doi:10.11887/j.cn.202503005 http://journal.nudt.edu.cn

S-Cypher:时态属性图模型上的时态图查询语言

蒋甜甜[1,2],陈观林[1,2],宋明黎[1],杭海天[1],王豪烨[2*]
(1. 浙江大学 计算机科学与技术学院,浙江 杭州 310027;2. 浙大城市学院 计算机与计算科学学院,浙江 杭州 310015)

摘 要:传统的图数据模型未考虑时间维度,可能会导致时态查询极其复杂,甚至破坏时间信息的完整性,为此,提出了一种时态属性图数据模型和相应的时态图查询语言S-Cypher。该时态图数据模型使用对象节点表示实体,引入属性节点和值节点表示实体的属性,在节点以及对象节点之间的边上记录有效时间以表达时态信息,其记录的有效时间均遵循一组时态约束。S-Cypher是Cypher的时态拓展,在保证兼容的同时不仅提供了一套简洁完善的时态图查询语法,包括时态数据类型、时态图模式匹配、时间窗口限定和时态路径;还提供了一套在Neo4j上进行S-Cypher时态图查询的实现方案。实验结果显示,S-Cypher的查询时间平均是Cypher的1.29倍,表明S-Cypher能够有效地管理Neo4j中的时态图数据,并具有良好的性能。

关键词:时态图;图数据模型;图查询语言
中图分类号:TP311 **文献标志码**:A **文章编号**:1001-2486(2025)03-041-10

论文拓展

S-Cypher: temporal query language on the temporal property graph model

JIANG Tiantian[1,2], *CHEN Guanlin*[1,2], *SONG Mingli*[1], *HANG Haitian*[1], *WANG Haoye*[2*]
(1. College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China;
2. School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China)

Abstract: Traditional graph data models lack explicit temporal dimension representation, which may lead to complex temporal queries and potential loss of temporal information integrity. To address this limitation, a temporal property graph data model and a corresponding temporal graph query language called S-Cypher were proposed. The temporal graph data model represents utilized object nodes to represent entities, and introduced property nodes and value nodes to represent entity properties. Valid time was recorded on nodes and edges between object nodes to express temporal information, and the recorded valid time adhered a set of temporal constraints. S-Cypher served as a temporal extension to Cypher, ensured compatibility while providing a concise and comprehensive temporal graph query syntax, including temporal data types, temporal graph pattern matching, time window constraints, and temporal paths. An implementation scheme for executing S-Cypher temporal graph queries on Neo4j was also provided. Experimental results demonstrate that the query time of S-Cypher is on average 1.29 times that of Cypher, indicating that S-Cypher can effectively manage temporal graph data in Neo4j with satisfactory performance.

Keywords: temporal graph; graph data model; graph query language

在数字化时代,各行各业都在持续地产生和积累大量数据,时态数据作为其中一类重要资源而日益受到关注。时态数据是指随着时间变化而产生的数据。无论是金融行业的股票交易[1],物联网的传感器监测[2],还是社交媒体的用户行为分析[3-4],时态数据都扮演着不可或缺的角色。这些数据中的时态信息反映着数据之间的时间关系和数据的演化趋势,具有深入分析挖掘的价值。

由于关系数据库的广泛应用,传统的时态图数据模型主要基于关系数据模型。但对于关系复

收稿日期:2024-11-07
基金项目:国家自然基金联合基金重点资助项目(U20B2066);国家自然科学基金资助项目(62302430);浙江省自然科学基金资助项目(LQ24F020017)
第一作者:蒋甜甜(1999—),女,浙江丽水人,硕士研究生,E-mail:jiangtiantian@zju.edu.cn
*通信作者:王豪烨(1994—),男,浙江宁波人,讲师,博士,E-mail:wanghaoye@hzcu.edu.cn
引用格式:蒋甜甜,陈观林,宋明黎,等. S-Cypher:时态属性图模型上的时态图查询语言[J]. 国防科技大学学报,2025,47(3):41-50.
Citation:JIANG T T, CHEN G L, SONG M L, et al. S-Cypher: temporal query language on the temporal property graph model[J]. Journal of National University of Defense Technology, 2025, 47(3): 41-50.

杂的数据场景,关系数据模型的性能较差。越来越多的研究表明,在以关系为导向的应用程序中,图数据模型可以提供更好的可解释性、可拓展性和性能。然而,传统的图数据模型并没有专为时间信息设计的结构。虽然可以直接在传统的图数据模型上存储简单的时间信息(例如,往节点中添加一系列的时间点属性,以代表该节点及其各个属性的有效时间),但这种方式无法存储节点的同一属性在不同时间下的属性值,也很容易破坏时间信息的完整性。另一种方式是使用时态图数据模型。时态图数据模型提供专门的数据结构存储时间信息,有效解决了以上问题。但以时态图数据模型存储时间信息后,图查询将会变得极其复杂,且很容易破坏时态约束,此时一个相应的时态图查询语言是必不可少的。

本文提出了一种基于属性图的时态数据模型及其查询语言 S-Cypher。该模型通过对象节点、属性节点与值节点构建时态图结构,在节点与边上记录遵循时态约束规则的有效时间,确保时态逻辑的完整性与一致性。S-Cypher 作为 Cypher 的时态扩展语言,新增时态数据类型、时态图模式匹配、时间窗口限定、时态路径查询及严格遵循约束的更新语法,兼顾兼容性与表达能力。为实现实际应用,设计了时态数据在 Neo4j 等图数据库中的存储建模规则,并提供基于时间函数插件的 S-Cypher 查询实现方案,支持在原生 Neo4j 环境中执行时态操作。

1 相关工作

1.1 时态数据模型及查询语言

20 世纪八九十年代,时态数据库得到了蓬勃发展,产生了大量研究成果,并已进入标准化和产品化阶段。虽然时态数据模型主要基于关系数据模型,但其中的许多基础理论成果也在时态图数据库中得到借鉴和应用,包括以下几个重要概念:

时间域(time domain):在自然界中,时间每时每刻都存在,可以被看作一个连续的实数域。考虑到计算机的特性,一般以离散形式来表示时间,可以形式化地记为一个 Ω 的线性域。Ω 是线性的,并通过有序的自然数进行离散化。

时间点(point)和时间区间(interval):时间点指 Ω 上的一个点,时间区间指一段时间,由开始时间和结束时间组成。其中,开始时间和结束时间都是一个时间点。

时间粒度(time granularity):指描述时间数据的最小单位,反映了时间域的离散程度。

有效时间(valid time):指一个对象(事件)在现实世界中发生并保持的那段时间,可以是单一的时间点、单一的时间区间、时间点的集合或时间区间的集合,甚至也可以是整个时间域。一个相近的概念是事务时间(transaction time),指一个数据库对象在数据库中进行操作的时间。本文旨在研究有效时间的管理,事务时间相关的版本化机制并不在讨论范围内。

NOW:*NOW* 是时态数据库中一个特殊的时间元素,指当前时间点,是一个时间变元,随着当前时间的变化而变化。

TempSQL[5]、TQuel[6]和 TSQL2[7]是三个具有代表性的时态查询语言,均是对关系数据库查询语言 SQL 或 Quel 的时态拓展。作为查询语言,它们大致等于元组演算,借助有效时间代数进行时态查询。S-Cypher 与这些时态查询语言有着共通之处,都支持了 Allen 时态区间关系[8]等时间代数。但由于 S-Cypher 本质上是一种图查询语言,因此还包括了对 *Match* 等图查询子句的时态拓展。

1.2 图数据模型及查询语言

目前有两种主流的图数据模型:属性图[9]和资源描述框架[10](resource description framework, RDF),通常选择在其一的基础上进行时态拓展。属性图由 Neo4j 推广并流行开来,可能是当下最流行的图数据模型。在属性图中,节点和关系都可以拥有属性。属性图用节点表示实体,用关系表示实体之间的连接,用属性描述节点和关系的特征,能够以灵活而直观的方式存储和表达图形数据。RDF 由三元组构成,每个三元组构成一个“主谓宾”的陈述,在图中表示为两个相连的节点和连接边。RDF 图适合构造复杂的知识图谱、语义网和推理系统,但其查询和遍历的效率相对较低,尤其是在面对大规模数据时。

经过长期发展,在属性图和 RDF 的基础上产生了多种成熟的图查询语言,如 Cypher[11]、SPARQL[12]、Gremlin[13]等。其中 Cypher 是当下最流行的图查询语言,也是事实上的标准,是 S-Cypher 进行时态拓展的基础。

1.3 时态图数据模型及查询语言

目前已有数十种时态 RDF 数据模型[14-16]被提出,也有大量时态 RDF 查询语言[15,17-18]被设计出来。但不同于时态 RDF 的高度发展和完善,属性图的时态化工作还较少。

根据标记时间信息的方式,时态属性图数据

模型主要分为两种:标记有效时间(valid-time-labeled)的时态属性图数据模型[19-21]和基于快照(snapshot-based)的时态属性图数据模型[22-24]。前者在节点、边性等图元素上标记有效时间以管理图数据的时态信息,后者使用一系列快照和增量更新序列存储时态图数据。快照数据库的优势在于能够快速获得某个时间点的大量数据,但具有一些局限性,比如以全局方式捕获快照不允许直接跟踪单个对象的变化;在分析信息随时间扩散的问题上,快照方法会导致错误的时态路径[25]。考虑到快照数据库的局限性,本文使用标记有效时间的时态属性图数据模型。

目前的时态属性图查询语言主要基于 SQL 或现有图查询语言进行时态拓展。Campos 等[20]提出了 TEG-QL,其语法类似于 SQL,可以方便地转化为 Cypher,但仅能使用 Snapshot 和 In 来限制查询有效时间在某个时间点或某个时间区间的节点和边。Debrouvier 等[21]介绍了 T-GQL,T-GQL 使用了一种 SQL 和 Cypher 的混合语法,并将重点放在了时态路径查询。Rost 等[26]报告了基于属性图查询语言(property graph query language, PGQL)的时序图查询语言 T-PGQL,这是一种将图模式匹配与类似 SQL 的语法和功能结合的图查询语言。

现有时态图查询语言的语法大多不完善且较为粗糙,有较大改善空间,且尚没有时态图数据语言对 Cypher 进行了良好的时态拓展。为此,本文设计了一种兼容 Cypher 且语法完善的时态图查询语言。

2　时态图数据模型

2.1　时态属性图

近年来,Debrouvier 等[21]提出了一个有影响力的基于属性图的时态图数据模型。该时态图数据模型将节点分为了对象节点、属性节点和值节点,分别代表实体、实体的属性键和实体的属性值。每个节点以及对象节点之间的边都携带一个有效时间。其中,属性节点可以在不连接到任何值节点的情况下存在,其有效时间由用户设置,需要包含其相连的所有值节点有效时间,但不做其他约束。这带来了一定程度的灵活性:可以单独设置对象节点的属性,而不用设置属性值,属性节点有效时间的设置也更加自由。但这却带来了更多的问题:

1)节点属性相关操作的复杂度增加。用户需要经常性地考虑属性节点的有效时间问题。例如,在删除值节点时,必须决定是否要更改属性节点的有效时间,甚至删除属性节点;同样,在添加值节点和更新值节点的有效时间时,也需要考虑是否要修改属性节点的有效时间。

2)属性节点和值节点之间依然存在着约束关系,这与属性节点的独立性矛盾。例如,用户先将某个属性节点的有效时间设置为[2000 年, 2010 年),又在 2015 年为该属性赋值,即添加一个有效时间为[2015 年, *NOW*)的值节点,此时系统将难以决定应该报错,还是应该修改属性节点的有效时间。系统难以判断属性节点的有效时间是否有意义,但报错又会限制用户操作。

3)属性节点的有效时间存在语义问题。属性节点的有效时间表示该属性在现实中存在的时间域,然而,若属性在某个时间域内没有属性值,该属性也就失去了其存在的实际意义。现实中不存在具有属性但没有属性值的情况。即使要记录属性值不明的情况,也可以直接将属性值设置为 *Unknow*。

为了解决这些局限性,本文改进了模型,使属性节点不能独立存在,且将属性节点的有效时间确定为其所有值节点有效时间的并集。定义 1 给出了改进的模型。

定义 1　时态属性图 G 可以表示为(N_o, N_a, N_v, E_o, E_p, E_v),其中,N_o、N_a 和 N_v 分别为对象节点、属性节点和值节点的集合,E_o、E_p 和 E_v 分别为对象节点之间的边、从对象节点到属性节点之间的边,以及从属性节点到值节点之间的边的集合。每个节点和每条边都包含一个内容(*content*)和一个有效时间(*VT*)。

图 1 为一个时态属性图示例。时态属性图用对象节点表示实体,用属性节点和值节点分别表示实体的属性键和属性值。对象节点之间的边表示实体间的关系,从对象节点到属性节点之间的边表示属性和实体之间的从属关系,从属性节点

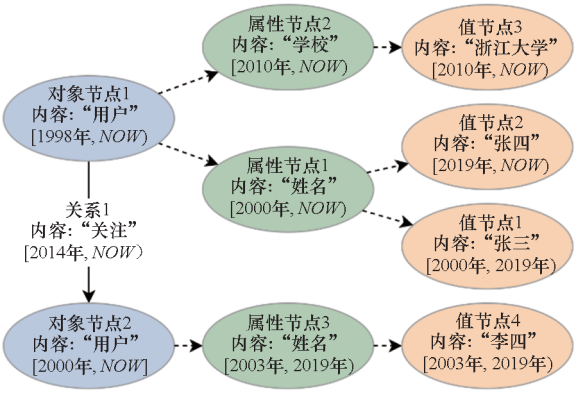


图 1　时态属性图示例

Fig. 1　An example of temporal property graph

到值节点的边表示属性值和属性键之间的从属关系。每个对象节点可以拥有多个属性,因此每个对象节点可以连接多个属性节点。同样地,每个属性键可以拥有多个不同时间下的属性值,因此每个属性节点也可以连接多个值节点。但反过来,每个属性节点和值节点分别只能连接一个对象节点和属性节点。

对象节点、属性节点、值节点和对象节点之间的边的内容分别表示实体标签、实体属性名、实体属性值和实体关系类型,它们的有效时间则表示所代表的实体、实体的属性键、实体的属性值和实体的关系在现实世界中存在或发生并保持的那段时间。有效时间由一个不相交的非空时间区间集组成,其中,每个时间区间由两个时间点——开始时间和结束时间构成。

2.2 约束

为保证时态数据的完整性,时态图数据库在数据更新操作期间,还必须要满足时态图数据模型的约束如下所示。其中,$id(n)$表示 n 的标识号,$e\{n_1,n_2\}$表示从 n_1 到 n_2 的边,$E=E_o\cup E_p\cup E_v$。$\forall n_1,n_2\in N_o\cup N_p\cup N_v\Rightarrow n_1=n_2\vee id(n_1)\neq id(n_2)$;$\forall n_1\in N_o,e\{n_2,n_1\}\in E\Rightarrow n_2\in N_o$;$\forall n_1\in N_p,\exists e\{n_2,n_1\}\in E,n_2\in N_o$;$\forall n_1\in N_p,e\{n_2,n_1\}\in E\Rightarrow n_2\in N_o$;$\forall n_1\in N_v,\exists e\{n_2,n_1\}\in E,n_2\in N_p$;$\forall n_1\in N_v,e\{n_2,n_1\}\in E\Rightarrow n_2\in N_p$;$\forall n_1\in N_p,\exists e\{n_1,n_2\},n_2\in N_v$;$\forall n_1,n_2\in N_o,e\{n_2,n_1\}\Rightarrow e\cdot VT\in n_1\cdot VT\cap n_2\cdot VT$;$\forall n_1\in N_o,n_2\in N_p,e\{n_2,n_1\}\Rightarrow n_2\cdot VT\in n_1\cdot VT$;$\forall n\in N_p\Rightarrow n\cdot VT=\cup_{n_i\in N_v,e\{n,n_i\}}n_i\cdot VT$;$\forall n_1,n_2\in N_o,e_1\{n_1,n_2\},e_2\{n_1,n_2\}\in E,e_1\cdot content=e_2\cdot content\Rightarrow e_1=e_2\vee e_1\cdot VT\cap e_2\cdot VT=\varnothing$;$\forall n_1,n_2\in N_v,n_3\in N_p,e_1\{n_3,n_1\},e_2\{n_3,n_2\}\in E\Rightarrow n_1=n_2\vee n_1\cdot VT\cap n_2\cdot VT=\varnothing$;$\forall n_1,n_2\in N_p,n_3\in N_o,e_1\{n_3,n_1\},e_2\{n_3,n_2\}\in E\Rightarrow n_1=n_2\vee n_1\cdot content\neq n_2\cdot conent$。

2.3 建模规则

本节提出了一组将以时态属性图形式组织起来的数据存储至 Neo4j 等属性图数据库的建模规则,具体如下:

1)对象节点、属性节点和值节点均使用节点进行建模,并分别赋予标签 *Object*、*Property* 和 *Value*。对象节点的内容由0到多个节点标签表示,属性节点和值节点的内容则由属性 *content* 表示。节点的有效时间由属性 *startTimes* 和 *endTimes* 共同表示。

2)对象节点和属性节点之间的边由边类型为 *Object_Property* 的边进行建模,属性节点和值节点之间的边则由边类型为 *Property_Value* 的边进行建模,这两类边均不拥有属性。虽然这两类边存储在数据库中,但其作用仅仅是确认属性节点和值节点的归属关系,对用户来说不可见,用户无法直接对其进行操作。

3)对象节点和对象节点之间的边使用边进行建模,其内容由边类型表示,有效时间由属性 *startTimes* 和 *endTimes* 共同表示。用户可以为其另外设置0到多个静态属性。

将时态图数据以时态属性图形式存储至 Neo4j 图数据库后,就可以借助 Neo4j 图数据库所提供的成熟图管理方法来管理,具体的实现方案见第4节。

3 S-Cypher

时态图查询语言 S-Cypher 可以看作 Cypher 的时态扩展语言。Cypher 的基础语法详见 Cypher 说明手册[11],本节旨在对 S-Cypher 与 Cypher 的区别,即时态扩展部分进行说明。S-Cypher 的时态拓展主要包括:

1)添加时态数据类型——时间区间、一系列时间代数,以及对特殊时间元素 *NOW* 的支持。

2)在 Cypher 的 *Match* 子句上添加@T 的相关语法,以便于进行时态图模式匹配。

3)支持3种时态路径的查询语法和4种时间窗口的限定语法。

4)添加 *Stale* 过时子句用法,并拓展 Cypher 的 *Set*、*Delete*、*Create* 子句对时态属性图的更新操作支持。

3.1 时态数据类型

S-Cypher 所使用的时态数据类型包括时间点、时间区间(interval)和时间段(duration)。其中时间点和时间段使用了 Cypher 所定义的5种时间点类型和时间段类型。每个时态数据库在建立时,都必须使用其中一种时间点类型来指定数据库的时间粒度,之后所设置的有效时间中的时间点均为该类型。

时间区间为 Map 类型,由两个相同类型的时间点——开始时间和结束时间构成,采用前端封闭、尾端开放的形式。

Cypher 提供时间点与时间段之间的算术运算和比较运算,以及一系列时间点函数和时间段函数。S-Cypher 除了支持 Cypher 内置的时间算法,还支持 Allen 的13种时态区间关系[8],以及几个常见的时间运算,包括 Coalesce、Intersect 和

Except,分别用于处理缺失时间值、计算多个时间区间的交集和计算两个时间区间的差集。

在 S-Cypher 中,*NOW* 有两种使用场景:一种是在时态查询语句中,匹配在当前时间有效的元素,此时转换器会将 *NOW* 用当前时间点代替。另一种是在时态更新语句中用 *NOW* 记录元素的结束时间,表示该元素在开始时间之后一直有效,此时转换器会将 *NOW* 用相应时间点类型的最大值替代,并存储在数据库中。

3.2　时态查询语法

时态图查询与传统图查询的区别在于时态图查询可将时态信息作为查询的限制条件。本节展示了时态查询语法中限制时态信息的多种方式。

3.2.1　时态图模式匹配

时态图模式匹配在传统图模式匹配的基础上,加入了时态条件的限制。和 Cypher 中的图模式一样,时态图模式也可以在 *Match* 子句中使用。

携带有效时间的时态图元素包括对象节点、属性节点、值节点以及对象节点之间的边。在时态图模式匹配中,可以使用@ T($<start>$, $<end>$)和@ T($<timing>$)对这些时态图元素的有效时间进行限制,其中,$<start>$、$<to>$和 $<timing>$的时间点类型与数据库的时间点类型一致。假设某元素的有效时间为 T,并在时态图模式中使用@ $T(t_1, t_2)$(或@ $T(t_1)$)对其有效时间进行限制,那么仅当$[t_1, t_2) \subseteq T$(或 $t_1 \subseteq T$)时,该元素满足匹配条件。

对于已匹配到的时态图元素,可以通过特殊操作符@ T 查询它们的有效时间,还可以进一步在 *Where* 子句中对时态图元素的有效时间进行更加复杂的限制。实体在不同时刻可能拥有不同的属性值,在访问实体的属性值时,可以使用#$T(t_1, t_2)$或#$T(t_1)$限制值节点的有效时间。时态信息查询的相关语法如表 1 所示。

表 1　时态信息的查询语法

Tab. 1　Query syntax of temporal information

查询方法	说明	示例
objectNode@ T	获取对象节点的有效时间	*n*@ T
relationship@ T	获取对象节点之间边的有效时间	*e*@ T
objectNode. attrName@ T	获取属性节点的有效时间	*n. name*@ T
objectNode. attrName#*Value* @ T	获取在默认时间区间、时间点上有效的值节点的有效时间,若有多个值节点,返回一个有效时间的列表	*n. name*#*Value*@ T
objectNode. attrName#$T(t_1[, t_2]^*)$@ T	获取在指定时间区间、时间点上有效的值节点的有效时间,若有多个值节点,返回一个有效时间的列表	*n. name*#T("2023")@ T, *n. name*#T("2021", "2023")@ T
objectNode. attrName	获取在默认时间区间、时间点上有效的值节点的内容,若有多个值节点,返回一个内容列表	*n. name*
objectNode. attrName#$T(t_1[, t_2]^*)$	获取在指定时间区间、时间点上有效的值节点的内容,若有多个值节点,返回一个内容列表	*n. name*#T("2023"), *n. name*#T("2021", "2023")

注:[]表示可选。

3.2.2　时态路径语义

在非时态图中,如果两个节点之间存在路径,则认为这两个节点是连通的,然而,这种说法并不适用于时态图。在时态图中,常见的有三种时态路径:连续有效路径[27]、成对连续路径[21]和顺序有效路径[21],在时态上的连续性有着不同的定义。时态路径可由元组$(n_{i(1)}, e_{j(1)}(T_1), n_{i(1)}, e_{j(2)}(T_2), \cdots, e_{j(k-1)}(T_{k-1}), n_{i(k)})$表示,其中,$n_i$ 表示 *id* 为 i 的对象节点,e_j 表示 *id* 为 j 的边,T_k 表示 $e_{j(k-1)}$的有效时间,$(n_{i(1)}, \cdots, n_{i(k)})$表示时态路径所遍历的对象节点,$(e_{j(1)}, \cdots, e_{j(k-1)})$表示时态路径所遍历的边。

连续有效路径中的边满足 $T = \bigcap_{i=1}^{k-1} T_i \neq \varnothing, i = 1, \cdots, k-2$,即路径中的所有边都同时在某个非空时间区间集 T 内有效。

成对连续路径中的边满足 $T_i \cap T_{i+1} \neq \varnothing$,即路径中的任意两条相邻边都在某个非空时间区间内同时有效。

顺序有效路径中的边满足 $T_i \cdot end \leqslant T_{i+1} \cdot start(i = 1, \cdots, k-2)$,即路径中边的有效时间呈单调递增分布。顺序有效路径适合应用在航班、

通信等调度问题上,可以与最早到达路径、最迟出发路径、最快路径和最短路径四种语义结合。

在 S-Cypher 中查询这些时态路径时,可以限制出发节点、目的节点、关系类型、路径长度,以及路径元素的有效时间。

3.2.3 时间窗口限定

除了在时态图模式中限制元素的有效时间,也可以使用 *At Time* 和 *Between*、*Snapshot* 和 *Scope* 限定子句的有效时间,从而简便时态查询。对于时态图查询语法,限制有效时间的优先级为:时态图模式匹配 > *At Time* = *Between* > *Scope* > *Snapshot*。

At Time 子句和 *Between* 子句仅作用于某一条子句,分别包含一个时间点和时间区间,不能对同一条子句使用。*At Time* 子句必须和 *Match* 子句或更新子句搭配使用,用于限制元素的有效时间或声明更新的操作时间。*Between* 子句必须和 *Match* 子句或 *Delete* 子句搭配使用,用于限制元素的有效时间。

Snapshot 子句和 *Scope* 子句是单独调用的,作用于当前会话下的所有语句,分别包含一个时间点和时间区间。*Snapshot* 子句可对所有语句生效,用于限制元素的有效时间或声明更新的操作时间。*Scope* 子句仅对 *Match* 子句和 *Delete* 子句生效,用于限制元素的有效时间。当会话同时使用了 *Snapshot* 和 *Scope* 时,时态查询语句和 *Delete* 子句优先根据 *Scope* 子句指定时间区间进行匹配,而时态更新语法(除 *Delete* 操作)忽略 *Scope* 子句,仅根据 *Snapshot* 子句指定的时间点进行匹配。

在不使用时间窗口子句时,时态图查询语句仅使用时态图匹配对查询图进行时态限制,时态图更新语句则默认使用当前时刻作为操作时间。

3.3 时态更新语法

S-Cypher 的时态更新语法包括对实体、实体关系和实体属性的创建、删除、过时和修改。相对于传统图更新操作,在对时态属性图做更新操作时,还需要处理节点和对象节点之间边的有效时间,并保证它们在更新前后均满足时态属性图的约束。

3.3.1 时态创建语法

时态创建语法包括对实体、实体关系和实体属性的创建,其中,实体和实体关系的创建使用 *Create* 子句实现,而实体属性则使用 *Set* 子句实现。

创建实体和实体关系时,分别创建一个对应的对象节点和边,其中边的有效时间必须落在其出点和入点的有效时间内,且与同出点、入点和内容的其他关系的有效时间没有重合区间。

创建实体属性时,在该实体对应的对象节点下创建属性节点、值节点和相连边。若已创建过相同属性名的属性,则不再新创建属性节点,而是在之前所创建的属性节点下新创建一个值节点。其中,值节点的有效时间不能与同属性节点下的其他值节点的有效时间重合;属性节点的有效时间必须落在所属对象节点的有效时间内。

假设操作时间为 t,那么新创建的对象节点、边和值节点的有效时间默认为$[t, NOW)$,属性节点的有效时间为相连值节点的有效时间的并集。也可以通过指定操作时间或直接指定所创建的对象节点、边和值节点的有效时间,前提是指定的有效时间满足时态约束。

3.3.2 删除语法

时态删除语法包括对实体、实体关系和实体属性的删除,使用 *Delete* 子句实现。

物理删除实体时,物理删除对应对象节点及其相连的属性节点、值节点和边;物理删除实体关系时,物理删除对应边;物理删除实体属性时,物理删除对应的属性节点、值节点和相连边,若属性节点不再有相连的值节点,属性节点也会一并删除。

3.3.3 过时语法

过时语法包括对实体、实体属性和实体关系的过时,使用 *Stale* 子句实现。过时操作将元素的结束时间由 *NOW* 改为操作时间,表示该元素不再实时有效。显然,过时结束时间只能为 *NOW* 的元素。

过时实体时,过时对应对象节点及其相连的边、属性节点和值节点;过时实体关系时,过时对应边;过时实体属性时,过时对应的属性节点和值节点。

假设操作时间为 t,那么所过时的节点或边的结束时间将由 *NOW* 修改为 t。t 可以由用户指定,但必须迟于开始时间。

3.3.4 时态修改语法

时态修改语法包括对对象节点、对象节点之间的边、值节点的内容、有效时间的修改,以及对实体属性的修改,均使用 *Set* 子句实现。

修改对象节点、边和值节点的内容时,分别修改对象节点的标签,边类型和值节点的 *content* 属性。其中,在修改边的内容时,需确认修改之后的边与同出点、入点和内容的其他边没有重合的有效时间。

修改对象节点、边和值节点的有效时间时,分

别修改对应节点和边的 *startTimes* 和 *endTimes* 属性,必须确保修改后:

1) 边的有效时间同时落在入点和出点的有效时间内,且与同出点、入点和内容的其他边的有效时间没有重合区间。

2) 值节点的有效时间与同属性节点下的其他值节点的有效时间没有重合区间。

3) 随值节点改变的属性节点的有效时间落在所属对象节点的有效时间内。

实体属性的修改操作可以看作过时操作和创建操作的结合,即先过时在操作时间下有效的值节点,再创建一个内容为指定属性值的值节点。

4　实现

为了利用 Neo4j 的已有技术来管理时态图数据,本节设计了一套将 S-Cypher 查询语句转换为 Cypher 查询语句,并运行在 Neo4j 图数据库上的实现方案,如图 2 所示。

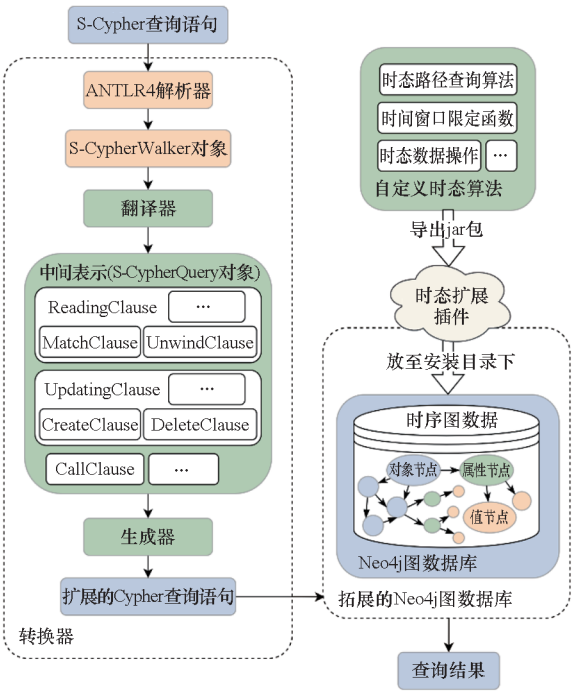


图 2　在 Neo4j 上运行 S-Cypher 语句的实现方案

Fig. 2　Implementation scheme for running S-Cypher statements on Neo4j

4.1　转换器

转换器的作用是将 S-Cypher 查询语句转换为 Cypher 查询语句,从而在 Neo4j 图数据库上查询时态图数据。具体流程如下:

1)使用 ANTLR4 解析时态图查询语句,为其生成语法解析树。ANTLR4[28]是一个开源的语法分析工具,已有 30 余年的历史,能够稳定高效地进行词法分析和语法分析。

2)生成 S-CypherWalker 对象。S-CypherWalker 继承了 ANTLR4 生成的解析树监听器,在遍历语法解析树的过程中,监听解析过程,并记录时态图查询语句的关键信息。

3)根据 S-CypherWalker 对象携带的关键信息,使用翻译器将其转换为中间表示 S-CypherQuery 对象。S-CypherQuery 对象是时态图查询语句的对象表示,包含了各类子句对象。

4) 根据 S-CypherQuery 对象的组成,使用生成器将其转换为扩展的 Cypher 查询语句。

4.2　时态拓展插件

对于无法直接转换为基础 Cypher 查询语句的时态图查询语句,如时态路径查询、复杂的时态数据操作等。可以借助 Neo4j 提供的用户自定义过程和函数机制[11]构建自定义时态算法转换。

通过使用 Java 编写用户自定义过程和函数来构建 S-Cypher 所需的自定义时态算法,并将其编译为 jar 文件,放入 Neo4j 的安装目录后,就可以直接在 Cypher 中调用自定义时态算法,从而在 Neo4j 中运行扩展的 Cypher 查询语句。

此外,在进行时态更新时,有可能会破坏时态约束,而翻译器无法检查所有非法时态更新语句。对于翻译器无法检测到的非法更新语句,本文借助自定义时态算法来进行检查,若更新语句违反了时态约束,则给出错误提示,确保前文提到的时态属性图的时间完整性约束。

在一个简单的社交网络时态图上进行了一系列非法的时间更新操作,并测试了每种违反约束的情况。对于每个违反时间约束的操作,S-Cypher 都会抛出错误,这表明 S-Cypher 可以通过调用时态函数来检查有效时间的有效性,并给出错误提示。由于篇幅限制,更多的非法更新语句参见文献[29],以下为其中一个非法 S-Cypher 查询示例。

例 1:创建一个于 1990 年出生名为“Mary Smith”的人和一个于 1937 年出生名为“Daniel Yang”的人,并创建一条 Mary Smith 与 Daniel Yang 之间的从 1937 年到 1990 年的朋友关系。

Create (n_1: Person @ *T* (" 1990 ") { name:" Mary Smith" }), (n_2: Person @ *T* (" 1937 ") { name:" Daniel Yang" }) *Create* (n_1) - [*e*: *Friend* @ *T* (" 1937 ", " 1990")] - (n_2)

查询结果：

> Constraint Error: The valid time of the edge must fall within the valid time of the connecting point.

结果显示，该语句不满足约束条件，查询失败。原因是新创建的朋友关系的有效时间并不在 Mary Smith 的有效时间内，违反了约束 8。这与人们的常识相一致，1990 年出生的 Mary Smith 不可能在 1990 年前就与 Daniel Yang 成为朋友。

5 实验

本节中的实验始终在 8 内核，16 GB 的单机上运行。实验使用了第 3 节的实现方案，在社区版 Neo4j 5.15.0 上进行时态查询。这些实验旨在回答以下问题：

1）与 Cypher 相比，S-Cypher 的时态语法是否更简便？

2）与 Cypher 相比，S-Cypher 时态查询的运行性能如何？

5.1 数据集

实验使用美国航班的真实数据[30]，选取了其中 2023 年的数据。图中对象节点的标签均为机场（Airport），机场节点之间的边的类型为航班和飞机尾部编号。节点的有效时间均设为[1987 年，*NOW*），航班边的有效时间则设为[<起飞时间>，<到达时间> ）。

5.2 测试

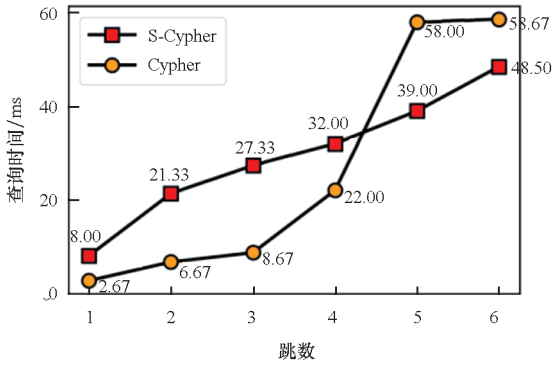
5.2.1 性能对比

首先，使用所提出的时态属性图模型和 S-Cypher 查询语言在 Neo4j 上进行时态查询。然后，与直接使用属性图模型和 Cypher 查询语言进行时态查询的方法进行性能对比。前者使用了第 4 节提出的实现方案，后者直接在节点上存储该节点及其所有属性的有效时间。

该测试选择 K 跳顺序有效路径查询作为工作负载。对于每个查询，随机选择一个机场节点，查询该机场只要最多经过 K 条航班就能到达的所有机场，并返回这些机场的编码。图 3 展示了两个方案分别进行 K 跳顺序有效路径查询的查询时间。

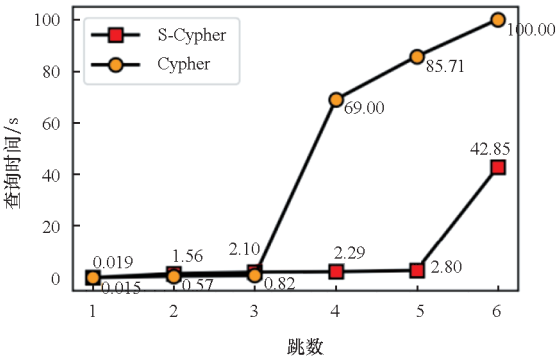
结果显示，在进行 1 ~ 6 跳顺序有效路径查询时，S-Cypher 顺序有效路径查询的查询时间随着跳数增加而增加，与 Cypher 相比，S-Cypher 查询的运行时间未显著增加，能够以良好的性能执行现实任务。

结果表明，当跳数较少或图规模较小时（具体为图 3（a）的前 4 跳，图 3（b）的前 3 跳），S-Cypher 的顺序有效路径查询的查询速度比 Cypher 慢，平均查询时间是 Cypher 的 2.47 倍。



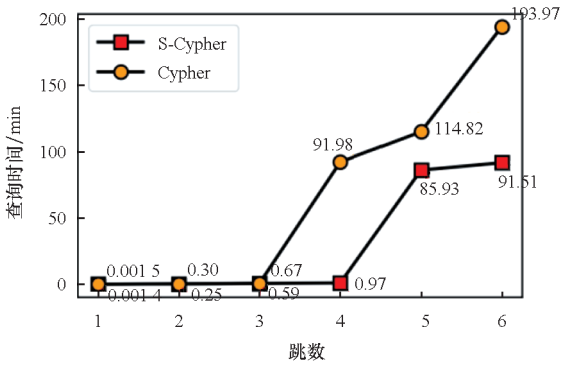
（a）查询包含 1 万条边的时态图

（a）Query a temporal graph containing 10 000 edges



（b）查询包含 10 万条边的时态图

（b）Query a temporal graph containing 100 000 edges



（c）查询包含 100 万条边的时态图

（c）Query a temporal graph containing 1 000 000 edges

图 3 不同规模时态图下 S-Cypher 与 Cypher 在 K 跳顺序有效路径查询中的性能对比

Fig. 3 Performance comparison between S-Cypher and Cypher for K-hop sequential path queries across temporal graphs of varying scales

然而，当跳数较多或图规模较大时，S-Cypher 的顺序有效路径查询有着更快的查询速度，平均查询时间为 Cypher 的 53%。这是由于 Neo4j 使用内

联机制进行了属性存储优化[31]。Neo4j 数据库使用固定大小的属性记录存储属性,属性名称直接存储在属性记录中,而属性值则通过一个指向动态存储区的指针或内联值存储在属性记录中。当属性数据少到可以编码到一个属性记录中时,属性值可以直接内联到属性存储文件,而不是存储在一个动态存储区中。显然,通过直接访问内联值获取属性值比从动态存储区查找属性值高效得多。

直接在节点上存储节点和节点属性的有效时间时,由于属性数量较多,节点只能通过访问动态存储区获取属性值。而在使用所提出的时态属性图模型存储数据时,每个节点和边只有 3 个属性,即 *content*、*startTimes* 和 *endTimes*,在有效时间的时间区间集较小时,可以直接通过访问内联值获取属性值,从而花费了更小的开销。当对有效时间进行索引时,时态查询时间将进一步缩短。

5.2.2　语法简便性

本节通过对比 S-Cypher 查询语句及转换后的 Cypher 查询语句的字符数,验证 S-Cypher 时态语法的简便性。其中,S-Cypher 查询语句和 Cypher 语句在结构和内容均相同的时序图数据上进行查询,其查询结果保持一致。

对简单的社交网络时态图和高速铁路交通时态图进行时态图查询。目标涵盖所有基本时态图查询函数,并对转换前后的查询语句进行比较。测试语句设计 S-Cypher 的所有语法,包括 *Match*、*Create*、*Delete*、*Stale*、*Set* 和 *Return*。

经过统计,S-Cypher 查询语句的平均字符数为 Cypher 的 20.4%,表明 S-Cypher 可以以更简洁的语法查询时态图数据。此外,由于需要确保时间约束,大多数 Cypher 查询语句需要依赖于我们定义的时态函数来实现时态图查询。简单地使用 Cypher 进行时态图数据操作并不能保证正确的时态约束,这也展示了 S-Cypher 的时态图查询能力。由于篇幅限制,详细的查询语句和字符数统计参见文献[29]。

5.3　局限性讨论

从实验结果看,在跳数较少或图规模较小时,S-Cypher 顺序有效路径查询的平均查询时间为 Cypher 的 2.47 倍,显示其查询性能还有进一步的完善空间。在未来的工作中,可以通过在图数据库中对有效时间、时态路径等时态对象建立索引来加快时态查询性能,也可以设计时态图数据的底层存储和索引结构来优化时态信息的搜索速度。

此外,由于 S-Cypher 的实现依赖现有图数据库,其时态查询的本质是在图模式匹配的过程中过滤有效时间符合要求的节点和边。但如果设计了专门的时态图数据引擎,就能够使用其他方式进行时态查询,比如优先对有效时间进行筛选,能够更高效地进行有效时间的范围匹配和多点匹配,再比如支持时态视图,能够对频繁访问的时间点内的数据进行查询。合适的时态图数据库引擎不仅能够提升查询性能,也方便扩展更复杂的时态查询算法,这也是未来的工作方向之一。

6　结论

为更高效地管理时态数据,本文提出了一种基于属性图的时态数据模型和相应的时态图查询语言 S-Cypher。该时态图模型使用对象节点表示实体,引入属性节点和值节点表示实体的属性,在节点以及实体与实体间的边上记录有效时间以表达时态信息,其记录的有效时间均遵循一组时态约束。

S-Cypher 是 Cypher 的时态拓展,保证兼容的同时提供了一套简洁完善的时态查询语法,包括时态数据类型、时态图模式匹配、时间窗口限定、时态路径和时态更新语法等,其中更新语法能够始终遵循时态属性图的约束。

本文还提出了一套将时态数据以时态属性图形式组织起来并存储到 Neo4j 等属性图数据库的建模规则,以及一套将 S-Cypher 查询语句运行在 Neo4j 图数据库上的实现方案,从而可以在安装了时间函数插件的 Neo4j 图数据库上查询 S-Cypher 查询语句,对时态数据进行管理。

最后,对 S-Cypher 的查询效果进行了实验验证。实验显示 S-Cypher 能够以简便的语法以及良好的性能进行时态图查询,且时态更新始终遵循约束。

研究表明,由于 Neo4j 的内联机制,在利用所提时态属性图模型来存储时态图,并处理海量数据和多跳查询时,可以提高时态查询的性能。这提供了一个新的研究思路,表明利用 Neo4j 的内联机制可以提高图数据的查询性能。

参考文献(References)

[1]　李晓杰, 崔超然, 宋广乐, 等. 基于时序超图卷积神经网络的股票趋势预测方法[J]. 计算机应用, 2022, 42(3): 797 – 803.
LI X J, CUI C R, SONG G L, et al. Stock trend prediction method based on temporal hypergraph convolutional neural network [J]. Journal of Computer Applications, 2022, 42(3): 797 – 803. (in Chinese)

[2]　BOLLEN E, HENDRIX R, KUIJPERS B, et al. Analysing

river systems with time series data using path queries in graph databases [J]. ISPRS International Journal of Geo-Information, 2023, 12(3): 94.

[3] 于健, 赵满坤, 高洁, 等. 基于高阶和时序特征的图神经网络社会化推荐算法研究[J]. 计算机科学, 2023, 50(3): 49 – 64.
YU J, ZHAO M K, GAO J, et al. Study on graph neural networks social recommendation based on high-order and temporal features [J]. Computer Science, 2023, 50(3): 49 – 64. (in Chinese)

[4] 孙男男, 朴春慧, 马新娜. 基于社交关系和时序信息的团购推荐方法[J]. 计算机应用, 2023, 43(6): 1719 – 1729.
SUN N N, PIAO C H, MA X N. Group buying recommendation method based on social relationship and time-series information [J]. Journal of Computer Applications, 2023, 43(6): 1719 – 1729. (in Chinese)

[5] TANSEL A U, CLIFFORD J, GADIA S, et al. Temporal databases: theory, design, and implementation [M]. Redwood City, California: Benjamin-Cummings Publishing Co., Inc., 1993.

[6] SNODGRASS R. The temporal query language TQuel [J]. ACM Transactions on Database Systems, 1987, 12(2): 247 – 298.

[7] SNODGRASS R T, AHN I, ARIAV G, et al. TSQL2 language specification [J]. ACM SIGMOD Record, 1994, 23(1): 65 – 86.

[8] ALLEN J F. Maintaining knowledge about temporal intervals[J]. Communications of the ACM, 1983, 26(11): 832 – 843.

[9] ANGLES R. The property graph database model [EB/OL]. [2024 – 11 – 01]. https://www.ceur-ws.org/Vol-2100/paper26.pdf.

[10] World Wide Web Consortium. RDF 1.1 concepts and abstract syntax [EB/OL]. (2014 – 02 – 25) [2024 – 11 – 01]. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[11] Neo4j. Neo4j documentation [DB/OL]. [2024 – 11 – 01]. https://neo4j.com/docs/.

[12] World Wide Web Consortium. SPARQL 1.1 query language [EB/OL]. (2023 – 03 – 21) [2024 – 01 – 19]. http://www.w3.org/TR/sparql11-query/.

[13] Apache TinkerPop. TinkerPop compendium [DB/OL]. (2023 – 11 – 21) [2024 – 01 – 19]. https://tinkerpop.apache.org/docs/current/.

[14] PUGLIESE A, UDREA O, SUBRAHMANIAN V S. Scaling RDF with time [C]//Proceedings of the 17th International Conference on World Wide Web, 2008: 605 – 614.

[15] RODRıGUEZ A, MCGRATH R, LIU Y, et al. Semantic management of streaming data [C]//Proceedings of the Semantic Sensor Networks, 2009.

[16] RULA A, PALMONARI M, NGONGA A C, et al. Hybrid acquisition of temporal scopes for RDF data[C]//Proceedings of the International Conference on Semantic Web: Trends and Challenges, 2014.

[17] TAPPOLET J, BERNSTEIN A. Applied temporal RDF: efficient temporal querying of RDF data with SPARQL[C]//Proceedings of the Semantic Web: Research and Applications, 2009.

[18] GAO S, GU J Q, ZANIOLO C. RDF-TX: a fast, user-friendly system for querying the history of RDF knowledge bases[C]//Proceedings of the International Conference on Extending Database Technology, 2016.

[19] GHRAB A, SKHIRI S, JOUILI S, et al. An analytics-aware conceptual model for evolving graphs[C]//Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, 2013.

[20] CAMPOS A, MOZZINO J, VAISMAN A. Towards temporal graph databases[EB/OL]. (2016 – 05 – 02) [2024 – 10 – 31]. https://arxiv.org/abs/1604.08568v2.

[21] DEBROUVIER A, PARODI E, PERAZZO M, et al. A model and query language for temporal graph databases[J]. The VLDB Journal, 2021, 30(5): 825 – 858.

[22] LABOUSEUR A G, BIRNBAUM J, OLSEN Jr. P W, et al. The G* graph database: efficiently managing large distributed dynamic graphs [J]. Distributed and Parallel Databases, 2015, 33: 479 – 514.

[23] KHURANA U, DESHPANDE A. Efficient snapshot retrieval over historical graph data[C]//Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE), 2013.

[24] HUANG H X, SONG J H, LIN X L, et al. TGraph: a temporal graph data management system[C]//Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016.

[25] BYUN J, WOO S, KIM D. ChronoGraph: enabling temporal graph traversals for efficient information diffusion analysis over time [C]//Proceedings of the IEEE 36th International Conference on Data Engineering (ICDE), 2020.

[26] ROST C, FRITZSCHE P, SCHONS L, et al. Bitemporal property graphs to organize evolving systems [EB/OL]. (2021 – 11 – 26) [2024 – 10 – 31]. https://arxiv.org/abs/2111.13499v1.

[27] RIZZOLO F, VAISMAN A A. Temporal XML: modeling, indexing, and query processing [J]. The VLDB Journal, 2008, 17: 1179 – 1212.

[28] PARR T. The definitive ANTLR 4 reference [M]. Dallas, Texas · Raleigh, North Carolina: The Pragmatic Bookshelf, 2013.

[29] Anon. S-Cypher: a temporal query language on the temporal property graph model. [EB/OL]. (2024 – 07 – 18) [2024 – 11 – 06]. https://zenodo.org/doi/10.5281/zenodo.12740355.

[30] Bureau of Transportation Statistics. Reporting carrier on-time performance (1987 – present) [DB/OL]. (2023 – 12 – 01) [2024 – 11 – 01]. https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&QO_fu146_anzr=b0-gvzr.

[31] ROBINSON I, WEBBER J, EIFREM E. Graph databases: new opportunities for connected data [M]. 2nd ed. Sebastopol, CA: O'Reilly, 2015.