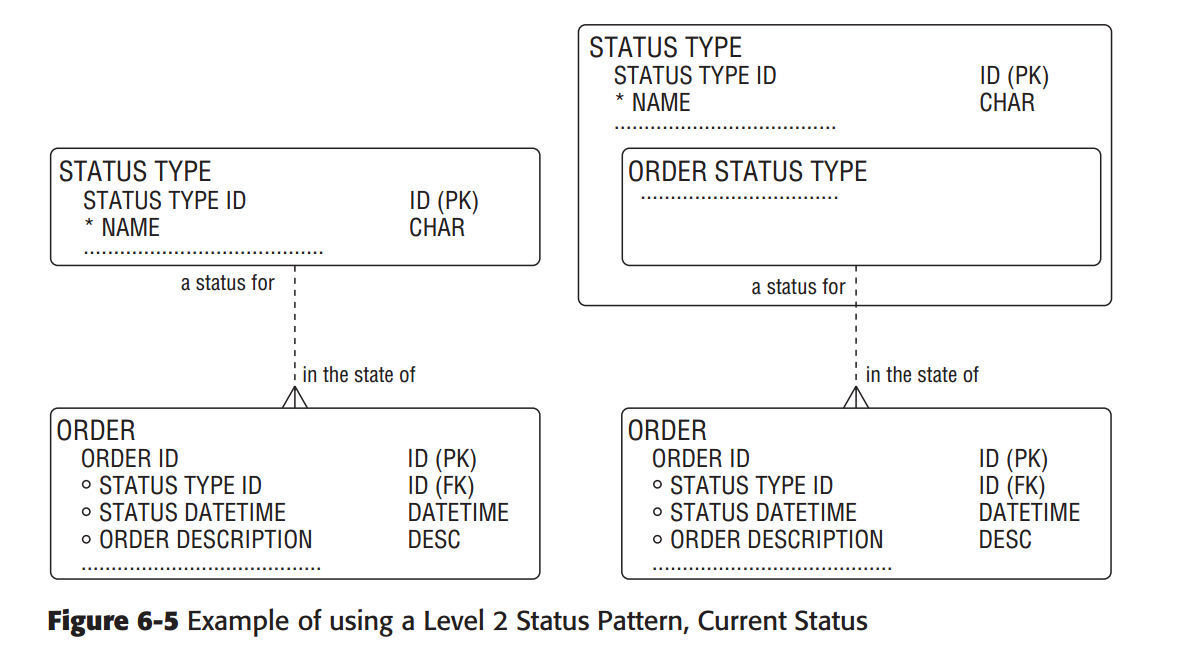
注：在第5章节中，我们有暗示，如果一个实体具有子类，那么通常会有相关的‘类型’或‘类别’实体。在这种模式的第二种变形中，我们具有一个**状态类型**子类，我们可以考虑添加**状态类型与类型**实体，可以使用该实体管理状态类型的分类。例如，我们可能会想对库存的各种状态类型进行分类，比如说针对事务与产品或当事人的状态类型，此外，如果需要更加灵活的方法来提供分类，我们可以使用状态类别模式来提高这种模式，我们稍后将在本章节介绍这种模式。

在大多数情况下，这两种变体的工作方式是相同的。在此模式中，数据专业人士没有像一级状态模式一样，明确地将状态作为属性来维护，而是将不同的状态类型作为**状态类型**实体的实例，**实体**只继承**状态类型**实体的外键以维护该实体的当前状态。当捕获状态时，**实体**的外键属性状态类型标识[status type id]与**状态类型**相关但是已从状**态类型**中迁移了。为了捕获时间成分，展示状态出现的时间，可以在状态日期时间[status datetime]中维护一个时间戳。

应当注意：**实体**可能没有一个特定的状态，这就是为什么**状态类型**（或**实体状态类型**）和**实体**之间的关系是可选的。在业务中，这种情况会经常出现。装运的状态是什么？答案有可能是未知的，也有可能是该状态没有被记录下来。此外，不是所有的状态都会有时间成分。例如，“过期账户”或“待定订单”等状态只能指明实体处于某种状态。这也是状态日期时间属性是可选性的缘由。

图6-5进一步介绍了这种模式的工作原理。假如进一步拓展前一个情形，大型手机制造商接收到了订单，想要捕获定能所能具有的所有可能的状态，但是又只想要维护订单的当前状态。手机制造商希望部分借鉴数据专业人士在处理当前状态时所提供的解决方案，创建一个原型解决方案来处理这些订单。

基于在前一部分所做的分析，数据专业人士将图6-4作为模版在图6-5的左边创建了满足原型需求的模型（在图6-5右边的模型是已被开发的备选模型）。数据专业人士将订单的可能的有效属性都作为**状态类型**的实例。在表6-2中作为**状态类型**实例的各种状态，然后关联到特定的订单，例如，“订单已开立”、“订单已录入”、“订单已取消”和“订单关闭”。这些状态与图6-1中作为属性的状态相对应。



**图6-5** 二级状态模式应用范例，当前状态

当订单通过订单执行流程的各种状态时，让我们来检查下订单的生命周期。在表6-2中订单标识为“12560”的“高级手机[Deluxe Mobile Phones]”在“2010年2月2日下午2点”是处于“开立状态”这个时间是销售人员接到客户电话的时间，在那一刻“订单已开立”是当前状态；然后在同一天的下午3点钟，客户将订单传真过来了，然后订单的状态就变成“订单已接收”，而**订单**状态日期时间[status datetime]则变成“2010年2月2日下午3点”。订单被传到订单处理部门，于“2010年2月2日下午4点”尽数输入订单录入系统。次日销售人员致电给下单客户向她确认订单以及订单的详细信息是否正确。“2010年2月3日上午9点”订单的当前状态在变成“订单已确认”。最后，成功关闭订单，该订单的状态最后在“2010年2月19日下午6点”变成“订单已关闭”。有个有趣地方需要注意，对于订单的时间，状态日期时间不必是连续性的；状态的时间成分只是该状态的有效时间。

为了便于阐释表6-2中的模式，我们展示了“以前的”当前状态。例如，订单“12560”有五个状态，如果在关系型数据库表中实现这个模式，应当/将要使用后面的当前状态来覆盖其中的四种状态。我们对“以前的”当前状态使用了斜体，而最新的当前状态使用了粗体，以区分它们。

**表6-2** 二级状态模式应用范例， 当前状态

**（表）**

在订单标识为“23000”的**订单**实例中，订单生命周期内包含六种状态：“订单已开立[Order Opened]”、“订单已接收[Order Received]”、“订单已录入[Order Entered]”、“订单已确认[Order Confirmed]”、“订单已取消[Order Cancelled]”，最后是“订单已关闭[Order Closed]”。这个也是参照图6-1的。在表的最后一行展示的是订单标识为“32999”的**订单**实例“手机配件[Mobile Phone Accessories]”，该实例的当前状态时“订单已开立”。这是个刚刚开立的订单。

### 应当在什么时候使用这种模式？

一般在以下几种情形中应当使用这种模式：

* **企业只需要捕获实体的一个状态。**这个模式建设一个实体在一个时刻点只能处于一种状态。例如，订单在一个时刻点处于“已接收”状态，然后才越过“已接收”进入“已录入”状态，以此类推。这是不规范的，因为很多时候，需要捕获一个以上的实体状态，但是这种情形也不是很少见，我们曾多次遇到这样的需求。
* **企业不关心实体的历史状态时。**当企业只关注当前状态不想维护实体起先所处的状态（例如，订单的接收时间是2010年2月2日下午3点），而后进入另一种状态（订单的录入时间是2010年2月2日下午3点）。因此，是用这种模式不能获得接收订单和录入订单的平均时间，因为一种状态替换了另一种状态。
* **需要提供任意数量的潜在状态类型时，随着这时间的推移可能会出现这些状态类型。**有可能是发现隐蔽或模糊的需求或新的流程，这些需求或流程需要新的状态，这种模式允许添加新的状态，只仅仅需添加**状态类型**的实例。因此，当新的状态类型出现时，不需要更改数据模型。

### 该模式有哪些缺点？

这种模式的缺点如下：

**这种模式遵循这样的规则：一个实体只能有一个当前状态，之前的状态都会被当前状态所覆盖。**因此，随着时间的推移，会出现一系列的状态，你不仅不能维护这些连续状态，而且使用这种模式实体不能维护在同一时间出现的两个状态（订单处于“开立”状态的同事也处于“已确认”状态）。

**当你只有一个当前状态，比如说“订单已开立”，这样可能会引起企业整体对这个状态具有不同的理解，此模式不能为企业的各组成部分记录各种状态。**“公司已开立”对装运部门可能是一个概念，而对会计部可能又是另一种概念。由于我们只能维护一个单一的状态，所以我们不能表示企业各个成分对此状态的不同见解。

**在生动解释实体可能具有各种状态方面，这种模式不如一级状态模式有效。**将状态作为状态类型的实例，没有特地将状态作为属性。因此，不能在模型图中看到这些状态，不容易为非技术受众所理解。

### 简介

在这个部分，研究了二级状态模式，当前状态，是如何满足某些企业的特殊需求的。

首先，如图6-4和图6-5所示，通过将所有的状态类型概括到状态类型中，此模式捕获了实体的各种状态。

由于是在状态类型实体中维护状态类型，二部是将状态作为特定的属性，所以这种模式适应变化更加容易，因为如果需要先的状态类型，你只需要添加一个**状态类型**的实例，而不需要更改模型。

只捕获实体的当前状态以满足特定的需求，这种模式的重点有些狭隘。然而，一方面，这种模式比一级模式更加灵活，因为它首次在本章节中介绍了状态类型。当发现新的状态类型时，此模式允许你添加新的实例。

由于这种模式的特殊性质，需要捕获实体的各种历史状态的企业，或是一个实体在给定的时间内具有一个以上的当前状态，都不应当使用这种模式。

## 三级状态模式

在当今飞速变化的环境中，出现新的流程、新的业务规则以及新的状态对企业而言是很常见的情形。例如，一个制造商想要获得ISO 9000认证。这将需要企业提升其整体的业务流程以体现其全新的卓越水平。由于改变的业务流程满足新的标准，所以为了遵照新的标准，很可能数据不得不经受不同的状态。

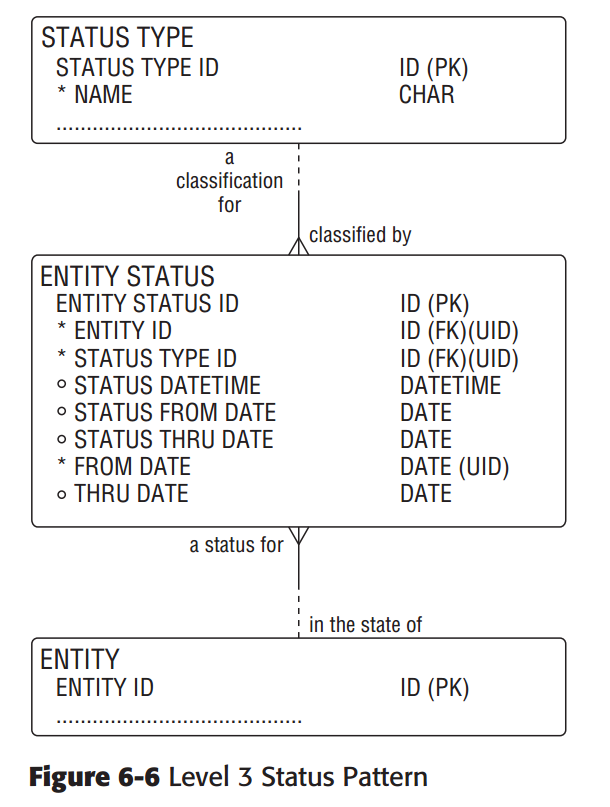
### 为何我们需要这种模式？

在建模工作中灵活性时一个重要的考虑因素。假如一个企业需要捕获所有与某特定领域的业务相关的状态，但是它并不确定他现今捕获了哪些状态。在比如一个企业指导其业务模型将会发生变化，但是并不知道模型会发生怎样的变化。这种情形是和常见的，尤其是哪些非传统企业，比如说在线视频下载企业、在线游戏企业或互联网搜索管理企业。大多数大型企业对其业务流程进行了重大的更改；例如，多数企业为反应新规则需要更改他们的业务流程，比如说在经历2008年的“信贷危机”后，企业引进了“萨班斯-奥克斯利法案”或颁布了新的银行规则。萨班斯-奥克斯利法案的颁布要求企业提出恰当的全新的核算机制和规范的管理流程，这些是会耗费大量的时间、人力和物力的。灵活处理这些状态有助于企业削减新的政府需求所带来的影响。

企业希望捕获实体的历史状态，能够让实体同时具有多个状态，这个模式还解决了企业的以上需求。

### 此模式是如何工作的

图6-6介绍了一种更加灵活的状态模式。这种模式，与二级状态模式，当前状态一样，使用了**状态类型**实体捕获了**实体**可能具有的所有状态。与前一种模式不同的是：这种模式假设实体具有多个状态，并且这个模式也可以捕获**实体**所有的现存状态。**状态类型**和**实体**之间的这种多对多关系，是通过添加**实体状态**实体来解决的。



**图6-6** 三级状态模式

在三级状态模式中，使用了更加灵活的方式来维护各类型的状态，即将这些状态作为**实体状态**的实例。“每个**实体**可以处于一个或多个**实体状态**，而每个**实体状态**只能被一个**状态类型**所分类。”这种模式易于添加新的状态类型，且可以在不更改数据模型的情况下，记录任意数量的状态类型。

实体状态使用了可选性的属性：状态日期时间[status datetime]或状态起始日期[status from date]以及状态终止日期[status thru date]，捕获相关状态的各类时间成分。因此，这种模式考虑到了在某个特定时间点出现的状态，比如说“订单已取消”状态是使用状态日期时间属性来记录的，需要日期范围的状态可以使用状态起始日期和状态终止日期这两个属性来维护（例如，订单从某一日期开立一致持续到另一个日期）。记住，没有时间成分的状态时可能存在的，比如说像“待定[In Abeyance]”或“过期[Overdue]”这样的状态，是不需要指出他们是在何时变成“待定[In Abeyance]”或“过期[Overdue]”的。

**注：在此模式中，对于与时间相关的属性，可以有多种选择方案。例如，模型可以包含状态日期属性，如果需要日期和时间的话，可以选择状态日期时间属性。对于‘时间范围’状态属性，模式可以使用状态起始日期或状态起始日期时间属性（同样适用于状态终止日期），使用哪个属性，视你的需求而定。这也适用于起始日期和终止日期属性。**

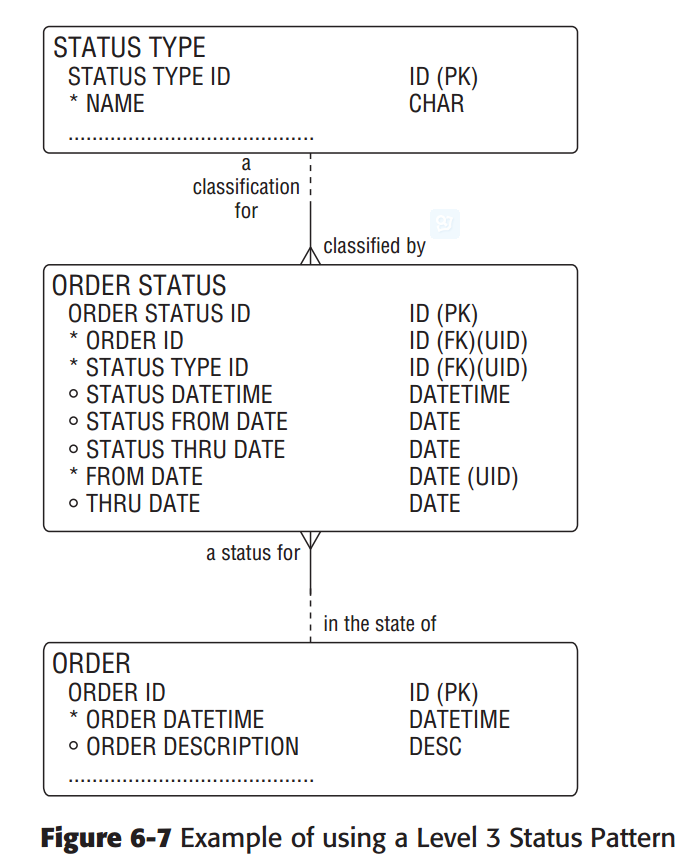
与本书的其他关联实体相似，这种模式包含了起始日期[from date]和终止日期[thru date]属性，以捕获实体状态实例是从何时开始有效的，直到何时无效。与状态日期时间[status datetime]、状态起始日期[status from date]和状态终止日期[status thru date]这些属性相比，起始日期和终止日期属性具有不同的意义。起始日期和终止日期属性捕获的是制定或设置状态的时间和状态有效性的截止日期，而‘状态日期’属性捕获的是状态自身的信息，例如，如果我们在**装运状态**实体中记录预计装运指定日期为“2010年1月5如下午两点”，那么就会为装运状态指定状态类型为“已指定[scheduled]”，该装运状态的状态日期时间是“2010年1月5日下午2点”。然而，在发现这个示例的时候，制定或设置该实例的时间，可能是在2010年1月1日，因而实例的起始日期[from date]是“2010年1月1日”，并且此示例会一直有效直至终止日期[thru date]。如果在1月4日，预计装运日期变成了“2010年1月7日下午4点”，那么第一个实例的终止日期[thru date]的记录将是“2010年1月4日”，一个新的实例将会被创建，该实例的起始日期[]from data是“2010年1月4日”，而状态日期时间[status datetime]是“2010年1月7日下午4点”，代表了新的预计装运日期。

起始日期[from date]和终止日期[thru date]也可用于没有状态时间成分的实体状态实例，比如说需要在状态类型中记录装运过期[shipment overdue]，而该状态没有任何相关的日期，比如指明状态过期了多长时间。起始日期还记录了装运首次过期的日期起始日期将记录装运不在被标为过期的时间。与其他的关联实例相似，起始日期属性（或起始日期时间，如果需要的话）是唯一标识（UID）的一部分。状态日期时间[status datetime]（或状态日期[status date]）不是唯一标识的组成成分，因为状态可能没有时间成分，比如说“过期”标识符。

这部分将继续使用大型手机制造商的案例。假如在数据专业人士向IT指导委员会展现了前一种模式，有人提出了新的需求。指导委员会的监察部人员提出正在制定的新政策需要订单，直到确认了客户信用度后才可以持有一定数额的资金。监察队知道需要添加其他的流程，但是并没有完成那些流程。数据专业人士被要求创建一种模型，该模型可以支持现已知的状态以及新的，未知的状态；这样的模型将会削减变化所带来的影响。

以三级状态模式作为模版以及与主要职员面谈后，数据张野人士制作了图6-7，此图表示“每个**订单**可以处于一种或多种**订单状态**”**订单状态**实体交叉引用了**订单**与**状态类型**。因此，可以通过**状态类型**的外键状态类型标识查看某一特定状态的名称。状态类型的名称属性维护的是状态类型标识的值，比如说“订单已开立”和“订单已关闭”。所以，此案例中，监察部领导表示需要添加新的状态以展示一个订单是否处于信用复核阶段，可将此状态添加为**状态类型**的实例，其名称为“信用控制[Credit Hold]”，特定的**订单**可能会有很多的状态类型与之相关联。

表6-3进一步阐释了此模式的工作方式。标识为“12560”的“高级手机[Deluxe mobile phone]”在经历订单执行的流程时，此订单有六种状态。在2010年2月1日下午2点，有一个源自客户的电话通知，客户表示想要下特定的订单，欲订购大量高级手机。因此，标识为“12560”的订单具有一个订单状态，其状态日期时间是“2010年2月1日下午2点”，状态类型是“订单已开立”。然后订单的接收和录入时间是一样的（“2010年2月2日下午2点”）；由于订单是通过互联网提交的，所以这种状况是可能存在的。因此，此订单有三个订单状态实例：“订单已开立”、“订单已接收”和“订单已录入”，这三种状态的状态日期时间一样，都是“2010年2月2日下午2点”。与二级模式的情况不同，一个状态不能替换另一个状态，而是他们可以同时存在。在“2010年2月3日”，订单被信贷部门标为“信用控制”。此订单的值足够大，以至于需要接受信用检查。这是监察部相关人员在IT知道委员会会议上所讨论的新的流程。订单的“信用控制”这一状态类型是从2月3日开始（状态起始日期[status from date]为“2010年2月3日”）一直持续到2月11日（其状态的终止日期[status thru date]为“2010年2月11日”）。“信用控制”状态类型是个范围状态，展现了订单接受信用检查的整个时间跨度。最后，在“2010年2月12日上午9点”，订单“12560”被关闭了，一个新的状态被设置，“订单已关闭”。



**图6-7** 三级状态模式应用示例

这里阐释了这种模式的灵活性和力量。当流程、规则、规章和企业发生变化时，以及这种模式的数据模型能够适应这些变化，并不需要更改模型。可以出现任意数量的新的状态类型，比如说“需要纠正的订单数据[order data needs corrections]”、“等待法定审查[order awaiting legal review]”、“订单投诉问题[order complaint issued]”和“订单已履行[order fulfilled]”等等。这种类型的数据模型结构可以适应接受企业很多变化的状态。

### 应当在何时使用这种模式？

我们在以下情形中使用这种数据模型模式：

* **需要灵活的方案时。**此模式提供了很大的灵活性，可以将任意数目的潜在状态添加为状态类型，在同一时间或随着时间的推移，实体可以有任意数目的状态，实体的同一个状态可以出现多次（例如，可以开立、取消、再次开立、和再次取消订单），这种模式可以记录：没有时间成分的状态、具有状态日期（和时间爱你）或状态日期范畴的状态，后两者可以通过状态起始日期[status from date]和状态终止日期[status through date]来维护。我们已为多数企业实现了这种模式，依据我们的经验，这种模式适用于很多的场景，因为即使随着时间的推移，需要更多的新的状态类型，此模式可以适应这些需求。
* **当没有明确定义研究中主题领域的状态时，或是对该领域的状态了解甚少时。**有些情形中，你并不清楚需要什么类型的状态，甚至是就眼前的情形来说，你也可能不清楚。例如，在本节的案例中，监察部相关人士需要一个新状态；这样的要求是很容易做到的，只需添加状态类型的实例“信用控制”。
* **当企业想要整合所有的状态，以使状态类型的管理变得更加容易时。**例如，**状态类型**实体可以捕获企业所有的状态，然后通过相关的**实体状态**实体将状态类型实体附属到相关实体。 这样可以使状态类型的管理变得更加容易，因为能够以同类型的结构来使用状态类型、对他们进行分类以及维护他们。

### 这种模式有哪些缺点？

此模式的弱点如下：

* **此模型不能执行特定的业务规则。**无论这样做是否有意义，这种模式可以让实体具有任意数量的同类型或不同类型的状态。例如，订单可以有很多的“订单已确认”状态，即使可能有业务规则规定事实上，订单只能确认一次。处理这种情况的一个方法是指定特定的规则，不是让数据模型自身执行这些规则，而是将这些规则作为数据模型的附属。或者是使用本书第8章节所介绍的业务规则模式。
* **当特定类型的状态具有特定的属性和/或关系时，这种模型不能适应这样的需求。**此模式通过同样的属性和关系在**状态类型**实体中维护了所有的状态类型。然而，如果有特定类型的状态需要特定的属性和/或关系，那么就可能需要对模式稍作修改，为状态类型添加子类，这样的做法与图6-5的右边所示的方式极为相似。
* **这种模式较为抽象，更加难以理解，因此，用于收集或验证需求，不如一级状态模式有效。**这种模式没有像一级状态模式那样，明确地展示出实体可以拥有的不同状态。由于各种状态类型被用来作为状态类型的实例，所以在向非技术受众阐释状态的需求范围时，使用这种模式不是很有效。

### 简介

这种模式是很重要的，因为它适用于常见的情形，在该情形中，实体在同一时间或随着时间的推移可以具有很多的状态。三级状态模式通过维护实体**状态类型**所示的各种允许状态之间的关系解决了这个问题。通过关联实体**实体状态**来解决这种多对多关系，实体状态维护了实体的当前状态和历史状态。在示例中，订单状态可以维护状态的日期和时间，该订单处于什么状态，可以使用外键引用其关联的**状态类型**及查询名称。

状态类型中的大多数容许状态要么发生在某一时刻点要么发生在一个日期范围内。如表6-3所示的订单“12560”的“订单已接收”和“订单已开立”状态。

大多数企业希望使用一致灵活的模型处理状态，此种模式可以为他们所用。当发现新的状态类型时，可以将这些状态类型添加到状态类型中，而不需要更改底层的数据模型。此外，每个实体可以具有任意数目的状态，甚至可以多次记录在不同时刻点出现的同一状态。然而，这种模式不执行特定的业务规则。例如，可能有规则规定状态应当只出现一次（例如，“未通过信用检查[Failed Credit Check]”）。尽管各种状态类型需要不同的属性，但是这种模式还是假设所有的状态类型具有相同的属性，**状态类型**可以具有子类，例如，*产品状态类型*、**订单状态类型**、**装运状态类型**等等。

## 四级状态模式

四级状态模式与本章节先前介绍的模式相比，要更加灵活。我们认为它是个“即插即用”型的模式。我们的意思是：使用这种模式，任何需要维护状态的实体只需要将其自身插入到此结构中，不需要添加任何新实体或属性（除了外键）。当新实体需要捕获状态时，只需要创建**实体**到**状态应用**的关系和**状态应用**的外键。

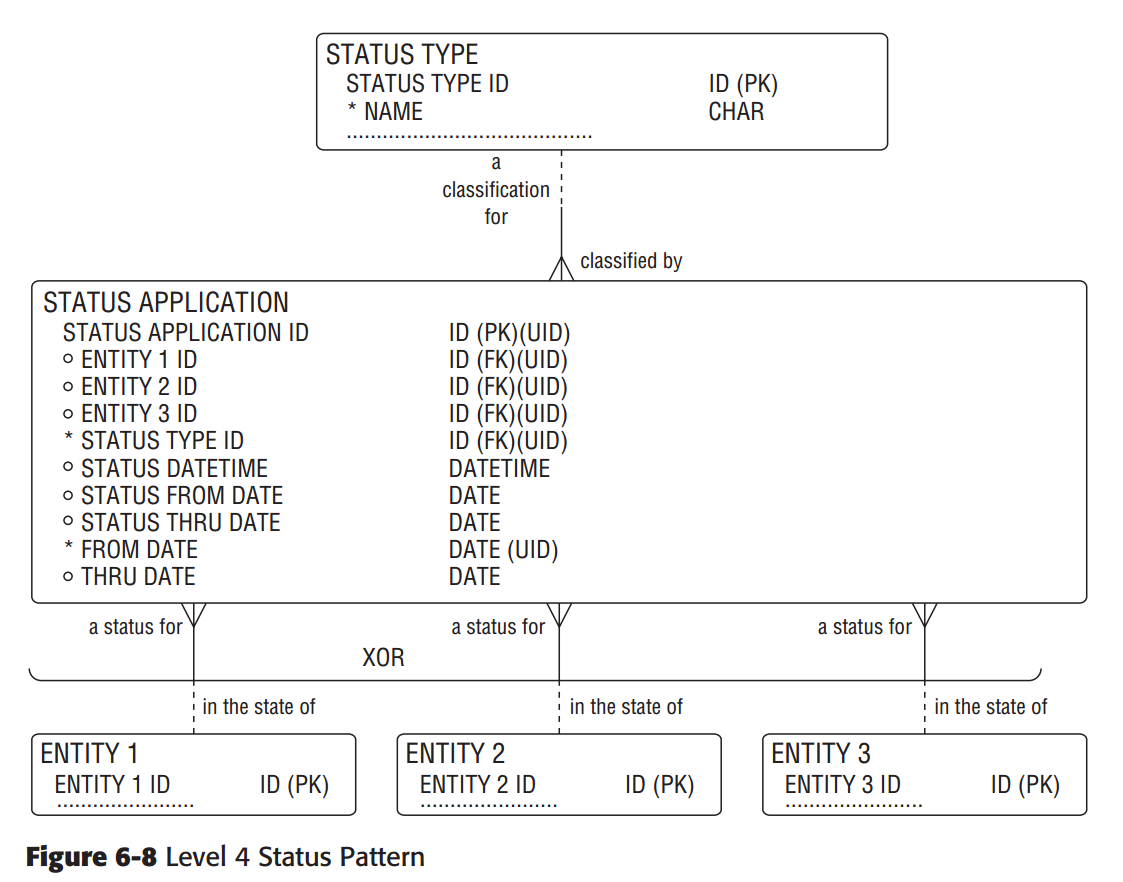
### 为什么我们需要这种模式？

这种模式与三级模式十分相似，但是在某一重要的方面，它是不同的：此种模式不是将单个的**实体状态**关联到状态类型，而是使用一个称为**状态应用**的单一的统一实体，该实体支持状态类型实体的所有关系。这个实体几乎相当于编程中的接口。换句话说，任何新的或现存的实体需要维护状态信息的话，你需要做的只是通过创建关系来关联接口状态应用，这种模式非常适用于以下企业：具有动态的数据环境，想要使用标准的、模块化的方法维护状态数据。

### 这种模式是如何工作的？

图6-8 说明了这种模式。**实体1**、**实体2**和**实体3**是具有状态的实体。这些实体中的每个实体都以相同的方式与状态应用连接。这些实体间的关系表示**实体1**（或**实体2**或**实体3**）可以处于一个或多个状态应用，而一个或多个**状态应用**必须只能是一个**实体1**（或**实体2** 或**实体3**）的状态，在图中，“异或（XOR）”横跨了状态应用到每个实体关系。这表示状态应用必须只能代表一个实体的状态，而该实体可以是实体1、实体2或实体3。换句话说，一个状态应用实例不能够同时应用于多个实体，并且是必须只能应用于一个实体。

“每个**状态类型**可以是一个或多个状态应用的分类，而每个**状态应用**必须只能为一个**状态类型**所分类。”状态应用与所有实体之间的关系规范化了状态应用到状态类型的多对多关系。正如模型图所示，状态应用具有三个源自实体（实体1标识、实体2标识、实体3标识）的非强制性外键。对于状态应用的每个实例，这些外键中的某个外键可能会有值。



**图6-8** 四级状态模式

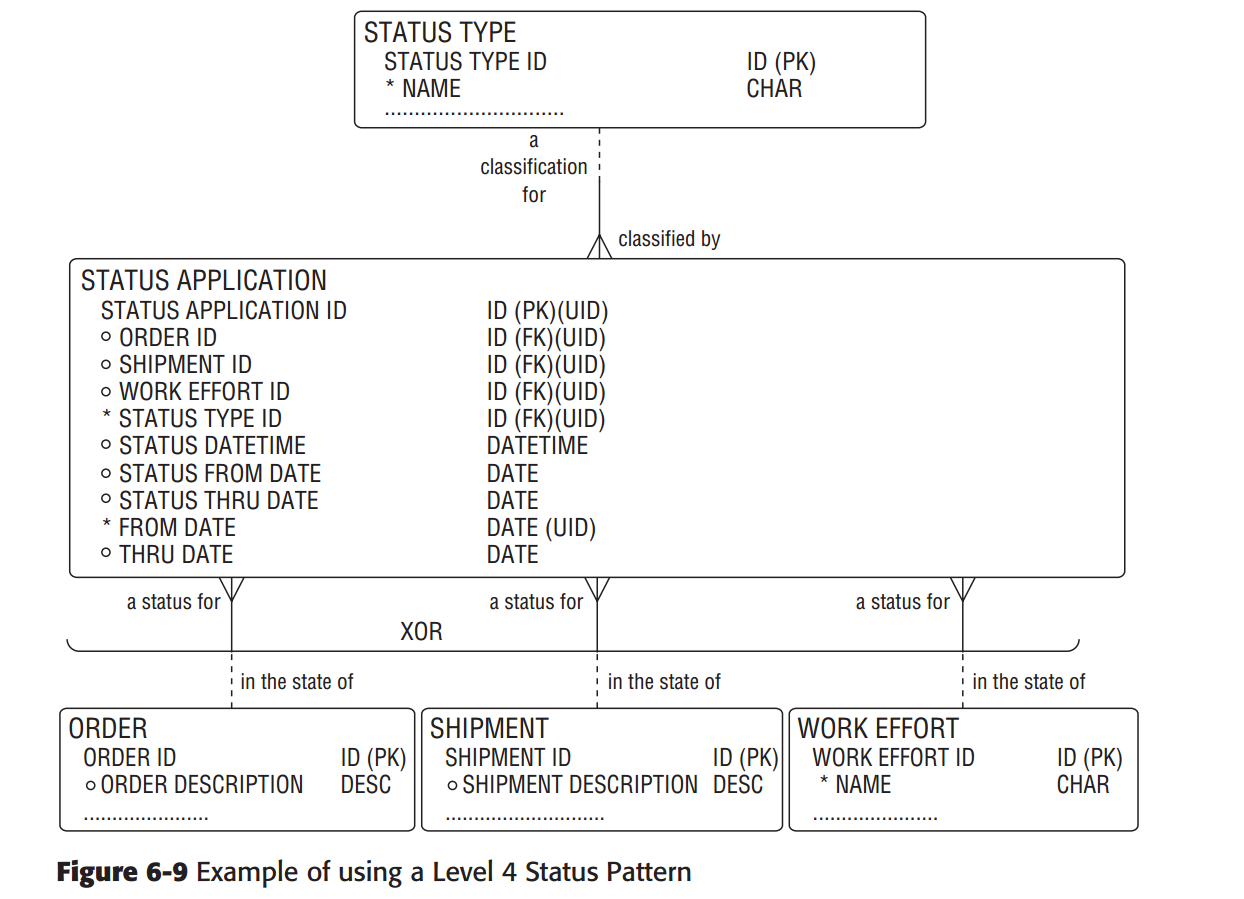
为什么我们使用这话模式呢？问题的答案在于大多数状态具有一组非常常见的属性，也就是，**状态日期时间**或**状态起始日期**/**状态终止日期**和**起始日期**/**终止日期**，就连通过关联实体所建立的实体到状态类型的关系也几乎是相同的。这种模式还假设状态类型实体，比如说**订单状态类型**、**装运状态类型**和**工作计划状态类型**都具有相同的属性，如果确实是这种情况，那为何不将他们整合起来呢！

**注：如果不同的状态类型具有不同的属性，可以让状态类型具有子类，比如说工作计划状态类型和订单状态类型。**

通过创建这种接口实体，状态应用，数据专业人士简化了任务：确保每个实体是以同样的方式与状态类型相关联的。不需要为每个需要访问状态类型的新**实体**创建新的**实体状态**实体，如图6-6和图6-7所示。通过关联到**状态应用**，**实体1**、**实体2**和**实体3**可以访问通用的关联实体，该实体提供了灵活性和一致性，并且还定义了完整的状态类型集。

最后，如果数据模型/数据结构中的应用状态或状态类型发生变化，所有与之相关的实体会立即意识到这种变化，而不需要更新每个**实体状态**实体，以确保数据的一致性。

图6-9 进一步说明了这种模式是如何工作的。继续前面提到的场景，手机制造商希望创建一致的方法，处理企业中的所有状态，这种方法符合其主数据策略。数据专业人士根据四级状态模式创建了图6-9。



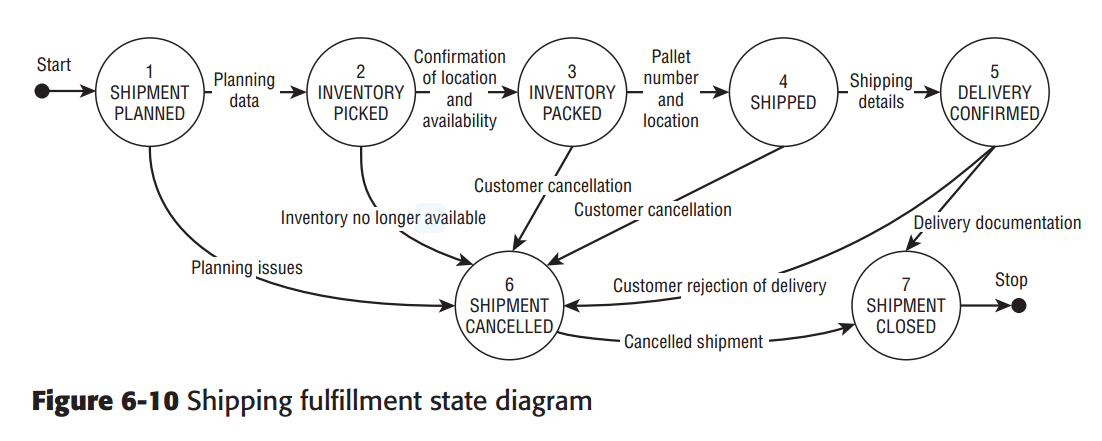
**图 6-9** 使用四级状态模式的示例

如图6-9所示，数据专业人士采用了三个实体，并将它们整合到四级状态模式中：**订单**、**装运**和**工作计划**。订单在先前的模式中已经出现过了。如果你查看下表6-4，它的表现与三级状态模式中的表现完全一致，唯一不同的是此模式没有使用**订单状态**的状态日期时间、状态起始日期和状态终止日期，而是使用了**状态应用**的状态日期时间、状态起始日期和状态终止日期。例如，标识为“12560”“高级手机[Deluxe Mobile Phone]”订单，该订单的状态类型的名称是“订单已开立[Order Opened]”，其状态应用的状态日期时间是“2010年2月2号下午2点”，状态类型名称为“订单已关闭[Order Closed]”的状态应用状态日期时间是“2010年2月12号上午9点”。

**（表）**

**表6-4** 四级状态模式应用范例，订单

现在我们来看一看**装运**的范例。装运是指“将材料、货物或其他物品从一个地点运送到另一个地点的运动，目的地和目标地点可能属于同一个当事人，也可能源自不同的当事人”。8如图6-10所示，每个**装运**所经历的完整生命周期包括：提货、打包和运输。图中的每个圆圈代表装运的一种状态。可以将这些状态转化成**状态类型**的实例。



**图6-10** 装运实现的状态模型图

表6-5中展现的是先前介绍的订单的装运。有个有意思的地方需要注意：这张表的结构与表6-4的结构相同。**装运**状态的处理方法以**订单**状态的处理方法一样。装运状态所使用的值与订单的不同，分别是“装运计划[Shipment Planned]”、“库存提货[Inventory Picked]”、“库存打包[Inventory Packed]”、“已发货[Shipped]”、“交付已确认[Delivery Confirmed]”、“装运已取消[Shipment Cancelled]”以及“装运已关闭[Shipment Closed]”，但是他们所采用的方法与订单的所使用的方法一样，使用状态类型来维护这些状态，再通过**状态应用**实体将他们应用至**装运**。

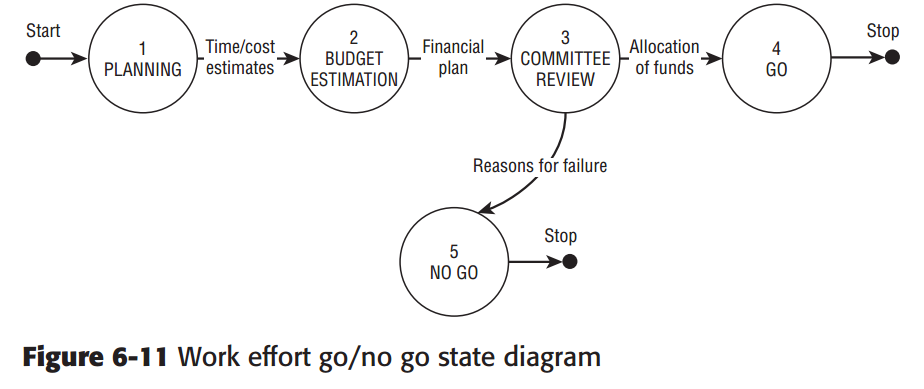
表6-5中的装运恰好是表6-4中的订单“12560”的装运。此装运实例具有装运标识“32”，装运名称为“伦敦高价手机和电话仓库[Deluxe Mobile Phones to Telephone Warehouse in London]”。你可以看出**装运**在“2010年2月3日上午9点”的状态是“装运计划[Shipment Planned]”。在订单确认环节检查订单的有效性；现在库存分派至订单的是已提货（将必需品从仓库中提取出来），订单于“2010年2月4日上午9点”所处的状态时“库存提取[Inventory Picked]”。将出货放入板条箱，而该装运的状态在“2010年2月6号下午3点钟”变成“库存打包[Inventory Packed]”。这表示出货已被包装成板条箱并且已放入货运集装箱。装运于“2010年2月6号下午8点钟”离开仓库，装运在该时刻获得的状态时“已出货”。它抵达客户仓库的时间是“2010年2月8日下午9点”，由客户接收部门电子签收。在该时刻点装运的状态被设置成“交付确认[Delivery Confirmed]”。稍后在当天下午5点，在交付人员确认这次装运所有工作都已完成时，该装运被关闭，其状态变为“装运已关闭”。

（表）

表6-5 四级状态模式应用范例，装运需要注意的是装运通常是个对时间非常敏感的流程，因为当事人一直在等着他们的交付。因此，在此流程中的每个状态都具有反应这种敏感度性的时间成分。此外，装运并没有这样的装运实例：通过状态起始时间[status from date]属性和状态终止时间[status thru date]属性来表示出状态的‘日期范围’。装运的生命周期可以根据“装运计划[Shipment Planned]”与“装运已关闭[Shipment Closed]”的时间差计算得出。数据专业人士想要提议的一种可能性是状态类型，例如，“装运开立”，可以使用状态起始日期[status from date]和状态终止日期[status thru date]之间的时间范围来表示其生命周期。

最后，我们来看一下另外一个数据实体，工作计划。工作计划可以定义为“执行了已计划、正在进行中或已完成的工作活动，它可能与工作需求的执行相关”。9订单可以通过装运（通常针对产品或产品零件）或者履行工作（通常是针对服务）来实现。这个实体的处理方式与之前的订单实体和装运实体的处理方式完全一致。尽管装运的状态集（即,“计划中[Planning]”、“概算[Budget Estimation]”、“委员会审查[Committee Review]”、“未通过[No Go]”、“通过[Go]”）是不同的，但是表6-1的结构还是与表6-5和表6-4的结构相同。

图6-11描述了工作计划的“通过”/“未通过”流程。此流程说明了当提出了工作计划时，企业是如何根据计划中、概算和委员会审查这一些列环节来决定是否应当继续该工作计划。在通过/未通过流程中的每个阶段中，每个圆圈表示工作计划的一个状态。



**图6-11** 工作计划通过/未通过状态模型图

从表6-6中你可以看到“数据仓库项目[Data Warehouse Project]”的提议，该工作计划标识[work effort id]是“904312”。“在2010年9月3日”，开始“策划[Planning]”此工作计划。需要注意的是这个示例只展现了日期而没体现时间，因此与**状态应用**中的状态日期时间[status datetime]的完整格式不符合。对于这种特殊的工作计划，可以使其不需要时间成分；然而，数据模型适应这个需求，只需要将具体的时间省去即可。因此，根据你的惯例，日期时间属性可以表示时间成分是可选的。约一个月之后，根据计划，与2010年10月4日评估该项目提案的成本。工作计划“数据仓库项目计划[Data Warehouse Project]”在该时刻取得了“概算”这一状态。然后有监察委员会 审查该项目的预算和计划。该工作计划于“2010年11月4日”处于“委员会审查”状态。不幸的是委员会不同意该概算并将工作计划取消的，因此，给其提供了一个“未通过”状态。“未通过”状态完全没有时间成分。它表示了该项目被当成是以取消的项目，当团队可以录入“未通过”决定的日期时，他们决定不记录此状态的任何时间。

**（表）**

**表6-6** 四级状态模式应用范例，工作计划

有意思的是维护不同状态的数据集，根据四级状态模式，可以使用相同的方法来处理。这种模式可用于替代很多其他状态关联实体（比如说**当事人状态类型**、**当事人关系状态**、**产品状态类型**、**库存项状态**、**账户状态**、**交流事件状态**、**发票状态类型**、**支付状态类型**以及其视情况而定的其他可应用的状态类型）。

### 应当在何时使用这种模式？

我们在以下几种情形中使用这个数据模型模式：

**当企业承诺创建非常灵活的，更容易适应变化的数据模型时。**使用此类模型很重要的一点是企业必须要充分理解这种一致的“即插即用”模式的价值，并且为了创建非常灵活一致的数据模型策略，企业愿意致力于更加泛化的模型（包括获得认可）。

**企业想巩固通用的实体、属性和关系，将其作为数据模型策略的一部分，以使管理这些对象更加容易些。**越来越多的企业做出有意识地决定使用这个方法建模，当新类型的实体需要记录状态数据时或当信息需要发生变更时，这种方法可以将添加新实体和属性的几率降到最小。

当企业致力于创建一种一致的方式来管理数据时，尤其是状态信息数据，例如，致力于数据驱动型数据管理策略的企业。这种模式的使用也有利于一致性的软件、数据架构和应用架构的创建。此模式提供了一个非常强大的战略性的优点，就是可以节省大量的金钱和时间。

### 这种模式的缺点是什么？

此模式的缺点如下：

* **由于这种模式更加的灵活，所以它也更加“繁琐(rope)【注释3】”且缺乏业务规则。**例如，当你给四级状态模式附属一个新实体时，无论该实体是否与当前实体相关，你都可以得到每个可用的状态类型，因此，这种模式并没有限制应用至该实体的状态的选择。这表示通常情况下，没有“信用限制”属性的实体，比如说**工作计划**， 可以访问该状态。为了给这种模式添加约束，可以给模型添加一些业务规则：约定一个实体可以使用哪些状态类型。我们认为这是个缺点，因为大多数实体并不是使用所有的状态类型，而状态类型可以是针对某一特定实体、业务流程或工作流的。
* **不同类型的状态实际上可能有不同的信息需求。**例如，或许有个只需要日期时间状态属性的实体。还有实体只需要当前状态，而无需维护过去的状态。所以这种模式需要每个实体在其数据架构中能够使用所有该模式的潜在属性，无论需要这些属性与否。
* **以非常概括的方式建模提供了更多的灵活性，但是这是以减少清晰度为代价的。**非数据专业人士一般很难理解这种模式，什么样的数据被捕获了以及在哪里捕获的。尽管这类模型因其概括性而丢失了可理解性，但是如果你致力于为每个状态集制作和维护状态装转换图（或有限状态机或数据流和状态转化的一些其他说明），这对于模式的理解是很有帮助的。
* **如果没有正确的实现这种模式，那么在两个实体中维护所有的状态可能会导致性能问题。**在两个统一实体：状态应用和状态类型中维护所有的状态信息是一个高度概括的模型，当这种模型被实现时，可能会对性能产生影响，因为搜寻可用的状态，在那时会被所需要的特定类型的状态给过滤掉（例如，订单状态、装运状态等等）。尽管性能方面的考虑并不真正是数据建模的一部分，但是有人认为它是处于查看下行流的考虑，这样做的目的是确保数据模型现实中是可实现的。如果以强健的物理架构实现了这种模式，应当不存在性能方面的问题。

### 简介

在这部分我们介绍了四级状态模式，这是中更加灵活的“即插即用”的模式，使用这种模式的企业可以自动关联需要状态的新实体，以构建非常完整的数据模型架构。

这种模式使用了“接口”实体— 像实体状态应用，任何新的或现存的实体都可以与该实体相关联。状态类型已然支持实体所需的各种状态，所以通过将其关联至**状态应用**，任何实体都可以访问所有的状态集以及全面的数据架构，全面的考虑是处于初期和将来的需要。

这种方法有很多的优点。它有助于企业为所有的状态类型提供统一一致的方法，不仅为状态的处理提供了高度一致的数据模型，而且开发与此模式同步的一致流程和架构，也是企业受益匪浅。对于怎样处理需要状态信息的新实体，它采取的是猜测决定。它是一种非常灵活稳定的模式，能够经受新状态类型和新需求的增添。综合的状态功能可以自动获取任何实体所需的状态信息。由于此模式的实体和属性较少且是以一致的方式维护的，因此简化了数据模型（数据库设计）的管理。如果企业决定更改其状态管理方式并且这个变更会影响数据模型（比如说总是指明所有的状态需要日期和时间而不是仅需要日期），可以在一个地方更改该模型。此模式的不足之处在于此模型几乎不执行业务规则，它没有为不同的实体维护不同的状态数据需求，并且可能会对下游数据库的性能造成影响。这种模式要求企业致力于灵活数据架构的投资。

另外一个缺陷就是缺乏易懂性。如果企业使用这样的模型，我们建议该企业坚定不移的补充数据模型的其他文档，比如说状态转化图、示例图、和/或实践范例。此外，此模式不能为不同的状态类型维护不同的属性、关系和/或规则。

## 状态类别模式

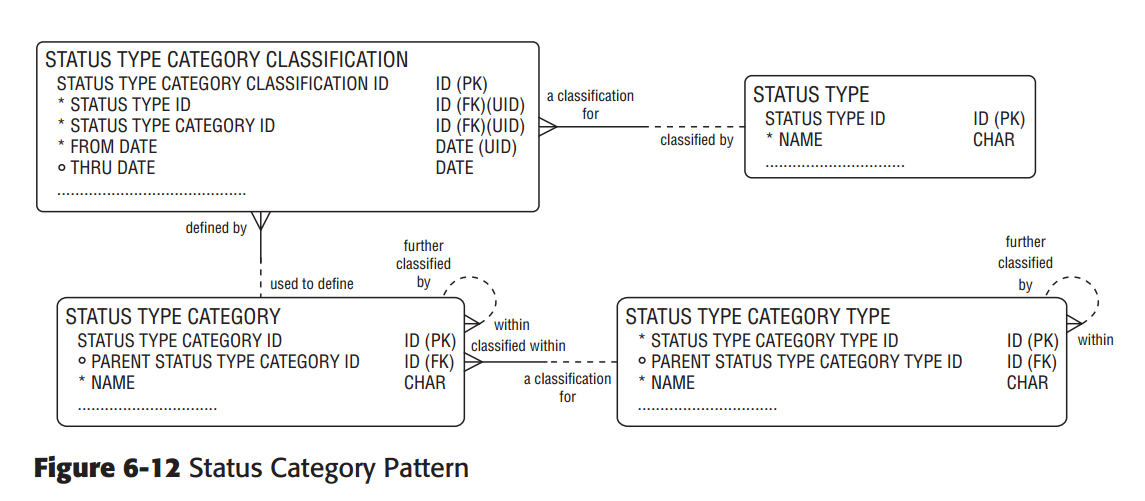
在前面部分，我们引用了不同的状态模型图，展示了订单（图6-1）、装运（图6-10）和工作计划（体6-11）的基本状态，这些模型图中的每一个都阐释了一个流程，在该流程中不同的状态彼此之间相互关联。状体之间不仅仅是相互关联，而且他们还是有限集合的成员。换句话说，每个状态可以直接与一个或多个其他的状态相关联，但是多有的状态都在某个状态组合（聚合或分层）中。例如，“订单已接收[Order Received]”、“订单已录入[Order Entered]”、“订单已确认[Order Confirmed]”、“订单已取消[Order Cancelled]”和“订单已关闭[Order Closed]”全都是订单执行流程中的成员。然而，除了订单执行流程外订单还可能具有其他的流程。例如，订单可能有订单规划流程，该流程有以下状态类型：“按时完成[on Schedule]”、“延期[Behind Schedule]”和“过期[Overdue]”。事实上，由于订单可能有很多不同的流程支持他们，所以可以有不同的状态类型集合或类别。状态类别模式捕获了这些类别或其他类型的状态类别。12

### 为什么我们需要这种模式？

通常我们会发现有多个状态类型集合，所以将状态类型分类到合适的集合是很重要的。这种模式支持了这一需求，让一个状态类型可以应用一个或多个状态分类集合。

### 这种模式是如何工作的？

图6-12展示了我们是如何创建状态分类的。这种模式应用了第5章节中的3级分类模式，可以用来增强2级、3级或4级状态模式。因此，通过这样的补充，任何数据类型都可以具有任意多种分类方法。该图说明了“每个**状态类型**[STATUS TYPE]可以为一个或多个**状态类型类别分类**[STATUS TYPE CATEGORY CLASSIFICATION(s)]实体所分类，而每个**状态类型类别分类**[STATUS TYPE CATEGORY CLASSIFICATION(s)]只能由一个**状态类型分类**[STATUS TYPE CATEGORY]来定义。”例如，装运的状态类型有“装运已计划[Shipment Planned]”、“库存已提货[Inventory Picked]”等等。这些被分类的状态类型可以作为“装运执行[Shipping Fulfillment]”状态类型类别[STATUS TYPECATEGORY]的成员。装运还有其他的**状态类型**[STATUS TYPE(s)]：“已规划[Scheduled]”、“已发货[Shipped]”、“途中[In Route]”和“取消[Cancelled]”。这些状态可以是“装运运输[Shipping Transportation]”状态的一个类别，该类别是另外一个**状态类型分类**[STATUS TYPE CATEGORY]。



**图 6-12** 状态类别模式

应用递归关系，你还可以将“装运执行[Shipping Fulfillment]”和“装运运输[Shipping Transportation]”都集中到“装运状态”状态类型类别[STATUS TYPE CATECORY]中。“每个**状态类型类别**[STATUS TYPE CATECORY]可以进一步地为一个或多个**状态类型类别**[STATUS TYPE CATECORY]所分类。”当报告装运（或装运集合）的所有状态时，这种模式是很有用的。

也可以根据**状态类型类别类型**[STATUS TYPE CATEGORY TYPE]对**状态类型类别**[STATUS TYPE CATEGORY]进行分类。例如，你可以将“订单执行[Order Fulfillment]”和“装运执行[Shipping Fulfillment]”状态类别归类于“事务流程[Transaction Processing]”**状态类型类别类型**[STATUS TYPE CATEGORY TYPE]，而将其他的状态类型归类到状态类型类别类型[STATUS TYPE CATEGORY TYPE]“参考数据状态[Reference data status]”中。在管理主参考数据时，这种模式将变得很重要。它便于你管理、维护和报告相似类别的状态。你必须十分小心，不要混淆了**状态类型类别**[STATUS TYPE CATEGORY]比如说“装运状态”和创建**状态类型类别类型**[STATUS TYPE CATEGORY TYPE]的类别，比如说“事务流程[Shipping Status]”之间的差别。**状态类型类别**[STATUS TYPE CATEGORY]实例是直接应用于**状态类型**[STATUS TYPE(s)]的，而且还用于对这些状态进行分类；**状态类型类别**[STATUS TYPE CATEGORYTYPE]是用于为**状态类型类别**[STATUS TYPE CATEGORYTYPE]创建一般的分类，它不是直接应用于状态类型的。

**注：正如我们先前所提到的，状态类别模式是使用分类模式增强状态模式的范例。你可以使用每个章节和每个部分中的模式，用于建立满足你特定需求的数据模型。取代将三级分类模式应用到状态模式，为满足特定的需求，你可以考虑选择使用维护隶属关系和方案的三级分类模式、甚至可以使用一级分类模式。**

表6-7展示了状态类别模式如何支持所有状态的组合，如**订单**（图6-1）、**装运**（图6-10）和**工作计划**（图6-11）状态模型图所示的状态。例如，“订单已开立[Order Opened]”状态具有一个状态类型类别标识“5000”，该标识与“订单执行[Order Fulfillment]”状态类别相对应。不是所有的订单状态都是此类别的成员。例如，你看到的“准时[On Schedule]”、“延期[Behind Schedule]”和“过期[Overdue]”则属于“订单规划[Order Schedule]”状态类别。还有意思的是“订单已确认”状态类型也是“订单规划”状态类别的成员。这种情况在复杂的流程中是很常见的。通常相同的状态可以影响不同的流程或被不同的流程所影响。因此，可以将“订单已确认[Order Confirmed]”分类成“订单执行[Order Fulfillment]”和“订单规划[Order Schedule]”。通过关联实体**状态类型类别分类**[STATUS TYPE CATEGORY CLASSIFICATION]，此模式支持这种分类。该关联实体允许**状态类型**[STATUS TYPE]分类成很多的**状态类型类别**[STATUS TYPE CATEGORY(s)]（当然在**状态类型类别**[STATUS TYPE CATEGORY]内也允许众多的**状态类型**[STATUS TYPE(s)]）。

查看下“装运计划”状态，正如所有其他的装运状态一样，它也是“装运执行”的成员。这种分类对报告所有的装运活动十分有用。通过使用“装运执行”，你可以访问状态组，而不是访问单独的状态。

在图6-11中介绍的，与通过/未通过工作计划评估流程的所有相关的状态都被集中到“工作计划通过/未通过”状态类别中。

表6-7中的大多数行所维护的状态类型是“事务状态[transaction status]”**状态类型类别类型**[STATUS TYPE CATEGORY TYPE(s)]，表6-7的最后四行只展现了几个“参看数据状态[Reference Data Status]”**状态类型类别**[STATUS TYPE CATEGORYTYPE(s)]，这说明这种模式可用于捕获企业的所有状态。

**表6-7** 状态类型的类别示例

### 应当在什么时候使用这种模式？

我们在以下几种情形中使用这种模式：

* **当企业想要增强二级、三级或四级状态模式时，当企业需要对不同的状态类型和状态类型类别进行分类的时候。**这种模式能够使用任意多种方法对状态类型进行分类。当然，有些企业不需要使用这种方式来管理他们的状态。
* **当企业需要一种便捷的方式来报告现存的和实体、流程或工作流中所使用的各种状态分类。**通过将最高层级的状态分类分解成第层级的状态分分类这种方式，使得报告所有相关状态变得更加容易。
* **当企业希望使用全面的方式来管理其状态数据时。**对于状态的数据管理很重要的一步是开始将状态分类成可管理的集合或丛集，这些集合或丛集在某种程度上是相关的。本部分所引用的状态也可能是集合的成员，是可用于流程的状态。

### 这种模式的缺点是什么？

此模式的缺点如下：

* **关于需要什么类型的状态分类，此模式没有有效地说明具体的数据需求。**它只是表明状态类型可以分类成任意数目的类别，状态类型和状态类型类别可能有无数种分类方法。这是种懒惰的方式，可以为数据建模人员“捕获所有的”状态类别，这些状态类别可以属于一个状态类型，并不需要分析需要哪些类别以及每个类别的需求是什么。不同的类别可能会有不同的规则或属性与之相关联。即使你使用这种模式建模，也需要分析和理解这些方面。
* **这种模式没有为特定的状态类型类别指定不同的属性或关系。**如果需要维护特定属性和关系的话，你可以使用二级分类模式而非三级分类模式。

### 简介

这种模式意义重大，因为它介绍了如何让**状态类型**成为一个或多个类别的成员。例如，“订单已确认[order Confirmed]”这一状态类型可以是“订单执行[order Fulfillment]”这一**状态类型类别**的成员，也可以是“订单规划[Order Schedule]”这一**状态类型类别**的成员。

在企业对其状态进行分类时，这种模式是非常有用的，我们已经演示了如何将第5章节中的三级分类模式应用到此状态模式中。因此分类是由**状态类型分类**实体的递归关系和**状态类型分类类型**的递归关系支持的。所以类别可以上下滚动。这种模式为企业报告和管理不同的状态类型提供了全面而便捷的方法。对于创建企业范围的有效管理状态的方法，这种模式可以是很重要的一步，比如说在主数据管理项目中。

这种模式没有有效阐释不同**状态类型**分类的特定的数据需求，并且由于它是概括型的模式，所以将它用于收集和验证非数据建模人士的数据需求不是很有效。我们强烈建议你使用这种模式时，要使用其他的文档展，比如说状态装换图、有限状态机图、示例图和实践示例等展现状态类型，以确保所有人对这种模式的工作原理有个清晰的理解。这种模式还假设了不同的类别具有相同的属性和关系。

## 维护多重隶属关系和规则的状态类型模式

随着企业业务流程的复杂化，数据可以通过多种途径传至企业，并且可能会经历各种状态、规则、阶段性关卡或设计规章，这些需要数据以特定的方式来体现。比如说萨班斯-奥克斯利法案这样的法规可能会要求在没有通过其他流程的情况下，数据信息不能处于某一特定的状态，例如，预算信息在通过状态“已完成规范性审查[Regulatory Compliance Review Completed]”之前，它是不能处于“已批准[Approved]”状态的。规则或需求可能会要求两个状态相互排斥，换句话说，一个状态排斥另一个状态。例如，具有很多总分类账账户的资产负债可以有两种状态：“已通过审核[Audited]”和“未通过规章审核[Failed Audit Compliance]”，但是这两种状态从不同时存在；因此，如果一个状态是当前状态，那么另一个状态则被排除。随着时间的推移，一种状态取代另一种状态也是这种情况，例如，状态“符合规定[Complied]”可以被状态“已通过规章审计[Passed Compliance]”所替代。

这些业务规则通常体现在流程指南当中，或者只能通过员工经验的传授而继承下来（最坏的情况）。有时他们被捕获成流程模型。有些先进的组织甚至将他们存储在元数据存储库中，在规则引擎中实现了业务规则，比如说维基百科网页。数据模型也可能会维护和支持这些业务规则。正如你即将看到的，在数据模型中捕获这些规则有很多优点（和一些缺点）。

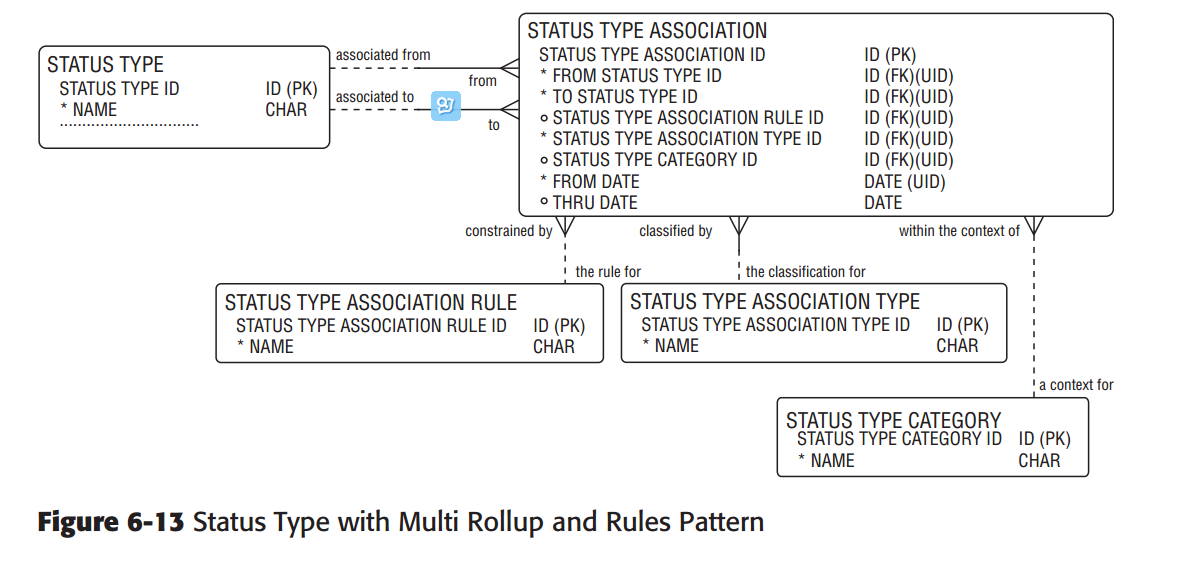
### 为什么我们需要这种模式？

这种模式的目的是捕获规则，这些规则约束了状态相互关联的方式，比如说在状态类型类别的上下文内，某一潜在状态类型是如何与另一个状态类型相关联的。例如，在“订单执行”状态类型类别的框架中，“开立订单”状态与“订单关闭”状态能否在同一时间是当前状态，，是否有些规则说明这是可以的？！这种模式必须十分灵活，能容纳多种状态类型、多种状态类型类别、介绍了状态之间关系的各种分类方式并且考虑到不同的规则，该模式在架构上要足够丰富。

在本章节中，展示了三种不同的状态转换图，分别介绍了订单执行[Order fulfillment]（图6-1）、装运执行[Shipment fulfillment]（图6-10）和工作计划通过/未通过[Work effort go/no go]（图6-11）。在每个模型图中，一个圆圈表示一个状态。在本章节的前部分（状态类别模式），向你介绍了如何将每个状态分成特定的类别。例如，表6-7中的“订单已开立”是根据名称为“订单执行”的状态类型类别来分类的。换句话说，我们捕获了一个事实，即“订单已开立”归类于“订单执行”这一状态类型类别。但是图6-1展示的“订单已开立”和“订单已接收”之间和“订单已接收”与“订单已录入”之间的关联（或转换）以及其他的状态之间的关联会怎么样呢？不同的状态可能会相互关联，正如“订单已开立”和“订单已接受”之间的关联以及其他状态之间的关联所示。你可能需要捕获这些关联。你还可能需要捕获这些关联所处的上下文环境。换句话说，在什么样的长些文中，“订单已开立[Order Opened]”状态与“订单已录入[Order Entered]”状态相关？或者是说，你想知道在“订单执行[Order Fulfillment]”的上下文中，“订单已开立[Order Opened]”与“订单已录入[Order Entered]”相关联。要做到这一点，在**状态类型分类**的上下文中，你必须能将某一**状态类型**与另一**状态类型**相关联。

### 这种模式是如何工作的？

图6-13 展示了实体、主要的属性和关系，用以捕获不同状态类型之间的关系、关于不同状态类型之间的关系规则和状态类型关联，并且假设该关联是在状态类型类别的上下文中。



**图6-13** 维护多种隶属关系和规则的状态类型模式

考虑到多种对等关系、聚合关系或分层关系，在这些架构中，状态可以称为成员，**状态类型**存在多对多递归关系。此模式中的状态类型关联实体解决了状态类型之间的多对多关系。源状态类型标识（**from status type id**）和目标状态类型标识（**to status type id**）是外键，每个标识都代表了状态类型的有一个实例。取代使用一贯的属性比如说父状态类型标识（**parent status type id**）和子状态类型标识（**child status type id**），我们用源状态类型标识（**from status type id**）和目标状态类型标识（**to status type id**）作为外键，因为我们考虑到在此模式中不仅仅有父—子关系，而是还有很多其他的关系类型。例如，一个状态可能与另一个状态是对等。

每个**状态类型关联**实体可以具有一个源自**状态类型关联规则**的外键，状态类型关联规则标识（**status type association rule id**）。此外，每个**状态类型关联**可以有一个源自分类实体的外键，状态类型关联类型标识（**status type association type id**），该分类实体的名称是**状态类型关联类型**（将这样的关联归类以“对等关联”、“聚合关系”或“分层关系”）。最后，每个状态类型关联可以具有一个源自**状态类型分类**实体的外键，即状态类型类别标识（**status type category id**）；这是因为该关联是处于状态类别分类的上下文当中。这些外键表示规则的实例和**状态类型关联**所支持的关联类型。

**注：此模式中的状态类型分类与第5章节中的状态类别模式所使用的状态类型类别实体相同。“目光敏锐”的读者可能已注意到对状态类型关联进行分类，我们使用的是二级分类模式。换句话说，我们确切是将状态类型关联类型和状态类型分类作为状态类型关联的分类。我们没有使用三级分类模式将这些不同的分类结合起来。**

状态类型关联规则可以维护以下几个值，但并不仅限于这些值：

* **优先型**表明某一状态类型必须在另一个状态类型之前出现。例如，可能会有规则规定：“订单录入”状态必须在“订单确认”状态之前出现。
* **兼容型**表示两个状态类型可以同时存在，因此**状态类型关联**维护的关联是有效的是允许的。
* **暗示型**表示一种状态类型暗示了可以推断出另一种状态类型。例如，取消销售订单状态，可能暗示还存在另外一个“订单已取消”的状态。这个规则表明即使两个状态有类似的意义，但是并不意味着他们可以彼此替换。在试图为整个企业映射状态时，这个规则是很有用的。可以说中国的订单录入流程中设置了“开立订单”状态。这暗示了欧洲订单录入流程也有“订单已开立”状态。
* **互斥型**是指如果第一个状态类别存在了，那么第二个状态类别是不允许同时存在的。这个规则在操作系统中很有用。例如，当产品的当前状态是“开发”时，产品的“生产”状态可能会适用“互斥性”规则。这并不表示每个可想到的规则如果没被另一个规则所支配，就必须为“互斥性”规则所约束。它的意思是如果你需要明确声明两个状态时互斥的，那么你可以使用此类型的规则。
* **替代型**表明第一个状态类别和第二个状态类别可以相互替换。这两个状态可以同时是当前状态。例如，两个不同的国家会有两个不同的装运流程，“交付”和“装运交付”，他们是可以相互替换的。
* **过时型**表示第一种状态已然被取代了，已经被第二种状态所取代了。例如，一个状态变更了其名称，那么原来的状态名称（源状态）可以被新的状态名称（目标状态）所取代。“过时性”和“替代性”之间的差别在于虽然状态可以相互替换，可以同时作为当前状态，但是过时性暗示了某一状态类型将不再使用。在你将数据从一个系统迁移到一个替代系统中时，这个规则是很有用的。

每个**状态类型类别**有一个名称，该名称为**状态类型**间的关联提供了上下文。例如，在表6-8你可以看到图6-1所介绍的**状态类型类别**“订单执行”。因此，在“订单执行”这一状态类型类别的上下文中，存在于“订单已开立”与“订单已接收”之间的“互斥性”规则是有效的，但是在另一个状态类型类别，“订单规划”的上下文中，它可能就是无效的。然而，在该上下文中可能会有不同的规则。

每个状态类型关联类型都有个名称，该名称为两个**状态类型**间的关联提供了特定的分类。这种分类类型描述了状态类型关联的“结构”。例如，表6-8中的大部分示例都展示了“对等”的**状态类型关联类型**，一个示例展示的是“聚合”关联类型（表的最后四行）。你如何管理状态类型间的关联取决于他们的结构类型。例如，若干状态类型处于“分层结构”中，删除顶层的状态类型或主状态类型，意味着同时删除了其他的状态类型。关于此话题的更多信息请参看本书的第四章节内容。

由于这只能模式使用**状态类型关联规则**实体来支持规则，所以你可以定义两个状态类型之间的关系行为。例如，在表6-8的第一行，在“订单执行”工作流的上下文中，“订单录入”和“订单确认”之间有个“优先性”规则，这表示“订单录入”状态类型必须优先于“订单确认”状态。这张表还展示了“订单录入”状态和“订单确认”状态之间还存在“兼容性”规则（伴随着很多同样具有兼容性规则的其他状态类型一起）。这表示在“订单执行”状态集中，“订单录入”和“订单确认”状态类型之间存在关系，所以在“订单执行”的上下文中，使用**状态类型关联**实体维护关联是有效可用的。这种模式还捕获了一些其他有意思的而在图6-1此状态模型图中并未介绍的关联。试想大型手机制造商的数据专业人士访问了一些主题专家，探讨了订单执行流程方面的内容，发现了那些不在状态模型图中各个状态类型之间存在一些非常有趣的规则和关联。例如，主题专家介绍了财务报表系统如何从订单录入系统中获取取消的订单的数据，并汇报信息。在财务报表系统中，使用“取消订购订单”表示“订单已取消”状态。所以，“取消订购订单”是财务报表系统所使用的状态。在此模式中数据专业人士可以捕获这种重要的关系。在表6-8的倒数第二行中， “订单执行”的上下文内，“取消订购订单”是与“订单已取消”相关联，并且此关联有个“暗示性”规则。

另外一个主题专家提到先前的订单录入流程使用了状态“开立订单[Open Order]”，在当前流程中该状态被“订单已录入[order opened]”所取代。很多年长的职员仍然引用以前的状态。表6-8捕获了“开立订单[Open Order]”已废弃，“订单已开立[Order Opened]”代替了它这一事实。

第三个主题专家提出在亚洲订单执行流程与入6-1所介绍的流程相同，除了所使用的状态类型名称不同。例如，表6-8的最后一行中，“订单已录入[Order Entered]”可以替换成“订单已输入[Order Inputted]”。

另一个主题专家发现“订单取消[Order Cancelled]”此状态类型与“订单收入记录[Order Revenue Recorded]”状态不能同时存在。换句话说，你不能取消一个尚有收益记录的订单。使用这种关系是出于虚报收入以及违反规章的担忧。因此，在表6-8的倒数第五行中，“订单取消”与“订单收入记录”之间存在“互斥性[Exclusion]”规则。

最后，企业已为“订单执行”的状态类别制定了业务规则，“因缺货而取消的订单”状态类型不可以作为“订单取消”的子类，因此，这两个状态类型之间存在“互斥性”规则。在此案例中，尽管这个状态类型可能在另外一个状态类别类型中可以应用，但是企业决定在订单执行流程中，不允许这个状态存在，流程会一直等待直到库存有货为止，以至于企业不会因缺货而丢失订单。这个规则是一个“聚合”的**状态类型关联类型**，因为它涉及到这两个状态类型之间的父-子聚合关系。或许你会说，“如果状态类型之间的关联不被许可，那么就不捕获该关联！”这么做是正确的，但是从数据管理的角度来说，与通过规则明确定义关系行为的这种管理方式相比**，**忽略互斥性规则（或任何支配行为的其他规则）的管理方式不够严谨。

你或许还会问，“规则不就是关联类型吗？那为什么还要分离出规则实体？”我们认为，区分规则和关联类型是有合理的原因的。**状态类型关联类型**是对状态类型间的关联进行分类的实体，例如，“对等关联”、“分层关联”或“聚合关联”。状态类型关联规则定义的是两个状态类型之间的行为方式。

将规则从类型中分离出来的第二个原因是它是一个很好的政策，可以确保在一个实体中维护支配状态之间的行为规则，而该实体与维护可行的关联分类的实体，比如说维护“对等关联”的实体，是不同的。这个观点与罗恩罗斯所阐释的概念类似，他表示一般情况下“规则应当在流程和工作流之外独立存在。”13

**注：在图6-13的备选模型中，使用关联实体状态类型关联规则（也有可能与状态类型类别相关甚至是在状态类型类别的上下文之内）解决了状态类型关联和规则类型之间的多对多关系。这个模式允许相同的状态关联实例与很多的规则相关联。例如，“订单已录入”和“订单已确认”之间可以有状态关联实体，该关联具有规则类型（与规则类型相关联）“优先型”和“兼容型”。**

这种模式包含为一些信息建模的方法，在状态模型图中，该方法被模拟成状态类型关联的实例。从数据管理和元数据管理的角度来说，这是十分有用的。很多的状态模型图被当成模型图出版，但是模型图所展现的逻辑只能通过应用架构或规则引擎来实现。这样很好，但是在数据层级上捕获状态间的关系也是有可能的。此模式考虑到数据架构中的规则说明文件可能会被修改和添加，所以可以通过更改或添加**状态类型关联**的其他实例来完成规则的添加或更改。这也意味着规则引擎和应用程序可以是较偏向‘数据驱动’的。正如你所了解，在数据库中更改一条数据的值比更改应用程序的代码要更加容易。

### 应当在什么时候使用这种模式？

这种模式十分复杂，但是很灵活，应当在以下情形中使用：

* **当企业为了让数据管理变得更加严谨，想要维护状态类型的规则时。**这种模式的规则的功能是能够让企业明确地定义状态类型之间的行为。如果企业的用意是在数据架构中维护状态信息，而不是将他们放入应用程序中，这种模式可以让你构建应用到数据的智能技术，从而更好地管理状态数据。
* **致力于数据管理投资的较为成熟的数据企业。**某些学科是需要使用这种模式的。这种模式是为下面这样的企业所定制的：具有强烈意愿并致力于发展成数据驱动型企业。

### 这种模式的缺点是什么？

这种模式的缺点如下：

* **这种模式很复杂。**即使是经验丰富的数据专业人士理解此模式都有困难。理念上而言，应当在较广泛的数据管理项目的上下文中实现和使用这种模式。使用这种模式需要训练编程人员和应用程序架构师的理解力，并且需要企业致力于投资整合数据管理策略。也就是说，一旦理解并使用了这种模式，它会是状态管理的强大工具。
* **在元数据存储库或规则引擎中可能已经维护了规则。**如果你的企业已经在使用元数据存储库或规则引擎，那么就可能在其中的一个地方维护了这些规则类型。

### 简介

在此部分你研究了如何通过规则管理和描述状态类型不同实例之间的关联。这种模式允许状态类型有多种相互关联的方式，允许随时间推移添加新的关联规则和关联类型。这种模式是很重要的，因为它能够在某个状态类型分类的上下文中管理状态类型的多种相互关联或可能相互关联的方式。例如，这些规则可以维护“互斥型”关联，而且在很多其他的规则当中，根据上下文状态所在的流程，还可以确保关联是“兼容型”的。这种模式是个强大的工具，有助于清晰地管理状态类别数据。

这是种复杂的模式，需要得到企业的认同。然而，以数据驱动的方式来管理状态数据，此模式的功能强大到令人难以置信

## 模式的总结

**表6-9** 是本章节所有模式的简介

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 一级状态模式 | 定义、描述状态并特定地支持将状态当成特定实体的“事件”属性来维护。 | 当具有静态的状态集合时，当实体不会出现新的状态时  当没有大量的状态时  当需要相对简单点可以用来收集和验证状态的数据需求的模型时  当数据专业人士希望展现企业对状态看法的不同之处时 | 当需要灵活的模型时，例如，可以在没更改模型的情况下，可以添加所需的状态。  当你需要管理状态之间的规则时。  若果有很多的状态，此模式更难以一致的方式管理所有的属性（以及属性的潜在变更）。  当需要实体的状态具有多种分类时 | 易于启动基本业务数据需求的建模工作，使得阐释通用术语和状态问题变得更加容易。  特别是将许可的状态作为实体的属性来展示。 | 不是灵活的模型，如果出现新的状态，会许需要添加新的属性，因此需要更改数据模型。  不能处理状态的分类或状态类型之间的规则。 |

表6-9（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 二级转台模式，当前状态 | 企业希望灵活地维护任意数目的实体许可的状态类型，但是希望只捕获某一给定时间点的当前状态时 | 当企业希望只捕获实体的当前状态以及实体可能具有的各种状态时。  企业不关注实体所具有的历史状态时。  随着时间的推移需要提供任意数目的可能会出现的潜在状态时 | 在给定的时间点实体可以有多个状态时。这是个特定的模型，在只需要维护一个当前状态的情形下使用。  希望捕获实体所具有的历史状态时。  需要为企业各分支机构捕获不同的状态类型时 | 非常适用于特定的需求（当企业只希望在某一时间点为实体捕获一个单一的状态时）。  它捕获了实体可能具有的许可状态类型，并且当新状态类型出现时，可以添加他们。 | 除了能满足特定的需求，他不是一种十分灵活的模型。  在整个企业中使用一个单一的状态是不能表达出企业各分支机构对状态的不同理解。并且还有可能需要存储不同的状态类型。  对于阐释、收集、和/或验证状态的数据需求，这种模式不是一种非常有效的方式。  由于一种状态会覆盖另一种状态，所以这种模式不能维护实体的历史状态。 |

**表6-9** （续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 三级状态模式 | 这是种可以非常灵活地维护状态数据的模型，支持任意数目的实体状态，内购容纳无数的可行的状态，并且还维护很多的方式，用来维护日期导向型的状态。 | 当需要一种灵活的方案来处理状态时，例如，可以接受任意数目的可行的状态类型，可以为实体分派无数的状态，并且对于状态时间的记录可以提供多种选择。  当所需的状态类型定义不是很明确或不是很了解时。  当企业需要十分一致的模型来处理所有实体的状态时。 | 作为一部分声明，或是用来收集和验证非技术人员的需求。  如果情形是系只需要记录实体的当前状态。  当不同的状态类型具有不同的属性或关系时（尽管可以通过添加状态类型的子类，然后记录每个状态子类的属性这样的方式对此模式进行修改）。  当需要更加灵活的模式时，需要让状态进一步模块化，或需要状态具有分类或规则时。 | 这是种可以非常灵活地处理状态的模型，此模式能容纳任意数目的状态类型，可以有无数的状态分派，随着时间的推移，若有必要，可以添加其他的状态类型。  这种模式为支持各类实体的状态和状态类型提供了一致的方法。 | 对于非数据专业人士而言，这种模式很难理解。  这个模式不支持特定的业务规则，即使有约束，但此模式允许状态有无数的可能性。  此模式不能维护特定状态类型的特定的属性和关系。  使用此模来定义范畴或收集以及验证状态的数据需求不是很有效。 |

**表6-9** （续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 四级状态模式 | 这是一种“即插即用”的模式，可用来对状态进行分类。当给需要维护状态的数据模型添加实体时，你所需要做的就只有将该实体关联到**状态应用**，这样这个实体便可以使用所有的**状态类型**。 | 当企业需要非常灵活的状态建模方式时，可以使用这种模式。  当企业希望以一致的方法来管理状态数据时。  当企业承诺使用集成的方式来管理状态数据时。 | 当企业的视角和状态数据的战略性视图有限时，对于那些企业使用这种模式就是杀鸡用牛刀了。  当需要专门的模型来帮助定义范围、收集和验证状态的数据需求时。  当需要为特定的状态类型指明特殊的属性和业务规则的关系时。 | 是非常灵活的模式，可以添加新的状态，能抵用各种事件成分来表示状态。  它可以一致的管理状态；由于所有的状态使用了同一个建模结构，所以它可以臆测出新实体的建模状态。  它可以构建数据驱动型架构，使得通用软件、服务和应用架构的开发更容易。 | 由于它是个高度概括的模型，所以对有些企业而言，它太复杂了，是个很难理解的模式。  缺少业务规则的执行。  不太适用于定义范围和/或收集以及验证数据需求。我们不会将此模型展示给非技术相关人士看。  它没有为状态各状态类型提供独有的属性和关系（除非构建状态类型的子类）。  可能会导致性能问题。 |

**表6-9** （续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 状态类别模式 | 通过将第5章节中的三级分类模式应用至本章节的**状态类型**实体上，这个模式以非常灵活的方式对状态类型进行了分类。可以使用这种模式来强化本章节中的二级、三级或四级模式，并且此模式可以全面管理不同的状态类别。 | 当企业需要以非常灵活的方式捕获不同状态类型的分类时。此模式对状态类别的管理有帮助。  当需要通过状态类型的分类来提高二级、三级和四级状态模式时。  在主数据管理工作中状态的分类很重要时 | 当企业的视图和状态分类数据的战略视图非常有限时。对于那些企业而言，使用此模式无异于杀鸡用牛刀。 | 这种模式非常的灵活，能够容纳无数的状态类型分类。  它为整个企业提供了非常全面地的状态管理方法。 | 可能对于某些企业而言，此模式太过复杂，在数据建模中它没有那么复杂，但是也没简单到不需要全面的状态类型类别。  各状态类型和状态类型类别必须具有相同的属性和关系（除非你创建**状态类型**和状**态类型类别**的子类）。  由于这是非常泛化的模型，使用这种模式来定义范围或收集以及验证状态的需求不是很有效。 |

**表6-9**（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 描述 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 维护多重隶属关系和规则的状态类型模式 | 此模式可用来强化二级、三级和四级模式，维护状态类型相互关联方式的规则，可以更好的管理不同的状态。 | 当企业需要捕获支配状态类型之间关系的行为规则时，以及需要状态类型以多种方式相互关联时。  当企业致力于使用数据驱动的方式管理状态类型和状态类型之间的规则时。  当企业致力于投资更加复杂的数据管理方案时。 | 当企业很少有复杂的状态类别数据视图时。对于那些企业来讲，使用这种模式是大材小用。  当企业数据管理技能有限或缺乏投资状态数据管理的决心时  当企业已有元数据环境或规则引擎时。 | 这种模式可以通过维护状态类型相互关联的各种方式和支配状态类型之间的行为规则。  它强化了二级、三级和四级模式。 | 对于有些企业而言，它太过复杂了。  由于这种模式非常的概括，所以使用它来定义范畴或收集验证状态的需求不是十分有效。  在元数据存储库或规则引擎中可能已经维护了这些规则。 |

**参考文献**

1 查看 www.wikipedia.org/wiki/State\_diagram.

2 此定义摘自 WordNet (r) 2.0 (August 2003).

3在有限状态机中，这个有时被描述成二级制输出的接收器和识别器。

请查看 *UML In a Nutshell by Sinan SiAlhir* (O'Reilly, September 1998).

4 此见解有技术编辑提供(Ed Landale).

5 此定义源自*Dictionary.com.* 请查看

*http : //dictionary.reference.com/browse/order.*

6 源自*www.dictionary.com.*

7与萨班斯奥克斯法案相关的一些常见变更, 请参看 *http : //www.sec.gov/divisions/corpfin/faqs/soxact 2002. htm.*

8关于装运模型的更多信息，请参看数据模型资源手册卷1，修订版的第5章节*A Library of Uniz,ersal Data Models for All Enteryrises,* By L. Silverston, (Wiley, 2001) 。

9 关于工作计划模型的更多信息，请参看数据模型资源手册卷1，修订版的第6章节， *A Library of Universal Data Models for All Enterprises, By L. Silverston, (Wiley, 2001)*

10 更多关于业务规则的信息，请参看本书的第8章节

11 更多关于企业如何使用这些模式创建一致的数据策略的信息，请查阅本书的第9章节和第10章节。

12 关于分层关系、集合关系和对等关系的更多内容请查看本书的第4章节，而更多关于数据分类的信息请参看本书的第5章节内容。

13 请看 *Ronald G. Ross*所著的*Principles of the Business Rules Approach*  (Addison-Wesley, 2003).

【注释1】current indicator 当前称呼？

【注释2】RESERVATION 一个实体 暂译“预约”

【注释3】“rope”本意是指绳索，这里用的是引用意，根据上下文臆测是“繁琐、复杂”的意思。