arXiv:2304.13097v1 [cs.DB] 25 Apr 2023

图数据模型桥接：RDF、RDF-star 和属性图作为有向无环图 [扩展摘要]

Ewout Gelling 埃因霍温理工大学
ewoutgelling1999@gmail.com

George Fletcher 埃因霍温理工大学 g.h.l.fletcher@tue.nl

Michael Schmidt 亚马逊云科技
schmdtm@amazon.com

摘要

如今，图数据库用户面临两种技术栈的选择：一方面，资源描述框架（RDF）是一种内置语义的数据模型，最初由 W3C 开发，用于在 Web 上交换互联数据；另一方面，带标签属性图（LPG）旨在实现高效的图处理，并深深植根于开发者和工程社区。这两种模型从不同的抽象层看待图（RDF 中的三元组，与 LPG 中通过内嵌属性连接顶点的边），至少在表面上呈现出不同的特性，采用不同的查询语言，并各自融入独立的软件生态系统。

在这篇短文中，我们介绍了一种名为 陈述图 的新型统一图数据模型。该模型融合了 RDF 和 LPG 的特性，并在不同层面实现了互操作性：(a) 能够将 RDF 数据和 LPG 数据作为单个互联图进行管理；(b) 支持使用任意 RDF 或 LPG 查询语言查询集成图；同时 (c) 为不依赖图技术栈的数据交换机制和格式扫清了道路。我们将这一新模型形式化为有向无环图，并概述了 RDF、LPG 和陈述图之间的一套双向映射。这些映射隐式定义了统一数据模型上 RDF 和 LPG 查询语言的读取查询语义，从而让图用户能够根据各自的图用例，灵活选择使用的查询语言。

作为这些构想的概念验证，我们还介绍了 1GPlay-ground：一个基于陈述图概念构建的内存数据库管理系统，可存储 RDF 数据和 LPG 数据，并支持使用 SPARQL 和 Gremlin 进行跨模型查询。

1 引言

资源描述框架（RDF [32]）和带标签属性图（LPG）是当今工业图数据库领域中最常见的两种图数据模型 [24]。

RDF 已作为 W3C 语义网 [8] 计划的一部分得到开发和标准化。在其核心定义中，RDF数据集是一组（主语、谓词、宾语）三元组，其中每个三元组都表示一个事实，例如（*Alice*、认识、*Bob*）。RDF 三元组可以分组到称为命名图的容器中，而 RDF数据集则定义为此类图的集合。在 RDF 之上，W3C 开发并标准化了支持建模和推理的更高层语言（RDFS [36], OWL [28]），以及一种名为 SPARQL [31]的声明式查询语言。尽管 RDF 并未明确使用图概念（例如顶点和边）进行定义，但 RDF 与图之间存在密切联系：每个三元组都可以理解为一条边，它从带有给定谓词 标签的主语 节点指向一个宾语 节点。事实上，W3C 本身将 RDF 称为“基于图的数据模型”[32]，并利用图来直观地可视化 RDF（例如参见 [34]）——然而，它并未提供 RDF 与节点、边等常见图概念之间的形式化映射。

相比之下，带标签属性图通过图术语和概念进行形式化 [2, 3, 10]。LPG 数据模型由顶点集合及连接这些顶点的边集合组成，其中顶点和边都可以通过键值对（即属性）进行描述（例如 [10]）。与 SPARQL 不同，Gremlin [13] 和 openCypher [22] 等 LPG 查询语言提供了专门的构造，用于访问和操作顶点与边（在 openCypher 中，边被称为“关系”），并将它们视为一等概念。

使 LPG 成为现实世界用例中突出选择的一项具体特性，是其对边属性的内置支持；例如，连接法兰克福机场（FRA）和伦敦希思罗机场（LHR）的 路线 边所关联的距离，可以通过属性 *distance -> “655”*的形式附加到路线边上。RDF 中所有内容都是三元组，因此没有内置机制可以为“边”附加属性（严格来说，RDF 中甚至不存在边这一概念）。在 RDF 中对此类场景进行建模的指定机制是具体化词汇 [36], , 它允许表达“关于陈述的陈述”。在上述示例中，可以通过另外四个三元组来表示：描述两个机场之间路线的三元组具有某个距离：

(FRA，路线，LHR)，// 原始三元组 (s, rdf:subject, FRA)、(s, rdf:predicate, route)、(s, rdf:object, LHR)、(s, distance, 655)

在此示例中，s 是表示底层三元组 (FRA，路线，LHR) 的新标识符；它通过指向该三元组的三个位置来定义，分别使用 RDF 命名空间中的保留谓词 rdf:subject、rdf:predicate 和 rdf:object。随后，在最后一个三元组中使用标识符 s 来附加距离。虽然 RDF 具体化在概念上是合理的，但其冗长性和较差的可用性一直受到批评 [17, 23]。其他用于“围绕”RDF 中缺乏边属性支持这一问题进行建模的方法也存在类似的局限性：例如，N 元关系 [29], 会改变图的拓扑结构并使查询复杂化；使用命名图进行具体化（如 [26] 中所提出的）会占用图容器，导致该容器无法再用于其他用途。

为应对 RDF 中边属性方面的（易用性）差距，W3C 最近成立了 RDF-star 工作组[1]，旨在制定 RDF 扩展，为在 RDF 中表达边属性提供简洁、易用的语法（并支持在 SPARQL 中对其进行查询）。虽然该工作组致力于解决 RDF 的一个重要易用性问题，但其成果充其量只能弥合某项标准中现有的差距，却无法克服 RDF 与 LPG 属于两个相互独立的技术栈这一根本问题：它们从不同的抽象层次看待图。边属性

[1] https://www.w3.org/groups/wg/rdf-star

只是众多差异中的一个例子；二者各自都拥有非常独特的优势（和劣势）。仅举几例：（i）RDF 在全球数据交换、发布和图合并方面提供了出色的支持（例如，[9]）；（ii）LPG 查询语言 Gremlin 和 openCypher 原生支持将路径作为数据类型，而 SPARQL 内置了跨不同端点的联邦查询支持 [30]；（iii）RDF 和 LPG 都拥有各自的软件生态系统、开发者社区（例如 Apache TinkerPop [13]）以及围绕图处理构建的工具链。

我们认为，归根结底，图用户只想解决自己的图用例。不得不在两种模型之间做出选择，更糟的是，被锁定在两种技术栈之一中，这与他们在需求不断变化、机遇不断涌现的世界中保持灵活性的愿望直接冲突。为了让这些用户能够以最便捷的方式解决图问题，本文探索了一种通过新颖的统一数据模型实现图数据模型互操作性的方法。该数据模型具有足够的表达能力，可以同时表示 RDF、RDF-star 和属性图数据模型。接下来，我们将介绍两个具体的真实世界用例，说明这种统一图数据模型的价值。

用例：美食旅行。 航空机票行业的一家新初创公司希望为客户提供前往目的地的最具成本效益的旅行方式。他们决定使用属性图数据库系统作为其后端数据存储。在此场景中，机场被建模为节点，机场之间的航班被建模为边。每条边都携带航班费用信息，其形式为边属性。使用 openCypher 路径查询，他们可以找到前往客户目的地的最具成本效益的路线，而该路线不一定总是直飞。面对激烈的竞争，这家初创公司决定转向更为小众的市场，即美食爱好者市场。他们新的价值主张是提供前往米其林星级餐厅的最经济航班。他们知道，DBpedia [21] 知识图谱包含大量关于认证餐厅的信息，并计划将其整合到自己的系统中。然而，问题也随之而来：这些数据只能以 RDF 形式访问。在架构中加入三元组存储将导致额外的查询开销，并增加系统复杂性。将两个数据集置于同一个数据系统中会更加实用且性能更佳。遗憾的是，将 RDF 数据转换为 LPG 格式也被排除为一种选择，因为这会使未来整合其他相关 RDF 数据变得更加困难（例如，添加一个筛选条件，以筛选位于被视为历史建筑中的餐厅）。本质上，这家初创公司需要一种将 RDF 和 LPG 数据结合到单一可查询结构中的方法，同时保持原始数据的完整性。

用例：事件知识图谱分析。事件知晓-图（EKG） [12, 27] 是一种基于 LPG 的数据模型，与流程挖掘领域相关。它描述业务流程中事件的逻辑流。事件被建模为图节点，并带有描述性活动名称，例如“创建发票”或“接收付-款”。事件之间的转换被建模为有向边，可通过实体的形式关联其他信息。实体是图中的独立节点，具有唯一标识符。例如，“接收付款”事件与“结清发票”事件之间的转换，可能关联一个包含相关付款信息的实体。此关联通过在转换的边属性中引用实体标识符来建立。最终，可以使用 EKG 来发现业务流程中特定实体的生命周期：通过 openCypher 路径查询，聚合沿着引用该实体的转换路径上的事件节点。

请注意，EKG 的设计受到底层数据模型特性与所需查询语言之间不匹配的影响。该领域要求将实体节点与事件转换边关联起来，而属性图模型本身并不支持这一功能。为了解决这一问题，系统在转换的边属性中引用实体。然而，这种间接链接可能在后续引发数据完整性问题。更自然的方法是使用 RDF-star 对该领域进行建模，在 RDF-star 中，实体可以通过具体化直接与转换建立链接。然而，这会使模型的访问更加复杂，因为 SPARQL-star 查询语言不支持在任意长度的路径遍历中收集节点。从本质上说，EKG 再次需要一种方式，将 RDF 和 LPG 技术栈中的关键功能结合到单一的可查询结构中。

2 贡献

为了应对这些在各类应用领域中普遍出现的实际用例，在这篇短文中，我们做出了以下贡献：

- 陈述图数据模型的形式化；一种适用于 RDF、RDF-star 和属性图的统摄性数据模型。
- 一个名为“1G Playground”[2]的开源陈述图概念验证实现；一种同时支持 RDF 和属性图，并允许通过 SPARQL 和 Gremlin 进行跨模型查询的内存数据库管理系统。

我们的贡献为不依赖图技术栈的图数据管理、查询和交换奠定了坚实基础，并为其发展铺平了道路。

3 背景与问题描述

人们曾尝试通过直接映射[1, 11, 16, 25]来定义 表层数据模型（RDF、RDF-star 和 LPG）之间的互操作性。这些方法主要利用了 RDF（-star）三元组和属性图边均为三元关系这一事实来转换信息。然而，由于这些模型的功能集并不完全兼容，如果不引入额外的语义含义，这类转换本质上无法做到无损。因此，我们采取了不同的图模型互操作性方法：我们不定义数据模型之间的直接映射，而是将这三种模型提升到一个共同的抽象层次，并将其用作跨数据模型映射中的中间层。

我们首先为这三种表层数据模型给出形式化定义。在此，我们将使用术语 具体类型 和具体元素。具体类型是某种

[2]https://github.com/aws-samples/amazon-neptune-samples/tree/master/1g- 游乐场

任意数据模型的一个构建块。设 $E_1, \ldots, E_k$ 表示某个 $k > 0$ 的具体类型。称元素 $e \in E_1, \ldots, E_k$ 为一个具体元素。

具体类型的一个示例是所有属性图标签 K 的集合。其中，标签 $city \in K$ 是具体元素的一个示例。在定义中，我们使用以下 6 种具体类型：字面量 L、IRI I、空白节点 B、标签 K、属性名称 P 和属性值 V。

RDF 三元组是一个元组 $t = (s, p, o)$，其中 $s \in I \cup B$ 被称为主语，$p \in I$ 被称为谓词，而 $o \in I \cup B \cup L$ 被称为宾语。RDF 图是由零个或多个 RDF 三元组组成的有限集合。RDF 数据集是如下形式的集合 $\{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$，其中 $0 \leq i \leq n$，且 G_D 和 G_i 都是 RDF 图，$g_i \in I \cup B$。图 G_D 被称为默认图，形式为 (g_i, G_i) 的对被称为命名图[34]。表 1 包含一个 RDF 数据集示例，其中对一个虚构餐厅的 DBpedia[21] 片段进行了建模。

表 1：包含餐厅数据的 Turtle[35] RDF 数据集。

RDF -（默认图）
@prefix ex: <http://example.org/> 。 @prefix dbr: <http://dbpedia.org/resource/> 。 @prefix dbp: <http://dbpedia.org/property/> 。 @prefix dbo: <http://dbpedia.org/ontology/> 。 ex:ChezSG a dbr:Restaurant; dbo:cuisine "法式，古典"@en; dbp:rating "米其林指南"@en ; dbp:city dbr:巴黎 . dbr:巴黎 a dbp:city ; dbp:name "巴黎"@en .

一个 *RDF-star* 三元组是一个元组 $t = (s, p, o)$，其中 $s \in I \cup B \cup \{t_1\}, p \in I$，以及 $o \in I \cup B \cup L \cup \{t_2\}$，其中 t_1 和 t_2 是 RDF-star 三元组（改编自 [17]）。RDF-star 三元组中不允许存在引用循环，即 $t \neq t_1$，t_2 也不能在 t_1 中引用 t，或递归地引用 t_2。随后，以类似于 RDF 对应概念的方式构造 *RDF-star* 图和 *RDF-star* 数据集的定义。

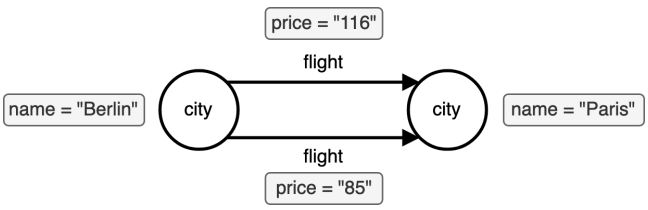

图 1：航班数据的属性图。

属性图的工作定义改编自 Tomaszuk 等人 [25]。该定义具有广泛的概括性和包容性，允许节点和边关联可变数量的标签和属性。属性图是一个元组：$G = (N, A, P, \delta, \lambda, \sigma, \rho)$，其中：

(1) N 是一个有限节点集，A 是一个有限边集，P 是一个有限属性集，并且 N、A、P 两两不相交。(2) δ：$A \to (N \times N)$ 是一个全函数，将 A 中的每条边关联到 N 中的一对节点；(3) λ：$(N \cup A) \to 2^K$ 是一个全函数，将节点和边关联到一个标签集（可能为空）；(4) σ：$(N \cup A) \to 2^P$ 是一个全函数，将节点和边关联到一个属性集（可能为空），并满足对于每一对 o_1、$o_2 \in dom(\sigma)$，$\sigma(o_1) \cap \sigma(o_2) = \emptyset$；(5) ρ：$P \to (P \times V)$ 是一个全函数，为每个属性分配一个属性名值对。

图 1 展示了一个属性图示例，其中柏林与巴黎之间的两趟航班被建模为边。

4 陈述图

陈述图数据模型必须足够灵活，以同时捕捉全部三种表层数据模型的独特特征。实现这一点的方法是使陈述图成为一种语义不可知的数据模型，也就是说，不对数据语义施加任何约束。我们将陈述图形式化为有向无环图（定义 4.1），其中每个内部节点恰好具有三条出边。这些边带有固定标签，用于指示它们所指向的顶点的顺序。陈述图的每个叶节点都与一个具体元素相关联，而每个内部节点都与一个所谓的陈述标识符相关联（该标识符属于陈述标识符的具体类型 S）。随后，一个内部顶点及其出邻居实质上构成一条可识别信息，因此我们将其称为 陈述。最后，内部节点可以直接连接到其他内部节点，但其连接边不得标记为 *predicate*。请注意，我们在此并未暗示任何物理实现策略；陈述图纯粹是一种逻辑模型和概念模型。图用户不会直接与陈述图交互，而是可以便捷地使用自己选择的表层数据模型。

定义 *4.1*。 A 语句图 G 是一个有向顶点边标记图 $G = (V, E, \tau)$，其中：

- V 是一个有限顶点集；
- $E \subseteq V \times \{subject, predicate, object\} \times V$ 是一个有限标记边集；
- τ：$V \to S \cup E_1 \cup \ldots \cup E_k$ 是一个为顶点标记陈述标识符或其他具体元素的全单射函数；

- 每个顶点 $v \in V$ 均满足以下条件：(1) 如果 $\tau(v) \in E_1 \cup \cdots \cup E_k$，则：(a) v 没有出边；(b) v 至少有 1 条入边。(2) 如果 $\tau(v) \in S$，则 v 恰好有 3 条出边：(a) $(v, subject, v_s)$，且 $\tau(v_s) \in S \cup E_1 \cup \cdots \cup E_k$；(b) $(v, predicate, v_p)$，且 $\tau(v_p) \in E_1 \cup \cdots \cup E_k$；(c) $(v, object, v_o)$，且 $\tau(v_o) \in S \cup E_1 \cup \cdots \cup E_k$。
- G 是无环的，即 E 中不存在从某个顶点到其自身的路径。

图2和图3各展示了一个语句图的示例。在这些图中，*subject*边用空心箭头表示，*predicate*边用双箭头表示，*object*边用单箭头表示。请注意，我们也可以采用另一种方法来形式化陈述图，例如将其定义为一类嵌套超图[19]。为便于阐述，我们采用有向无环图的观点。

5 具体语句图

本节将介绍适用于表层数据模型的简洁语句图片段，我们将其称为语句图图像。这些图像旨在做到最简，即每个图像只体现其对应表层数据模型的特征，不包含其他内容。语句图图像与其对应的表层数据模型之间存在无损双向映射，这表明它们在功能上等价。为简洁起见，这套映射未在此列出，但收录于本扩展摘要完整版的附录中[15]。

5.1 RDF数据

在以下定义中，我们会用到语句图中的顶点同构概念。如果两个内部顶点都指向相同的顶点，我们就认为它们是同构的；它们本质上表达了“相同”的陈述。这样便可以在语句图中表达RDF（星号）的集合语义。

RDF-star 的特征通过一个*RDF-star*-图像（定义 5.1）来体现，其中将陈述分为两组：三元组陈述和成员关系陈述。三元组陈述体现 RDF-star 三元组的语义；此处通过允许在*subject*和*object*位置引用陈述标识符来支持具体化（参见约束 1、2 和 3）。成员关系陈述则将一个三元组陈述指定给某个图名称或默认图标识符 $G_D \in \mathrm{I}$。这些陈述的 *predicate* 位置使用常量具体元素 $in \in \mathrm{I}$。最后一项约束（4）首先说明成员关系陈述的结构（4a），然后断言每个三元组陈述至少在另一个陈述的上下文中被引用（4b），并通过禁止顶点同构来确保陈述唯一性（4c）。

定义 *5.1*。语句图是*RDF-star*-图像，当：
(1) 对所有边 $(v, subject, v_s) \in E, \tau(v_s) \in \mathrm{S} \cup \mathrm{I} \cup \mathrm{B}$；(2) 对所有边 $(v, predicate, v_p) \in E, \tau(v_p) \in \mathrm{I}$；(3) 对所有边 $(v, object, v_o) \in E, \tau(v_o) \in \mathrm{S} \cup \mathrm{I} \cup \mathrm{B} \cup \mathrm{L}$；(4) 对于每个内部顶点 $v \in V$，设 v_s、v_p、v_o 是从 v 发出的具有入边 *subject*、*predicate*、*object* 的顶点，则满足以下约束：(a) 如果 $\tau(v_p) = in$，则 $\tau(v_s) \in \mathrm{S}$，且 $\tau(v_o) \in \mathrm{I} \cup \mathrm{B}$，并且 v 没有入边；(b) 如果 $\tau(v_p) \in \mathrm{I} \setminus in$，则 v 至少有 1 条入边；(c) 不存在与 v 同构的 $v' \in V$。

一种*RDF* 映像（定义 5.2）是 RDF-star-图像的一种特殊情况，其中内部节点只能在成员关系语句的上下文中被引用。因此，除继承的约束外，每个 RDF 映像还必须满足另外两个约束：(1) 内部节点绝不能在 *object* 位置引用其他节点；(2) 每个内部节点都必须至少有一条入边来自

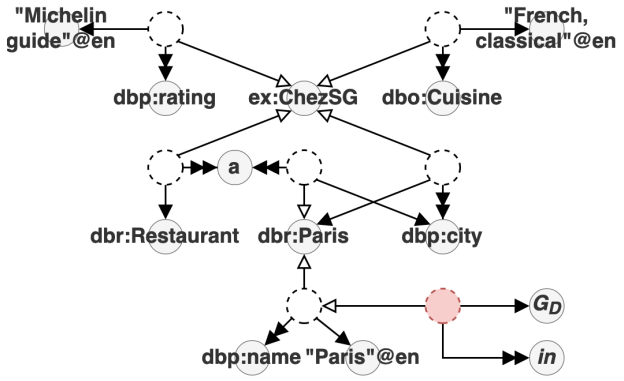

图 2：餐厅数据的语句图。这是表 1 中给出的 RDF 数据集的 RDF 映像。

成员关系语句（即属于至少一个图）。图 2 展示了一个语句图，它是表 1 中数据集的 RDF-图像。在此图中，标红的节点是成员关系语句；为避免图面过于杂乱，其他成员关系语句均未显示。

定义 *5.2*。当语句图是 RDF-star-图像且满足以下条件时，它是一个*RDF* 映像：
(1) 对于所有边 $(v, \ object, \ v_o) \in E, \tau(v_o) \in \mathrm{I} \cup \mathrm{B} \cup \mathrm{L}$；
(2) 对于每个内部顶点 $v \in V$，设 v_s, v_p, v_o为从 v引出的、带有入 *subject*、*predicate*、*object*边的顶点，则以下约束成立：(a) 如果 $\tau(v_p) \in \mathrm{I} \setminus in$，则 $\tau(v_s) \in \mathrm{I} \cup \mathrm{B}$ 且 $\tau(v_o) \in \mathrm{I} \cup \mathrm{B} \cup \mathrm{L}$，并且 v 至少有 1 条入边。

5.2 LPG 数据

设 N 为属性图节点标识的具体类型。为简单起见，我们假设每个属性图节点都与该集合中的一个具体元素相关联。属性图的特征通过一个*LPG* 映像（定义 5.3）来描述，其中语句被区分为三组：边语句（4a）、节点标签语句（4b）和属性语句（4c）。具体元素 $l_p \in 2^{\mathcal{K}}$用在节点标签语句的 *predicate* 位置。图 3 展示了一个语句图，它是图 1 中属性图的 LPG 映像。在该图中，*n1* 和 *n2* 是来自 N 的元素。

定义 *5.3*。语句图是一个*LPG* 映像，当：
(1) 对所有边 $(v, subject, v_s) \in E, \tau(v_s) \in \mathrm{N} \cup \mathrm{S}$；(2) 对所有边 $(v, predicate, v_p) \in E, \tau(v_p) \in \mathrm{P} \cup 2^{\mathrm{K}}$；(3) 对所有边 $(v, object, v_o) \in E, \tau(v_o) \in \mathrm{V} \cup \mathrm{N} \cup 2^{\mathrm{K}}$；(4) 对每个内部顶点 $v \in V$，令 v_s、v_p、v_o 为从 v 出发、通过入边 *subject*、*predicate*、*object* 相连的顶点，则以下约束中恰有一个成立：(a) 如果 $\tau(v_p) \in 2^{\mathrm{K}} \setminus\setminus \ l_p$，则 $\tau(v_s) \in \mathrm{N}$ 且 $\tau(v_o) \in \mathrm{N}$；(b) 如果 $\tau(v_p) = l_p$，则 $\tau(v_s) \in \mathrm{N}$ 且 $\tau(v_o) \in 2^{\mathrm{K}}$，并且 v 没有入边；(c) 如果 $\tau(v_p) \in \mathrm{P}$，则 $\tau(v_s) \in \mathrm{S} \cup \mathrm{N}$ 且 $\tau(v_o) \in \mathrm{V}$，并且 v 没有入边。

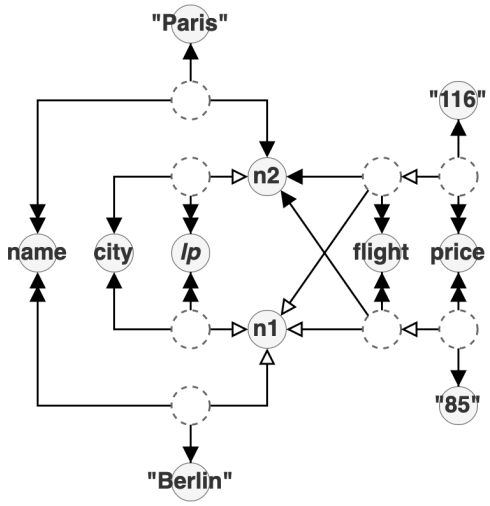

图 3：航班数据的语句图。这是图 1 中属性图的 LPG 映像

5.3 OneGraph 数据

一个 *OneGraph* 映像 （定义 5.4）是由 RDF 映像、RDF-star 映像和 LPG 映像组合而成的语句图。下一节将使用它来简化关于互操作性的推理。“OneGraph 映像”这一名称源自 Lassila 等人 [20] 提出的 OneGraph（1G）愿景。正如 [14], 所示，我们对语句图，尤其是 OneGraph 映像的形式化，是 OneGraph 愿景的一种具体实现。

定义 *5.4*。 语句图 G 是 *OneGraph* 映像，当 G 的每个组件都是 RDF-star-图像或 LPG 映像时，不考虑顶点同构的约束。

6 互操作性

我们已经展示了，陈述图充当表层数据模型的公共抽象层，使得可以在单一结构下聚合这些模型的实例。本节将重点讨论陈述图如何促进它们之间的互操作性。我们通过一系列映射来说明这一点，这些映射展示了如何将 OneGraph 映像转换为 RDF 映像、RDF-star-图像或 LPG 映像。这些映射隐式定义了表层数据模型之间的跨模型读取查询语义。例如，可以先将 RDF 数据集转换为 RDF 映像、 LPG 映像和属性图，然后使用 Gremlin 对其进行查询。

这些映射使用了以下全函数，用于关联 RDF 世界与属性图世界中自然相似的具体类型。它们是：$literalvalue: \mathrm{L} \mapsto \mathrm{V}$、$blank2node: \mathrm{B} \mapsto \mathrm{N}$、$IRI2node: \mathrm{I} \mapsto \mathrm{N}$、$IRI2prop: \mathrm{I} \mapsto \mathrm{P}$，以及 $IRI2label: \mathrm{I} \mapsto \mathcal{K}$。每个函数也都有一个逆函数，用于实现相反的目标，例如 $value2literal: \mathrm{V} \mapsto \mathrm{L}$，等等。这些函数用于转换与给定语句图的叶节点相关联的具体元素。它们的条目作为映射的参数。例如， $literal2value$ 中的一个条目，将字面量 "Paris" @en 与属性值 "Paris" 关联起来。

在以下转换中，顶点的移除是递归的；也就是说，移除一个顶点时，所有指向该顶点的出边所关联的顶点也都会被移除。

6.1 OneGraph 映像到 RDF-(star)-映像

如果执行可选步骤 (5)，以下映射会将 OneGraph 映像 $G = (V, E, \tau)$ 转换为 RDF 映像；如果跳过可选步骤，则转换为 RDF-star 映像。

(1) 对于每个叶节点 $v \in V$，在需要时使用相应的类型转换函数转换 $\tau(v)$，并合并具有等价值的节点 τ；(2) 将顶点 v_{in}和 v_{G_D}添加到 V中；如果这些顶点尚不存在于 V中，则添加 $\tau(v_{in}) = in$和 $\tau(v_{G_D}) = G_D$；(3) 对于每个内部顶点 $v \in V$，将顶点 v_{st}添加到 V，将 $\tau(v_{st}) \in \mathrm{S}$，并将边 $(v_g, subject, v)$、$(v_g, predicate, in)$ 和 $(v_g, object, v_{G_D})$ 添加到 E；(4) 对于每个内部顶点 $v \in V$，移除 V中与 v同构的顶点；(5)（可选）对于每个内部顶点 $v \in V$，令 v_s、v_p、 v_o为从 v引出的、具有入 $subject$、 $predicate$、 $object$边的顶点。如果 v_s或 v_o是内部顶点，且 $\tau(v_p) \neq in$，则移除 v；(6) 移除 V中没有入边或出边的顶点。

通过此映射得到的表 1 中 RDF 数据集的 RDF 映像如图 2 所示。

6.2 OneGraph 映像到 LPG 映像

以下映射将 OneGraph 映像 $G = (V, E, \tau)$ 转换为 LPG 映像。

(1) 对于每个叶节点 $v \in V$，如有需要，使用适当的类型转换函数转换 $\tau(v)$，并合并具有等价值的节点以得到 τ。遇到 $\tau(v) \in \mathrm{I}$ 时：(a) 如果 v具有入边 $subject$或 $object$，且没有入边 $predicate$；则设置 $\tau(v) = IRI2node(\tau(v))$；(b) 如果 v 具有入边 $predicate$ ，且没有入边$subject$或 $object$；则设置 $\tau(v) = IRI2label(\tau(v))$；(c) 如果 v具有入边 $subject$ 或 $object$，且具有入边 $predicate$ ； (i) 将节点 v添加到 V，令 $\tau(v) = IRI2node(\tau(v))$；(ii) 令 $\tau(v) = IRI2label(\tau(v))$；(iii) 令所有 $predicate$边指向 v，令所有 $subject$和 $object$ 边指向 v。(2) 对于每个内部顶点 $v \in V$，令 v_s、v_p、 v_o 为从 v 引出的、分别带有入边 $subject$、 $predicate$、 $object$ 的顶点：(a) 如果 v从某个顶点 v 接收入边，则在满足以下任一条件时移除 v ：(i) 指向 v 的入边未标记为 $subject$；(ii) $\tau(v_o) \notin \mathrm{N}$；(iii) v没有指向某个顶点 v_o 的 $object$边，且该顶点满足 $\tau(v_o) \in \mathrm{V}$。(3) 移除 V 中没有入边或出边的顶点。

通过此映射获得的图 1 属性图的 LPG 映像如图 3 所示。

7 1G Playground

接下来，我们介绍 Statement Graphs 的开源实现，并将其称为“1G Playground”，同时重点阐述启发这项工作的 One-Graph 愿景。开源的 1G Playground[3] 以本文的基础性工作为补充，提供了一个实用且可演示的产品，用于阐释 Statement Graphs 的关键理念，并作为概念验证。它由两个遵循传统客户端-服务器模型的组件组成。在客户端，我们提供了一个类似于 psql4 的 REPL 环境；服务器端则充当一个简单的内存数据库管理系统（见图 4）。服务器通过 REST 接口公开以下功能的端点：数据加载、数据检索、查询以及当前设置的修改。数据可以通过多种不同的 RDF 和属性图格式进行加载和检索，例如：N-Quads[33], Turtle[35],Graphson5，以及其他许多格式。Playground 还提供了一种新颖的共享序列化格式，允许用户以统一的“1G”语法导出其 RDF 和 LPG 数据（可通过标志 *-og* 使用）。

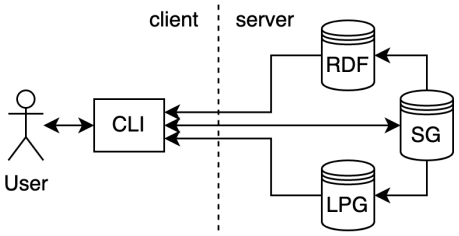

图 4：1G Playground 的架构

该 Playground 的服务器组件由单个 OneGraph 映像组成（图中以 SG 表示），并配有两个派生数据存储：一个用于 LPG（由 Tinker-Pop 支持），另一个用于 RDF（由 RDF4J[6] 支持）。用户无法直接修改这些数据存储，只能写入语句图本身。LPG 和 RDF 输入首先会被转换为相应的映像，然后再添加到中心图中。第 6 节所述的转换随后用于根据更新后的语句图重新加载这两个派生数据存储。该 Playground 允许用户配置 *IRI2label*、*label2IRI*、*IRI2prop* 和 *prop2IRI* 函数，从而让用户能够在一定程度上控制这些转换。所有其他具体类型转换（例如 *literal2value*）都会隐式执行。借助这一架构，Playground 可以在语句图上执行 SPARQL 和 Gremlin 读取查询，而无需从头实现自定义查询引擎。

该演练场包含一组交互式演示场景，这些场景由我们开发，用于帮助用户理解该工具，并展示语句图及其功能。使用 *scenario* 命令可以探索三个场景，更多相关信息请参见 GitHub 页面。

[3]https://github.com/aws-samples/amazon-neptune-samples/tree/master/1g- 游乐场 [4]https://www.postgresql.org/docs/current/app-psql.html [5] https://tinkerpop.apache.org/docs/3.4.1/dev/io/#graphson [6]https://rdf4j.org/

8 相关工作

RDF、RDF-star 与带标签属性图之间的映射。近年来，人们投入了相当多的精力来设计 RDF、RDF-star 和 LPG 之间直接的模型间映射，例如 [1, 11, 16, 25]。我们的工作通过提供一种统摄性数据模型，支持在所有这些表层数据模型之间进行统一转换，从而区别于并丰富了这方面的研究。此外，从数据模型的视角出发，有助于更为根本地理解图是什么，以及以 RDF、RDF-star 和 LPG（以及其他形式）建模的图之间有哪些共性与差异。

统一（图）数据模型。图建模（例如， [4, 10, 18]）以及统一“元”数据模型（例如， [6, 7]）是数据管理研究中的经典主题。与我们研究最密切相关的是 OneGraph [20] 和多层图 [5]方案，它们同样采用以陈述为中心的方法来统一 RDF、RDF-star 和 LPG。我们的方法通过将语句图视为有向无环图，为这项工作提供了补充性视角；有向无环图易于理解、直观，且便于操作和推理。

9 结论与未来工作

我们认为，图用户只希望解决自己的图用例，而在当代图工作中经常需要在多个图数据模型和图技术栈之间进行衔接，这与这一愿望直接冲突。在本文中，我们为设计和构建能够化解这一冲突的图系统奠定了坚实基础：通过语句图这一直观而简单的统一图数据模型，衔接 RDF 与属性图的世界。我们还介绍了开源工具包 1G Playground，展示了语句图的可行性和实用价值。

我们的工作引出了许多有趣的研究挑战。一个主要方向是开发和研究围绕基于语句图的系统构建的更广泛框架，例如查询语言与查询重写、模式与约束语言及其实施方法、索引策略以及序列化格式。在这些未来工作方向中的许多方面，进一步开发开源工具 1G Playground，可能是一条具体可行的推进路径。

参考文献

[1] Ghadeer Abuoda、Daniele Dell’Aglio、Arthur Keen 和 Katja Hose。2022。《将 RDF-star 转换为属性图：转换方法的初步分析——扩展版》。载于*QuWeDa* 知识图谱存储、查询与基准测试研讨会。[2] Renzo Angles、Angela Bonifati、Stefania Dumbrava、George Fletcher、Alastair Green、Jan Hidders、Bei Li、Leonid Libkin、Victor Marsault、Wim Martens、Filip Murlak、Stefan Plantikow、Ognjen Savkovic、Michael Schmidt、Juan Sequeda、Slawek Staworko、Dominik Tomaszuk、Hannes Voigt、Domagoj Vrgoc、Mingxi Wu 和 Dusan Zivkovic。2023。《PG-Schema：属性图的模式》。载于*SIGMOD*。[3] Renzo Angles、Angela Bonifati、Stefania Dumbrava、George Fletcher、Keith W. Hare、Jan Hidders、Victor E. Lee、Bei Li、Leonid Libkin、Wim Martens、Filip Murlak、Josh Perryman、Ognjen Savkovic、Michael Schmidt、Juan F. Sequeda、Slawek Staworko 和 Dominik Tomaszuk。2021。《PG-Keys：属性图的键》。载于*SIGMOD*。2423–2436。[4] Renzo Angles 和 Claudio Gutierrez。2008。《图数据库模型综述》。*ACM*计算调查40，1，文章 1（2008），39 页。[5] Renzo Angles、Aidan Hogan、Ora Lassila、Carlos Rojas、Daniel Schwabe、Pedro Szekely 和 Domagoj Vrgoč。2022。《多层图：图数据库的统一数据模型》。载于第 *5* 届 *ACM SIGMOD* 联合国际会议论文集

图数据管理经验与系统研讨会（GRADES）和网络数据分析（NDA）。计算机协会，文章 11，6 页。[6] Paolo Atzeni、Giorgio Gianforme 和 Paolo Cappellari。2009。《通用元模型及其字典》。大型数据与知识中心系统汇刊 1（2009），38–62。[7] Paolo Atzeni 和 Riccardo Torlone。1993。《一种用于管理多模型和转换模式的元模型方法》。信息系统 18，6（1993），349–362。[8] Tim Berners-Lee、James Hendler 和 Ora Lassila。2001。《语义网》。Scientific American 284，5（2001），34–43。[9] Christian Bizer、Tom Heath、Kingsley Idehen 和 Tim Berners-Lee。2008。《万维网上的链接数据（LDOW2008）》。载于第 17 届万维网国际会议论文集。1265–1266。[10] Angela Bonifati、George Fletcher、Hannes Voigt 和 Nikolay Yakovets。2018。《图查询》。Morgan & Claypool 出版社。[11] Julian Bruyat、Pierre-Antoine Champin、Lionel Médini 和 Frédérique Laforest。2021。《PREC：属性图的语义转换》。第 1 届图上的圆化方形研讨会（ 2021 2110 12996 1 ）。[12] https://doi.org/arXiv: . v 2021 Stefan Esser 和 Dirk Fahland。. 图数据库中的多维事件数据。《数据语义期刊》，109–141。https://doi.org/10.1007/s13740-021-00122-1[13] Apache 软件基金会。 [无日期。]。Apache TinkerPop：Gremlin 查询语言。检索于 2023 年 2 月 26 日，网址：https://tinkerpop.apache.org/gremlin。 html[14] Ewout Gelling。2022。《图数据模型桥接：RDF、RDF-Star 和作为有向无环图的属性图》。硕士论文。埃因霍温理工大学。[15] Ewout Gelling、George Fletcher 和 Michael Schmidt。2023。《图数据模型桥接：RDF、RDF-star 和作为有向无环图的属性图》 [扩展摘要]。https://github.com/EwoutGelling/Bridging-data-models [16] Olaf Hartig。2014。《RDF* 与属性图的协调》。arXiv:1406.3399v3（ 2014 10 48550 1409 3288 ）。https://doi.org/ . /ARXIV. . [17] Olaf Hartig 和 Bryan Thompson。 2014 . RDF 中具体化的另一种方法的基础。arXiv 预印本 arXiv:1406.3399（2014）。[18] Borislav Iordanov。2010。《HyperGraphDB：广义图数据库》。Web-Age 信息管理（2010），25–36。[19] Cliff Joslyn 和 Kathleen Nowak。2017。《超图：递归超图结构的定义》。arXiv:1704.05547v1（2017）。https://doi.org/10.48550/ ARXIV.1704.05547[20] O. Lassila、M. Schmidt、B. Bebee、D. Bechberger、W. Broekema、A. Khandelwal、K. Lawrence、R. Sharda 和 B. Thompson。2023。《OneGraph 愿景：打破图模型锁定的挑战》。语义网期刊 14（2023）。第 1 期。[21] Pablo N Mendes、Max Jakob 和 Christian Bizer。2012。《DBpedia：多语言跨领域知识库》。载于第八届语言资源与评测国际会议论文集（LREC'12）。欧洲语言资源协会（ELRA），1813–1817。http://www.lrec-conf.org/proceedings/ lrec2012/pdf/570_Paper.pdf [22] Neo4j 公司。 [无日期。]。openCypher。检索于 2023 年 2 月 26 日，网址：http://opencypher. org/ [23] Vinh Nguyen、Olivier Bodenreider 和 Amit Sheth。2014。《不喜欢 RDF 具体化？使用单例属性来表达关于陈述的陈述》。载于第 23 届万维网国际会议论文集。759–770。[24] Yuanyuan Tian。2023。《从行业视角看图数据库世界》。ACM SIGMOD Record 51，4（2023），60–67。[25] Dominik Tomaszuk、Renzo Angles 和 Harsh Thakkar。2020。《PGO：在 RDF 中描述属性图》。IEEE Access 8 （ 2020 ），118355–118369。https://doi.org/ 10.1109/access.2020.3002018[26] Johannes Trame、Carsten Keßler 和 Werner Kuhn。2013。《链接数据与时间——通过事件为研究者生命线建模》。载于《空间信息理论：第 11 届国际会议 COSIT 2013》，英国斯卡伯勒，2013 年 9 月 2 日至 6 日。论文集 11。Springer，205–223。[27] W.M.P. van der Aalst 和 J. Carmona。2022。《流程挖掘手册》。（2022），274–320。[28] W3C。2004。《OWL Web 本体语言》。https://www.w3.org/TR/owl-ref/ [29] W3C。2006。《在语义网上定义 N 元关系》（W3C 工作组说明）。https://www.w3.org/TR/swbp-n-aryRelations/ [30] W3C。2013。《SPARQL 1.1 联邦查询》。https://www.w3.org/TR/sparql11- federated-query/ [31] W3C。2013。《SPARQL 1.1 查询语言》。https://www.w3.org/TR/sparql11-query/ [32] W3C。2014。《RDF 1.1 概念与抽象语法》。https://www.w3.org/TR/ rdf11-concepts/ [33] W3C。2014。《RDF 1.1 N-Quads》。https://www.w3.org/TR/n-quads/ [34] W3C。2014。《RDF 1.1 入门》。https://www.w3.org/TR/rdf11-primer/ [35] W3C。2014。《RDF 1.1 Turtle》。https://www.w3.org/TR/turtle/ [36] W3C。2014。《RDF Schema 1.1》。https://www.w3.org/TR/rdf-schema/

10 附录

10.1 属性图到语句图

可以将以下映射应用于属性图 $PG=(N, A, P, \delta, \lambda, \sigma, \rho)$，以获得 LPG 映像 $G=(V, E, \tau)$

(1) 对于每个 $n \in N$：(a) 将顶点 v_s、v_p、v_o 和 v_{st} 添加到 V，令 $\tau(v_s)=n$、$\tau(v_p)=l_p$、$\tau(v_o)=\lambda(n)$，并令 $\tau(v_{st})$ 为 S 中一个未使用的元素 (b) 将边 $(v_{st}, subject, v_s)$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E。(c) 对于每个属性 $(p_{name}, p_{value}) \in \rho(\sigma(n))$：(i) 将顶点 v_p、v_o 和 v_{st} 添加到 V，令 $\tau(v_p)=l_p$、$\tau(v_o)=\lambda(n)$，并令 $\tau(v_{st})$ 为 S 中一个未使用的元素 (ii) 将边 $(v_{st}, subject, v_s)$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E。(2) 对于每个 $e \in A$，其中 $\delta(e)=(n_1, n_2)$：(a) 将顶点 v_s、v_p、v_o 和 v_{st} 添加到 V，令 $\tau(v_s)=n_1$、$\tau(v_p)=\lambda(e)$、$\tau(v_o)=n_2$，并令 $\tau(v_{st})$ 为 S 中一个未使用的元素 (b) 将边 $(v_{st}, subject, v_s)$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E。(c) 对于每个属性 $(p_{name}, p_{value}) \in \rho(\sigma(e))$：(i) 将顶点 v_p、v_o 和 v_{st} 添加到 V，令 $\tau(v_p)=l_p$、$\tau(v_o)=\lambda(n)$，并令 $\tau(v_{st})$ 为 S 中一个未使用的元素 (ii) 将边 $(v_{st}, subject, v_{st})$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E。(3) 对于 V 中具有相同 τ 值的顶点进行合并；

10.2 语句图到属性图

可以将以下映射应用于 LPG-Image $G=(V, E, \tau)$，以获得属性图 $PG=(N, A, P, \delta, \lambda, \sigma, \rho)$。

(1) 令 V_{st} 为 V 中的内部顶点集合。在遍历 V_{st} 时，令 v_s、v_p、v_o 为从一个 $v \in V_{st}$ 接收传入 $subject$、$predicate$、$object$ 边的顶点。(2) 对于每个满足 $\tau(v_p)=l_p$ 的 $v \in V_{st}$：(a) 令节点 $n=\tau(v_s)$，将 n 添加到 N，设置 $\lambda(n)=\tau(v_s)$。(3) 对于每个满足 $\tau(v_s) \in \mathrm{N} \wedge \tau(v_o) \in \mathrm{N}$ 的 $v \in V_{st}$：(a) 令节点 $n_1=\tau(v_s)$ 和 $n_2=\tau(v_o)$，将边 $e=(n_1, n_2)$ 添加到 A，设置 $\lambda(e)=\tau(v_p)$。(4) 对于每个满足 $\tau(v_o) \in \mathrm{V}$ 的 $v \in V_{st}$：(a) 如果 $\tau(v_s) \in \mathrm{N}$，则令节点 $n=\tau(v_s)$，将属性 p 添加到 P，并添加条目 $\sigma(n)=p$ 和 $\rho(p)=(\tau(v_p)\ \tau(v_o))$。(b) 如果 $\tau(v_s) \in \mathrm{S}$，则获取先前为 v_s 创建的边 e，将属性 p 添加到 P，并添加条目 $\sigma(e)=p$ 和 $\rho(p)=(\tau(v_p)\ \tau(v_o))$。

10.3 将 RDF 数据集转换为语句图

可将以下映射应用于 RDF 数据集 $D=\{G_D、(g_i, G_i)、\cdots、(g_n, G_n)\}$，以获得 RDF 映像 $G=(V, E, \tau)$

(1) 将 v_{in} 添加到 V，令 $\tau(v_{in})=in$(2) 对所有元素 $g \in D$：(a) 如果 g 是命名图，则令 $(name, graph)=(g_i, G_i)$；如果 g 是默认图，则令 $(name, graph)=(G_{default}, G_D)$；(b) 将新顶点 v_{graph} 添加到 G，令 $\tau(v_{graph})=name$；(c) 对所有 RDF 三元组 $(s, p, o) \in graph$：(i) 将 v_s 添加到 G，令 $\tau(v_s)=s$；(ii) 将 v_p 添加到 G，令 $\tau(v_p)=p$；(iii) 将 v_o 添加到 G，令 $\tau(v_o)=o$；(iv) 将 v_{st} 添加到 G，令 $\tau(v_{st})$ 为 S 中未使用的元素 (v) 将边 $(v_{st}, subject, v_s)$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E；(vi) 将 v_{mem} 添加到 G，令 $\tau(v_{mem})$ 为 S 中未使用的元素 (vii) 将边 $(v_{mem}, subject, v_{st})$、$(v_{mem}, predicate, v_{in})$ 和 $(v_{mem}, object, v_{graph})$ 添加到 E；(3) 合并所有同构顶点；(4) 对于 τ，合并 V 中值相等的顶点。

10.4 将语句图转换为 RDF 数据集

可将以下映射应用于 RDF 映像 $G=(V, E, \tau)$，以获得 RDF 数据集 $D=\{G_D、(g_i, G_i)、\cdots、(g_n, G_n)\}$，令 D 仅包含默认图。

(1) 令 V_{st} 为 V 中的内部顶点集合。遍历 V_{st} 时，令 v_s、v_p、v_o 为来自某个 $v \in V_{st}$ 的入边分别为 $subject$、$predicate$、$object$ 的顶点；(2) 对于每个满足 $\tau(v_p) \neq in$ 的 $v \in V_{st}$：(a) 创建 RDF 三元组 $t=(s, p, o)$；(b) 令 $s=\tau(v_s)$、$p=\tau(v_p)$、$o=\tau(v_o)$；(c) 令 V_{mem} 为所有存在指向 v 的出边的顶点集合；(d) 对于所有 $v_{mem} \in V_{mem}$，令 v_o 为来自 v_{mem}、入边为 $object$ 的顶点：(i) 令 $name=\tau(v_o)$；(ii) 若 $name=G_D$，则将三元组 t 添加到 D 中的默认图。(iii) 若 $name \neq G_D$，则将三元组 t 添加到 D 中名称为 $name$ 的命名图；若该图不存在，则新建一个。

10.5 RDF-star 数据集到语句图

可以将以下映射应用于 RDF-star 数据集 $D=\{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$，以获得 RDF-star-图像 $G=(V, E, \tau)$

(1) 将 v_{in} 添加到 V，令 $\tau(v_{in})=in$(2) 对所有元素 $g \in D$ 执行以下操作：(a) 如果 g 是命名图，则令 $(name, graph)=(g_i, G_i)$；如果 g 是默认图，则令 $(name, graph)=(G_{default}, G_D)$；(b) 将一个新顶点 v_{graph} 添加到 G，令 $\tau(v_{graph})=name$；(c) 对所有 RDF-star 三元组 $(s, p, o) \in graph$ 执行以下操作：(i) 如果 s 是 RDF-star 三元组，则对 t 执行步骤 (i)、(ii)、(iii)、(iv) 和 (v)。令 v_s 为在步骤 (v) 中获得的内部顶点；(ii) 否则，将 v_s 添加到 G，令 $\tau(v_s)=s$；(iii) 将 v_p 添加到 G，令 $\tau(v_p)=p$；(iv) 如果 o 是 RDF-star 三元组，则对 t 执行步骤 (i)、(ii)、(iii)、(iv) 和 (v)。令 v_o 为在步骤 (v) 中获得的内部顶点；(v) 否则，将 v_o 添加到 G，令 $\tau(v_o)=o$；(vi) 将 v_{st} 添加到 G，令 $\tau(v_{st})$ 为 S 中未使用的元素；(vii) 将边 $(v_{st}, subject, v_s)$、$(v_{st}, predicate, v_p)$ 和 $(v_{st}, object, v_o)$ 添加到 E；(viii) 将 v_{mem} 添加到 G，令 $\tau(v_{mem})$ 为 S 中未使用的元素；

(ix) 将边 $(v_{mem}, subject, v_{st})$、$(v_{mem}, predicate, v_{in})$ 和 $(v_{mem}, object, v_{graph})$ 添加到 E；(3) 合并所有同构顶点；(4) 对于 τ，将 V 中具有相同值的顶点合并。

10.6 语句图到 RDF-star 数据集

可将以下映射应用于 RDF-star-图像 $G = (V, E, \tau)$，以获得 RDF-star 数据集 $D = \{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$，令 D仅包含默认图。

(1) 令 V_{st}为 V中的内部顶点集合。在迭代 V_{st}时，令 v_s、v_p、v_o为从一个 $v \in V_{st}$接收 $subject$、$predicate$、$object$边的顶点；(2) 对于满足 $\tau(v_p) \neq in$的每个 $v \in V_{st}$：

(a) 创建 RDF-star 三元组 $t = (s, p, o)$；(b) 若 $s \in S$，则令 $s = \tau(v_s)$，并对 v_s执行步骤 (a)、(b)、(c) 和 (d)。令 s为步骤 (d) 中获得的 RDF-star 三元组；(c) 令 $p = \tau(v_p)$；(d) 若 $o \in S$，则令 $o = \tau(v_o)$，并对 v_o执行步骤 (a)、(b)、(c) 和 (d)。令 o为步骤 (d) 中获得的 RDF-star 三元组；(e) 令 V_{mem}为指向 v的顶点集合；(f) 对所有 $v_{mem} \in V_{mem}$，令 v_o为从 v_{mem}接收$\overline{object}$边的顶点：(i) 令 $name = \tau(v_o)$；(ii) 若 $name = G_D$，则将三元组 t添加到 D中的默认图；(iii) 若 $name \neq G_D$，则将三元组 t 添加到 D中名为 $name$ 的命名图；如果不存在此类图，则创建一个新的命名图。