A systematic review of graph query language operators for graph analytics

Felipe F. Vasconcelos[1], Renan B. Catarin[1], Fábio J. Coutinho[2], Mirian Halfeld-Ferrari[3], Cristina Dutra Aguiar[1]

[1]Universidade de São Paulo (USP), São Carlos-SP, Brazil

[2]Universidade Federal de Alagoas (UFAL), Maceió-AL, Brazil

[3]Université d'Orléans, INSA CVL, LIFO, UR 4022, Orléans, France

felipevsc@usp.br, fabio@ic.ufal.br, mirian@univ-orleans.fr, cdac@icmc.usp.br

Abstract. *Graph analytics has gained significant relevance in recent years, enhancing the expressiveness of graph data systems by enabling advanced analytics within declarative query systems. In this context, graph query language operators have emerged to bridge the gap between graph processing and declarative querying. However, to the best of our knowledge, no prior work surveys advances in this area. We conduct a systematic review of graph query language operators for graph analytics, outlining their evolution, and classifying studies into five categories: domain-specific solutions, graph pattern matching and querying, graph mining, temporal graph analytics, and fuzzy and approximate querying. Finally, we discuss limitations, trends, and research opportunities.*

1. Introduction

Graphs are used to model complex and interconnected data. However, while graph query languages support expressive pattern matching and navigational queries, advanced graph analytics remain difficult to express declaratively. In practice, such analytics are often performed using external graph processing engines with their own programming models and languages, which require users to have specialized knowledge [Lissandrini et al. 2023].

As a result, analytical tasks such as community detection, similarity search, and fuzzy queries remain difficult to perform for non-graph specialists. This creates silos in which only a limited number of experts can conduct in-depth analyses, preventing wider adoption [Bonifati et al. 2025]. A recent SIGMOD panel highlighted the importance of meeting users where they are, providing graph analytics capabilities through familiar query interfaces to improve usability and knowledge extraction [Bonifati et al. 2025].

Several studies have explored systems, architectures, and implementations for graph analytics. [Yan et al. 2017] surveys vertex-centric platforms such as Pregel, while [Sahu et al. 2020, Wang et al. 2020, Jin et al. 2022] discusses systems and their architectural approaches for graph analytics. Regarding query languages, [Angles et al. 2017] surveys graph query languages such as Cypher and SPARQL, and [Boukham et al. 2025] discusses domain-specific languages designed for analytical tasks. However, while previous work has focused on graph processing systems and frameworks, little attention has been paid to how graph analytics is being integrated into declarative query languages.

In this work, we present a systematic review focusing on graph query language extensions that introduce new clauses, referred to in this paper as *operators*, to support

graph analytics tasks. To the best of our knowledge, this is the first work focusing on graph query language extensions that bridge the gap between graph processing and declarative querying. We focus on extensions to widely used languages such as Cypher, GQL, and SPARQL, analyzing how they incorporate analytical functionality directly into declarative query environments. By studying language extensions rather than standalone systems or algorithms, we investigate how the expressiveness of query languages can be improved while preserving familiar interfaces for users.

The contributions of this work are: (i) First systematic survey of graph query language extensions designed for graph analytics; (ii) A categorization and synthesis of the proposed operators and their analytical capabilities; (iii) An analysis of the research questions defined during the planning phase of the systematic review; (iv) A discussion of research opportunities for extending query languages with advanced analytics over graph.

Our work is structured as follows: Section 2 discusses concepts; Section 3 overviews graph analytics; Section 4 presents the methodology; Section 5 synthesizes the findings; Section 6 presents the discussion, while Section 7 concludes the paper.

2. Background

2.1. Data graphs

Data graphs can be defined in various ways. Among the most common representations are labeled graphs, which associate labels with nodes and edges to model relationships (edges) between entities (nodes), and property graphs, which extend labeled graphs by associating sets of property–value pairs with both nodes and edges. In contrast, RDF represents data as triples of the form $(subject, predicate, object)$, where relationships are expressed as directed, labeled edges connecting resources, emphasizing standardization and interoperability in semantic web contexts [Besta et al. 2023].

2.2. Graph Analytics

Graph analytics are a set of techniques and algorithms for analyzing graph structures beyond basic query operations. These methods enable users to discover patterns and extract knowledge from graph data [Bonifati et al. 2025]. Graph analytics capabilities can be integrated into graph data systems in several ways, including user-defined functions, plugins, dedicated analytical frameworks, or extensions to existing query languages. In this work, we focus on the latter, specifically query language extensions that introduce operators for expressing analytical tasks declaratively. It is important to distinguish these approaches from graph processing systems such as the Pregel and GraphX platforms, which expose low-level, algorithm-oriented programming models designed for iterative graph computation or a new declarative language. In contrast, the goal of graph analytics in existing query languages is to support exploratory and ad hoc analysis through familiar, high-level declarative abstractions.

Besides the proposals found in the literature, graph database systems, like Neo4j, and libraries such as the Neo4j Graph Data Science (GDS) also provide graph analytics solutions directly integrated into DBMSs and programming environments. However, these solutions are outside the scope of this review.

2.3. Graph Query Languages

Numerous graph query languages exist, each for different data models and use cases. Cypher is one of the most widely adopted graph query languages, due to the popularity of the Neo4j ecosystem and the openCypher initiative [Besta et al. 2023, Angles et al. 2017]. SPARQL is the standard query language for RDF data, specifically designed to query semantic data, tightly integrated with ontologies, enabling features such as reasoning and inference [Ghrab et al. 2018, Angles et al. 2017].

To address the lack of a standardized graph query language, industry and academia have collaborated on a new ISO standard language called Graph Query Language (GQL)[1]. GQL is a comprehensive database language for creating, querying, updating, and deleting property graph data, drawing inspiration from languages such as Cypher and PGQL while incorporating concepts from SQL [Francis et al. 2023, Gheerbrant et al. 2024]. Although full implementations are still emerging, academic projects[2] and commercial systems are starting to adopt features aligned with the standard[3].

3. Overview of Graph Analytics

Graph analytics has gained relevance in recent years. Early research focused on graph processing systems designed to execute iterative graph algorithms at scale. Surveys such as [Elshawi et al. 2015] and [Singh and Patgiri 2016] review these systems, discussing their architectures and challenges, including handling high-degree vertices and supporting data-driven computations. In [Elshawi et al. 2015], the authors identified the lack of high-level language abstractions as a key limitation of these solutions, noting the difficulty users face when expressing graph analytics tasks in low-level programming models.

These efforts consolidated graph processing systems as a major research direction, emphasizing performance and scalability for large-scale analytical workloads. However, a turning point in the field emerged with the Dagstuhl Seminar report *"Big Graph Processing Systems"* [Bonifati et al. 2020]. Daghstul[4] is a world-class seminar for devising new ideas and research topics, identifying key research directions. The report highlighted the limitations of imperative and system-centric programming models for advanced analytics, which impose a steep learning curve and limit end-user productivity. It emphasizes the need for high-level declarative abstractions capable of expressing graph analytics tasks.

Complementing this perspective, [Sahu et al. 2020] conducted a large-scale user study showing that analytics tasks are among the most time-consuming activities in graph data management. Their findings reveal a strong demand for more expressive query languages and off-the-shelf analytical capabilities. Motivated by these challenges, recent research has increasingly explored extending graph query languages with analytical operators and higher-level abstractions. This transition marks a shift from how to compute graph analytics to how to express them declaratively, enabling users to perform advanced analytics directly within declarative query environments [Bonifati et al. 2025].

[1]We use 'graph query language' for general languages and GQL for the ISO standard

[2]`https://github.com/OlofMorra/GQL-parser` and `https://github.com/TuGraph-family/gql-grammar`

[3]`https://neo4j.com/blog/cypher-and-gql/cypher-gql-world/`

[4]`https://www.dagstuhl.de/en/seminars/dagstuhl-seminars`

4. Methodology

The proposed methodology is based on the work of [Nakagawa et al. 2017], following the planning (Section 4.1), conduction (Section 4.2), and publication (Section 5) model.

4.1. Planning phase

Objective and Research Questions. The objective of this work is to identify studies that propose query language operators for graph analytics. In this context, we define operators as high-level abstractions, such as language clauses or user-defined functions, that are applied over data through query or programming languages. Based on this objective, we formulate the following research questions: **(RQ1)** *What studies propose query language extensions to support graph analytics?*; **(RQ2)** *Which query languages are extended to support graph analytics?*; **(RQ3)** *Which graph data models are used by the proposed extensions?*; **(RQ4)** *What types of graph analytics tasks are supported by these operators?*; **(RQ5)** *What language-level operators are proposed to support graph analytics?*

Search String. From our research questions and objective, we derived a set of keywords and synonyms. These terms were combined using logical operators (AND/OR) to construct the following search string: *((“graph analytic*” OR “graph analysis” OR “graph quer*”) AND (“operator” OR “language extension*” OR “domain specific language” OR “domain-specific language*” OR “query language*”))*.

Search Engines. To conduct this systematic review, we employed three search engines, IEEEXplore[5], ACM Digital Library[6] and Elsevier Scopus[7], due to their reach and availability. Searches were performed over titles, abstracts, and author keywords using the filtering mechanisms provided by each platform.

4.1.1. Selection Criteria and Procedure

For selecting the studies, we have defined the inclusion and exclusion criteria described in Table 1. These criteria help us include studies that address at least one of the defined research questions and propose operators or language extensions for graph analytics. For instance, we exclude works that do not integrate their solutions into graph query languages, such as graph algorithms, theoretical papers and surveys. We considered studies published from 2020 onwards, as the Dagstuhl Report, discussed in Section 3, marks a shift towards discussing high-level abstractions for graph analytics, and earlier works focus on graph processing systems rather than language extensions.

Then, we define the following quality evaluation criteria to assess the selected studies: (i) Formal specification of the proposed operators, including syntax and semantics; (ii) Description of the operator implementation; (iii) Empirical evaluation of the proposed operators. For each quality criterion, we assign a score of 0 when the criterion is not met, 0.5 when it is partially met, and 1 when it is fully met.

[5]https://ieeexplore.ieee.org/Xplore/home.jsp
[6]https://dl.acm.org/
[7]https://www.scopus.com/pages/home

Inclusion Criteria		Exclusion Criteria	
#	**Criteria**	**#**	**Criteria**
IC1	Papers that propose a query language extension.	EC1	Papers that use a query language without describing it.
IC2	Papers that showcase a query language extension.	EC2	Papers that propose algorithms without integrating into a query language.
IC3	Papers whose primary focus are graph analytics.	EC3	Papers focusing on graph processing systems, graph engines or graph theory.
IC4	Papers published since 2020, or explicitly recommended by experts.	EC4	Secondary studies such as surveys, tutorials, and benchmarks.
IC5	Papers that answer at least one of the defined research questions.	EC5	Papers not published in peer-reviewed conferences or journals.
		EC6	Papers not written in English.

Table 1. Inclusion and exclusion criteria used in the study selection process.

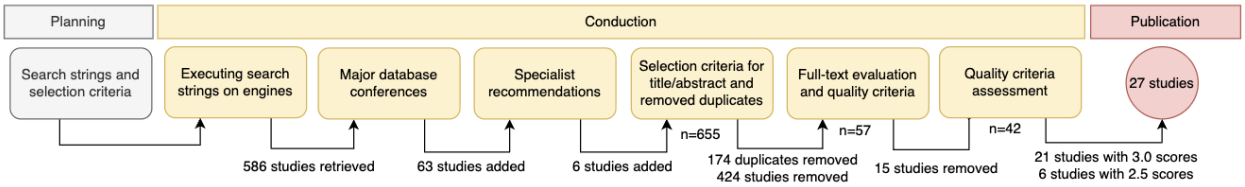

Figure 1. Methodology for conducting the systematic review.

4.2. Conduction phase

Following the definition of the selection criteria, we executed the search strings in the search engines. This search returned a total of 586 entries, 110 from ACM Digital Library, 108 from IEEEXplore and 368 from Scopus. Then, we conducted a search to include studies from major database conferences ranked A* in CORE[8] that were related to graph analytics. We collected 27 entries from VLDB, 16 from SIGMOD, and 20 from ICDE. We also included 6 papers from specialist recommendations (researchers and professors) and removed duplicate entries. This process resulted in a total of 481 papers for evaluation.

For the initial selection, we read the titles and abstracts to filter papers based on the selection criteria. From the 481 entries analyzed, 57 were selected. The selected studies were read in full to verify compliance with the selection criteria and assign the quality criteria scores. After this evaluation, we selected a final set of 27 studies that achieved a quality score higher than 2.5, meaning they fulfilled at least two quality criteria and partially fulfilled one. Figure 1 summarizes the study selection process and the distribution of papers across each step.

5. Data Synthesis and Categorization

This section synthesizes the 27 selected studies and categorizes them according to the analytical tasks they address. Table 2 summarizes our findings, presenting the analytical task, languages, and data models for each paper. The table is ordered by year, showing

[8] https://portal.core.edu.au/conf-ranks/

the evolution of the field. We organize the articles into five groups: task-specific solutions, graph pattern matching and querying, graph mining, temporal and streaming graph analytics, and fuzzy and approximate querying.

Task-specific Solutions. This group of works focuses on extending graph query languages to support specialized tasks through domain-specific operators. For example, [Ferrada et al. 2024] extends RDF with operators such as **SIMILARITY JOIN ON** and **CLUSTER BY**, as well as **TOP K DISTANCE** and **WITHIN K DISTANCE** to support *k*-nearest neighbors and range queries. Similarly, [Liu et al. 2025] incorporates vector search into TigerGraph by extending operators such as **ORDER BY**, **JOIN**, and **WHERE**, and introducing the **VECTOR_DIST** function.

Other works focus on enabling new data models and query capabilities. In this direction, [Hu et al. 2024] proposes CypherPlus to support multimodal data querying, introducing operators such as **x->y** and predicates like ::, :,! :, <: and >: for similarity and containment. Likewise, [Lu et al. 2020] enhances big knowledge graph querying by introducing new data structures and new operators, including **(?N)** for networks, **(?E)** for embedding, **(?C)** for chains, and **distance** and **Length** functions for graph-analysis. Finally, some works introduce domain-specific reasoning and transformations. For instance, [Pachera et al. 2025] enables causal analysis over property graphs through operators such as **EXTRACT**, **PROBABILITY**, and **DO-CALCULUS**, making graph databases aware of causal knowledge. [Bonifati 2025, Bonifati et al. 2024a, Bonifati et al. 2024b] propose a declarative framework for authors to specify transformation queries that convert an input property graph into a new property graph using a declarative, rule-based formalism. They propose operators such as **CONSTRUCT** and rule-based specifications compiled into openCypher to enable transformations with explicit identity and property management.

Graph Pattern Matching and Querying. This group of studies focuses on extending graph query languages with advanced pattern-matching capabilities, enabling more expressive and flexible querying over graph structures. Some approaches introduce new operators for discovering complex connections and constraints. For example, [Anadiotis et al. 2023b, Anadiotis et al. 2023a] propose the **CTP** operator to perform connecting tree pattern queries, allowing users to identify relationships between groups of nodes. Similarly, [Li et al. 2020a] targets network reachability by introducing the **EXIST** operator and the predicate "$nr : in$", enabling policy-aware reachability queries over network graphs. Complementarily, [Jamshidi et al. 2022] introduces the notion of anti-vertices to express the absence of nodes in patterns, extending the **MATCH** clause with constructs such as **(a)–(!b)–(c)** where "!" indicates an anti-vertex. Other works extend pattern matching with more expressive path and structural constraints. In this direction, [Terekhov et al. 2021] incorporates context-free path querying by extending Cypher with the **PATH PATTERN** construct and support for context-free constraints. Finally, [Li et al. 2020b] focuses on motif-based querying, introducing the primitive **M** and operators such as **MDIS**, **MPPR**, and others to support motif discovery, counting, and ranking.

Graph Mining. This group of works focuses on extracting patterns and analytical insights from graph data, often by extending query languages with mining and recursive capabilities. Some approaches explicitly target graph mining tasks, such as [Cambria et al. 2025] which introduces the **MINE GRAPH RULE** operator to extract association rules from property graphs, supported by constructs such as **GROUPING**, **HEAD**, **BODY**, **SUP-**

Paper	Task	Lang.	Model	Year
Task-Specific Solutions				
[Lu et al. 2020]	Big KG query	SPARQL	KG/RDF	2020
[Hu et al. 2024]	Multimodal querying	Cypher	PG	2024
[Ferrada et al. 2024]	Similarity & clustering	SPARQL	KG/RDF	2024
[Bonifati 2025, Bonifati et al. 2024a, Bonifati et al. 2024b]	Graph transformations	Cypher	PG	2024–25
[Liu et al. 2025]	Vector search	GSQL	PG	2025
[Pachera et al. 2025]	Causal analysis	GQL	PG	2025
Pattern Matching and Querying				
[Li et al. 2020a]	Network reachability	SPARQL	KG/RDF	2020
[Li et al. 2020b]	Motifs	Cypher	KG/RDF	2020
[Terekhov et al. 2021]	Context-free paths	Cypher	Labeled	2021
[Jamshidi et al. 2022]	Anti-vertex matching	Cypher	PG	2022
[Anadiotis et al. 2023a, Anadiotis et al. 2023b]	Tree patterns	SPARQL	PG/KG	2023
Graph Mining				
[Zhao et al. 2021]	Analytics over RDBMS	SQL	Relational	2019
[Hogan et al. 2020]	Graph queralytics	SPARQL	KG/RDF	2020
[Bamberg et al. 2025]	Recursive CTEs	SQL	Relational	2025
[Cambria et al. 2025]	Rule mining	GQL	PG	2025
Temporal and Streaming				
[Debrouvier et al. 2021]	Temporal querying	Cypher	Temporal PG	2021
[Arenas et al. 2022]	Temporal RPQs	G-Core	Temporal PG	2022
[Pacaci et al. 2022]	Persistent queries	G-Core	Streaming PG	2022
[Rost et al. 2024, Tommasini et al. 2024]	Continuous queries	Cypher	PG	2024
Fuzzy and Approximate Querying				
[Affeldt et al. 2020]	Structural preferences	SPARQL	KG/RDF	2020
[Pivert et al. 2020]	Fuzzy preferences	Cypher	PG	2020
[Liu 2022]	Fuzzy aggregation	SPARQL	KG/RDF	2022
[Aly et al. 2025]	Music retrieval	Cypher	PG	2025

Table 2. Summary of selected studies grouped by analytical category and ordered by publication year. PG = Property Graph; KG = Knowledge Graph.

PORT, and **CONFIDENCE**. Other works aim to enable graph analytics within relational systems. [Zhao et al. 2021] proposes a set of 13 graph-oriented operators, such as **UNION BY UPDATE**, **NEST**, and **RECURSIVE UNION ALL**, to unify graph and relational processing. Similarly, [Bamberg et al. 2025] extends SQL recursion by enhancing the **WITH RECURSIVE** clause with **USING KEY** in DuckDB, avoiding the accumulation of obsolete tuples during evaluation. Finally, some works emphasize recursion as a core mechanism for graph analytics. For instance, [Hogan et al. 2020] proposes an extension for graph queralytics, introducing operators such as **LET** for assigning results of queries to variables; **DO-UNTIL** loops to iteratively repeat queries; **QVALUES** for returning variables; **TIMES** and **FIXPOINT** to support iterative and analytical querying.

Temporal and Streaming Graph. This group of works focuses on enabling queries over time-evolving data. Some approaches incorporate explicit temporal semantics into

graph querying. For example, [Debrouvier et al. 2021] introduces temporal path operators such as **PATH**, **FASTESTPATH**, and **EARLIESTPATH**, along with constructs like **SNAPSHOT**, **BETWEEN**, and **WHEN** to handle temporal intervals and states. Similarly, [Arenas et al. 2022] extends the **MATCH** clause to bind variables to temporal objects, and proposes operators such as **PREV** and **NEXT** for time navigation, as well as **FWD** and **BWD** for structural traversal without altering time.

Other works address continuous and streaming scenarios. [Rost et al. 2024, Tommasini et al. 2024] propose Seraph, which introduces operators such as **WITHIN** and **EVERY** for window specification, and **EMIT** and **SNAPSHOT** for result projection, alongside mechanisms like **REGISTER QUERY** and **ON ENTERING** for continuous evaluation. Complementarily, [Pacaci et al. 2022] treats PATH as first-class citizens in streaming graphs, proposing operators such as **WINDOW**, to capture a time window, **SLIDE** to create slide intervals, **PATTERN** to find subgraph patterns, **PATH** to find paths valid at a specific point in time, as well as extensions to **FILTER** and **UNION**.

Fuzzy and Approximate Querying. This group of works extends graph query languages to support imprecision, either through fuzzy semantics or approximate matching for different domains. Fuzzy approaches enrich query expressiveness by incorporating degrees of satisfaction. For example, [Liu 2022] proposes aggregation queries on knowledge graphs, extending **COUNT, AVG, SUM, MIN, MAX**, and **GROUP BY** to allow data to be grouped using fuzzy terms. Similarly, [Aly et al. 2025] modify the MATCH clause to enable fuzzy retrieval of melodic patterns, adding operators such as **TOLERANCE** and **ALPHA** to define tolerances for pitch, duration, and gap. [Pivert et al. 2020] adds quality measurements and quality-aware preference queries to property graphs. They propose the following operators querying these measurements: **DEFINE** to specify fuzzy terms in a graph pattern; **QTPREF** to define the quality preference element and **QTAWARE** to specify the list of quality measures of interest. Approximate querying, in turn, focuses on preference-based relaxation of exact matches. In this context, [Affeldt et al. 2020] introduces the **OPTIMAL** operator, enabling users to express structural and value-based preferences to retrieve the most suitable results.

6. Discussion

6.1. Research Question Assessment

RQ1 - What studies propose query language extensions to support graph analytics? Our review identified 27 studies that propose extensions to existing query languages to support graph analytics tasks for multiple domains. These works aim to provide language constructs (*operators*) designed to encapsulate complex analytical tasks, reducing the need for users to implement algorithms manually. They span multiple paradigms, including pattern matching extensions, mining operators, temporal constructs, and approximate querying mechanisms. Figure 2 presents the number of studies per year, suggesting renewed interest in extending the query languages with analytical capabilities. This trend aligns with the discussion in Section 3, highlighting the need to improve the expressiveness of graph query languages to support advanced analytics and lower the barrier for non-specialist users, shifting the focus from how to compute to how to express analytics.

RQ2 - Which query languages are extended to support graph analytics? Figure 3 illustrates the number of papers per language (left) and the most used language in each

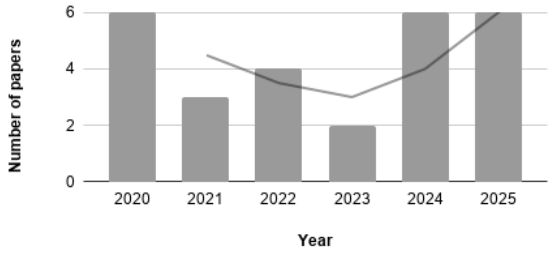

Figure 2. Number of papers per year (2020-2025).

year (right). This shows that Cypher is the most frequently extended language, followed by SPARQL. Several works adopt Cypher due to the maturity of its ecosystem and adoption in graph database systems, which lowers the barrier for proposing and evaluating extensions. More recent studies (2025) have started exploring GQL-based extensions, indicating early adoption of the standard, tying the number of works with Cypher. This suggests that future research may increasingly focus on GQL as a unifying foundation for graph analytics. These finding are consistent with RQ3, where property graphs emerge as the dominant data model, reinforcing the central role of the Cypher and GQL ecosystem.

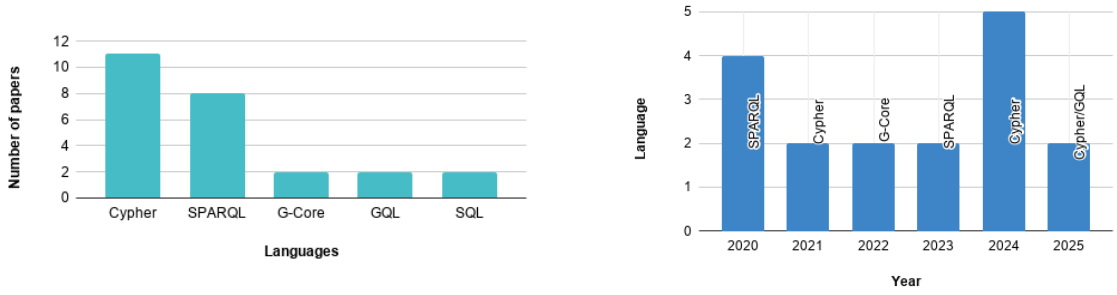

Figure 3. Distribution of query languages (left) and the most used query language per year (right).

RQ3 - Which graph data models are used by the proposed extensions? As shown in Figure 4 and Table 2, property graphs are the most used data model among the studies. 13 works rely on the standard property graph model, while 3 others modify it to support features such as temporal or streaming data. The prevalence of property graphs can be related to their adoption in widely used graph databases and query languages, such as Neo4j and Cypher, and their role as the primary data model underlying the GQL standard [Francis et al. 2023]. The second most common data model are RDF and knowledge graphs, being used by 8 studies. RDF represents graph data as triples, enabling semantic interoperability and reasoning capabilities. Knowledge graphs build upon RDF to represent structured domain knowledge. Finally, two studies operate over relational data with graph representations, enabling graph analytics within relational database systems, while one study uses labeled directed graphs.

RQ4 - What types of graph analytics tasks are supported by the proposed operators? The reviewed studies show that graph query language extensions support a diverse set of analytical tasks. Rather than being limited to traditional pattern matching, these

Figure 4. Most used data models in the studied papers (2020-2025).

extensions enable multiple dimensions of graph analytics. First, domain-specific approaches focus on embedding specialized analytical capabilities, such as similarity search, causal analysis, and vector-based querying, directly into query languages. Second, several works enhance the expressiveness of pattern matching and path querying, enabling more flexible structural queries through constructs such as CTPs and anti-vertex constraints. Third, graph mining approaches introduce capabilities for discovering patterns, clusters, and rules, often relying on recursive or iterative constructs. In addition, temporal and streaming extensions enable the analysis of evolving graphs, while fuzzy and approximate querying approaches incorporate uncertainty and preference-based semantics into query evaluation. Overall, these categories highlight a shift from purely structural querying toward richer analytical capabilities, while also revealing that most existing solutions remain task-specific, reinforcing the need for more general and composable abstractions.

RQ5 - What language-level operators are proposed to support graph analytics? The analyzed studies, described in greater detail in Section 6, propose a diverse set of language-level operators to enable graph analytics directly within query languages, increasing the expressiveness of graph queries while abstracting the complexity of the underlying analytical algorithms. Domain-specific and fuzzy/approximate approaches tend to introduce task-oriented operators that directly express analytical intent, such as SIMILARITY JOIN, VECTOR_DIST, MINE GRAPH RULE, EXTRACT, PROBABILITY, DO-CALCULUS, as well as preference and fuzzy constructs such as OPTIMAL, DEFINE, QTPREF, QTAWARE, TOLERANCE, and ALPHA. In contrast, graph mining approaches more frequently rely on lower-level constructs, such as DO, UNTIL, FIXPOINT, TIMES, and USING KEY, which enable iterative and recursive computations. Although expressive, these constructs are closer to algorithmic execution and require users to explicitly structure analytical workflows.

Graph pattern matching and querying extensions occupy an intermediate position, introducing operators such as PATH PATTERN, CTP, anti-vertex constraints in MATCH, and motif-based operators like MCOUNT and MDIS, which enhance structural expressiveness while remaining largely composable within query pipelines. Similarly, temporal and streaming approaches extend query semantics with operators such as PREV, NEXT, WINDOW, SLIDE, WITHIN, and SNAPSHOT, enabling analysis over evolving graphs. Overall, these categories reveal a spectrum of abstraction, ranging from algorithm-oriented constructs to task-oriented operators, reflecting a broader shift toward expressing graph analytics in terms of high-level declarative primitives.

6.2. Limitations

This systematic review focused on query language extensions and operators for graph analytics. This leads to limitations inherent in the objective of this search, such as excluding works that propose standalone algorithms, graph processing systems, or domain-specific frameworks that are not integrated into a query language. While this decision allows a focused analysis of language-level abstractions, it may omit relevant analytical techniques implemented outside query languages, such as higher-level formalisms and proposals.

The selection process discussed in Section 1 also presents limitations, as it applies strict inclusion and quality criteria, requiring studies to explicitly propose or formalize language extensions and provide a sufficient description of their operators. Finally, the review restricts the search to studies published from 2020 onward, motivated by the shift towards declarative graph analytics highlighted in Section 3. Although this time frame captures recent developments, earlier proposals that influenced current research directions may not be fully represented.

6.3. Research Opportunities and Tendencies

The findings of this work reveal key research opportunities in integrating graph analytics into query languages. Current proposals span multiple languages, with Cypher and, more recently, GQL emerging as primary targets, highlighting the need for extensions tailored to these ecosystems. Moreover, as most approaches remain task-specific, an important direction is the design of composable operators that unify analytical tasks and integrate seamlessly into emerging standards such as GQL. In addition, extending query languages to support domains such as embeddings, probabilistic reasoning, learning-based analytics, LLMs, and similarity search represents an important direction for richer analytics ecosystems. Despite the diversity of proposed operators, relatively few studies address execution strategies, optimization techniques, or cost models, revealing a gap between language design and system-level support. Besides this, there is still a need for comparing graph analytics proposals regarding expressiveness and capabilities, discussing the robustness of the solutions. Future work should therefore focus not only on expressiveness but also on efficient execution and optimization of within query processing engines.

7. Conclusion

This paper presents a systematic review of query language extensions that enable graph analytics. After analyzing 27 studies, we identified a growing number of proposals introducing language-level operators to increase the expressiveness of graph query languages and to support advanced analytical tasks. Our findings highlight a fundamental shift toward declarative graph analytics, where analytical tasks are expressed as query language operators rather than implemented as standalone algorithms or external systems.

The analyzed studies show that graph analytics are being embedded directly into query languages, with a focus on expressing analytical tasks rather than their underlying implementations. We also observe that Cypher and SPARQL are the most commonly extended languages, while property graphs and RDF-based knowledge graphs remain the dominant data models, with recent work beginning to target GQL as a standard. Overall, this evolution toward more expressive and analytics-oriented query languages lowers the barrier for non-expert users and opens new opportunities for developing unified and efficient graph analytics frameworks.

Acknowledgments and Use of GenAI

The first, second and last authors were supported by the São Paulo Research Foundation (FAPESP), the Brazilian Federal Research Agency (CNPq), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Finance Code 001. The work was also partially supported by the project TopOL with the financial support of the French National Research Agency (ANR). This work was done in the context of the project TREMPLIN-AIDAQ. The authors declare that GenAI tools (ChatGPT) were used exclusively for linguistic revision of this manuscript. All technical content, scientific contributions, and conclusions presented are the sole responsibility of the authors.

References

Affeldt, T., Mennicke, S., and Balke, W.-T. (2020). Preference-driven control over incompleteness of knowledge graph query answers. In *In Proc.the 12th ACM Conference on Web Science*, pages 212–220.

Aly, A., Pivert, O., and Thion, V. (2025). Fuzzy retrieval of musical scores based on melodic patterns. In *2025 IEEE FUZZ*, pages 1–6. IEEE.

Anadiotis, A. C., Manolescu, I., and Mohanty, M. (2023a). Integrating connection search in graph queries. In *2023 IEEE 39th ICDE*, pages 2607–2620. IEEE.

Anadiotis, A. C., Manolescu, I., and Mohanty, M. (2023b). More power to sparql: From paths to trees. In *European Semantic Web Conference*, pages 32–36. Springer.

Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., and Vrgoč, D. (2017). Foundations of modern query languages for graph databases. *ACM Computing Surveys (CSUR)*, 50(5):1–40.

Arenas, M., Bahamondes, P., Aghasadeghi, A., and Stoyanovich, J. (2022). Temporal regular path queries. In *2022 IEEE 38th ICDE*, pages 2412–2425. IEEE.

Bamberg, B., Hirn, D., and Grust, T. (2025). How duckdb is using key to unlock recursive query performance. In *Companion of the 2025 International Conference on Management of Data*, pages 31–34.

Besta, M., Gerstenberger, R., Peter, E., Fischer, M., Podstawski, M., Barthels, C., Alonso, G., and Hoefler, T. (2023). Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *ACM Computing Surveys*, 56(2):1–40.

Bonifati, A. (2025). Versatile property graph transformations. *In Proc.the VLDB Endowment*, 18(12):5516–5526.

Bonifati, A., Iosup, A., Sakr, S., and Voigt, H. (2020). Big graph processing systems (dagstuhl seminar 19491). *Dagstuhl Reports*, 9(12):1–27.

Bonifati, A., Murlak, F., and Ramusat, Y. (2024a). Transforming property graphs. *In Proc.the VLDB Endowment*, 17(11):2906–2918.

Bonifati, A., Ozsu, M. T., Tian, Y., Voigt, H., Yu, W., and Zhang, e. (2025). A roadmap to graph analytics. *ACM SIGMOD Record*, 53(4):43–51.

Bonifati, A., Ramusat, Y., Murlak, F., Fejza, A., and Echahed, R. (2024b). Dt-graph: declarative transformations of property graphs. *In Proc.the VLDB Endowment (PVLDB)*, 17(12):4265–4268.

Boukham, H., Younsi Dahbi, K., and Chiadmi, D. (2025). Domain-specific languages for algorithmic graph processing: A systematic literature review. *Algorithms*, 18(7):445.

Cambria, F., Invernici, F., Bernasconi, A., and Ceri, S. (2025). Mine graph rule: A new gql operator for mining association rules in property graph databases. *The VLDB Journal*, 34(4):54.

Debrouvier, A., Parodi, E., Perazzo, M., Soliani, V., and Vaisman, A. (2021). A model and query language for temporal graph databases. *The VLDB Journal*, 30(5):825–858.

Elshawi, R., Batarfi, O., Fayoumi, A., Barnawi, A., and Sakr, S. (2015). Big graph processing systems: State-of-the-art and open challenges. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 24–33. IEEE.

Ferrada, S., Bustos, B., and Hogan, A. (2024). Similarity joins and clustering for sparql. *Semantic Web*, 15(5):1701–1732.

Francis, N., Gheerbrant, A., Guagliardo, P., Libkin, L., Marsault, V., Martens, W., Murlak, F., Peterfreund, L., Rogova, A., and Vrgoc, D. (2023). A researcher's digest of gql. In *The 26th ICDT, 2023*, pages 1–1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Gheerbrant, A., Libkin, L., Peterfreund, L., and Rogova, A. (2024). Gql and sql/pgq: Theoretical models and expressive power. *arXiv preprint arXiv:2409.01102*.

Ghrab, A., Romero, O., Jouili, S., and Skhiri, S. (2018). Graph bi & analytics: current state and future challenges. In *In DAWAK 2018*, pages 3–18. Springer.

Hogan, A., Reutter, J. L., and Soto, A. (2020). In-database graph analytics with recursive sparql. In *International Semantic Web Conference*, pages 511–528. Springer.

Hu, C., Zhao, Z., Mao, A., and Shen, Z. (2024). A model and query language for multi-modal hybrid query. In *In Proc.the 36th International Conference on Scientific and Statistical Database Management*, pages 1–4.

Jamshidi, K., Mariappan, M., and Vora, K. (2022). Anti-vertex for neighborhood constraints in subgraph queries. In *In Proc.the 5th ACM SIGMOD GRADES-NDA Workshop*, pages 1–9.

Jin, H., Qi, H., Zhao, J., Jiang, X., Huang, Y., Gui, C., Wang, Q., Shen, X., Zhang, Y., Hu, A., et al. (2022). Software systems implementation and domain-specific architectures towards graph analytics. *Intelligent Computing*.

Li, W., Peng, P., Qin, Z., and Zou, L. (2020a). Nrgqp: A graph-based query platform for network reachability. In *International Conference on Web Information Systems Engineering*, pages 55–63. Springer.

Li, X., Cheng, R., Najafi, M., Chang, K., Han, X., and Cao, H. (2020b). M-cypher: A gql framework supporting motifs. In *In Proc.the 29th ACM CIKM*, pages 3433–3436.

Lissandrini, M., Hose, K., and Pedersen, T. B. (2023). Example-driven exploratory analytics over knowledge graphs. In *EDBT*, pages 105–117.

Liu, S., Zeng, Z., Chen, L., Ainihaer, A., Ramasami, A., Chen, S., Xu, Y., Wu, M., and Wang, J. (2025). Tigervector: Supporting vector search in graph databases for advanced rags. In *Companion of the 2025 International Conference on Management of Data*, pages 553–565.

Liu, Y. (2022). A fuzzy query method oriented to knowledge graph. In *Journal of Physics: Conference Series*, volume 2221, page 012051. IOP Publishing.

Lu, R., Wang, C., Huang, X., and Zhang, S. (2020). B-sparql: A typed language for querying the big knowledge. In *2020 IEEE ICKG*, pages 173–180. IEEE.

Nakagawa, E. Y., Scannavino, K. R. F., Fabbri, S. C. P. F., and Ferrari, F. C. (2017). *Revisão sistemática da literatura em engenharia de software: teoria e prática*. Elsevier Brasil.

Pacaci, A., Bonifati, A., and Özsu, M. T. (2022). Evaluating complex queries on streaming graphs. In *2022 IEEE 38th ICDE*, pages 272–285. IEEE.

Pachera, A., Palmiotto, M., Bonifati, A., and Mauri, A. (2025). What if: Causal analysis with graph databases. In *51st VLDB*, volume 18, pages 4009–4016. VLDB Endowment.

Pivert, O., Scholly, E., Smits, G., and Thion, V. (2020). Fuzzy quality-aware queries to graph databases. *Information Sciences*, 521:160–173.

Rost, C., Tommasini, R., Bonifati, A., Della Valle, E., Rahm, E., Hare, K. W., Plantikow, S., Voigt, H., and Selmer, P. (2024). Seraph: Continuous queries on property graph streams. In *EDBT/ICDT 2025 Joint Conference*.

Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., and Özsu, M. T. (2020). The ubiquity of large graphs and surprising challenges of graph processing: extended survey: S. sahu et al. *The VLDB journal*, 29(2):595–618.

Singh, D. K. and Patgiri, R. (2016). Big graph: Tools, techniques, issues, challenges and future directions. In *CS & IT Conference Proceedings*, volume 6. CS & IT Conference Proceedings.

Terekhov, A., Pogozhelskaya, V., Abzalov, V., Zinnatulin, T., and Grigorev, S. V. (2021). Multiple-source context-free path querying in terms of linear algebra. In *EDBT*, pages 487–492.

Tommasini, R., Rost, C., Bonifati, A., Della Valle, E., Rahm, E., Hare, K. W., Plantikow, S., Selmer, P., and Voigt, H. (2024). Property graph stream processing in action with seraph. In *Companion of the 2024 International Conference on Management of Data*, pages 492–495.

Wang, R., Yang, Z., Zhang, W., and Lin, X. (2020). An empirical study on recent graph database systems. In *International Conference on Knowledge Science, Engineering and Management*, pages 328–340. Springer.

Yan, D., Bu, Y., Tian, Y., and Deshpande, A. (2017). Big graph analytics platforms. *Foundations and Trends in Databases*, 7(1-2):1–195.

Zhao, K., Su, J., Yu, J. X., and Zhang, H. (2021). Sql-g: Efficient graph analytics by sql. *IEEE TKDE*, 33(5):2237–2251.