# 角色应用：当事人是如何参与的？

“这张发票上的运达客户是谁？”“谁是这个项目的项目经理？”

“这次的股票交易对手是谁？”所有的这些问题都有两个共同的方面。其一，他们全是关于当事人的问题，即，“谁是……？”其二，这个“谁”总是关于一个当事人是如何被卷入到其他实体中的。换而言之，当事人在业务活动的上下文情境中或其他实体的上下文情境中扮演了什么角色（例如，在订单、产品等实体的上下文中当事人的角色）？

在二章节中我们讨论了处理当事人在企业的整体上下文中所扮演的角色的重要性。记录客户、供应商、合作伙伴、雇员、物流服务供应商、卫生保健提供商、子公司和交易对手的信息对于你成功地理解整个企业是很重要的。但是仅仅了解这些声明角色是不够的。某个人或组织所参与的企业具体业务活动或其他实体，这样的上下文情境也需要清楚。例如，正如第二章节所示，一般而言，一个组织在企业中可以被声明成一个客户，而在订单的上下文中则可以被声明为“运达客户”和/ 或“付款客户。”所声明的“客户”可能不是十分的特定而不足以用于支持订单，而“运达客户”和“收款客户”出了相关订单的范围，可能毫无意义。第二章节所介绍的声明性角色支持了有关当事人所扮演角色的“企业视图”的需要。当事人是如何参与到一些商务活动或实体中的，本章对于这一点给出了解释。

**注：尽管通常情况下“运达客户”和“付款客户”都是上下文角色，但是在某些情况中他们也可以是声明角色。例如，你可能想要声明一个有效的运达和收款客户，因此在使用这些角色之前要设置他们为声明角色。**

## 这类模式的意义是什么？

理解一个人和组织与业务活动或实体互动的上下文情境是很重要的。通过下面的问题，给出最佳的描述，“**当事人**是如何具体地参与到我们企业的各个方面的”

* 执行业务的过程中涉及到哪些人员和组织的属性？1
* 执行业务的过程中人员和组织扮演了什么样的角色？
* 执行业务时，人员和组织能否扮演多个角色，还是不允许他们这样做？

正如我们前一章节所述，有些人发现将声明角色想成是设置该角色而将上下文角色想成是使用一个角色是很有用的。可能有声明角色，后来被用在另一个实体的上下文中。例如，你可以使用**供应商**这一声明角色，然后可能需要将这个声明角色关联到一个**产品**实体，来指明供应商可以提供什么产品。因此，你可以使用第2章节中的模式来设置**供应商**这一声明性角色，然后你可以将这个声明角色关联到**产品**。然而，还有很多方法可以将这个声明角色关联到另一个实体，本章展示了这样做的模式和备选方案。

你也可能有不需要设置成声明角色的上下文角色。例如，对于**装运**你可能有一个“收货员”角色。一般情况下，这个角色不设置成声明性角色。尽管如此，这仍然是个上下文角色，本章节中的模式展示了为当事人到另一个实体的任何类型的关系建模的备选方案，不管事先是否声明过该当事人的角色。

## 本章节包括哪些内容？

本章最初定义了什么是上下文角色。然后本章描述了用于支持任何企业建模需求的数据模型模式，根据特定的业务活动或实体为企业所做之事建模。本章描述了每个模式是如何支持上下文角色所牵涉到的属性和关系。

本书的大多数章节都一样，每种模式的建模风格刚开始是最特定的风格（一级上下文角色模式），随章节逐步进展到较为泛化的风格（三级上下文角色模式）。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方式。在本章的最末我们也介绍了混合型模式，即在一个单一的模式中结合了特定和泛化两种建模风格。总之，这章节包括以下内容：

* 上下文角色的定义
* 不同层次的上下文角色模式
* 何时使用和何时不使用上下文角色模式
* 深入了解每种模式
* 每种模式的简介、优缺点和何时使用以及何时不使用他们

## 什么是上下文角色？

上下文可以被定为“事件发生的情形；背景”2，一个角色可以被定义为“指派给一个人或组织的行为和活动或是一个人或组织需要或期望的行为和活动。”3所以，这在我们的模式中是什么意思？此处我们描述的是一种情形，在该情形中，人或组织可以在某个业务事件、事务、事件或数据块的上下文中行为或活动。换句话说，上下文角色定义了当事人是如何卷入到另一个实体的上下文中的。

理解上下文角色不仅需要给实体指派一个角色，而且需要理解所描述的角色在关联实体的上下文中是具有重大意义的。例如，对于一个大型工程公司可以给某个特定的项目指派一个职员，如“项目负责人。”在某个特定项目的上下文中，该上下文角色就是“项目负责人”。

不要将上下文角色同声明角色相混淆，这点很重要。我们通过问以下这些问题，来强调他们之间的区别：

* **这个角色在某个特定的业务信息块、事务或事件的上下文中是否很重要？**如果答案是肯定的，那么这个角色通常是个上下文角色。例如，确保订单质量的人可能扮演了“质量保证人”这一上下文角色，这个角色是不需要声明成声明性角色的。
* **这个角色在企业整体的上下文情境中是否重要？**如果答案是肯定的，那么该角色就是个声明性角色，**客户**角色可能不仅是对于某一个事务很重要，而且通常情况下该角色对于整个企业来说都很重要。声明角色可用于和依赖于本章上下文角色模式的应用。例如，可以先声明一个**客户**。然后将这个**客户**关联到某一个特定的**订单**。

## 一级声明角色模式，属性

经常看到特定的角色与一个实体相关联，并作为该实体的属性。例如，**订单**中运达和付款客户定义为该订单的属性。付款客户和运达客户的名称被包含在某个特定订单实例的属性中。

### 为何我们需要这种模式？

图3-1所示的是一级上下文模式、属性，为上下文角色建模提供了最特定的方法。这些角色被当作某一特定实体的属性来维护。这个模式为某个特定数据块也提供了所有角色范围的视图。另外，使用这类模式的数据模型；物理数据库；星型构架维度；中介ETL（提取、转换和加载）数据库多得惊人。此外，使用这种模式还有合理的缘由；例如，当一个角色不需要自身的记录时。例如，在人员实体中，可能需要有一个“母亲的名称”这样的属性。很多时候，另外一个人员实例的存在可能是没有任何意义的，也不用展示人员与其母亲之间的关系。基于这些原因我们认为了解这种模式的优缺点很重要。

这种模式支持声明角色的基本方面，例如：

* 使用定义展示某个实体的角色，就行使用属性定义来捕获每个角色一样。
* 这些角色与实体自身的关系
* 一个人员或组织在同一时间能够扮演多个上下文角色。

现今一些数据专业人士将这种模式视作“如何不这样做”的模式，因为这种模式一旦实现可能会导致冗余数据和不一致数据的出现。例如，如果Kathy Morris是项目负责人和项目工作人员，那么你可能将她的名称存储在同一个实体的两个地方。换句话说，这种模式可能在两个不同的地方处理了相同的实体。从关系型模型的角度来看这个问题，这个结论是正确的。如果你的企业按照这种方式来建模，那么这种模式不太适合你。

然而，公平的来说，将这种模式被当作“错误”立即摒除是忽略了该方法的一些正面特质。例如，这个模式的一个用途是可以描述数据的需求、展示范围以及为初步掌握某一主题领域的通用术语提供了易于理解的方法。它也能是功能强大的工具，有助于与相关人士进行交流沟通。同时，它能够帮助快速启动数据建模工作，因为它很容易为商业人士和非数据专业人士所理解。

正如我们前面所说，将这种模式作为备选模式来展示的一个很重要的原因是，可能有些情况，你确实仅需要维护某个角色的单一的一条信息。在这种情形中，重要的是要考虑将角色作为当事人的一个属性而不是创建一个当事人的实例以及与这些当事人相关联的角色。例如，在**雇员应用**中，需要有一个属性表示最后一个雇员名称，不用维护该组织雇员角色的单个实例。（虽然视情况而定，但是你可能希望为该组织建立具有他们角色的当事人模型，如果有此需要，将要获得更多关于该组织作为独立实体的数据）。

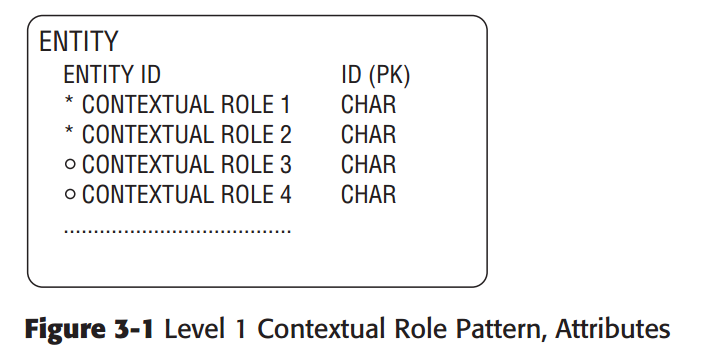
当只需要某事务的一条信息时，该信息通常被当作一个属性，因为角色没有其他信息，所以这种模式将这些角色当作属性来存储。William G. Smith, 一个著名的数据建模行业的领导者，指出实体的“意愿和手段”的规则（这是他所发展的六条规则中的一条，是关于什么使得事物变成一个实体），该规则是“业务必须有意愿和手段至少捕获和存储候选实体的一个非键属性。”4所以在某些情况下，当不需要角色的其他信息时，可以把角色当作属性来建模。

**注：关于前一段内容的另一种意见是：如果你所需的是一条信息，例如父亲的名称/母亲的名称，因为没有其他的人员信息，那么这就可能不是一个角色。这种观点认为这只是捕获了数据的任一实体的一个描述性属性。然而，一种不同的观点是：因为角色定义了一个当事人是如何参与到企业或另一个实体当中的。所以如果它的本质是一个角色，至于它维护了多少数据已无关紧要。 例如，某个人员的妈妈都扮演了该人员的“母亲”这个角色，不管我们维护了多少关于她的数据。**

这种类型的模式是关于企业通常如何看待某个事物、事件或其他业务操作所涉及到的人员和组织。我们通常使用这个模式作为展示一个事物、事件或其他业务数据中相关当事人的第一步。在这些情形下，可以把这个模型当作一个有效的工具使用，通过使用一个简单的数据模型，使得数据专业人士开发数据需求变得更加容易。

### 这种模式是如何工作的？

图3-1阐释的是如何使用**实体**的属性来介绍该实体的上下文角色。上下文角色1、2、3、4属性表示的是**实体**允许的上下文角色。上下文角色属性维护在**实体**上下文中扮演了特定上下文角色的人员或组织名称的数据值。



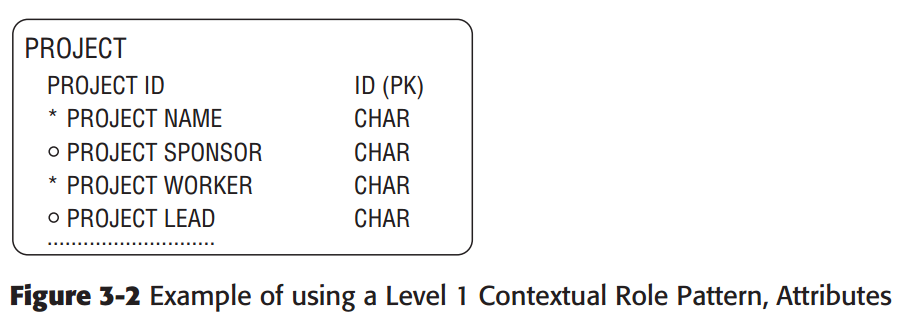
**图3-1** 一级上下文角色模式，属性

这是个非常简单的结构，支持实体所具有的所有角色。它是一种用来展示这些角色的非常特定和精确的方法。 仅仅捕获了与该**实体**相关的一组上下文角色—不多，不少。每个角色均可被指定为必要的或可选的。这意味着某些角色必须为**实体**的每个实例而存在。每个上下文角色属性都应当具有一个角色的定义。

图3-2进一步说明了这种模式如何做到支持某一数据块的特定的上下文角色。方案如下：XYZ是一家大型国际技术公司，该公司有一个辅助性的需求，需要维护过去项目的数据，该项目是在他们加入XYZ公司之前，该公司的以前的雇员所做的。作为整体数据建模工作的一部分，有必要有一个**项目**实体来展示以前项目的名称和一些与此项目相关联的人员的名称。

在了解数据需求后，数据专业人士基于一级上下文角色模式，将角色作为属性的模式，创建了一个初始模型。图3-2包含的是项目实体所需的上下文角色，即项目赞助商、项目工作人员和项目负责人。这些都是公司希望维护的角色，没有维护这些当事人和/或他们的角色，也没有开发一个外延的模型，因为这只是个附属需求，除了这些实体的名称不需要任何其他的信息：5

* 项目赞助商可以是指资助项目的人或组织。
* 项目工作人员是可以定义为执行或被指派到某个特定项目的活动的某个人。
* 项目负责人可以是指为某个特定项目提供指导的人或组织



**图3-2** 一级上下文角色模式，属性的应用示例

项目名称属性也被作为项目的一部分来处理。这个属性可以被定义为文本引用[注释1]，通过这个文本引用指定项目的实例并将该项目实例同其他项目的实例区分开来。项目本身是“一项需要协同合作的工作”。

需要注意的是项目工作人员是一个强制性的属性。这表示必须给**项目**分配项目工作人员。项目赞助商和项目负责人不是强制性的。这表示一个项目实例不是必须要有指派的项目赞助商或项目负责人。

在表3-1中你可以看到四个不同的**项目**实例，这些项目实例具有不同的项目名称：“客户主数据项目（Customer Master Project）”、“人力资源数据库（Human Resources Database）”、“信息架构标准（Information Architecture Standards）”和“萨班斯-噢克斯利法案报告（Sarbanes-Oxley Reporting）”，其项目标识分别是1001、1002、1003和1004。每个**项目**实例都有一个组织资助该项目，该组织体现在项目赞助商属性中。项目赞助商的定义并没有排除某个人作为项目赞助商。只是在我们的示例中所有的项目赞助商都是组织。在案例“客户主数据项目”中，项目的赞助商是“主数据管理部”。这表示XYZ公司的“主数据管理部门”将用它的预算来资助“客户主数据项目”。

每个不同的**项目**实例也都有人扮演项目工作人员这一角色。你可以从表3-1中看到“Neena Davies”从事于“客户主数据项目”和“人力资源数据库”这两个项目。每个项目工作人员实例都包含了从事该项目的人员的名称。项目工作人员的定义将组织作为项目工作人员这种情况排除在外

**表3-1**特定上下文角色，项目示例

最后，从表3-1中你可以看到每个项目示例都可以包含一个项目负责人。在“信息架构标准”的案例中，其项目负责人是“Vinnie Chintappaly”。**项目**“萨班斯-噢克斯利法案报告”没有项目负责人，这种情况是允许的，因为项目负责人的属性不是强制的。“Una Corr”负责“客户主数据项目”和“人力资源数据库”这两个项目。基于XYZ公司所提供的定义，项目负责人可以是人也可以是组织。

### 何时应当使用这种模式？

我们通常在以下情形中使用这种模式：

* **有定义明确的实体，该实体有一组静态的上下文角色。**根据调查该实体只需要三个上下文角色，并且这是不会改变的。
* **有些情形数据专业人士需要业务需求更加容易理解，或许需要为其他IT专业人员或管理人员阐释作用域声明。**上图可以作为作用域声明的一部分，展示了与业务相关的三个不同的关键角色。
* **我们有些情形中，只需要人员的名称数据项，不需要维护一条完整的记录或该人员的实例。**例如，有时数据需求是只需要某个人员父母的名称（例如，某些国家中的就职申请或护照申请需要），可以把它看作是上下文角色并作为一个属性来处理，除此外不需要维护该父母的任何额外信息。

### 这种模式的缺点是什么？

这种模式的缺点如下：

* **这种模式不适合动态的环境，在动态环境中，会频繁的添加新的角色类型，或者在动态的环境中这些角色不是很好理解。**随着时间的推移，新的角色出现时，我们已了解先前模式的实现导致了很多的问题，因为新角色要求更改模型以及基于该模型的所有底层数据库。
* **相同的信息，比如说人员或组织的名称，可能在很多不同的角色中重复。**这将会导致两个问题。第一，你可能会存储冗余信息。第二，更为重要的是冗余信息经常不同步。这可能会引起数据质量问题，例如，一个角色中将“xyz Corporation”作为名称存储，而在另一个实体中存储的却是“XYZ Incorporated”。
* **这种模式并没有明确的将人员角色与组织角色划分开。**在先前的模式中，定义表明了项目工作人员是特定于人的角色，但是实体关系模型并不强制执行这个规则。
* **在这种模式中，很容易将声明角色同上下文角色相混淆，如果某一角色是否是声明性角色的话。**例如，在图3-2中，你所看到的项目负责人属性。一个公司的项目负责人可以被当成整个企业的项目负责人而不是仅仅是某个特定项目的负责人。项目负责人的定义表明它是一个特定项目的角色，但是在下一节你能看到当实体**项目负责人**被当作其自身的声明角色实体来维护时，其定义扩大了（如图3-4）。当试图搜集实体的角色作用域时，了解这个角色是否是声明性的和/或上下文的是很重要的。
* **这种模式不适应多重性属性。**公平的假设在很多项目中，不止一个项目工作人员参与该项目。这意味着如果一个项目有50个不同的项目工作人员，那么需要将他们分别作为独立的属性全部捕获。当需求发生变化时，这个可能不是最有效的数据建模方法。

### 简介

在本节中，对于展示定义明确的静态实体的所有角色的范围，一级上下文角色模式是很有用的。将每个角色当作实体的属性分别的展现并且每个角色都有一个定义。这种模式的意义是作为的一种方式，以一种非常易于理解的建模风格阐释了与某一实体相关的角色。

这种模式以一种非常易于理解的方式帮助启动数据建模工作，了解常见的上下文角色术语以及展示与一个实体相关的上下文角色范围。

这本章你也能看到在同一时间或随时间推移，人和组织能够扮演多个上下文角色。如表3-1所示，“Una Corr”在一个项目中扮演了项目负责人角色，而在另一个项目中扮演了项目工作人员角色，这样会导致数据完整性问题，因为维护了同一当事人的冗余信息和不一致信息。有些角色只能被定义为组织或人员上下文角色。例如，项目工作人员被定义为“执行或被分配到某一个特定项目活动的人员”。

此处的缺点是该定义表明这个角色是人员角色，但是数据模型并不遵守这个规定。对于某些企业而言，将组织角色和人员角色分开来可能很重要。但这种模式并强行将人员角色与组织角色划分开来。最后，这种模式非常适合于定义非常明确的静态环境。如果角色发生变化，这种模式会导致模型以及后续实现的不稳定性，因为需要更改底层数据结构。一般而言，不应当在关系型数据库环境中实现这种模式。

我们在很多旧系统中遇见过这种模式，如在星型结构的维度中或在ETL环境的中介数据库中。我们认为了解这种模式的优缺点是很重要的，这样你才能够对于何时使用这种模式还是使用一种不同的声明角色模式这样的问题做出明智的决定。

## 一级上下文角色模式，关系

有时希望为上下文角色建立稍加灵活些的模型，但仍保持特定的建模风格。当这个数据可以作为上下文角色与声明性角色之间的关系来处理时，可能也希望避免一次又一次地将数据作为属性来处理。

这种模式降低了角色信息与其他实体数据的耦合性。这在语义上意味着你意识到可以在相关的实体外定义上下文的角色信息。先前模式是将角色信息作为实体中的属性来维护。这种模式提出了一种不同的语义观点，例如角色是一条独立的信息，可以凭借其自身的能力作为一个实体，然后与另一个实体相关联。我们认为这两种观点都一定的道理，因为使用哪一种模式取决于特定情况。这两种观点没有哪一种是完全正确的，也没有哪一种是完全错误的。

### 为什么我们需要这种模式？

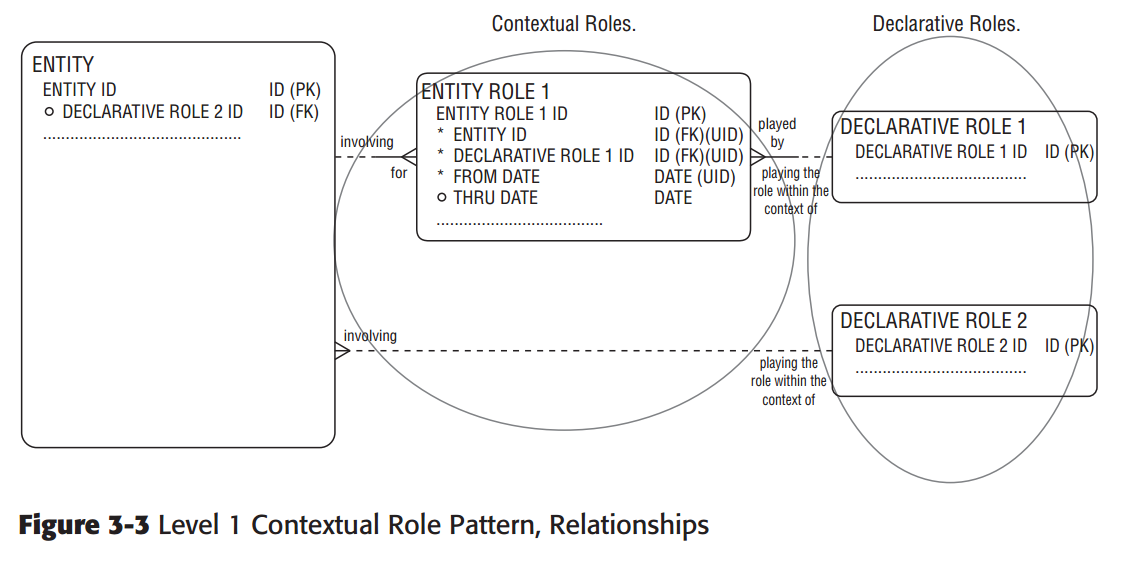
这种模式摒除了将角色数据作为不同的上下文角色的属性这样的需求。该模式通过一个上下文角色实体，将声明角色与一个实体关联。上下文角色实体表示在该实体的上下文中声明角色（人或组织）所扮演的角色.例如，**物流服务提供商**（声明角色）可以是某次**装运**的**装运承运人**（上下文角色）。

这个一级模式与前一个模式相比，更为泛化一些。在角色自身的实体中明确的展示了这些角色，这种模式的优势是可以只维护一次角色信息，然后维护与某一特定实体相关的特定角色实例。换句话说，该模式依赖于前一章节中（第二章）所描述的声明角色（一级）的应用，然后通过一个上下文角色将这些声明角色关联到这个实体。这种模式的意义在于你使用声明角色来维护角色信息，如名称、信用等级、税务标识或关于该角色的其他信息。然后通过一个上下文角色将声明角色关联到特定实体。例如，你可以声明一个**客户**（声明角色），该**客户**在一个**订单**中可以当作是**运达客户**（上下文角色）。

### 这种模式是如何工作的？

在图3-3你所看到的**实体**，代表上下文角色所参与的业务数据。例如，这个实体可以代表一个**项目**、**订单**、**装运**或任何其他的业务数据。在图的最右边你看到的是两个特定的声明角色（**声明角色1**、**声明角色2**）。他们可以是客户、供应商、和代理商。这些声明角色包含该声明角色的所有的特定属性，如名称或标识。

**注：图3-3中所示的声明角色是使用一级声明角色模式作为模版来创建的。有关一级声明角色的更多详细信息请参照第二章。有些人可能把这种模式当成了二级模式，因为在本章中该模式比其他的一级模式更加泛化；然而，我们之所以选择指定它为一级模式，是因为它使用了一级声明性角色。**



**图3-3**一级上下文角色模式，关系

在这种模式中有两种备选方案可以将上下文角色与相关的实体关联起来。第一，可能声明角色与该角色所参与的实体存在多对多关系。例如，可以有多个项目工作人员参与到一个项目中，而一个项目工作人员可以参与多个项目。如图所示，可以通过添加关联实体**实体角色1**来解决**声明角色1**和**实体**之间的这种多对多关系。图3-3描述的是“每个**声明角色1**可以在一个或多个**实体角色1**的上下文中扮演该角色”以及“每个实体可以包括一个或多个**实体角色1**”。**实体角色1**包含了所有上下文角色实例。

你也可以将上下文当作角色与相关实体之间的关系来处理。这点可以直接从图3-3一对多关系中看出。当你有一个特定的业务规则，指出实体有一个且只能有一个给定角色的实例，这样做是很有用的。图3-3中定义关系为“**每个声明角色2**可以在一个或多个**实体**的上下文内扮演角色，每个实体可以与一个且与扮演一个**声明角色2**相关联”。

如果我们继续前一节所介绍的XYZ公司的案例，你可以看到最初数据专业人士基于一级上下我让你角色模式所制作的图3-2，来展示项目的上下文角色的初始作用域。根据与股东的进一步讨论，XYZ公司决定项目数据事实上比他们原先认为的更加的至关重要，他们需要维护他们各自实体中的每个角色的信息以至于能够以一致的方式只维护一次这些角色的实例。数据专业人士基于备选的一级上下文角色模式，把上下文角色当作关系来处理，制作了图3-4，介绍了声明角色**赞助商**、**工作人员**或**项目负责人**。股东表明**赞助商**、**工作人员**和**项目负责人**是声明性角色，因为无论什么特定项目，XYZ公司都可以给这些角色指派人员或组织。换句话说，不管XYZ公司从事哪个特定项目，**赞助商**、**工作人员**和**项目负责人**在企业整体范围内具有意义。此外，股东还表示**赞助商**、**工作人员**和**项目负责人**在一个项目的上下文中扮演了非常特定的角色，即，扮演了某一项目的**项目赞助商**、**项目工作人员**和**项目负责人**。

**图3-4**一级上下文角色模式，关系的应用示例

在图3-4中，你所看到的**赞助商**、**工作人员**和**项目负责人**均被列为可以参与到一个项目当中的角色。每个不同的声明角色实体都包含一个名称（如赞助商名称、工作人员的名字、工作人员的姓氏、负责人的名字和负责人的姓氏）属性，该名称属性反映了人或组织所扮演的角色。数据专业人员根据项目管理者的要求给项目实体添加了一些重要的属性，即计划开始时间，预计费时。

所以图3-4中所介绍的实体哪些是上下文角色实体？对于这个问题的答案我们需要参考上下文角色的定义。这里所介绍的是一个实例，在该实例中人或组织作为**赞助商**、**工作人员**和**项目负责人**来处理，并体现了他们在项目的背景环境中关系、行动或活动。

上下文角色**项目赞助商**和**项目工作人员**反映了这个关联。他们定义了人员或组织在**项目**内的上下文范围。但是项目负责人这一上下文角色是在何处参与到**项目**中的呢？关系定义了这种关联“每个**项目负责人**可以负责一个或多个项目，而每个项目可以有一个且只能由一个项目负责人负责”。如果你查看下表3-2，你可以看到这张表包含四个不同的**项目**，项目名称分别是“客户主数据项目”、“人力资源数据库”、“信息架构标准”和“萨班斯-噢克斯利法案报告”。“萨班斯-噢克斯利法案报告”项目不止一个赞助商，“信息技术部门”和“审计部门”。并且赞助商“信息技术部门”资助了“萨班斯-噢克斯利法案报告”和“信息架构标准”这两个项目。项目可以具有多个赞助商以及一个赞助商可以支持多个项目是很常见的。**项目赞助商**上下文角色支持这个需求。

表3-3阐释了非常常见的情形，一个项目可以有很多的工作人员而一个工作人员可以从事许多不同的项目。正如前面所示的四个项目。每个工作人员的实例可以在不同的时间段内与项目实例相关联。起始日期和终止日期定义了**项目工作人员**上下文角色的实例的有效时间范围。例如，对于项目“客户主数据项目”，**工作人员**“Neena Davies”的从2007年10月10日开始一直持续到2009年1月18日，在此期间在项目中扮演了工作人员的角色。

表3-4阐释了**项目**与**项目负责人**的特定关系。表3-4包含了与前一张表相同的项目。“每个项目可以由一个且只能由一个项目负责人来负责”。在这个案例中“Una Corr”负责“客户主数据项目”和“人力资源数据库”这两个项目。“Vinnie Chintappaly”负责“数据架构标准”项目。“萨班斯-噢克斯利法案报告”还没有负责人，因为没有给该角色指派负责人。

**表3-2**一级上下文角色，项目、赞助商和项目赞助商的示例

**表3-3** 一级上下文角色，项目、项目工作人员和工作人员的示例

**表3-4**一级上下文角色，项目、项目负责人的示例

### 何时应当使用这种模式？

我们在以下时刻使用这种模式：

* **在相同的个人或组织扮演了（声明）角色且可以多次扮演同类型角色的时候。**例如，一个雇员从事于多个项目或一个项目有多个雇员来做。
* **需要严格特定的声明业务规则时。**例如XYZ公司表明一个项目只能由一个人负责。这种模式介绍和维护了这一规则。
* **为了使数据模型更加易于理解需要特定的声明角色时**。我们认为这种模式不仅有效的展示了某个特定数据领域的一组上下文角色，而且还有这些角色与他们相关实体的关系。这种模式可以用作为作用域声明的一部分用来阐释数据需求。

### 该模式的弱点是什么？

这种模式的缺点如下：

* **这种模式重复了声明角色的同一当事人信息。**例如，如果某人既是赞助商也是工作人员，那么两个实体都维护了这个人的信息。从某种角度来看，这种模式只能解决冗余问题—如果当事人在不同的项目中扮演了相同的角色，名称（和关于这个当事人的任何其他数据）只被维护一次，但是如果当事人扮演了两个或多个角色，那么当事人的信息就会被重复。
* **这种模式可能不是最适合于动态环境的模式，或者不太适合于不太熟悉的环境。**如果发掘或添加新的角色，那么模型也需要添加新的实体和关系。例如，在很多情形中，随时间推移，流程发生变化，要求添加新的角色到数据模型。比如说可以为一个项目添加一个质量保证经理或一个项目主管，且在这种模式中，这样的变化将要求添加新的角色。
* **如果业务规则改变，模型也需要改变。**例如，如果为了具有多个项目负责人而更改业务规则，那么模型（和任何基于这个模型的底层数据库设计）也需要做相应的改变。

### 简介

本节中你所看到的是一级上下文角色模式，将上下文角色作为关系来呈现，对于在定义明确的静态主题域中展示所有角色的作用域和数据需求，这个是非常有用的模式。此模式的意义在于上下文角色被当作声明角色与相关实体之间部分关系来处理。如图3-4所示的三个上下文角色—其中两个作为关联实体**项目赞助商**和**项目工作人员**，还有一个作为关系“每个项目负责人可以负责一个或多个项目而每个项目可以有一个且只能有一个项目负责人”。

这种模式解决了重复数据的信息冗余问题，如图3-1和图3-2所示的实体属性，但是它并未解决在角色信息当中重复了当事人信息这个问题，因为同一当事人可以扮演多个角色。这是个特定模型，应当在静态的定义明确的情况下使用。如果这个环境是变化的且不太了解的，用这种模式可能会出现问题，因为发掘新的角色实体和关系时，需要添加这些新发现的实体和关系。

最后，这种模式有助于（和不利于）在数据模型中强制执行业务规则。你可以在图3-4中看到该模式执行了一个*项目*只能有一个**项目负责人**这条规则。但是，如果有业务规则发生变化或需要新的角色，模型将需要做相应的改变。

## 二级上下文角色模式

前一个上下文角色模式的一个问题是可能冗余地重复人员和组织的信息。当数据专业人员是为一个非常特定的案例建模时，或只希望使用这个模型来展示调查研究中的上下文角色的作用域和数据需求时。如果实现了这种模式，冗余地捕获人员和组织的信息可能会导致重大的数据质量问题。在某一特定的主题领域中，它可能不是实现上下文角色的最佳实现方案。

### 为何我们需要这种模式？

二级上下文角色模式可以将上下文角色关联到现存的**当事人角色**信息。一个运达客户可以是你正在发送订单货物的目标公司；收款客户可以是一个不同的组织，你可以为该组织开具装运发票。这两个组织可能都已存在于你的账单和装运系统中了。换句话说，对于这些客户，客户声明角色可能已经存在了。7不用再次创建关于客户的信息，你可以将客户所扮演的上下文角色同保存在当事人角色中的客户信息关联起来。因此你创建一次客户（或声明一个当事人作为客户），然后使用这个角色并根据上下文将它同另一个实体，比如说装运或发票关联起来。或者视情况而定，你也可以选择声明**收款客户**角色和**运达客户**角色来展示这些是当事人的有效角色，然后使用本节中的上下文角色模式，将这些角色关联到另一个实体，例如，订单。

**注：当事人和当事人角色的概念贯穿于整个通用数据模型，而在数据模型资源手册的第一卷以及本书的第二章节中都对这些概念做了非常详细的讨论。基本上来说，当事人既可以是人也可以是组织（在某些情况下，可能是自动化代理商或者甚至是某些类型企业中的动物），但是只能是其中之一。对整个企业而言当事人角色是指当事人所扮演的声明性角色。**

### 这种模式是如何工作的？

在图3-5中你可以看到一种更加灵活的模型，该模型将前一种模式所介绍的所有角色作为**当事人角色**实体的子类来处理。因此，这种模式为不再重建同一当事人的信息提供了一种方法。例如，你可以维护一次与**当事人**相关的人员名称（和任何其他数据信息），然后展示该当事人能够扮演多个角色，例如客户、供应商、或物流服务提供商。但是要意识到声明的这些角色不是上下文角色—他们是为整个企业而定义的角色，而不是为某个特定的业务活动而定义的。

**当事人角色**包含可以提供所有声明角色所需的通用属性和关系，例如起始日期和终止日期表明了声明角色何时开始有效和何时该角色实例不再有效。在这个模型图中你可以看到“每个**当事人**可以扮演一个或多个**当事人角色**而每个**当事人角色**必须属于一个且仅属于一个**当事人**”。**当事人**包含人员或组织的通用信息，比如说现名称属性（名字、姓氏和名称）。

除了**角色类型**实体还展现了**当事人角色**的子类（**声明性角色1,2**），在某种程度上，**角色类型**和**当事人角色**的子类是在为相同类型的信息建模，因为**角色类型**实例一般与**当事人角色**的子类相对应。之所以为子类和角色类型都建模是因为可能有数据与当事人角色的特定子类相关，例如，雇员的薪资，也可能有数据与角色类型相关，例如，分配给各类角色的授权和权限。

**图3-5 二级上下文角色模式**

实体指与上下文信息相关的信息和数据，而这些信息和数据是数据建模人员所要处理的。这类数据的示例是**项目**（如图3-6）、**订单**、**装运**和**发票**。

图3-5展示了两个不同的上下文角色。**特定上下文角色**捕获的是担当**声明角色1**的当事人所扮演的上下文角色。换而言之，“每个当事人可以担当一个或多个当事人角色”，在**声明角色1** 这个案例中“每个**声明角色1**可以在一个或多个特定上下文角色中扮演角色”。例如，**当事人**可以扮演**客户**角色（声明角色），而**客户**可以扮演**订单收款客户**、**订单运达客户**等上下文角色。

图3-5所介绍的第二种上下文角色存在于**声明角色2**和**实体**之间的关系中。“每个**声明角色2**可以在一个或多个**实体**的上下文中扮演角色”。例如，独资经营者可能是唯一的**支票账户**的“签字人”。

**图3-6** 二级上下文角色模式应用示例

这里很重要的一点是在这种模式中有两种不同的方法可以处理上下文角色。如果**实体**只能有一个当事人扮演与**声明角色2**相关的上下文角色，在一对多关系捕获上下文角色。如果实体可以有多个当事人扮演**声明角色1**，那么这个上下文角色被存储在**特定上下文角色**中。

图3-6展示了这种模式是怎么为捕获特定业务规则提供功能强大的方法。XYZ公司的项目管理人员描述了一些业务规则，他们想要看到这些业务规则作为他们的模型的一部分。他们表明“只有被声明为**工作人员**的当事人才可以成为**项目工作人员**，一个项目可以有多个**项目工作人员**”。他们还表明“一个**工作人员**可以在同一时间参与多个不同的**项目**”。

此外，他们还说“只有被声明为**项目负责人**的当事人才能负责一个**项目**，一个**项目**只能有一个**项目负责人**”。在**当事人**可以担当**项目负责人**和设置成**当事人角色**前，**当事人**必须通过一整套的培训课程。

这是为了确保公司如何保持所交付项目的一致性。

最后，根据数据专业人士的一些提示，项目管理人员表示他们认为只有一个被设置为**赞助商**的当事人可以作为**项目赞助商**参与项目。

这对公司而言是个新的概念，但是有人认为它可以帮助管理**项目**所涉及的角色。基于这个信息，创建了**当事人角色**至不同的上下文角色和**项目**的关系。在图3-6中你可以看到“每个**项目赞助商**必须由一个且只能由一个**赞助商**扮演”。同时，“每个**项目**可以由一个或多个**项目赞助商**来资助”。这表明只有扮演了**赞助商**角色的**当事人**才可以作为**项目赞助商**参与到**项目**当中，且一个**项目**可以有多个**赞助商**。第二，你可以看到“每个**项目工作人员**必须由一个且只能由一个**工作人员**来扮演”，“必须给每个**项目**指派一个或多个**项目工作人员**”。这意味着通过关联实体-**项目工作人员**，很多项目能够有很多不同的**当事人**担当**工作人员**。最后，“每个项目可以有一个且只能有一个**项目负责人**”。

表3-5包含了所有的**当事人**和当事人所扮演的**当事人角色**。认定了11个当事人：“Neena Davies”、 “Una Corr”、 “Paul Lane”、“Vinnie Chintappaly”、“Master Data Management”、“Human Resources Dept.”、“Yi Lan Tsang”、“Information Technology Dept.”、 “Steve Toland”、“John Teevan” 和 “Audit Dept.”你可以从所列的**当事人**中看出，**当事人**既包括组织也包含人。

每个**当事人**可以扮演一个或多个**当事人角色**。**当事人**“Una Corr”扮演了“项目负责人”和“工作人员”这两个角色。根据**当事人**所扮演的**当事人角色**，每个**当事人**反应了**当事人角色**属性的特定信息。例如，“Neena Davie”扮演了**工作人员**这一**当事人角色**，并具有一个工号“4412”。**当事人**“Audit Dept”扮演了具有“A++”等级的**供应商**这一**当事人角色**，这表示它具有资助项目的良好记录。**当事人**“Vinnie Chintappaly”扮演了**项目负责人**这一**当事人角色**，具有“3级领导培训水平”，这个指明了扮演项目负责人的这个人的当前能力。

**表3-5**含有当事人角色子类、当事人、角色类型和当事人角色的二级声明角色模式示例

**表3-6**含有当事人角色子类、当事人、角色类型、项目、赞助商和项目赞助商的二级上下文角色模式示例

表3-6中的**赞助商**扮演了**项目**中的**项目赞助商**这一上下文角色。表3-6还包含了四个不同**项目**的项目名称，分别是“客户主数据项目”、“人力资源数据库”、“信息架构标准”和 “萨班斯-噢克斯利法案报告”。“萨班斯-噢克斯利法案报告”的多个实例说明一个**项目**可以有多个**赞助商**。这种模式支持XYZ公司给数据专业人士所定制的业务规则，即一个项目可以由多个当事人来资助。此外还应注意到“信息技术部门（Information Technology Dept.）”赞助“萨班斯-噢克斯利法案报告”和“信息架构标准”这两个项目。这说明了一个赞助商可以赞助多个项目。

表3-7包含了**项目**和被指派到**项目**的**工作人员**的示例。每个**工作人员**可以在不同的时间段与项目相关联，即他们被分配到该项目的时间段—起始日期和终止日期表示的是这个人从事该项目的时间段。例如，“人力资源数据库”项目中有两个**工作人员**扮演了**项目工作人员**这一角色。“Neena Davies”从“2009年1月1日”开始从事“客户主数据项目”直到终止日期的第二天结束。“Paul Lane”于“2007年1月28日”被指派到“人力资源数据库”项目，并于“2009年9月4日”结束任命。

正如前面所示表3-8包含了四种不同**项目**，其项目名称分别是“客户主数据项目”、“人力资源数据库”、“信息架构标准”和“萨班斯-噢克斯利法案报告”。业务表明每个**项目**可以有一个且只能有一个**项目负责人**。在“Una Corr”这个案例中，负责“客户主数据项目”和“人力资源数据库”这两个项目，而“Vinnie Chintappaly”负责项目“信息架构标准”。“萨班斯-噢克斯利法案报告”**项目**没有负责人；**项目**与**项目负责人**之间的关系不是强制性的。XYZ公司告诉数据专业人士他们不需要保存任何关于项目负责人被指派到项目的日期时间信息。由于以上原因在项目中没有给出起始日期和终止日期（负责项目的起始时间和负责项目的终止日期）。

**注：前一张表有起始和终止日期，但是这张表没有，因为从项目负责人到项目是一对多的关系。考虑到以往情况，我们即可以给项目添加项目负责的起始日期和终止日期的属性也可以将这种关系更改成多对多关系以满足这个需求。XYZ公司决定它不需要这样的功能。**

**表3-7**含有当事人角色子类、当事人、角色类型、项目和工作人员以及项目工作人员的二级上下文角色模式

**表3-8**含有当事人角色子类、当事人、角色类型、项目和项目负责人的二级上下文角色示例

### 何时应当使用这种模式？

我们在以下时刻使用这种数据模型模式：

* **为了能够更好的理解信息需求，需要建立特定的模型。**
* **认同使用当事人角色的概念来展示相同的当事人可以扮演多个角色，然后在另一个实体的上下文中应用每个角色。**
* **决定整合声明角色和上下文角色的应用时**。根据不同的原因这两种类型的角色都很重要。声明角色定义了对于整个企业而言当事人是谁。上下文角色反映的是当事人是如何参与到其他实体中的，比如说那些代表商业活动的实体。他们是两种截然不同的概念，但是却紧密相关。通常可以声明尚未参与一些业务活动的当事人。例如，创建一个新职员，但是还没有将其指派到某一特定项目或是尚未将其与其他的实体相关联。这种模式展示了这两种模式可以如何相互支持彼此。
* **需要在数据模型中指定业务规则。**如图3-6的示例所示，项目可以有一个且只能有一个项目负责人。该数据模型中的关系特定地支持这一规则。
* **需要通过使用当事人和当事人角色中的通用属性来避免重复属性和关系**

### 该模式的弱点是什么？

这种模式的缺点如下：

* **添加当事人角色概念相当于你添加了一个泛化层次，而这可能很难获得一些企业的认同。**有些企业不认为**当事人角色**与**当事人**是不同的概念。在多数不同的企业中盛行的观点是“你就是指你的所作所为”。这种观点不是错的。它是一个有效的备选方案，将在“仅作为备选模式的维二级上下文角色模式，当事人”中探讨这一观点。
* **这种模式强制XYZ公司使用声明性角色。**换句话说，如果XYZ公司发掘了一个新的上下文角色，比如说“项目风险总监”，那么他们将不得不添加一个声明角色。如果没有定义支持这个角色的声明角色，会发生什么呢？XYZ公司将不得不创建这个新声明角色和新的上下文角色。

**注：可以这么说，刚才所提到的缺点其实也是优点。一些企业具有严格的数据政策它可能强制所有的上下文角色使用声明角色。此外，可以拓展这种模式，可以仅使用上下文角色且只需将待处理的实体关联到角色实体，而不是将其声明为声明性角色。例如，可以有一个数据录入人员角色实体，且该实体不是当事人角色的子类。由于任何人都可以为某一特定事务执行此操作，所以如果企业不想必须声明“数据录入人员”这样的角色，可以使用这种方式。**

* **这种模式仍存在灵活性问题，随着时间的推移，当流程发生变化时，添加了新角色，就可能需要更改底层的数据模型。**假如后来需要添加一个像质量保证管理员这样的角色呢？因此如果你基于这种模式实现了数据库设计，可能要为每个新出现的角色添加表而付出很大的代价。
* **另一个灵活性上的缺点是如果随着时间的推移，基数发生变化，数据模型也需做相应的更改。**例如，如果x z y公司决定在同一时间多个当事人可以成为一个**项目**的**项目负责人**，这就需要做相应的改变了。
* **这种模式没有将仅限于人和仅限于组织的角色区分开来。**例如，在这种模式中它并不禁止组织作为项目工作人员。这个可以通过修改当事人角色实体来解决，使当事人角色具有子类**人员角色**和**组织角色**。
* **当从角色到当事人存在一对多的关系时，这种模式不能对谁在何时扮演了什么样的角色的历史记录做出说明。**例如，假如随着时间的推移项目负责人发生了变化会怎么样呢？在这种模式中，以前的项目负责人将只会被覆盖，除非将角色到当事人之间的关系更改成多对多关系。

**注：可以说， 历史记录应当由数据模型之外的其他技术手段来处理，比如说数据库的审计功能或使用历史记录快照表。**

### 简介

本节中我们介绍了一种特定的模式，对于关联了一些其他实体的当事人，很多企业可以使用这种模式来为该当事人所扮演的不同上下文角色建模。重要的是这种模式使用声明角色，当事人的声明性角色可以作为创建上下文角色的基础，这些上下文角色是关联到当事人角色的，而该当事人角色也参与到另一个实体当中，例如**项目**、**订单**和**装运**。

尽管在当事人和当事人角色模式中声明角色有益处，但是对于某些企业而言，这个结构可能太过笼统，或者它不能满足企业的语义需求，这些企业认为当事人所扮演的角色是该当事人自身的一部分。

它也可能对于某些企业来讲太过“苛刻”。换而言之，尽管可以拓展这种模式，也能够添加不作为当事人角色子类的上下文角色实体，但是他们不认为有必要让每个上下文角色都具有一个声明角色。

这种方法有助于捕获数据模型中的特定业务规则。但它的特定性质也有不利的一面，在不改变数据模型的前提下，它不能够十分的灵活地适应变化，如新角色或技术发生变化。这意味着如果实现了这种数据模型，可能会付出很大的代价，因为当业务规则发生变化时，随后数据结构可能也需做相应的更改。

## 只作为为备选模式的二级上下文角色模式，当事人

在有些情况下，将上下文角色直接与**当事人**关联而不是与**当事人角色**相关联是有意义的，因为为当事人所扮演的角色设置一个**当事人角色**可能会不是很恰当。例如，在实体**发票**中有一个开票方和一个收票方，所以先声明“开票方”和“收票方”角色，然后将这些角色关联到发票，这样做可能没什么意义。相反，通过两个关系将发票与**当事人**相关联可能更有意义，一个是**当事人**与开票方的关系，另一个是**当事人**与收票方的关系。

此外，有些企业（或数据建模专业人员）不使用当事人角色概念。有些企业认为将上下文角色与声明角色绑定太过繁琐。因此，如果企业以这种 来看待他们的业务，数据模型可能需要支持这种另类的角度。

### 为何我们需要这种模式？

这种模式支持上下文角色的需求和他们与人员和组织直接关联的方式。一个**当事人**可以是**人**或**组织**，在**订单**的上下文中当事人可以扮演上下文角色**订单收款客户**。

还需要保持所有角色的**当事人**信息的一致。通过将上下文角色直接关联到**当事人**，这种模式可以维护一次且只用维护一次**当事人**信息。

**注：需要注意的是无论是直接关联到当事人的关系（正如此种模式）还是通过当事人角色来建立的关系（正如前一种模式所示），当事人的信息只存一次。**

在这种模式中，不是必须要“声明”角色。 由于这个特点，这种模式更加的灵活，因为任何当事人都可以扮演上下文角色。例如，

可以指定任何当事人作为项目赞助商而不仅限于那些被声明作赞助商的当事人，这是个可接受的业务规则。

### 这种模式是如何进行工作的？

在前一种模式中，上下文角色不是直接与**当事人**相关联的，而是通过**当事人角色**间接关联到**当事人**的。如图3-7所示，这种模式使用的是**当事人**直接关联到**上下文角色**，然后将**上下文角色**关联到**实体**或者直接将**当事人**关联到**实体**。这意味着什么呢？这种模式认识到可能会存在这样的人员或组织实例：参与了没有特别指定声明角色的业务活动、事务或实体。或者，参与了与声明性角色相关的活动、事务或实体。但是有些使用声明角色的例外情况。例如，你可以有一个**卖主**角色，但是偶尔，你可能从一个不在卖主清单上的**当事人**那儿购买。这开创了二级上下文角色所没有的灵活性和简洁性。这种模式的缺点是，有些企业可能认为这种模式在语义上对他们而言太过宽泛。换而言之，他们可能认为认为当时人只能参加一些具有特定声明的企业级角色的业务活动，也就是前一种模式所示的当事人的声明角色。

类似于前一种实体，实体代表的是任何实体，比如说当事人（人员或组织）所参与的业务活动或事务。**实体**可以使一个**订单**、**发票**或**装运**。**当事人**可以直接与**实体**关联或可以通过上下文角色来关联。

同一**当事人**可以以多种不同的方式参与到一个**实体**中。例如，一个**人员**可以是**订单**的运达客户和收款客户。该**订单**也可以有多个**当事人**与它相关联，例如订单的付款**人员**和订单的签收**组织**。上下文角色实体包含了这些上下文角色的不同实例，每个上下文角色实体（上下文角色1,2,……）代表了了在这些上下文中当事人所扮演的不同类型的角色。这一点可从关系当中看出，关系表明“当事人可以在一个或多个**上下文角色1**的上下文中扮演该角色”且“每个**实体**可以扮演一个或多个**上下文角色 1**”。实体到上下文角色的关系正说明了在同一时间**实体**可以具有多个扮演这一特定角色**当事人**。例如一个**项目**可以具有多个**赞助商**。

如果**当事人**与**实体**的关系的本质是一对多关系，那么也可以把上下文角色建模成**当事人**到**实体**的直接关系。图3-7中直接关系表明了“每个**实体**只能有一个当事人”，图中还包含上下文角色，该上下文角色从当事人出发到其相关的商务活动或事务（也就是实体）。使用这种方法展示了特定的业务规则，例如只能由一个当事人扮演每个实体实例的特定角色。

图3-7仅作为备选模式的二级上下文角色模式，当事人

需要注意的是**人员**和**上下文角色2**之间的关系。这种关系展示了这种模型支持某些特定的业务规则，**人员**可以是唯一类型的**当事人**，而该当事人可以扮演某个特定类型的上下文角色。例如，在图3-8中，你可以看到类似类型的关系，“**人员**可以被任命为一个或多个**项目工作人员**。”这样做很有意义，因为通常情况下只有人可以被指定为项目的工作人员。

通过拓展前一节中所示的案例，可以对这种模式做进一步的解释。XYZ公司可以选择通过直接关联到**当事人**而非使用当事人角色来为角色建模。因此，数据专业人士制作了图3-8，作为适用于企业的上下文角色建模的备选方案。每个上下文的角色—**项目赞助商**、**项目工作人员**和**项目负责人**可以根据关系来处理“每个人员可以负责一个或多个项目，而每个项目可以由一个且只能由一个人来负责”。这些关系与图3-6中的关系大致一样，除了他们是直接从**当事人**（或者**人员**和**组织**）而非**当事人角色**出发的。由于随着时间的推移可能还有其他的关系，所以项目负责当事人标识属性为项目中的当事人标识外键提供了修饰符，我们发现将当事人与其他实体相关联时，这是一个很有用的约定。

**图3-8** 只作为备选模式的二级上下文角色模式的应用实例，当事人

表3-9包含项目和专门针对于项目赞助商的相关上下文角色的示例。表3-9包括四个不同的**项目**实例，其项目名称分别是“客户主数据项目”、“人力资源数据库”、“信息架构标准”和“萨班斯-奥克斯利法案报告”，“萨班斯-奥克斯利法案报告”项目的多个实例展示了两个不同的**当事人**充当了**项目赞助商**—“审计部门”和“信息技术部门”。**项目赞助商**这一上下文角色也介绍了常见的情况，一个当事人可以赞助多个项目。表3-9展示了这样一个示例，**赞助商**“信息技术部门”赞助了“萨班斯-奥克斯利法案报告”和“信息架构标准”这两个项目。

**表3-9** 含有当事人、项目和项目赞助商的二级上下文角色示例 只作为备选模式

**表3-10** 含有当事人、项目和项目工作人员的二级上下文角色示例，只作为备选模式

**表3-10**包含了项目和从事于某个项目的工作人员的有关上下文角色的示例。如今很多人在他们的日常工作中从事多个不同的项目。在下面的示例中你可以看到“Neena Davies”在2008年1月1号至2008年1月18号这期间被指派到“客户主数据项目”和“人力资源数据库”这两个项目当中。

一个**人员**也可以在不同的时间段与**项目**建立关系。也就是他们被指派到该项目的时间段。起始日期和终止日期规定了项目工作人员上下文角色实例的有效时间范围。例如，对于项目“客户主数据项目”，Neena Davies 从“2007年10月10号”开始从事该项目，一直到“2009年1月18号”才结束。对于同一个项目，“Yi Lan Tsang”是从“2008年2月15号”开始的，并且认为他此后仍然从事于该项目。

**项目**“人力资源数据库”中，有两个人员扮演了**项目工作人员**角色。“Neena Davies”于“2009年1月1日”开始“客户主数据项目”，直至终止日期的后一天结束。“Paul Lane”于“2009年1月28日”被指派到“人力资源数据库”项目中，直至“2009年9月4日”任命结束。

表3-11中包含三个不同的**项目**，其项目名称分别是“客户主数据项目”、“人力资源数据库”和“信息架构标准”。每个**项目**可以有一个且只能有一个**项目负责人**来负责该**项目**，而一个**项目负责人**可以负责一个或多个**项目**。在这一案例中，“Una Corr”负责了“客户主数据项目”和“人力资源数据库”这两个项目，而“Vinnie Chintappaly”则负责了项目“信息架构标准”。正如我们早前所述，项目负责人总是一个人。因此，我们获取了人员的名字和姓氏。

表3-11 含有当事人们、当事人、项目和项目负责人的二级上下文角色示例，只作备选模式。

### 何时应当使用这种模式？

我们在以下情形中使用这种数据模型：

* **企业不使用或不赞同那些使用了当事人和当事人角色概念的声明性角色，相反，当当事人角色出现在其他实体中时，就定义当事人角色。—例如，项目、订单或发票的角色。**并不是必须要为参与到某些业务活动中的**当事人**特地定义其声明性角色。这种模式具有更佳的灵活性，因为任何当事人都可以扮演上下文角色，因此没有必要将该上下文角色关联到一个声明角色。例如，将任何当事人定为当事人供应商就可能是一个可接受的业务规则。
* **直接与当事人建立关系来代替使用当事人角色间接建立关系更加有意义。**例如，在**支付**实体中你 可以看到从**支付**到**当事人**有两个关系，即**当事人**可以收到和发出支付。在这个案例中，你可能不想设置声明性角色为款项的“接收者”和“发出者”。因此，在同一模型中，可能有些上下文角色可以关联到当事人角色，其他的上下文角色可以与当事人建立关系，至于使用哪种关联方式视情况而定。
* **当你想指明数据模型的业务规则时。**从图3-8中的示例，你可以看到一个**项目**有一个且只能有一个**当事人**来负责它。数据模型的关系明确支持这个业务规则。前一种模式也支持。
* **需要在当事人实体中只维护一次人员或组织的信息。**前一种模式亦是如此。

### 该模式的缺点是什么？

这种模式的缺点如下：

* **这种模式仍然存在灵活性问题。**因为这模式与角色具有特定的关系，随着时间的推移，如果添加了上下文角色、角色的基数发生改变或需要关于谁扮演了什么角色的历史记录，将需要更改模型。例如，需要一个新的项目角色“项目顾问”。另一个示例是业务规则可能发生了改变，现在可以由两个项目负责人负责一个项目。所有的这些变化都将要求更改数据模型，如果此数据模型是数据库实现的基础，那么更改数据模型将是个问题，并且为了适应这些需求，需要改变底层的表，所以处理这些变化的流程可能要付出很高的代价。
* **有些企业可能喜欢或需要将新的上下文角色强制链接到声明性角色的语义上的严密性。**换句话说，有些企业可能会表明每个上下文角色都必须由一个企业范围的角色（声明性角色）来支持。这样的一个原因是添加角色过程将会更加一致，此过程可以包括在使用角色（添加上下文角色）之前，要事先设置该角色（添加声明性角色）。

### 简介

这一节中我们介绍了一种特殊的模式，多数企业可以使用这个模式来为当事人所扮演的不同的上下文角色建立模型。这种模式的意义在于**当事人**是直接与另一个实体的上下文角色相关联的，例如**项目**、**订单**或**装运**。这个模式允许任何**当事人**扮演另一个实体中的角色比如说商业事务或活动，且不用特别地将该角色声明为声明性角色。这个方法适合于不同意使用**当事人角色**方法的企业，或可以提供替换方案的企业，而该方案允许上下文角色既能关联到**当事人**也可以关联到**当事人角色**的，至于关联到哪个实体视具体情况而定。

这种模式实现特定的业务规则也很容易，例如，只允许**人员**成为**项目工作人员**，或如图3-8所示的一个**项目**只能由一个人负责。

与前一种模式类似，这种模式也存在一定的灵活性问题。随着时间的推移，当发现或添加新角色时，当业务规则和流程发生变化时，他们都要作为上下文角色实体添加到模型中，所以就要求更改模型。业务规则可能还要更改关系基数，这样也可能需要更改模型。这种模式可能不太适合以下企业：企业的环境是动态的或围绕他们的上下文角色或业务规则很可能随着时间的推移发生改变。有些企业可能需要语义上的严密性，强制上下文角色与声明角色有关联，换句话说，就是强制每个特定的上下文角色都有一个企业范围的角色（声明性角色）来支持它。

## 三级上下文角色模式

到目前为止，所有的模式均支持定义明确的静态的上下文角色。例如，在先前的XYZ公司的案例中所认定的三个不同的角色，**项目负责人**、**项目工作人员**和**项目赞助商**。在商业发展迅速的今天，**当事人**所扮演的上下文角色可以发生变化或当新业务情形出现时，发现新角色或废止老角色，这是很常见的。通常人们对于一个新出现的业务领域还不是十分的了解，可能还不能全面认定当事人所扮演的角色。

### 为什么我们需要这种模式？

三级上下文角色模式支持非常灵活的解决方案的需求，能够允许动态的添加和改变（和废止）某个实体的上下文角色的属性。这种模式不假设所有的上下文角色与其他实体相关，比如说前面已提出的商业事务或活动。并且上下文角色可以发生变化；新角色类型可以接管老角色类型以反映业务流程和实践的变化。这种模式支持这些需求。

### 这种模式是如何进行工作的？

图3-9介绍了这种模式是如何构建的。在前一模式中，明确定义了每个上下文角色，如图3-7.在这种模式中所有的上下文角色是通过**上下文角色**实体来维护的。这个上下文角色实体充当了**实体**、**当事人**和**角色类型**之间的三叉路口，允许任何当事人扮演任何类型的角色，并且可以多次扮演该角色。因此，特定实体如**订单**可以有很多的当事人，该当事人可以扮演很多不同类型的角色，且可以扮演很多次（起始日期和终止日期不同），只需添加额外的**上下文角色**实例，这些实例将当事人与该实体的**角色类型**关联了起来。

**图3-9** 三级上下文角色模式

**注：此处的角色类型是相同的角色类型，可用于当事人角色的分类。换句话说，角色类型支持声明角色（如第2章节所示）与上下文角色的分类。如果你想将声明类型的角色与上下文类型的角色区分开来，你可以创建两个角色类型的子类，分别称为上下文角色和声明性角色，或者你可以为当事人角色类型（为声明性角色）创建角色类型子类（例如，订单角色类型、装运角色类型、工作计划角色类型等等）。**

**注：上下文角色与声明性角色之间可能存在关系（和关于这些关联的规则）。例如，“供应商（声明角色）”可能是唯一的一类能够扮演“医疗服务商（上下文角色）”这一角色类型的声明性角色。我们将在第4章的规则的三级递归模式中详细介绍同一实体的不同实例之间的这些关系。**

当业务流程的成熟时，我们可以使用这种模式轻松地添加、废除或更改不同类型的上下文角色。例如，随着美国Sarbanes-Oxley 法案的出现，为了支持这一法案的需求，创建了一些监督金融交易的新类型的角色，并且废除了其他类型的角色。例如，会计事务可能需要额外的角色，如“质量保证审核人”、“审计员”、“法务主管”等等。**角色类型**允许你在不更改模式的前提下添加和删除不同类型的**上下文角色**。

正如先前的模式，**实体**代表具有重要意义的事物，它有一组业务数据，例如需要建立模型的经济业务或活动。**当事人**是指参加到商业活动当中的**人员**或**组织**。“每个**当事人**可以在一个或多个上下文角色的上下文中扮演角色”。为了支持维护角色变更历史记录这一业务需求，上下文角色实体包含起始日期和终止日期。最后“每个角色类型可以是对一个或多个上下文角色的描述”。换句话说，通过扮演不同角色类型的上下文角色，当事人可能与实体建立了千丝万缕的关系。

如果我们拓展下前一节的案例，你可以了解到数据专业人士提出了一种备选模式，因为他角色XYZ公司具有动态的业务环境。基于与主要管理人员的会晤和三级上下文角色模式的应用，他制作了图3-10。

图3-10说明了这种模式的威力。**项目**实体包含标识**项目**实例（项目名称）的属性，和它的开始（预定起始日期）以及它持续的时间（预计耗时）。通过**当事人角色**，XYZ公司的每个当事人可以以某种形式与项目的实例建立或不建立关联。

在先前的模式中，我们使用了**项目工作人员**、**项目赞助商**—特定的上下文角色（如图3-8所示），和支持项目领导层角色的关系。在本模式中这些上下文角色均由**当事人**、**项目角色**和**角色类型**实体支持。

图3-10 三级上下文角色模式应用范例

**注：如果你想模型更加泛化些，你可以使用工作计划来替换项目。这样同一模型将不仅能够维护项目，而且还能够维护计划、活动、任务和任何其他类型的工作计划。5**

在图3-9（和图3-10）中你可以看到角色类型实体具有递归关系，此关系表明“每个**角色类型**都可以被进一步的分解成一个或多个**角色类型**并且这些**角色类型**可以在一个且只能在一个**角色类型**之内”。这种结构允许你对角色类型进行分类或组织。例如，你可以将某些角色分成仅限于人的角色，而其的角色归类于当事人角色，或二者选其一，你可以将某些角色类型实例归类到角色类型实例内，如“当事人角色类型”（声明性角色）、“订单角色类型”（可以与订单相关联的上下文角色）、“装运角色类型”（可以与装运相关联的上下文角色）等。如果需要对具有很多上级角色的角色进行分类（仅限于人的角色和订单角色类型），那么围绕该**角色类型**的关系可能需要是多对多的递归关系。

表3-12中所示的是与其他**角色类型**相关的角色**类型示例**，从中你可以看到“项目工作人员”和“项目负责人”有个上级角色“仅限于人员的角色”。而“项目赞助商”的上级角色却是“当事人角色”。我们有充分的理由来捕获这些关系。首先，提供了一种方法将不同类型的角色归类到上级目录中，这样汇报角色会更加的方便。例如，企业可能会问，“什么样的角色只能够由人来扮演？”我们可以根据“仅限于人员的角色”这一**角色类型**来回答这个问题。其次，图3-9和3-10中介绍的泛化的三级模式似乎以“掩盖”了有些角色是仅限于人来扮演的而其他的角色是仅限于组织的。通过使用**角色类型**我们在某种程度上解决了这个问题。尽管所使用的**角色类型**没有明确地执行本章节先前所述的“仅限于人员”的规则，但是它允许你表示某一**角色类型**是个“仅限于人员的角色”。

**表3-12** 角色类型的层级

**角色类型**分类另一个有趣的方面是有些角色类型从不直接由实体的当事人来扮演。例如，一个人应当是从不扮演“仅限于人员的角色”。换句话说，有些角色类型的实例只用于其他角色类型的分类。

**注：有些建模人员可能看中另一个方案，而不是存在于实体、当事人和角色类型之间的三叉路口模式，你可以展示这种实体到当事人角色是多对多关系的模式。这个选项的一个缺点是在使用该角色之前要声明下该角色并且要将其作为当事人角色的一个实例。这样做的问题上下文角色不是必须事先声明。例如，可能不需要声明电话交流事件的“接听者”或“拨打者”角色（正如数据模型资源手册卷1中所介绍的交流事件的子类），但是他们还是很重要的上下文角色。**

表3-13包含了项目和与项目相关的项目角色的范例，表3-13包括“客户主数据项目”、“人力资源数据库”、“信息架构标准”和“萨班斯-奥克斯利法案报告”。当事人可以在**项目**的上下文中扮演很多的由**角色类型**所描述的**项目角色**。例如，**当事人**“Una Corr”扮演了“项目工作人员”和“客户主数据项目”的“项目负责人”。

**表3-13** 三级上下文角色当事人、角色类型、项目和项目角色的应用范例

对于“Sarbanes-Oxley法案报告”项目具有多个“项目赞助商”角色类型的实例，这说明了一个**项目**可以有多个“赞助商”，在“审计部门（Audit Dept.）”，案例“审计部门（Audit Dept.）”和“信息技术部门（Information Technology Dept.）” 都赞助了“Sarbanes-Oxley法案报告”和“信息架构标准”。上下文角色模式解决了**当事人**与**项目**之间的多对多关系，并且为**当事人**和**项目**之间的上下文角色信息的提供了存储空间。

因此，这种模式允许维护许多当事人，这些当事人可以扮演很多的角色，甚至允许同一个当事人多次扮演相同的角色（因为随着时间的推移同一个当事人可能不止一次扮演了相同的角色）。因此，如果添加或发掘了新角色，你可以添加的实例，并且不需要更改模型。在表3-13的最后一行说明了这一点，这一行需要一个新的角色类型“质量保障管理员”，数据模型很容易地支持了这一新的角色。

这种模式始终支持多对多关系，因此，它追踪角色的历史记录，例如，某人开始是项目负责人，让后另一个人又担任了项目负责人，正如“人力资源数据库”项目中，项目负责人从“Una Corr”变成“Vinny Chintappaly”。此外，由于该模式允许角色到实体的关系是多对多关系，所以这种模式不存在需要改变基数的问题。所以，如果业务规则发生变化，一个项目可以有多个项目负责人，数据模型将会支持这一需求。但是，也由于这种模式始终支持多对多的关系，我们“丢失”了在一个给定的时间点中一个项目只能由一个项目负责人这个规则。如果你选择使用这种模式，将不得不在数据模型之外支持这个规则。

这种模式的另外一个优点是在一个地方维护所有的角色类型。这就意味着可以一致地将业务规则应用到所有的角色类型中。因此，为了以同样的方式处理所有角色，数据建模人员和架构师可以使用此模式。

### 什么时候应当使用这种模式呢？

我们在以下时刻使用这个模式：

* **我们希望为上下文角色建模提供更灵活的解决方案。**可以新的角色添加成为**角色类型**。当当事人扮演了该角色，通过一个**角色类型**，它可以作为特定的**上下文角色**的一个实例来存储。这种模式允许添加或更改新角色，且不用更改底层结构，这对于处理变化所带来的影响具有重要的意义。
* **需要一种通用的方法来为所有的角色建模。**可以使用泛化的**上下文角色类型**和**角色类型**实体为很多实体的上下文角色建立模型，因此，将所有的角色类型存储在一个地方，并且以同样的方式为上下文角色建模。
* **企业具有动态的环境，随着时间的推移，期望添加新角色或期望更改角色的属性比如说基数。**
* **企业没有一个良好的处理所需上下文角色的方法，因此，这些角色可能会经常更改。**
* **企业希望只在一个地方存储和维护所有的角色类型。**因为这种模式没有特定的上下文角色，所有不同类型的角色都放在角色类型中。

### 此模式的弱点是什么？

这种模式的缺点如下：

* **这种模式是一种更加泛化的建模方式，因此更加难以理解。**随着泛化程度的递增，可能使得人们对这种模式的理解和应用变得更加困惑。要注意保证所有的股东都能很好的理解该结构。
* **因为这种模式更加难以理解，所以很难使用这类建模结构作为处理信息需求的手段。**如果有人使用这种模式，我们建议展现示例数据（如表3-13所示）以帮助说明这种模式是如何工作的。
* **当使用泛化风格的模型时，如本模式中的模型风格，该模型不能处理特定的业务规则。**例如，XYZ公司所使用的规定其项目负责人角色的业务规则，在现在的数据模型中已经丢失。一个规则是一个**项目**只能存在一个项目负责人。而在这个模型中，一个**项目**在同一时间可以有多个项目负责人。当你在做一个物理数据库设计时，可以使用一个存储程序或数据库约束来实现这类规则，因此，当强制执行当前的业务规则时，比如说一个项目不允许有多个项目负责人负责，该模式的多对多关系可以适应将来潜在的需求。
* **这种模式不能支持上下文角色的可选性。**假如项目可以具有一个负责人，但是不是必须要有一个赞助商呢？
* **这种模式没有明确地区分哪些角色仅限于人，而哪些角色仅限于组织。**我们使用**角色类型**递归关系部分地解决了这个问题，但是它没有明确地支持业务规则。
* **这种模式可以处理具有不同属性的不同的上下文角色。**如果**项目赞助商**具有“赞助比例”属性，但是该属性并不适用于其他的上下文角色，这样该怎么办呢？我们可以创建一个具有特定项目角色属性的**项目角色**子类，但是这是一个泛化的模式。通过添加特定的属性，我们反映了该模式的一本性质。
* **由于是泛化的角色类型实体，所以很容易错误的使用不恰当的角色类型。**这个实体包含了所有的**角色类型**，有些角色类型并不与所有的**实体**相关联。例如，这种模式表示项目可能具有收款客户，发票可以具有项目赞助商。我们可以通过创建**角色类型**的子类（比如说**项目角色类型**和**发票角色类型**）来解决这个问题。另一个方案是完全不使用**角色类型**，为**项目**和**发票**的上下文角色特地创建**项目角色类型**和**发票角色类型**。

### 简介

本节我们介绍一种更加两会的方式来为上下文角色建模。这种模式适合于以下企业：该企业要么是具有动态的业务环境要么是很难预测或了解随时间推移而需要的上下文角色。使用这种适应性很强的模型，可以不更改模型而轻易地添加上下文角色，只需添加项角色类型并将他们分配到上下文角色实例中。如果使用这种模型作为数据库设计的基础，数据库设计会更加稳定和灵活，因为即使当业务流程发生变化时并需要更改角色时，也不一定非要更改模型。

基于这种模式的数据模型更加的泛化和灵活，因此可能导致有些受众对此感到困惑或不能充分地理解这种数据模型。所以，通常使用这种模式作为物理数据库设计的基础，而不使用它作为阐释业务代表的信息需求的手段。这种模式提供了更大的灵活性，但是同时也丢失了该数据模型层级上的严格规范的业务规则。

## 混合型上下文角色模式

在这一章的这一节中，我们探讨下混合模式，这种模式结合了二级模式和三级模式的特征。在此模式中，将两种风格的混合而产生有效的模型，而该模型是集合了这两种模式的最佳特点。

### 为何我们需要这种模式？

我们发现了一种非常有效的数据建模策略，就是使用特定和泛化这两种建模风格来建立模型。一种是为了更好的理解信息需求，可以使用特定风格的模型，而要为数据库设计提供一个坚实、灵活的基础，可以使用泛化风格的模型。通过创建两个不同的模型，使用这些不同的建模风格，或者正如以下模式所示，一个模型可能包含了特定和泛化这两种建模风格。当你具有一个这样的模型时，你可以给不同的受众展现不同的数据模型视图。例如，你可以给业务代表展示特定建模风格的视图，而给架构师则展示泛化建模风格的视图。

这种建模风格的另一个优点是它提供了可能性的备选方案，可供物理数据库设计选择。物理数据库设计人员可以用三种方式实现这类模型：特定方式、泛化方式、或者是两者都使用，对于关键的数据可以使用特定的结构，对于其他可能出现的需求使用泛化的结构。例如，一个项目可能有几个关键的上下文角色比如说项目赞助商、项目工作人员和项目负责人，对于这些角色可以使用模式的特定方式。随着时间的推移还可能出现额外的角色，因此可用模式的泛化方式来适应这些需求。

此外，处于不同成熟度的大型企业可能具有不同的业务线。这种模式可以帮助很好的理解跨国公司的核心业务。并定义了核心业务中的所有的角色，并描述了围绕这些角色的所有规则。而跨国公司新的业务线或其所感兴趣的领域可能还没有被充分的了解或介绍。因此可能没有详细定义或发掘所有的上下文角色和围绕该角色的规则。企业可能需要以一致的方式处理这两种情形。混合型上下文角色模式满足这一需求。

**注：这种模式的方法是为同类数据建模提供可选的方案：一种是使用特定的方式，而另一种是使用更加泛化的方式；这并不等于说它同意冗余地维护相同的数据。我们不认为这将会为冗余的数据建模，因为我们不提倡你使用两种方式维护相同的数据实例。**

### 这种模式是如何进行工作的？

图3-11介绍了这种模式是如何构建的。你应当认识此表中二级和三级模式的组成成分。

**图3-11** 混合型上下文角色模式

从该图的顶中间和右侧部分可以看到二级上下文角色模式的结构。这个结构支持声明特定的上下文角色。“每个**当事人角色**在包含一个或多个**特定上下文角色**的上下文中扮演该角色”特定的上下文角色与当事人所扮演的声明性角色相关。它定义了一个非常特殊的业务规则，每个**特定的上下文角色**必须由**声明性角色1**来扮演（声明性角色1也可以与任何声明性角色或**当事人角色**相关）。在此案例中，**特定上下文角色**实例存在于声明性角色的上下文内。这类建模风格一般用于十分了解的实体、商业事务或事件，其角色的业务规则和需求是非常稳定的，不会改变的。

在该图更加特定的部分可以看出另一个业务规则，即在图中特定上下文角色充当了一种关系。“每个**声明性角色2**可以在包含一个或多个**实体**的上下文中扮演该角色并且每个实体可以与一个且只能与一个**声明角色2** 相关联”（例如，正如图3-12所示，每个**项目**只有一个项目负责人）。这是一个非常特别的业务声明，所以一个可能会影响建模风格的一个有价值的问题是，“业务规则是否会始终是这种情况？”

**图3-12** 混合型上下文角色模式应用范例

这种模式也提供了更为泛化的方式，支持上下文角色。你应当能够认识位于模型图底部的三级上下文角色模式结构。这种模式创建了灵活的结构，可以将上下文角色当作**一般上下文角色**的实例来添加。“每个通用的上下文角色可以用于一个或多个**实体**。”当发掘一个上下文角色时，可以将它添加成**一般上下文角色**的实例，而它的角色类型则可以添加到**角色类型**中。

这种模式一个意思的地方就是在这种模式中有公用的实体。实体当然是公用的；否则，上下文角色（一般的或特定的）将没有任何意义！当事人和角色类型对于这两种风格的角色也是公用的。当事人需要适合于两种风格的上下文角色，因为你也不想为当事人所扮演的每个角色一次又一次的维护当事人信息。

有意思的是**角色类型**也是公用的。**角色类型**通过关联到**当事人角色**的方式支持特定角色模型。它也为**一般上下文角色**的上下文，因为这是维护角色的名称的地方。

图3-12提供了这种模式的范例。延续先前的案例，XYZ公司已决定其公司需要三级模式的灵活性，但是也希望像二级模式一样可以为核心角色维护一些特定的业务规则。基于这样的需求，数据专业人士创建了数据模型，如图3-12所示。

从图3-12，你可以看到图3-6所示的二级模式与图3-10所示的三级模式相结合的模式。将核心声明性角色**赞助商**、**工作人员**和**项目负责人**建模为特定的上下文角色。XYZ公司所规定的这些角色是其业务的基础。

充当这些角色的**当事人**与**项目**是具有特定的关系。只有扮演**工作人员角色**的**当事人**可以以**项目工作人员**的身份参与到一个项目中。也就是，扮演了声明性角色**工作人员**的人可以通过扮演**项目工作人员**这一上下文角色参与到**项目**中。

**注：这一章的二级上下文角色部分已提供了当事人角色与项目的关系的详细解释。**

如果你观察下图的底部，你可以看到与图3-10一样的泛化结构。在此机构中，“每个**当事人**可以在包含一个或多个**项目角色**的上下文中扮演该角色”。当发掘新的角色时，这种灵活的结构允许你通过添加新角色的**角色类型**实例这种方式来添加角色。

**注：这一章的三级上下文角色部分已提供了当事人与项目角色的关系的详细解释。**

这种模式的优点（和缺点）在于可以使用两种不同的方式为某个特定当事人所扮演的角色建模。例如，表3-14中的“客户主数据项目”和“人力资源数据库”。“Neena Davies”从2010年10月10日开始担任“客户主数据项目”的工作人员这一角色，从2010年1月1号开始她也是“人力资源数据库”项目的工作人员。在这两个案例中扮演了**工作人员**（声明性角色）的她以**项目工作人员**的身份与她所参与的项目相关联。如果你观察下表3-15，你可以发现同一当事人“Neena Davies”还作为“技术作家”参与了“客户主数据项目”。在这个示例中，模式的泛化部分允许你使用添加**角色类型**的实例这种方式来轻松地添加新角色，而不用更改数据模型或结构。

**表3-14** 混合型上下文角色模式，工作人员和项目工作人员的范例

**表3-15**混合型上下文角色，项目角色范例

所以，是否**项目角色**包括了所有与**项目**相关的上下文角色（包含特定的上下文角色如“项目工作人员”和“项目赞助商”）还是只包含**当事人角色**未处理的那些角色以及特定的上下文角色？这个问题的答案是项目角色可以包含那些已建模的角色，特别是像“项目工作人员”这样的角色，但是我们并不推荐这样做。我们认为重要的是将该上下文角色作为特定的上下文角色来维护，比如说**项目工作人员**，或者使用泛化的**项目角色**/**角色类型**结构来维护该上下文角色，例如，表3-15所示的“技术作家”。如果你使用泛化的**项目角色**/**角色类型**结构来维护“项目工作人员”和“项目赞助商”的角色类型，我们建议你也不要使用特定的项目工作人员、项目赞助商的结构来维护这类角色。我们认为你不应当使用特定和泛化这两种方式冗余地维护上下文角色信息。例如，你最终可能会重复计算从事某个项目的工作人员数量。或者，如果使用了这两种方式，你将面临以下问题，对于程序员编写查询、添加和更新项目工作人员的代码，哪种才是正确的方法，以及对于整个企业而言哪种才是正确的语义视图。

这种模式的优势是你可以获得特定的二级建模风格和泛化的三级模式的建模风格的优点。当我们需要维护“Neena Davies”的关系时，声明她为工作人员并作为项目工作人员参与到项目中。当我们发掘新角色“技术作家”时，我们想维护此角色，但是并不想添加新的声明性角色实体，因为它并不是XYZ公司业务的核心角色。通过给项目角色添加一个新的角色类型“技术作家”，“Neena Davies”参与到该项目中。

关于这种模式很重要的一点是它可以提供不同的数据视图，因为它是以特定和泛化的方式为角色建模的。如果你是为了理解信息需求和关于这些角色的业务规则，可以使用特定的建模风格。如果是为数据库设计提供灵活的基础，你可以展示泛化的建模风格。使用者两种方式来建模，你可以向不同的受众展示不同的模型视图以及也可以为数据库设计提供多种可能性。例如，物理数据库设计人员可以评估他们是否想要在数据库设计中实现特定的模型、灵活的模型或者两者都实现。可以基于模型所需的灵活性和性能以及对模型的简洁程度的考虑而做出这个决定。还有很重要的一点需要强调，我们不推荐你在同一时间使用者两种方式来维护角色的数据。

**注：这项技术用一个单一的“混合型”模式将特定和泛化的风格结合了起来，尽管我们并没有在其他章节特别地介绍混合型模式，但是这项技术可以应用于本书所给出的许多其他类型的模式。我们将在第9章节中多这项技术做进一步的介绍，在该章节中我们将应用该模式开发一个企业数据模型。**

### 该在何时使用这种模式？

我们在以下时刻使用这种模式：

* **企业有一组核心的上下文角色，企业希望维护这些角色并为其建立特定的模型，但是它也希望在不更改底层模型的前提下，能够灵活的维护新的上下文角色。**
* **我们希望展示企业的上下文角色的不同视图，但是用一个单一的数据模型来维护这些角色。**一种视图可以是针对商业用户的，一种视图是针对数据库架构师和/或设计人员。针对商业用户的视图可以是特定的模型，而针对数据库架构师和/或数据库设计人员则可以是泛化的结构。
* **我们希望维护一个单一的逻辑数据模型，以至于我们不需要维护和交叉引用风格各异的各种模型。**不同的模型具有不同的用途可能会很有意义。然而，根据我们的经验，维护两个或两个以上的数据模型（除了其他模型，比如说物理数据库设计模型和流程模型）以及交叉引用这些模型是需要很多约束的。也可以很公正的说，维护相同模型的两个部分是要花费精力的且还需要约束，但是如果泛化和特定的这两部分是在同一的模型中，这样做是很有帮助的。

### 这种模式的缺点是什么？

这种模式的弱点如下：

* **在这种模式中，有两种方式可以维护另一个实体中的当事人相关信息与或角色，并且你有可能维护了不同的名称下的相同角色。**例如，在表3-14和表3-15中，“Neena Davies”被定义为**项目工作人员**和“技术作家”。谁又会说这不是一样的角色，或者可能技术作家是项目工作人员的一类呢？
* **如果你尝试回答下面这个问题，“我们企业中所有不同的角色类型是什么？”，你将不得不对两个地方进行查询。**首先你将下需要获得所有的特定上下文角色，例如，**项目赞助商**、**项目工作人员**等，然后你需要处理所有的**角色类型**实例，例如，“技术作家”等。
* **该模式可能会使人感到困惑。**这就是为什么数据专业人士将需要根据受众而创建不同的视图。对于那些喜欢坚持一种风格的数据模型的企业，他们根据特定情形的需求选择泛化的或泛化的建模风格，而不使用这类混合型风格，可能更会更好。
* **不是一种模型具有两种不同的选择，而是各具优势的两个不同的数据模型：**非常特定的模型展示了角色可以以任何一种方式与业务代表进行有效的交流并通过其验证，另一种是针对技术受众的架构数据模型是较泛化的模型，是数据库设计的基础。有这两种不同数据模型，对于根据需求精确地创建这些模型会更加有效率。

### 简介

在这一节中我们介绍了为上下文角色建立灵活的模型的方式和为上下文角色建立特定的模型的方式。图3-11中所创建的模式是二级上下文角色模式和三级上下文角色模式的合体。这种风格的模型功能很强大，因为通过使用特定风格的模型，它提供了一种理解信息需求的方法，通过使用灵活风格的模型，它为数据库设计提供了稳定、灵活的基础。

这种模式表示数据专业人士无须维护两个不同的数据模型，一个是针对业务人员的，一个是针对数据架构师和/或设计师的，因为同一模型可以支持不同的视图。但是也意味着你必须处理同一模型的两种不同的建模风格。

使用这种模式，有两种可能的途径维护当事人在另一个实体、商业事务、活动或事件中的相关信息。如果滥用了这种模式可能会导致混乱或使用不同途径存储同一个角色的冗余信息。因为存在误用的可能性，有些企业可能还不适应混合型的模型。这种模式具有了一级模式和三级模式的优点，但同时也有了这些模式的缺点。

## 模式的总结

表3-16包含的是本章节中的所有模式的简介

**表3-16** 模式的简介

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 一级上下文角色模式，属性 | 最为泛化的上下文角色建模方式  不同的实体包含相关的角色，并将这些角色作为实体的属性。 | 作为声明范围的一部分，用来收集和验证数据需求。  当有一组定义明确的静态的上下文角色时。  当一个角色与另一实体相关，只需要这个角色的一条信息，而该角色不需要其自身的实例时。 | 当建模、设计或实施需要灵活性时 | 易于理解和使用 | 它不是灵活的模型。  可能会包含了人员和组织的冗余数据导致错误数据出现。  可能会有很多不同的特定类型的角色（例如，“工作人员”），而要展示每个可能的角色的属性是不太切合实际的（例如，维护30个具有可能性的角色的30个属性）。  没有明确地区分人员与组织角色。  容易混淆声明性角色与上下文角色。  不能追踪角色的历史记录。 |

**表3-16** 模式的简介（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 一级上下文角色模式，关系 | 这种模式将角色信息从包含角色的实体中分离出来。语义上这表明你意识到可以在相关实体之外定义角色的信息。 | 当同一当事人或组织扮演一个声明性角色，且可能以相同的方式参与多次时（上下文角色）。  当你需要用严谨的方式处理上下文角色的规则时。  当你需要某一领域中特定的个体角色时，换句话说，作为特定范围声明或需求声明。  当企业没有引进当事人的概念，却使用一级声明性角色模式时。 | 当需要较灵活的模型时。  在要添加或更改角色或规则的动态的环境中。 | 特别声明了上下文角色的规则。  开始将声明性角色绑定到上下文角色上。  语义上强调了在相关实体之外维护上下文角色是很有意义的。他们被当作实体来维护（或关系），而不是作为属性来维护。 | 不是非常灵活的模型，不容易适应新角色。  重复了当时的冗余信息。  可能会导致数据质量问题。不能追踪角色的历史记录。 |

**表3-16** 模式的简介（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 二级上下文角色模式 | 这种模式使用声明性角色，这些声明性角色可以 作为为当事人创建上下文角色的基础，该上下文角色与另一个实体的当事人的相关信息关联。 | 当企业希望将当事人和当事人角色整合到上下文角色当中时。  当你希望避免维护当事人和当事人角色的冗余数据时。  为了更好地理解信息需求，需要特定的模型时  当你需要以严谨的方式处理上下文角色的规则时。 | 有些企业不使用声明性角色（当事人角色）和这种模式可能不适合他们。  当你需要将仅限于人员的角色、仅限于组织的角色和当事人角色区分开来时。  当你需要更大的灵活性，需要一个模型能够适应角色的添加和角色属性的更改时。 | 对于为角色指定角色需求的范围，这是个有效的方法。  指定了数据模型中的业务规则。  可以不存储当事人和当事人角色的冗余数据。  将声明性角色同上下文角色整合在一起。（在非常动态的环境中，可以认为这是个缺点） | 有些企业可能理解或使用当事人角色结构有困难。  不是非常的灵活，不可以在不改变底层数据模型的情况下，更改其特定的属性。  没有将仅限于人的角色、仅限于组织的角色和当事人角色区分开来。（然而，这点可以通过人员角色和组织角色子类来解决）  如果不存在现存的饿声明性角色，你可能需要为新发掘的上下文角色添加新的声明性角色。  对于实体与角色之间的关系是一对多的关系，这种模式不能追踪该角色的历史记录。 |

**表3-16** 模式的简介（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 二级上下文角色模式，当事人，只作备选模式 | 这种模式直接将当事人关联到他们的上下文角色。这种模式满足支持上下文角色的需求以及支持他们直接关联到人员和组织的方式。 | 当你希望避免存储当事人的冗余的信息时。  当企业不使用当事人角色或企业不通过当事人角色绑定上下文角色时  当直接与当事人关联比使用当事人角色更有意义时，例如，当角色是很一般的角色时，比如说发票的“收据人”和“开票人”。  当你需要处理上下文角色的特定的规则时  当你需要将仅限于人的角色、仅限于组织的角色以及当事人角色进行区分时 | 当企业希望通过他们的当事人角色来维护上下文角色时。  根据将来的需求，需要添加和更改角色时。  当你需要上下文角色绑定到声明性角色语义上的严密性时。 | 相对而言，是简单的模型，易于理解。  对于记录范围和业务规则很有效。  处理了上下文角色以及他们与实体关系的特定属性。  区分了仅限于人的角色、仅限于组织的角色和当事人角色。  更加灵活一些，因为你可以将任何当事人直接关联到相关的实体，而不需要事先将当事人声明为特定角色。 | 这种模式还是会具有灵活性方面的问题。当发掘新的角色时，将需要添加新的实体。  不能追踪角色的历史记录。 |

**表3-16** 模式的简介（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 三级上下文角色模式 | 三级上下文角色模式提供了灵活的解决方案，允许动态地添加（和废止）某个实体的上下文角色。 | 当上下文角色的范围尽可能会发生变化时。当需要灵活性时。  围绕泛化的上下文角色和角色类型实体为所有角色创建常见的业务规则时。  当在动态变化的环境中建模时。  作为实现模型，它适应很适应变化的。 | 作为作用域声明或这是收集和验证非技术人员的数据需求。  由于该模式泛化的特质，丢失了一些业务规则（例如，基数规则和可选规则）。  当角色具有不同的属性，而需要将这种模式改成更特定的模型时。  如果企业希望通过上下为角色与特定当事人角色的特定关系来维护上下文角色时。 | 这种模式是支持上下文角色的灵活一致的方法。  可以追踪什么人在什么时候扮演了什么样的角色这样的历史记录。 | 由于更加的泛化，所以可能很难被非数据专业人士所理解。  当你使用泛化风格的建模模式时，比如说使用这种模式，模型不能处理特定的业务规则、特定的技术以及特定的属性。  因为这种模式难以理解，所有更难使用这类建模结构作为处理信息需求的手段。  由于它使用了泛化的角色类型，所以可能会关联不恰当的角色，而这些角色应当是不可能出现在特定的上下文当中的。 |

**表3-16** 模式的简介（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模式 | 说明 | 何时使用 | 何时不使用 | 优点 | 缺点 |
| 混合型上下文角色模式 | 这种模式为上下文角色的建模提供了特定和灵活的建模方式。它是一个单一的模型，将二级模式和三级模式结合在一起。 | 当需要为一些核心的特定的角色建立模型，并且随着时间的推移可能还需要其他的角色，而该模型能够适应这些角色。  当需要处理特定的上下文角和灵活的上下文角色时。  当企业希望用一个单一模型支持灵活的和特定的这两种上下文角色时。 | 当企业不认同混合型的建模风格时 | 是一种非常灵活的解决方案，同时也能处理上下文角色的特定的业务规则。  提供了一种方法，在必须维护唯一的一个模型时，针对不同的受众展现不同的视图。  为物理数据库设计提供了多种选择。 | 这是最复杂的一种模式。  角色可以通过两种途径与当事人关联，如果误用了这种模式，可能或冗余地存储角色的信息。  由于这是混合风格的模式，所以可能会使人们感到很困惑。  有些人可能认为使用两种不同的数据模型可能会更加有利。  由于角色是通过两种不同的建模方式来维护的，所以可能很难发现所有可能的角色。  这种模式不如从两种独立的模式中选择那样自由，两种分开的模式，允许你建立一个所需的特定模型，并可以建立另外一个尽可能泛化的模型。 |

**参考文献**

1 See Chapter 2 0f *The Data Model Resource Book, Revised Edition, Volume 1: A Library of Universal Data Models for all Enterprises* (Wiley, 2001).

2　Paraphrased　from　dictionary.com(http://dictionary. reference.com/browse/context .

3　Paraphrased from dictionary.com　(http://dictionary.reference.com/ browse/role).

4 With permission from William G. Smith, Founder of William G. Smith and Associates, Syllabus for Conceptual Data Modeling Course, 2008.

5 The project concept is really a part of the WORK EFFORT concept that is defined in Chapter 6 *The Data Model Resource Book, Revised Edition, Volume l: A Library of Universal Data Models for All Enterprises* (Wiley, 2001).

6Paraphrased from dictionary.com　(http://dictionary.reference.com/ browse/project).

7 有关声明性角色的更多内容请参看本书的第二章节

【注释1】 textual reference 暂译“文本引用”