知识图谱构建模块：用于开发更符合 FAIR 原则的知识图谱的易用框架

Vogt,Lars[1] orcid.org/0000-0002-8280-0487；Konrad,Marcel[1]
orcid.org/0000-0002-2452-3143Prinz,Manuel[1] orcid.org/0000-0003-2151-4556[1] *TIB* 莱布尼茨科学技术信息中心*Welfengarten 1B，30167* 汉诺威，德国

通讯作者：lars.m.vogt@gmail.com

摘要

知识图谱和本体为实施FAIR原则、使数据和元数据具备可发现、可访问、可互操作和可复用性提供了颇具前景的技术解决方案。然而，它们也带来了自身的挑战。本文讨论了九项此类挑战，并将其与认知互操作性标准以及它们未能满足的特定FAIREr原则（FAIR + 可探索性）联系起来。我们介绍了一种易于使用的开源知识图谱框架，该框架基于知识图谱构建模块（KGBB）。KGBB是用于知识处理的小型信息模块，每个模块都基于特定类型的语义单元。通过相互关联多个KGBB，可以指定一个由KGBB驱动的FAIREr知识图谱。除了实现语义单元外，KGBB框架还明确区分并解耦内部的内存数据模型与数据存储、数据展示和数据访问/导出模型。我们认为，这种解耦对于解决知识管理系统中的许多问题至关重要。我们讨论了所设想的KGBB框架架构，包括：（i）面向不同类型KGBB的开放可访问KGBB仓库；（ii）用于管理和运行FAIREr知识图谱的KGBB引擎，包括自动溯源跟踪、编辑变更日志以及语义单元的版本管理；（iii）KGBB函数仓库；（iv）低代码KGBB编辑器，领域专家可以利用它创建新的KGBB并指定自己的FAIREr知识图谱，而无需考虑语义建模。最后，我们讨论这九项挑战，以及KGBB框架如何为它们所提出的问题提供解决方案。尽管本文讨论的大部分内容完全属于概念层面，但我们可以指出两个原型，它们展示了使用语义单元和KGBB管理并构建知识图谱在原则上的可行性。

简介

根据**FAIR指导原则**（1），数据和元数据必须能够被机器和人类发现、访问、互操作和复用。近年来，FAIR指导原则日益受到研究人员、产业界以及知识管理工具开发者的关注（1–6）。科学界和科研政策领域的利益相关者也开始认识到FAIR的重要性。FAIR同样与经济相关——2018年，欧盟（EU）委托开展的FAIR研究数据成本效益分析估计，缺乏FAIR研究数据每年给欧盟经济造成的损失至少为102亿欧元；如果考虑FAIR数据在数据品质和机器可读性方面的积极影响，则这一数字还应增加160亿欧元（7）。因此，欧洲开放科学云（EOSC）高级专家组建议构建FAIR数据与服务互联网（IFDS）（8），使所有相关的数据丰富型机构、科研项目和公民科学项目能够依据FAIR指导原则开放其数据和元数据，同时保留对数据伦理、隐私或法律方面的完全控制权（遵循Barend Mons提出的数据访问而非数据共享理念（9））。这要求提供丰富的机器可操作数据和元数据，将其组织成**FAIR数字对象**（10,11）；其中每个对象都可以通过其自身的唯一持久可解析标识符（UPRI）单独引用，并开发适当的概念和工具，以支持人类可读的界面输出和搜索能力。尽管构建IFDS的工作已经展开（见GO FAIR倡议和EOSC），但许多数据丰富型机构和公司在其数据和元数据的FAIR性方面仍远非理想。

凭借透明的语义学、高度结构化的句法和标准化格式，**知识图谱**与**本体**相结合，为实施FAIR指导原则提供了一种前景广阔的技术解决方案（12,13）。在知识图谱中，实例、类及其关系都可以表示为拥有各自 UPRI 的资源，按照主语-谓词-宾语的三元组语法表示实体之间的关系。因此，每一种特定关系都被建模为由三个不同数据点组成的结构化集合。相比之下，关系数据库采用表格格式，实体之间的关系是在数据表的列之间建模的，而不是在单个数据点之间建模的。其结果是，对于稠密连接数据上的复杂查询，知识图谱的性能优于关系数据库；研究数据及其相关元数据通常正属于此类数据。因此，知识图谱尤其适用于 FAIR 研究与开发的场景，以及所有需要实现最佳且详细数据检索的任务。

然而，使用知识图谱记录数据和元数据，并不能保证其具备 FAIR性；要做到这一点，还需要遵循其他指南，例如，始终使用相同的语义数据模型来建模相同类型的数据和元数据陈述，以实现模式互操作性（14），并将数据和元数据组织成 FAIR数字对象（10,11）。此外，作为一种相对较新的概念和技术，知识图谱也带来了自身特有的技术、概念和社会挑战。仅仅看看这个有些模糊的知识图谱 概念（12）以及缺乏普遍认可的标准，就能清楚地说明这一点：知识图谱涵盖了差异显著的技术和概念形态，例如属性图，如Neo4J ，以及基于资源描述框架（RDF）的方法、RDF 存储的使用，还有网络本体语言（OWL）及描述逻辑的应用。下文将讨论九个用户故事，说明知识图谱应用的用户和开发者所面临的一些技术、概念和社会挑战。

我们讨论这些用户故事，是因为我们认为，在 FAIR 和 GoingFAIR 战略的背景下，重点主要放在实现数据和元数据的机器可操作性上，而在这一过程中，人类可操作性以及相应知识管理框架（例如 RDF）的可用性却被忽视了。九个用户故事分别反映了数据、元数据和知识管理框架在人类可操作性方面不足的一个侧面。为应对这一问题，我们建议（14）扩展 **EOSC 互操作性框架**（EOSC IF）（10），在四个互操作层之外，即 **1）技术互操作性**（即信息技术系统在实施或访问时必须能够与其他信息技术系统协同工作，不受任何限制或仅受控访问）、**2）语义互操作性**（即与通用语义资源相关的上下文语义）、**3）组织互操作性**（即与通用流程资源相关的上下文流程）和 **4）法律互操作性**（即与通用许可资源相关的上下文许可），再纳入 **5）认知互操作性**这一层。

我们将**认知互操作性** 理解为

> “信息技术系统的一种特性，即以最高效的方式与人类用户沟通其信息，为他们提供工具和功能，直观地支持他们全面了解数据、查找感兴趣的数据，并允许他们从给定的数据点出发，以语义上有意义且直观的方式探索其他数据点，从而兼顾人类的一般认知条件——不仅考虑人类偏好如何与技术交互（人机交互），还考虑他们如何与信息交互（人类—信息交互）。在知识图谱的背景下，这些工具应使用户了解其内容，帮助用户理解其含义，支持其内容的传达，提供提高其内容可信度的手段，支持将其集成到其他工作流和软件工具中，并明确说明可以采取哪些操作。此外，信息技术系统的认知互操作性也可以理解为该系统的一种特性，即开发者能够轻松地实现和使用该系统，运维人员能够轻松地对其进行管理。”(14)

因此，认知互操作性关注的是数据结构和知识管理系统对人类用户和开发者的可用性——这是迄今为止在一定程度上被忽视的一个方面，尤其是在知识图谱和语义技术的语境中。作为提升认知互操作性的指导方针，我们建议扩展 FAIR 指导原则，纳入**人类可探索性原则**，从而形成**FAIREr（可探索性提升）指导原则**(14)（有关这些原则的列表，请参见方框1）。

框 1 | FAIREr 指导原则 (14)。其中包括 FAIR指导原则 (1)（常规字体）以及拟议的补充原则，包括人类可探索性原则（粗体）。

可发现：

F1. （元）数据被赋予全局唯一且持久的标识符
F2. 数据由丰富的元数据描述（定义见下文 R1）
F3. 元数据清晰明确地包含其所描述数据的标识符
F4. （元）数据已在可搜索资源中注册或编入索引
F5.1 具有相同扩展的术语必须相互映射
F5.2 术语最好包含多语言标签，并注明所有相关同义词
F6.1 相同类型的陈述使用相同的（元）数据模式，并通过其

陈述元数据中的标识符 **F6.2 相同陈述类型的（元）数据模式必须彼此对齐并映射**
要做到可访问： A1. 使用标准化的通信协议，通过标识符检索（元）数据 A1.1 该协议是开放、免费的，且可由所有人实现 A1.2 该协议允许在必要时进行身份验证和授权程序 **A1.3 该协议符合现行数据保护法规（例如《通用数据保护法规》（GDPR）** A2. 即使数据不再可用，元数据仍可访问
实现可互操作： I1. （元）数据采用正式、可访问、共享且广泛适用的知识表示语言 I2. （元）数据使用遵循 FAIR 原则的词汇表 I3. （元）数据包含指向其他（元）数据的限定引用 **I4. （元）数据说明其建模所用的逻辑框架（例如 OWL，因此也就是描述逻辑或通用逻辑交换框架，因此也就是一阶逻辑）** **I5. （元）数据使用一种形式体系，明确区分词汇性、断言性、偶然性、原型性和全称陈述** **I6.1-2 参见 F5.1-2** **I7.1-2 参见 F6.1-2**
可复用： R1. （元）数据通过大量准确且相关的属性进行了丰富描述 R1.1 （元）数据以清晰且可访问的数据使用许可发布 R1.2 （元）数据附有详细的溯源信息 R1.3 （元）数据符合领域相关社区标准
要可探索： **E1. （元）数据被划分为语义上有意义的子集，每个子集都有自己的全局唯一且持久的标识符，使其能够被引用和识别，并实例化一个语义上定义的对应类** **E1.1 （元）数据中包含的每个二元和多元命题都被组织到各自的陈述子集中** **E1.2 （元）数据中包含的每棵粒度树都被组织到各自的粒度树子集中** **E1.3 （元）数据中包含的每个参照框架都被组织到各自的上下文子集中** **E2. （元）数据子集可以（递归地）组合，并将复杂性封装为易于管理的单位** **E2.1 某些子集按照不同的表征粒度层级进行组织** **E3. （元）数据的人类可读显示与机器可操作的（元）数据存储相分离，从而减少显示的（元）数据的复杂性，使其仅包含与人类读者相关的内容** **E3.1 用户界面提供基于表单的文本选项和思维导图式图形选项，以便访问和探索（元）数据** **E3.2 用户界面支持对（元）数据进行缩放，利用不同的表征层级粒度、粒度树和参照框架，其中陈述子集包含实际陈述作为其标签** **E3.3 用户界面支持对陈述作出陈述** **E3.4 用户界面无需了解（图）查询语言即可查询（元）数据**

讨论完九个用户故事后，我们将简要总结语义单元的概念，以及如何应用语义单元将知识图谱组织和构建为具有语义意义的子图。接下来，我们介绍知识图谱构建模块（KGBB）的概念，将其作为用于管理不同类型语义单元的模块化管理单位。我们讨论不同类型的 KGBB 及其部分功能和特性，以及如何使用 KGBB 来指定由KGBB驱动的FAIREr知识图谱应用。随后，我们将描述 KGBB 框架的不同组成部分。除各种 KGBB 外，该框架还包括 KGBB 引擎，该引擎基于 KGBB 中的信息运行 FAIREr 知识图谱应用，并提供自动溯源跟踪、编辑变更日志和版本管理。在简要介绍 KGBB函数的构想后，我们将讨论 KGBB函数如何为数据分析和可视化目的向知识图谱添加基于规则的功能，并进一步讨论用于定义新 KGBB 以及组合现有 KGBB 以指定一个

以 KGBB 规范图形式呈现的 KGBB 驱动的知识图谱应用。最后，我们讨论 KGBB 框架对开发和使用更具 FAIR 性的知识图谱应用的影响及其潜在挑战，包括其为用户故事中举例说明的问题提供解决方案的潜力。

我们想指出，我们在此呈现的大部分内容都是概念性工作的成果，仍有待进一步开发、测试和实施。目前，只有两个应用程序采用 KGBB 框架开发。其中一个是面向众包学术知识图谱的小型原型，实现了 KGBB 和语义单元的一些基本理念，因此可作为若干核心组件的基础概念验证（https://github.com/LarsVogt/Knowledge-Graph-Building-Blocks）（16）。另一个原型是在一个硕士论文项目中开发的，它已经支持对某些 KGBB 类型进行通用规范定义，从而能够定义不同类型的基本知识图谱应用。它还支持将知识图谱存储在多个数据库中，即 RDF 和 Neo4j（https://gitlab.com/TIBHannover/orkg/semantic-building-blocks）。

框 2 | 约定

本文将 FAIR 知识图谱称为机器可操作语义图，用于记录、组织和表示词汇陈述、断言（例如经验数据）、全称陈述和偶然陈述，因此它们混合了 ABox 和 TBox 表达式（这与本体形成对比：本体主要包含全称陈述，因而包含 TBox 表达式和词汇陈述）。我们想指出的是，讨论 KGBB 时所依据的背景包括基于 RDF 的三元组存储、作为推理形式框架的 OWL 和描述逻辑，以及作为三元组存储替代方案的带标签属性图，因为这些是知识图谱领域的主要技术和逻辑框架，得到广泛的用户和开发者社区支持，并且已有公认的标准。我们知道，也存在支持 n 元组语法和更高级逻辑（例如一阶逻辑）的替代技术和框架 (15,16)，但缺少相应的工具和应用程序，或这些工具和应用程序并未得到广泛使用，因此无法将其发展为支持完善、可扩展且易于使用的知识图谱应用程序。

在本文中，我们用常规下划线表示本体类，用斜体下划线表示属性（即 Neo4j 中的关系），并用 ID 编号来标识每个类或属性。ID 编号由本体前缀、冒号和数字组成，例如*isAbout*（IAO:0000136）。如果某个术语尚未纳入任何本体，我们会用 * 标示，例如类 *度量测量陈述单元*。我们用“常规下划线”表示类的实例，其中标签指类标签，ID 编号指类本身。此外，我们所说的资源，是指具有唯一标识（例如统一资源标识符 URI），且可用于描述某些内容的事物。因此，资源代表某种存在，也代表你想要讨论的事物。在 RDF 中，三元组陈述中的 主语 和谓词 始终是资源，而宾语 则可以是资源或字面量。资源可以是属性、实例或类；属性在三元组中占据谓词位置，实例指个体（=特殊事物），类则指共相。

为清晰起见，在正文和所有图示中，我们不使用资源的 UPRI，而是使用人类可读的标签来表示资源；这隐含地假设每个属性、每个实例和每个类都有其自身的 UPRI。

用户故事 1：Tom 需要让自己的数据和元数据符合 FAIR，但不想查看庞大而复杂的图

Tom 是一名生物学家，希望存储其基因组-表型实验的数据和元数据，以便日后进行分析和记录。他发现了特定基因与表型特征之间的各种因果关系，并希望将这些关系连同相关数据和元数据记录在一个 FAIR 且机器可操作的知识图谱中，然后将其与其他

可比数据整合，从而形成一个非常庞大且高度复杂的图。他想要探索这个图，但其复杂性以及其中包含许多似乎与他无关的节点，让他感到不知所措。

和所有人类一样，Tom 擅长高效地传达信息、省略背景知识，并使用一些含糊的陈述来指代一般性的思想图式，也会使用隐喻和转喻，因为他知道其他人类仍然能够理解他的意思，并根据上下文推断缺失的信息。另一方面，对于机器而言，所有相关信息都必须明确陈述。因此，Tom 面临着一个困境：数据和元数据的机器可操作性与人类可操作性之间存在冲突：**他越是推动数据表示朝着机器可操作的方向发展，这些表示就越复杂，因此也越不便于人类操作**(14)（见图 1，中部）。这体现了阻抗失配，让 Tom 感到沮丧。

如果 Tom 想要以机器可操作格式将数据和元数据存储在知识图谱中，同时又在用户界面（UI）中以易于理解、便于人类阅读的方式呈现它们，他就必须通过开发新的探索和导航方法与工具，提升 **图的认知互操作性**，并在不同的表征粒度层级之间缩放，从而将图的复杂性降低到仅包含他当下相关的信息。理想情况下，他会**将图中的数据存储与用户界面中的数据呈现分离 [E3.;E3.1]**, ，这样，只有机器需要而人类无关的信息便只由机器访问，而不会显示在用户界面中（见图 1，底部）。

由于 Tom 的项目属于更大的项目计划的一部分，Tom 甚至希望能够区分**针对其数据定义的不同视图**，具体取决于用于呈现这些视图的设备（例如浏览器或智能手机）、用户拥有的访问权限，或所需的详细程度。用于描述图模式的工具能够强制以标准化方式对同一类型的数据进行建模和表示，例如形状约束语言 SHACL 和数据形状 DASH（17）、形状表达式 ShEx （18、19），或合理本体模板 OTTR（20、21），只能在一定程度上解决这一问题。Tom 需要更全面的解决方案。

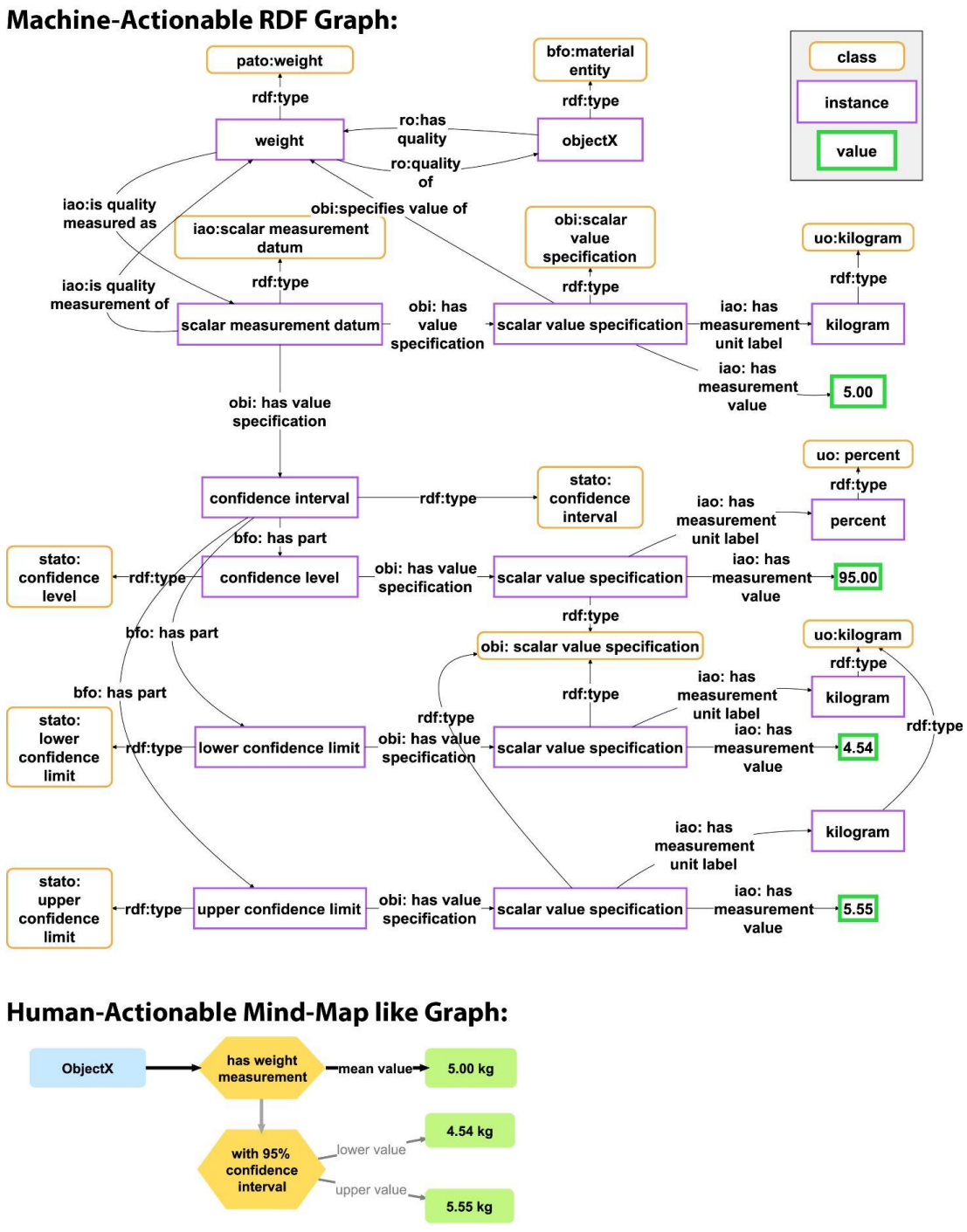

图1：人类可读陈述及其机器可操作表示和人类可操作表示的比较。顶部：关于观察结果的人类可读陈述：ObjectX 的重量为 5 千克，95% 的置信区间为 4.54 至 5.55。**中部**：将同一陈述表示为 ABox语义图的机器可操作表示，采用 RDF 语法，并遵循生物医学研究本体（OBI）（22）中有关测量数据的一般模式，该本体属于开放生物与生物医学本体（OBO）Foundry。 **底部**：将同一陈述表示为思维导图式图的人类可操作表示，将 RDF图的复杂性简化为与人类读者实际相关的信息。 [图片摘自（14）]

用户故事 2：Catherine 和 Felix 希望将 FAIR 数据添加到一个范围广泛的众包知识图谱中

Catherine 和 Felix 是领域专家，也是同事。他们都致力于描述进化上密切相关生物的解剖学。他们已在各自的领域发表了多篇论文，现在希望将已发表的解剖学描述添加到一个众包学术知识图谱中，该图谱允许用户将学术出版物的内容（而不仅仅是书目元数据）添加到图中。这个知识图谱的范围非常广泛，旨在涵盖所有可能研究领域的学术知识。遗憾的是，在比较各自创建的图后，他们发现彼此的数据建模方式不同，导致机器无法整合和比较他们的解剖学描述。换言之，他们的数据无法互操作。投入大量精力添加这些描述后，Catherine 和 Felix 对这一结果感到沮丧。他们不会继续使用这个应用程序。

Isabelle 是这个应用程序的开发者。由于应用程序的范围非常广泛，她允许用户自由添加数据，并随时创建新的属性和类。她基本上采用了维基百科式的社区驱动众包方法。她曾建议 Catherine 和 Felix 使用她的知识图谱，如今却对他们的反馈感到沮丧。Isabelle 理解他们的批评，但不知道如何改进自己的应用程序。

RDF/OWL 和带标签属性图都具有很强的表达能力。任何给定的陈述都可以通过许多不同的方式建模。因此，如果用户可以不受任何限制地自由添加内容，那么数据很可能既不可互操作，也不易发现，从而仅具备最低程度的机器可操作性，并不能真正符合 FAIR。

如果适当的图模式能够公开提供，供知识图谱开发者复用，那么就可以使用 SHACL、ShEx 或 OTTR 等形状约束语言和框架来实现这些图模式。使用这些语言和框架可以确保通过它们添加的所有数据和元数据都具备互操作性（即模式互操作性）。此外，如果每个此类图模式都有自己的 UPRI，数据陈述就可以在其元数据中引用这些图模式。借助这些信息，可以根据共同使用的图模式标识符识别出可能具备互操作性的数据。实际上，这意味着知识图谱中的所有数据陈述都必须引用其关联的图模式（例如，以形状规范的形式），并且理想情况下还应能够识别出与特定图模式相对应的子图（即给定图模式的所有实例化）。当满足这些标准时，知识图谱中记录的经验数据及其元数据才真正符合 FAIR指导原则（23） **[F6.1/I7.1]**。

用户故事 3：Sara 和 Bob 不懂图查询语言，学习起来很困难

Sara 是一名软件开发人员，她受聘参与一个研究项目，负责开发一个用于存储和记录项目数据和元数据的数据库。她拥有使用关系数据库的经验，但不了解基于图的数据库及其**图查询语言**，例如 SPARQL 或Cypher。这些语言的复杂性使她一直没有学习它们，部分原因还在于，凭借当前的关系数据库专业知识，她已经能够实现项目所需的大部分功能。Sara 会主张，缺失的部分只有通过投入大量技术和资金才能实现。

Bob 是一名工程领域专家，他同样没有图查询语言方面的经验。当 Bob 听说有一种用于工程相关研究数据的新数据库，并意识到查询其中的内容需要使用 SPARQL 或 Cypher 时，他感到很沮丧。他的研究工作非常繁忙，不愿意额外投入时间去学习一种相当复杂的查询语言。如果征求他的意见，他总是会更倾向于使用具有典型搜索功能的关系数据库，而不是使用带有 SPARQL 或 Cypher 查询界面的知识图谱。

所描述的场景反映了我们的个人经验：大多数用户和软件开发人员都没有使用基于图的数据库的经验，也不了解图查询语言及其优势，因此看不到学习它们的必要性。即使是熟悉这些语言的人也表示，编写复杂查询可能颇具挑战。**显然，必须编写 SPARQL 或 Cypher 查询构成了与知识图谱交互的入门障碍，并阻碍了其更广泛的使用**（24）。公开可用且可复用、能够链接到特定图模式的查询模式，或允许使用表单或自然语言编写查询并自动将输入转换为 SPARQL 或 Cypher 查询的用户界面，将有助于解决这一问题 **[E3。4]**。

用户故事4：Karl 希望让用户能够直观地在知识图谱中对陈述作出陈述

Karl 是一名软件开发人员，希望开发一个知识图谱应用，用于记录研究项目中的测量数据，并识别经验数据与各种因果假设之间可能存在的关系。他的工作之一，是让用户能够在知识图谱中对陈述作出陈述，并在应用及其用户界面中以透明且用户友好的方式组织这些信息。RDF再ification（25）和 RDF-star（26,27）适合用于指代单个三元组，但如果要对更大的子图作出陈述，例如对一个以超过20个三元组建模、带有95%置信区间的测量数据项作出陈述，那么这种方式在查询时会变得低效且复杂（见图1，中部）。相比之下，使用命名图似乎是更好的解决方案（25），但需要明确规定哪些三元组属于哪个命名图，并据此制定标准，以便系统地将数据图组织成独立的子图。Karl 对这两种解决方案都不满意。

自动将数据图结构化并组织成不同的**语义上有意义且可识别的子图**，并使其具有不同的表征粒度，可为知识图谱的结构化提供一种高效方式，让用户能够直观地对陈述进行陈述 [**E1.1-3;E2.;E3.3**]。

用户故事5：Dan 和 Anna 描述相同类型的测量数据，但使用不同的社区特定数据模型

Dan 是一名植物学家，Anna 是一名生态学家。Dan 描述植物有机体的解剖学，包括各种品质测量。他希望按照开放生物学和生物医学本体（OBO）Foundry 的模型，将这些描述记录在知识图谱中（图 2，上方）。Anna 描述植物—昆虫关系。在此背景下，特定的植物品质具有相关性，因此她也对其进行描述。在不了解 Dan 的描述的情况下，她将自己的描述记录在同一个知识图谱中。然而，由于她是一名生态学家，她遵循的是该模型的

可扩展观测本体（OBOE）（图 2，底部），与 OBO模型有很大差异。添加完描述后，她意识到自己还需要一些植物测量数据，并想使用 Dan 的数据。然而，她很快就感到沮丧，因为他们的数据不容易实现互操作，而且 Anna 没有编程技能，无法编写脚本将 Dan 的数据转换为 OBOE模型。

标准通常由各个社区制定。对于长期不变的既定标准，制定符合该标准的图模式可能很简单；但要在多个社区中遵循不同且经常变化的标准，就颇具挑战。图2展示了两种用于建模重量测量、差异显著的图模式。如果在同一个知识图谱中同时采用这两种模式，相应的数据便无法直接互操作。

通过将数据从既定存储模型对齐并映射到不同的既定访问模型，将数据访问与数据存储解耦，从而定义**数据和元数据模式交叉映射**，便可解决这一问题。如果能够随时将新的访问模型添加到知识图谱中，就可以根据新提出的标准和格式调整其数据和元数据，从而适应不断变化的研究格局。 **[F6.2/I7.2]**

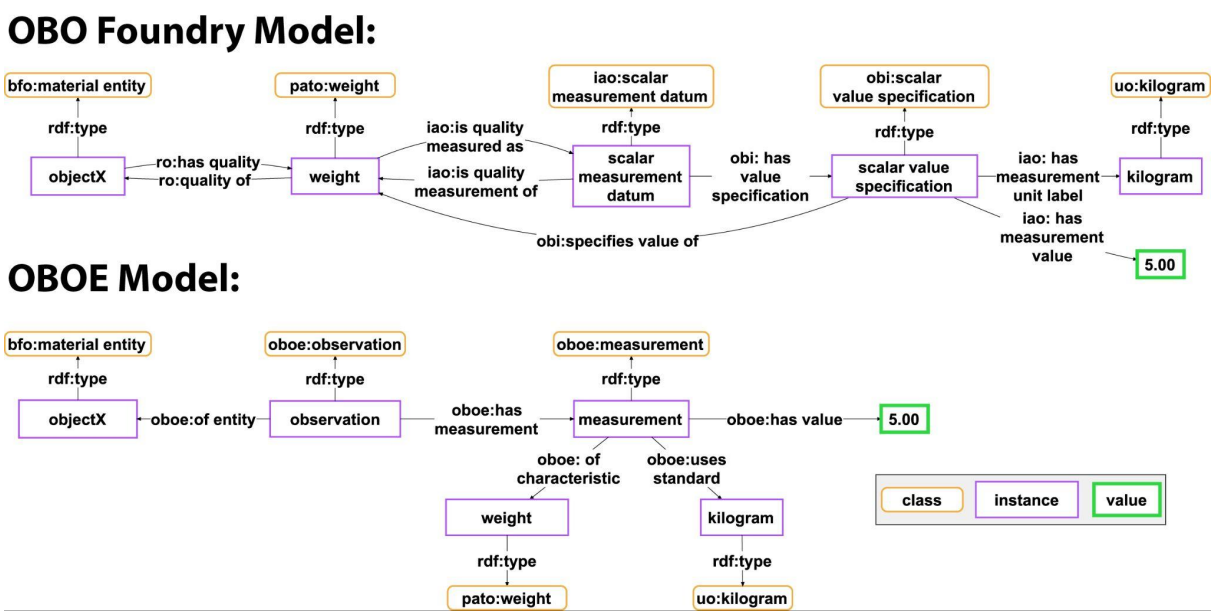

图2：用于建模同一陈述的两种图模式比较。不同社区通常会商定不同的模型，以表示同一类型的信息。例如，陈述 *"ObjectX* 的重量为 *5* 千克*"* 可以按照此处展示的两种图模式进行建模。 **顶部**： 生物医学研究本体 (OBI) (22)，隶属于 开放生物与生物医学本体(OBO) Foundry，该模式常用于生物医学领域。 **底部**： 可扩展观测本体(OBOE) 的模式，该模式常用于生态学社区。

用户故事6：John 想更改其生物多样性知识图谱后端的数据库技术

John 从事生物多样性丧失方面的研究。他参与了一个项目，该项目使用各种传感器等设备监测特定生境，以记录当地天气数据。每个传感器每十分钟提供一次与特定位置相关的数据。John 感到很沮丧，因为他在过去五年里开始使用一种特定的数据库技术构建自己的生物多样性知识图谱，而在

最初几年，一切进展顺利，监测站的数量不断增加，但他意识到自己使用的数据库扩展性不佳——查询时空数据所需的时间每年都越来越长。

John 做了一些研究，发现过去几年中，存储技术和查询技术都有了显著改进。他希望更换其知识图谱所使用的数据库，但这并不容易，而且 John 目前没有所需的资源。

诸如 Ontop这类实现基于本体的数据访问（OBDA）范式的虚拟知识图谱系统，已被提出用于将针对知识图谱表达的 SPARQL 查询转换为在关系型数据源上执行的 SQL 查询（28–31），从而支持对 RDF 三元组存储和关系数据库执行联邦查询。通过 **将存储模型与存储技术解耦**，并通过适当的应用程序接口允许将数据图存储在（i）各种图数据库中，例如 RDF 存储、Wikibase，或 Neo4j等属性图数据库中；（ii）其他 NoSQL数据库中，例如键值数据库；或者（iii）以语义表的形式存储，这种语义表在语义上与该图同构，并存储在关系数据库中，OBDA 方法可以扩展到包括更多存储技术。这样便能够将时空结构数据作为语义表存储在关系数据库中。此外，如果需要采用 RDF 或属性图表示数据，则可以随时导出该表，并将其存储在相应的替代技术中。能够将存储模型与存储技术解耦的知识管理框架，将提升其对知识图谱应用开发者的**认知互操作性**。

用户故事 7：Mila 希望将语义结构不清晰的（旧有）数据从表格格式导入 FAIR 知识图谱

Mila 是一座自然历史博物馆的馆长，该馆收藏着大量标本。博物馆还保存着过去项目留下的各种藏品相关数据集，每个数据集都有各自特定的格式和数据方案。Mila 希望她的博物馆参与一项即将启动的计划，该计划旨在构建一个关于藏品相关数据的大型国家知识图谱。该知识图谱要求数据符合 FAIR 原则。起初，Mila 很愿意为此贡献博物馆的数据。然而，她很快意识到，要将数据传入知识图谱并使其符合 FAIR 原则，就必须将数据转换为知识图谱的数据模式。这项工作非常耗时，而且需要语义学和本体方面的经验，而博物馆并不具备这些经验。这让 Mila 感到沮丧，于是她打了退堂鼓。

许多有价值的研究数据集在语义上尚未结构化，并以表格格式（例如 CSV、Excel 工作表）存储在本地计算机上。因此，这些遗留数据集不符合 FAIR 原则，难以被发现、访问、与其他现有数据集成并用于研究。将其转换为 FAIR 格式可能极具挑战性且耗时。通常，这需要采用利用深度学习算法的复杂半自动自然语言处理方法，或投入大量人工工作。支持遗留数据 FAIR 化的知识管理框架可提高其**知识图谱应用程序的认知互操作性**。

用户故事 8：Nick 必须区分其知识图谱中的全称陈述、典型陈述和断言陈述

Nick 是一名数据科学家，参与一个项目，该项目收集临床研究数据，旨在研究某些药物在妊娠期间对人体大体解剖学胎儿发育的潜在影响。这些数据将使用人体解剖学本体中的术语记录在 FAIR 知识图谱中，涵盖基于超声记录的个体胎儿解剖描述、有关妊娠期间所服用药物的协议，以及以原型解剖描述形式呈现的统计平均胎儿解剖发育情况。由于人类胎儿解剖结构存在显著差异（即个体胎儿解剖结构之间形态空间的范围），因此，预计不同解剖描述之间会存在一定程度的变异。

Nick 的职责是为这些数据建模，开发合适的图模式。数据包括本体术语及其类公理，这些内容将用于对数据进行推理。由于类公理规定了该类的每个实例必然满足的条件，因此它们是**全称陈述**；在 OWL 中，它们构成 TBox（术语盒）。另一方面，对单个胎儿的解剖描述由仅对某个特定个体为真的陈述组成，因此属于**断言陈述**（另见*实例解剖学* (32)）；在 OWL 中，它们构成 ABox（断言盒）。最后，**典型的** 解剖描述包括仅对某个类的大多数实例为真、但不一定对所有实例都为真的陈述，例如，手通常有拇指作为其组成部分，但没有拇指的手仍然是手。这些是**偶然（典型）陈述**（另见*规范解剖学* (32)）。

Nick 更倾向于使用带标签属性图，而不是 RDF 元组存储，因为前者直观易用，还可以为图中的每个节点和关系添加属性-值对，从而支持对陈述进行陈述（参见*用户故事4*）。然而遗憾的是，属性图缺少一套统一的形式语义与推理标准，因此无法在形式上区分断言性、原型性和全称陈述（33,34）。目前已有方案提出如何在属性图中表示全称陈述。这些方案会为边添加逻辑属性，例如存在限制公理，从而形成对象属性映射（33,34）。然而，目前仍缺少一种由万维网联盟推荐的标准形式体系，用于将 OWL 映射到属性图；因此，也无法基于数据进行推理。于是，Nick 放弃了最初使用属性图的想法，转而考察 RDF 元组存储和 OWL 这一替代技术方案。

由于 OWL 基于描述逻辑，因此它区分断言与全称陈述，但并未提供区分原型陈述的形式体系。此外，对全称陈述的 OWL 表示往往会产生不必要的复杂图，不仅人类读者难以直观理解，还涉及空白节点，导致难以对其进行查询 (34)。此外，以 TBox 表达式形式表示的全称陈述会作为本体类的公理记录在 OWL文件中，因此不属于知识图谱的论域，用户也就无法在知识图谱中对其作出陈述。Nick 感到沮丧，因为没有任何技术能满足他的所有要求。Nick 需要一个用于知识图谱应用的框架，能够促进对断言、原型和全称陈述进行形式上的区分 **[I5。]**。

用户故事 9：Paula 想构建一个 FAIR 知识图谱，但她既没有语义学方面的经验，也没有所需的资源

Paula 是一位从事生物多样性研究的生物学家。她通常为规模较小的专业化项目争取资金，而这些项目没有足够的资源聘请语义学专家，以对项目数据进行语义建模并将其记录在 FAIR 知识图谱中。这让 Paula 感到沮丧，因为她相信数据和元数据应该共享和重用，也希望自己的数据和元数据符合 FAIR 原则，从而为更大型的生物多样性数据集作出贡献。她所在的机构设有 IT部门，可以协助她部署现有的软件解决方案，但他们没有语义数据建模方面的经验。

Paula 需要的是一个系统，让她能够在 FAIR 知识图谱中记录数据和元数据，而无需用 OWL 或带标签的属性图对数据进行语义建模。Paula 唯一需要贡献的就是她的领域知识。遗憾的是，基于知识图谱技术的知识管理框架通常**缺乏大范围推广所需的认知互操作性**。

将知识图谱组织成不同类型的语义单元

在 (23) 中，我们提出将知识图谱**组织成可识别的三元组集合，即对人类读者具有语义意义、因而代表表示单元（简称：语义单元）的子图**(23) **[E1.]**。每个语义单元在知识图谱中都有自己的 UPRI **[F1.;F4;E1.]**，用于将该语义单元标识为一种资源，同时也标识其关联的子图，即该语义单元的数据图（图3）。因此，引用语义单元的 UPRI 等同于引用其数据图的内容。

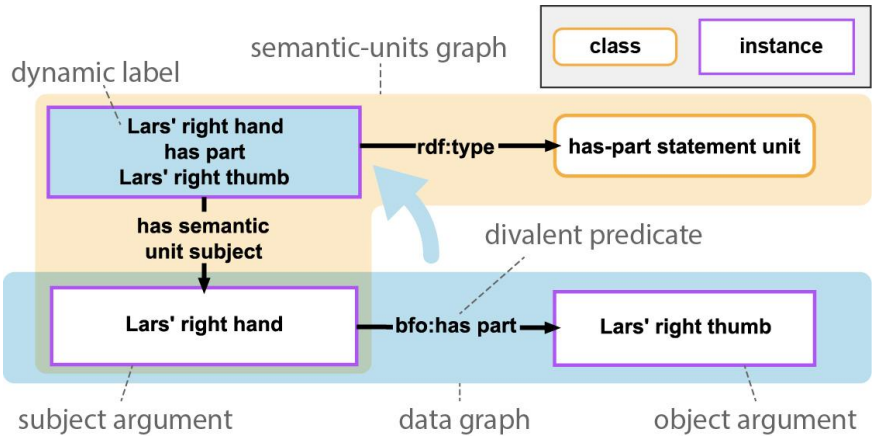

图 3：对基于两个实例之间具有部分关系的命题进行建模的陈述单元示例 数据图中表达该陈述的内容显示在底部蓝色框内，其中“Lars 的右手”是该陈述的主语论元，“Lars 的右拇指”是宾语论元。语义单元图显示在桃色框内，其中包含表示该语义单元的三元组。它说明，表示陈述单元的资源（带边框的蓝色框，此处显示其动态标签）代表数据图中的具有部分陈述（由蓝色箭头指示），是*具有部分关系陈述单元*的一个实例，并且“Lars 的右手”是该语义单元的主语。*“具有部分关系陈述单元”*的 UPRI 同样也是该语义单元数据图（蓝色无边框框中的子图）的 UPRI。[图引自（23）]

语义单元引入了一种新型**表征实体** (35)，与实例和类并列。实例、类及其关系构成了我们所说的**数据图层**，属于

知识图谱；而语义单元及其关系则为知识图谱增添了一个**语义单元图层**（图3）。在语义单元图层中，每个语义单元都由其 UPRI 表示；其对应的子图，即其数据图，则是知识图谱数据图层的一部分。

语义单元资源实例化一个相应的 **语义单元类 [F1;F4;E1.]**。该类包含可读性描述，说明其实例及其关联的数据图所涵盖的信息类型。语义单元可以采用纳米出版物（36–38）的形式组织；在正确实现的情况下，它还可作为 FAIR数字对象（10,11）发挥作用，因此满足欧盟委员会 *FAIR*数据专家组 针对 FAIR 实现所提出的要求。

应用于知识图谱时，不同类型的语义单元通过将其数据图组织为若干部分重叠、部分包含的子图层，为图增添组织框架；每个子图都构成特定语义单元的数据图，而该语义单元又通过自身的节点表示在知识图谱的语义单元图层中。**因此，语义单元是一种资源，其代表的抽象层级高于单个三元组所处的低层抽象。**我们将陈述单元、复合单元和问题单元区分为语义单元的基本类别（14,23）。

陈述单元

在组织和构建知识图谱时，如果目标是大幅提升其内容的人类可操作性，就必须着重识别**命题（即陈述）**并使其可被引用，因为命题是人类希望交流和记录的信息基本单位。即使由包含值的表格组成的数据集，也隐含地表示了命题的集合，这些命题可以从与表格或其行列相关联的上下文信息中推导出来。

因此，我们将**陈述单元**视为最基本的语义单元类别，所有其他类别都建立在其基础之上。陈述单元是**代表在语义上对人类读者有意义的最小、独立命题的信息单位（即子图） [E1.1]** (23)。陈述单元**在数学上划分知识图谱的数据图层**，使得其中每个三元组都属于恰好一个陈述单元的数据图。

陈述单元为知识图谱的语义单元图层增加了另一个更粗粒度的**表征粒度层级**，位于细粒度的单个三元组层级之上**[E2.1]**。

命题通过陈述来传达。根据语言学中的**谓词-论元结构** ，陈述的主要动词及其助动词构成该陈述的谓词。谓词具有**配价** ，决定其完成含义所需的**论元** 的数量和类型。**附加语** 也可以与谓词相关联，但并非完成谓词含义所必需。附加语提供可选信息，例如具有部分陈述中的时间说明。因此，每个陈述都包含一个主语短语作为其论元之一；根据其底层谓词，还可能包含一个或多个宾语短语作为其他论元，以及其他附加语。

我们可以说，每个陈述单元都通过将作为命题谓词主语论元的资源与某个字面量或其他资源关联起来，来记录一个特定命题；后者是该命题的一个宾语论元或附加语（见图3）。

可以根据陈述单元所建模的谓词类型来区分不同类型的陈述单元（例如，具有名字、具有部分、派生关系、具有值、给予，通过……从……前往……，时间为……陈述单元类）。因此，陈述的谓词不一定总是二元的，如陈述"主语 具有部分 宾语"中的谓词；它也可以是**多元 [E1. 1]** ，因而无法用单个三元组建模，例如"主语 给予 宾语_*A* 给宾语_*B*"；"主语 通过宾语_*A* 从宾语_*B* 前往宾语_*C* ，时间为宾语_*D*"。

除了根据陈述单元所依据的谓词进行分类外，还可以根据陈述的类别将其分类为词汇陈述单元、断言陈述单元、偶然（原型）陈述单元和全称陈述单元：

1. **词汇陈述单元** 是关于语言项目（如术语）的陈述，包含给定资源的标签或其同义词等信息。本体使用注释属性对词汇陈述进行建模。

如果该陈述不是词汇陈述，则其他所有类别的陈述都根据其主语所具有的资源类型加以区分。除了命名个体（即实例）、类、关系（即 RDF 属性）和语义单元之外，通过引入某实例和每个实例资源作为表征实体的新类型，可以将非词汇陈述单元分类如下：

2. 任何以 **命名个体资源** 为主语的陈述单元，都是**断言陈述单元**。命名个体资源的类归属通过属性 类型（RDF:type）建模。3. 任何以 **某实例资源** 为主语的陈述单元，都是**偶然陈述单元**。某实例资源（23）指给定类的某个未指定实例，其类归属通过属性 *某实例属于* 建模。当某实例资源用于陈述单元时，其含义是"i C"（其中i = 实例 ∃ ∈且 C = 类），读作"存在一个类 *C* 的实例*i* ，使得……成立"。偶然陈述可用于陈述给定类的实例通常但并非必然具有的特征。相应的**原型陈述单元** 构成偶然陈述单元的子类~~。4. 任何以 **类** 或~~**每个实例资源** 为主语的陈述单元，都是**全称陈述单元**。每个实例资源（23）指给定类的每个实例，其类归属通过属性 *每个实例属于* 建模。当每个实例资源用于陈述单元时，其含义是"i C"（其中 i = 实例且 C = 类），∀ ∈读作"类*C*的每个实例*i* 都具有……"。

每个给定的陈述单元至少实例化一个基于谓词的陈述单元类和一个基于陈述类别的陈述单元类。例如，在知识图谱中对某个特定宾语的重量测量进行建模时，会使用多个三元组来表示；每个三元组都属于同一陈述单元的数据图，而该陈述单元会实例化*重量测量陈述单元*和*断言陈述单元*，其主语则是一个命名个体，代表某个物质实体（BFO:0000040）。

语义单元还为区分断言、偶然性陈述、典型陈述和全称陈述 **[形式语义] I5.**（参见 用户故事 *8*）。此外，语义单元提供了一种形式体系，用于表示全称陈述，而无需使用空白节点，因为它们可以

可使用某实例和所有实例资源，直接在知识图谱中对其进行建模和记录。因此，与 OWL 公理表达式相反，**全称陈述可以成为知识图谱论域的一部分，而不受本体类公理的限制**。语义单元还提供了用于将**否定和基数限制表示为实例图**（即 ABox）的形式体系，无需包含任何空白节点，并且**与 OWL 公理表达式相比，表示方式大为简化**，尤其是在使用动态标签或动态思维导图模式时（比较图 4 中的 C 与 D 和 E）。语义单元的这些"特性"显著提升了知识图谱的整体表达能力及其认知互操作性。

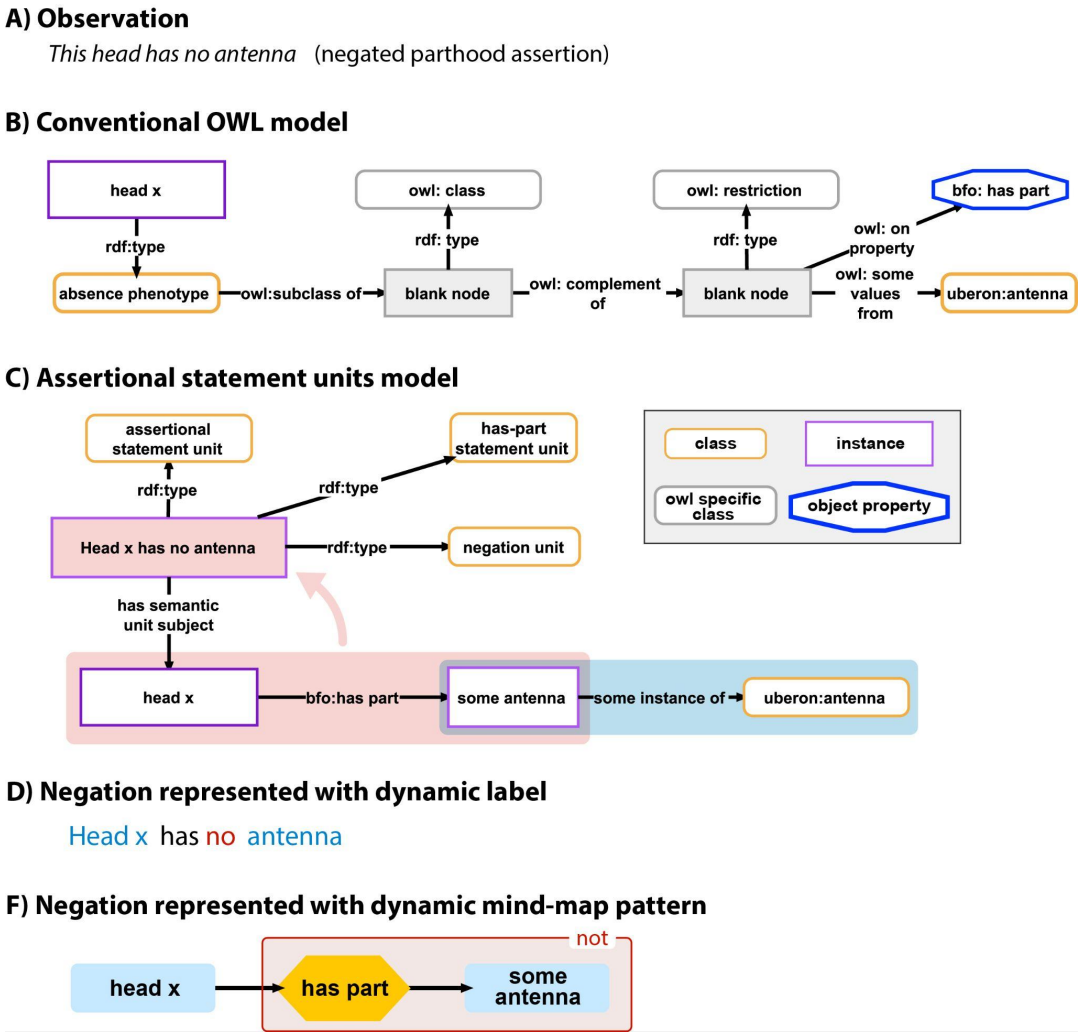

图 4：缺失观察、其基于OWL 的建模、其基于语义单元的建模，以及在用户界面（UI）中以人类可操作的方式表示该陈述的两种替代方法之间的关系。 **A)** 关于给定头部没有触角这一观察的人类可读陈述。**B)** 将A)中的断言转换为映射到 RDF的 OWL表达式。请注意，*缺失表型* 被定义为一组涉及两个空白节点的子类关系和补类限制。 **C)** 同一陈述可以使用两个语义单元进行建模。其中一个对具~~有部分关系进行建~~模并将其否定（带紫色边框的红色框及其动态标签，以及红色无边框框中的数据图）。因此，它既是*具有部分陈述单元* 的实例，也是 *断言陈述单元* 和 *否定单元* 的实例。另一个语义单元是 *某实例识别单元* 的实例，并通过属性*某实例属于*将"某个触角"与触角（UBERON:0000972）关联起来。其数据图显示在蓝色框中。~~二者共同对A) 中的观察进~~行建模。~~**E)、F)**利用陈述单元的动态显示模式，可以在~~知识图谱应用的用户界面中以人类可读的形~~式显示该图，~~既可以通过文本显~~示，也可以通过~~—~~

动态标签 E) 或通过动态思维导图模式 F) 以思维导图的形式呈现。 为使表示更加清晰，*B)* 和 *C)* 中未显示*"head x"* 与 *head (UBERON:0000033) 之间*的关系。 [图片取自 (14)]

通过为每个陈述单元类分配一个特定的图模型，规定该类型的陈述单元必须如何建模（例如采用 SHACL 形状规范 (17) 的形式；理想情况下，该规范具有自身的标识符，并包含其槽位的约束），语义单元也支持**模式互操作性**(14) **[F6.1;I7.1;I1.;I2.]**。通过纳入所采用逻辑框架的规范（例如描述逻辑或一阶逻辑），所有基于相同逻辑框架的陈述单元均可实现**逻辑互操作性** (14) **[I4.]**。

根据 SHACL 形状中定义的槽位，可以将特定资源的标签映射为带有相应变量的字符串，从而创建**动态标签**，用于在用户界面中对知识图谱进行可视化时呈现语义单元 **[E3.1;E3.2]** (见图 4D)。动态标签的生成方式是：解析主语资源的标签、各宾语资源的标签以及给定陈述单元实例中的宾语字面量，并根据预定义模板将它们组合成人类可读陈述。例如，模板"人物_主语 乘交通工具_宾语*A*，从出发地_位置_宾语*B*前往目的地_位置_宾语*C*，日期为日期时间_宾语*D*" 使用输入"*Anna* | 火车 | *Berlin* | *Rome* | *52019*年*8*月*5*日"后，将生成动态标签"*Anna* 乘坐火车 从*Berlin* 前往*Rome* ，日期为*2019*年*8*月*5*日"。

同样，可以为知识图谱用户界面中陈述单元的图形表示指定**动态思维导图模式 [E3。1]** （见图 4E）。与动态标签一样，动态思维导图模式解析陈述单元的主语和客体中的标签与字面量，并创建陈述的思维导图式图形表示。因此，这些表示不一定要遵循主语-谓语-宾语的 RDF 语法，多元谓词可以直接在图中表示，而不必转换为二元三元组陈述的集合。此外，由于动态思维导图模式旨在提高陈述单元的人类可操作性，因此不必显示陈述单元数据图中的所有信息，而可以仅限于对人类读者实际相关的内容 **[E3。]**。它们可以忽略为实现陈述单元的机器可操作性而额外需要的所有信息。

复合单元

复合单元是一种语义单元，它将陈述单元和其他复合单元组织成更大但仍然具有**语义意义的**数据图（图 5） **[I3.;E1.;E2.]**。从技术上讲，复合单元因此是语义单元的容器。它不拥有自己的数据图——合并其关联语义单元的数据图即构成该复合单元的数据图。与陈述单元类似，每个复合单元都是一种语义单元，在图中由其自身的节点表示，拥有自己的 UPRI，并实例化相应的复合单元本体类 **[E1.]**。可以区分出复合单元的几个子类别（23）：

1. **项目单元**是一个复合单元，它关联所有共享同一主语资源的陈述单元；该主语资源随后也会成为项目单元的主语资源 **[E2.;E3.2]**。例如，所有在主语位置包含代表某个特定用户的资源的陈述单元，都会与该用户的个人资料项单元相关联，其中包括关于用户名、电子邮件地址、主页等的陈述单元。项目单元表示处于**表征粒度的下一个更粗层级，即高于陈述单元层级**的语义单元。项目单元中包含的信息非常适合呈现在专用的用户界面页面上

或作为知识面板呈现，因为它包含以相应资源为主语位置所作的所有陈述。根据主语资源的类型，可以区分**实例项单元**（即命名个体资源）和**类项单元**（即类、某实例或所有实例资源）。

2. **项目组单元**是一种复合单元，包含至少两个项目单元；这些项目单元通过至少一个陈述单元在语义上彼此关联，该陈述单元的主语与其中一个项目单元的主语相同，其宾语之一则与另一个项目单元的主语相同 **[E2.;E3.2]**。我们将以这种方式把一个项目单元链接到另一个语义单元的陈述单元称为**项目链接**。所有通过此类项目链接链彼此关联的项目单元共同构成一个项目组单元。项目组单元中的信息非常适合以相互链接的用户界面页面集合呈现，其中每个页面显示一个项目单元的内容。例如，在描述物质实体的组成部分或食谱的混合步骤时，我们可以在相应的项目单元中描述每个组成部分或步骤，并在用户界面中将其作为单独页面呈现；其中一个陈述单元分别链接到描述其组成部分或子步骤的项目单元，从而形成一个相互链接的页面树，用户可以浏览和探索。由于项目单元可以双向相互链接，因此无法明确确定项目组单元的特定主语。如果其所有相关语义单元的主语都是命名个体资源，则该项目组单元称为**实例项组单元**；如果这些主语全部是某实例、每个实例或类资源，则称为**类项组单元**。项目组单元在**高于项目单元层级的下一个更粗表征粒度层级**上表示语义单元。借助陈述单元、项目单元和项目组单元，知识图谱可以组织为**五个表示粒度层级**，从三元组、陈述、项目、项目组一直到整个知识图谱（图6）（14,23）。

3. **粒度树单元**是一个复合单元，由两个或更多陈述单元组成，这些陈述单元组合起来表示一个**粒度树**（39–41） **[E1.2;E3.2]**。任何基于**偏序关系**的陈述单元类型，例如部分关系、类-子类包含关系或派生关系，都可以产生相应的**粒度视角**（42–44），特定的粒度树则是其中的一个实例。与粒度树单元相关联的陈述单元按照树形层级彼此链接，其中一个陈述单元的宾语是另一个陈述单元的主语。由于由此形成的嵌套结构及其从根节点到叶节点的隐含方向性，可以将粒度树单元的主语规定为满足以下条件的陈述单元的主语：这些陈述单元的宾语与其他陈述单元的主语相同，但其主语不与同一粒度树单元关联的其他陈述单元的宾语相同。粒度树单元以一种与上述表征粒度近乎正交的方式组织知识图谱。粒度树的一个例子是汽车装配部件的部分学，或多细胞生物的解剖部位的部分学，也可以是本体中类术语的分类体系。

4. **粒度项目组单元**是一种复合单元，它采用一个粒度树单元，并包含该树中每个资源所对应的项目单元 **[E2.;E3.2]**。换言之，它包含所有主语资源属于同一粒度树单元的项目单元。该粒度树单元的主语也就是粒度项目组单元的主语。如果知识图谱中的数据高度连通，项目组单元往往会变得非常庞大。粒度项目组单元可以为这类大型项目组单元提供额外结构，从而用于图探索（14）。

5. **上下文单元**是一种复合单元，它包含所有语义单元：合并这些语义单元的数据图后，会形成一个连通图，其中每个资源和三元组都能通过一系列三元组与其他每个资源相连，而这些三元组也属于该上下文单元的数据图；但属性为 *isAbout* （IAO:0000136）的三元组除外，因为它们属于一个**关于陈述单元**。~~关于~~陈述单元将一个信息人工制品与该人工制品包含相关信息的实体关联起来。因此，它们是指向图中参照框架发生变化位置的索引 **[E1.3; E3.2]**。因此，每个关于陈述单元都划定了图中两个不同参照框架之间的边界，也划定了两个不同上下文单元之间的边界。与粒度项目组单元类似，上下文单元还通过将项目组单元以及知识图谱总体组织成不同的上下文单元，进而组织成属于不同参照框架的子图，为它们提供额外结构。上下文单元的一个特殊情况是图中的词汇陈述单元集合。由于这些单元包含关于语言项目的陈述，它们共享同一个**术语参考框架**，因此构成一个**术语上下文单元**。

6. **数据集单元**是一种复合单元，定义为由特定语义单元组成的有序集合 **[E2.;E3.2]**。数据集单元可以是同质的，仅包含相同类型的语义单元；也可以是由不同类型的陈述单元和复合单元汇编而成。数据集单元的 UPRI 可用于标识和引用其他语义单元类别未涵盖的任意语义单元集合。例如，可用于引用已导入知识图谱的任意数据集，或指代由特定机构贡献或由特定用户创建的所有语义单元。

7. **列表单元**是由单个资源组成的特定有序或无序列表，其中列表成员身份通过成员关系陈述单元指定。

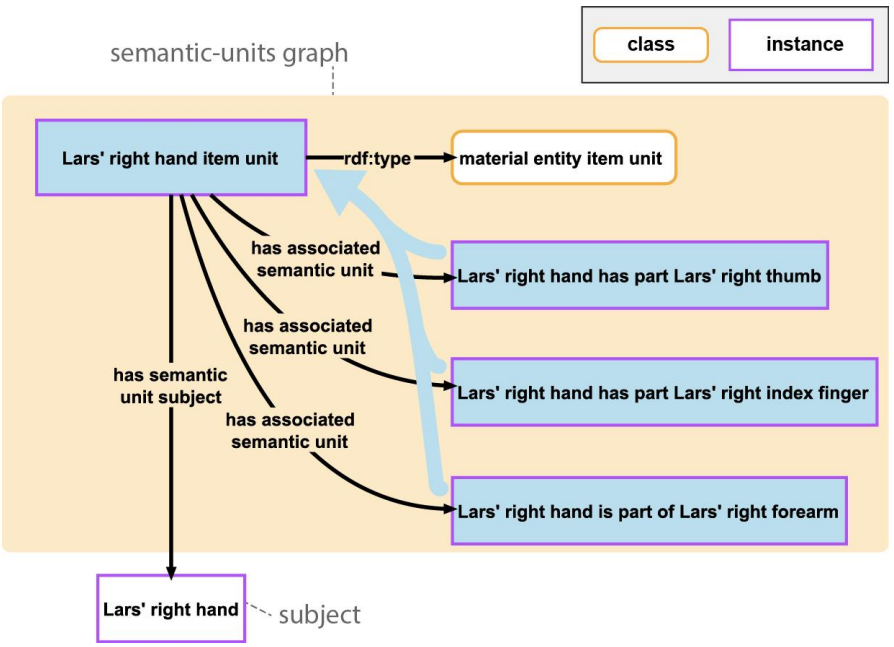

图 5：复合单元示例该复合单元由多个陈述单元组成。复合单元只能间接拥有数据图，即通过合并其关联陈述单元的数据图来获得。不过，复合单元资源（此处为 *'Lars' 的右手项目单元'*）代表合并后的数据图（蓝色箭头所示）。复合单元拥有语义单元图（显示在桃色方框中），用于记录与之关联的语义单元。 [图取自 (23)]

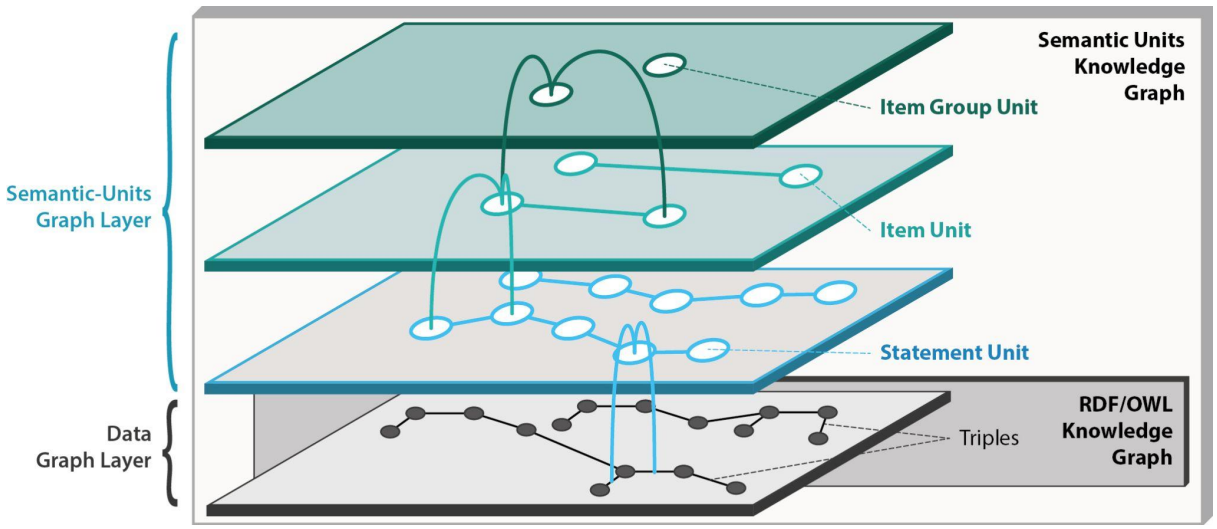

图 6：五个表征粒度层级。向知识图谱引入语义单元，会在其数据图层之上增加一个语义单元图层；该图层在三元组层级和整个图层级的基础上，增加了陈述单元层级、项目单元层级和项目组单元层级，从而形成五个表征粒度层级。 [图取自（23）]

问题单元

在**问题单元** (14)中，现有语义单元类被用作**来源**，并被"改写"为一个问题，该问题可以像其他语义单元一样记录在图中。这样一来，查询就成为知识图谱中的宾语。通过使用某实例、每个实例或类资源作为资源对象和主语的变量，并使用带有（未充分指定）值的数据类型规范作为字面量对象的变量，再将它们解释为任何问题单元的数据图中的通配符，相应的查询便会返回匹配子图。因此，陈述问题单元要么使用命名个体资源和带有完全指定值的数据类型规范来定义断言陈述单元的实例，并要求返回布尔答案；要么使用命名个体、某实例、每个实例和类资源的组合，以及带有未充分指定值的数据类型规范来定义陈述单元类，并要求返回该类的实例化陈述单元列表。如果我们将陈述单元理解为一个带变量的字符串（例如，*Anna* 于火车 上从 *Berlin* 前往*Rome* ，日期为*2019*年*8*月*5*日），并在知识图谱中使用特定图模式对其建模，且在陈述单元类中将其记录为多元谓词，那么我们就可以使用同一个字符串向知识图谱查询相应陈述：可以提出布尔真/假 问题（*Anna*是否于火车 上从*Berlin* 前往*Rome* ，日期为*2019*年*8*月*5*日？），也可以将某个变量替换为用于资源主语和宾语的某实例、每个实例或类资源，或替换为带有未充分指定值、用于字面对象的数据类型规范。这样，就可以在陈述单元中将给定的主语位置或宾语位置指定得不那么充分，例如，用*某城市*（ENVO:00000856）代替*Berlin*（geonames:2950159）。此时，它会在任何问题单元的数据图中充当未充分指定的主语或宾语，并提出一个假定答案为单个资源列表的问题（"某人 是否于火车 上从*Berlin* 前往*Rome*，日期为*2019*年*8*月*5*日？"）；也可以替换更多变量，并假定答案为陈述列表（某人 是否乘坐某种交通工具 从*Berlin* 前往*Rome* ，日期为日期*[*年份*2019]*？）。此外，将一个陈述问题单元中的宾语资源重新用作另一个陈述问题单元的主语资源，可以构建更复杂的问题单元。基于 SHACL 形状规范的

根据涉及的陈述单元类，查询构建器可以推导出相应的图查询，而知识图谱可以回答这些查询。问题单元与查询构建器及相应的用户界面相结合，不仅能提供一个直观易用的框架，让用户无需了解图查询语言即可指定 SPARQL 或 Cypher 查询（参见 用户故事 *3*） **[E3。4]**, 还可以通过在图中以问题单元的形式记录具体的能力问题，支持本体开发 (45)。

表示不同的参照框架，并区分知识图谱中的本体层、诊断层和话语层

交流通常涉及关于不同信息层的陈述，即本体层、诊断层和话语层（参见 (23)）。**本体层** 包含关于某种现实的陈述，涵盖领域知识和实证观察，例如“铅的熔点为 *327.5 °C*”。本体信息回答“这是什么？”这类问题，因此承载语义意义，并提升接收者的推理词汇能力 (46,47)。**诊断层** 提供有关如何识别给定概念或类的实例的信息。这通常包括指定某些方法并描述预期结果，例如“硫化钠试剂盒通过变黑或变灰来指示铅的存在”。诊断信息回答“它看起来是什么样？”或“我如何识别/辨认它？”这类问题，因此承载有关实体认识论表象的信息，有助于明确指称实体，并提升接收者的指称词汇能力 (46,47)。**话语层** 提供话语性的上下文信息，因为我们有时会对他人所作的陈述发表评论，例如“作者 *A* 声称铅的熔点为 *327.5 °C*”。此外，在交流过程中，我们会频繁切换**参照框架**。例如，当我们谈论一项以数据集为产出的研究活动，而该数据集又包含关于某个实体的事实（如对某种多细胞生物解剖学的描述）时，在描述研究活动时，我们采用研究活动参照框架；在描述多细胞生物时，则切换到研究对象参照框架。由于所有这些不同类型的陈述都可以使用相应类型的语义单元进行建模和分类，因此，在采用语义单元的框架中，将这三个层次及其可能的相互关系和参照框架的转换一并表示在知识图谱中十分简单 (23)。

比较语义单元与本体

本体之于种类词和术语学，正如语义单元之于陈述和陈述集合。要传达**含义**，总是需要不止一个术语——你需要将多个术语置于语境中，并通过谓词将它们联系起来。换句话说，这需要**陈述**。因此，如果我们想以 FAIR 的方式传达含义，就需要两者：术语和陈述。

基于 RDF/OWL 的本体，其类定义采用 OWL 公理的形式，本质上是由相互关联的术语构成的集合，每个术语都像一个占位符，对应一个或多个全称陈述——即该术语的 OWL 公理集合。不过，这些类公理只能

描述必然适用于该类每个实例的内容，并未涵盖针对该类某一给定实例集所能作出的所有陈述。

我们发现，由语言的句法和语法定义的句子结构，与数据和元数据模式的结构之间存在相似性（图 7）。自然语言句子中术语的**位置** 承担特定的语法和**语义角色** ，对陈述的含义有重要贡献（48–50）。表格数据模式或知识图谱中值或节点的位置同样如此。因此，我们可以将数据模式理解为将自然语言句子的结构转译为机器可操作的数据结构的尝试。语义单元通过提供这样的结构，为 FAIR陈述及其集合提供形式化表示和形式语义，而本体则为 FAIR术语提供形式语义。

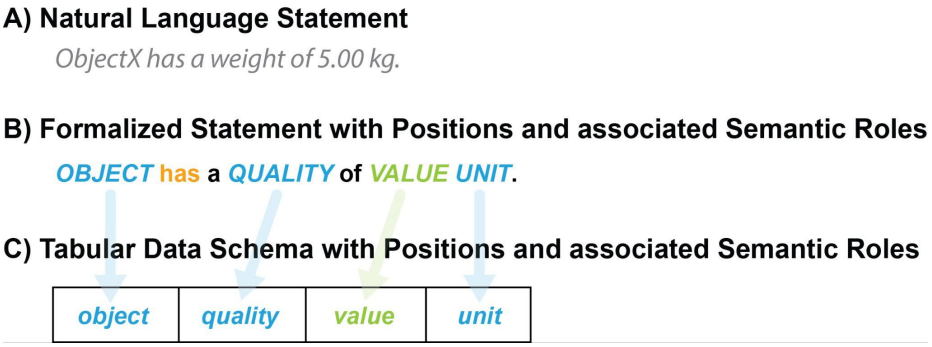

图 7：自然语言陈述、形式化陈述和表格数据模式的结构对比。A) 自然语言陈述。**B)**将该陈述的结构转换为形式化表示具有定义的位置及其关联的语义角色（以颜色表示——蓝色代表资源，绿色代表字面量）。**C)** 表格数据模式中的相同结构。

RDF三元组的结构无法提供与句子结构相对应的形式，因为其谓词不一定映射到自然语言陈述的谓词。正如我们上文所讨论的，许多谓词是多元的，而 RDF/OWL属性——至少从技术上来说——必然是二元的。因此，本体属性与自然语言谓词并非一一对应，我们通常必须用多个三元组来表示自然语言陈述。因此，如果我们想创建 FAIR 数据和元数据，并且认识到数据和元数据总是以陈述的形式存在，那么我们不仅必须使用受控词表来表示构成陈述所需的术语，还必须使用受控陈述类型，并为其关联相应的图模式，其中包含相应的位置和语义角色。语义单元通过语义单元类提供这类受控陈述类型，并为其关联谓词和形状规范；后者规定了如何在 RDF 中表示陈述。

要在术语之间建立互操作性（即**术语互操作性**(14)），就需要将不同本体中含义和指称相同的本体术语相互映射。本体术语之间的映射属于**同质定义映射** (51)：定义左侧只有一个需要映射的词汇元素，右侧则有多个无需映射的词汇元素[1]。

另一方面，要实现陈述之间的互操作性（即**模式互操作性** （14）），则需要对不同位置及其相关语义角色进行对齐，而这些位置和语义角色分布在

[1] 然而，要在两个含义和所指相同的本体术语之间建立完整的术语互操作性，也必须映射其本体定义（即类公理）中使用的术语，前提是这些术语的含义和所指也相同，因为术语的含义是通过其定义传达的，而定义以陈述的形式呈现。

同一陈述类型的不同句子结构中。换言之，对同一陈述类型进行建模的不同数据和元数据模式中的主语、谓词和宾语槽位必须对齐，并且其中的术语必须跨受控词表进行映射。这要求首先对这些模式进行形式化规定，例如，以指定为 SHACL 形状的图模式形式进行规定。随后，还必须对具有相同陈述类型指称的形状进行对齐和映射。这些就是**本体模式对齐**（即 TBox 对齐）（51）或**ABox 对齐**；在后者中，必须先对多个词汇元素进行对齐，然后在**模式交叉映射**中进行映射。

因此，我们可以总结如下：由于术语和陈述结构是可靠传达含义所必需的，我们需要用于**FAIR术语**的受控词表（即**本体**），以及用于**FAIR陈述**的**模式交叉映射**，以提供跨模式的术语位置对齐，从而实现数据和元数据的语义互操作性。后者可以借助语义单元来提供。

语义单元和更具FAIR性的知识图谱

不同类型语义单元的分类及其人类可读的内容描述，结合来自其主语的附加信息，以及在复合单元情况下来自其关联语义单元的附加信息，可提供相关信息，帮助用户探索图并识别图中令其感兴趣的部分。

语义单元的模块化和递归组织还支持**关于陈述的陈述 [E3.3]**。由于每个语义单元在语义单元图层中都由其自身的节点和自身的UPRI表示，因此可以使用该单元的UPRI对其进行断言。此类断言可以是关于特定语义单元内容的陈述，例如指定其确定性或置信度，或指明其时间有效性；也可以是将两个语义单元相互关联的陈述，例如指定一个语义单元所表达的经验观察与另一个语义单元所表达的因果假设相矛盾。通过将知识图谱组织成可识别且具有语义意义的单元，可以开发用户界面，以非常直观的方式实现关于陈述的陈述。此类用户界面及其配套工具和服务的开发，也因此受益于按照清晰且直接的模式实现语义单元。

语义单元可用于图对齐、子图匹配、知识图谱剖析，以及敏感数据访问限制的管理 **[A1.3]**。基于语义单元，可以为基于实例的图（ABox 表达式）指定用于建模缺失陈述、否定和基数限制的图模式。此外，将 FAIR 知识图谱组织为语义单元，有助于分离本体信息、诊断信息和话语信息。最后，将 FAIR 知识图谱构建并组织为语义单元，通常有助于提高其数据和元数据的**认知互操作性**，从而将图转变为**FAIREr 知识图谱**（FAIR + 可探索性得到提升），即一种**可由人类用户直观探索**的 FAIR 知识图谱，通过将图的复杂性降低到用户当前感兴趣的程度来实现这一点。如果您想进一步了解语义单元的概念和 FAIREr 指导原则，请参阅文献（14,23）。

知识图谱构建模块（KGBB）框架

将知识图谱的数据图层划分为陈述单元，进一步用复合单元对其进行组织，并将问题（即查询）记录为问题单元，这为我们提出由知识图谱构建模块驱动的知识图谱概念奠定了基础。**知识图谱构建模块（KGBB）是一种用于知识处理的小型信息模块，与某种特定类型的语义单元相关联**（另请参阅知识图谱单元（52），其中首次讨论了一个类似的想法）。根据陈述单元、复合单元和问题单元的区分，我们将陈述 KGBB、复合KGBB和问题型 KGBB区分为 KGBB 的顶级类别。

KGBB 仅提供信息，包括其关联语义单元类型的存储模型和显示模式。要使这些信息发挥作用，需要一个**知识图谱构建模块应用引擎（简称：KGBB 引擎）**来处理这些信息，并将其应用于相应的应用逻辑。KGBB 引擎使用来自不同 KGBB 的信息，并通过相应的应用程序接口与知识图谱应用的持久化层和表示层进行通信。KGBB 引擎还包含一个**KGBB 查询构建器**，它将基于 KGBB 提供的存储模型并依据这些模型进行结构化的用户基于表单的输入转换为 SPARQL、Cypher 或 MySQL 查询 **[A1.;A1.1]**, ，从而使 KGBB 驱动的知识图谱用户无需掌握（图）查询语言即可编写查询 **[E3.4]**。此外，KGBB 引擎支持对语义单元进行**版本管理**，实现**自动溯源跟踪 [F1.-F4.;F6.1/I7.1]**, ，并且理想情况下还会为每个语义单元提供**详细的编辑历史**。

每个 KGBB 都会提供 KGBB 引擎所需的信息，以管理知识图谱中属于其对应语义单元类型的数据和元数据。知识图谱中所有 KGBB 之间可能存在的交互，都在 KGBB 规范图中进行了规定。引擎负责管理知识图谱应用，而规范图通过将其限制为符合知识图谱范围的陈述和语境组合，定义了其**可能的数据与命题空间**。为此，必须针对知识图谱中使用的每种语义单元类型，规定一个 KGBB 及其与其他 KGBB 之间可能的交互。

除了相关 KGBB 集合和 KGBB 引擎之外，还需要一个 **KGBB 编辑器**，使不一定具备语义学、编程或图查询语言背景的领域专家能够描述新的 KGBB，包括其**显示模式**、与其他 KGBB 的连接和潜在交互，以及相应的语义单元类。KGBB 编辑器提供了一个可视化界面，还可用于随时为每个 KGBB 指定额外的**访问和导入模型**，从而能够适应不断出现的新数据格式和数据标准。

归根结底，KGBB 的设计理念是，每个 KGBB 都与 KGBB 引擎交互，从而独立于其他 KGBB 运行，并且 KGBB 可以存储在 GitHub**KGBB 仓库**中并向公众开放，以便其他人重复使用 **[F6.1/I7.1]**。每个 KGBB 都必须记录为一个本体类 **[F1.;I1.;I2.]** ，并将其所有属性继承给其子类。可以将现有 KGBB 及其对应的语义单元类加载到 KGBB 编辑器中，以附加属性的形式为其添加更多约束，然后将其与新的语义单元类一起保存，分别作为已加载 KGBB 和已加载语义单元类的子类。这样，所有 KGBB 共同构成一个 KGBB 分类体系，并形成与之对应的语义单元类分类体系。基于这一分类体系，便可以使用 KGBB 编辑器来描述某个特定的

通过采用这些 KGBB 类的实例并将它们相互关联，在**KGBB 规范图**中指定它们可能的交互，从而构建 FAIREr 知识图谱应用。

一个**KGBB 前端** 提供了基础用户界面，使用户能够访问由KGBB驱动的FAIREr知识图谱的内容，搜索特定信息，并通过文本和思维导图式图形展示浏览和探索其内容，从而受益于通过语义单元实现的图的模块化组织。KGBB 前端还提供用户界面工具，支持用户通过缩放、筛选和提供上下文信息来探索图，从而支持不同的**视觉信息查找策略**；这些工具利用图在陈述、项目、项目组、粒度树、粒度组、问题和上下文单元方面的结构与组织，从而提高其数据和元数据的**认知互操作性** ，并满足**FAIREr 人类可探索性原则 [E1.-E3.]** （详细讨论请参见（14））。

采用**六边形架构** 方法（53,54），可以开发**KGBB-Function**模块，为特定的KGBB或整个KGBB 驱动的知识图谱应用添加更多数据分析、处理和可视化功能。

KGBB、KGBB 规范图、KGBB 引擎、KGBB 前端和 KGBB 编辑器、KGBB 功能模块以及 KGBB 仓库共同构成了我们所说的 **KGBB 框架**。KGBB 框架使领域专家能够通过复用和组合现有的 KGBB，并根据需要创建新的 KGBB，以简便的方式构建自己的更具 FAIR 性的知识图谱应用。下面我们将详细介绍该框架。

通用语义单元图存储模型

为满足与模式互操作性相关的 FAIREr 指导原则**[F6.1-2;I7.1-2]** （14），知识图谱中属于同一语义单元的所有数据和元数据必须使用相同的模式以相同的方式进行建模。换言之，通过用户输入或数据导入添加到知识图谱中的同一类型的所有数据和元数据，以及所有自动溯源跟踪过程，都应分别应用相同的存储模型。因此，给定语义单元的总体存储模型必须同时涵盖其语义单元图存储模型部分和数据图部分（参见图3）。总体语义单元存储模型的这两个部分可以在各自的 KGBB 中彼此独立地指定，每个部分使用自己的 YAML 文件，并遵循 LinkML的表示法，该表示法可转换为实例SHACL 形状。

每个 YAML 文件都规定了带有约束的槽位/属性，前端和 KGBB 引擎可利用这些约束进行验证和输入控制。约束包括将数据类型指定为值域，例如“*xsd:float*”、最大值和最小值或模式约束，以及针对资源的类限制。

由于每种语义单元都有其对应的 KGBB，不同类型语义单元之间的类-子类关系分类体系会映射到相应的 KGBB 分类体系。每个 KGBB 类的存储模型规范会沿着这些类-子类路径继承到其 KGBB 子类。这使存储模型管理更加高效：对父 KGBB 存储模型的更改会自动传递给其所有子KGBB。

类 *KGBB* 位于 KGBB 分类体系的根节点。为此 KGBB 指定了一个**通用语义单元图存储模型 [I1.;I2.;R1.]** ，涵盖属于语义单元的语义单元图部分（参见图3中的桃色方框），并将其继承给所有其他 KGBB 类。

此存储模型主要涵盖与语义单元资源本身相关的溯源元数据；它不涵盖语义单元的数据图，因为只有陈述单元需要为其数据图指定存储模型。复合单元没有会对其数据图增加约束的约束，因为其数据图由相关陈述单元的数据图合并而成。因此，陈述型 KGBB 除了需要为其语义单元图部分指定存储模型外，还需要为其数据图部分指定存储模型（参见图3中无边框的蓝色方框），我们将在下文进一步讨论。

尽管继承了其父KGBB的存储模型，但任何陈述或复合KGBB都可以通过添加进一步的约束来补充该模型，从而限定有效主语以及有效客体的范围，以满足其所管理的特定类型语义单元的具体要求。这些新增的约束随后分别向下继承至其子KGBB。

KGBB 的语义单元图存储模型规范包括以下槽位（值域）（图8）：

1. **标签** （字符串）：用于人类阅读的标签，指示该语义单元所承载的信息类型，例如"重量测量"。此标签通过属性标签（RDFS:label）进行指定。

2. **类型** （语义单元类 *UPRI*）：创建特定语义单元类的新实例时，KGBB 引擎会自动将其归类为该类的实例，该类的 UPRI 由其管理KGBB指定。如果KGBB管理的是陈述单元，KGBB 引擎会将其归类为基于谓词的陈述单元类 **[F1.;A1.;E1.]**，并进一步将其归类为词汇陈述单元，或者根据其主语资源的类型，将其归类为断言性、偶然性（甚至原型性）或全称陈述单元 **[I5.]**。因此，每个给定的陈述单元始终至少被归类为两个陈述单元类的实例。这些实例化关系通过属性 类型 （RDF:type）进行指定。

3. **具有语义单元主语** （主语资源的 *UPRI*），可选：指定在此语义单元中建模的命题里占据主语位置的资源。属性 **hasSemanticUnitSubject** 指定该主语资源的 UPRI。每个陈述单元以~~及许多但并非所有复合~~单元都具有主语。

4. **KGBB_统一资源标识符** （*KGBB UPRI*）：每个语义单元都通过属性 **KGBB_统一资源标识符* 及其对应的 UPRI 值，指示哪个 KGBB 正在管理其数据。

5. **语义单元溯源 [F2.;F3;F4;A1.;R1.]**

a. **创建者** (用户*UPRI*)：使用属性 *创建者*，记录创建此语义单元的人。b. **创建日期** (日期时间)；使用属性 *创建日期*，记录此语义单元的创建时间。c. **创建所用应用程序** (知识图谱应用的 *UPRI*)：使用属性 *创建所用应用程序*，标识用于创建数据的应用程序。~~d. **导入来源** (导入数据集的 *UPRI*)~~，可选：使用属性 *导入来源*，标识导入此语义单元内容的数据集。e. **导入日期** (日期时间)，可选：使用属性 *策展日期*，记录导入数据集的时间。

f. **策展人** (用户 *UPRI*/软件 *UPRI*)，可选：使用属性 *策展人*，记录策展此语义单元的人。策展人可以是具有特定策展人角色的用户，也可以是软件代理。g. **策展日期** (日期时间)，可选：使用属性 *策展日期*，记录策展此语义单元的时间。h. **删除者** (用户*UPRI*)，可选：使用属性 *删除者*，记录删除此语义单元的人。仅当语义单元被删除时才会添加此项。~~语义单元~~资源本身不会被删除，而是会跟踪此信息，从而使语义单元的元数据仍可访问 **[A2.]**。i. **删除日期** (日期时间)，可选：使用属性 *删除日期*，记录~~删除此语义~~单元的时间。仅当语义单元被删除时才会添加此项。

---

6. **数据生产过程元数据 [F2.;F3;F4;A1.;R1.]**

a. **数据生产元数据** （描述数据生产过程的语义单元的 *UPRI*），可选：在某些情况下，语义单元包含的信息代表特定过程（例如，使用特定方法和特定仪器进行的测量过程）的输出，而这些过程的元数据对于评估该语义单元的数据图至关重要。由于过程描述通常包含多个陈述单元，因此本身也代表项目单元甚至项目组单元，所以此类元数据必须记录在其自身的语义单元中，包含数据的语义单元可通过属性 **dataProductionMetadata** **[F6.1/I7.1]**引用这些语义单元。

---

7. **版本ID**（版本 UPRI），可选：指示该语义单元是否属于其自身或与之关联的其他语义单元的某个特定已记录版本的信息。此信息通过属性 **versionID** 进行追踪。由于一个语义单元可以与多个其他语义单元相关联，因此此~~槽位可以~~多次实现。

8. **数据集单元ID** （数据集单元UPRI），可选：用于表明语义单元是否属于某个有记录的数据集单元的信息。该信息通过属性 **datasetUnitID** 进行跟踪。由于一个语义单元可以关联多~~个数据集单元~~，因此此槽位可以多次实现。

9. **可编辑** （布尔值）：用于表明语义单元应被锁定还是开放编辑的信息。如果不允许编辑，则值为‘*false*’。布尔属性 **editable** 表明语义单元是否开放编辑。

除了与每个语义单元一同直接存储的元数据之外，KGBB 引擎还会动态获取该语义单元及其所有关联语义单元的对象-位置实例中各创建者的贡献者列表（对象-位置实例详见下文）。它还会从同一组对象-位置实例及关联语义单元中最新的创建日期获取最后更新元数据，以及获取其中规定版权许可、任何访问权限限制和有关所使用逻辑

框架的法律声明元数据。引擎会将这些信息以“贡献者：”、“最后更新：”、“版权许可：”[2], “访问限制：”[3], 以及“逻辑框架：[4]”的形式提供给表示层。

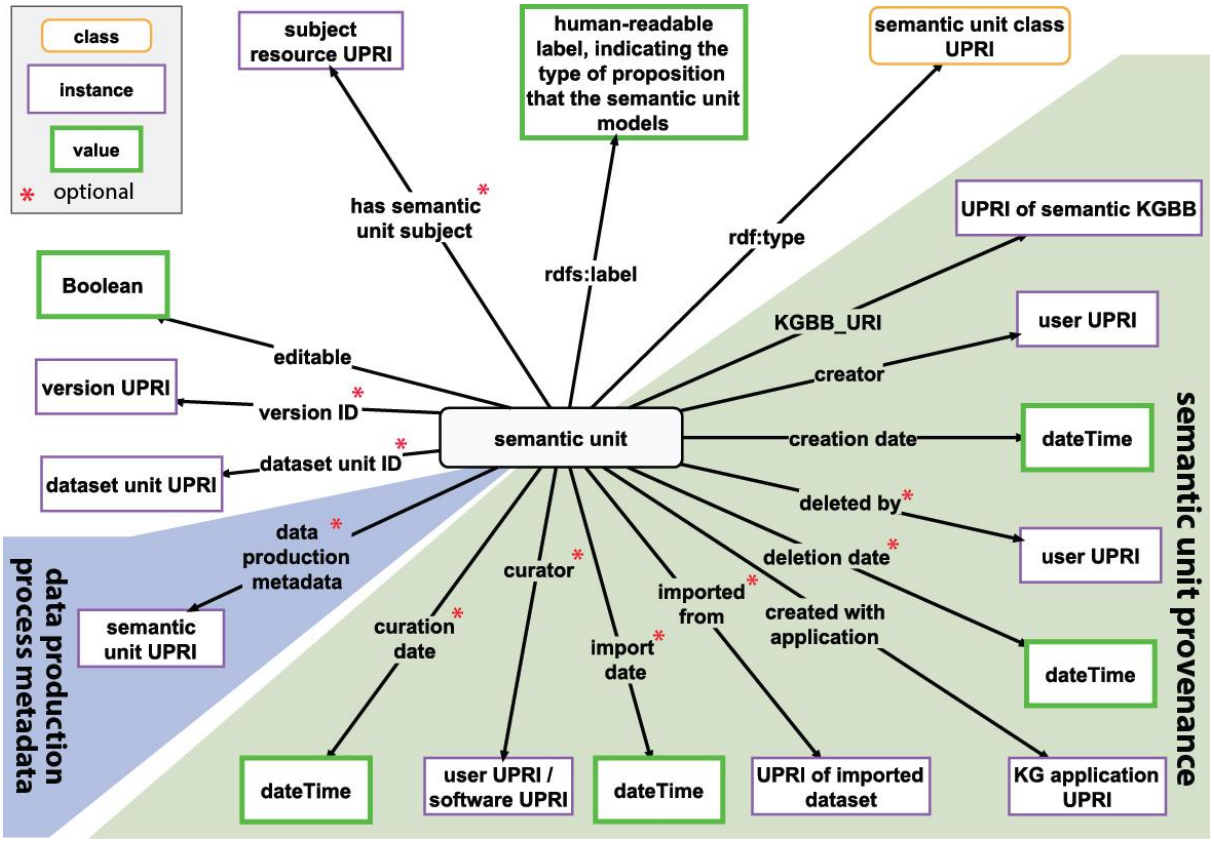

图 8：为 *KGBB* 指定的通用语义单元图存储模型。该图展示了每个语义单元的语义单元图中所追踪的信息。语义单元所追踪的不同类型的元数据以各自的背景颜色显示。某些关系是可选的，具体取决于上下文，并以红色星号标出；其他关系则可以多次实现。有些复合单元没有主语资源，因此省略了相应的关系。该模型从 *KGBB* 继承而来，并沿用至所有其他 KGBB 类，但也可以通过补充约束进行扩展，以满足任何特定 KGBB 类的要求。

陈述 KGBB

陈述单元表示最基础的语义单元类型，并实例化相应的陈述单元类。每个依据其命题所基于的谓词类型（即基于谓词的判据）定义的陈述单元类，都必须指定其专属的陈述 KGBB 类 **[F1;F4;E1.]**。陈述 KGBB 类提供以下信息：

1) 以机器可操作、标准化且符合 FAIREr 的方式，规定用于存储属于特定陈述单元类型的子图的数据模型。该模型基于上文所述的通用语义单元图存储模型，可通过相应陈述单元类型所需的其他规范进行扩展，并且

[2] 采用其所有关联语义单元中限制最严格的版权许可作为许可证。[3] 此信息用于指明哪些类型的语义单元包含相关信息，但由于特定的访问限制而无法访问，例如涉及人物相关信息时 **[A1。3]**，或包含有关濒危物种出现数据的地理引用时。[4] 此项指定可应用于数据的逻辑框架，例如使用 OWL 的描述逻辑，或采用通用逻辑交换格式（CLIF）的一阶逻辑 **[I4。]**。

由数据图的存储模型加以补充，该模型针对每个陈述 KGBB 而设。模型及扩展采用 LinkML 表示法进行规定，并提供输入约束，可供前端用于输入控制，也可供 KGBB 引擎用于输入验证 **[F6.1/I7.1;I1.;I2.;R1.;E1.1]**。

2）一个或多个**显示模板**（即数据视图），用于从机器可操作数据图中识别人类读者感兴趣的数据点。显示模板用于将陈述单元数据图中的信息转换为人类可读的文本陈述或思维导图式图形表示，并分别以动态标签或动态思维导图模式的形式呈现在用户界面（UI）中，从而**将数据展示与数据存储解耦 [E3.;E3.1]**。

3）可选地，一个或多个**访问模板**。借助访问模板，KGBB 引擎可以自动将符合存储模式的数据转换为符合既定格式和模式的数据表示，以便按照不同的图模式访问陈述单元的数据图，并按照不同的既定标准以各种格式导出数据和元数据，包括 RDF/OWL、CSV、JSON 和 Python 等。因此，**数据存储不仅与数据访问解耦，而且数据和元数据适用于其他应用程序的程度也得到提高，从而降低了软件开发人员使用知识图谱的门槛 [F6.2/I7.2]**。

4) 可选地，提供一个或多个**导入模板**，以支持将遗留数据导入知识图谱。5) 除了基于谓词的类型规范之外，还必须说明每个陈述单元属于词汇陈述、断言性陈述、偶然陈述、原型陈述还是全称陈述；陈述 KGBB 与 KGBB 引擎还必须为此类陈述提供相应的形式语义 **[I5.]**。

陈述 KGBB 存储模型

通用语义单元图存储模型的陈述单元特定扩展

当一个陈述单元将其某个宾语资源与另一个陈述单元的主语资源共享时，两个陈述单元的数据图便可彼此连接（另请参阅下文的主语—宾语连接），例如，在项目组单元中创建连接两个项目单元的项目链接时。因此，由于只有陈述单元拥有自己的数据图，因而需要附加元数据，从 KGBB 继承的通用语义单元图存储模型需要扩展，纳入以下附加槽位（值域）：

10.**由语义单元描述的对象** （语义单元的 *UPRI*），可选：如果陈述单元通过将其主语资源与语义单元 *A* 共享，并将其宾语资源作为语义单元 *B* 的主语，从而连接语义单元 *B* ，那么该陈述单元将通过属性 **objectDescribedBySemanticUnit** 与语义单元 *B* 建立关联。

11.**基于图模式** (数据图存储模型 *UPRI*)：对于每个陈述单元，都必须提供有关用于对该陈述单元的数据图进行建模的底层数据模式或图模式的信息。属性 **basedOnGraphPattern** 链接到该陈述单元的数据图存储模型的 UPRI。此信息对于评估知识图谱的哪些部分

具有可比性和互操作性至关重要，因此有助于提升数据的 FAIR性（参见模式互操作性(14)）**[F2.;F3;F4;F6.1/I7.1;A1.;I1.;I2.;E1.1]**。

12.**HasConstraintNode** （约束节点的*UPRI*）[5]：用于识别适用于特定输入事件、进而适用于特定对象位置实例的约束的信息。该信息与编辑用途相关，并且具有上下文相关性。此关系通过属性 **hasConstraintNode** 指定。约束节点资源随后通过属性 **hasConstraint** 链接到关于约束本身的信息，并以LinkML约束表达式作为其值；同时，通过属性 **appliesToObjectPosition** 链接到关于该约束所适用对象位置的信息，并以相应对象位置类的UPRI作为其值。

13. **法律陈述元数据**

a. **许可证** （版权许可*UPRI*）：关于使用属性 **license** **[R1.1]** 对陈述单元数据图内容所适用版权许可的信息。b. **AccessRestrictedTo** （用户组或用户角色*UPRI*），可选：如果知识图谱应用区分不同的用户组或用户角色，则可以使用属性 **accessRestrictedTo** **[A1.3]** 指定谁有权访问陈述单元的数据图。

14. **LogicalFramework** （表示逻辑框架的资源的 *UPRI*）：关于对该陈述进行建模时所依据的逻辑框架（例如描述逻辑、一阶逻辑）的信息，通过属性 **logicalFramework** **[I4.]**。

15. **完整性与真实性**

a. **ConfidenceLevel** （表示特定置信度的资源的 *UPRI*），可选：关于该陈述的置信度或确定程度的信息，通过属性 **hasConfidenceLevel** 表示。说明陈述的置信度非常重要，尤其是在学术语境中（55,56）；缺少置信度可能会导致引文失真（57）等问题。

b. **有效期**[6]

i. **ValidityStartDate** （*dataTime*），可选：关于该陈述有效起始日期的信息，使用属性 **validityStartDate**。ii. **ValidityEndDate** （*dataTime*），可选：关于该陈述有效截止日期的信息，使用属性 **validityEndDate**。

16.**Reference** （表示特定 *Reference* 的资源的 *UPRI*），可选，可多个：有关特定出版物的信息，该出版物是使用属性 **reference** 的陈述之证据来源。

除了与每个陈述单元直接存储的元数据外，KGBB 引擎还会动态获取当前版本陈述单元中所有对象位置实例（参见通用对象相关数据图存储模型）的导入来源溯源信息，并将这些信息以“包含从以下来源导入的数据：”的形式提供给表示层。如上所述，KGBB 引擎也会对该陈述单元的最后更新信息和贡献者溯源信息执行相同的处理。

[5] 本文稍后将讨论为何需要此信息。[6] 在许多情况下，特定陈述仅在某个时间段内有效，即为真。例如，陈述“乔治·*W*·布什 担任美国总统”的有效期开始日期为 2001 年 1 月 20th 日，结束日期为 2009 年 1 月 20th 日。

此外，我们将（23）中的陈述单元概念扩展为包含对陈述谓词的某些逻辑属性的说明，即其谓词是否是**传递**、**对称的**或**非对称的**。这些信息与陈述单元类一同存储，其中还包括**对陈述谓词含义的人类可读定义**。逻辑属性的说明不仅是识别用于建模偏序关系的谓词，以及由此可能产生粒度树的陈述单元类型所必需的，还可用于基于规则的函数，从而用于KGBB函数。

一种用于陈述型 KGBB 的通用数据图存储模型

知识图谱数据图层中的每个三元组，都是且仅是一个陈述单元的数据图的一部分。因此，陈述单元包含知识图谱的实际数据，包括用户输入的所有内容和数据导入的所有内容。将陈述单元理解为最基本的语义单元类型，并让所有其他语义单元都引用陈述单元，KGBB 框架还考虑到命题是人类交流的主要单位，因此以符合人类认知的方式组织知识图谱。因此，为保证知识图谱中所有数据和元数据的 FAIR性，每个陈述 KGBB 类都必须为其对应的陈述单元类型提供一个**数据图存储模型 [F6.1/I7.1]** 。随后，KGBB 引擎将这两部分的信息——来自陈述单元的语义单元图模型和数据图模型——作为通用查询模式的输入，用于通过 KGBB 查询构建器执行 CRUD 操作，即创建（=写入）、读取（=搜索）、更新和删除知识图谱中相应类型的陈述单元。数据图模型不仅定义图模式，还定义可用于输入控制的任何输入约束。每个陈述型 KGBB 都会为其关联的陈述单元类提供自身的存储模型，该模型可明确地作为其每个陈述实例的元数据进行引用 **[F6.1/I7.1]**。

每个陈述 KGBB 的数据图存储模型都为存储特定类型的陈述提供了模型。换句话说，我们必须找到一种良好的方式来建模命题。在构思这类模型时，我们决定刻意采用极简方法，优先选择**精简模型** 而非复杂模型，目标是**降低整体建模复杂性**。我们还希望**尽可能贴近自然语言陈述的结构** ，以满足通过提升数据和元数据的**认知互操作性** (14)使其更加 FAIR 的需求，从而**减轻整体建模负担**。

自然语言中陈述的谓词-论元结构对于大多数用户而言既广为人知又直观。我们希望在语义建模的语境下利用这一知识，使得指定新的陈述型 KGBB 及其对应的陈述单元类和相关存储模型成为一项简单直接的任务，而且无需任何语义学背景，因为**语义建模步骤可以自动化**。只有提出一种**高度通用的模型结构**，才能实现这一点。该结构必须适用于每一种陈述类型，而不受其**n元性**的影响。为了保持精简，它只应存储恢复命题含义所必需的信息；这应当等同于只存储用户创建新陈述时必须提供的信息。例如，对于重量测量陈述，无需创建图2顶部所示的完整子图，只需存储测量对象和品质的资源，以及值和单位即可，重点在于**始终能够重构给定陈述的原始用户输入或数据导入**

单位，以便将其用于并映射到各种访问格式和语法。该模型还必须满足的另一项标准是，能够无缝地从中推导出查询。

换言之，目标是开发一种总体通用结构，从中可以推导出所有可能的语句 KGBB 类的数据图存储模型，从而使 **KGBB 编辑器能够根据领域专家提供的一些一般信息，自动指定新定义的语句 KGBB 类的数据图存储模型**，而该领域专家不一定具备语义学方面的经验。

在寻找这种通用结构的过程中，我们意识到，当以非常抽象的层次比较不同类型陈述单元的命题时，可以区分出两个相互独立的一般特征，它们可用作定义通用建模结构的一般参数：

1）句法位置、语义角色、主题标签—论元和附加语的数量。每个陈述单元都对一个在语义上对人类读者有意义的最小独立命题进行建模（23）。可以根据命题底层的谓词（即关系）区分不同类型的命题，从而形成前文所述的基于谓词的陈述单元分类。如上所述，按照**谓词-论元结构**，每个谓词都有一个**配价**，用于确定其完成意义所需的**论元**的数量和类型。例如，一个具有部分陈述需要两个论元—主语代表拥有某个部分的实体，宾语代表该部分—从而形成命题“主语具有部分宾语”。如果不指定主语和宾语，具有部分谓词就无法完整表达其意义。**附加语**还可以与谓词发生关系，但它们不是完成谓词意义所必需的，而是提供可选信息，例如具有部分陈述中的时间戳说明：“主语具有部分宾语在时间戳”。主语短语和宾语短语是最常出现的论元和附加语。因此，我们可以说，每个陈述单元都通过将作为命题谓词的主语论元的资源，与作为该谓词的宾语论元和附加语的一个或多个字面量或资源建立关系，来记录一个特定命题。陈述单元中某个命题的主语论元，就是我们所称的该陈述单元的**主语**；宾语论元和附加语则是其**必需和可选的宾语**。每个陈述单元都有一个这样的主语以及一个或多个宾语。

根据这一模型，我们可以根据谓词在给定命题中关联的主语和宾语总数，区分不同类型的谓词（以及每个陈述单元所依据的不同类型的关系）。例如，“*Sarah* 与*Bob*见面”是一个具有**二元关系**的命题，其中“*Sarah*”是该陈述的主语，“*Bob*”是其宾语。如果我们在命题中加入日期，例如“*Sarah* 于*2021*年*7*月*4*日与*Bob*见面”，它就成为一个具有两个宾语的**三元关系**[7]。再加入位置，它甚至会成为一个**四元关系**，如“*Sarah* 于*2021*年*7*月*4*日 在纽约市与*Bob* 见面”。原则上，这种情况可以无限延伸，不过仍受限于人类读者能够理解多元关系的维度[8]。不考虑这一限制，根据其底层谓词所关联的主语和宾语数量，可以将命题区分为二元命题、三元命题、四元命题等。此外，**基于论元的区分以及**

[7] 基础形式本体2.0的许多属性实际上是三元关系，因为它们依赖时间（58）。例如：“主语 位于宾语_*A*于*t*”。[8] 人类在记忆中只能同时保持5—9个项目（59）。

附加语，我们可以区分出完成陈述谓词含义所必需的客体和可选客体。

谓词的每个论元和附加语都可以理解为具有特定的**位置**，并在陈述的句法中承担特定的**语义角色**。这与 Kipper 等人（48）的动词词库VerbNet有关。他们扩展了 Levin 动词类（49），为每个类加入句法框架的抽象表示，并明确对应**句法位置**（即句法树中的位置）与这些位置所表达的**语义角色**（即（48）意义上的主题角色）（图 9）。每个动词类都会列出该类实例的谓词-论元结构所允许的语义角色，以及这些实例共有的基本句法框架。

谓词-论元结构中的论元列表可以用一组**主题标签**来描述，这些标签取自预先定义的可用标签集合（例如施事、受事、主题等）；句法框架则用这些主题标签构成的有序序列表示。主题标签的作用类似于语义角色的描述符，这些语义角色会映射到给定句法框架中的相应位置（48）。

在PropBank中，Palmer 等人（50）通过为动词的论元编号，来定义单个动词层面的语义角色。第一个论元通常承担语义角色**施事**，第二个论元通常但不一定承担**受事**或**主题**。对于编号更靠后的论元，并未规定一致定义的角色。我们的方法定义了 Palmer 等人所称的**框架集**，即一组特定的语义角色，并定义一种存储模式来提供相应的句法框架，其中第一个论元映射到我们的主语位置。

鉴于每个数据项都是命题的结构化表示，数据模式无非是框架集的形式化表示，例如，其中的表格为某一类型的命题定义句法位置及其对应的语义角色。在许多情况下，它表示将自然语言陈述中人类可操作的句法结构转换为机器可操作的表格结构（另见图 7）。在这里，我们需要一种通用存储模型，既支持重构数据项所依据的实际命题，又通过形式化结构对其加以记录，以确保数据具备人类和机器可操作性。将陈述抽象为其句法位置及对应的语义角色，似乎是满足这两项要求的可行方案。

由于我们无需能够表示自然语言陈述的全部表达能力，因此可以将范围限定为结构相当简单的陈述：由主语、及物动词或谓词，以及若干宾语构成。我们不考虑被动形式，也不考虑时态（后者可通过在陈述单元层级指定有效期来处理），并且无需区分动词或谓词表达其论元时可能采用的不同句法交替形式。**因此，我们的存储模型类似于一种高度简化的框架集，其中指定了一个主语位置，以及若干必需和可选的宾语位置；每个位置都关联一个以主题标签形式表示的语义角色，以及相应的约束规范**。它的结构是对句法树结构的抽象，因此也是由短语构成的树。这种树用于将发送者头脑中的思想网络转化为一串词语，使接收者能够理解这些词语，并将其还原为思想网络（60）。换言之，它基于一种与人类认知条件可互操作的结构。

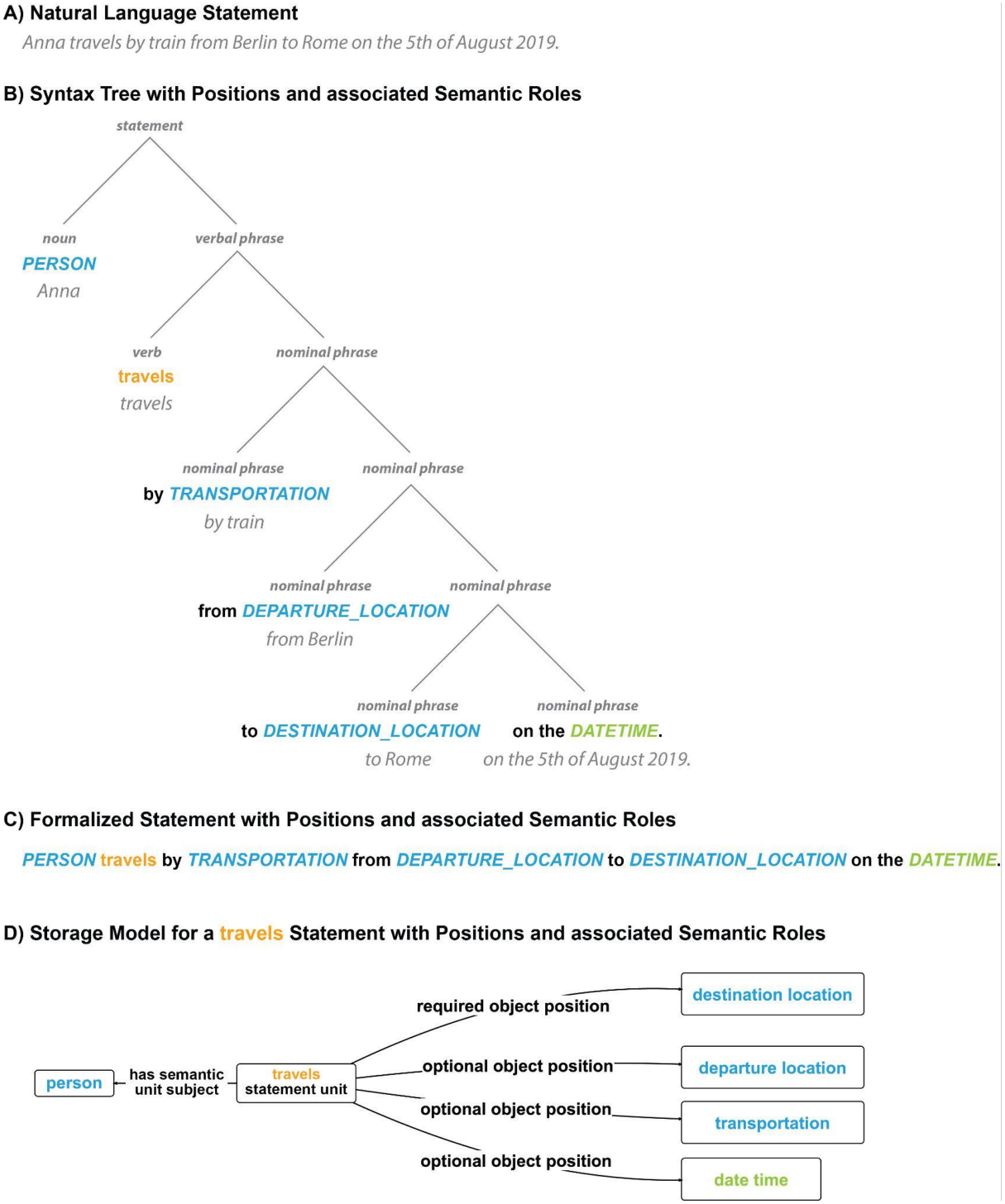

图9：从自然语言陈述的结构到该陈述在知识图谱中的形式化存储模型结构。A）包含动词 *travels* 的自然语言陈述。**B）**对应的句法树，其中句法位置及其对应的语义角色以不同颜色标示。**C）**将其转换回形式化的自然语言陈述，并保留相同的句法位置和语义角色。**D）**将形式化的自然语言陈述转换为图模式，涵盖相同的句法位置和语义角色。

2）宾语类型与宾语位置类。再来看上面的例子，我们会发现，当我们在知识图谱中对陈述“*Sarah* 遇见了*Bob* 在*2021*年*7*月*4*日”进行建模时，宾语“*Bob*”和“*2021*年*7*月*4*日”的建模方式会有所不同。“*Bob*”很可能会被建模为实例化 person 类（foaf:Person）的资源，而“*2021*年*7*月*4*日”则很可能会被建模为与数据类型xsd~~:date~~相关联的字面量。因此，除了上述提到的

区分论元（即主语和必需的宾语）与附加语（即可选宾语），并为每个论元和附加语指定相应的语义角色和主题标签后，便可按宾语类型将其区分为资源和字面量，并指定**基于其相应语义角色的类/数据类型约束**。资源则可以是命名个体、某实例、每个实例、类或属性资源的URI。从此处起，我们将其称为**资源对象**，将基于字面量的宾语称为**字面量对象**。命题可根据其包含的资源对象数量和字面量对象数量进行区分。

在陈述 KGBB 中，每个具有区分性的宾语位置都在其自身的 **宾语位置类**中定义。此类的实例随后在特定陈述单元中承担相应宾语位置的角色，数据输入通过链接到该宾语位置实例的方式予以记录。宾语位置只有在给定陈述类型的上下文中才有意义，因此也只有在给定的陈述 KGBB 及其关联的陈述单元类的上下文中才有意义。引入宾语位置类是为了满足下文所讨论的版本管理和历史跟踪方法的特定要求（参见版本管理与编辑历史跟踪），同时也记录适用于其相应输入类型的输入限制。

给定存储模型所区分的对象位置类数量取决于命题底层类型的多元性，因此具有 KGBB 特异性。陈述单元的每个对象位置实例都通过特定属性链接到该陈述的主语资源；该属性的类型取决于其所链接的宾语是否为必需项，即属于必需对象位置还是可选对象位置，分别由属性 *requiredObjectPosition* 和 *optionalObjectPosition* 表示。这一点在类层面也有说明：每个陈述单元类以及每个陈述 KGBB 类都分别通过注释属性 *hasRequiredObjectPositionClass* 和 *hasOptionalObjectPositionClass* 链接到相应的必需对象位置类和可选对象位置类。

这两个标准，即：(i) 具有相关语义角色的句法位置，具体包括一个主语位置以及若干必需对象位置和可选对象位置；以及 (ii) 与之对应的对象位置类及其各自的对象类型——资源或字面量——为我们的 **面向陈述 KGBB 的通用数据图存储模型方法**[9]设定了维度和边界：主语位置、谓词、必需对象位置和可选对象位置，以及具有语义角色规范的相应对象位置类，共同构成了对图中任何类型的陈述进行建模所需的全部元素。

然而，数据图存储模型仍缺少的是：对与对象位置实例链接的信息的存储模型进行规范。因此，数据图存储模型必须由两部分组成：(1) **主体相关数据图存储模型**，它将主语资源链接到各种对象位置实例，并且会因陈述单元所依据的命题类型而异；(2) 通用的**对象相关数据图存储模型** ，适用于陈述单元数据图中的每个对象位置实例。

总体数据图存储模型针对*陈述 KGBB*进行规定，并由其沿类-子类路径继承至所有其他陈述 KGBB 类。客体相关部分主要涵盖与每个对象位置实例相关的溯源元数据。它与主体相关部分共同构成数据图存储模型；该模型进一步与从*KGBB*继承而来的通用语义单元图存储模型相结合，形成陈述 KGBB 的总体存储模型。该总体存储模型由*陈述 KGBB*沿类-子类路径继承至语句 KGBB 的所有具体类型。

[9] 请记住：只有陈述单元才有数据图，因此需要数据图存储模型。

主体相关数据图存储模型
根据底层谓词的论元对象和附加语对象的数量，以及由此决定的陈述关系的 n 元性，特定的陈述 KGBB 可以包含多个指定类型的对象位置——每个论元对应一个，每个附加语对象位置对应一个。与主语相关的模型区分这两种对象位置：一种用于指定论元对象，另一种用于指定附加语对象（另见图 11）：

1. **必需对象位置** (此论元类型专属的对象位置类实例的 *UPRI*)：指定与某项输入相对应的对象位置类实例的 UPRI。该输入是完成陈述中谓词含义所必需的，因此也是论元所必需的。此项通过属性 **requiredObjectPosition** 进行记录。2. **可选对象位置** (此附加语类型专属的对象位置类实例的 *UPRI*)，可选：指定与某项可选输~~入相对应的对象位置类~~实例的 UPRI，因此也指定一个附加语。此项通过属性 **optionalObjectPosition** 进行记录。

在存储模型的 LinkML 规范中，必需对象位置所属的槽位声明为 '*required: true*'，表示此槽位必须有一个值。

例如，在带时间戳的 has-part 陈述 KGBB 中，主体相关数据图存储模型有三个链接到主语资源的宾语位置。其中两个是附加语的可选位置，用于指定时间间隔，因此对应字面量对象；必需对象位置则指定主语的部分，因此是资源对象。因此，此带时间戳的 has-part 陈述 KGBB 的主体相关数据图存储模型由一个必需对象位置和两个可选对象位置组成。

对象相关数据图存储模型
如前所述，存储模型中所区分的宾语位置数量取决于 KGBB，每个对应陈述单元中的对象位置资源都记录为特定对象位置类的实例。KGBB 类通过注释属性 **hasObjectPositionClass** 与其对象位置类相关联。因此，每个对象位置资源都会实例化其相应的对象位置类，并记录其对象位置的实际输入，以及有关~~特定输入事件的元数~~据。因此，可以将 **特定陈述单元的对象位置实例理解为记录属于特定对象位置的特定输入事件**（例如，具有部分陈述中的部件指定位置）。

输入本身取决于特定的陈述 KGBB，其相关的任何约束都必须在 KGBB 类级别进行指定，而元数据的存储方式规范则具有通用性，适用于任何陈述单元，因此是在 *statement KGBB* 中指定的，并由其继承至所有其他陈述 KGBB 类。通用~~对象相关数据~~图存储模型的规范包含每个对象位置的以下槽位（值域），无论该对象是必需对象还是可选对象（图 10）：

1. **对象位置类型** (对象位置类的 *UPRI*)：用于识别此实例所属对象位置的信息。这样，应用程序便可以在图中的带置信区间的测量陈述单元内，区分均值对象位置与上、下置信区间值对象位置。

此关系通过属性 类型(RDF:type) 指定。

2. **输入类型标签** (字符串)：为人类读者提供信息，以识别宾语位置的输入类型。它承担主题标签的功能，并通过属性 *_inputTypeLabel_* 进行追踪，其值为人类可读的标签（例如“重量单位”或“重量值”）。或者，也可以由相应的宾语位置类提供此信息，并从中获取。

3. **输入**；通过用户输入或数据导入提供的实际输入，具体取决于宾语是资源对象还是字面量对象。将输入链接到宾语位置实例，而不是将该实例用作实际输入，可以使资源对象和字面量对象以相同方式处理（区别仅在于它们各自链接的输入不同），并且可以追踪每个输入事件的元数据。不过，任何输入约束的规范都必须在相应的宾语位置类中进行指定。

a. **资源URI** (资源*UPRI*)，仅适用于具有资源对象的宾语位置：属性 *_resourceURI_* 的值为输入资源的UPRI，用于记录对象资源；该对象资源表示命名个体、某实例、每个实例、类或属性资源中的一种 **[F1.;A1.]**。这仅适用于 b. **字面量** (字面量)，仅适用于具有字面量对象的宾语位置：属性 *_literal_* 的值用于记录对象字面量。

4. **逻辑属性**(指定逻辑属性的 *UPRI*)，可选：通过属性 *逻辑属性*，说明陈述所依据的谓词的某一特定逻辑属性（即传递性、对称性、非对称性）是否适用于此宾语。

5. **当前版本** (布尔值)：用于指示对象位置实例是否包含属于此陈述单元当前版本的输入，或包含已更新、因此已过时的输入的信息。跟踪语义单元的编辑历史需要此信息，具体内容将在下文进一步讨论。布尔属性 *_currentVersion_* 表示对象位置实例是否属于该单位的当前版本。

6. **输入溯源 [F2.;F3;F4;A1.;R1.]**

a. **创建者** (用户*UPRI*)：关于谁使用属性 *_creator_* 将此输入添加到数据图中的信息。
b. **创建日期** (日期时间)：关于何时使用属性 *_creationDate_* 将此输入添加到数据图中的信息。 c. **创建所用应用程序** (知识图谱应用的 *UPRI*)：关于使用哪个应用程序通过属性 *_createdWithApplication_* 将此输入添加到数据图中的信息。 d. **导入来源** (外部数据集的 *UPRI*)，可选：关于此输入是从哪个外部数据集导入的，使用属性 *_importedFrom_* 表示。

7. **版本ID** （版本UPRI），可选：用于指明该对象位置实例是否属于此陈述单元或与此陈述单元相关联的某个语义单元的特定已记录版本的信息。此信息通过属性 *_versionID_* 进行追踪。由于给定输入可能属于该陈述单元的多个版本，且一个陈述单元可能与多个其他语义单元相关联，因此此槽位可以多次实现。

8. **数据集单元ID** （数据集单元UPRI），可选：用于指明包含此特定对象位置实例的陈述单元是否属于某个特定已记录数据集单元的信息。此信息通过属性 **datasetUnitID** 进行追踪。由于一个陈述单元可能与多个数据集单元相关联，因此此槽位可以多次实现。

在存储模型的 LinkML 规范中，属于必需宾语位置的槽位声明为“*required: true*”，可以实例化多次的槽位声明为“*multivalued:true*”；而槽位的任何值约束都可以声明为具有“*range:*”，其值可以是数据类型规范，也可以是类的统一资源标识符；对于任何字面值，还可以声明“*minimumvalue:*”“*maximumvalue:*”和“*pattern:*”。可以使用这些声明，在 LinkML 中指定特定宾语位置的字面对象和资源对象的约束。

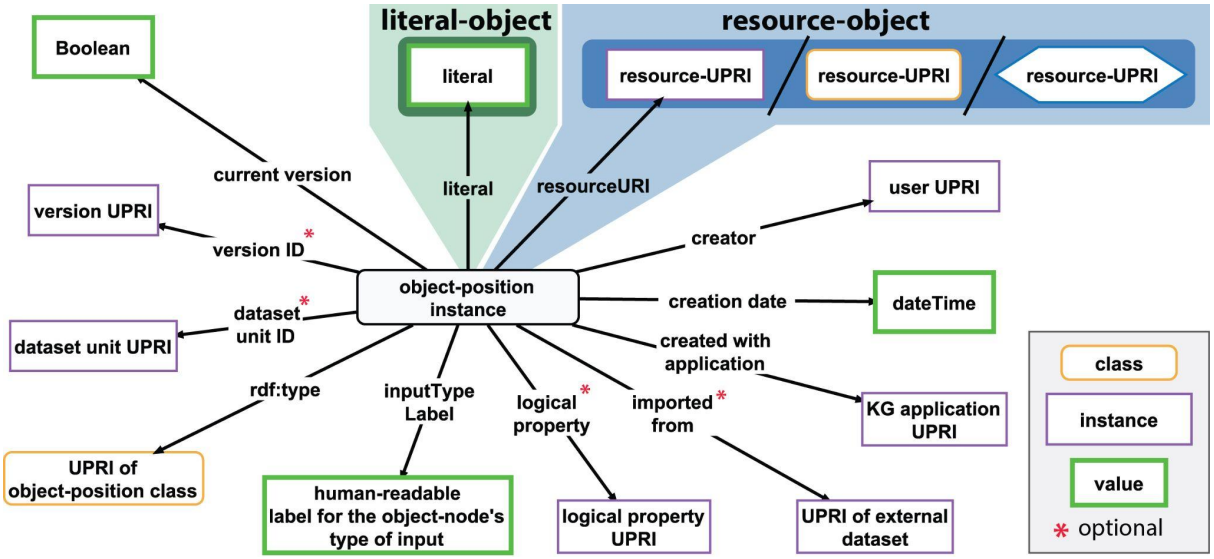

图10：*陈述 KGBB* 所规定的通用对象相关数据图存储模型。该图展示了每个客体节点所跟踪的信息。根据客体节点的类型，图中的部分内容不会被实现（字面对象未实现的部分以蓝色背景表示，资源对象未实现的部分以绿色背景表示）。某些关系是可选的，具体取决于陈述单元是否属于版本化语义单元或数据集单元、输入是否来自外部数据集的导入，或者逻辑属性是否适用（红色星号表示）。某些关系可以实现多次。该模型从*陈述 KGBB* 继承至所有其他陈述 KGBB 类，但可以补充额外约束，以满足任何特定陈述 KGBB 类的要求。相应陈述单元所依据命题的客体，在字面对象的情况下以带圆角的绿色框表示，在资源对象的情况下以带圆角的蓝色框表示。

创建新的陈述单元时，KGBB 引擎会根据其主语位置上的资源类型自动对其进行分类。如果该资源是**命名个体资源**，则该陈述将被归类为**断言陈述单元**，并且 KGBB 引擎只允许在该陈述的任一资源宾语位置上使用命名个体资源。如果主语资源是**类** 或**所有实例资源**，KGBB 引擎会将该陈述单元归类为**全称陈述单元**，并且只允许在其任一资源宾语位置上使用某实例资源或类资源。对于主语位置上的**某实例资源**，情况稍微复杂一些。在这种情况下，用户必须决定将该陈述单元归类为**偶然陈述单元还是原型陈述单元**，这会对推理产生逻辑影响（23）。不过，无论哪种情况，KGBB 引擎都只允许在该陈述的任一资源宾语位置上使用某实例资源。

整体陈述 KGBB 存储模型——组合各个部分
总而言之，对于任何给定的陈述 KGBB 类，其存储模型由三个部分组合而成（见图 11）：

1. **通用语义单元图存储模型**，如上所述，通过增加九个槽位进行扩展；2. **数据图存储模型的主体相关部分**；以及 3. **数据图存储模型的客体相关部分**。

每个部分都可以通过添加输入约束，并指定从 *KGBB* 到任何特定陈述 KGBB 类的类-子类路径上的宾语位置来扩展。这些约束限定了对于相应陈述 KGBB 类而言，哪些主体和客体被视为有效。

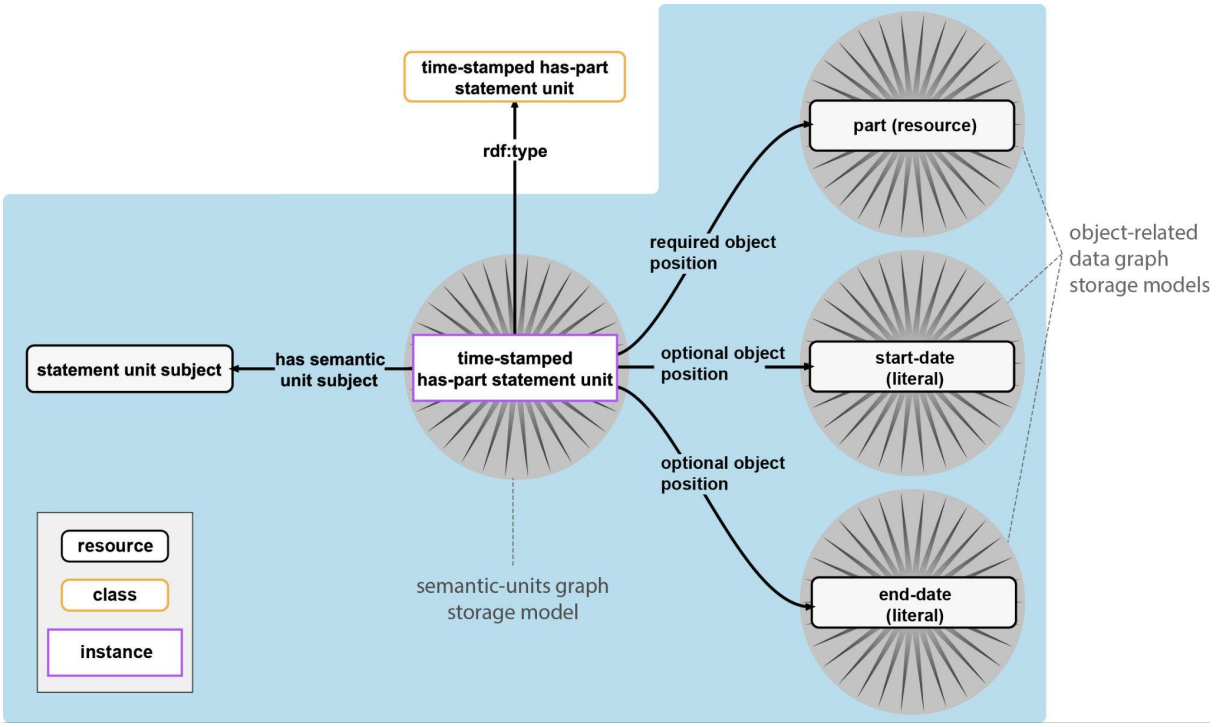

图11：四元带时间戳的部分关系陈述 KGBB 类存储模型示例。带时间戳的部分关系命题有一个必需的资源对象，用于指定部分；还有两个可选的字面量对象，用于指定时间间隔。相应的带时间戳的部分关系陈述 KGBB 类的存储模型由以下部分组成：1）数据图，用于对主语与特定部分关系命题的客体节点之间的关系建模；该命题由陈述 KGBB 建模，此处以蓝色方框表示。该图遵循四元命题的通用存储模式，其中包含一个必需的资源宾语位置和两个可选的字面量宾语位置——该数据图对具有部分关系建模这一点，可以从陈述单元实例化了 *带时间戳的部分关系陈述单元* 类这一事实推知。2）数据图在语义单元图中的表示形式，即单个资源，也就是陈述单元资源；该资源又实例化了 *带时间戳的部分关系陈述单元* 类，并链接到数据图的主语资源。3）链接到每个客体节~~点的客体相关信息。四个带有向外~~射线的灰色圆圈标示了该图的补充位置，以及由哪些存储模型部分提供所需的图模式。

存储模型包含用于主语的槽位、用于各种宾语位置及其关联宾语输入的槽位，因此必然涉及一组变量，KGBB 引擎在创建新的陈述单元时必须指定这些变量。在 KGBB 引擎能够指定一个

适当的创建查询之前，必须根据用户提供的输入、依据上下文自动生成的输入或通过数据导入提供的输入来指定这些变量。反过来，输入可以根据为每个宾语位置指定的约束进行验证，任何导入的数据也可以根据总体存储模型进行验证。图11展示了如何组合存储模型的不同部分，以形成陈述 KGBB 类的总体存储模型。

访问模板

为陈述 KGBB 类（或任何其他类型的 KGBB 类）指定的每个存储模型，都可以关联一个或多个访问模型，KGBB 引擎可利用这些访问模型，自动将来自该单位的数据图或语义单元图的数据和元数据，转换为遵循访问模型所规定的模式和格式的相同数据和元数据表示形式。

由于存储模型会单独考虑每个宾语以及每个输入，并为其分配自身的宾语位置，因此可以系统地识别和重构原始用户输入或数据导入。因此，数据图的存储模型及其所有相关信息可以与任何访问模型及其主语资源相适配，并且可以将不同的宾语位置映射到访问模型中相匹配的位置，从而形成**模式交叉映射 [F6.2/I7.2]**（见图 12）。访问模式中无法直接与主语或存储模型的任何宾语位置相适配的所有资源，通常会被添加到访问模型中，以便为机器提供更多上下文信息，从而正确处理和分析数据，实现对数据的推理，改进查询，或满足既定的社区标准。

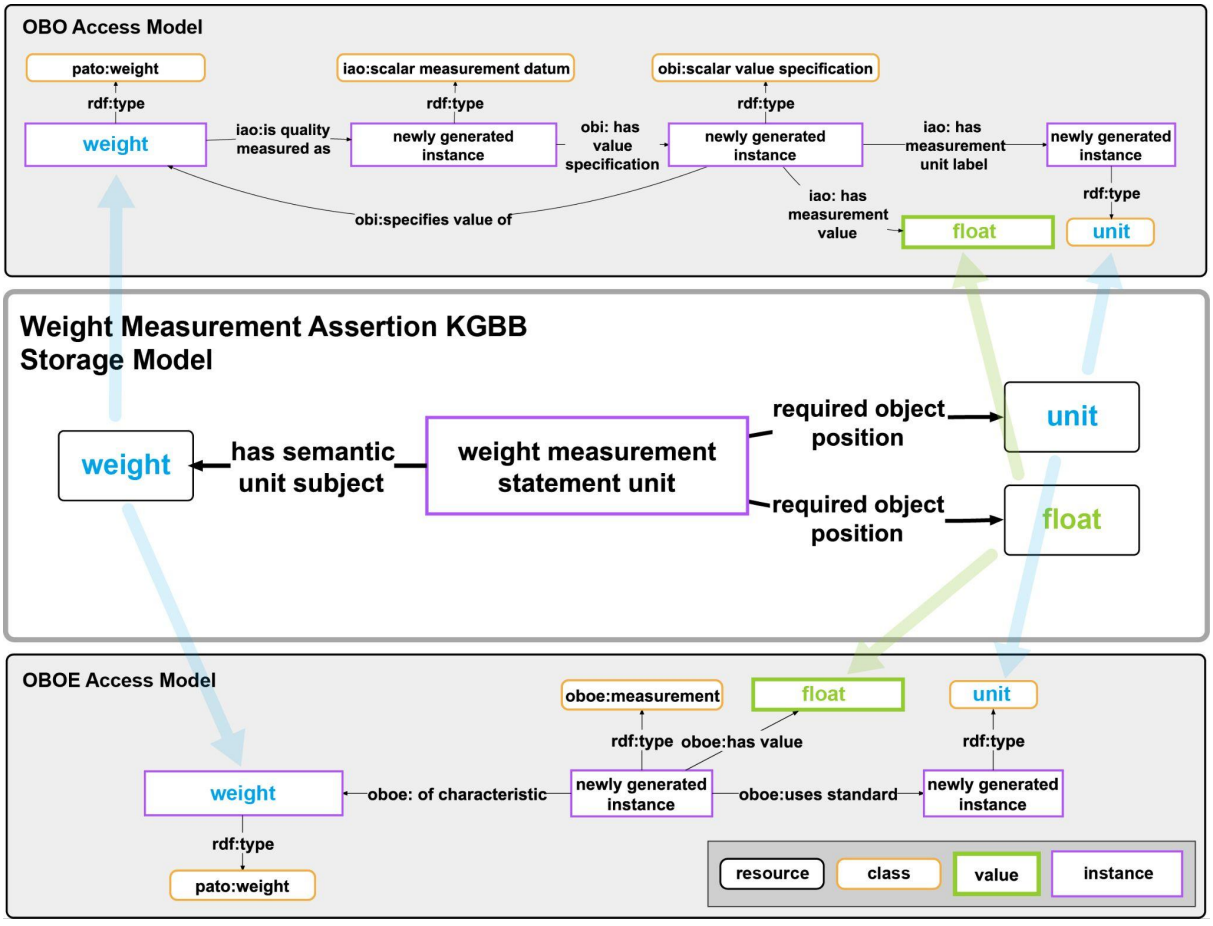

图 12：将陈述 KGBB 类的主体相关数据图存储模型映射到其两个访问模型。中部：重量测量陈述 KGBB 类的主体相关存储模型。**顶部**和**底部：**两个不同的访问数据模型。请注意，存储模型中的变量会在访问模型中重复使用，因此可以

可以轻松地从存储映射到访问模型（见箭头）。访问模型为资源引入了额外的变量，这些变量必须在每次数据导出时重新创建（"新生成的实例"框）。

与特定语义单元或其对象位置实例之一相关的所有溯源和元数据，都可以映射到任何溯源和元数据模式或模型，前提是已指定相应的访问模型，例如使用 PROV-O、PAV 本体 (61)、DublinCoreMetadata Initiative 或 DataCite 模式，从而生成**元数据模式交叉映射 [F6.2/I7。2]**。

除了支持根据不同图模型访问数据和元数据外，KGBB 引擎还必须使数据和元数据能够以不同格式访问，例如 RDF/OWL、GraphQL、Python 或 Java 数据类、JSON 和 CSV。由于所有这些格式都必须为给定的陈述单元类提供数据槽位，并将其映射到相应的位置及相关语义角色，因此，映射到非图格式的方式应与映射到图格式的方式类似。

使用 KGBB 编辑器，可以根据需要为 KGBB 类指定并添加新的访问模型和格式，从而适应新兴的数据和元数据格式及标准[10]。这考虑到了这样一个观察结果：FAIR性不足以作为数据和元数据具有高品质的指标——数据的使用往往取决于其**适用性**，也就是说，数据必须以适当的格式提供，并符合既定的标准和协议，以便能够直接使用；例如，特定的分析软件要求数据采用特定格式时，便是如此。

访问模型可以像存储模型一样，沿 KGBB 类分类体系继承。此外，给定的访问模型可以与一个或多个**访问模型族**关联。通过引用特定的访问模型族，用户可以访问由不同 KGBB 管理的数据，每个 KGBB 都应用属于特定访问模型族的访问模型。这样，用户便可以使用一套一致且可互操作的模型（例如，符合 OBO/OBI 的数据模型）访问图中的数据。

每个访问模型都应尽可能提供相关文献参考、策展人以及可应用于该模型的逻辑框架的信息 **[I4.]**, 用于标识图模式的 UPRI **[F6.1/I7.1;A1.;I1.]**, 以及典型的溯源数据 **[F1.;F2.;F3;F4;A1.;I1.;I2.]**。

陈述型 KGBB 及其对应的陈述单元类，可以理解为一种形式化的多元谓词建模方法。每个陈述单元类都表示一个多元谓词，该谓词被建模为本体类，而非本体属性。因此，在 KGBB 框架中，对传递性或定义域和值域规范等属性公理进行推理并不简单，而且基于 OWL 的框架所使用的工具无法直接在 KGBB 框架中重用。因此，KGBB 框架必须通过为每个陈述 KGBB 默认提供一个**基于 OWL 的访问模板**，以实现**与 OWL 和描述逻辑的互操作性** 。指定新的陈述 KGBB 时，KGBB 编辑器会利用为新指定的陈述单元类以及新建陈述 KGBB 的存储模型所提供的信息，自动为该陈述 KGBB 指定一个基于 OWL 的访问模板。

以目的地位置为必需宾语、所有其他客体为可选客体的旅行陈述单元为例，我们可以为将旅行陈述单元转换为基于 OWL 的陈述定义以下属性，每个属性都被标注为 *travelingStatementUnit*类的成员属性（图13）：

[10] 对于新的格式，这可能需要为 KGBB 引擎的应用程序接口指定新的适配器，并对 KGBB 编辑器进行调整。

1. 对象属性 *travelsTo*，作为 *requiredObjectPosition* 的子属性，定义域和范围分别指定为‘PERSON’和‘LOCATION’；2. 对象属性 *travelsFrom*，作为 *optionalObjectPosition* 的子属性，定义域和范围分别指定为‘PERSON’和‘LOCATION’；3. 对象属性 *travelsWith*，作为 *optionalObjectPosition* 的子属性，定义域和范围分别指定为‘PERSON’和‘TRANSPORTATION’；以及 4. 数据属性 *travelsOn*，作为 *optionalObjectPosition* 的子属性，定义域指定为‘PERSON’，范围指定为‘datatype:DATETIME’。

属性的标签可以从文本显示模板（见下文讨论）中自动获取，定义域和值域规范可以从存储模型的输入限制中获取，而传递性等逻辑属性公理则可以从对象位置类的相应规范中获取（参见 KGBB 编辑器）。这种方法可以实现从任何陈述 KGBB 的存储模式到基于 OWL 的访问模板的通用转换。

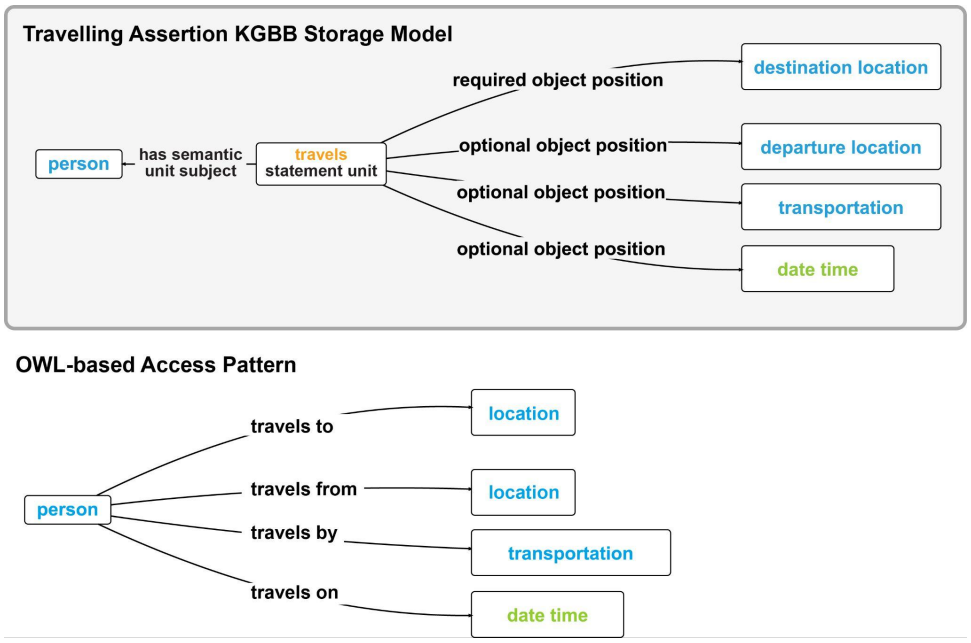

图 13：比较的存储模型陈述 KGBB 及其基于 OWL 的访问模型。 **Top:** The主体相关存储模型遍历陈述单位，由其陈述类。 **底部：** The变换into基于 OWL 的访问模型，其中the限制on the主语and the各种宾语KGBB 的位置存储模型are已翻译into基于 OWL 的模型中的定义域和值域规范。在此，还可以根据需要为每个属性指定传递性等逻辑公理。

数据和元数据的导入与导出

导入模板可以采用多种格式进行定义，方式与将主语、不同的宾语位置及其关联的语义角色映射到各种访问模型和格式的方式相同。这使得可以系统地导入不同格式的数据。导入前仍需对数据进行格式化，使其符合相应导入模板的格式。

预训练大语言模型和其他自然语言处理工具可以通过实体识别以及将值映射到本体术语来支持导入。ChatGPT 等工具可用于根据具有一组指定主语位置和宾语位置的存储模型，对自然语言陈述进行语义化；每个位置都以主题标签为特征。为每个位置指定适当的本体后，该工具可以根据自然语言陈述中的信息，识别与各位置匹配的值。结果可以返回给用户

以使用动态标签进行验证。如有必要，用户可以手动调整结果，然后确认。

通过将陈述型 KGBB 的存储模型用作一种**通用语言**，KGBB 框架支持对**操作数据和元数据模式交叉映射**进行定义。如果数据和元数据存储在使用不同模式和格式的不同数据库中，而你希望将其再次导出为其他不同的模式和格式，那么你只需指定导入到相应陈述类型的 KGBB 存储模型中的导入规则，以及用于导出的相应访问模板。因此，KGBB 存储模型就像所有不同模式和格式之间的中介：只需定义与中介之间的双向映射，而无需定义所有可能的模式和格式组合之间的映射，从而大幅降低映射负担。

显示模板—人类可读的数据视图

使用陈述 KGBB 类提供的存储模型结构，KGBB 引擎会根据给定陈述单元的数据图生成数据结构。为了能够以人类可操作的方式显示语义单元的内容，前端需要某个**显示模板**，该模板规定如何将数据结构转换为人类可读陈述。显示模板会组织数据结构中的信息，并结合额外的标签和标题，以便在用户界面中呈现（另请参见下文的复合显示模板）。

例如，如果将带有 95% 置信区间的重量测量值存储在相应的陈述单元中，那么相应的存储模型将指定三个字面量对象位置和一个资源对象位置，即一个主值、一个上界值、一个下界值和一个单位。基于表单的显示模板可以指定，这些信息应在前端显示为“重量（95% 置信区间）：[字面对象_1:字面量]([字面对象_2:字面量]-[字面对象_3:字面量]) [资源对象_4:资源标签]”。另一个例子是旅行陈述，其相应存储模型指定一个必需的资源对象位置（即目的地）和三个可选对象位置（即出发地、交通工具和日期时间）。相应的显示模板会在前端将旅行陈述呈现为“人物乘坐 交通工具 从出发地_位置 前往 目的地_位置 ，日期为日期时间”。换句话说，主语位置和不同的宾语位置（即带有关联语义角色的句法位置）会映射到字符串中的相应变量，从而形成一条人类可读陈述（见图 14）。我们将这类文本显示模板称为**动态标签 [E3.;E3.1]**。

除了此类文本显示模板外，还可以为**陈述单元的思维导图式表示**指定**动态思维导图模式[E3.;E3.1]**。动态思维导图模式使用一个标签来表示相应陈述类型所依据的谓词；如果存在多个对象位置，还会使用标签来表示各个客体与谓词之间的关系。因此，该陈述可以可视化为思维导图式图，其中每个主语和宾语都表示为一个节点，其标签取自陈述单元底层数据图中对应资源的标签（见图 14）。此类陈述单元的图形表示还可以组合起来，形成更大语境和相互关系的思维导图，从而连接多个陈述单元的动态思维导图模式，例如以图形方式显示复合单元。与基于表单的表示相比，以思维导图式表示复杂实体间相互关系更易于理解，因此能提高知识图谱的人类可操作性。

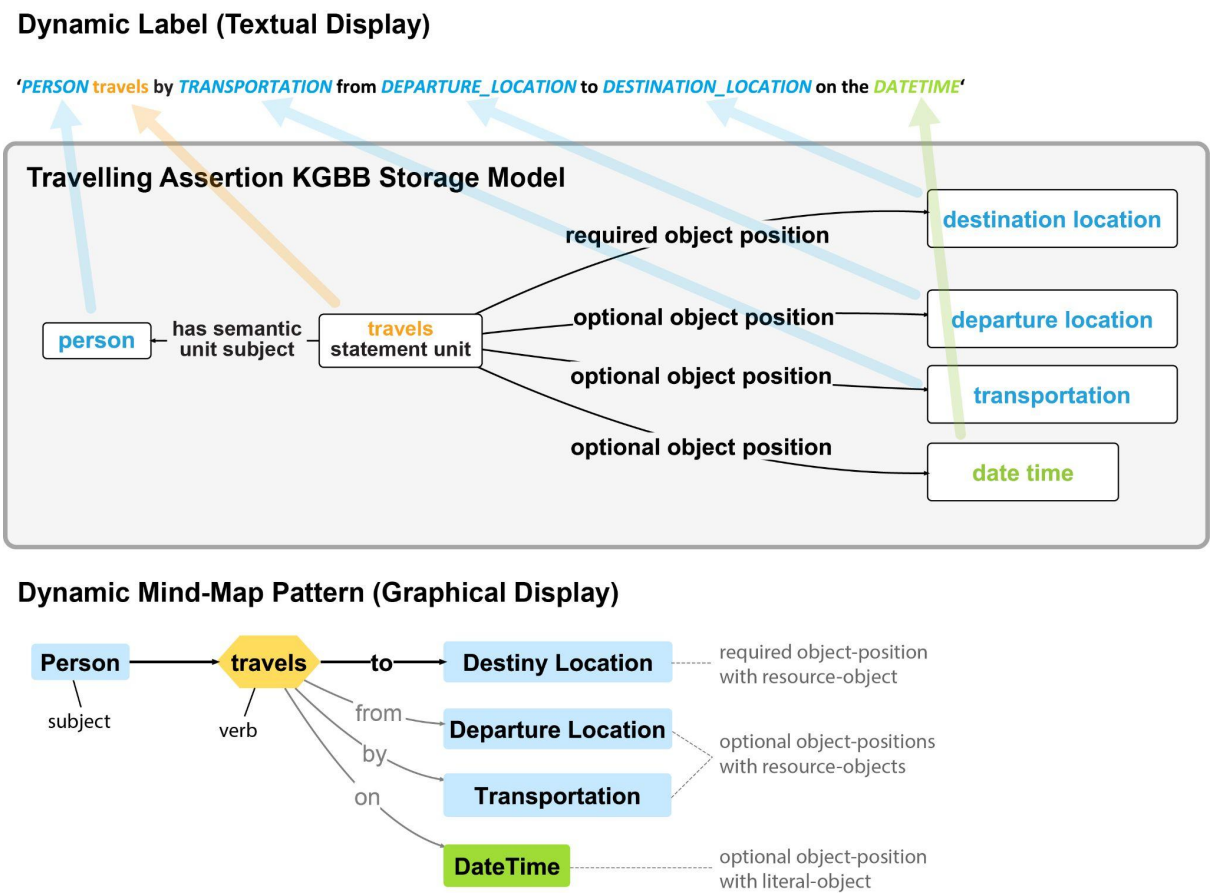

图14：基于其存储模型的陈述单元的文本和图形显示。中部：由其陈述 KGBB 类提供的旅行陈述单元的主体相关存储模型。**顶部：**基于其陈述 KGBB 提供的动态标签，对旅行陈述单元进行文本显示。请注意，存储模型中的主语位置和宾语位置如何映射到动态标签模板中的变量位置（即具有关联语义角色的位置）（箭头）。**底部：**基于其陈述 KGBB 提供的动态思维导图模式，对旅行陈述单元进行图形显示。存储模型中的主语位置和宾语位置到动态思维导图模式中节点的映射，与到动态标签模板的映射相似（为使展示清晰，此处未显示箭头）。

动态标签模板还可以体现该陈述单元属于**词汇性**、**断言性**、**偶然性**、**原型性**还是**全称陈述单元**。对于具有部分陈述，断言性动态标签模板可以采用"此主语 具有此部分"的形式，偶然性模板可以采用"某个主语 可能具有某些部分"的形式，原型性模板可以采用"某个主语 通常具有某些部分"的形式，而全称性模板则可以采用"每个主语必然具有某些部分"的形式。

陈述单元的动态标签也可以在图中向用户展示以供其探索图时，用作其陈述单元资源的标签 **[E3.2]**。这样，用户无需访问图中通常不太容易被人类读者理解的实际三元组层级，便可以在表征粒度的陈述单元层级访问陈述单元中包含的信息。

复合 KGBB

复合 KGBB 的工作方式不同于陈述型 KGBB，因为它们组织和管理来自陈述单元及其他复合单元的数据，但不必为其数据图提供存储模型，因为它们的数据图是通过合并其关联语义单元的数据图而获取的。因此，其存储模型仅涵盖语义单元图（参见图 8），因而也只需规定与复合单元资源本身相关的信息。

通用语义单元图存储模型的复合单元专用扩展

由于复合单元关联着两个或更多语义单元，并且两个复合单元可以通过一个链接陈述单元彼此链接，该链接陈述单元的主语资源与其中一个复合单元的主语资源相同，其宾语资源则与另一个复合单元的主语资源相同，因此，从 *KGBB* 继承的通用语义单元图存储模型需要扩展，纳入以下额外槽位（值域）：

11.**具有相关语义单元** (语义单元的 *UPRI*)，可选：如果该语义单元还关联着另一个语义单元，则属性 *具有相关语义单元* 指定该语义单元的 UPRI。只有复合单元会关联其他语义单元。

12. **具有链接语义单元** (语义单元的 *UPRI*)，可选：如果语义单元是复合单元 *A* ，并且它通过一个陈述单元与复合单元 *B* 相连，其中复合单元 *A* 与陈述单元共享同一个主语资源，且陈述单元的宾语资源是链接的复合单元 *B* 的主语资源，那么复合单元 *A* 到复合单元 *B* 的这一链接通过属性 *具有链接语义单元* 明确表示。

许多复合 KGBB 类还会对其主语施加更多约束。例如，物质实体项 KGBB 类会约束其主语，使其成为相应本体类的实例，例如“物质实体”（BFO:0000040）。理想情况下，这类约束应能在 LinkML 模板中指定。

除了与每个复合单元直接存储的元数据外，KGBB 引擎还会动态获取该复合单元的贡献者、最后更新日期和导入来源溯源元数据，以及其所有相关语义单元中的版权许可、访问限制和逻辑框架规范，并将这些信息提供给表示层。

复合 KGBB 定义了知识图谱的命题空间

复合 KGBB 不仅有助于将知识图谱组织成各种类型的复合单元，从而将语义单元集合组织成更大的子图，并为这些子图分配各自的 UPRI 以供引用；它们还通过将陈述 KGBB 和其他复合 KGBB 与数据输入 / 导入相连接，限制特定语境中可实例化的陈述单元，从而对数据输入 / 导入进行组织。例如，过程项 KGBB 允许使用发生体部分关系陈述 KGBB，而物质实体项 KGBB 则不允许，因为发生体部分关系仅适用于过程等发生体之间，而不适用于物质实体等延续体之间。

通过链接到其他 KGBB，**一组复合 KGBB 指定了知识图谱的论域——它定义了其命题空间**，并将其限制为由陈述和

有意义且属于相应知识图谱范围内的语境。复合 KGBB 与其他 KGBB 相关联，因此通过复合 KGBB 新增的数据可以触发其他 KGBB 实例化相应的语义单元。资源可以在不同 KGBB 之间共享，以便在新添加的语义单元中重复使用；这些新添加的单位则通过共享资源连接到整体数据图。

通过以这种方式将不同的 KGBB 彼此关联，可以通过指定特定情境中要使用的相关 KGBB，以高度模块化且可扩展的方式定义知识图谱的结构。其中一些关系是在 KGBB 类层面上指定的，适用于该类的每个 KGBB 实例；但在使用 KGBB 编辑器通过 KGBB 类的实例描述特定知识图谱应用时，还可以添加更多关系。此外，由于每个以这种方式与复合 KGBB 相关联的陈述 KGBB 都有自己的存储模型，因此所有相同类型的命题仍将使用相同的存储模型，并且相应知识图谱中的所有数据和元数据都将真正符合 FAIR **[F6.1/I7.1]**。

换言之，使用项目组知识图谱构建模块的一个实例，可以通过 KGBB 编辑器将相应的项目知识图谱构建块实例链接起来，从而指定可与其关联的项目单元类型。不同 KGBB 之间这些可能的交互关系在**KGBB 规范图**中进行了描述，该图为相应的 FAIREr 知识图谱应用提供规范。在此规范图中，可以为每个项目知识图谱构建块实例指定其与陈述知识图谱构建块实例之间可能的交互关系，从而定义可与此类项目单元关联的陈述单元类型。规范图还会指定哪些类型的陈述单元是必需的，因此在能够将项目单元添加到知识图谱以及首先添加到此类项目组单元之前，必须提供这些单元。规范图还会指定项目单元是否只能包含一个或多个此类陈述单元（关于如何指定由 KGBB 驱动的知识图谱应用中不同KGBB 实例之间的交互，以定义不同 KGBB 之间的关系，请参阅相关内容）。

复合显示模板

复合显示模板是用于复合KGBB类的显示模板，用于确定其复合单元实例中的数据应如何在用户界面中以文本信息的形式呈现**[E3.;E3.1]**。通常，每个复合KGBB都可以在其类描述中指定一个或多个此类复合显示模板。复合显示模板决定其关联语义单元中的数据的呈现顺序，包括占位文本、工具提示、标题、标签等（图15）。每个复合KGBB类都可以使用 JavaScript 模板引擎或 Web 模板引擎 Jinja2指定一个复合显示模板。可以为不同语境定义不同的模板，从而使每个使用相应KGBB的知识图谱应用程序都能够定义自己的用户界面，并按照自身的用户界面外观和风格呈现属于该语义单元的数据。此外，这还支持针对不同应用程序或不同类型用户使用不同的用户界面。

换言之，用于复合KGBB的复合显示模板始终是基于表单的模板。复合单元不需要思维导图式显示模板——它们可以通过合并其关联陈述单元各自的思维导图式表示，在用户界面中以思维导图式表示呈现。

每个新定义的 KGBB 类都应至少配备一个可直接使用的基于表单的显示模板，并且在需要时可以对其进行更改和调整。因此，复合单元的基于表单的显示模板会嵌入其关联陈述

和复合单元的基于表单的显示模板。对于陈述型 KGBB，基于表单的模板还应辅以类似思维导图的
显示模板。

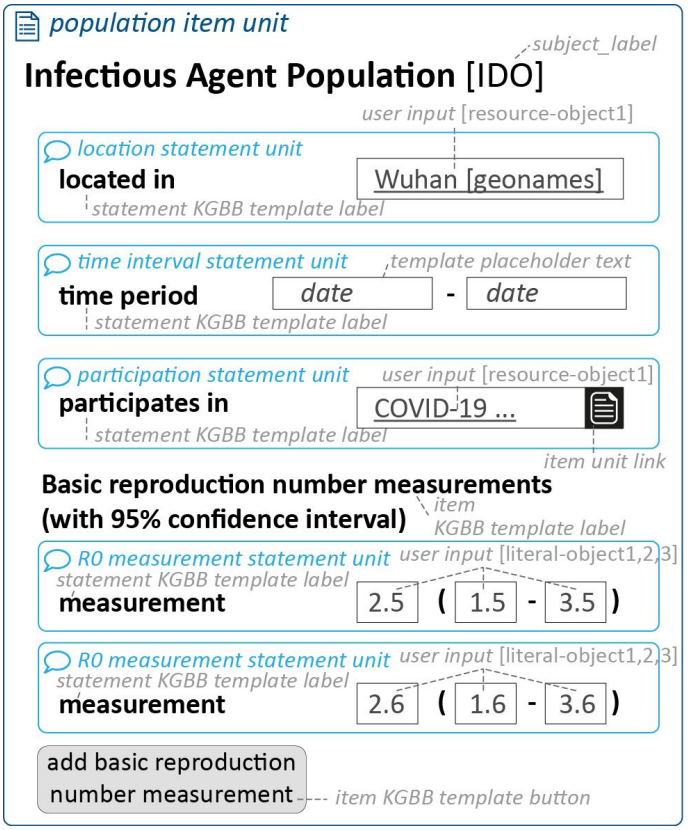

图15：用于编辑人口项目单元的用户界面页面示例。该页面利用相应项目单元知识图谱构件类的复合显示模板，并根据各自的动态显示模板嵌入其关联陈述单元的表示。此页面的整体结构以及显示的标签均由相应的复合显示模板定义。页面展示了特定时间段内的武汉人口数据，这些数据由一个物质实体条目单元及其关联陈述单元的数据图提供。此页面的整体结构以及显示的标签均由相应的显示模板定义，其中物质实体项 KGBB 的复合显示模板嵌入其关联陈述 KGBB 的动态显示模板。图中深蓝色边框框内显示项目单元的表示，浅蓝色边框框内显示其关联陈述单元内容的表示。浅灰色标签标明了不同框中各类信息的来源。

问题知识图谱构件与 KGBB 查询构建器

可以通过使用现有陈述 KGBB 的存储模型，并允许在相应的主语和宾语位置使用命名个体、某实例、每个实例和类资源，然后将其分类为问题单元，来创建新的问题单元。根据所使用的存储模型，KGBB 引擎可以将该问题转换为查询：如果主语和宾语位置包含完全指定的输入（即命名个体资源），则查询将检索布尔值真/假 答案；如果一个或多个此类位置包含未充分指定的输入（即某实例、每个实例或类资源），则查询将检索与问题单元所提供的陈述单元描述相匹配的陈述单元列表作为答案。KGBB 引擎根据相应陈述 KGBB 的动态标签模板，为该问题单元推导出一个显示模板。因此，如果陈述 KGBB 的动态标签模板采用“人物 乘坐交通工具 从 出发地_位置 前往 目的地_位置，时间为 日期时间”的形式，则问题单元的显示模板将采用“人物 是否乘坐 交通工具 从出发地_位置 前往 目的地_位置，时间为 日期时间？”的形式，并且相应的问题单元可以在表征粒度的陈述单元层级上，通过相应的动态标签作为节点，或通过动态思维导图作为图模式，表示于任意图可视化中。

同一理念为 KGBB 前端的查询界面奠定了基础：用户可以使用输入表单创建自己的查询，通过命名个体创建陈述单元，并将其作为输入，

还可以使用某实例、每个实例、类资源和数据类型作为主语和宾语位置的输入，无需使用任何图查询语言 **[E3.4]**。KGBB 查询构建器会将问题单元转换为实际的图查询。在此，陈述型 KGBB 存储模型的通用结构为开发这一转换步骤的流程提供了有力支持。

因此，问题单元不仅可以记录图中的问题，还可以间接记录相应的查询。这样一来，FAIREr 知识图谱的能力问题便可以以一种可直接执行的方式记录在知识图谱自身中。

KGBB 查询构建器可以使用主语位置和各种宾语位置作为变量，按照通用查询模式转换一个**陈述问题单元**，并使用布尔运算符 AND 和 OR 将不同的陈述问题单元组合成一个**复合问题单元**。

陈述问题单元的存储方式与陈述单元相同：应用其对应陈述 KGBB 的存储模板，但将其归类为 *陈述问题单元* 的实例，并使用属性 *基于陈述 *KGBB** 将其与对应的陈述 KGBB 关联。复合问题单元的存储方式与复合单元类似：指明其关联的陈述问题单元，并在其数据图中使用布尔运算符将这些问题单元关联起来。

通过指定 KGBB 实例之间的交互来定义 KGBB 驱动的知识图谱应用

通过跨语义单元共享资源来连接数据图

属于某个陈述单元数据图的所有三元组构成一个连通图，但如何以语义上有意义的方式连接多个陈述单元中的这些数据图呢？这可以通过指定相应 KGBB 之间可能的交互来实现。每个 KGBB 都是一个类。这些类的实例可以使用 KGBB 编辑器来描述其可能的交互。两个 KGBB 实例之间的每种交互都会使相应的两个语义单元实例数据图彼此连接。可以区分出五种连接类型：

1. **主语—主语连接**：陈述单元 A 和 B 可以通过在各自的主语位置共享同一资源而连接（图 16A ）。 2. **主语 — 宾语连接**：陈述单元 A 和 B 可以通过在陈述单元 A 的一个宾语位置与陈述单元 B 的主语位置共享同一资源而连接（图 16B）。 3. **主语—语义单元连接**：陈述单元和语义单元可以通过将语义单元的资源置于陈述单元的主语位置而连接（图 16C）。 4. **宾语—语义单元连接**：陈述单元和语义单元可以通过将语义单元的资源置于陈述单元的一个宾语位置而连接。 5. **宾语—宾语连接**：陈述单元 A 和 B 可以通过在各自的一个宾语位置共享同一资源而连接。

此连接类型列表已穷尽所有可能，不存在其他类型的连接。后两种连接类型取决于能否在对象位置复用现有资源，而这又取决于相应陈述型 KGBB 的数据图存储模型的输入约束；前三种则必须定义为特定 KGBB 之间可能发生的交互。

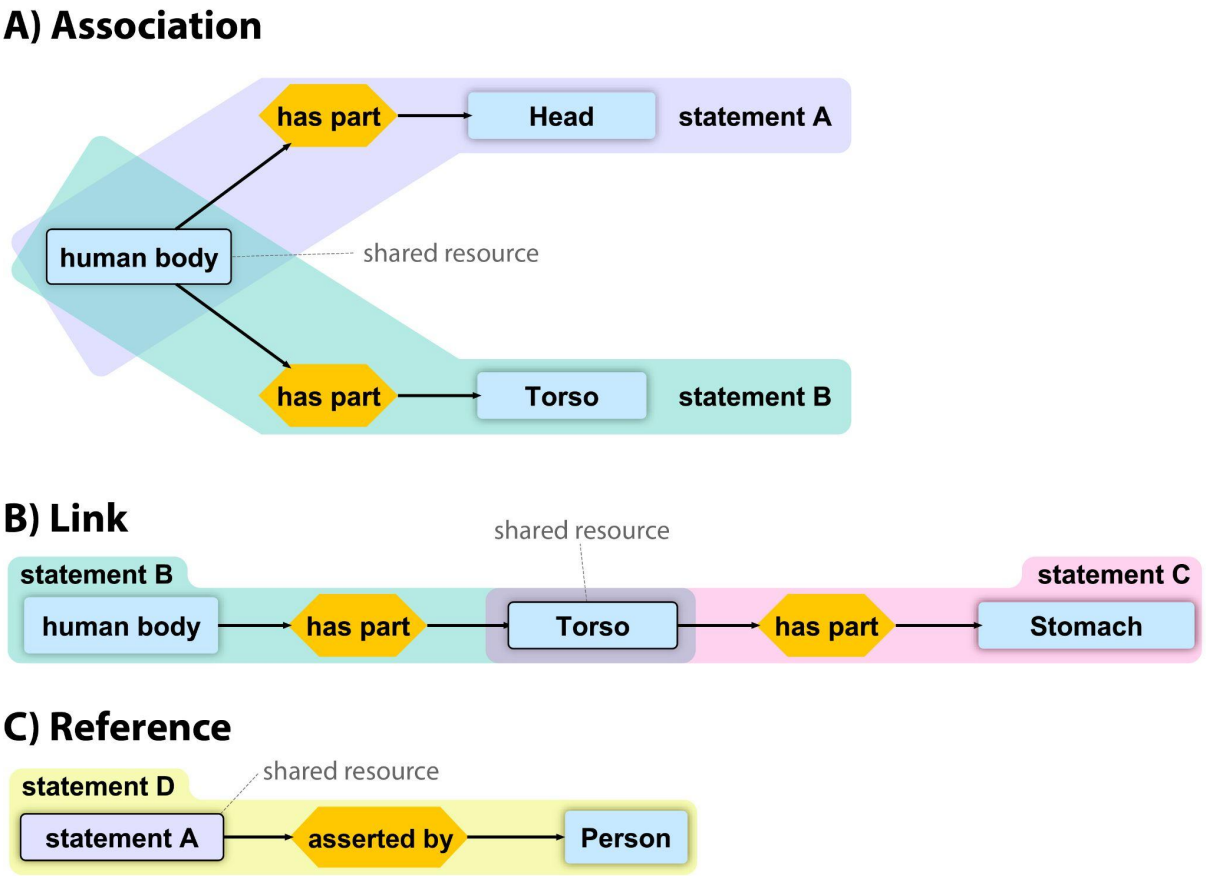

图 16：通过指定 KGBB 之间可能的交互，在图中连接陈述单元的三种方式。A）通过关联。两个或多个陈述单元可以共享同一资源，分别将其作为主语。这种连接通过关联节点来描述。**B）通过链接。**某个资源可以是一个陈述单元的宾语，同时也是另一个陈述单元的主语，从而连接这两个陈述。这种连接通过链接节点来描述。**C）通过引用。**一个语义单元的资源可以是另一个陈述单元的主语资源，从而将二者连接起来。这种连接通过引用节点来描述。

KGBB 驱动的更符合 FAIR 原则的知识图谱应用可以通过特定类型的**交互节点** 将 KGBB 类的实例彼此关联起来加以定义，这些交互节点用于界定 KGBB 之间可能的交互。产生主语—主语连接的交互通过**KGBB 关联节点**指定；产生主语—宾语连接的交互则通过**KGBB 链接节点**指定；产生主语—语义单元连接的交互则通过**KGBB 引用节点**指定（图 16）。

通过关联 KGBB 实例来指定知识图谱应用的示例

有些命题在逻辑上依赖于其他命题，例如，重量测量依赖于具有“重量”（PATO:0000128）这一品质的“物质实体”（BFO:0000040）的存在。因此，定量重量测量命题依赖于相应的

定性-重量-质量命题，并且要求先对后者进行规定。在由KGBB驱动的FAIREr知识图谱应用中，可以使用一个质量陈述 KGBB 实例和一个重量测量陈述 KGBB 实例来对这种依赖关系建模，并通过链接与关联节点的组合，将二者分别关联到一个测量复合 KGBB 实例；该实例除其他测量外，也对重量测量进行建模。

一个**关联节点** 实例将测量复合 KGBB 的一个实例（作为源 KGBB）与质量陈述 KGBB 的一个实例（作为相应的目标 KGBB）关联起来。因此，所有测量复合单元都将与相应质量陈述单元的主语资源共享主语资源，而后者的动态标签形式为“宾语 具有品质”。随后，通过在质量陈述 KGBB 的一个实例（作为链接的源 KGBB）与重量测量 KGBB 的一个实例（作为其目标 KGBB）之间设置**链接节点** ，即可指定将前者品质位置的对象资源重新用作后者的主语资源，从而将质量陈述单元与重量测量陈述单元连接起来。

当质量陈述 KGBB 实例化一个以“重量”（PATO:0000128）为资源对象的质量陈述单元时，KGBB 引擎会触发测量复合 KGBB，使其变为活动状态，并提供相应的输入表单，供用户输入所需信息，以根据通过链接节点和重量测量 KGBB 提供的信息创建相应的重量测量陈述单元；该测量陈述单元的动态标签形式为“主语*[重量]* 的值单位”。随后，重量测量 KGBB 将质量陈述单元的资源对象作为其主语，并与重量测量陈述单元的节点共同构建数据图。

如果必须详细描述作为重量测量陈述依据的特定对象的质量测量过程，以便将其作为测量复合单元的相关元数据提供，就需要指定相应的过程复合 KGBB。此 KGBB 可用于引用重量测量过程中遵循的测量规程，也可记录使用了哪些设备、设备的设置如何，以及由谁在何时何地进行了测量。过程复合 KGBB 将涉及多个不同陈述 KGBB 实例之间的交互，并通过相应的关联节点逐一指定。现在，为了表明过程复合单元为测量复合单元提供元数据，这两个复合单元资源必须属于相应的陈述单元，其中一个位于主语位置，另一个位于宾语位置。虽然在对象位置使用复合单元资源的可能性已在相应陈述 KGBB 的存储模型中指定为输入约束，但在陈述单元的主语位置使用复合资源则必须通过**引用节点**来指定。

因此，我们区分三种不同类型的 KGBB节点，用于指定语义单元的数据图之间可能出现的三种连接类型，这些连接可定义为 KGBB 之间的交互：关联节点、链接节点和引用节点；另外两种可能的连接类型（即，宾语-语义单元 和宾语-宾语）则通过相应数据图存储模型中的输入约束来指定。

这三种KGBB节点与KGBB共同构成了指定KGBB驱动的FAIREr知识图谱应用中KGBB之间所有可能交互所需的全部要素。因此，KGBB驱动的知识图谱应用可以完整地规范为一个KGBB规范图，该图通过一组KGBB节点实例将KGBB实例彼此关联。下面将更详细地介绍这三种KGBB节点。

KGBB关联节点

通过*“KGBB 关联”*节点将两个KGBB实例彼此关联，可以指定给定类型的哪些目标语义单元以及多少个目标语义单元能够与源语义单元关联，并规定应遵循哪些约束。所有以这种方式关联的目标语义单元，都通过属性*具有相关语义单元*与源语义单元关联。源 KGBB始终是复合KGBB，而目标 KGBB可以是陈述KGBB或复合KGBB。此外，使用*“KGBB 关联”*节点将源KGBB与目标 KGBB关联，还隐式指定了**源语义单元的主语资源同时也是目标语义单元的主语资源**。*“KGBB关联”*节点的规范包括以下属性（值域）（图17）：

1. **最小计数** (正整数范围，包括*‘0’*)：指定源 KGBB 存在所需的目标 KGBB 语义单元数量。换句话说，如果关联节点设为大于‘0’的数字，则必须与源语义单元一同创建指定数量的目标语义单元。具体做法是通过属性 *SH:minCount* 将 *’KGBB 关联’* 节点与一个正整数值关联起来，该属性对应于 SHACL 的 SH:minCount。值为‘0’表示不存在此类要求。

2. **最大计数**（正整数范围，包括*‘0’*）：指定目标 KGBB 可以与源 KGBB 的单个语义单元建立关联的语义单元数量。具体而言，通过将 *’KGBB 关联’* 节点与一个正整数范围通过属性 *最大计数* 进行关联来指定，该属性对应于 SHACL 的 SH:maxCount。值为‘0’表示不受数量限制[11]。

3. **carryOverSubjectRangeConstraintTo** (目标 *KGBB* 的对象位置类的 *UPRI*)，可选：表示源 KGBB 中为源语义单元的主语指定的约束，必须应用于此处指定的目标 KGBB 的对象位置。相关内容记录在 *’KGBB 关联’* 节点中，方法是通过 *carryOverSubjectRangeConstraintTo* 属性将其与相应对象位置类的 UPRI 关联起来。一个关联节点可以包含多个此类约束传递规范。实例化关联节点时，将使用约束节点跟踪该约束，并将其链接到目标 KGBB 的语义单元资源（参见陈述单元特定扩展通用语义单元图存储模型）。

复合 KGBB 可以通过 *’KGBB 关联’* 节点与各种其他复合 KGBB 和陈述 KGBB 关联；每个相应节点都会指明是否需要目标语义单元（以及需要多少个），以及此类型的语义单元是否只能有一个或可以有多个与该复合单元关联。例如，用户驱动的知识图谱应用可能会为用户设置个人资料项单元，且每个个人资料项单元都要求有一个用户电子邮件陈述单元。只有当用户提供其电子邮件地址后，才会创建相应的个人资料项。

在上文的重量测量示例中，测量复合 KGBB 通过一个*’KGBB 关联’*节点与质量陈述 KGBB 相关联。该节点进一步指定为 *最小计数* ‘1’，表示一个测量复合 KGBB 必须至少关联一个质量陈述单元。该节点还进一步指定为 *最大计数* ‘1’，表示

[11] 管理用于建模谓词的陈述单元的目标 KGBB，如果谓词具有**函数式逻辑属性**，则其最大计数只能为‘1’。

测量复合单元只能关联一个质量陈述单元，不能多也不能少。

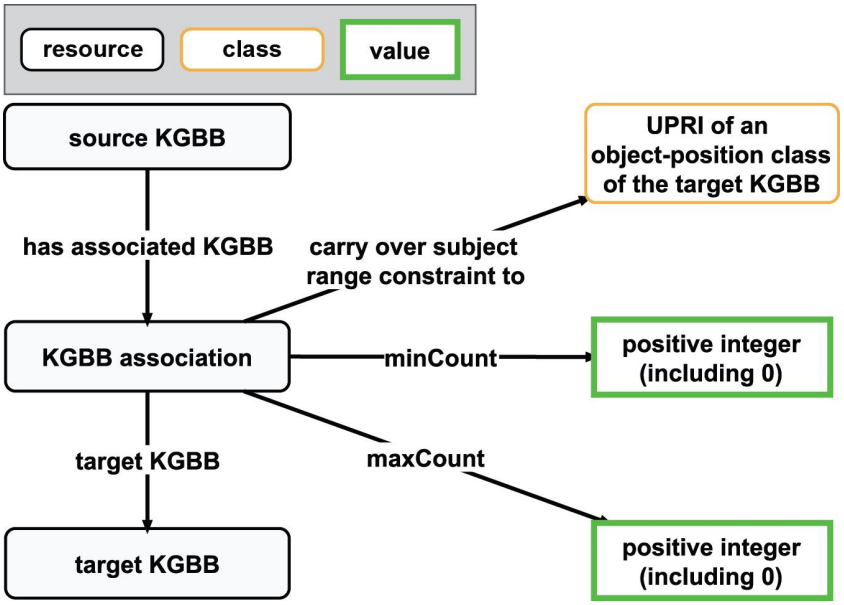

图17：KGBB 关联节点的规范。 KGBB关联节点通过属性 **hasAssociatedKGBB** 和 **targetKGBB** 将源 KGBB 链接到目标 KGBB。源是一个复合 KGBB，而目标原则上可以是任意类型的 KGBB。节点本身通过属性 **minCount** 指定目标 KGBB 所管理的语义单元类型所需的实例数量。因此，用户必须指定该数量，才能将源的复合单元添加到知识图谱中；“0”表示没有此类要求。属性 **maxCount** 指定了目标 KGBB 所管理的语义单元类型的实例数量上限，即可添加到源复合单元中的实例数量，其中“0”表示没有数量限制。

可以将目标 KGBB 所管理的语义单元类型的实例添加到源复合单元中，其中“0”表示没有数量限制。属性 **carryOverSubjectRangeConstraintTo** 指定目标存储模型中对象位置的 UPRI；必须将源存储模型中为主语位置指定的相同约束应用于该对象位置。对于每个关联节点，源复合单元的主语资源也是目标语义单元的主语资源。

总而言之，通过关联节点，可以将特定类型的陈述单元与特定类型的复合单元相关联，从而将数据图组织和构建为具有不同表征粒度的各级子图，并由此确定哪些信息属于给定类型的复合单元，因而将在用户界面中一并呈现（另见(14)）。

KGBB链接节点

当给定陈述单元的宾语资源应作为另一个语义单元的主语资源使用时，始终需要一个*“KGBB 链接”*节点。链接KGBB始终是陈述 KGBB，而目标 KGBB 可以是陈述 KGBB，也可以是复合KGBB。KGBB 链接规定：目标语义单元所链接到的链接陈述单元，通过属性 *由语义单元描述的对象* 与该目标语义单元相关联。将链接KGBB与目标 KGBB 相关联的*“KGBB 链接”*节点规范包括 *最小计数* 和 *最大计数* 属性，这些属性也用于规范*“KGBB 关联”*节点。不过，与 *carryOverSubjectRangeConstraintTo* 属性不同，它需要 *useAsSubject* 和 *ifObject* 属性（图 18）：

1. **UseAsSubject** （链接*KGBB* 的宾语位置类的 *UPRI*）：表示目标 KGBB 应将链接KGBB 的指定宾语位置类用作其语义单元的主语。具体做法是通过属性 **useAsSubject**，将 *‘KGBB 链接’* 节点与相应宾语位置类的 UPRI 关联起来。

2. **ifObject** （本体资源的统一资源标识符），可选：指定一个特定的本体资源；通过 **useAsSubject** 指定的对象位置中的资源必须是该资源的实例、某实例或所有实例资源，

或者必须是其某个子类的实例。只有资源满足这一约束时，目标 KGBB 才能创建其语义单元。具体做法是通过属性 *ifObject* 将 *'KGBB link'* 节点关联到相应的本体类 URI。借助此信息，KGBB 引擎可以根据给定源语义单元相应对象位置的资源类型，判断应激活哪个目标 KGBB。因此，同一个源 KGBB 可以链接到多个目标 KGBB，每个链接都由各自的链接节点进行关联。

关联节点规定了可以与给定类型的复合单元相关联的内容，从而限制了该复合单元的命题空间；而链接KGBB的陈述单元可以理解为一个**开放命题**，它使用户能够通过这一开放命题向已有的复合单元添加新的语义单元。换言之，链接节点定义了图如何通过向已有复合单元添加新的复合单元而不断扩展。

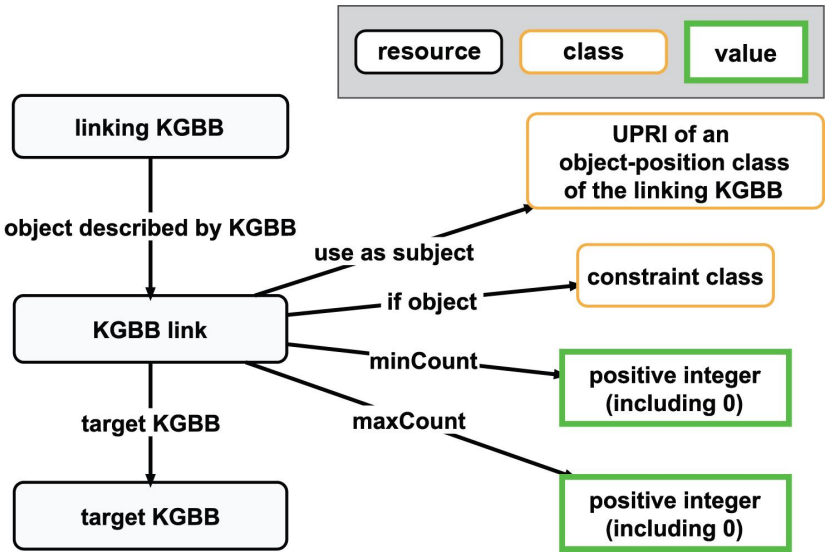

图18：KGBB 链接节点的规范。 KGBB链接节点通过属性 *objectDescribedByKGBB* 和 *targetKGBB* 将链接KGBB与目标KGBB连接起来。链接KGBB始终是一个陈述 KGBB，而目标 KGBB 可以是任何类型的 KGBB。该节点本身通过属性 *minCount* 指定目标 KGBB 所管理的语义单元类型所需的实例数量。因此，用户必须指定这些实例，才能将链接KGBB的语义单元添加到知识图谱中，其中"0"表示没有此类要求。属性 *maxCount* 指定语义单元类型的实例数量上限。

由目标 KGBB 管理的对象可添加到链接语义单元中，其中"0"表示没有数量限制。属性 *useAsSubject* 用于指定链接 KGBB 的存储模型中的一个对象位置（通过对象位置类的 UPRI），表示必须将该类的一个实例重新用作目标语义单元中的主语资源。最后，*ifObject* 用于指定目标语义单元的主语资源必须满足的约束。只有当该资源满足此约束时，目标 KGBB 才允许将其用作主语并创建目标语义单元。

以上述重量测量示例为例，质量陈述 KGBB 通过一个 *'KGBB 链接'* 节点与重量测量陈述 KGBB 相关联。该节点进一步指定为 * 最小计数 * '0'，表示质量陈述单元不一定要链接重量测量陈述单元。通过 *useAsSubject* 和 品质位置的 UPRI，该节点被指定为使用质量陈述单元中品质位置的资源，作为重量测量陈述单元的主语。通过 *ifObject* 和值'重量'（PATO:0000128），该节点还被指定为：只有当 QUALITY 位置的资源为'重量'（PATO:0000128）时，才触发重量测量陈述 KGBB。最后，该节点被指定为 *最大计数* '0'，表示质量陈述单元可以链接多个重量测量陈述单元。

KGBB 引用节点

当某个语义单元的 UPRI 应作为另一个语义单元的主语时，始终需要一个 *'KGBB 引用'* 节点。源 KGBB 可以是任何类型的 KGBB，而目标 KGBB 只能是陈述 KGBB。KGBB 引用节点规定，目标陈述单元所链接到的源语义单元通过属性 **refersToStatementUnit** 与该目标陈述单元相关联。用于将源 KGBB 与目标 KGBB 关联起来的 *'KGBB 引用'* 节点，只需指定 *最小计数* 和 * 最大计数~~* 属性，这两个属性~~也用于指定 *'KGBB 关联~~'* 节点（图 19）~~：

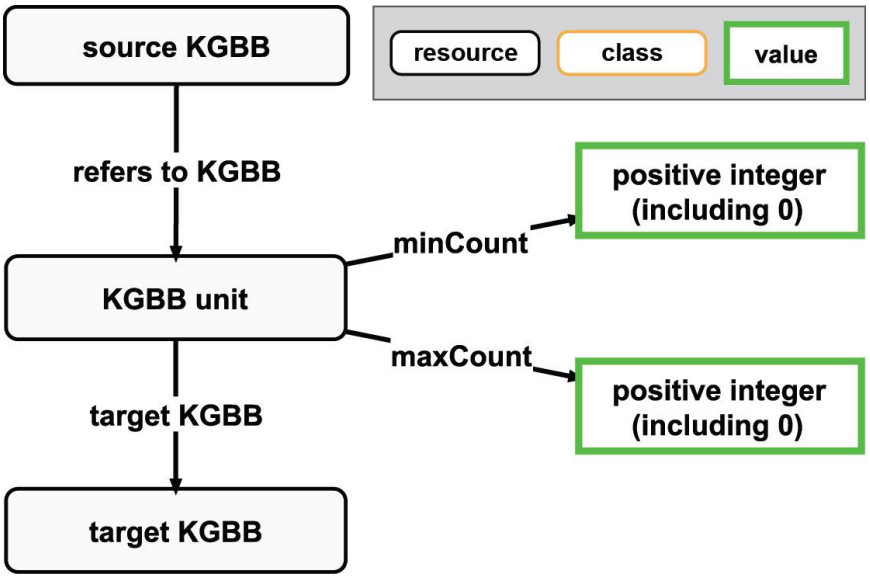

图19：KGBB 引用节点规范。 KGBB 引用节点通过属性 **refersToKGBB* 和 **targetKGBB* 将源 KGBB 链接到目标 KGBB~~。源 KGBB~~ 可以~~是任意类型~~的 KGBB，而目标 KGBB 始终是陈述 KGBB。该节点通过属性 **minCount* 指定源 KGBB 所管理的语义单元类型所需的实例数量，因此用户必须指~~定该数量~~，才能将目标 KGBB 的语义单元添加到知识图谱中，其中"0"表示没有此类要求

。属性 **maxCount* 指定可添加到源语义单元的目标 KGBB 所管理的语义单元类型的实例数量上限，其中"0"表示不限制数量。对于每个引用节点，源语义单元的 UPRI 都是目标陈述单元的主语资源。

关于陈述的陈述需要引用节点。每当需要将某个语义单元作为整体来引用，并在一个陈述单元中对其作出陈述时，都要使用引用节点，将相应的陈述 KGBB 与被引用语义单元的 KGBB 联系起来。例如，某个特定多细胞生物的表型描述由相应的描述项目组单元涵盖，而该项目组单元需要通过一个关于陈述单元，与其所描述的多细胞生物关联起来；该生物在图中由多细胞生物（UBERON:0000468）的一个实例表示。这一操作将由 KGBB 引擎利用相应引用节点中的信息进行管理。项目组知识图谱构建块与关于陈述 KGBB 之间所需的交互~~无法用关一联或链接~~节点来描述，因为二者都不允许将项目组单元的资源用作关于陈述单元的主语。

结合 KGBB 关联和 KGBB链接节点形成 KGBB 关系环

有时，向已有的复合单元添加一个陈述单元，应当自动创建另一个相同类型的复合单元。例如，向生物项目单元添加一个具有部分陈述，表明该生物以头部为其部分时，应当自动添加相应的头部项目单元，且两个项目单元的类型均为物质实体条目单元。这可以在构成

环的 KGBB 关系链层面进行规定：该关系链从物质实体项 KGBB 开始，该 KGBB 指定了一个 *‘KGBB 关联’* 节点，其目标具有部分陈述 KGBB 通过一个 *‘KGBB 链接’* 节点链接回同一个物质实体项 KGBB（图 20）。目标具有部分陈述 KGBB 是一个链接KGBB，其实例化会自动实例化相关的源物质实体项KGBB。这会产生具有相关联的具有部分陈述单元的物质实体条目单元，而其部分对象则由另一个物质实体条目单元进一步描述。这两个物质实体条目单元之间的语义关系通过属性 *具有链接语义单元* 记录在图中，将前一个项目单元连接到后一个项目单元。

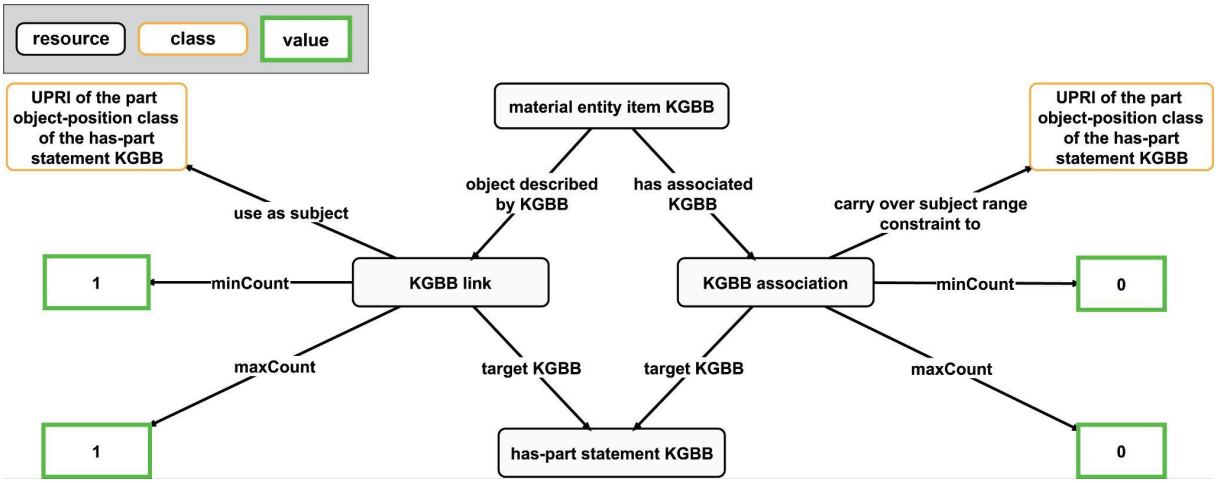

图 20：KGBB 关系循环的规范。物质实体项 KGBB 通过结合关联节点和链接节点的具有部分陈述 KGBB 连接回自身。关联节点规定，适用于物质实体条目单元的主语资源的约束，也必须应用于关联的具有部分声明单元中任何新增的部分（*carryOverSubjectRangeConstraintTo* *UPRI*）；创建物质实体条目单元时无需创建关联的具有部分陈述单元（*最小计数* 0）；并且可以将多~~个具有部分陈述单元关联到物质~~实体条目单元（*最大计数* 0）。关联节点的目标是一个具有部分关系陈述 KGBB，该 KGBB 又通过链接节点连接回物质~~实体项 K~~GBB。链接节点规定，具有部分陈述单元的部分对象位置中的宾~~语资源必~~须用作物质实体条目单元的主语（*useAsSubject* *UPRI*）；创建具有部分陈述单元时必须创建所链接的物质实体条目单元（*最小计数* 1；例如，这将导致在创建具有部分陈述“生物体 *X* 具有部分头部 *Y*”时自动创建一个“头部 *Y* 项目单元 ”）；并且只能从该具有部分陈述单元链接一个物质实体条目单元（*最大计数* 1）。由于具有~~部分关系陈~~述 KGBB 仅链接到一个 KGBB，因此该链接节点未指定 *ifObject* 属性。

因此，在向具有部分关系陈述 KGBB 提供输入时，系统会根据该输入自动实例化一个新的物质实体条目单元。换言之，对于描述生物体 *X* 的项目单元，用户指定生物体 *X* 具有部分头部 *Y*时，相应的具有部分关系陈述 KGBB 将实例化具有部分陈述单元，同时也会实例化描述头部 *Y*的项目单元。因此，生物体 *X* 项目单元会通过 *具有链接语义单元* 与头部 *Y* 项目单元建立关联。在这个新创建的项目单元中，用户随后可以指定头部 *Y*的各~~个部分；每个指定的~~部分都会实例化相应的具有部分陈述单元及其链接的物质实体条目单元。这样一来，便可以创建多个相互链接的项目单元，并最终创建一个描述生物体 *X*部分学的项目组单元。

KGBB 关系环可用于知识图谱，使用户能够提供其所需粒度层级的信息：首先创建相互关联的项目单元的部分学，然后向其选择粒度层级的项目单元提供额外信息。

此类环的使用不限于部分关系，还适用于任何偏序关系，例如类-子类包含关系、派生关系或先后关系。

关联节点和链接节点的组合也可通过某个陈述 KGBB，将不同类型的复合 KGBB 彼此连接起来。不过在这种情况下，链接节点的目标 KGBB 并非关联节点的源 KGBB，因此不会形成环路。

KGBB 规范图：指定 KGBB 驱动的更符合 FAIR 原则的知识图谱应用

为 FAIREr 知识图谱应用要涵盖的不同类型语义单元定义好所有 KGBB 后，只需按照上述说明，通过关联节点、链接节点和引用节点将这些 KGBB 的实例彼此关联起来，定义不同 KGBB 之间可能的交互，即可完整指定该应用。规范本身**以 KGBB 规范图的形式呈现，通过定义其数据与命题空间——即其数据和元数据可以填充的空间——全面描述特定的 KGBB 驱动的更符合 FAIR 原则的知识图谱应用**。

借助 KGBB 编辑器，KGBB 框架支持对 KGBB 驱动的知识图谱应用进行规范。KGBB 编辑器将允许领域专家以更符合 FAIR 原则的方式描述自己的知识图谱应用。编辑器将能够访问 KGBB 仓库中的所有 KGBB 类，领域专家可据此创建其 FAIREr 知识图谱所需的 KGBB 实例。同一 KGBB 类的不同实例可通过上述不同交互节点，以多种方式连接到多个 KGBB 的实例，从而**根据特定情境指定它们的具体用法**。例如，同一个具有部分关系陈述 KGBB，既可用于描述物质实体各部分的部分学，也可用于描述方法各步骤的部分学。前一种情况下，可使用一个*‘KGBB 关联’*实例，将一个物质实体项 KGBB 实例与一个具有部分关系陈述 KGBB 实例关联起来，并在关联节点中指定只有物质实体可作为主语；后一种情况下，则可使用另一个*‘KGBB 关联’*实例，将一个方法项 KGBB 实例与另一个具有部分关系陈述 KGBB 实例关联起来，这次在关联节点中指定只有方法可作为主语。

所有复合 KGBB 均使用关联、链接和引用节点进行描述，并可作为复合 KGBB 类存储在仓库中，供其他领域专家在其知识图谱应用中重复使用。每个 KGBB 类都通过属性 描述(DCTERMS:description) 提供其管理的数据类型的可读描述。属性 标签(RDFS:label) 提供可读标签，而属性 *管理* 则将 KGBB 实例关联到其管理的陈述单元类。属性 *存储模型规范属性* 将 KGBB 实例关联到相应的 LinkML 规范。存储模型所引用的各种对象位置类通过属性 *_hasObjectPositionClass_* 与 KGBB 实例关联。附加属性，即 *访问模板属性*、*导入模板属性*、*动态标签模板属性* 和 *思维导图图模板属性*，关联到一个或多个访问、导入和显示模板。

最后，由于 KGBB 类的存储模型中指定的约束可能会引用特定的本体类，以限制允许出现在特定宾语位置或

主语位置中的 UPRI 集合，因此某些 KGBB 只能与特定本体结合使用。属性 *useWithOntology* 指定了这些本体[12]。

属于给定 KGBB 驱动的知识图谱应用的所有 KGBB 实例，都通过属性 *employsKGBB* 和该应用的 UPRI 与该应用相关联。通过属性 *dataEntryStartingPoints* 将应用的 UPRI 链接到特定 KGBB，可以将这些 KGBB 指定为向知识图谱应用添加语义单元的起始点；这对于任何众包知识图谱应用尤其有用。

总之：每个 KGBB 驱动的知识图谱应用都可以描述并指定为一组相互关联的 KGBB 实例，这些实例通过上文提到的节点和属性连接起来。如果将这些内容以 RDF/OWL 的形式提供，并且相应的 KGBB 类也以 RDF/OWL 的形式进行指定，同时在该本体中定义所有属性和 KGBB 特定术语，那么，**KGBB 驱动的更符合 FAIR 原则的知识图谱应用的规范本身就可以以语义透明且真正符合 FAIR 原则的方式表示为 KGBB 规范图，从而形成更具 FAIR 性的知识图谱软件工具**。

KGBB 引擎

我们希望开发一整套用于创建和管理更具FAIR性的知识图谱的工具。这套工具将提供一个**FAIREr 知识图谱应用开发框架**，即**KGBB 框架**，确保每个由 KGBB 驱动的知识图谱创建的语义单元都是一个 FAIR数字对象，其数据和元数据不仅是机器可操作的，而且完全人类可操作，因此便于人类使用，从而满足人类读者的**认知互操作性**要求（14）。KGBB 框架将支持研究人员和领域专家建立自己的更具FAIR性的知识图谱应用。

每个 KGBB 都是一个独立运行的知识图谱处理模块，可以独立于其他 KGBB 进行指定，并作为 OWL文件发布；KGBB 引擎可以读取该文件，也可以通过 KGBB 仓库将其公开为可访问且可复用的资源。这代表了我们实现目标的第一步。模块化架构使研究人员能够复用他人指定的 KGBB。

KGBB 引擎在 KGBB 框架中发挥着核心作用，因此是其关键组件之一。它将知识图谱应用中各个 KGBB 类及 KGBB 规范图所包含的信息付诸实践，将 KGBB 转化为小型知识处理模块，用于管理 FAIREr 知识图谱应用。KGBB 引擎必须提供以下功能（图 21）：

1. 通过其 KGBB API，它会**加载并处理来自 KGBB 规范图的信息，以及相应 KGBB 类规范及其存储、显示、访问和导入模板**，这些模板源自 KGBB 本体。尤其是存储模板，它们以 YAML 文件形式编写，并遵循 LinkML 的表示法，且可转换为 SHACL 形状，有助于向最终用户隐藏底层数据模型的复杂性，同时支持专家用户和开发者与图进行交互。存储模板可用于查询生成和数据可视化，其

[12] 对于具体的知识图谱应用，构建随其内容不断扩展的自有本体，并引用这些本体而非外部本体，可能更为有益。在这种情况下，KGBB 规范必须对此作出说明。

可复用性确保了所有共享相同 KGBB 的 KGBB 驱动知识图谱应用之间的数据一致性。

2. 借助内置的**KGBB 查询构建器**，KGBB 引擎为**KGBB 前端** 提供了易于使用且直观的查询界面，该界面采用从 FAIREr 知识图谱应用中所用各个 KGBB 的存储模型推导出的 CR UD 查询。用户可以合并不同 KGBB 的读取查询，将各自的查询结果取并集或交集，从而创建新的、更复杂的查询。此外，用户可以利用各个 KGBB 提供的输入表单，通过添加查询参数（例如值限制或类资源限制）来缩小并优化单个查询，限定特定类型语义单元中每个宾语位置和主语位置的值域。这些查询甚至可以作为**陈述和复合问题单元** 记录在图中，用户也可以与其他语义单元一同探索这些查询。

3. KGBB 引擎通过其 CRUD API 处理图中所有语义单元上的 **CRUD 操作**。在 KGBB 查询构建器的支持下，它根据 KGBB 存储模型提供的信息推导出 CRUD 查询。该 API 支持在后端使用不同的数据库技术，包括 RDF 元组存储、属性图（例如Neo4j）、其他 NoSQL 数据库（如键值数据库）、Wikibase以及关系数据库。这样一来，**它将持久化层与应用层解耦**，甚至能够将现有知识图谱从一种后端技术迁移到另一种技术，或在联邦知识图谱中使用不同的后端技术。

4. 当新的陈述单元被添加到知识图谱时，KGBB 引擎会根据该单元的主语资源，判断该陈述单元属于断言式、偶然式、原型式还是全称陈述单元，**并对其进行分类**，并在用户界面中显示其内容时使用相应的显示模板。它还会将任何指定了基数限制的某实例或每个实例识别单元归类为*基数限制单元*，并将任何否定语义单元归类为*否定单元*（关于这些语义单元类别的详细讨论，请参见（23））。

5. 每当新的命名个体、某实例、每个实例或类资源因作为新添加到知识图谱中的语义单元的主语或资源对象而被添加到知识图谱时，KGBB 引擎都会自动将相应的识别单元添加到知识图谱，并以该资源为其主语。识别单元包含与特定资源相关的所有信息，例如该资源是命名个体、某实例还是每个实例资源，以及其类归属；对于命名个体，还包括其标签（关于识别单元的讨论，请参见（23））。

6. KGBB 引擎将语义单元的数据图（按照该单元的 KGBB 存储模型进行存储）映射到 KGBB 访问模板和显示模板所指定的模型上，以便访问和展示数据。这可以通过访问 API 以多种方式加以利用：(i) 例如，同一数据可以遵循来自 **访问模板**的不同图模式，在用户界面中以不同方式作为图进行可视化；(ii) 基于 KGBB 提供的基于表单和类似思维导图的 **显示模板**，数据可以与用户界面中的表示层进行传递，其中模板会筛选复杂的数据结构，提取与人类用户相关的信息，并使用 **动态标签和动态思维导图模式**来呈现陈述、复合单元和问题单元节点，从而 **将人类可读的数据展示与机器可操作的数据存储解耦**；

并且 (iii) 数据可以遵循不同的标准，以不同格式导出；这些标准基于每个 KGBB 类中指定的不同访问模板（**模式/元数据交叉walk**）。此外，KGBB 引擎还可以以 JSON、RDF 或 CSV 的形式提供一个或多个特定语义单元的数据图中的数据，例如，R 可以读取 CSV。该引擎还可以将存储模型中的底层数据结构提供为 Java 或 Python 数据类。因此，数据不必按照三元组所提供的低抽象层级进行组织。这样一来，KGBB 应用所管理的数据便会**与应用程序分离，并且可以直接用于其他框架**。

7. 通过其 **访问 API**，KGBB 引擎支持通过 REST API 调用和一组键值参数检索数据，无需发送 SPARQL 或 Cypher 查询。外部应用程序可使用它访问 FAIREr 知识图谱的内容。

8. 通过其 **导入 API**，KGBB 引擎支持导入符合 KGBB 类所提供的某种导入模板结构的遗留数据。
9. KGBB 框架中的所有 API 都将遵循开放 API 倡议，例如 SmartAPI ，该倡议旨在最大限度地提升基于 Web 的 API 的 FAIR性。结合上述对 KGBB 规范图中每个 KGBB 驱动的知识图谱应用的描述，即将其描述为相互连接的 KGBB 类实例，**KGBBFramework 提供了一个 FAIREr 知识图谱框架**。

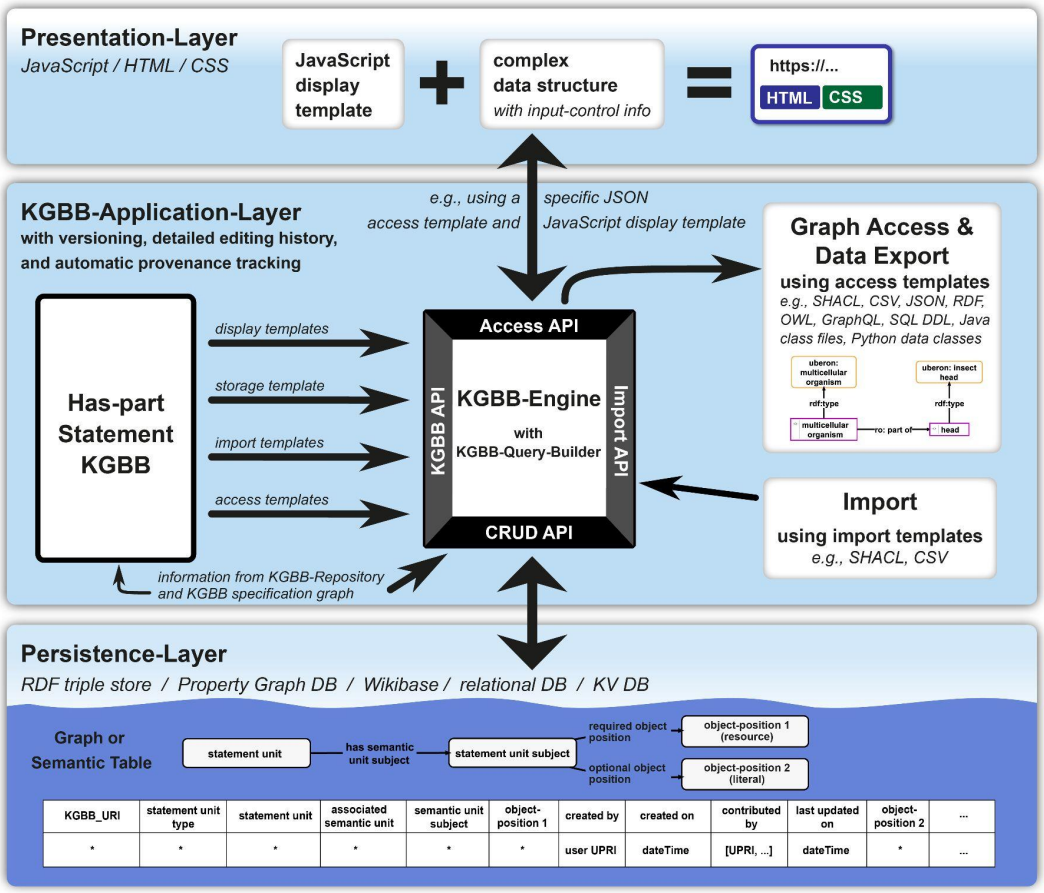

图21：知识图谱构建模块应用引擎（KGBB 引擎）、给定的具有部分陈述知识图谱构建模块、表示层和持久化层之间交互的示意图。

版本管理与编辑历史追踪

能够追踪特定语义单元层面的编辑历史可能非常重要，尤其是在知识图谱快速演化的情况下，或者其内容源于某种协作编辑活动，甚至众包活动的情况下：每个用户都可以编辑图中的任何语义单元，包括其他用户创建的语义单元。如果知识图谱允许用户引用其中一个或多个特定语义单元，从而将知识图谱转变为用于科学传播的宝贵资源，那么它还需要某种版本控制机制，以提供可引用的资源来持续保存被引用的内容。借助这种机制，知识图谱可以通过用户输入持续演化，同时仍然具备可引用性。KGBB 引擎的版本控制机制支持这一点，并遵循两个基本理念：

1）软删除：输入一旦添加到图中，就无法删除或更改。每个编辑步骤都会向相应语义单元的数据图中添加新的三元组，并将提供的信息链接到相应的对象位置实例。换言之，属于特定对象位置类的对象位置实例不会被更新，而是在每个编辑步骤中向图中添加新的实例。为了识别新添加的对象位置实例，会使用布尔属性 *~~currentVersion~~*，并将其设为‘*true*’。例如，如果用户更新了重量测量陈述单元的 VALUE位置，那么迄今为止包含最新添加值、且其属性 *~~currentVersion~~* 的值为‘*true*’的 VALUE位置实例~~将被设为‘*false*~~’，并添加一个新的 VALUE位置实例，其 *~~currentVersion~~* 值为‘*true*’。当用户导航到该陈述单元时，系统会显示该实例中的信息。

以这种方式添加的每个宾语位置实例都会实例化相应的对象位置类，因此相应的 KGBB 可以识别给定语义单元中所有相同宾语位置的实例。此前添加的同一对象位置类的所有宾语位置实例仍保留在相应陈述单元的数据图中，随时都可访问其信息，并可根据需要按创建日期排序，从而为陈述单元的每个宾语位置提供**详细的编辑历史**，也为整个陈述单元乃至与之关联的任何语义单元提供编辑历史。因此，以前文的重量测量示例为例，用户界面可以显示由谁在何时录入了哪个重量测量值。每个宾语位置都包含相应的信息。在管理协作编辑知识图谱内容的工作时，这些信息非常有价值；对于众包构建的更具FAIR性的知识图谱，这些信息也有助于确定谁在何时添加或编辑了什么内容。

相同的方法不仅适用于单个宾语位置，也适用于特定的语义单元。相应的语义单元资源还通过属性 *~~currentVersion~~* 指定其状态。如果用户想删除某个语义单元，则不会实际**删除**相应的单元，甚至可能连同所有与之关联的单元一起删除，而是将该单元及其可能关联的单元的 *~~currentVersion~~* 设为‘*false*’。默认情况下，应用程序不会显~~示 *currentVersion~~* 设为‘*false*’的资源信息；在用户界面中，这些信息看起来像是已被删除，但它们仍保存在知识图谱中，所有相应的溯源元数据也仍~~可访问 [A2.]~~。

2) 用户可以为任何语义单元创建一个版本，该版本此后无法再编辑，因此能够永久保存。 用户可以为特定语义单元创建版本。每个版本都由图中的独立节点表示，并拥有自己的 UPRI，其形式可以是

DOI。因此，这些版本可以在出版物中引用和引证。与其他语义单元一样，版本节点也通过相应的属性记录元数据。此外，每个版本还可以拥有一个 **内容标识符 (CID)** ，它使用密码学哈希来实现**去中心化内容寻址**。

给定语义单元可以存在多个版本。表示最新版本的节点通过属性 *hasCurrentVersion*(PAV:hasCurrentVersion) 与其语义单元节点相连，而所有其他版本节点则通过属性 *hasVersion*(PAV:hasVersion) 与其相连。一个语义单元的版本节点通过一系列 *previousVersion*(PAV:previousVersion) 属性彼此相连，从最新版本开始，依次指向其各自的先前版本。

属于版本化语义单元的所有对象位置实例也会更新，将版本化语义单元的 UPRI 添加为其各自 **versionID** 属性的值。借助此信息，只需查询所有该属性值为此 UPRI 的对象位置实例，即可随时访问相应版本所属的数据。此属性可以多次应用于同一对象位置实例，以追踪多个版本 UPRI。这是必要的，因为给定语义单元的并非所有宾语位置都会在两个版本之间经过编辑，因此某个特定对象位置实例可能属于多个版本。

软删除和版本管理这两个方面，已在一个小型的、基于 Python 的 FAIRscholarly 知识图谱应用原型中得到演示。该应用使用语义单元和 KGBB，并采用 Neo4j 作为持久层技术（62）。原型支持语义单元的版本管理，并自动跟踪其编辑历史和溯源。原型可通过 https://github.com/LarsVogt/Knowledge-Graph-Building-Blocks 获取。

存储和识别陈述单元

如前所述，KGBB 引擎计划支持将语义单元存储在五种不同类型的持久层技术中。在 **RDF** 三元组存储中进行存储时，遵循 (23) 中所述的**纳米出版物** （36–38,63）通用模式；这要求三元组存储支持命名图，并且包含陈述单元数据图的命名图与陈述单元本身具有相同的 UPRI，从而使该资源既代表陈述单元，也代表陈述单元的数据图。

对于 **带标签属性图** （例如 Neo4j）等 NoSQL 数据库，知识图谱中的每个关系都通过相应的 **属性-值对**指定其所属的语义单元；由于一个关系可以关联多个语义单元，因此某些值是不同语义单元的 UPRI 数组。属性 **statementUnitURI** 标识给定关系所属的陈述单元，而 **compoundUnitURI** 标识复合单元，**listUnitURI** 则标识列表单元。属性 **versionID** 和 **datasetUnitID** 指定这些关系所属的版本化语义单元和数据集单元。借助这种表示法，可以通过一条简单直接的 Cypher 查询，识别属于给定语义单元的所有三元组。对于陈述单元，该查询可以如下所示：

'MATCH (n {current_version:"true"}) WHERE ("x" IN n.statementUnitURI) RETURN n',

其中 *x* 是特定陈述单元的 UPRI。这同样适用于其他类别的（版本化）语义单元。

将与语义单元相关联的数据存储在 **关系数据库** 中时，需要根据每个 KGBB 实例总体存储模型的不同组成部分，为其创建多个表格。 **语义头表** 存储来自某个单位语义单元图的数据；复合单元与陈述单元之间的唯一区别在于其中部分属性的存在情况。每新增一个语义单元，就会向此表中添加一行。该表由其对应的语义单元图存储模型所规定的相同槽位和值组成（参见 通用语义单元图存储模型 以及 通用语义单元图存储的陈述单元特定扩展模型）。

为了存储属于陈述单元的数据，需要额外的表来捕获该单元数据图中的数据。 **主体相关数据图表** 用于存储陈述单元主体相关数据图存储模型中的数据（参见 主体相关数据图存储模型）。在表格的给定行中，属性 **requiredObjectPosition** 和**optionalObjectPosition** 可以根据相应 KGBB 存储模型中的宾语位置数量重复出现。除此之外，陈述单元还会针对存储模型中的每个宾语位置，分别设置一个额外的 **对象相关数据图表**，其中包含的槽位和值与其对象相关数据图存储模型中定义的相同（参见 对象相关数据图存储模型）。

语义头表与一组主体相关表和对象相关表相结合，构成了语义单元数据在**语义表** 中的表示形式，可将其存储和管理在关系数据库中。语义表在语义上与基于图的表示同构，例如 RDF 中带命名图的表示，或带标签属性图中的表示——不会丢失任何信息。换言之，可以在这些不同的后端技术之间来回迁移数据，而不会丢失信息。

我们计划将 KGBB 驱动的知识图谱应用的存储选项扩展到其他 NoSQL 数据库，例如 **键值数据库**，但我们仍需评估它们在原理上是否适用。Wikibase 也同样适合作为另一种存储选项进行评估。

首个 KGBB 引擎原型

在一个硕士论文项目中，我们通过开发 KGBB 引擎的基础原型，评估了 KGBB 框架的能力。该原型可处理一组已定义的 KGBB 规范，以创建 KGBB 驱动的应用。迄今为止，它能够处理所有类型的多元陈述单元、项目单元，以及其存储模型遵循 LinkML 表示法模式并以 YAML 编写文档的类型化陈述单元。此外，它还能读取 KGBB 驱动的知识图谱应用的语义规范，使用 KGBB 类的实例、关联及上述链接节点，并创建相应的知识图谱应用。该原型还可通过 KGBB-CRUD API 和 LinkML 存储模板使用多种数据库技术，既支持通过 SPARQL 查询生成来使用 RDF 三元组存储，也支持通过 Cypher 查询生成来使用 Neo4j 数据库。由于采用模块化方式，未来还可以支持其他存储技术，例如 SQL 数据库或键值存储（https://gitlab.com/TIBHannover/orkg/semantic-building-blocks）。KGBB 引擎的开发遵循六边形架构方法（53,54），通过使用各自的 API 和适配器将新功能集成到现有功能堆栈中，从而支持以协作方式开发新功能。

KGBB函数

KGBB函数 可以通过其访问 API 与 KGBB 引擎进行通信，从而从图中检索数据，并以 JSON、RDF 或 CSV 的形式将数据发送回图中；或者，它们可以利用引擎生成的 Java 和 Python 数据类，以便更好地与底层数据结构进行交互。这降低了软件开发人员向 KGBB 框架贡献新的 KGBB函数的门槛。KGBB函数可以存储在 KGBB 仓库中并共享，以便重复使用。

这类函数的一个示例是单位转换，这是一种可与任何测量陈述 KGBB 关联的 KGBB 函数，用于在各种测量单位之间转换测量陈述单元的值，例如将千米转换为英里。另一个示例是比例计算 ，用于自动计算某个实例的密度值，例如根据给定宾语的质量和体积进行计算。计算结果可以通过其导入 API 反馈给 KGBB 引擎，并自动向图中添加相应的密度陈述单元，或将其作为额外的动态内存信息提供。这样，质量、体积和密度测量陈述单元对应的三个陈述型 KGBB 就可以在比例计算 KGBB 函数中指定，从而构成功能模块。排名、演绎、归纳，以及基本上任何基于推理或基于规则的推理得出的其他类型的动态信息，都可以通过 KGBB 函数提供。

此外，还可以在 KGBB函数中实现用于可视化语义单元中数据的工具，例如使用 Vega-Lite（64）创建交互式数据图形，或使用 GeoSPARQL、OpenStreetMap 以及其他基于地图的可视化功能来管理地理空间数据。

通过针对 KGBB 引擎的不同 API（尤其是 Access REST API）开发 KGBB函数，可以将 KGBB函数作为**与特定知识图谱构建模块相关联的功能构建模块，从而形成交互单元**添加到任何知识图谱应用中。

一些 KGBB函数将是 KGBB 引擎不可或缺的组成部分，例如用户访问控制、知识图谱管理操作、数据验证服务，或按特定类型的标签（rdfs:label、skos:prefLabel、kgbb:dynamicLabel）进行搜索等便利服务；其他 KGBB函数则将构成可选服务层，例如图可视化服务、数据分析工具或评论功能。

KGBB 编辑器

为了进一步实现开放、免费使用 KGBB 框架这一目标，我们还可以开发一款易于直观使用的**低代码KGBB编辑器**，使**没有语义学、图建模和编程经验的研究人员和领域专家**能够通过将其指定为 KGBB 规范图的形式，自行定义 KGBB 及由其驱动的FAIRer知识图谱应用（用户故事 *9*）。

需要指定新的 KGBB 类时，KGBB 编辑器的用户会先从 KGBB 类分类体系中选择一个合适的通用 KGBB 类，再为其添加更多约束和参数，从而得到一个更具体的 KGBB。或者，需要指定新的陈述 KGBB 时，KGBB 编辑器会引导用户回答以下一系列问题，这些问题的答案

使 KGBB 编辑器能够创建新的陈述 KGBB 类及其关联的陈述单元类，**而无需用户做出任何语义建模决策**：

1. **该陈述的谓词或动词是什么？**这是陈述单元类标签的一部分。 2. **请提供对您希望添加到知识图谱中的陈述类型的描述或特征说明。它将涵盖哪类命题？**提供新创建的陈述单元类的人类可读定义。也可以采用对该陈述谓词进行定义的形式。 3. **请提供一些陈述示例。** 4. **请指定该陈述应涵盖的宾语位置数量。**5. **请为主语位置和每个宾语位置提供简短且有意义的标签。**每个标签都发挥**主题标签**的作用，用于描述与各个位置相关联的**语义角色**（参见 (48)）。这些标签有多种用途，包括作为输入字段的占位标签，以及作为文本和图形显示模板中变量的占位标签。此外，它还为相应的对象位置类提供标签。 6. **请指定这些宾语位置中，哪些是构成语义完整陈述所必需的（即论元），需要与主语和谓词共同构成陈述；其余宾语位置则均为可选项（即附加语）。**这将指定陈述 KGBB 的主体相关存储模型中，通过属性 **requiredObjectPosition** 链接的宾语位置，其余宾语位置则通过 **optionalObjectPosition** 属性进行链接。在编辑器的用户界面中，可以通过复选框完成此操作。 7. **请为每个宾语位置提供简短描述，说明该宾语位置涵盖的客体类型，并给出一些典型示例。**这将为相应的对象位置类提供人类可读定义。该文本还可用作添加或编辑相应陈述单元时对应输入字段的工具提示。 8. **请确定每个宾语位置所对应的客体必须以资源还是字面量的形式表示。如果是资源，请选择最能限定该宾语位置所允许客体类型的本体类；如果是字面量，请选择数据类型，并在需要时定义特定数据类型约束。**这将为每个宾语位置提供输入约束，并将其记录在陈述 KGBB 的对象相关数据图存储模型中。 9. **请确定陈述的谓词是否适用任何逻辑属性（例如传递性）；如果适用，请指定受这些属性影响的宾语位置。**KGBB 编辑器会~~通过相应的布尔型注~~释属性（例如 **transitive**），将此信息添加到相应的对象位置类中。此信息为可选项。 10.**请使用主语和各个宾语位置的主题标签以及谓词，编写一条人类可读陈述。**为该陈述单元提供动态标签模式。

基于陈述型 KGBB 存储模型的通用结构，KGBB 编辑器随后会处理这些信息，并自动指定存储模型、创建相应的 KGBB 类、陈述单元类、所有对象位置类，以及用于

动态标签的显示模板。这里的对象位置类发挥的作用类似于 ProbBank（50）中的句法位置及其相关语义角色。现在，新创建的 KGBB 已可直接使用。

在后续需要用户参照存储模型的步骤中，例如指定访问模板时（此时需要将资源从存储模型映射到访问模型；参见图 7、9、12、13、14），KGBB 编辑器会使用主语位置和不同对象位置类的主题标签。

讨论

对 KGBB 框架的主要批评之一是，知识图谱中每种所需的命题类型都需要指定一个陈述型 KGBB。这一局限源于这样一个事实：在 KGBB 驱动的知识图谱应用中，陈述型 KGBB 定义了命题空间。

尽管这一批评有其道理，但我们想回应的是，要实现真正的 FAIR 数据和元数据，就必须能够单独识别知识图谱中的所有陈述，并且每条陈述都必须引用其建模所依据的图模式（模式互操作性[F6.1/I7.1]；用户故事 *2*）。对知识图谱中使用的每种陈述类型进行建模，并非基于 KGBB 的方法所独有，而是普遍适用于 FAIR 知识图谱。与其他知识图谱框架的区别在于，KGBB 编辑器将大力支持为新型陈述创建语义模型，而且无需具备语义学、RDF、OWL 或任何图查询语言方面的经验。此外，陈述型 KGBB 一经指定并发布到 KGBB 仓库，其他任何参与方都可以重复使用，从而减少定义任何 KGBB 驱动的知识图谱应用所需的工作量。

KGBB、EOSC 互操作性框架与认知互操作性

欧洲开放科学云 (EOSC) 是一个用于托管和处理研究数据的环境，旨在支持欧盟科学。在 FAIR指导原则的背景下，EOSC 创建了一个 EOSC 互操作性框架，将互操作性区分为技术、语义、组织和法律互操作性（10）。我们相信，KGBB 框架尤其有助于改善前两种类型的互操作性。

运行由 KGBB 驱动的知识图谱应用的机构可以与合作伙伴共享数据，甚至共同构建联邦虚拟知识图谱，同时由各自机构自行管理数据，从而确保完全掌控数据在伦理、隐私和法律方面的事宜（遵循 Barend Mons 提出的数据访问而非数据共享(9)）。此外，通过将应用层与持久化层解耦，并允许将现有知识图谱迁移到各种不同的存储技术中；通过提供指定其他访问模型的可能性，并支持以各种格式访问知识图谱中的数据（包括 Java 和 Python 数据类）；通过支持以不同粒度记录数据和元数据；通过将 KGBB 规范图纳入知识图谱本身，从而确保有关应用本身的语义透明性；并通过公开、免费提供 KGBB、KGBB函数、KGBB 引擎、KGBB 编辑器和 KGBB 前端

，KGBB 框架符合 EOSC 互操作性框架关于**技术互操作性**的建议(10)。

另一方面，通过使用为其类术语提供 UPRI 和公开可用定义的本体术语及其他受控词表，通过跟踪和记录元数据，并允许扩展已记录的元数据，以同时纳入学科元数据，以及通过提供指定元数据交叉映射的方式；通过将人类可读的数据展示与机器可操作的数据存储解耦，前者侧重于提供人类可读的数据视图，后者则涉及一致应用 KGBB 专用存储模型，以确保数据和元数据的语义互操作性；通过提供应用社区专用标准来访问数据和元数据的可能性；并通过为每个语义单元的每个数据图记录其数据建模所使用的存储模型，KGBB 驱动的知识图谱应用也满足了关于 **语义互操作性** 的建议（10）。

除了这两个互操作层之外，我们认为，KGBB 框架还通过为图添加更多结构，并将其组织成属于不同表征粒度层级的各种语义单元层，从而支持提高数据和元数据的认知互操作性；通过为人类读者提供直观的图探索工具，利用这些新增的层和粒度层级，以及不同的粒度树和参照框架，将图的复杂性降低到人类读者当前关注的内容，并以使其能够过滤掉仅与机器相关的信息的方式组织图；同时使其能够跨不同的表征粒度层级进行放大和缩小。提高< 스타일 id='1'>数据和元数据的人类可探索性 ，因而支持向人类读者传达图内容的方式——随着图规模不断扩大，这一点变得日益重要——从而不仅确保数据和元数据的机器可操作性，也确保其人类可操作性，进而确保其人性化，由此满足关于**认知互操作性** 的建议（详细讨论见（14））。

KGBB 框架还通过提供易于使用的工具，降低部分复杂性层级和要求（例如必须精通查询语言、形状语言或 RDF 和 OWL），从而满足知识图谱应用的开发者和运维人员在认知互操作性方面的要求。KGBB 编辑器将根据特定陈述单元类型所需输入和可选输入的规范，自动指定存储模型；KGBB 查询构建器则将基于该存储模型生成通用 CRUD 查询，因此开发者和运维人员无需自行定义存储模型和编写查询。此外，借助其模块化架构，一旦定义了 KGBB 并开发了 KGBB函数，就可以通过低代码KGBB编辑器，以直接明了的方式加以复用和连接，从而在 KGBB 规范图中指定自己的知识图谱应用。每个 KGBB 驱动的知识图谱应用都会通过此类 KGBB 规范图进行记录，该规范图以语义透明且符合 FAIR 原则的方式表示图中所使用的不同 KGBB 及其连接关系。因此，对知识图谱底层数据模型总体架构的理解也能得到支持，并有助于实现 KGBBFramework 所提供的整体认知互操作性。

重新审视用户故事

在 KGBB 框架中，数据和元数据的人类可操作性与机器可操作性之间不存在权衡，因为由 KGBB 驱动的知识图谱应用**将数据展示与数据存储解耦**。每个 KGBB 都为相应类型的语义单元的数据和元数据提供基于表单的文本显示模板以及类似思维导图的图形

显示模板，用于提供数据和元数据的一个或多个视图。换言之，每个 KGBB 都为表示层提供信息，说明应如何展示相应单位的数据图中的哪些节点所包含的哪些信息；还可以使用不同的预定义显示模板，以不同方式展示相同的数据和元数据。这为指定依赖上下文的数据和元数据表示提供了概念基础。因此，存储模型支持以机器可操作格式存储信息，并明确提供机器所需的全部信息；而显示模板则定义如何将数据图中的信息缩减为人类读者实际相关的内容，以及如何在用户界面中呈现这些信息。KGBB 框架中实现的语义单元概念（23）由此为开发适当的用户界面提供了新的概念框架，从而增强数据和元数据的**认知互操作性**，将 FAIR 知识图谱转变为 FAIREr 知识图谱，即更易于人类探索的知识图谱（14）。因此，**使用 KGBB 框架，Tom可以生成大型、复杂的 FAIR 数据图，同时仍能直观地进行探索**（用户故事 ***1***）。

使用 KGBB，并且可以将不同的 KGBB 实例彼此连接起来，同时每个 KGBB 类都提供显示模板和访问模板，以及带有输入约束的存储模型，这使知识图谱应用的开发者能够在 KGBB 实例及其可能交互的抽象层面上指定其应用程序。由此产生的 KGBB 规范图规定了应用程序允许其 KGBB 之间进行的所有交互。在向此类知识图谱应用添加数据时，用户只能看到引导其向图中添加内容的输入表单——他们无需自行做出语义和数据建模选择。由于相同类型的每个语义单元都使用特定于 KGBB 的相同存储模型进行存储，并作为 FAIR数字对象记录在图中，因此通过 KGBB 贡献的所有数据都将完全可互操作且符合 FAIR。因此，**作为 KGBB 驱动的众包知识图谱的用户，Catherine 和 Felix 可以添加 FAIR 数据，而无需自行做出任何语义建模决策**（用户故事 ***2***）。

通过将知识图谱的数据图层组织为不同类型的语义单元，使用各自的 UPRI 表示这些语义单元，指定它们所属的语义单元类，并为每个语义单元类关联相应的 KGBB，使其提供存储模型，KGBB 引擎便可据此自动生成 CRUD 查询。FAIREr 知识图谱应用的用户可以使用不同 KGBB 提供的查询，并在 KGBB 前端和 KGBB 查询构建器的支持下，以模块化方式像拼接查询构建块一样将它们组合起来，形成单个查询结果的并集或交集，甚至可以将其保存并记录为问题单元。每个 KGBB 提供的输入表单可用于指定查询参数，以进行更详细的查询，让用户无需精通任何图查询语言，也能直观地创建查询。由于 KGBB 引擎会根据每个陈述 KGBB 提供的存储模型规范自动生成 CRUD 查询，因此，没有编程经验、也不了解 Cypher 或 SPARQL 等图查询语言的研究人员，也可以使用 KGBB 查询构建器组合引擎提供的查询，创建新的查询。因此，对于 KGBB 驱动的知识图谱应用，Sara 和 Bob 在查询知识图谱时不一定要了解图查询语言，因为他们可以使用 KGBB 引擎的 KGBB 查询构建器，直观地创建自己的查询（用户故事 3）。

KGBB 引擎会自动将知识图谱组织成多个层级的、**具有语义意义的子图**，并识别不同的

粒度树和图中的参照框架，同时为每个此类子图分配一个 UPRI。因此，生成的语义单元**可被识别**，并可在三元组陈述中**引用**，从而轻松实现针对陈述作出陈述。自动将数据图构建并组织为不同的语义单元，为组织知识图谱提供了一种高效方式，使用户能够直观地针对陈述作出陈述；同时也提供了清晰直观的实现模式，有助于开发相关的搜索模式。因此，**通过采用 KGBBFramework，Karl 可以运行一个 KGBB 驱动的知识图谱应用，为用户提供直观的方式来针对陈述作出陈述**（用户故事***4***）。

KGBB 框架通过将数据从存储模型映射到为每个陈述 KGBB 定义的访问模型之一，实现了**数据访问与数据存储的解耦**。用户可以根据指定的任意访问模板，访问属于特定语义单元的数据和元数据。每个 KGBB 都允许指定一个或多个此类访问模型。用户可以随时使用低代码 KGBB编辑器添加其他访问模型。如果定义了多个访问模型，KGBB 引擎甚至支持在不同访问模型之间进行**模式交叉映射**。因此，KGBB 框架允许将新兴标准纳入现有的、由 KGBB 驱动的知识图谱中，从而能够适应不断发展的研究格局。此外，还可以为相同类型的数据纳入不同的社区特定数据模型。这同样适用于所有溯源信息及其他类型的元数据，从而也支持全面的**元数据交叉映射**。在访问或导出数据和元数据时，用户只需从指定的访问模型（或访问模型族）中选择要使用的模型即可。因此，**通过为 KGBB 指定不同的访问模型，Dan 和 Anna 可以将其测量数据作为 FAIR 数据贡献出来，同时仍能够以符合各自社区特定领域标准的格式或模型访问这些数据**（用户故事***5***）。

KGBB 框架**将存储模型与存储技术解耦**。KGBB 引擎提供了一个应用程序接口，使知识图谱能够存储在各种数据库技术中，并可从中访问，从而允许知识图谱应用随时更换其数据库技术。此外，将整体数据图组织成语义单元，可以显著支持数据图的查询特性，因为查询能够系统地拆分为不同步骤。这使存储模型在查询方面具有更大的灵活性。由于可以通过指定各种访问模型来提供更详细的模型，因此存储模型可以针对高效存储数据和元数据以及可扩展性进行优化，无需考虑任何特定领域的标准和要求。此外，例如，如果给定的知识图谱存储在某个图数据库中，而该数据库在某个时刻不再具备良好的可扩展性，那么相应的数据和元数据始终可以迁移到关系数据库中，并存储为语义表；由于存储模型通用结构的扁平图模式，它们也会以表格格式存储。所有语义都得以保留。如果需要数据的 RDF 或 Neo4j 表示形式，则可以使用相应的访问模板，将该表作为 RDF 或带标签属性图进行访问。因此，**借助 KGBB 驱动的知识图谱应用，如果当前技术遇到扩展性问题，John 始终可以更换持久化层中使用的技术**（用户故事***6***）。

将知识图谱组织成语义单元，可以把 KGBB 及其主语位置和宾语位置分配到表格数据集的特定行和列中。深度学习文本挖掘算法可用于识别单元格中的值，并将其映射到数据类型和本体术语。因此，可以大幅支持并加快从

表格格式导入遗留数据并逐步对其进行语义化。此外，如果知识图谱中已经包含大量数据，这些数据可用作深度学习算法的训练数据。由于数据已按 KGBB 及其存储模型参数进行分区和关联，因此可以轻松识别适用于特定 KGBB 的基于人工智能的自然语言处理任务的训练数据集。因此，**KGBB 驱动的知识图谱应用或可支持 Mila 将语义结构不明的（遗留）数据从表格格式（例如 CSV、Excel 工作表）导入 FAIR 知识图谱中**（用户故事 ***7***）。

通过将知识图谱组织为不同的语义单元，并将每个陈述单元划分为通用型、偶然型、原型型、断言型或词汇型陈述单元，同时利用资源来表示命名个体、某实例、每个实例和类实体，KGBB 驱动的知识图谱提供了一种概念框架，有助于在知识图谱中对通用型、偶然型、原型型、断言型和词汇型陈述进行形式化区分。此外，结合使用四种类型的资源实体与语义单元，可以实现否定和基数限制的表示法，无需使用任何空白节点，而且比其他基于 OWL 的表示法更容易理解（参见（23））。因此，**通过使用 KGBB 驱动的知识图谱应用，Nick 可以在 FAIRER 知识图谱中记录临床研究数据，区分通用型、原型型和断言型陈述，并直观地记录缺失情况、一般意义上的否定以及基数限制**（用户故事 ***8***）。

通过提供一种可应用于任何命题、进而适用于任何类型陈述单元的通用存储模型，KGBB 编辑器使不具备任何语义学经验的领域专家也能够定义新的陈述型 KGBB，并借助这些 KGBB 创建新的陈述单元类，而无需考虑其底层数据模型——领域专家只需回答一系列问题即可。因此，**通过使用低代码KGBB编辑器，Paula 无需考虑如何对这些陈述型 KGBB 进行语义建模，就能够定义她所需的陈述型 KGBB，以记录其项目中的数据(**用户故事 ***9*****)**。

相关工作与未来工作

据我们所知，语义单元的概念或类似概念此前尚未被提出。由于 KGBB 在概念上依赖于语义单元并与之相关，因此没有工具或框架可供我们将 KGBB 框架与之比较，但我们可以将其某些方面和特征与其他现有框架进行比较。

在范围上最相似的可能是实现了 **基于本体的数据访问（OBDA）范式** 的系统。通过一个概念层访问数据，这一范式已成为使用语义网技术（65）访问存储在遗留来源中的数据的流行范式；该概念层为连接到系统的所有不同关系型遗留数据源提供查询词汇。概念层以 RDF(S) 或 OWL 本体表示，并使用 R2RML 映射连接到底层关系数据库。这使得可以使用 SPARQL 查询关系数据库。

商业企业知识图谱平台Stardog 遵循 OBDA 范式，使企业能够访问并统一存储在各种不同类型的专有数据库系统中的所有非结构化和半结构化数据源，从而形成一个虚拟连接的企业知识图谱。Stardog 声称能够帮助统一企业中可能存在的所有不同数据孤岛，让数据保留在原处，同时仍可通过联邦式

查询引擎实现对这些数据的利用。Stardog 的数据虚拟化引擎访问孤立数据源，为知识图谱提供更多上下文。反过来，其语义图谱在孤立数据源之上增加了一个元数据驱动的抽象层，使用户能够查询这些数据，为其注入含义，并返回一致且统一的数据表示。它还支持应用替代模式，以满足不同利益相关者的特定需求和要求。其推理引擎运用机器学习和推理能力，创建新的连接，并通过新增节点和关系补充现有数据。

Ontop 是一个开源框架，同时通过将SPARQL 查询转换为 SQL 查询来实现 OBDA 范式，从而将一组关系数据库转换为**虚拟知识图谱**，其内容由领域本体（66）丰富。

Metaphactory 是用于构建 FAIR 知识图谱应用的企业平台。它具备多种内置功能，可为不同类别的用户提供服务，涵盖不具备语义学或计算机科学背景的领域专家、专家用户和数据科学家，以及数据管理员和应用开发者。它提供基于 Web 组件的用户界面，并采用可定制的模板机制，允许为同一类型的资源指定关联的模板页面，使专家用户能够创建自己的用户界面。这与 KGBB 专用的基于表单的显示模板类似，专家用户也可以配置这些模板。
Metaphactory 还提供基于知识图谱模式的创作控制，其功能与 KGBB 存储模板所提供的控制类似。此外，与 KGBB 框架一样，Metaphactory 支持通过可复用组件快速构建知识图谱应用，并支持协作式知识建模和知识生成。通过支持集成 Ontop 等基于本体的数据访问引擎，或 Stardog 等同时集成三元组存储的引擎，Metaphactory 可以虚拟集成原生 RDF 三元组存储与可通过 SPARQL 访问的关系数据库 (67)。然而，尽管 Metaphactory 遵循开放标准，它仍是一个非开源的商业平台。

尽管存在一些重叠，但 Metaphactory、Stardog 和 Ontop 的整体范围和目标与 KGBB 框架大相径庭。这三个系统都不支持语义单元，也不支持任何可用于提升知识图谱可探索性及其认知互操作性的类似方法。它们专注于将已有的非语义数据源整合到虚拟知识图谱中，而 KGBB 框架则着重于向新建的 FAIREr 知识图谱体系中添加新数据。

Apache Stanbol 是一个开源的模块化软件栈，包含一组可复用组件，用于语义内容管理。它提供内容增强服务，可为内容添加语义信息；提供推理服务，可根据内容增强过程中提取的语义信息检索更多语义信息；还提供用于定义和操作数据模型（例如本体）的服务，以存储语义信息。 **语义MediaWiki** 是一个免费、开源的扩展，适用于MediaWiki ，可通过在 wiki 页面中存储和查询数据，将 wiki 转变为灵活的知识管理系统。Apache Stanbol 和 MediaWiki 均~~不支持语义~~单元或任何类似概念，也都不会创建~~知识图谱~~，而是为非语义化内容添加语义信息。

SHACL 和ShEx 是用于描述 RDF图结构（即形状）的形状约束语言，用于标识谓词及其关联的基数和数据类型。形状可用于传达数据结构、创建、集成或验证图，以及生成用户界面表单和代码。

合理本体模板 (OTTR)采用了形状的理念，并将其用作知识库的构建模块。模板提供了一个抽象层级，相较于底层的 RDF 三元组或 OWL 公理，它更适合用于管理知识库。模板存储在模板库中，支持重复使用，从而使不同知识库之间的建模保持一致。由于模板可以引用其他模板，OTTR 的模块化程度可与 KGBB 框架相媲美。此外，OTTR 将模板与以数据形式表示的实例化明确分开，从而清楚地区分了知识库设计与知识库内容。这一点也与 KGBB 框架相似。现有工具可将 CSV 或关系数据映射到模板。

与 OTTR 类似的是 LinkML。 **LinkML** 是一种通用、平台无关、面向对象的数据建模语言和框架，旨在让大众也能使用语义网标准，并让 FAIR 数据的存储、验证和分发变得简单（68）。它可与知识图谱应用配合使用，为多种数据类型制定模式，范围从简单的扁平清单标准到利用继承的复杂互相关联规范化数据。LinkML 能很好地融入大多数开发者和数据库工程师熟悉的框架，例如 JSON 文件、关系数据库、文档存储和 Python 对象模型，同时通过将所有元素映射到 RDF URI，提供坚实的语义基础。使用 LinkML，您可以通过采用 LinkML 表示法编写 YAML 文件，对数据模式和数据字典进行建模。

LinkML 还提供各种工具。生成器可自动将模式 YAML 转换为多种格式，包括 JSON Schema、JSON-LD/RDF、SPARQL、OWL、SQL DDL、ShEx、GraphQL、Python 数据类、Markdown 和 UML 图。加载器和转储器可在这些格式之间转换某个模式的实例。因此，生成器能够与其他技术栈提供的工具集成。SPARQL 生成器可根据模式生成一组 SPARQL 查询，而 Excel 电子表格生成器则可创建模式的电子表格表示。LinkML 目前支持四种不同的数据验证策略：1）通过实例化 Python 对象进行验证；2）通过 JSON Schema 进行验证；3）通过生成 SPARQL 约束，对三元组存储或 RDF 文件中的三元组进行验证；或 4）通过生成 ShEx 或 SHACL 对 RDF 进行验证。

此外，还存在针对特定编程语言的对象模型生成器，例如 Python、Pydantic、Java 或 TypeScript。Java 生成器可根据 LinkML 模型生成 Java 类文件，并可选择支持用户提供的 Jinja2 模板。此外，还有针对特定数据库框架的生成器，包括 SQL DDL 和 SQLAlchemy。换言之，LinkML 既支持复杂、相互关联的规范化关系数据，也支持扁平的反规范化数据，例如来自电子表格和 CSV 的数据。

LinkML 提供了我们计划用于文档编写的表示法，以及用于处理各种 KGBB 存储模型的工具集。

我们也在持续关注 抽象维基百科 项目，尤其是与 **构造单元** 和抽象内容语言相关的部分。构造单元提供谓词陈述的抽象表示，因此遵循了一种与我们针对陈述型 KGBB 的通用存储方法具有某些相似之处的理念，其中区分了必需的宾语位置和可选的宾语位置。此外，包含构造单元的超级构造器这一可能性，与语义单元也有某些相似之处：复合单元由一组相互关联的语义单元组成，其中一些语义单元本身也可能是复合单元。我们同样感兴趣的是，将一个构造单元用多个句子进行语言化，以提高其可读性；同时，还可以为其提供多种可能的类句实现，例如使用不同的语言，并且所有这些实现都在渲染时提供。

结论

如今，面对必须快速生成新的 FAIR 数据和元数据，同时满足能够与来自各个领域的现有数据和元数据来源实现整合这一要求的压力，知识图谱管理平台必须支持快速开发数据管理解决方案。这些解决方案应适用于日常研究，能够体现知识图谱技术所带来的附加价值，并且还应能由预算有限的小型项目，甚至由个体研究人员为其个人研究知识图谱实施（69）。

理想情况下，这样的平台能够满足不同类别用户群体的需求：**领域专家**对应用逻辑或图的底层数据结构并不感兴趣。他们不关心为了让数据和元数据变得机器可操作而必须添加的额外信息，而是希望以直观的方式访问图，并且图中只呈现他们当前感兴趣的信息。

专家用户和数据科学家则希望与数据交互，也许使用 Jupyter Notebook、R 或其他数据分析工具进行分析；他们需要能够轻松访问知识图谱的数据和元数据，并采用自己偏好的格式，还希望知识图谱提供便捷高效的工具，以支持其数据管理操作。

最后，**数据管理员、本体工程师和应用工程师**需要能够轻松地将知识图谱基础设施整合到组织的整体软件栈中，同时全面掌控其数据在伦理、隐私或法律方面的问题（遵循 Barend Mons 提出的数据访问而非数据共享 （9））。他们希望该平台能让自己在知识图谱之上开发新的目标应用程序时更加轻松。

除了在知识管理系统中实现语义单元这一核心理念外，KGBB 框架还能够清晰地区分内部的内存中的应用数据模型、数据存储模型、数据显示模型以及数据访问/导出模型，并将应用数据模型与在数据库中存储数据及在用户界面中展示数据这一相对技术性的方面解耦。我们认为，这种解耦对于解决知识管理系统中许多内在问题至关重要。在由KGBB驱动的FAIREr知识图谱应用中，KGBB 引擎基于一组相互链接的 KGBB 实例所提供的信息，管理不断增长的知识图谱，并将其组织和构建为多个不同类型的语义单元；该引擎支持以直观方式对陈述作出陈述，为用户界面中的用户提供输入表单和人类可读的数据视图，强制执行图模式，以确保数据和元数据在内部具备互操作性和机器可操作性，从而使其更符合 FAIR 原则；同时，它还支持为访问和导出指定额外的数据模式，并可进一步添加用于适应新出现标准的数据模式；此外，它提供从每个 KGBB 的存储模型派生的 CRUD 查询，这些查询可以在查询构建器中以直观方式使用、组合和进一步指定，用于搜索和探索知识图谱。除此之外，通过在 KGBB 规范图中指定由 KGBB 驱动的FAIREr知识图谱，可以构建一个自描述知识图谱应用，满足 FAIR 软件的要求。

借助低代码KGBB 编辑器和开放可访问的KGBB 仓库，KGBBFramework 提供了一套资源和工具，通过允许其他软件应用程序经由其 Access REST API 与图进行互操作，提升知识图谱的数据和元数据的整体互操作性。该 API 支持通过一组键值参数进行 REST API 调用来检索数据，无需发送 SPARQL 或 Cypher 查询。领域专家可以定义自己的

KGBB 驱动的更符合 FAIR 原则的知识图谱应用，开发者则可以访问并将数据导出为 JSON、RDF 或 CSV。更具FAIR性的知识图谱的用户可以直观地探索图中的内容，而机器则能够访问所需的所有信息。

KGBB 框架将提供一个开源的知识图谱管理应用平台，借助其核心 KGBB 引擎，为不同领域提供可复用的数据管理解决方案。由于具备灵活的数据和元数据访问选项，该平台还兼容其他应用程序，并且可以轻松集成到现有的软件栈中。其真正的模块化架构支持灵活性、可扩展性和可定制性，同时仅需较低的初始投入即可完成设置。

我们认为，除了关注数据和元数据的机器可操作性之外，现在也到了开始关注其人类可操作性及人性化程度的时候，从而将 FAIR 知识图谱转变为更具FAIR性的知识图谱，以满足认知互操作性的要求。KGBB 框架为我们如何实现这一目标提供了一种建议。

致谢

我们感谢 Philip Strömert、Roman Baum、Björn Quast、Peter Grobe、István Míko 和 Kheir Eddine，他们与我们讨论了本文所 प्रस्तुत的部分观点。本文中的所有论元和陈述均由我们独自负责。本研究得到了 ERC H2020项目“ScienceGraph”（819536）的支持。我们还要感谢德国的纳税人。

Reference

1. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A，等。《科学数据管理与治理的FAIR指导原则》。Sci Data。2016年12月；3(1):160018。2. Jacobsen A, de Miranda Azevedo R, Juty N, Batista D, Coles S, Cornet R，等。《FAIR原则：诠释与实施考量》。Data Intell。2019年11月；(November):10–29。3. Bahim C, Casorrán-Amilburu C, Dekkers M, Herczog E, Loozen N, Repanas K，等。《FAIR数据成熟度模型：协调FAIR评估的方法》。Data Sci J。2020年10月27日；19:41。4. Garijo D, Corcho O, Poveda-Villalon M。《FOOPS!：用于FAIR原则的本体陷阱扫描器》。：4。5. Poveda-Villalón M, Espinoza-Arias P, Garijo D, Corcho O。《理解FAIR本体》。载于：Keet CM, Dumontier M，编者。《知识工程与知识管理》[互联网]。Cham：Springer International Publishing； 2020 [引自 2021]年9月3日。第255–70页。（计算机科学讲义；第12387卷）。网址：http://link.springer.com/10.1007/978-3-030-61244-3_186。Hasnain A, Rebholz-Schuhmann D。《依据五星开放数据原则评估FAIR数据原则》。载于：Gangemi A, Gentile AL, Nuzzolese AG, Rudolp S, Maleshkova M, PaulheimH，等，编者。《语义网：ESWC 2018卫星活动》，ESWC 2018计算机科学讲义，第 11155 [卷，互联网]。Springer；2018年。第469–77页。网址：http://link.springer.com/10.1007/978-3-319-98192-5_607。欧盟委员会研究与创新总司、普华永道欧盟服务。《FAIR研究数据的成本效益分析：缺乏FAIR研究数据的代价》。 [互联网]。卢森堡：出版办公室； 2018 [引自 2021]年9月22日。网址：

https://data.europa.eu/doi/10.2777/02999 8. Ayris P, Berthou JY, Bruce R, Lindstaedt S, Monreale A, Mons B，等。《实现欧洲开放科学云》 [互联网]。2016年，第24页。网址：https://ec.europa.eu/research/openscience/pdf/realising_the_european_open_science_cloud_2016.pdf 9. Mons B。《CODATA主席寄语，Barend Mons》 [互联网]。2019年。网址：https://codata.org/about-codata/message-from-president-barend-mons/message-from-president-geoffrey-boulton/ 10. 欧盟委员会研究与创新总司、EOSC执行委员会。《EOSC互操作性框架：EOSC执行委员会FAIR与架构工作组报告》。 [互联网]。卢森堡：出版办公室； 2021 [引自 2021]年8月20日。网址：https://data.europa.eu/doi/10.2777/620649 11. 欧盟委员会FAIR数据专家组。《让FAIR成为现实：欧盟委员会FAIR数据专家组最终报告与行动计划》。2018年。78页。12. Hogan A, Blomqvist E, Cochez M, D'amato C, Melo GD, Gutierrez C，等。《知识图谱》。ACM Comput Surv。2021年7月；54(4):1–37。13. Meid S, BaumR, Bhatty P, Grobe P, Köhler C, Quast B，等。《SOCCOMAS：一种自描述、内容无关的语义本体控制Web内容管理系统应用》。Biodivers Inf Sci Stand。2017年8月；1:e20033。14. Vogt L。《将FAIR拓展为FAIREr：数据与元数据的认知互操作性和人类可探索性》 [互联网]。ResearchGate；2023年。网址：https://www.researchgate.net/publication/361341263_Extending_FAIR_to_FAIREr_Cognitive_Int eroperability_and_the_Human_Explorability_of_Data_and_Metadata 15. Ceusters W。《指称对象跟踪在生物医学信息学中的地位》。载于：《术语学、本体及其实现》。Springer Nature；2020年。第50页。16. Ceusters W, Elkin P, Smith B。《电子健康记录和生物医学本体中的阴性发现：实在论方法》。Int J Med Inf。2007年12月；76:S326–33。17. Knublauch H, Kontokostas D。《形状约束语言（SHACL）——W3C 2017年7月20日推荐标准》2017 [互联网]。2017年。网址：https://www.w3.org/TR/shacl/ 18. Staworko S, Boneva I, Gayo JEL, Prud' hommeaux EG, Solbrig H。《ShEx用于RDF的复杂性与表达能力》。：18。19. Gayo JEL, Prud' hommeaux E, Staworko S, Solbrig H。《迈向一种基于正则表达式导数的RDF验证语言》。CEUR研讨会论文集。2015年；1330:197–204。20. Lupp DP, Hodkiewicz M, Skjæveland MG。《工业资产维护模板库：构建可扩展、可维护本体的方法》。载于：《第13届可扩展语义网知识库系统国际会议（SSWS 2020）》，与国际语义网会议（ISWC 2020）同期举行。2020年。第49–64页。21. Skjæveland MG, Lupp DP, Karlsen LH, Forssell H。《利用合理本体模板进行实用的本体模式实例化、发现与维护》。载于：Vrandečić D, Bontcheva K, Suárez-Figueroa MC, Presutti V, Celino I, Sabou M，等，编者。《语义网——ISWC》2018 [互联网]。Cham：Springer International Publishing； 2018 [引自 2020]年3月26日。第477–94页。（计算机科学讲义；第11136卷）。网址：http://link.springer.com/10.1007/978-3-030-00671-6_2822. Bandrowski A, Brinkman R, Brochhausen M, Brush MH, Bug B, Chibucos MC，等。《生物医学研究本体》。PLoS ONE。2016年；11(4):1–19。23. Vogt L, Kuhn T, Hoehndorf R。《语义单元：将知识图谱组织为语义上有意义的表示单元》 [互联网]。ResearchGate；arXiv；2023年。网址：https://www.researchgate.net/publication/361209259_Semantic_Units_Organizing_knowledge_graphs_into_semantically_meaningful_units_of_representation/stats；https://arxiv.org/abs/2301.01227 24. Booth D, Wallace E。《专题X：EasyRDF》 [互联网]。第二届美国语义技术研讨会

2019年。2019年。可在以下网址获取：https://us2ts.org/2019/posts/program-session-x.html 25. Frey J, Müller K, Hellmann S, Rahm E, Vidal ME. RDF 存储库中的元数据表示评估。Ngonga Ngomo AC, Fundulaki I, Krithara A, Ngonga Ngomo AC, Fundulaki I, Krithara A，编。语义网期刊。2019年1月；10(2):205–29。26. Hartig O. RDF* 与 SPARQL* 基础。载于：CEUR 研讨会论文集 1912。2017年。27. Arndt D, Broekstra J, DuCharme B, Lassila O, Patel-Schneider PF, Prud'hommeaux E 等。RDF-star 与 SPARQL-star 草案社区组报告，2019年7月1日 2021 [互联网]。可在以下网址获取：https://w3c.github.io/rdf-star/cg-spec/2021-07-01.html 28. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Poggi A, Rodriguez-Muro M 等。本体与数据库：DL-Lite 方法。载于：Tessaris S, Franconi E, Eiter T, Gutierrez C, Handschuh S, Rousset MC 等，编。面向信息系统的推理网语义技术[互联网]。Berlin, Heidelberg：Springer Berlin Heidelberg； 2009 [引自2019年11月23日 2021]。第255–356页。（计算机科学讲义；卷5689）。可在以下网址获取：http://link.springer.com/10.1007/978-3-642-03754-2_7 29. Ding L, Xiao G, Calvanese D, Meng L. 统一基于本体的地理数据集成与地理可视化分析的框架。《ISPRS国际地理信息杂志》。2020年7月28日；9(8):474。30. Bereta K, Xiao G, Koubarakis M. Ontop-spatial：基于地理空间数据库的 Ontop。第18页。31. Xiao G, Ding L, Cogrel B, Calvanese D. 虚拟知识图谱：系统与用例概览。《数据智能》。2019年5月；1(2):201–23。32. Rosse C, Mejino JLV. 生物医学信息学参考本体：解剖学基础模型。《生物医学信息学杂志》。2003年；36(6):478–500。33. Mungall CJ. RDF* 与属性图语义学的拟议策略 [互联网]。2019年。可在以下网址获取：https://douroucouli.wordpress.com/2019/07/11/proposed-strategy-for-semantics-in-rdf-and-property-graphs/ 34. Mungall CJ. Shadowboxing：在 ABox 中镜像 TBox [互联网]。2017年。可在以下网址获取：https://github.com/cmungall/shadowboxing-paper/blob/master/tbox2abox.pdf 35. Rector A, Schulz S, Rodrigues JM, Chute CG, Solbrig H. 超越 Gruber："本体"在当今生物医学信息系统中的应用及 OWL 的局限。《生物医学信息学杂志X》。2019年；2(3月):1–15。36. Mons B, Velterop J. 电子科学时代的纳米出版物 电子科学时代的纳米出版物。载于：科学话语语义网应用研讨会（SWASD 2009）。2009年。37. Kuhn T, Taelman R, Emonet V, Antonatos H, Soiland-Reyes S, Dumontier M. 通过去中心化纳米出版物服务进行语义微贡献。《PeerJ计算机科学》。2021年3月8日；7:e387。38. Giachelle F, Dosso D, Silvello G. 在网络上搜索、访问和探索生命科学纳米出版物。《PeerJ计算机科学》。2021年2月4日；7:e335。39. Bittner T, Smith B. 粒度分割的分类体系。载于：Montello DR，编。《空间信息理论：地理信息科学基础》，计算机科学讲义第2205卷 [互联网]。Berlin：Springer；2001年。第16页。可在以下网址获取：http://portal.acm.org/citation.cfm?doid=505168.505197 40. Bittner T, Smith B. 粒度分割理论。载于：Duckham M, Goodchild MF, Worboys MF，编。《地理信息科学基础》 [互联网]。London：Taylor & Francis Books；2003年。第117–49页。可在以下网址获取：http://portal.acm.org/citation.cfm?doid=505168.505197 41. Bittner T, Smith B. 粒度、模糊性与近似的统一理论。载于：COSIT空间模糊性、不确定性与粒度研讨会论文集。2001年。42. Keet CM. 粒度形式理论——以粒度增强生物学与应用生命科学信息系统 [互联网] [博士论文]。 [Bozen]：博尔扎诺自由大学；2008年。可在以下网址获取：http://www.meteck.org/files/AFormalTheoryOfGranularity_CMK08.pdf 43. Vogt L. 生物材料实体的时空结构粒度。《BMC生物信息学》 [互联网]。

2010年5月；11(289)。可在以下网址获取：
http://www.ncbi.nlm.nih.gov/pubmed/20509878 44. Vogt L. 层级与构成单元——构建生命科学领域粒度框架。《生物医学语义学杂志》。2019年12月；10(4):1–29。45. Chungoora T. 本体能力问题 [互联网]。2020年。可在以下网址获取：
https://tishchungoora.medium.com/ontology-competency-questions-3d213eb08d33 46. Marconi D. 论词汇能力的结构。《亚里士多德学会学报》。1995年6月1日；95(1):131–50。47. Vogt L, Mikó I, Bartolomaeus T. 解剖学与生物学中的类型概念表明，本体必须适应研究的诊断需求。《生物医学语义学杂志》。2022年12月；13(18):27。48. Kipper K, Dang HT, Palmer M. 基于类构建动词词库。第17届全国人工智能会议论文集 AAAI-2000。2000年；49. Levin B. 《英语动词类别与变化：初步研究》。芝加哥大学出版社；1993年。366页。50. Palmer M, Gildea D, Kingsbury P. 命题库：语义角色标注语料库。《计算语言学》。2005年3月；31(1):71–106。51. Fillottrani PR, Keet CM. 将不同建模决策联系起来的异构 TBox 映射模式。载于：计算机科学讲义（包括人工智能讲义和生物信息学讲义子系列） [互联网]。2017年。第371–86页。可在以下网址获取：https://link.springer.com/10.1007/978-3-319-58068-5_2352. Vogt L, D'Souza J, Stocker M, Auer S. 使用知识图谱单元在学术知识图谱中表示研究贡献。载于：ACM/IEEE数字图书馆联合会议论文集， 2020 [互联网]。中国虚拟会议：ACM； 2020 [引自10月27日 2020]。第107–16页。可在以下网址获取：https://dl.acm.org/doi/10.1145/3383583.3398530 53. Cockburn A. 六边形架构 [互联网]。2005年。可在以下网址获取：
https://alistair.cockburn.us/hexagonal-architecture/ 54. Garrido de Paz JM. 端口与适配器模式（六边形架构） [互联网]。2018年。可在以下网址获取：
https://jmgarridopaz.github.io/content/hexagonalarchitecture.html 55. de Waard A, Labs E, Maat HP. 科学话语中的认知情态与知识归属：类型分类体系及特征概述。载于：学术话语结构检测研讨会论文集。2012年。第47–55页。56. Prieto M, Deus H, de Waard A, Schultes E, García-Jiménez B, Wilkinson MD. 学术断言确定性的资料驱动分类。《PeerJ》。2020年4月21日；8:e8871。57. Greenberg SA. 引文失真如何造成无根据的权威性：引文网络分析。《英国医学杂志在线版》。2009年；339(7714):210–3。58. Smith B, Almeida M, Bona J, Brochhausen M, Ceusters W, Courtot M 等。基础形式本体2.0。2015年；可在以下网址获取：
h
t
tps://git
hub.com/BFO-ontology/BFO/raw/master/docs/bfo2-reference/BFO2-Reference.pdf 59. Miller GA. 神奇的数字七加减二：我们处理信息能力的某些局限。《心理学评论》。1956年；63(2):81–97。60. Pinker S. 《风格感：21世纪思考者的写作指南》。Viking Penguine；2014年。61. Ciccarese P, Soiland-Reyes S, Belhajjame K, Gray AJ, Goble C, Clark T. PAV 本体：溯源、创作与版本管理。《生物医学语义学杂志》。2013年1月；4(1):37。62. Vogt L. 以语义单元构建 FAIR 知识图谱——一个原型。ResearchGate；2022年。63. Groth P, Gibson A, Velterop J. 纳米出版物的构成。《信息服务与使用》。2010年；30(1–2):51–6。64. Deagen ME, McCusker JP, Fateye T, Stouffer S, Brinson LC, McGuinness DL 等。基于科学知识图谱的 FAIR 与交互式数据图形。《科学数据》。2022年12月；9(1):239。65. Calvanese D, Cogrel B, Kalayci EG, Ebri SK, Kontchakov R, Lanti D 等。使用 ontop 框架进行 OBDA。第23届意大利高级数据库系统研讨会（SEBD 2015）。2015年；（1月）：296–303。66. Calvanese D, Lanti D, Mendes De Farias T, Mosca A, Xiao G. 通过以下方式访问科学数据

使用 Ontop 构建知识图谱。Patterns。2021年10月；2(10):100346。67. P. Haase、DM Herzig、A. Kozlov、A. Nikolov、J. Trame。metaphactory：知识图谱管理平台。M. Kejriwal、V. Lopez、JF Sequeda、M. Kejriwal、V. Lopez、JF Sequeda，编。语义网。2019年10月；10(6):1109–25。68. S. Moxon、H. Solbrig、D. Unni、D. Jiao、R. Bruskiewich、J. Balhoff，等。《关联数据建模语言（LinkML）：基于机器可读语义的通用数据建模框架》。CEUR研讨会论文集。2021；3073:148–51。69. P. Chakraborty、S. Dutta、DK Sanyal。个人研究知识图谱。arXiv220411428 Cs[互联网]。4月25日 2022 [，引用于5月3日 2022]；可访问：http://arxiv.org/abs/2204.11428