图数据模型与关系数据库技术

Malcolm Crowe
荣休教授，计算机科学，西苏格兰大学，佩斯利，英国 电子邮箱：
Malcolm.Crowe@uws.ac.uk

Fritz Laux 荣休教授，商务计算，罗伊特林根大学，罗伊特林根，德国 电子邮箱：
Fritz.Laux@reutlingen-university.de

摘要—近期有关数据库应用开发平台的研究尝试使用注解或属性，将概念数据模型的声明式表述纳入应用程序代码。一些近期研究利用元数据，将此类表述的详细信息纳入物理数据库。这种方法具有显著优势，因为它能够在同一数据库的多种应用中强制执行该模型。在先前的工作中，我们讨论了类型化图数据模型（TGM）对企业集成的优势。此类模型可以在图形数据库中发挥类似作用，并借助现有的统一建模语言（UML）支持。理想情况下，还应支持采用不同模型设计的系统之间的集成，例如图形数据库与关系数据库之间的集成。在这项工作中，我们使用关系数据库管理系统（DBMS）中的元数据实现了这一方法。

关键词—类型化图模型；图模式；关系数据库；实现；信息集成。

I. 引言

多年来，数据库实现过程一直包含概念数据建模阶段，通常会借助使用注解或属性的声明式结构来支持这一阶段 [1]。一些近期的数据库管理系统已将元数据纳入关系模型，以此搭建通往物理数据库的桥梁。这种方法带来了显著优势：同一数据库的所有应用程序都可以遵循统一的数据模型。在先前的工作中 [2], , 我们为 TGM 提供了映射规则，使数据模型能够在图形数据库中发挥类似作用，并使用统一建模语言（UML）的记法 [3]。在此类早期概念模型构建过程中，增量式交互探索会有所帮助 [4] , 因为自动化集成工具可能会以不恰当的方式组合内容；而使用数据类型 [5] 有助于确保语义信息不仅包含在模型中，也包含在最终数据库中。本文简要介绍了 TGM 的一种此类实现：在关系型数据库管理系统中使用元数据 [6], , 其设计部分受到 PostgreSQL 社区近期发展的启发 [7]。

与原始关系模型一样，类型化图模型（TGM）以图模式的一个实例为基础，具有严谨的数学基础。

本文的结构如下：第二节回顾 TGM，第三节讨论实现细节，并提供一个示例。第四节给出结论。

II. 类型化图模型与信息集成

我们将通过声明节点和边的实例来为数据库构建 TGM，而不是指定节点表和边表。

A. 类型化图形式体系

本节回顾 TGM 的非正式定义， [2], 其中元素（节点、边、数据类型等）用小写字母表示，元素集合用大写字母表示。集合的集合用粗体大写字母表示。一个典型示例是 n ∈ N ∈ N ⊆ ℘(N)，其中 N 是任意集合，℘(N) 是 N 的幂集。

设 T 表示简单或结构化（复杂）数据类型的集合。数据类型 t:=(l,d)∈T 具有名称 l 和定义 d。简单（预定义）类型的示例包括 (int,整数集)、(字符,ASCII)、(%,[0..100]) 等。也可以定义复杂数据类型，例如订单行 (OrderLine,(posNo,partNo,partDescription,quantity))。必须在 T 中标识这些组件，例如 (posNo,int>0)。只要所定义的结构包含有限数量的组件，就允许递归。

定义 1（类型化图模式，TGS）类型化图模式是一个元组 TGS=(NS,ES,ϱ,T,τ,C)，其中：

- NS 是具有数据类型为 t:=(l,d)∈T 的属性的命名（带标签）对象（节点）n 的集合，其中 l 是标签，d 是数据类型定义。
- ES 是具有结构化属性 p:=(l,d)∈T 的命名（带标签）边 e 的集合，其中 l 是标签，d 是数据类型定义。
- ϱ 是一个将每条边 *e* 关联到一对对象集合 (O,A) 的函数，即 ϱ(e):=(Oe,Ae)，其中 Oe,Ae∈℘(NS)。Oe 称为边 e 的尾部，Ae 称为边 e 的头部。
- τ 是一个为边 *e* 的每个节点 *n* 指定一对正整数 (in,kn) 的函数，即 τe(n):=(in,kn)，其中 in ∈N0 且 kn ∈N 。函数 τ 定义边连接的最小-最大多重性。如果最小值 in 为 0，则该连接是可选的。
- C 是一组完整性约束，图数据库必须遵守这些约束。

用于定义数据类型 T 的表示法可以自由选择；这些数据类型用于节点类型*NS*和边类型*ES*。在此实现中，标识符和表达式将使用 SQL，并搭配强类型关系数据库引擎。完整性约束*C*会对模型施加限制，超出边连接多重性的结构限制τ 。在*C*中的典型约束是对实例图内容的语义限制。

定义 2（类型化图模型）类型化图模型是一个元组 $TGM=(N,E,TGS,\varphi)$，其中：

- N 是由模式 TGS 的 NS 中数据类型构成的具名（带标签）节点 n 的集合。
- E 是由模式 TGS 的 ES 中类型的属性构成的具名（带标签）边 e 的集合。
- TGS 是如上所定义的类型化图模式。
- φ 是一个同态，它将 TGM 中的每个节点 n 和边 e 映射到 TGS 中相应的类型元素，形式化表示如下：

$$\varphi:TGM\rightarrow TGS$$
$$n\mapsto\varphi(n):=n_S(\in N_S)$$
$$e\mapsto\varphi(e):=e_S(\in E_S)$$

由于 φ 将每个元素（节点或边）映射到唯一的数据类型，因此图模型中的每个元素都有明确定义的数据类型。该同态保留结构，这意味着边类型的基数也会得到强制执行。在此实现中，TGM 的节点和边的声明会逐步构建关联的 TGS，其中也包括所隐含类型系统 T 的构建。在提交任何插入、更新或删除操作之前，都会对所有节点和边执行数据类型和约束检查。

B. 数据集成过程

信息集成要充分发挥其效益，就必须将源数据及其完整语义进行集成。我们认为，成功的关键因素之一是尽可能准确地对源和目标信息建模。TGM 的表达能力和灵活性使我们能够使用同一模型，精确描述源和目标的元数据，从而简化源与目标之间的匹配和映射。数据集成过程包括以下任务：

1) 将源建模为 TGS S_i (i = 1, 2, ..., n) 2) 将目标模式 T 建模为 TGS G 3) 将源 S_i 与 TGS G 进行匹配和映射 4) 检查并提高质量 5) 将 TGS G 再转换回 T

一旦基本模型拟定完成，第 3 步和第 4 步可以在交互式过程中同时进行。对于 EII 和其他数据集成项目而言，这样的过程至关重要，因为这些项目要求信息质量达到极高的准确性，而使用不同的映射还可以进一步提高信息质量。

要启动这一流程，可能需要从数据专家处收集结构和类型信息，或者从其他信息来源获取。如果数据来源是数据库，那么其固定的结构可作为良好的起点。否则，必须先识别相关数据，并在可能的情况下获取其元数据。这包括数据项的编码和名称。数据所有者提供的计量单位及其他元数据用于将所有度量调整到相同的尺度。Laux 的论文 [5] 举例说明了如何将关系型、面向对象和 XML 模式转换为 TGS。

如果数据源是非结构化或半结构化的，例如文档或 XML/HTML 数据，那么信息检索（IR）和统计分析中的概念与机制可能有助于识别某些隐含结构，或发现离群值及其他易受影响的数据。如果数据具有自描述性（JSON、键值对或 XML），则可借助词库或本体进行语言匹配。尽管如此，仍建议使用实例数据或咨询信息专家来验证匹配结果。我们在图 1 中用统一建模语言表示了适用于单个企业的两种可能的 TGS。这个小例子已经展示了该模型在细节层次和抽象程度方面的灵活性。

III. 在关系数据库模式中的实现

实现类型化图建模系统的前提是，关系数据库管理系统必须具备强大的类型系统。如果这一条件已经满足，那么只需对用于创建和更改结构化类型的常规 SQL 语法稍作扩展，并添加一些用于区分节点类型、边类型与其他结构化类型的元数据，就能相对简单地添加图建模功能。

那么，上文所述的图模式与大多数数据库管理系统中的模式之间，主要区别在于数据库表的列和属性具有预先定义的顺序。此外，对于给定的节点类型或边类型，只有一个基表存储该类型的实例。构建图的一种方法是在这些表中插入行。

在数据库管理系统中增加图支持，旨在简化图定义和查询任务。我们新增 CREATE 和 MATCH 语句，接下来将对此进行介绍。

A. 添加到 SQL 的面向图的语法

在常规 SQL CREATE 语法的基础上，我们增加一个用于内联构建图的选项：

创建：CREATE Node { Edge Node } {',' Node { Edge Node }}。

节点：'(' NewG | id ')'。NewG：id {':' 标签 } [doc] 。–边：'-[' NewG ']->' | '<-[' NewG ']-'。

在此语法中，单引号括起来的字符串是词元，其中包括 TGM 的几种新词元类型。在相应的源输入中，不加引号的字符串用于表示不区分大小写的标识符，双引号括起来的字符串则用于表示区分大小写的标识符，并且其中可以包含其他 Unicode 字符。与 SQL 中的常见做法一样，输入中的字符串常量将

单引号括起来，而 doc 是一种类 JSON 结构，用于提供一组属性和值表达式，其中还可能包含范围和多重性的元数据定义。

此类声明式语句会在数据库中为每个标签创建一个基表。

可以使用此语法引入节点和边，以及新的节点类型和边类型。数据库引擎会为每个不同的标签创建一个基表，其中包含足以表示相关属性的列。对于节点类型（或边类型），这些数据库基表为每个节点（或边）保存一行，其中包括节点引用。它们可以像通常那样配置索引、约束和触发器。

在常规 SQL 数据操作语言中，我们加入了 MATCH 查询语法。它的语法类似，但可以包含节点和边的未绑定标识符，以及它们的标签和/或属性。

匹配：MATCH 节点 {',' 节点 } [WhereClause] 语句。

MATCH 子句的第一部分定义了一个图表达式。如果图表达式仅包含常量值，且其所有节点和边都存在于数据库中，我们就称该图表达式已绑定。MATCH 算法沿着节点表达式进行，通过为未绑定标识符赋值，将其中越来越多的节点和边与数据库中的节点和边进行匹配。如果无法继续处理 MATCH 子句的下一部分，我们就撤销最近一次绑定并尝试其他值，以此进行回溯。如果处理到达 MATCH 语句的末尾，绑定集合就会在默认结果中贡献一行，并受可选 WHERE 条件约束。随后，这些行将作为后续语句的值来源。

B. TGM 的使用概述

按照 [5] 中的建议，我们将考虑在分析中使用 TGM，其中设想了一个交互式过程。数据库中包含的节点和边组合起来，形成一组最初为空的不相交图。向数据库添加一个节点，就会向该集合添加一个新条目。添加一条边时，其两个端点要么位于同一个图中，要么这条边会连接两个先前不相交的图。如果集合中的每个图都由一个代表节点（例如 uid 最小的节点）标识，并维护其所含节点和边的列表，那么随着数据不断添加到数据库中，就可以轻松管理这组图。

如果移除一条边，包含该边的图此时最多可能分成两部分：最简单的算法是将该图从集合中移除，然后把其中的节点和边重新添加进去。

添加了图信息的数据库可以直接用于常规数据库应用处理，其优势在于能够执行面向图的查询和面向图的存储过程。数据库引擎的常规处理会自然地强制执行模型的类型要求，也会强制执行面向图的元数据中指定的任何约束。节点和边是常规表中的行，可以使用常规 SQL 语句访问和完善。尤其是，使用常见的点式语法，可以对属性进行 SET 和更新，也可以将其设为 NULL 来移除。

随着 TGM 的开发并与其他图形数据合并，系统将检测到冲突，诊断信息也将有助于找出整合模型新部分时遇到的障碍，从而完善截至当时已开发的模型。

C. 一个示例：入门

客户-供应商订购系统可能会包含一些有问题的 CREATE 语句，例如：

```
CREATE
(Joe:客户 {"Name":'Joe Edwards', Address:'10 Station Rd.'}), (Joe)-[:已订购 {"日期":date'22/11/2002'} ]->
(Ord201:"订单")-[:商品项 {数量: 5}]->("16/50x100" : 木螺钉),
(Ord201)-[:商品项 {数量: 5}]->("12厘米纤维": 墙塞),
(Ord201)-[:商品项 {数量: 1}]->("500ml" : 橡胶胶水)
```

此处将边的主键留给引擎生成；如果愿意，也可以显式指定。Name、Order 和 Date 用双引号括起来，因为它们是 SQL 中的保留字。默认情况下，数据库引擎会将所示的整个 CREATE 语句视为一个事务：如果语法检查器检查无误，该语句就会自动提交。

这里很容易批评用户提供的内容：图可以通过拆分“Fibre 12cm”和“16/8x100”之类的复合信息而受益，这样既能阐明各组件的含义，也便于处理。设计者之后可以进行这些修改。

假设开始之前数据库为空，那么如果提交上面的第一行，就会创建一个新的基表客户（一个节点类型）。

创建类型 客户 为（"名称" 字符，地址 字符）节点类型

节点类型元数据标志会将一个类型为字符的主键列标识符添加为第一列，这样新的客户表就会有一行初始记录。

```
('JOE','Joe Edwards','10 Station Rd.')
```

这样就可以了。下一行定义了另外四个基表：两个节点类型和两个边类型：

CREATE TYPE "订单" 节点类型 CREATE TYPE 木螺钉 节点类型 CREATE TYPE 已订购 as (" 日期 " 日期) 边类型 (客户,"订单") 创建类型 商品项 as (数量 整数) 边类型("订单",木螺钉)

这样也能行，但这可能不是分析师想要的结果，因为 Item 边类型连接的是木螺钉类型的节点。如果提交了这个定义，我们之后就无法再创建连接到墙塞的 Item 边。

但目前尚未提交，因此数据库引擎发现这个问题时，只需将第二行中的规范`:WoodScrew` 替换为`:WoodScrew:&1` ，并对 WallPlug 和 RubberGlue 做类似的修改。

这会为这些节点类型添加一个新的匿名基节点类型，并为其生成一个系统名称

```
创建类型 &1 节点类型
```

于是，节点类型提案变为

```
CREATE TYPE 商品项 as (数量 整数) 边类型("
订单",木螺钉)

CREATE TYPE 木螺钉 UNDER &1 CREATE
TYPE 墙塞 UNDER &1 CREATE TYPE 橡胶胶水
UNDER &1
```

可以告知分析师，已完成此操作，之后他们可以为新的节点类型 &1 另选一个更合适的名称（例如 PRODUCT？）。这种泛化过程可以作为标准的数据库转换提供。

生成节点和边并提交事务后，节点和边的数据将按如下方式安装到数据库中：

```
客户（'Joe','Joe Edwards', '10 Station Rd.'）
“订单”（'Ord201'）木螺钉（'16/50x100'）
墙塞（'Fiber 12cm'）橡胶胶水（'500ml'）已
订购（'&2','Joe','Ord201', '日期'
22/11/2002'）商品项（'&
3','Ord201','16/50x100'）（'&
4','Ord201','Fiber 12cm'）（'&
5','Ord201','500ml'）
```

这样处理相当简洁。我们看到，元数据标志节点类型为节点类型添加了一个主键，作为第一列的标识符；而元数据标志边类型则为边类型添加了三个初始列：标识符（主键）、离开节点（指向离开节点类型的外键）以及到达节点（指向到达节点类型的外键）。还要注意，商品项的到达类型是新的匿名类型 &1。

值得注意的是，正如[2]中所设想的，这一机制允许模式在数据库设计过程中自底向上演进。通常的模式优先策略仍然可用，而且为方便起见，这两种方法也可以结合使用。无论采用哪种方式，数据库在各个阶段都将包含严格且可强制执行的关系模式，因为任何无法强制执行的声明都会在提交到数据库之前被拒绝。

在模型的细化过程中，可以添加约束和其他元数据。这些细节以及前文提到的增强型诊断信息，都是当前研究的主题。会议报告将提供演示这一流程的机会，并进一步介绍 MATCH。

IV. 结论

本文旨在报告我们在类型化图建模工作流中取得的一些进展。该项目可在 GitHub [8] 免费下载和使用，且不受任何专利或其他限制。

该软件目前处于“alpha”阶段，实现了上述所有构想，但缺少与模型设计者进行建议中的交互。测试套件包含运行示例的一个版本，以及其他示例，用于展示关系型模型和类型化图模型概念在 Pyrrho 数据库管理系统中的集成。

参考文献

[1] 甲骨文，《Oracle 产品文档》（在线），网址：https://docs.oracle.com/javaee/7/tutorial/persistence- intro.htm#BNBPZ [检索日期：Feb, 2023][2] F. Laux 和 M. Crowe，“使用类型化图模型进行信息集成”，DBKDA 2021：第十三届数据库、知识与数据应用进展国际会议，IARIA，2021 年 5 月，第 7- 14 页，ISSN：2308-4332，ISBN：978-1-61208-857-0[3] E. J. Naiburg 和 R. A. Maksimschuk，《面向数据库设计的统一建模语言》。Addison-Wesley Professional，2001[4] R. De Virgilio、A. Maccioni、A.、R. Torloner，“图数据库的模型驱动设计”，载于 Yue、E. 等（编）《概念建模》，33rd 国际会议（ER 2014），施普林格，2014 年 10 月，第 172-185 页，ISSN：0302-9743，ISBN：978-3-319- 12205-2[5] F. Laux，“类型化图模型”，DBKDA 2020：第十二届数据库、知识与数据应用进展国际会议，IARIA，2020 年 9 月，第 13-19 页，ISSN：2308-4332，ISBN：978-1-61208-790-0[6] M. Crowe 和 F. Laux，“数据库技术演进”，IARIA《高级软件国际期刊》，第 15 卷（3-4），2022 年，第 224-234 页，ISSN：1942-2628[7] S. Shah 等，《PostgreSQL 数据计算平台》（PgDCP）（在线），网址：https://github.com/netspective-studios/PgDCP [检索日期：Feb 2023][8] M. Crowe，PyrrhoV7alpha，https://github.com/MalcolmCrowe/ShareableDataStructures [检索日期：Feb 2023]

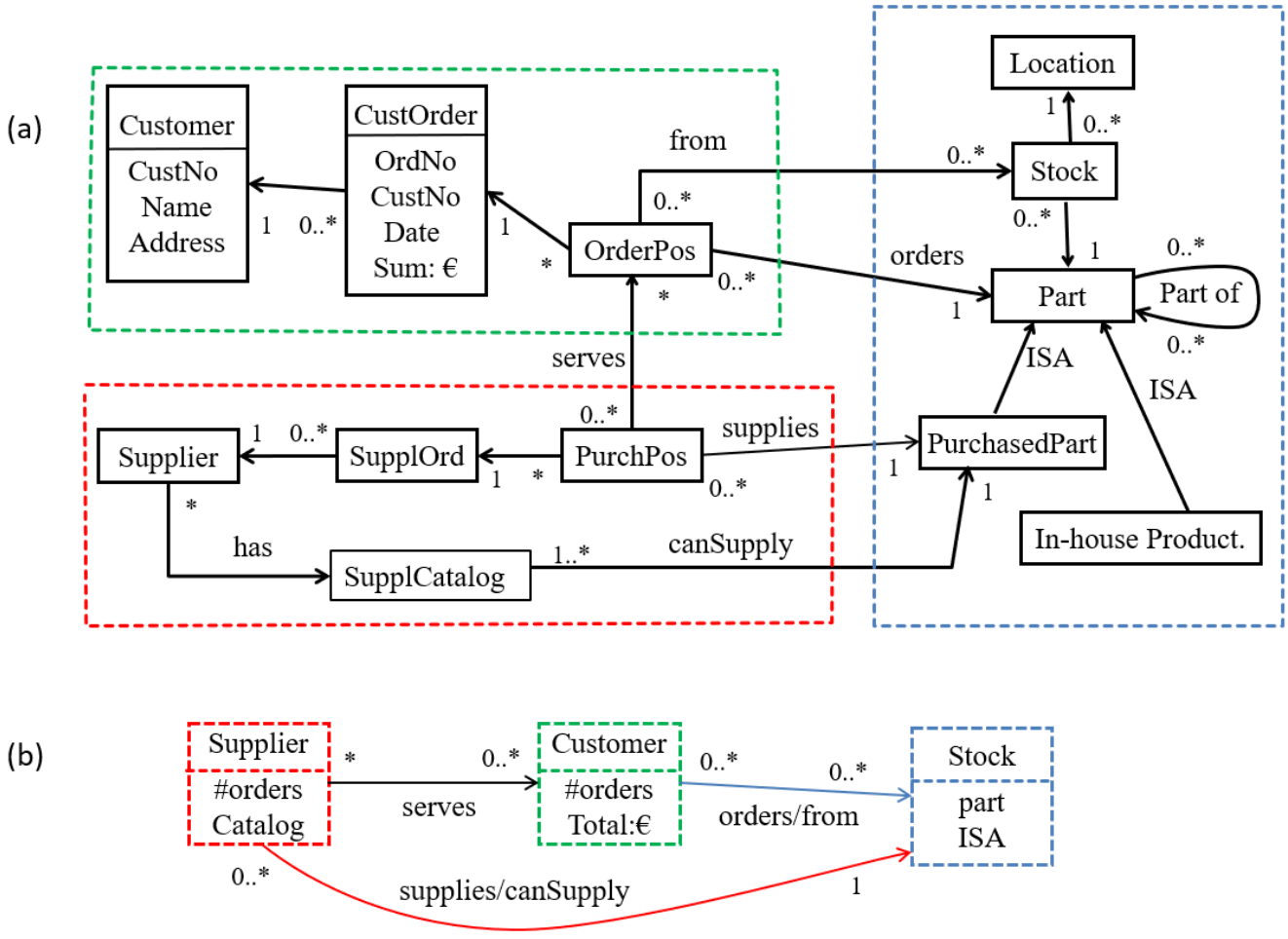


图 1. 展示两个详细层级的商业企业 TGM 示例