**第3节知识图谱中模式、标识和上下文的表示形式**

在本节中，我们将介绍数据图的各种增强和扩展——与模式、标识和上下文相关——这些增强和扩展为积累知识提供了额外的结构。此后，我们将使用第2节中讨论的模型之一，将数据集合表示为节点和边。我们称之为知识图，它有一个可能通过模式、标识、上下文、本体和/或规则的表示来增强的数据图。这些附加表示可以嵌入到数据图中，或在其上分层。本文将讨论模式、识别和上下文的表示形式，而本体和规则将在第4节中讨论。

**3.1模式**

与关系模型相比，将数据建模为图形的好处之一是可以选择放弃或推迟模式的定义。然而，当将数据建模为图形时，图解可用于规定图形遵循或应该遵循的高级结构和/或语义。我们讨论三种类型的图模式：语义、验证和合并。

**3.1.1.语义图式。**语义模式允许定义图形中使用的高级术语（又称 专业词汇或特别含义的用语）的含义，这有助于使用这些术语对图形进行推理。例如，看看图1，我们可能会注意到一些基于节点所指实体类型的节点自然分组。因此，我们可能决定定义类别来表示这些类别，如事件、城市等。事实上，图1已经说明了三个低级别类别——开放市场、食品市场、饮料节——用边缘标记类型将类似实体分组。随后，我们可能会观察到我们想要捕捉的这些类之间的一些自然关系。在图10中，我们展示了一个事件的类层次结构，其中子类被定义为其父类的子类，这样，如果我们在图中找到边缘eId15食品节日类型，我们也可以看出边缘eId15节日类型和eID15事件类型。

除了类之外，我们可能还希望定义边标签，又称属性。回到图1，我们可以认为属性city和venue是更一般的location的子属性，例如，给定边缘Santa Lucía 属于城市Santiago，我们也可以推断出。例如，我们也可以考虑，bus和flight是一个更一般属性的子属性。因此，属性也可以形成层次结构。我们可以进一步定义属性域，指出具有该属性的边从中延伸出来的节点的实体类别；例如，我们可以定义connect to的域为一个类Place，这样，考虑到前面的子属性关系，我们可以得出Arica类型。相反，我们可以定义属性的范围，指示具有该属性的边延伸到的节点的实体类别；例如，我们可以将city的范围定义为一个City类，从而推断出Arica类型。

定义RDF图语义模式的一个突出标准是RDF模式（RDFS）标准[66]，该标准允许在RDF图中使用的类和属性之间定义子类、子属性、域和范围，这些定义可以序列化为图。我们在表1中说明了这些功能的语义，并在图11中提供了一个具体的定义示例，作为运行示例中使用的术语示例。然后可以将这些定义嵌入到数据图中。更一般地说，图中使用的术语的语义可以比这里看到的更深入地定义，这是RDF图的Web本体语言（OWL）标准[228]所支持的。我们将在后面的第4节中回到这种语义。

语义模式通常是为不完整的图形数据定义的，其中两个节点（如Viña del Mar flight Arica）之间没有边并不意味着这种关系在现实世界中不成立。因此，从图1的图表中，我们不能假设Viña del Mar和Arica之间没有航班。相比之下，如果采用封闭世界假设（CW A）被采用，就像许多经典数据库系统中的情况一样，可以假设数据图是对世界的完整描述，从而可以确定两座城市之间不存在航班。不采用CWA的系统被称为采用开放世界假设（OWA）。CWA的一个结果是，在数据图中添加一条边可能会与之前假设的错误的内容相矛盾（由于信息缺失），而对于OWA，一条被证明为错误的语句在添加更多边后仍然是错误的。

考虑到我们的运行示例，假设旅游组织对其知识图中描述的一切都有完整的知识是不合理的。然而，如果系统无法法对诸如“是否存在”之类的问题明确回答“是”或“否”，例如“Viña del Mar和Arica之间是否有航班？”，这就很不方便，尤其是当该组织确定其对航班有完整的了解时。OWA和CWA之间的折衷方案是局部封闭世界假设（LCWA），其中数据图的部分被假设为完整的。

**3.1.2.验证模式。**当图形被用来大规模地表示各种各样的、不完整的数据时，OWA是最合适的语义选择。但在某些情况下，我们可能希望保证我们的数据图——或其中的特定部分——在某种意义上是“完整的”。例如，回到图1，我们可能希望确保所有活动至少有一个名称、一个地点、一个开始日期和一个结束日期，以便使用数据的应用程序——例如，向用户发送活动通知的应用程序——可以确保他们拥有所需的最少信息。此外，我们可能希望确保活动的城市是被声明为城市（而不是推断它是一个城市）。我们可以在验证模式中定义此类约束，并根据生成的模式验证数据图，列出违反约束冲突的情况（如果有的话）。因此，虽然语义模式允许推断新的图形数据，但验证模式允许验证现有的图形数据。

定义图形验证模式的标准方法是使用形状[278，287，398]。形状目标是数据图中的一组节点，并指定这些节点上的约束。形状的目标可以通过多种方式定义，例如以类的所有实例、属性的域或范围、查询结果、通过给定属性连接到另一形状目标的节点等为目标。然后可以在目标节点上定义约束，例如限制给定属性上取值的数量或类型。形状图形由一组相互关联的形状组成。

形状图可以被描述为类似UML的类图，图12展示了一个基于图1的形状图的示例，定义了四个相互关联的形状上的约束。每个形状用一个方框表示，如地点、事件等都与一组约束相关联。当且仅当节点满足形状上定义的所有约束时，节点才能定义为该形状。在每个形状框中，对一致性节点可以与属性（例如名称、开始等）相关的节点的数量（例如，[1..\*]表示一对多，[1..1]表示精确的一对一）和类型（例如，字符串、日期时间等）进行约束。另一种选择是对符合特定形状的节点数量施加约束，该符合节点可以与属性相关（从而在形状之间生成边）；例如，Event Venue Vince 1..\*表示一致性节点Event必须至少与一个具有符合Venue 形状属性的节点相关。形状可以继承父形状的约束——用△连接器表示，与City and Venue的情况一样，其一致性节点也必须符合Place形状。

给定形状和目标节点，可以检查节点是否符合该形状，这可能需要检查其他节点的符合性；例如节点EID15符合Event形状不仅基于其本地属性，而且基于从Santa Lucía 到 venue 和 Santiago 到 city的一致性。一致性依赖也可能是递归的，其中从Antiago 到City的一致性要求它符合Place，这要求Viña del Marand 和arica符合Place，等等。相反，EID16不符合Event，因为它不具备示例形状图所需的Start和End属性。

在声明形状时，数据建模者可能事先不知道某些节点可以拥有的整个属性集。开放的形状使节点具有形状未指定的其他附加属性，而闭合的形状不具有这些属性。例如，如果我们将边Santiago 创建者Pedro de Valdivia添加到图1中所示的图中，那么如果Santiago仅在该形状被定义为开放的，则符合City形状（因为该形状没有提及创始人）。

实用形状语言通常支持额外的布尔特征，例如形状的连接（and）、析取（or）和否定（not）；例如，我们可以说Venue的所有值都应该符合Venue and（not City）的形状，明确表示数据图中的Venue不应该直接指定为城市。然而，自由组合递归和否定的形状语言可能会导致语义问题，这取决于它们的语义是如何定义的。为了说明，考虑下面的案例灵感来自理发师悖论〔287〕，涉及形状理发师（Barber），其符合节点shave必须至少一个符合Person和（不是Barber）的节点。现在，考虑到（仅） Bobb是否符合Person？如果是，即Bob符合Barber，则Bob通过不剃除至少一个符合Person 和（不是Barber）的节点来违反约束。如果否，即Bob不符合Barber，则Bob通过修剪此类节点来满足Barber约束。基于分层[57]、部分赋值[100]和稳定模型[171]，提出了避免这种矛盾情况的语义。

虽然验证模式和语义模式服务于不同的目的，但它们可以相互补充。具体而言，验证模式可以考虑语义模式，例如，验证应用于包括推断的数据图。以图10的类层次结构和图12的形状图为例，我们可以将Event形状的目标定义为类型为Event类型的节点（类）。如果我们首先对语义模式的类层次结构进行推断，那么这个Event形状现在将指向ID15和EID16。然而，语义模式的存在可能需要调整验证模式。例如，考虑到前面提到的类层次结构，需要在类型属性上定义一个宽松的基数。在这种情况下，开放形状也可能是首选的，而不是枚举节点上可能推断出的所有可能属性的约束。

我们在附录B.3.2中提供了形状和相关概念的高级定义。最近出现了两种用于RDF图的形状语言：形状表达式（ShEx），作为W3C社区小组报告[398]发布；和SHACL（形状约束语言），作为W3C推荐[278]发布。这些语言支持所讨论的功能（以及更多功能），并已被用于验证与医疗保健[494]、科学文献[206]、空间数据[80]等相关领域的图形。关于ShEx和SHACL的更多细节可以在Labra Gayo等人的书中找到[287]。最近提出的一种语言可以作为ShEx和SHACL的共同基础的语言，揭示了它们的相似性和差异[286]。Angels[12]对属性图提出了类似的模式概念。

**3.1.3紧急模式。**语义和验证模式都需要领域专家明确指定定义和约束。然而，数据图通常会显示潜在的结构，这些结构可以自动提取为紧急模式[388]（又名图表摘要[81302467]）。

商图是一种常用于定义紧急模式的框架，它根据某种等价关系划分数据图中的节点组，同时保留图的某些结构属性。以图1为例，我们可以根据上下文直观地区分不同类型的节点，例如链接到场馆节点的事件节点，以及链接到城市节点的事件节点，等等。为了描述图的结构，我们可以考虑节点的六个分区：事件、名称、地点、类别、日期时间、城市。实际上，这些分区可以基于节点的类或形状来计算。将每个分区的节点合并为一个节点，同时保留边，得到图13所示的商图：此商图的节点是数据图中的节点的分区，当且仅当x∈X和z∈Z，使得x,y,z在数据图中时,边X,y,Z在商图中。

定义商图的方法有很多，不仅取决于节点的划分方式，还取决于边的定义方式。不同的商图对于它们所保持的结构可能提供不同的保证。形式上，我们可以说每个商图都模拟了它的输入图（基于数据节点和商节点之间的集合成员关系的模拟关系），这意味着x∈X ，x是输⼊节点， X是商节点，如果x,y,z是数据图中的一条边，那么商图中一定存在一条边X,y,Z使得z∈Z。例如，图13的商图模拟了图1的数据图。但是这个商图似乎表明（例如），EID16将在数据图中具有开始和结束日期，而事实并非如此。双相似性给出了一个更强的结构保存概念，在这种情况下，它进一步要求如果X,y,Z是商图中的一条边，那么对于x∈X、 一定存在z∈Z,使得x,y,z在数据图中；图13的商图中的EID16无法满足这一要求，因为在原始数据图中，EID16没有标记为Star或End的输出边。图14展示了商图的一个双相似版本，将分为两个节点，反映出它们不同的输出边。双相似性的一个有趣的性质是它保留了前向路径：给定一个不带逆的路径表达式r和两个双相似图，当且仅当它匹配另一个双相似图中的相应路径时，r才会匹配一个图中的路径。例如，我们可以验证一条路径是否与图1中的匹配，如果且仅当图14中有一条路径与匹配，即x∈X和z∈Z。

根据划分节点的等价关系，可以用多种方式定义商图。此外，根据保留数据图结构的（双）模拟关系[81]，有许多方法可以定义其他类似或双相似图。我们在附录B.3.3中为商图、模拟和互模拟的概念提供了形式化定义。 此类技术旨在将数据图汇总为更高级别的拓扑结构。为了减少商图的内存开销，实际上，节点可以用分区的基数和/或分区的高级标签（例如，事件、城市）来标记，而不是存储分区中所有节点的标签。

还提出了各种其他形式的非基于商图框架的紧急模式；例子包括基于关系表的紧急模式[388]，形式概念分析[189]，等等。紧急模式可用于提供人类可理解的数据图概述，帮助定义语义或验证模式，优化图的索引和查询，指导数据图的集成，等等。我们参考了Čebirić等人[81]的调查了解更多细节。

**3.2识别**

在图1中，我们使用像Antiago这样的节点，但这个节点指的是哪个Santiago？我们是指智利圣地亚哥、古巴圣地亚哥、孔波斯特拉圣地亚哥，还是指独立摇滚乐队圣地亚哥？根据Santa Lucía Santiago city等边缘，我们可以推断它是上述三个城市之一（而非摇滚乐队），根据图表描述智利旅游景点的事实，我们可以进一步推断它指的是智利圣地亚哥。然而，在没有进一步细节的情况下，这种形式的消歧节点可能依赖于在更困难的情况下容易出错的启发式算法。为了帮助避免这种歧义，首先我们可以使用全局唯一标识符，以避免在使用外部数据扩展知识图谱时出现命名冲突，其次我们可以添加外部标识链接，以消除节点与外部源之间的歧义。

**3.2.1.全局标识符。**假设我们希望比较智利和古巴的旅游业，我们已经获得了一个适合古巴的知识图谱。使用图对数据建模的部分好处是我们可以通过取两个图的并集来合并它们。然而，如图15所示，使用像Antiagoo这样的模糊节点可能会导致命名冲突：该节点在两个图中都指两个不同的现实世界城市，合并后的图表明圣地亚哥是智利和古巴的一个城市（而不是两个不同的城市）。避免这种命名冲突的实际方法是在相应的图中使用Chile:或cuba:这样的命名空间，只要使用不同的名称空间，像Chile:Santiago 和cuba:Santiago这样的节点就不会发生冲突。

在语义网的上下文中，RDF数据模型更进一步，建议对节点和边缘标签使用全局网络标识符。然而，RDF 1.1 建议使⽤国际化资源标识符 (IRI) 来识别城市或事件等⾮信息资源，⽽不是采⽤统⼀资源定位符 (URL) 来识别⽹⻚等信息资源的位置。因此，例如，在维基数据[515]的RDF表示中这是一个用于补充维基百科的知识图，在第10节中进行了更详细的讨论.而URL https://www.wikidata.org/wiki/Q2887指的是一个可以在浏览器中加载的网页，它提供了有关圣地亚哥的人类可读元数据, IRI http://www.org/entity/Q2887指的是城市本身。区分两种资源（网页和城市本身）的标识符可以避免命名冲突；例如，如果我们使用URL来标识网页和城市，我们可能会在图形中得到一条边，例如（边下有可读的标签）：

这样的边缘让人模棱两可：佩德罗·德·瓦尔迪维亚是该网页的创始人，还是该城市的创始人？将IRI用于描述它们的网页URL不同的实体，可以避免这种不明确的情况，Wikidata因此定义了前一个边缘，如下所示：



使用IRIs用于城市、个人和创始人，与描述的网页不同。

如果使用HTTP IRI来标识图形的实体，则在查找IRI时（通过HTTP），web服务器可以以RDF等格式返回（或重定向到）该实体的描述。这进一步使RDF图能够通过Web链接到外部RDF图中描述的相关实体，从而产生链接数据[37,216]（在第9节中讨论）。

**3.2.2外部身份链接。**假设旅游局选择在他们控制的⽹络服务器上使⽤诸如http://turismo.cl/entity/之类的 IRI 定义 chile: 命名空间，从⽽允许诸如chile:Santiago之类的节点，IRI的快捷方式http://turismo.cl/entity/Santiago–可在网上查阅。虽然使用这种命名方案有助于避免命名冲突，但使用IRI并不一定有助于确定资源的身份。例如，外部地理知识图可能会将同一个城市指定为IRI geo:SantiagoDeChilein，在其自己的名称空间中，我们无法直接知道这两个标识符指的是同一个城市。如果我们合并这两个知识图，我们将得到同一个城市的两个不同节点。

有很多方法可以确定一个实体的身份。第一种方法是将实体与图形中唯一的标识信息相关联，例如其地理坐标、邮政编码、成立年份等。每一条额外的信息都消除了关于所指城市的歧义，提供了（例如）更多选项，用于将城市与外部来源中的类似城市进行匹配。第二种选择是使用标识链接来声明本地实体与外部源中的另一个相关实体具有相同的标识；这个概念的一个实例化可以在OWL标准中找到，该标准定义了与相关实体相关的OWL:same作为属性。使用此属性，我们可以在RDF图中声明边chile:Santiago OWL:Sameas geo:SantiagoEchile，从而在两个图中的相应节点之间建立标识（身份）链接。OWL标准定义的OWL:Sameas语义允许我们组合两个节点的数据。这种语义将在第4节后面讨论。身份链接的计算方法也将在后面的第8节中讨论。

**3.2.3种数据类型。**考虑图1左侧的两个日期时间：我们应该如何分配这些节点全局标识符？例如，直观地说，将IRIs分配给这些节点是没有意义的，因为它们的语法形式告诉我们它们指的是什么：2020年3月的具体日期和时间。这种语法形式可以被机器进一步识别，这意味着使用适当的软件，我们可以按升序或降序排列这些值，提取年份，等等。

大多数实用的图形数据模型都允许定义数据类型值的节点。RDF使用XML Schema Datatypes（模式数据类型）（XSD）[386]，其中数据类型节点以一对（l,d）的形式给出，其中I是一个词法字符串，如“2020-03-29T20:00:00”，D是一个表示数据类型的IRI，如XSD:dateTime。然后，节点被表示为“2020-03-29T20:00:00”^^xsd:dateTime。RDF中的数据类型节点称为文字，不允许有出边。RDF数据中常用的其他数据类型包括xsd:string、xsd:integer、xsd:decimal、xsd:boolean等。如果省略该数据类型，则假定该值为xsd:string类型。然后，基于RDF构建的应用程序可以识别这些数据类型，将它们解析为数据类型对象，并根据它们的标准定义应用等式检查、规范化、排序、转换和转换。在属性图的上下文中，Neo4j[332]还定义了一组关于属性值的内部数据类型，包括数字、字符串、布尔值、空间点和时间值。

**3.2.4词汇化。**实体的全局标识符有时会有一种人类可解释的形式，比如aschile:Santiago，但标识符字符串本身并不具有任何形式上的语义意义。在其他情况下，所使用的全局标识符可能无法设计人类的可解释标识符。例如，在维基数据中，智利圣地亚哥被确定为WD:Q2887，这样的方案具有提供更好的持久性和不偏向于特定人类语言的优势。例如，当该国家或地区从斯威士兰改名时，Eswatini的Wikidata标识符（wd:Q1050）不受影响，并且不需要在创建（更可读）IRIs的语言之间进行选择，例如wd:Eswatini（英语）、wd:Eswatini（斯威士兰）、wd:Esuatini（西班牙语）等。

由于标识符可以是任意的，因此通常会添加为提供人类可解释的边节点标签，例如wd:Q2887“Santiago”rdfs:label，指示人们在语言上如何引用主题节点。这种形式的语言信息在知识基础方面起着重要作用，这样用户可以更清楚地识别知识图谱中特定节点实际引用的真实世界哪个实体[115]；它还允许使用文本语料库交叉引用实体标签，以查找可能涉及给定实体的文档[317]。标签可以用别名（例如）或注释（例如）来补充，以进一步帮助确定节点的身份。

诸如“Santiago”之类的节点表示字符串文字，而不是标识符。取决于特定的图形模型，此类文字节点也可以定义为一对（s,l），其中s记录字符串和l表示一种语言代码；例如，在RDF中，我们可以声明chile:City“City”@enrds:label，chile:City“Ciudad”@esrds:label等，用不同的语言表示节点的标签。在其他模型中，可以指定相关的语言，例如，通过边缘上的元数据。带有人类可解释的标签、别名、注释等的知识图（在各种语言中）有时被称为（多语言）词汇化知识图谱[53]。

**3.2 5存在节点。**在对不完整信息建模时，我们可能在某些情况下知道图中必须存在与其他节点具有特定关系的特定节点，但无法识别所讨论的节点。例如，我们可能有两个在同一地点共同举办的活动：chile：EID42和chile：EID43，其地点尚未宣布。一种选择是简单地忽略场地边缘，在这种情况下，我们会丢失这些活动有场地，并且两个活动有相同场地的信息。另一种选择可能是创建一个新的IRI来表示场地，但从语义上来说，这与已知场地是无法区分的。因此，一些图模型允许使用存在节点，在这里表示为一个空白圆：

这些边缘表示chile:EID42和chile:EID42有一个共同的地点，但没有标识。RDF中支持存在节点作为空白节点[106]，空白节点也常用于支持图形中复杂元素的建模，如RDF列表[106、235]。图16举例说明了一个RDF列表，它使用链表结构中的空白节点来编码顺序。虽然存在节点可能很方便，但它们的存在可能会使图上的操作复杂化，例如确定两个数据图是否具有相同的结构存在节点[106、234]。因此，有人提出了在图中定义存在节点的方法——用规范标签替换它们[234，305]。其他作者则呼吁尽量减少在图形数据中使用此类节点[216]。

**3.3上下文**

图1的数据图中呈现的许多（可以说是所有）事实在特定的上下文中可以被认为是真实的。就时间背景而言，圣地亚哥自1541年以来仅作为一座城市存在，1956年从Aricato（阿里卡）飞往圣地亚哥，等等。就地理背景而言，图表描述了智利发生的事件。关于出处，与ID15有关的数据出自2020年1月4日的尼亚姆网页，因此据说是真实的。也可以使用其他形式的上下文。我们可以进一步结合上下文，比如根据《安康条约》（出处），表明自1883年（当时的）以来，阿里卡是智利的一座城市（地理）。

根据上下文，我们在这里指的是真理的范围，因此谈论的是一些数据被认为是真实的上下文[198、324]。图1的图表在很大程度上隐藏了上下文。然而，明确上下文可以从不同的角度解释数据，例如了解2016年的真实情况，不包括后来发现有虚假数据的网页的真实情况等。如前面的示例所示，可以在不同的层面上考虑图形数据的上下文：单个节点、单个边或边集（子图）上。我们现在讨论可以在不同级别明确上下⽂的各种表⽰。

**3.3.1.直接代表**。表示上下文的第一种方式是将其视为与其他数据没有差别的数据。例如，图1的事件 eId15的日期可以被视为表示一种时间上下文的形式，表明在时间范围内，诸如这样的边缘是真实的。另一个选项是将表示为边的关系（如）更改为节点，如图3所示，允许为该关系分配额外的上下文。虽然在这些示例中，上下文是以特定的方式表示的，但已经提出了一些规范，以更标准的方式将上下文表示为数据。时间本体[103]就是一个例子，它规定了如何以可互操作的方式在RDF图中描述时间实体、时间间隔、时间瞬间等，以及它们之间的关系，如之前、重叠等。另一个例子是PROV数据模型[181]，它指定了如何在RDF图中描述起源出处，其中实体（如图、节点、物理文档）从其他实体派生，由活动（如提取、作者身份）生成和/或使用，并归属于代理（如人员、软件、组织）。

**3.3.2物化**。通常，我们可能希望直接定义边缘本身的上下文；例如，我们可能希望声明边缘圣地亚哥航班阿里卡从 1956 年开始有效。虽然我们可以使用将边转化为节点的模式（如图3所示）来直接表示这样的上下文，但另一个选项是具体化，它允许以通用的方式（或者在图形的情况下，用于定义边的边）对语句进行陈述。在图17中，我们给出了三种具体化形式，可用于在有向边标记图[224]中对上述边的时间上下文进行建模。我们用e来表示一个任意的标识符，该标识符表示可以关联上下文信息的边缘本身。与直接表示不同，e表示的是边缘，而不是飞行。RDF具体化[106]（图17a）定义了一个新的节点来表示边缘，并将其连接到边缘的源节点（通过主题）、目标节点（通过对象）和边缘标签（通过谓词）。相比之下，n元关系[106]（图17b）将边的源节点直接连接到带有边标签的边节点e；然后将边的目标节点连接到e（通过值）。最后，单例属性[361]（图17c）将e用作边缘标签，将其连接到指示原始边缘标签的节点（通过单例）。文献中还提出了其他形式的物化，例如NdFluents[183]。一般来说，物化的边并不表示它具体化的边；例如，我们可以具体化一条边，说明它不再有效。我们参考了Ernández等人[224]的工作，以进一步比较具体化替代方案及其相对优势和劣势。

**3.3.3.更高元的表示**。作为物化的替代方法，我们可以使用更高的算术表示来建模上下文。再次讨论，图18展示了时间上下文的三种更高的算术表示。首先，我们可以使用命名图（图18a）来包含边，然后在图名上定义时间上下文。其次，我们可以使用一个属性图（图18b），其中时间上下文被定义为边缘上的属性。第三，我们可以使用RDF\*[208]（图18c）：RDF的一个扩展，允许将边定义为节点。在这些选项中，最灵活的是命名图表示法，我们可以通过将多条边放在一个命名图中，一次将上下文分配给多条边；例如，我们可以在图18a的命名图中添加更多边，这些边也从1956年起有效。最不灵活的选项是RDF\*，在没有边缘id的情况下，它不允许将不同的上下文值组分配给边缘的值；例如，考虑到，如果我们向该边缘添加四个上下文值，以说明其从2006年至2010年有效并且有效期为2014年至2018年，我们无法将这些值配对，而是可能需要创建一个节点来表示不同的总统（在其他模型中，我们可以使用两个命名图或边缘ID）。

**3.3.4.注释**。到目前为止，我们已经讨论了在图形中表示上下文，但我们还没有谈到自动推理上下文的机制；例如，如果从圣地亚哥出发的夏季航班只有季节性航班，我们可能希望从圣地亚哥出发寻找其他航线，以便在加利福尼亚州举办冬季活动。虽然公交车、航班等的日期可以直接在图中表示，或者使用具体化，但编写一个查询来手动与相应的时间上下文相交将是乏味的，甚至可能根本不可能。另一种选择是考虑注释，这些注释提供上下文域的数学定义，以及该域中可能的关键操作，然后可以自动应用。

一些注释为特定的上下文域建模；例如，时间RDF[203]允许以时间间隔注释边缘，例如，而Fuzzy RDF[477]允许诸如圣地亚哥之类的真实度来注释边缘，比如生地亚哥具有半干旱气候，表示它或多或少是正确的（度数为0.8）。

其他形式的注释是领域无关的；例如，带注释的RDF[125，503，546]允许表示各种形式的上下文建模为半环：由域值（例如，时间间隔、模糊值等）组成的代数结构，以及两个主要操作符来组合域值：满足和加入。我们在图19中提供了一个示例，其中G是使用简化时域中的值进行注释，该时域由代表一年中天数的整数集（1–365）组成。为了简洁起见，我们使用区间表示法，例如，{[150，152]}表示集合{150，151，152}。查询Q然后要求从圣地亚哥飞往有活动的城市的航班；此查询将检查并返回反映每个答案的时间有效性的注释。为了得出这些答案，我们首先需要在兼容航班和城市边缘上应用注释的结合，应用Meet运算符来计算两条边都适用的注释。在我们的场景中，定义meet的自然方式是一组的交集，例如，在事件注释{[150152]}和航班注释{[1120]，[220，365]}上应用meet会导致空时间间隔{}，这可能会导致从结果中筛选城市（取决于查询评估语义）。然而，在美国，我们发现了两个不同的非空交点：{[123，125]}用于eid16和{[276，279]}用于EID17。考虑到我们对城市（一个投影变量）感兴趣，而不是对事件感兴趣，因此我们可以使用连接运算符将这两个注释组合起来，返回其中一个结果为真的注释。在我们的场景中，定义连接运算符的自然方式是几组天的并集，给出{[123，125]，[276，279]}。我们在附录B.4.1中提供了正式定义。4.1基于Zimmermann等人[546]提出的图形注释的一般框架。[546]⽤于图上的注释。

**3.3.5其他上下文框架**。还提出了其他框架，用于对图形中的上下文进行建模和推理。一个值得注意的例子是上下文知识库[240]，它允许将单个（子）图分配给它们自己的上下文。与命名图不同，上下文是沿着一个或多个维度显式建模的，其中每个（子）图必须为每个维度取一个值。每个维度都进一步与其值的偏序相关联——例如2020-03-22⪯2020-03⪯2020年——允许选择和组合在不同粒度级别的上下文中有效的子图。Schuetz等人[442]提出了一种类似的上下文在线分析处理（OLAP）形式，它基于由维度构成的数据立方体，其中单个单元包含知识图。这样就可以支持“切片和切块”（根据给定维度选择知识）以及“汇总”（在更高级别上聚合知识）等操作。我们请读者参考相关文件以了解更多细节[240，442]。