arXiv:2304.00531v1 [cs.DB] 2 Apr 2023

S2CTrans：搭建从 SPARQL 到 Cypher 的桥梁*

Zihao Zhao[1,2],Xiaodong Ge[1,2], 和沈志宏[1(✉)]

[1] 中国科学院计算机网络信息中心，中国[2] 中国科学院大学，中国{zhaozihao,gexiaodong,bluejoe}@cnic.cn

摘要。 在图数据应用中，数据主要采用两种模型进行维护：RDF（资源描述框架）和属性图。属性图模型在业界得到广泛采用，因此属性图数据库在图遍历查询性能方面通常优于 RDF 数据库。然而，用户往往更倾向于使用 SPARQL 作为查询语言，因为它是 W3C 推荐的标准。因此，探索 SPARQL 到属性图查询语言的转换，对于增强图查询语言的互操作性，以及支持使用 SPARQL 高效查询属性图至关重要。尽管 SPARQL 与 Cypher 等属性图查询语言在语义表示和处理逻辑上存在显著差异，本文通过图关系代数证明了将 SPARQL 转换为 Cypher 以执行图遍历查询的可行性。我们提出了 S2CTrans 框架，该框架能够在保留原有语义的同时实现 SPARQL 到 Cypher 的转换。使用柏林SPARQL基准测试（BSBM）数据集进行的实验结果表明，S2CTrans 成功将 SPARQL 1.1 规范中的大多数 SELECT 查询转换为类型安全的Cypher语句，在保持结果一致性的同时，提高了使用 SPARQL 进行数据查询的效率。

关键词： 图查询语言 · RDF · 属性图 · SPARQL · Cypher

1 引言

知识图谱模型通过实体、概念、属性及其关系来表示现实世界，可为主题研究提供实用且有价值的见解，并具有广阔的应用前景 [10]。目前，知识图谱存储主要依赖两种模型：资源描述框架（RDF） [9] 和属性图 [1]。Jena [19]，等 RDF 数据库维护前者，而 Neo4j [17]，等属性图数据库则管理后者。

由于属性图模型简单、直观且性能优越，它已在图数据库行业得到广泛采用。

* Zihao Zhao 和 Xiaodong Ge 对本文贡献相同。

总体而言，属性图数据库在图遍历和模式匹配任务中的表现优于 RDF 数据库。然而，由于 SPARQL 是 W3C 长期推荐的标准语言，用户在进行数据查询时往往更倾向于使用 SPARQL。2019年 W3C 图数据网络标准化研讨会 [21]呼吁弥合 RDF 与属性图查询语言之间的差距，使系统能够使用属性图数据模型管理数据，同时让用户能够通过 SPARQL 查询数据。

SPARQL 与以 Cypher [6],为代表的属性图查询语言，在语义表示和处理逻辑方面存在差异，这使标准化过程充满挑战。为了推动知识图查询语言的标准化，并提升语义网与图数据库社区之间的互操作性，有必要将 SPARQL 转换为 Cypher。

这一转换面临的主要挑战如下：

– 证明 SPARQL 与 Cypher 在图遍历查询中的语义等价性。– 通过模式映射和数据映射，解决 RDF 模型与属性图模型存储之间的冲突。– 设计模式匹配映射和解修饰符映射方法，将 SPARQL 转换为 Cypher。

在本研究中，我们建立了基于图关系代数的 SPARQL 语义，并提出了 S2CTrans——一种可证明保持语义的 SPARQL 到 Cypher 转换方法。随后，我们在公开数据集上使用全面的查询功能对 S2CTrans 进行了评估。我们的贡献可概括如下：

– 我们基于图关系代数确立了 SPARQL 的语义，并展示了 SPARQL 与 Cypher 在表示图查询方面的语义等价性。–我们介绍了 S2CTrans 框架，该框架为模式匹配和解修饰符提供映射方法，从而能够将 SPARQL 转换为 Cypher。– 我们在大规模数据集上进行了全面的查询测试，以评估使用 S2CTrans 框架将 SPARQL 转换后，Cypher 在图数据库中的性能提升。

我们的工作示意图如图1所示。在数据层面，我们使用 Neo4J Labs 开发的 neosemantincs 插件 [20]，将 RDF 图进行语法和语义转换为属性图。这涉及将 RDF 三元组存储为属性图中的节点、关系和属性。在查询层面，该图展示了上文讨论的前两项贡献。虚线箭头 ① 表示 [8] [11], 中定义的 Cypher 图关系代数，而虚线箭头 ② 表示 [14] 中定义的 SPARQL 基于映射的语义。我们的贡献由虚线箭头 ③、 ④ 和 ⑤表示，它们定义了一种基于图关系代数的

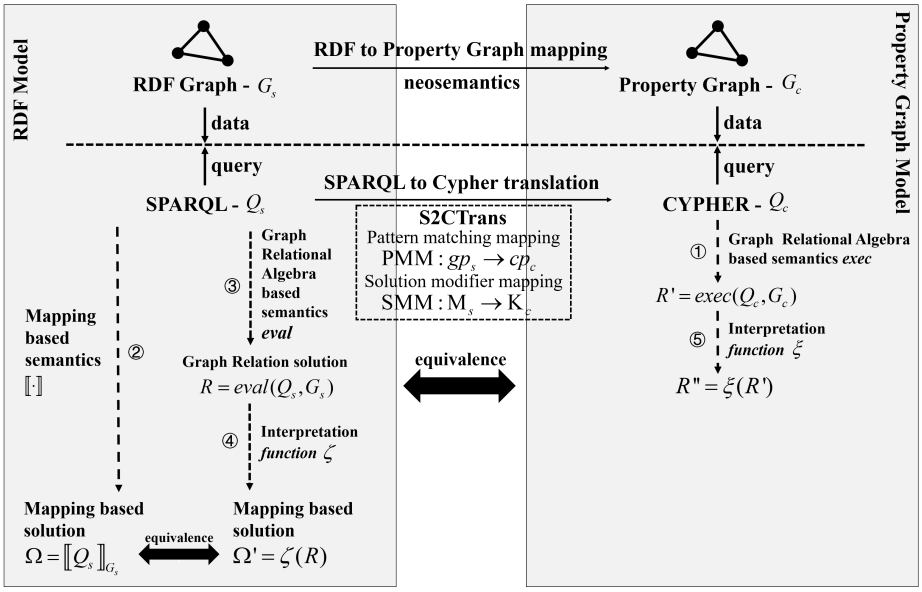


图 1. SPARQL 到 Cypher 转换概览。

SPARQL 语义。此外，实线箭头表示我们对 SPARQL 到 Cypher 的转换定义所作的贡献，其中包括模式匹配映射（**PMM**）和解修饰符映射（**SMM**）。

本文其余部分安排如下。第 2 节介绍本文工作的预备知识。第 3 节定义基于图关系代数的 SPARQL 语义，并证明 SPARQL 到 Cypher 转换的可行性。第 4 节介绍 S2CTrans 的系统架构、映射方法和局限性。第 5 节介绍评估策略和测试，并分析大规模数据集上的实验结果。第 6 节回顾知识图查询语言交互性方面的相关工作。最后，第 7 节总结本文并讨论未来可能的研究工作。

2 预备知识

2.1 SPARQL 图模式

设 I、B、L 和 V 分别表示两两不相交的无限集合，依次由国际化资源标识符（国际化资源标识符）、空白节点、字面量和变量组成。下面，我们将形式化定义 RDF 三元组、RDF 图、三元组模式和 SPARQL 图模式。

定义 1. *RDF* 三元组与 *RDF* 图。*RDF* 三元组 t 是一个元组 $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$，其中 s、p 和 o 分别是主语、谓词和宾语。*RDF* 图 G_s 是 *RDF* 三元组的集合。

定义 2. 三元组模式。三元组模式 tp 是一个三元组 $(sp, pp, op) \in (I \cup V \cup L) \times (I \cup V) \times (I \cup V \cup L)$，其中 sp、pp 和 op 分别是主语模式、谓词模式和宾语模式。它们的形式化定义分别见公式 *(1)* 和 *(2)*[3] [4]。表 *1* 列出了本文使用的主要符号。

$$\varphi(x) = \begin{cases} \text{Variable, A variable starting with ?. eg: ?x} \\ \text{IRI, Identify a resource. eg: foaf:know} \\ \text{Literal, Property Value. eg: 100} \end{cases} \tag{1}$$

$$\phi(tp) = \begin{cases} Type, & p = \text{rdf:type / a} \\ VarEdge, & \varphi(pp) = \text{Variable} \\ IRIEdge, & \varphi(pp) = \text{IRI} \wedge pp \in T \\ Property, & \varphi(pp) = \text{IRI} \wedge pp \in P \end{cases} \tag{2}$$

表 1. 符号列表

SPARQL 概念	符号	Cypher 概念	符号 / 集合表示法
SPARQL 查询	Q_s	Cypher 查询	Q_c
三元组模式	$tp = (sp, pp, op)$	节点模式	χ
图模式	gp	关系模式	ρ
三元组元素的类型	$\varphi(x)$ $x \in \{sp, pp, op\}$	路径模式	ϵ / $\mathcal{E}$
三元组模式的类型	$\phi(tp)$	节点	n / $\mathcal{N}$
解修饰符	$\mathbf{M}_s$	关系	r / $\mathcal{R}$
S2CTrans 概念	**表示法**	属性键	p / P
模式匹配映射	$\mathbf{PMM}$	节点标签	l / L
解修饰符映射	$\mathbf{SMM}$	关系类型	t / T
Cypher 关键字	$\mathbf{K}_c$	名称	a / $\mathcal{A}$
-	-	值	v / $\mathcal{V}$

定义 3. *SPARQL* 图模式。*SPARQL* 图模式 gp 由以下抽象语法定义：

$$gp \rightarrow tp \mid gp \ \mathbf{AND} \ gp \mid gp \ \mathbf{OPT} \ gp \mid gp \ \mathbf{UNION} \ gp \mid gp \ \mathbf{FILTER} \ expr_s$$

其中，AND、OPT 和 UNION 分别是二元运算符，对应于 SPARQL 的合取、OPTIONAL 和 UNION 构造。FILTER 根据给定表达式$expr_s$限制图模式匹配的解。

2.2 Cypher 图模式

定义 4. 属性图。属性图定义为 $G_c = (\mathcal{N}, \mathcal{R}, st, \mathcal{L}, \mathcal{T}, \mathcal{P})$，其中：

– $\mathcal{N}$ 是节点集合。[3] 本文仅考虑将谓词作为变量，以表示未知边。[4] 我们对 RDF 数据集进行过滤，以获取关系类型 IRI T 和属性键 IRI P。

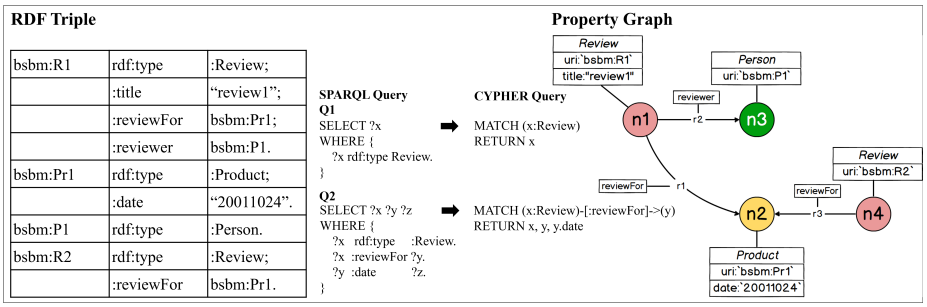


图 2. BSBM 数据集中 RDF 三元组及其对应属性图、SPARQL 查询及其对应 Cypher 查询示例。

– $\mathcal{R}$ 是关系集合– st： $\mathcal{R} \to \mathcal{N} \times \mathcal{N}$ 将源节点和目标节点分配给关系。– $\mathcal{L}$： $\mathcal{N} \to 2^L$ 为每个节点分配一组标签。– $\mathcal{T}$： $\mathcal{R} \to T$ 为每个关系分配单一类型。– $\mathcal{P}$： $(\mathcal{N} \cup \mathcal{R}) \times P \to \mathcal{V}$ 是一个有限的部分函数，它将（节点或关系）和属性键映射到一个值。

定义 5. 节点模式和关系模式。节点模式 χ 是一个三元组 $(a, L, \mathcal{P}_v)$，其中：$a \in \mathcal{A} \cup \{nil\}$ 是可选名称。以图 2 中的节点 '$(n1 : Review\ \{uri : $ **'bsbm:R1'** $title : \text{“}review1\text{”}\})$'为例， $n1$ 的节点模式表示为

$$\chi_1 = (n1, \{Review\}, \{uri \mapsto \text{'bsbm:R1'}, title \mapsto \text{“}review1\text{”}\}).$$

关系模式 ρ是一个元组 $(d, a, T, \mathcal{P}_e, I)$，其中： $d \in \{\to, \leftarrow, \leftrightarrow\}$指定模式的方向。 $a \in \mathcal{A} \cup \{\text{nil}\}$是可选名称。 I定义关系模式的范围。以图 2 中的关系'$-[r1 : reviewFor] \to$'为例， $r1$ 的关系模式表示为

$$\rho_1 = (\to, r1, reviewFor, \texttt{nil}, \texttt{nil}).$$

定义 6. *Cypher* 图模式和组合图模式。*Cypher* 图模式 ϵ 由节点模式和路径模式组成，其定义如下所示的抽象语法：

$$\epsilon \to \chi \mid \chi\rho\epsilon$$

A匹配 子句定义了一个图模式。一个查询可以由多个跨越多个 **MATCH**子句的模式组成。Cypher 组合模式 cp 由以下抽象语法 [5]定义：

$$cp \to \epsilon \mid cp\ \textbf{AND}\ cp \mid cp\ \textbf{OPT}\ cp \mid cp\ \textbf{FILTER}\ expr_c$$

[5] 在 Cypher 语句中，**UNION** 关键字用于合并两个查询的结果，而非图模式。

2.3 图关系代数

定义 7。 图关系。给定一个属性图 G_c，如果满足以下条件 *[8,11]*，则关系 R 是图关系：

$$\forall A \in sch(R) : dom(A) \subseteq \mathcal{N} \cup \mathcal{R} \cup expr_c$$

$sch(R)$是 R的模式，即包含属性名的列表。 $dom(A)$是属性 A的定义域。 $\mathcal{N}$ 和 $\mathcal{R}$ 分别是 G_c的节点和关系。 $expr_c$ 表示 G_c中属性、标签、类型和函数的表达式。

要访问节点的某个属性，我们使用表达式 $n.p$ 来访问属性 p对应的值。此外，表达式 $\mathcal{L}(n)$返回节点 n的标签， $\mathcal{T}(r)$返回关系 r的类型，而 $\mathcal{P}(n)$返回节点 n的属性集。

定义 8. ***GetNodes*** 算子。考虑一个属性图 G_c ，其节点集为 $\mathcal{N}$。***GetNodes*** 算子记作 $\bigcirc_{(x)}$，它返回一个仅含属性 x 的图关系，其中包含 G_c *[8,11]*的节点。以图 *2* 中的查询 $Q1$ 为例，对应的图关系代数如下：

$$\bigcirc_{(x:Review)}$$

定义 9. ***Expand*** 算子。设 R 为一个图关系， $x \in sch(R)$ 为一个属性。 $ExpandOut$ 算子 $\uparrow^{(y)}_{(x)}[e]$ 向 R添加新列 y 和 e ，其中包含从 x中的节点沿着出向关系 e 可到达的 y 节点。以图 *2* 中的 $Q2$ 查询为例，相应的图关系代数如下所示：

$$\uparrow^{(y)}_{(x)} [_r{:}\text{reviewFor}] \bigcirc_{(x:Review)} \ ^{6}$$

3 SPARQL 和 Cypher 的语义表示

本节将按照图 1 中虚线箭头的编号顺序，介绍基于语义的 SPARQL 和 Cypher 表示。

3.1 Cypher 查询解的图关系表示。

令 $exec$ 表示一个函数，用于定义 S2CTrans 生成的 Cypher 语句所基于的图关系代数语义。该函数接收组合图模式 cp 或 Cypher 查询 Q_c 以及属性图 G_c，并返回一个图关系 R。 $exec$ 的定义见图 3(a)[7][11]。

[6] 所查询的图模式可能包含匿名节点和关系。在代数形式中，我们用以下划线开头的名称表示它们，例如 x 和 r。 [7] 该图的高分辨率版本收录于附录 S2CTrans-tech-report [22]。

– 规则 1 定义了在属性图 G_c上执行图模式 ϵ 的方式。共有两种情况：

- ϵ 是一个节点模式。将其与图 G_c中的节点集 $\mathcal{N}$ 进行匹配。如果它的标签和属性为 NULL 或与 n，($n \in \mathcal{N}$)一致，则称 n 满足节点模式 ϵ_χ，记作 $(n,\ G_c) \models \epsilon_\chi$。

- ϵ 是一个路径模式。除了满足节点模式 ϵ_χ之外，节点之间相应的关系模式 ϵ_ρ 还需要与 G_c中的关系集 $\mathcal{R}$ 相匹配。如果 G_c中存在一条路径 p，且 p 中的每个子关系都满足相应的 ϵ_ρ，则称 n 和p 满足路径模式 ϵ，记作 $(n \cdot p, G_c) \models \epsilon$。最后，对查询结果进行投影。– 规则 2 将两个组合图模式cp_1 和 cp_2 的 AND 运算定义为图关系 $R_1 = exec(cp_1, G_c)$ 和$R_2 = exec(cp_2, G_c)$的内连接。– 规则 3 将两个组合图模式cp_1 和 cp_2 的 OPT 运算定义为图关系 $R_1 = exec(cp_1, G_c)$ 和$R_2 = exec(cp_2, G_c)$的左外连接。– 规则 4 将两个查询 Q_{c1} 和 Q_{c2} 的 UNION 运算定义为图关系 $R_1 = exec(Q_{c1}, G_c)$ 和 $R_2 = exec(Q_{c2}, G_c)$的外并。– 规则 5 将组合图模式 cp 的 FILTER 表达式 $expr_c$ 定义为图关系 $R_1 = exec(cp, G_c)$中元组 R 的子集。

3.2 SPARQL 查询解的基于映射的表示。

设映射 μ： $V \to I \cup B \cup L$ 是一个部分函数，它将 RDF 术语分配给 SPARQL 查询中的变量。 μ 的定义域 $dom(\mu)$ 是 V 的子集， μ 在该子集上有定义。因此，SPARQL 查询解的基于映射的表示是一个映射集 Ω，其中包含映射 μ。我们将 Σ 定义为所有可能映射集构成的无限集合，每个映射集都表示一个 SPARQL 查询解。

SPARQL 的基于映射的语义定义为函数 $[\![\cdot]\!]$，该函数接受一个图模式表达式或 SPARQL 查询以及一个 RDF 图 G_s，并返回一个映射集，记作 $[\![\cdot]\!]$： $gp \to \Omega$。 $[\![\cdot]\!]$ 的定义见图 3(b)，其中分别定义了在 RDF 图 G_s 上对三元组模式 tp 、 gp_2 和 gp_2 求值的过程，以及对 tp**AND**gp_2、 gp_1 **OPT** gp_2、 gp_1 **UNION** gp_2 和 gp **过滤器** $expr_s$ 求值的过程。关于 $[\![\cdot]\!]$ 的详细说明及示例见 [14]。

3.3 SPARQL 查询解的图关系表示。

设元组 r： $IVL \to IBL \cup \{\text{NULL}\}$ 是一个全函数，它将 RDF 图中的 RDF 术语分配给 SPARQL 查询中的国际化资源标识符、字面量和变量。SPARQL 查询解的图关系表示是一个由元组 R 构成的集合r。我们将 $\mathscr{R}$ 定义为所有可能图关系构成的无限集合，其中每个图关系都表示一个 SPARQL 查询解。

Rule 1: $R = exec(\epsilon, G_c)$ Node Pattern: $(n, G_c) \models \epsilon_\chi,\ R = \pi_* \bigcirc_{(n)}$ where $(\epsilon_\chi.L \subseteq \mathcal{L}(n) \vee \epsilon_\chi.L$ is NIL$) \wedge (\epsilon_\chi.P \subseteq \mathcal{P}(n) \vee \epsilon_\chi.P$ is NIL$)$ Path Pattern: $(n \cdot p, G_c) \models \epsilon,\ R = \pi_* \uparrow_{(n_{m-1})}^{(n_m)} [p_{m-1}] \cdots (\uparrow_{(n_1)}^{(n_2)} [p_1] \bigcirc_{(n_1)})$ where $(n, G_c) \models \epsilon_\chi \wedge (\epsilon_\rho.T = \mathcal{T}(p) \vee \epsilon_\rho.T$ is NIL$)$	$[\![tp]\!]_{G_s} = \{\mu_s \mid dom(\mu_s) = var(tp)$ *and* $\mu_s(tp) \in G_s\}$	**Rule 6:** $R = eval(tp, G_s)$ $R(s,p,o) = \{(t.s, t.p, t.o) \mid t \in G_s \wedge genCond(tp)\},\ R_2 = genPR(tp)$ $\phi(tp) = Type \rightarrow R = \bigcirc_{(tp.sp:tp.op)}$ $\phi(tp) = Property \rightarrow R = \bigcirc_{(tp.sp\{tp.pp \rightarrow tp.op\})}$ $\phi(tp) = VarEdge \rightarrow R = \uparrow_{(tp.sp)}^{(tp.op)} [tp.pp] \bigcirc_{(tp.sp)}$ $\phi(tp) = IRIEdge \rightarrow R = \uparrow_{(tp.sp)}^{(tp.op)} [_p : tp.pp] \bigcirc_{(tp.sp)}$
Rule 2: $R = exec(cp_1$ **AND** $cp_2, G_c) = R_1 \bowtie R_2$ $R_1 = exec(cp_1, G_c), R_2 = exec(cp_2, G_c)$	$[\![gp_1$ **AND** $gp_2]\!]_{G_s} = [\![gp_1]\!]_{G_s} \bowtie [\![gp_2]\!]_{G_s} = \Omega_1 \bowtie \Omega_2$	**Rule 7:** $R = eval(gp_1$ **AND** $gp_2, G_s)$ $R_1 = eval(gp_1, G_s),\ R_2 = eval(gp_2, G_s)$ $R = R_1 \bowtie_{\wedge_{[a_i \mid a_i \in \xi(R_1) \cap \xi(R_2)]} (R_1.a_i = R_2.a_i \ \vee\ R_1.a_i \text{ is NULL} \vee R_2.a_i \text{ is NULL})} R_2$
Rule 3: $R = exec(cp_1$ **OPT** $cp_2, G_c) = R_1 \rtimes R_2$	$[\![gp_1$ **OPT** $gp_2]\!]_{G_s} = [\![gp_1]\!]_{G_s} \rtimes [\![gp_2]\!]_{G_s} = \Omega_1 \rtimes \Omega_2$	**Rule 8:** $R = eval(gp_1$ **OPT** $gp_2, G_s)$ $R = R_1 \rtimes_{\wedge_{[a_i \mid a_i \in \xi(R_1) \cap \xi(R_2)]} (R_1.a_i = R_2.a_i \ \vee\ R_1.a_i \text{ is NULL} \vee R_2.a_i \text{ is NULL})} R_2$
Rule 4: $R = exec(Q_{c1}$ **UNION** $Q_{c2}, G_c) = R_1 \cup R_2$ **OR** $R_1 \uplus R_2$ $R_1 = exec(Q_{c1}, G_c), R_2 = exec(Q_{c2}, G_c)$	$[\![gp_1$ **UNION** $gp_2]\!]_{G_s} = [\![gp_1]\!]_{G_s} \cup [\![gp_2]\!]_{G_s} = \Omega_1 \cup \Omega_2$	**Rule 9:** $R = eval(gp_1$ **UNION** $gp_2, G_s)$ $R = R_1 \cup R_2$ **OR** $R_1 \uplus R_2$
Rule 5: $R = exec(cp$ **FILTER** $expr_c, G_c) = \sigma_{expr_c}(R_1)$ $expr_c ::= v \mid a \mid expr_c$ **op** $expr_c \mid expr_c$ **OR** $expr_c \mid expr_c$ **AND** $expr_c$ where $a \in \mathcal{A}, v \in \mathcal{V},$ **op** $\rightarrow < \mid \leq \mid > \mid \geq \mid =$.	$[\![gp$ **FILTER** $expr_s]\!]_{G_s} = \sigma_{expr_s}[\![gp]\!]_{G_s} = \sigma_{expr_s}\Omega$ $expr_s ::= ?X$ **op** $L \mid ?X$ **op** $?Y \mid expr_s \ \vee\ expr_s \mid expr_s \ \wedge\ expr_s$ where $?X, ?Y \in dom(\mu_s),$ **op** $\rightarrow < \mid \leq \mid > \mid \geq \mid =$.	**Rule 10:** $R = eval(gp$ **FILTER** $expr_c, G_s)$ $R_1 = eval(gp, G_s), R = \{r \mid r \in R_1 \ \wedge\ expr_s(r)\} = \sigma_{expr_s}(R_1)$ $expr_s ::= ?X$ **op** $L \mid ?X$ **op** $?Y \mid expr_s \ \vee\ expr_s \mid expr_s \ \wedge\ expr_s$ where $expr_s(r) = true.\ ?X, ?Y \in \xi(R).\ r(?X), r(?Y)$ is not **NULL**. **op** $\rightarrow < \mid \leq \mid > \mid \geq \mid =$.
(a) Graph relational algebra based semantics of CYPHER	**(b) Mapping-based semantics of SPARQL**	**(c) Graph relational algebra based semantics of SPARQL**

为了匹配三元组模式并修改查询解，我们需要构建函数 $genCond$ 和 $genPR$。由于 Cypher 和 SQL 的查询匹配过程相似， [5] 中的 $gencond$ 函数可以复用。由于两种解的表示方式不同， $genPR$ 函数需要重写。

算法 1 根据变量和表达式修改解模式等价性。

输入： 三元组模式 tp
输出： 图关系代数表达式仅投影.....的属性图关系 $R(s,\ p,\ o)$ 对应变量、节点或 tp 中的关系，并重命名投影属性。

函数 $\mathrm{genPR}(tp)$
  $project\text{-}list = s$
  $rename\text{-}list = s \rightarrow tp.sp$
  如果 $(\phi(tp) = IRIEdge$ 或 $\phi(tp) = VarEdge)$ AND $tp.pp \neq tp.sp$ **则**
    $project\text{-}list \mathrel{+}= \{p, o\}, rename\text{-}list \mathrel{+}= \{p \rightarrow tp.pp, o \rightarrow tp.op\}$
  结束如果
  如果 $\phi(tp) = Type$ AND $\varphi(tp.op) = Variable$ **那么**
    $project\text{-}list \mathrel{+}= o,\ \ rename\text{-}list \mathrel{+}= o \rightarrow \mathcal{L}(tp.sp)$
  结束如果
  如果 $\phi(tp) = Property$ AND $\varphi(tp.op) = Variable$ **那么**
    $project\text{-}list \mathrel{+}= o,\ \ rename\text{-}list \mathrel{+}= o \rightarrow (tp.sp)。(tp.pp)$
  结束 if
  返回 $\pi_{project\text{-}list \rightarrow rename\text{-}list}(R)$
结束函数

我们将基于图关系代数的 SPARQL 语义定义为函数 $eval$ ，该函数接收图模式表达式 gp 或 SPARQL 查询 Q_s 以及 RDF 图 G_s ，并返回图关系 R。$eval$ 的定义见图 3(c)。

– 规则 6 分两步定义了在 RDF 图 G_s 上对三元组模式 tp 的求值。首先，创建具有模式 sch 的图关系 $R(R) = (s,\ p,\ o)$，并将所有满足由 $genCond(tp)$ 生成的条件、与 tp 匹配的三元组 $t \in G_s$ 存入 R。然后，根据由 $genPR(tp)$ 生成的图关系代数表达式，对 R 的属性进行投影和重命名，并创建新的图关系 R_2 。最后，将 R_2 赋值为该三元组模式的解。

– 规则 7 定义了两个图模式 gp_1 和gp_2 的**AND** 运算：将图关系 $R_1 = eval(gp_1, G_s)$ 与 $R_2 = eval(gp_2, G_s)$ 进行内连接。连接条件确保，对于每一对公共关系属性（$R_1.a_i, R_1.a_i$），如果 $a_i \in sch(R_1) \cap sch(R_2)$，则它们的值相等 $R_1.a_i = R_2.a_i$ ，或者其中一个或两个值为 NULL。最后，将连接结果表中的冗余属性合并为一个。

– 规则 8 定义了两个图模式 gp_1 和gp_2 的**OPT** 运算：将图关系 $R_1 = eval(gp_1, G_s)$ 与 $R_2 = eval(gp_2, G_s)$ 进行左外连接。– 规则 9 定义了两个图模式 gp_1和 gp_2 的**UNION** 运算：将图关系 $R_1 = eval(gp_1, G_s)$ 与 $R_2 = eval(gp_2, G_s)$ 进行外并。– 规则 10 定义了图模式 gp 的**FILTER** 表达式 $expr_s$ 的求值：取图关系 $R_1 = eval(gp, G_s)$ 中满足条件 $expr_s(r)$ 为真的元组 R 子集。

3.4 解释函数

图关系代数到映射尽管 SPARQL 基于图关系代数和映射有不同的解表示形式，但二者都使用 RDF 图元素标识元组。为证明二者等价，我们定义一个解释函数 $\zeta : \mathscr{R} \to \Sigma$，用于关联图关系与基于映射的表示。该函数接收图关系 $R \in \mathscr{R}$ 并返回映射集 $\Omega \in \Sigma$，使每个元组 $r \in R$ 都按以下方式分配一个映射$\mu \in \Omega$：若 $x \in sch(r)$ $x \in V$ 且 $r(x)$ 不为 NULL，则$x \in dom(\mu)$ 和 $\mu(x) = r(x)$。

以图 2 中的查询 Q2 为例，以下映射表明，当使用不同表示形式时，解释函数 ζ 可作为建立 SPARQL 查询解之间等价关系的工具：

$$R = \begin{array}{|l|l|l|l|} \hline \textbf{x} & \textbf{:reviewFor} & \textbf{y} & \textbf{y.date} \\ \hline \text{bsbm:R1} & \text{:reviewFor} & \text{bsbm:Pr1} & \text{“20011024”} \\ \text{bsbm:R2} & \text{:reviewFor} & \text{bsbm:Pr1} & \text{“20011024”} \\ \hline \end{array} \xrightarrow{\zeta}$$

$$\Omega = \begin{array}{|l|} \hline ?x \to \text{bsbm:R1}\ ?y \to \text{bsbm:Pr1}\ ?z \to \text{“20011024”} \\ \hline ?x \to \text{bsbm:R2}\ ?y \to \text{bsbm:Pr1}\ ?z \to \text{“20011024”} \\ \hline \end{array}$$

执行 **到** 求值 $exec$ 和 $eval$ 的解表示都是图关系。然而，由于属性图的存储机制以及 neosemantics [8]的转换策略，它们在表示节点和关系时所用的模式不同。在 $exec$中使用节点标识符和关系标识符，而在 $eval$中使用其统一资源标识符。尽管两种表示不同，但二者都能唯一标识一个节点或关系 [9]。因此，我们定义解释函数 $\xi : \mathscr{R} \to \mathscr{R}$，用于修改图关系的模式和元组，以说明 $exec$ 与 $eval$ 之间的等价性。ξ 主要包括以下三个步骤：

[8] Neosemantics 会将实体统一资源标识符存储在 RDF 数据集中，作为属性图中的节点属性，因此每个节点都可以通过节点标识符和统一资源标识符属性唯一标识自身。[9] 本文默认任意两个节点之间不存在重复的关系类型。

– 将模式中两个节点之间的关系修改为相应的关系统一资源标识符。– 将节点标识符映射到相应的节点统一资源标识符属性。– 将关系标识符映射到相应的关系类型。

我们仍以图 2 中的查询 Q2 为例，展示解释函数 ξ的具体映射方法，并证明 $eval$与 $exec$等价。

$$R' = \begin{array}{|lll|l|}\hline \textbf{x} & \textbf{xy} & \textbf{y} & \textbf{y.date} \\ \hline \text{n1} & \text{r1} & \text{n2} & \text{“20011024”} \\ \text{n4} & \text{r3} & \text{n2} & \text{“20011024”} \\ \hline \end{array} \xrightarrow{\xi}$$

$$R'' = \begin{array}{|lll|l|}\hline \textbf{x} & \textbf{:reviewFor} & \textbf{y} & \textbf{y.date} \\ \hline \text{bsbm:R1} & \text{:reviewFor} & \text{bsbm:Pr1} & \text{“20011024”} \\ \text{bsbm:R2} & \text{:reviewFor} & \text{bsbm:Pr1} & \text{“20011024”} \\ \hline \end{array} = R$$

4 S2CTrans

我们设计并实现了 S2CTrans，这是一个可将 SPARQL 等效转换为 Cypher 的框架。S2CTrans 已开源[10]。

4.1 系统架构

S2CTrans以SPARQL 查询作为输入，通过使用 Jena ARQ [19] 解析策略、图模式匹配与解修饰符转换策略以及 Cypher-DSL [12]构建策略，在保留原始语义的基础上生成 Cypher 语句。我们在图 4 中概述了 S2CTrans 的架构，并讨论五步执行流水线中每个步骤的作用。

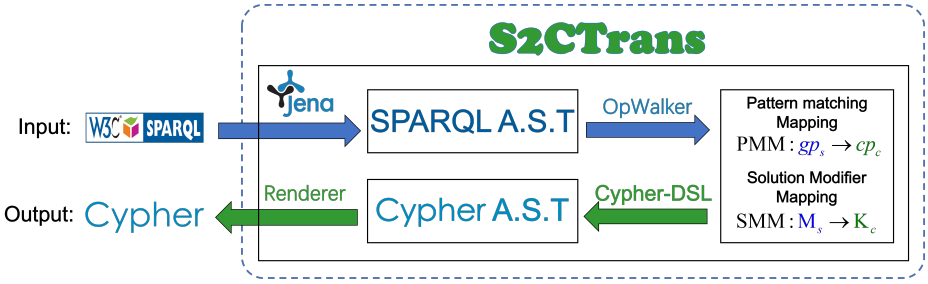


图 4. S2CTrans 流水线架构。

– 步骤 1： 首先由 Jena ARQ 模块解析输入的 SPARQL 查询。该模块可以检查语法错误、验证其是否为有效的 SPARQL 查询，并生成抽象语法树（AST）表示。

[10] https://github.com/MaseratiD/S2CTrans

– **步骤 2：** 获得解析后的 SPARQL 抽象语法树后，使用 OpWalker 自底向上访问图模式匹配部分和解修饰符部分。– **步骤 3：PMM** 将 SPARQL 图模式 gp_s 映射为 Cypher 的组合图模式 cp_c，然后 **SMM** 将 SPARQL 解修饰符 $\mathbf{M}_s$ 映射为 Cypher 子句关键字 $\mathbf{K}_c$。– **步骤 4：** Cypher-DSL 根据模式元素类型和运算符优先级生成最终的合取遍历，并构建 Cypher 抽象语法树。– **步骤 5：** 最后，Renderer 将 Cypher 抽象语法树渲染为完整的 Cypher 语句。借助 neosemantics 插件，可直接在 Neo4j 中执行此语句查询属性图的结果。

映射函数由两部分组成：图模式匹配**PMM** 和解修饰符映射**SMM**。

4.2 模式匹配映射

图模式匹配是图查询语言中最基础且重要的查询操作 [2,7]。在表 4.2 中，我们以图 2 为例，展示部分三元组模式 tp 到 Cypher 图模式 ϵ 及 Cypher-DSL 模式构造语句的对应映射。由于篇幅限制，图模式映射算法在附录中介绍。该算法中的映射函数**PMM** 将 SPARQL 图模式转换为 Cypher 图模式元素。

表 2. 三元组模式及其对应 Cypher 图模式的汇总列表。

三元组模式 - tp			Cypher 图模式 - PMM(tp)	
sp	*pp*	*op*	**图模式 ϵ**	**Cypher-DSL 模式构造**
?x	RDF 类型	:Review	$\chi = (x,$ Review $\varnothing)$	Cypher.节点(“Review”).named(“x”);
?x	:标题	“review1”	$\chi = (x, \varnothing, \{title \mapsto$ “$review1$”$\})$	Cypher.named(“x”).withProperties(“标题”,“review1”);
?x	**reviewFor**	**?y**	$\chi_x = (x, \varnothing, \varnothing)$	Cypher.anyNode(“x”);
			$\rho = (\rightarrow, \mathcal{r},$ reviewFor $\varnothing, nil)$	x.relationshipTo(y, “reviewFor”);
			$\chi_y = (y, \varnothing, \varnothing)$	Cypher.anyNode(“y”);
?x	RDF 类型	**?y**	$\chi.getLabel()$	标签函数(x);
?x	**?y**	?z	$\rho = (\rightarrow, y, \varnothing, \varnothing, nil)$	x.关系连接(z);
			$\rho.getType()$	类型函数(y);
?x	:标题	**?y**	$\chi.getProperty$(标题)	x.property(“标题”);

4.3 解决方案修饰符映射

通过**PMM** 算法获得图模式后，通常会添加条件来修饰图模式匹配的解。基于 SPARQL 和 Cypher 在图关系代数表达式方面的语义等价性，**SMM**算法构建了一个映射表（如表3所示），以实现将 SPARQL 解修饰符 $\mathbf{M}_s$ 映射到 Cypher 子句关键字 $\mathbf{K}_c$。

该表总结了图查询修改操作及其对应的图关系代数，以及 SPARQL 和 Cypher 子句构造的形式。变量和表达式已在**PMM** 算法中映射到图模式元素。

表3. SPARQL 解修饰符及对应 Cypher 子句关键字汇总。

操作	**代数**	**SPARQL 解修饰符 - $\mathbf{M}_s$**	**Cypher 子句关键字 - $\mathbf{K}_c$**
选择	$\sigma_{condition}(r)$	过滤器($Expr_1$ &&(||) $Expr_2$)	筛选 $Expr_1$ and(或) $Expr_2$
投影	$\pi_{x_1,\ x_2,\ ...}(r)$	选择 ?x_1 ?x_2 ...	返回 x_1, x_2, ...
去重	$\delta_{x_1,\ x_2,\ ...}(r)$	选择去重 ?x_1 ?x_2 ...	返回 去重 x_1, x_2, ...
限制	$\lambda_s^l(r)$	限制 l 跳过 s	限制 l 跳过 s
排序	$\varsigma_{\uparrow x_1,\ \downarrow x_2,\ ...}(r)$	ORDER BY ASC(?x_1) DESC(?x_2)	ORDER BY x_1 ASC, x_2 DESC

通过**PMM** 算法和**SMM** 算法，我们得到 Cypher 图模式和子句关键字。Cypher-DSL 根据图模式元素和运算符优先级构造 CypherAST。最后，我们使用 Renderer 构造完整的 Cypher 语句。

5 实验

5.1 评估标准

我们在若干顶尖 RDF 数据库上执行 SPARQL 查询，并在图数据库 Neo4j 上执行翻译后的 Cypher 查询。我们通过翻译速度、查询执行时间和结果一致性来评估 S2CTrans。

5.2 实验设置

数据集 本实验使用 W3C 推荐的 Berlin SPARQL 基准测试（BSBM）数据集，其中包含描述电子商务用例的合成数据，涉及产品、生产商等类别。我们使用 BSBM-Tools 分别生成了 1000 万个三元组，并通过 neosemantics 插件映射出相应的属性图版本。数据集详情见附录。

查询语句 我们总共创建了 40 个 SPARQL 查询，涵盖 30 种不同的查询功能。这些查询是在系统研究 SPARQL 查询的语义后选出的 [14]。查询详情见附录。

系统设置 我们在以下数据库上执行查询语句，以评估 S2CTrans 的性能提升：**属性图数据库：** Neo4j v4.2.3 **RDF 数据库：** Virtuoso v7.2.5、Stardog v7.6.3、RDF4J v3.6.3、

Jena TDB v4.0.0 所有实验均在以下机器配置上进行：CPU：英特尔酷睿处理器（Haswell）2.1GHz；RAM：16 GB DDR4；硬盘：512 GB 固态硬盘；操作系统：CentOS 7。为确保实验结果可复现，我们提供实验脚本、数据集和查询语句[11]。

5.3 结果评估

根据上述评估标准，我们在该数据集上对 RDF 数据库执行 SPARQL 查询，并在 Neo4j 属性图数据库上执行转换后的 Cypher 查询。最后，我们对比并分析查询结果。每个查询平均运行 10 次，以获取平均值。由于论文篇幅有限，语句转换结果和查询结果见 S2C 技术报告 [22] 的附录。

– **一致性：** 在所有实验中，SPARQL 查询结果与通过 S2CTrans 转换得到的 Cypher 查询结果完全相同。这表明 S2CTrans 能够将 SPARQL 查询语句等价地转换为 Cypher 查询语句。

– **性能分析：**

- **转换时间：** S2CTrans 在 BSBM-10M 上处理 40 个查询的平均转换时间为 $\mathbf{23.7ms}$。与查询时间相比，转换时间占比较小。
- **查询性能：** 我们在冷启动和热启动两种场景下，对不同规模的数据集进行了细致测试，所有测试均得到了相似的结果。图 5 展示了系统冷启动阶段的查询执行时间。其他测试结果详见附录。对于大多数查询语句，Neo4j 的表现优于 RDF 数据库。此外，对于包含多跳路径和较长关系的查询，Neo4j 的性能比 RDF 数据库高出 1 至 2 个数量级。主要原因是，RDF 数据库需要花费大量时间执行连接操作并生成执行计划，而 Neo4j 使用无索引邻接，大幅提升了查询效率。

实验结果证明，所提出的 S2CTrans 能够成功地等价转换并执行 SPARQL 查询。S2CTrans 使用户能够通过 SPARQL 查询属性图。

6 结论

本文首先基于图关系代数和映射语义，确立了将 SPARQL 转换为 Cypher 的可行性。随后，我们介绍了 S2CTrans，这是一种支持 SPARQL 到 Cypher 转换的新方法。该

[11] https://github.com/MaseratiD/S2CTrans

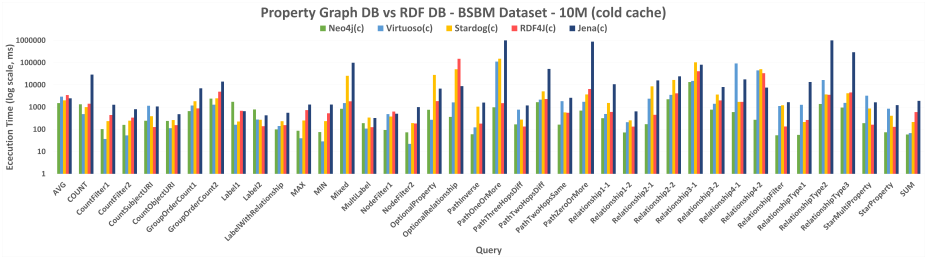


图 5. 属性图数据库与 RDF 数据库对比 - BSBM 数据集 10M

方法可以将大多数 SPARQL 语句转换为类型安全的Cypher语句。此外，我们使用属性图数据库和 RDF 数据库，在大规模数据集上开展实验评估，验证了该方法的有效性和适用性。评估结果凸显了将 SPARQL 查询转换为 Cypher 查询所带来的显著性能提升，尤其是在多关系查询和星形查询方面。尽管 S2CTrans 目前仍存在一些限制，但它代表着推动图查询语言标准化，以及增强语义网与图数据库社区之间数据和查询互操作性的重要一步。未来，我们计划进一步完善 S2CTrans，以支持更多 SPARQL 转换，并研究从 Cypher 到 SPARQL 的转换。

参考文献

1. Renzo Angles。《属性图数据库模型》。载于第 12 届 Alberto Mendelzon 数据管理基础国际研讨会论文集，第 2100 卷，2018年。2. Renzo Angles、Marcelo Arenas、Pablo Barcel´o、Aidan Hogan、Juan L. Reutter 和 Domagoj Vrgoc。《现代图数据库查询语言基础》。ACM Comput. Surv.，50(5):68:1–68:40，2017年。3. Renzo Angles、Harsh Thakkar 和 Dominik Tomaszuk。《RDF 与属性图的互操作性：现状与问题》。载于第 13 届 Alberto Mendelzon 数据管理基础国际研讨会论文集，第 2369 卷，2019年。4. Mohamed Bahaj 和 Nassima Soussi。《保持语义的 SQL 到 SPARQL 查询转换：用于左外连接》。Int. J. Web Appl.，9(4):138–144，2017年。5. Artem Chebotko、Shiyong Lu 和 Farshad Fotouhi。《保持语义的 SPARQL 到 SQL 转换》。Data Knowl. Eng.，68(10):973–1000，2009年。doi:10.1016/j.datak.2009.04.001。6. Francis、Nadime 等。《Cypher：一种不断演进的属性图查询语言》。2018 年国际数据管理会议论文集。2018年。7. Kongzhang Hao、Zhengyi Yang、Longbin Lai、Zhengmin Lai、Xin Jin 和 Xuemin Lin。《Patmat：一种支持 Cypher 的分布式模式匹配引擎》。载于第 28 届 ACM 信息与知识管理国际会议（CIKM）论文集，第 2921–2924 页，2019年。8. J¨urgen H¨olsch 和 Michael Grossniklaus。《用于转换 Neo4j 中图模式的代数及等价关系》。载于 EDBT/ICDT 2016 研讨会论文集

联合会议，EDBT/ICDT 研讨会 2016，CEUR 研讨会论文集第 1558 卷，2016年。9. Graham Klyne、Jeremy J. Carroll 和 Brian McBride。《RDF 1.1 概念与抽象语法》，W3C 推荐标准，2018年。10. Xiaodong Li、Reynold Cheng、Matin Najafi、Kevin Chen-Chuan Chang、Xiaolin Han 和 Hongtai Cao。《M-Cypher：支持模式子图的图查询语言（GQL）框架》。载于 CIKM '20：第 29 届 ACM 信息与知识管理国际会议论文集，第 3433–3436 页，2020年。11. J´ozsef Marton、G´abor Sz´arnyas 和 D´aniel Varr´o。《用关系代数形式化 openCypher 图查询》。载于《数据库与信息系统进展——第 21 届欧洲会议 ADBIS 2017》，计算机科学讲义第 10509 卷，第 182–196 页。Springer，2017年。12. Gerrit Meier 和 Michael Simons。《Neo4j Cypher-DSL》。https://neo4j-contrib.github.io/Cypher-dsl/current/，2021年。13. Ezequiel Jos´e Veloso Ferreira Moreira 和 Jos´e Carlos Ramalho。《Sparqling Neo4j（短文）》。载于第 9 届《语言、应用与技术研讨会》，第 83 卷，第 17:1–17:10 页，2020年。14. Jorge P´erez、Marcelo Arenas 和 Claudio Guti´errez。《SPARQL 的语义与复杂性》。ACM Trans. Database Syst.，34(3):16:1–16:45，2009年。15. Jyothsna Rachapalli、V. Khadilkar、Murat Kantarcioglu 和 B. Thuraisingham。《Retro：一种保持语义的 SQL 到 SPARQL 转换框架》。载于 ISWC 2011，2011年。16. Amit P. Sheth。《信息系统互操作性焦点的转变：从系统、语法、结构到语义》，第 5–29 页。Springer US，1999年。17. Neo4j 团队。《Neo4j 运维手册 v4.2》。https://neo4j.com/docs/operations-manual/current/，2020年。18. Harsh Thakkar、Dharmen Punjani、Jens Lehmann 和 S¨oren Auer。《一举两得：通过 Gremlinator 使用 SPARQL 查询属性图数据库》。载于第 1 届 ACM SIGMOD 图数据管理体验与系统（GRADES）及网络数据分析（NDA）联合国际研讨会论文集，第 12:1–12:5 页，2018年。19. K. Wilkinson。《Jena 属性表实现》。载于：Smart PR 编。《可扩展语义网知识库系统第 2 届国际研讨会论文集》，第 35–46 页，2006年。20. Neo4j Labs。《neosemantics（n10s）：Neo4j RDF 与语义工具包》。https://neo4j.com/labs/neosemantics/，2021年。21. Taelman、Ruben、Miel Vander Sande 和 Ruben Verborgh。《GraphQL 与 RDF 之间的桥梁》。W3C 图数据网络标准化研讨会。W3C。2019年。22. Zihao Zhao、Xiaodong Ge 和沈志宏。《S2CTrans：搭建从 SPARQL 到 Cypher 的桥梁》。arxiv。

7 附录

7.1 图模式映射算法

算法 2 模式匹配映射**输入：** SPARQL 图
模式 gp**输出：** Cypher 图模式集 $\mathcal{E}$

1: 函数 PMM(gp**) 2:** $\mathcal{E} \leftarrow \varnothing$**3: 对每个三元组模式** $tp \in gp, tp =$ **(**sp, pp, op**) 执行 4:** $\epsilon \leftarrow \mathcal{E}.getPattern$**(**$sp$**) 5: 如果** $\epsilon = \varnothing$ **则 6: 如果** ϕ**(**tp**)** $= Type$ **则** $\epsilon = \chi =$ **(**sp, op, nil**) 结束如果 7: 如果** ϕ**(**tp**)** $= Property$ **则** $\epsilon = \chi =$ **(**$sp, \varnothing, pp \mapsto op$**) 结束如果 8:** $\chi_s \leftarrow$ **(**$sp, \varnothing, nil$**),** $\chi_o \leftarrow$ **(**$op, \varnothing, nil$**) 9: 如果** ϕ**(**tp**)** $= IRIEdge$ **则** $\rho =$ **(**$\rightarrow$**,** $p, pp, \varnothing, nil$**),** $\epsilon = \chi_s \rho \chi_o$ **结束如果 10: 如果** ϕ**(**tp**)** $= VarEdge$ **则** $\rho =$ **(**$\rightarrow, pp, \varnothing,$ $\varnothing, nil$**),** $\epsilon = \chi_s \rho \chi_o$ **结束如果 11: 否则，如果** ϵ **是节点模式，则 12: 如果** ϕ**(**tp**)** $= Type$ **则** $\epsilon = \epsilon.addLabel$**(**$op$**) 结束如果 13: 如果** ϕ**(**tp**)** $= Property$ **则** $\epsilon = \epsilon.addProperty$**(**$pp \mapsto op$**) 结束如果 14:** $\chi_o \leftarrow$ **(**$op, \varnothing, nil$**) 15: 如果** ϕ**(**tp**)** $= IRIEdge$ **则** $\rho =$ **(**$\rightarrow$**,** $p, pp, \varnothing, nil$**),** $\epsilon = \epsilon \rho \chi_o$ **结束如果 16: 如果** ϕ**(**tp**)** $= VarEdge$ **则** $\rho =$ **(**$\rightarrow, pp, \varnothing, \varnothing, nil$**),** $\epsilon = \epsilon \rho \chi_o$ **结束如果 17: 否则 18:** $\chi_1 \leftarrow \epsilon.getStartNode$**(),** $\rho \leftarrow \epsilon.getRelation$**(),** $\chi_2 \leftarrow \epsilon.getEndNode$**() 19:** $\chi \leftarrow sp = \chi_1.getName$**() ?** χ_1 **:** χ_2**20: 如果** ϕ**(**tp**)** $= Type$ **则** $\epsilon = \chi.addLabel$**(**op**) 结束如果 21: 如果** ϕ**(**tp**)** $= Property$ **则** $\epsilon = \chi.addProperty$**(**$pp \mapsto op$**) 结束如果 22:** $\chi_3 \leftarrow$ **(**$op, \varnothing, nil$**) 23: 如果** $\chi = \chi_1$ **则 24: 如果** ϕ**(**tp**)** $= IRIEdge$ **则** $\rho' =$ **(**$\leftarrow$**,** $p, pp, \varnothing, nil$**),** $\epsilon = \chi_3 \rho' \chi \rho \chi_2$ **结束如果 25: 如果** ϕ**(**tp**)** $= VarEdge$ **则** $\rho' =$ **(**$\leftarrow, pp,$ $\varnothing, \varnothing, nil$**),** $\epsilon = \chi_3 \rho' \chi \rho \chi_2$ **结束如果 26: 否则 27: 如果** ϕ**(**tp**)** $= IRIEdge$ **则** $\rho' =$ **(**$\rightarrow$**,** $p, pp, \varnothing, nil$**),** $\epsilon = \chi_1 \rho \chi \rho' \chi_3$ **结束如果 28: 如果** ϕ**(**tp**)** $= VarEdge$ **则** $\rho' =$ **(**$\rightarrow,$ $pp, \varnothing, \varnothing, nil$**),** $\epsilon = \chi_1 \rho \chi \rho' \chi_3$ **结束如果 29: 结束如果 30: 结束如果 31:** $\mathcal{E}.addPattern$**(**ϵ**) 32: 结束循环 33: 返回** $\mathcal{E}$**34: 结束函数**

7.2 SPARQL 属性路径翻译

对于 SPARQL 1.1 中属性路径的转换，我们首先提取三元组模式中谓词的路径标识符，然后按照表 4 中的策略构造相应的 Cypher 关系模式，并将其添加到

Cypher 组合图模式中。最后，在属性图上执行模式匹配。

表 4. 将 SPARQL 属性路径转换为 Cypher 路径模式

路径类型	三元组	图模式	说明
谓词路径	?s iri:rel ?o .	$\rho = (\rightarrow, rel', \text{iri:关系}\ \varnothing, nil)$	长度为一的路径。
逆向路径	?s ^iri:rel ?o .	$\rho = (\leftarrow, rel', \text{iri:关系}\ \varnothing, nil)$	逆向路径（从宾语到主语）。
零次或多次路径	?s iri:rel* ?o .	$\rho = (\rightarrow, rel', \text{iri:rel}\ \varnothing, (0, nil))$	长度为零或更多的路径。
一次或多次路径	?s iri:rel+ ?o .	$\rho = (\rightarrow, rel', \text{iri:rel}\ \varnothing, (1, nil))$	长度为一或多的路径。
零次或一次路径	?s iri:rel? ?o .	$\rho = (\rightarrow, rel', \text{iri:rel}\ \varnothing, (0, 1))$	长度为零或一的路径。
序列路径	?s iri:rel1/iri:rel2 ?o .	$\rho_1 = (\rightarrow, rel1', \text{iri:rel1}, \varnothing, nil)$ $\rho_2 = (\rightarrow, rel2', \text{iri:rel2}, \varnothing, nil)$ $\epsilon = \chi_s \rho_1 \chi \rho_2 \chi_o$	由 rel1 构成的序列路径，后接 rel2。

表5. SPARQL 与 Cypher 图模式二元运算映射。

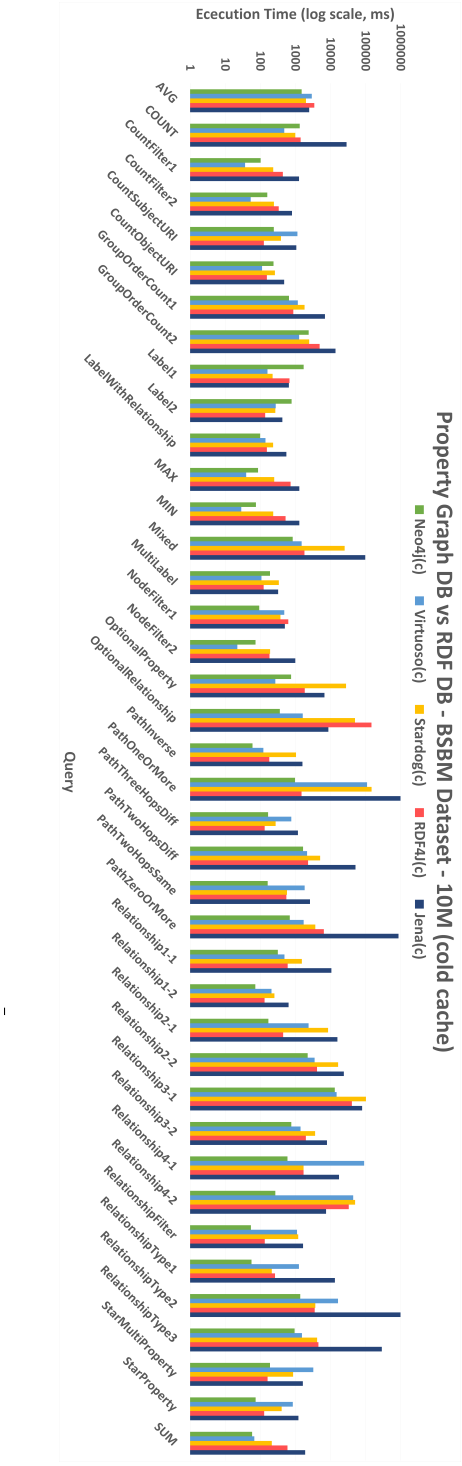
运算符	SPARQL	Cypher
AND	选择 $varlist$ 筛选 gp_1. gp_2.	匹配 cp_1, cp_2 返回 $varlist$
OPT	选择 $varlist$ 筛选 gp_1。 OPT gp_2。	匹配 cp_1 可选匹配 cp_2 返回 $varlist$
并集	选择 $varlist$ 筛选 gp_1。 并集 gp_2。	匹配 cp_1 返回 $varlist_1$ 并集 匹配 cp_2 返回 $varlist_2$
过滤器	选择 $varlist$ 筛选 gp。 过滤器 $Expr_s$。	匹配 cp 筛选 $Expr_c$ 返回 $varlist$

7.3 查询功能

实验部分的查询特征组件如表6所示。

7.4 查询语句与结果

表 7 展示了该实验部分查询翻译结果和查询执行结果。


Property Graph DB vs RDF DB - BSBM Dataset - 10M (cold cache)
Ececution Time (log scale, ms)
1
10
100
1000
10000
100000
1000000
Query
Neo4j(c)
Virtuoso(c)
Stardog(c)
RDF4J(c)
Jena(c)
AVG
COUNT
CountFilter1
CountFilter2
CountSubjectURI
CountObjectURI
GroupOrderCount1
GroupOrderCount2
Label1
Label2
LabelWithRelationship
MAX
MIN
Mixed
MultiLabel
NodeFilter1
NodeFilter2
OptionalProperty
OptionalRelationship
PathInverse
PathOneOrMore
PathThreeHopsDiff
PathTwoHopsDiff
PathTwoHopsSame
PathZeroOrMore
Relationship1-1
Relationship1-2
Relationship2-1
Relationship2-2
Relationship3-1
Relationship3-2
Relationship4-1
Relationship4-2
RelationshipFilter
RelationshipType1
RelationshipType2
RelationshipType3
StarMultiProperty
StarProperty
SUM

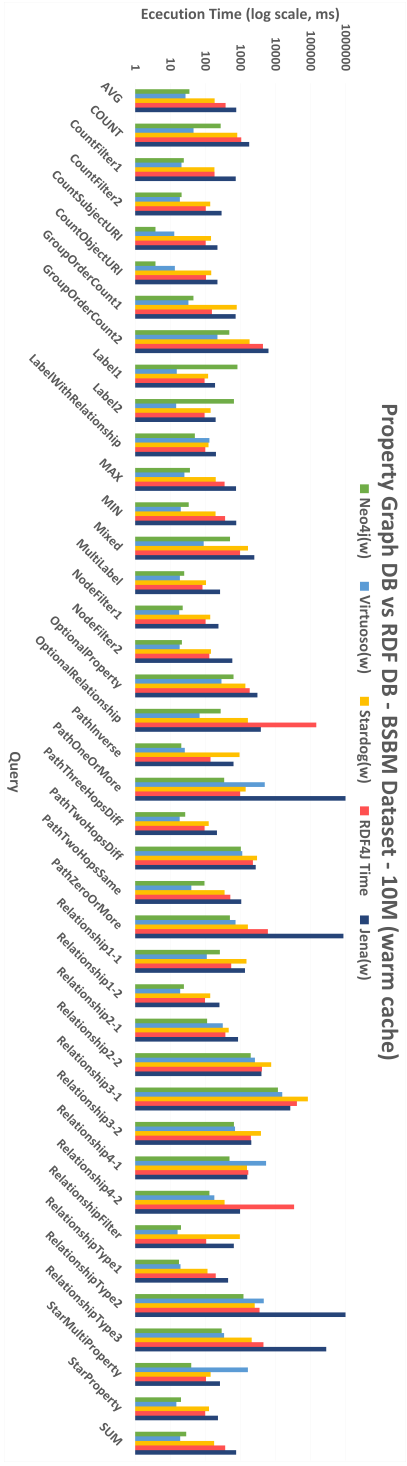

Property Graph DB vs RDF DB - BSBM Dataset - 10M (warm cache)
Ececution Time (log scale, ms)
1
10
100
1000
10000
100000
1000000
Query
AVG
COUNT
CountFilter1
CountFilter2
CountSubjectURI
CountObjectURI
GroupOrderCount1
GroupOrderCount2
Label1
Label2
LabelWithRelationship
MAX
MIN
Mixed
MultiLabel
NodeFilter1
NodeFilter2
OptionalProperty
OptionalRelationship
PathInverse
PathOneOrMore
PathThreeHopsDiff
PathTwoHopsDiff
PathTwoHopsSame
PathZeroOrMore
Relationship1-1
Relationship1-2
Relationship2-1
Relationship2-2
Relationship3-1
Relationship3-2
Relationship4-1
Relationship4-2
RelationshipFilter
RelationshipType1
RelationshipType2
RelationshipType3
StarMultiProperty
StarProperty
SUM
Neo4j(w)
Virtuoso(w)
Stardog(w)
RDF4J Time
Jena(w)

表6。查询特征组件列表

查询	聚合器	筛选器	排序	去重	限制	可选模式	#Tps	#Projs
计数	计数						2	1
计数筛选器1	计数	√(1)					3	1
计数筛选器2	计数	√(2)					4	1
计数主题URI	计数						3	1
计数对象 URI	计数						3	1
MAX	MAX	√(1)					3	1
MIN	MIN	√(1)					3	1
SUM	SUM	√(1)					3	1
AVG	AVG	√(1)					3	1
组排序1	计数	√(1)	√(1)	√	√		7	2
组排序2	计数		√(1)	√	√		4	2
节点筛选1		√(1)		√	√		4	1
节点筛选2		√(2)		√	√		4	1
关系筛选器		√(1)		√	√		6	1
标签1					√		2	1
标签2					√		3	2
带关系的标签					√		4	1
多标签		√(1)	√(2)	√	√		5	1
路径逆向		√(1)	√(1)	√	√		6	1
路径一个或多个		√(1)	√(1)	√	√		6	1
零次或多次路径		√(1)	√(1)	√	√		5	1
两跳同一节点路径		√(1)	√(1)	√	√		5	1
两跳不同节点路径			√(1)	√	√		5	2
三跳不同节点路径				√	√		3	1
一对一关系		√(1)	√(1)	√	√		5	1
关系1-2		√(2)	√(2)	√	√		5	2
关系2-1		√(1)	√(2)	√	√		8	2
关系2-2			√(2)	√	√		6	2
关系3-1		√(1)	√(2)	√	√		9	2
关系3-2		√(2)	√(2)	√	√		9	2
关系4-1		√(1)	√(2)	√	√		10	2
关系4-2		√(2)	√(2)	√	√		10	2
关系类型1				√	√		4	1
关系类型2				√	√		3	2
关系类型3				√	√		3	1
星形多属性			√(1)	√	√		14	7
星形属性		√(1)	√(1)	√	√		6	1
可选属性		√(1)	√(1)	√	√	√	6	1
可选关系		√(1)	√(1)	√	√	√	7	1
混合	MAX	√(1)	√(1)	√	√	√	6	2
总计	**40**	-	-	-	-	-	-	-

表 7.数据集描述

	BSBM-1M	**BSBM-10M**
RDF 三元组	1,000,313	10,031,929
PG节点	148,542	1,479,142
PG关系	377,201	3,821,818
PG标签	160	594

表 8。查询示例
