S2CTrans: Building a Bridge from SPARQL to Cypher* **

Zihao Zhao[1,2], Xiaodong Ge[1,2], Zhihong Shen[1(✉)], Chuan Hu[1,2], and Huajin Wang[1]

[1] Computer Network Information Center, Chinese Academy of Sciences, China
[2] University of Chinese Academy of Sciences, China
{zhaozihao,gexiaodong,bluejoe,huchuan,wanghj}@cnic.cn

Abstract. In graph data applications, data is primarily maintained using two models: RDF (Resource Description Framework) and property graph. The property graph model is widely adopted by industry, leading to property graph databases generally outperforming RDF databases in graph traversal query performance. However, users often prefer SPARQL as their query language, as it is the W3C's recommended standard. Consequently, exploring SPARQL-to-Property-Graph-Query-Language translation is crucial for enhancing graph query language interoperability and enabling effective querying of property graphs using SPARQL. This paper demonstrates the feasibility of translating SPARQL to Cypher for graph traversal queries using graph relational algebra. We present the S2CTrans framework, which achieves SPARQL-to-Cypher translation while preserving the original semantics. Experimental results with the Berlin SPARQL Benchmark (BSBM) datasets show that S2CTrans successfully converts most SELECT queries in the SPARQL 1.1 specification into type-safe Cypher statements, maintaining result consistency and improving the efficiency of data querying using SPARQL.

Keywords: RDF · Property graph · SPARQL · Cypher

1 Introduction

Currently, knowledge graph storage primarily relies on two models: Resource Description Framework (RDF) [5] and property graph [1]. RDF databases, such as Jena, maintain the former, while property graph databases, like Neo4j, manage the latter.

In general, property graph databases outperform RDF databases in graph traversal and pattern matching tasks. However, users tend to favor SPARQL for data querying, as it is a long-standing W3C recommended standard language.

* This work was supported by the National Key R&D Program of China(Grant No.2021YFF0704200) and Informatization Plan of Chinese Academy of Sciences(Grant No.CAS-WX2022GC-02)

** Zihao Zhao and Xiaodong Ge contributed equally to this paper.

The 2019 W3C Workshop on Web Standardization for Graph Data [11] called for bridging the gap between RDF and property graph query languages, allowing systems to manage data using the property graph data model while enabling users to query data with SPARQL.

Differences in semantic representation and processing logic exist between SPARQL and the property graph query language, represented by Cypher [3], making the standardization process challenging. There are three main challenges of this translation: (a) Proving the semantic equivalence of SPARQL and Cypher in graph traversal query. (b) Resolving the conflict between RDF model and property graph model storage through schema mapping and data mapping. (c) Designing the pattern matching mapping and solution modifier mapping method to translate SPARQL into Cypher.

In this study, we establish a graph relational algebra-based semantics for SPARQL and introduce S2CTrans, a provably semantics-preserving SPARQL-to-Cypher translation method. We then evaluate S2CTrans using comprehensive query features on public datasets. This paper introduces the S2CTrans framework, which offers a mapping method for pattern matching and solution modifiers, enabling the translation from SPARQL to Cypher. We perform a comprehensive query test on large-scale datasets to evaluate the performance improvement of Cypher in graph databases after translating SPARQL using the S2CTrans framework.

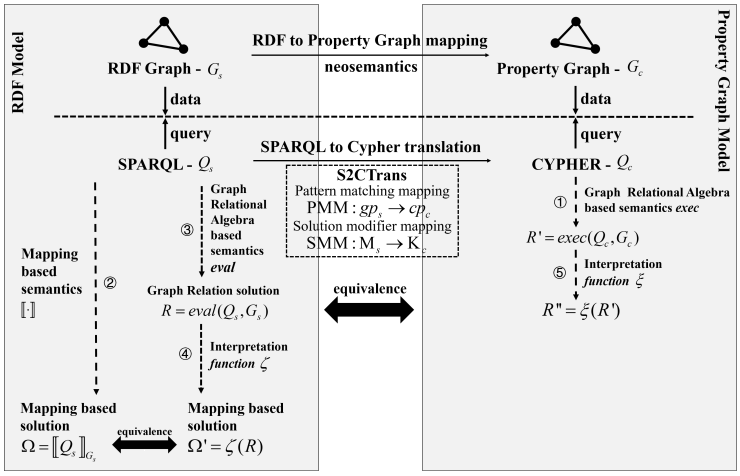

Fig. 1. Overview of SPARQL-to-Cypher translation.

The diagram of our work is illustrated in Figure 1. At the data level, we implement a syntactic and semantic transformation of RDF graph to property graph using the neosemantincs plug-in [10] developed by Neo4j Labs. This involves storing RDF triples into property graphs as nodes, relationships, and properties. At the query level, the figure illustrates the first two contributions discussed above. The dashed arrow ① represents the graph relational algebra of Cypher, while the dashed arrow ② represents the mapping-based semantics of SPARQL defined in [8]. Our contributions are represented by the dashed arrows ③, ④, and ⑤, which

define a graph relational algebra based semantics of SPARQL. Additionally, the solid arrows represent our contributions to the definition of the SPARQL-to-Cypher translation, which includes the pattern matching mapping (**PMM**) and the solution modifier mapping (**SMM**).

2 S2CTrans

2.1 System Architecture

We design and implement S2CTrans, a framework which could equivalently translate SPARQL into Cypher. S2CTrans has been open-sourced[3]. S2CTrans takes SPARQL query as input, and generates Cypher statement with the original semantics by using Jena ARQ [9] parse strategy, graph pattern matching and solution modifiers transformation strategy and Cypher-DSL [7] construction strategy. The S2CTrans works as a five-step execution pipeline:

- **Step 1:** The input SPARQL query is first parsed by the Jena ARQ module. It can check for syntax errors, verify whether it is a valid SPARQL query and generate an abstract syntax tree (AST) representation.
- **Step 2:** After obtaining the AST parsed by SPARQL, `OpWalker` is used to access the graph pattern matching part and solution modifier part from bottom up.
- **Step 3: PMM** maps the SPARQL graph pattern gp_s to the Cypher combining graph pattern cp_c, and then **SMM** maps the SPARQL solution modifiers $\mathbf{M}_s$ to Cypher clause keywords $\mathbf{K}_c$.
- **Step 4:** Cypher-DSL generates the final conjunctive traversal and constructs Cypher AST according to the pattern element type and operator priority.
- **Step 5:** Finally, the Cypher AST is rendered as a complete Cypher statement by `Renderer`. This statement can be directly queried in Neo4j with the `neosemantics` plug-in to get the result of property graph.

2.2 Pattern Matching Mapping

Graph pattern matching is the most basic and important query operation in graph query languages [2]. Due to page constraints, the graph pattern mapping algorithm is introduced in the appendix of S2CTrans-tech-report [12]. The mapping function **PMM** in the algorithm translates SPARQL graph pattern into Cypher graph pattern elements.

2.3 Solution Modifiers Mapping

After the graph pattern is obtained by **PMM** algorithm, conditions are usually added to modify the solution of graph pattern matching. Based on the semantic equivalence of SPARQL and Cypher in graph relational algebraic expressions,

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

SMM algorithm constructs a mapping table (as shown in Table 1) to implement the mapping of SPARQL solution modifiers $\mathbf{M}_s$ to Cypher clause keywords $\mathbf{K}_c$. This table summarizes graph query modification operations and the corresponding graph relational algebra, as well as the forms of SPARQL and Cypher clause construction. The variables and expressions have been mapped to graph pattern elements in **PMM** algorithm.

Table 1. A consolidated list of SPARQL solution modifiers and corresponding Cypher clause keywords.

Operation	Algebra	SPARQL Solution Modifiers - $\mathbf{M}_s$	Cypher Clause Keywords - $\mathbf{K}_c$
Selection	$\sigma_{condition}(r)$	FILTER($Expr_1$ &&(||) $Expr_2$)	WHERE $Expr_1$ *and(or)* $Expr_2$
Projection	$\pi_{x_1,x_2,...}(r)$	SELECT ?x_1 ?x_2 ...	RETURN x_1, x_2, ...
De-duplication	$\delta_{x_1,x_2,...}(r)$	SELECT DISTINCT ?x_1 ?x_2 ...	RETURN DISTINCT x_1, x_2, ...
Restriction	$\lambda_s^l(r)$	LIMIT l SKIP s	LIMIT l SKIP s
Sorting	$\varsigma_{\uparrow x_1,\downarrow x_2,...}(r)$	ORDER BY ASC(?x_1) DESC(?x_2)	ORDER BY x_1 ASC, x_2 DESC

Through **PMM** algorithm and **SMM** algorithm, we get the Cypher graph pattern and clause keywords. Cypher-DSL constructs Cypher AST according to graph pattern elements and operator precedence. Finally, we use `Renderer` to construct a complete Cypher statement.

3 Experiments

3.1 Evaluation criteria

We execute SPARQL queries on several top-of-the-line RDF databases, and execute translated Cypher queries on graph database Neo4j. We evaluate S2CTrans by the translation speed, query execution time and result consistency.

3.2 Experimental setup

Dataset: This experiment uses the Berlin SPARQL Benchmark(BSBM) dataset recommended by W3C, which consists of synthetic data describing e-commerce use cases, involving categories such as products, producers, etc. We generated 10M triples respectively by BSBM-Tools, and the corresponding property graph version is mapped using the neosemantics plug-in. The details of dataset are introduced in the appendix of S2CTrans-tech-report [12].

Query statements: We created a total of 40 SPARQL queries, covering 30 different query features. These queries were selected after systematically studying the semantics of SPARQL queries [8]. The queries are detailed in the appendix of S2CTrans-tech-report [12].

System Setup: We execute the query statements on the following databases to evaluate the performance improvement of S2CTrans: **Property Graph Database:** Neo4j v4.2.3 **RDF Databases:** Virtuoso v7.2.5, Stardog v7.6.3, RDF4J v3.6.3, Jena TDB v4.0.0 All experiments were performed on the following machine configurations: CPU: Intel Core Processor (Haswell) 2.1GHz; RAM: 16 GB DDR4; HDD: 512 GB SSD; OS: CentOS 7. In order to ensure the reproducibility of the experimental results, we provide the experimental script, dataset and query statement[4].

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

3.3 Result Evaluation

According to the evaluation criteria described above, we perform SPARQL query on RDF databases and the translated Cypher query on property graph database Neo4j on the dataset. Finally, we compare and analyze the query results, make sure the consistency. Among them, each query runs an average of 10 times to get the average value. Due to the limited space of the paper, the statements translations and query results are shown in the appendix of S2C-tech-report [12]. The average translation time of S2CTrans of 40 queries on BSBM-10M is **23.7ms**. Compared with the query time, it accounts for a small proportion. We meticulously conducted tests on datasets of various scales under both cold-start and warm-start scenarios, and all tests yielded similar results. Figure 2 presents the query execution time during the system's cold-start phase. Among most query statements, Neo4j performs better than the RDF databases. Moreover, in the queries with multi-hop paths and long relationships, the performance of Neo4j is 1 to 2 orders of magnitude higher than RDF database. The main reason is that RDF databases spend a lot of time in executing join operations and forming execution plans, while Neo4j uses index-free adjacency, which greatly improves the query efficiency.

The experiment results prove that the proposed S2CTrans is successful in equivalent translating and executing SPARQL queries. S2CTrans enables the users to query property graph by SPARQL.

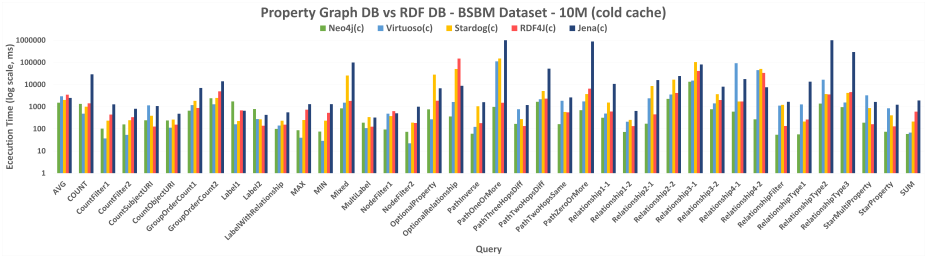

Fig. 2. Property graph database V.S. RDF database - BSBM Dataset_10M

4 Conclusion

In this paper, we introduce S2CTrans, a novel approach that supports SPARQL-to-Cypher translation. This method can convert most SPARQL statements into type-safe Cypher statements. Moreover, we employ property graph databases and RDF databases to conduct experimental evaluations on large-scale datasets, validating the effectiveness and applicability of our approach. The evaluation highlights the substantial performance gains achieved by translating SPARQL queries to Cypher queries, particularly for multiple relationship and star-shaped queries. Although S2CTrans currently has several limitations, it represents an

important step toward promoting the standardization of graph query languages and enhancing the interoperability of data and queries between the Semantic Web and graph database communities. In the future, we plan to further refine S2CTrans to support more SPARQL translations and investigate the translation from Cypher to SPARQL.

References

1. Renzo Angles. The property graph database model. In Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, volume 2100, 2018.
2. Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. Foundations of modern query languages for graph databases. ACM Comput. Surv.,50(5):68:1–68:40, 2017.
3. Francis, Nadime, et al. "Cypher: An evolving query language for property graphs." Proceedings of the 2018 international conference on management of data. 2018.
4. Jürgen Hölsch and Michael Grossniklaus. An algebra and equivalences to transform graph patterns in neo4j. In Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, volume 1558 of CEUR Workshop Proceedings, 2016.
5. Graham Klyne, Jeremy J. Carroll, and Brian McBride. Rdf 1.1 concepts and abstract syntax, W3C Recommendation, 2018.
6. József Marton, Gábor Szárnyas, and Dániel Varró. Formalising opencypher graph queries in relational algebra. In Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, volume 10509 of Lecture Notes in Computer Science, pages 182–196. Springer, 2017.
7. Gerrit Meier and Michael Simons. The neo4j Cypher-dsl. https://neo4j-contrib.github.io/Cypher-dsl/current/, 2021.
8. Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. Semantics and complexity of SPARQL. ACM Trans. Database Syst., 34(3):16:1–16:45, 2009.
9. K. Wilkinson. Jena property table implementation. In: Smart PR, ed. Proc. of the 2nd Int'l Workshop on Scalable Semantic Web Knowledge Base Systems, pages 35–46, 2006.
10. Neo4j Labs. neosemantics (n10s): Neo4j RDF & Semantics toolkit. https://neo4j.com/labs/neosemantics/, 2021.
11. Taelman, Ruben, Miel Vander Sande, and Ruben Verborgh. "Bridges between GraphQL and RDF." W3C Workshop on Web Standardization for Graph Data. W3C. 2019.
12. Zihao Zhao, Xiaodong Ge, and Zhihong Shen. S2CTrans: Building a Bridge from SPARQL to Cypher. arxiv.