Summary Chart of Cypher, PGQL, and G-Core

Title Summary Chart of Cypher, PGQL, and G-Core
Authors Stefan Plantikow, U.S. National Expert
Date 2018-05-14

Contents

1. Introduction

The property graph data model is a versatile, application-oriented data model for working with graphs. Property graphs connect nodes via relationships. Both nodes and relationships (called entities) are categorized using multiple labels and may have associated attributes (called properties). Two nodes can be connected via multiple relationships and nodes may also be connected via a relationship to itself.
There is an increasing interest in declarative query languages for the property graph data model. This discussion paper compares Cypher 9, PGQL 1.1, and G-Core languages for property graph querying.

Cypher is the original declarative query language for the property graph data model that was designed by Neo4j. Syntactically, Cypher follows SQL by structuring queries as a tree of clauses.
Cypher introduced the idea of using visual "Ascii-Art" syntax to express the topological constraints of a pattern. This syntax made pattern matching functionality approachable, firmly established declarative pattern matching in the market, and was a great success with real-world users.
Cypher is used in five commercial products and is the most widely implemented property graph query language. Noteworthy, Cypher is the only language discussed here that features both a data manipulation and an index and a constraint definition language. Cypher is actively developed in an open process by the openCypher implementer's group.
The current, wholly implemented version of Cypher is Cypher 9; more advanced features like support for path patterns, working with multiple graphs (including returning graphs and graph construction) are being developed for Cypher 10 and have been partially implemented in Cypher for Apache Spark.

PGQL is a declarative property graph query languages designed by Oracle Corporation. It was inspired by both SQL and Cypher and features a similar "Ascii-Art" pattern syntax. PGQL was the first property graph query language that supported regular path patterns. Syntactically, PGQL aims to follow SQL syntax where possible.
The current version of PGQL is PGQL 1.1 and has been implemented in Oracle PGX.

G-Core is the research query language designed by the LDBC (Linked Data Benchmarks Council) query language task force. G-Core features "Ascii-Art" pattern syntax from Cypher, path patterns from PGQL, and explores more advanced features like graph returning queries as well as graph construction.
The current version of G-Core is published in an upcoming SIGMOD paper and has not yet been implemented (a research grade implementation is currently being developed).

This paper discusses the ongoing convergence between the syntax and the semantics of the features of these three languages, provides a feature comparison chart in section 5, and argues for their potential unification.

2. References

[CYPHER2018]	Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor, *"Cypher: An Evolving Query Language for Property Graphs"*, SIGMOD 2018, `http://homepages.inf.ed.ac.uk/libkin/papers/sigmod18.pdf`
[CYPHER9]	Cypher Language Group, *"Cypher Query Language Reference Version 9"*, `https://s3.amazonaws.com/artifacts.opencypher.org/openCypher9.pdf`
[CYPHER10]	Cypher Language Group, *"Cypher 10 Improvement Proposals",* `https://github.com/opencypher/openCypher/labels/cypher10`
[PGQL2016]	Oskar van Rest, Sungpack Hong, Jinha Kim, Xuming Meng, Hassan Chafi, *"PGQL: a Property Graph Query Language",* GRADES 2016
[PGQL11]	Oracle Corporation, PGQL Specification 1.1 `http://pgql-lang.org/spec/1.1/`
[GCORE2018]	Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter Boncz, George Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan Sequeda, Oskar van Rest, and Hannes Voigt, *"G-CORE A Core for Future Graph Query Languages"*, SIGMOD 2018, `https://arxiv.org/pdf/1712.01550.pdf`

3. Language features

3.1. Shared Core

All three languages share a common core consisting of

- The central means of data access is pattern-matching using "Ascii-Art" syntax, either from an implicit default graph or from an explicitly specified graph from the system catalog.
- Data is returned as a stream of records via projection or a single graph via graph construction.
- Query syntax follows the syntactic tradition of SQL, i.e. queries describe a tree of clauses, each introduced by a distinct keyword and optionally further qualified by sub-clauses, expressions and patterns.
- All three languages support single-argument operator clauses (like matching, or projection) as well as multi-argument operator clauses (like graph union).

To give an example of this common core, below we show the same query in all three languages:

Cypher 9	`FROM languageGraph`
	`MATCH (a:Engineer)-[:LIKES]->(l:Language)`
	`WHERE (l)-[:SUPPORTS]->(:Feature {name: "Pattern Matching"})`
	`RETURN a.name, l.name`

PGQL 1.1	`SELECT a.name, l.name FROM languageGraph`
	`MATCH (a:Engineer)-[:LIKES]->(l:Language)`
	`WHERE EXISTS (`
	`  SELECT *`
	`  MATCH (l)-[:SUPPORTS]->(:Feature {name: "Pattern Matching"})`
	`)`

G-Core	`SELECT a.name, l.name`
	`MATCH (a:Engineer)-[:LIKES]->(l:Language) ON languageGraph`
	`WHERE EXISTS (`
	`  CONSTRUCT ()`
	`  MATCH (l)-[:SUPPORTS]->(:Feature {name: "Pattern Matching"})`
	`  ON languageGraph`
	`)`

The languages currently differ - although with substantial foreseeable convergence - on the following points:

- Query structure and clause order
- Scope of the language design
- Path variables
- Path patterns
- Supported pattern matching semantics (match cardinality, morphisms, and length restrictions)
- Graph construction

3.2. Language design scope

All three languages support basic pattern matching with enumeration and reachability queries, as well as projection, sorting, filtering, and aggregation of matches.
Cypher is distinct in that it also supports a full data manipulation language (DML) for creating, updating, and deleting nodes and relationships, as well as an "upsert" feature (called `MERGE`) that allows either matching or uniquely creating whole patterns. The future addition of DML capabilities have been discussed for both PGQL and G-Core but no specification of such features have been put forth so far.
Cypher also supports a basic data definition language (DDL) by providing pattern-based constraints. Neo4j additionally features pattern-based indices as a vendor-specific extension.

3.3. Query structure and clause order

The languages differ in the chosen syntactic order of clauses. PGQL follows SQL in that it interprets clauses from bottom-to-top (i.e. the first `SELECT` clause describes the returned data). G-Core prefers bottom-to-top order but has been designed with optionally supporting top-to-bottom order. Cypher uses top-to-bottom-order as a natural way of expressing sequences of DML statements that also fits very well with how graph queries gradually build more complex sets of variable bindings. It can be envisioned that a future language might support both bottom-to-top and top-to-bottom clause orders.

3.4. Path variables

Languages differ slightly in their treatment of path variables. Cypher supports immutable path values while G-Core supports mutable, persistent path objects with mutable attributes. PGQL plans to support path variables in the future but at this time has not specified their semantics.

3.5. Path patterns

PGQL pioneered support for full regular path patterns. Both Cypher and G-Core have proposed similar features. A close reading of existing path pattern proposals shows a high degree of semantic and syntactic convergence.

3.6. Pattern matching semantics

Pattern matching semantics describe how patterns are interpreted against a graph in order to generate matches (variable bindings). Different semantics may be can be distinguished in terms of three influencing factors:

- **Cardinality of matches**: enumeration of all matches vs reachability queries vs top-k queries
- **Path length restriction**: shortest paths, cheapest paths, all paths
- **Pattern morphism**:
    - homomorphism (no restriction on matches)
    - isomorphism (no node matched twice in a match)
    - edge-isomorphism (no edge matched twice in the same match)

All three languages support reachability and enumeration queries, as well as shortest path queries. PGQL and G-Core also support top-k queries and cheapest paths. Recent proposals for Cypher also discuss the addition of such features to the language.
Of particular interest is the choice of morphism which represents a trade-off between ease-of use, complexity-theoretic tractability, and avoiding infinite result sets. Cypher uses edge-isomorphism by default which helps avoid the generation of infinite result sets for all path enumeration queries. Cypher also supports homomorphism for rigid patterns and shortest path matching. PGQL and G-Core use homomorphism by default. PGQL uses reachability semantics for path patterns while G-Core advocates shortest path semantics by default for path patterns. There is convergence in so far that both PGQL and G-Core are suggesting additional features for emulating isomorphism while a recent proposal for Cypher suggests the addition of configurable pattern matching semantics to the language.

3.6. Graph construction and views

There are proposals for all three languages for the addition of graph construction, i.e. the dynamic construction of new graphs from pattern matches derived from existing graphs. Graph construction and the ability to return graphs makes property graph query languages closed under graphs - an important theoretic property that enables query composition. Graph construction is also a good mechanism for the construction of (potentially updatable) views.
G-Core and PGQL proposals favour defining graph construction by giving a set of construction patterns while current proposals for Cypher follow a more layered approach that re-uses Cypher's existing DML syntax. There is clearly convergence on the need to add graph construction capabilities to property graph query languages.

4. Comparison chart

The following table presents an overview of the discussed feature sets of Cypher 9, PGQL 1.1, and G-Core.

Feature	Cypher 9	PGQL 1.1	G-Core
Basic (rigid) Pattern Matching			
• *Cypher-style "Ascii-Art" Syntax*	[X]	[X]	[X]
• *Enumeration: Find ALL matches*	[X]	[X]	[X]
• *Reachability: Find ANY match*	[X]	[X]	[X]
• *Optional match*	[X]	[X]	[X]
Regular Path Pattern Matching			
• *Variable length path variables* *- Reachability: ANY path* *- Enumeration: ALL paths* *- Enumeration: ALL shortest* *- Enumeration: ALL cheapest* *- Top k shortest/cheapest*	Path values [X] [X] Edge-Iso. [X] [/] Cypher 10 [/] Cypher 10	[/] In Paper [X] [-] [/] Planned [/] Planned [?]	Path objects [X] [-] [X] [X] [X]
• *Full RPQs with macros* *- Label conjunctions* *- Label alternatives* *- Label concatenation* *- Label grouping*	[/] Cypher 10 [X] [X] Edges only [/] Cypher 10 [/] Cypher 10	[X] [X] [X] [X] [X]	[X] [X] [X] [X] [X]
Language			
• *Project, filter, sort, group* • *Multi-part projection queries*	[X] [X]	[X] [-]	[X] [-]
• *Basic expression syntax* • *Three valued logic* • *GraphQL style map projections*	[X] [X] [X]	[X] [X] [-]	[X] [-] [-]
Updates and Construction / DML			
• *Creation and deletion of entities*	[X]	[-]	[-]
• *Match-or-create patterns*	[X]	[-]	[-]
• *Updating properties and labels*	[X]	[-]	[-]
• *Graph construction and aggregation*	[/] Cypher 10	[-]	[X]
Schema / DDL			
• *Pattern-based constraints* • *Pattern-based indices (vendor ext.)*	[X] [X]	[-] [-]	[-] [-]
Implementations			
• *Commercial products / Total* • *Total*	5 10	1 1	0 research impl. in development

5. Conclusion

While the presented languages still differ in their syntax, semantics, and the provided feature set we argue that they all are following the same underlying language design approach that follows in the syntactic tradition of SQL and develops the idea of pattern matching introduced by Cypher.
Furthermore, there seems to be clear convergence and an increasing agreement on the set of desirable features for a future joint query language that incorporates the ideas from Cypher, PGQL, and G-Core into a next generation property graph query language.

6. An ITI and ISO contribution from Neo4j Inc.

This contribution to DM32.2 is a Deliverable under the terms of clause 2.2.1 of the Agreement for Membership in the InterNational Committee for Information Technology Standards (“INCITS”), a Division of the Information Technology Industry Council (“ITI”) to which Neo4j Inc. is a party.

It is also a contribution to the ISO/IEC JTC1/SC32 Working Group 3 (Database Languages), under the terms of POCOSA. This document is neither a standard nor a draft of a standard.