The GQL Manifesto

Title	The GQL Manifesto
Author	Alastair Green, Individual Expert, Neo4j Inc.
Status	Discussion Paper
Date	Date of original publication, 13 May 2018, at https://gql.today
	Referenced in **DM32.2-2018-0087**
	Date of submission to DM32.2, 5 July 2018

1. References

[DM32.2 2017-0026]	“(Property) Graphs & SQL/Graph”, Jan Michels, Oskar Van Rest, Sungpack Hong, Jinha Kim, Matthew Perry, Zhe Wu, Hassan Chafi, January 2017
[WG3:OKJ-001]	Draft Minutes of CPT meeting of SC32/WG3 January 2017, ISO

2. Text of the Manifesto

Property graph data has a big presence

Amazon Neptune, Oracle PGX, Neo4j Server, SAP HANA Graph, AgensGraph (over PostgreSQL), Azure CosmosDB, Redis Graph, SQL Server 2017 Graph, Cypher for Apache Spark, Cypher for Gremlin, SQL Property Graph Querying, TigerGraph, Memgraph, JanusGraph, DSE Graph … it's hard to keep up.

And an even bigger future

Thousands of applications already use the rich, intuitive graph data model. The ability to express, find and extract complex relationships and patterns; to execute graph algorithms; to register transactions; and to perform graph analysis is proving its value every day, in every domain of business and science. And adoption is accelerating.

Relational data has SQL

SQL is one of the key underpinnings of modern information technology. We need a declarative query language for the powerful – and *distinct* – property graph data model, to play a similar role.

Many graph databases and services are not features of relational systems, but native, optimized implementations. And many graph queries, when implemented over relational systems, do not need to be tied to tables on the surface. There is a big role for graph extensions to SQL, but they are unlikely to be the leading edge of graph querying.

Property graph data needs GQL

Like SQL, the new GQL (Graph Query Language) needs to be an industry standard.

Different languages for different products help no one. An international standard will let the whole graph data market grow, to the mutual benefit of all vendors and all users. One language, one skill set. A common query language focuses support around data modelling, ETL and visualization tools for graph data, and portable queries mitigate vendor lock-in.

But GQL also needs to be tuned and agile to meet the needs of the expanding property graph data industry. It should work with SQL, but it should not be confined by SQL. GQL would be a language that complements the traversal API of Apache Tinkerpop's Gremlin as well as

SPARQL for RDF triple-stores. This results in better choices for developers, data engineers, data scientists, CIOs and CDOs alike.

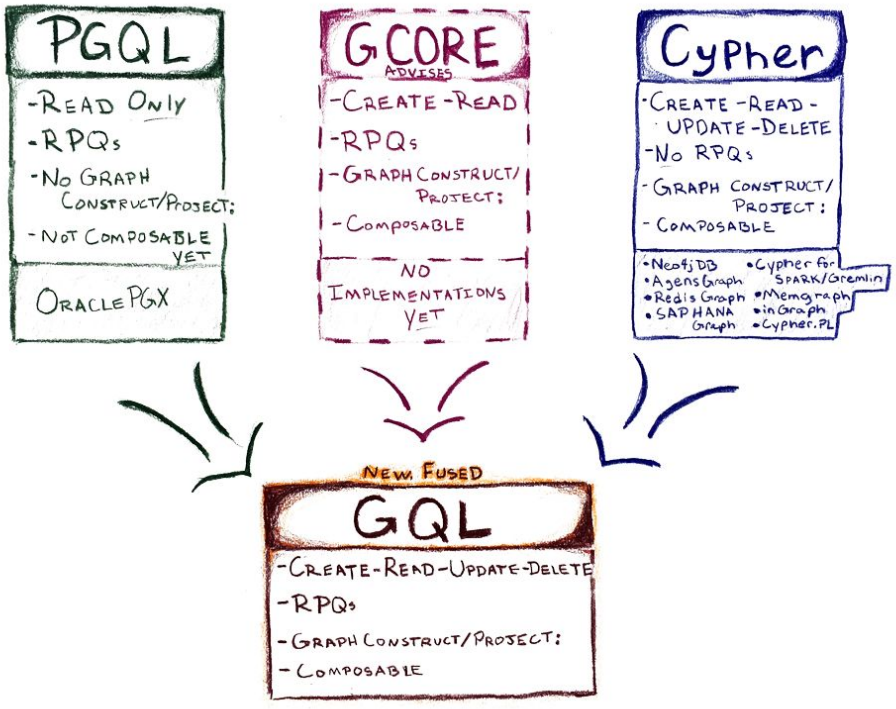
Three languages

Right now there are three property graph query languages that are closely related.

We have **Cypher** (from Neo4j and the **openCypher** community). We have **PGQL** (from Oracle). And we have **G-CORE**, a research language proposal from Linked Data Benchmark Council (co-authored by world-class researchers from the Netherlands, Germany, Chile, the U.S, and technical staff from SAP, Oracle, Capsenta and Neo4j).

You'd be hard pressed to tell them apart for many common graph queries. They have very similar data models, syntax and semantics. Each one is ahead of the game or behind the game in one way or another.

Their authors *share* many ambitions for the next generation of graph querying such as a composable graph query language with graph construction, views and named graphs; and a pattern-matching facility that extends to regular path queries, where regular expressions find and retrieve patterns even more concisely and flexibly.



Three into one should go

My name is Alastair Green. I help to organize work on graph query language standards and research at Neo4j, Inc., working with talented engineers, language architects and standards specialists within our company and among collaborators in industry and academia.

Sponsored by our CEO, Emil Eifrem, and our VP of Products, Philip Rathle, the Neo4j team is advocating that the database industry and our users *get away* from two or three blurred photocopies of the same ideas, and *get together* to define and standardize **one**language.

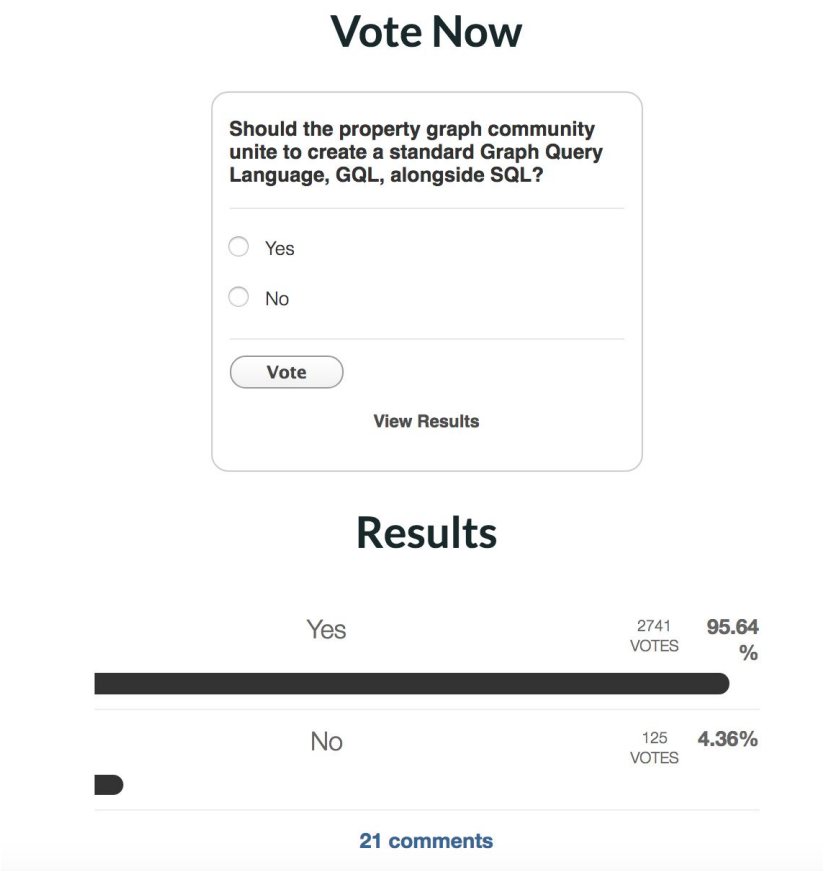
Bringing PGQL, G-CORE and Cypher together, we have a running start. Two of them are industrial languages with thousands of users, and combined with the enhancements of a research language, they all share a common heritage of ASCII art patterns to match, merge and create graphs.

One property graph query language, one GQL

We'll be working with anyone who wants to help create GQL as a new open industry standard. We'd love it to be an ISO standard alongside SQL. Or perhaps it will end up in another venue.

But that's not the most important thing. A technically strong standard, with strong backing among vendors and users: that's what matters most. So we'll be appealing for your public, vocal support. If there is a will, there will be a way.

3. Results of public web vote on the Manifesto, as of 5 July 2018



4. GQL and SQL PGQ

The GQL Manifesto is a public document, freely available on the Web. It proposes the fusion of three kindred languages: openCypher, PGQL and G-CORE to form a single property graph query language, GQL, which would stand alongside SQL.

This is a similar concept to the idea of a “stand-alone” property graph query language, advocated in [DM32.2 2017-0026] by a group of Oracle authors at the start of 2017.

The proposal for a New Work Item for such a stand-alone language was endorsed by the Cape Town meeting of WG3 in January 2017 [WG3:OKJ-001], alongside a proposal to seek approval from SC32 for a parallel project split for SQL Property Graph Querying.

Unfortunately, and mistakenly in my view, the idea of a NWIP for stand-alone property graph query language, was dropped at the June 2017 Okayama meeting of WG3, leaving only the proposal for a SQL Property Graph Query project split.

We in Neo4j were unaware of the Oracle paper and of the decisions taken in Cape Town at the time when I wrote The GQL Manifesto. This “pre-history” had never been drawn to the attention of the SQL PGQ Ad Hoc working group or to Neo4j colleagues individually by anyone in DM32.2 or WG3 or the Ad Hoc prior to the publication of the Manifesto.

Nor were we in Neo4j informed of any discussion leading to the Okayama reversal of the Cape Town decision on stand-alone graph querying, despite having been finally invited to join the Ad Hoc for SQL PGQ in early May 2017, after its second meeting, an invitation which we immediately accepted.

We in Neo4j Inc. agree with the Oracle authors of [DM32.2 2017-0026] on the content and relationship of a stand-alone GQL, and property graph extensions to SQL.

> “rather than reinventing the wheel, the SQL-embedded language should reuse as much as possible from the yet to-be standardized stand-alone graph query language. Therefore, it seems imperative to develop these two approaches as closely together as possible.”

In the public Manifesto I did not dwell on SQL PGQ as a potential input for two reasons: first, the inheritance from PGQL and Cypher has been explicit in the work of the Ad Hoc; second because the work of the Ad Hoc and of DM32.2 is not public, whereas the three languages openCypher, PGQL and G-CORE are documented in public open-source materials.

It is therefore worth emphasizing three additional points:

1. SQL PGQ innovations arising from the work of the Ad Hoc are a “fourth natural input” into the proposed GQL language.

2. GQL should incorporate a relevant profile of SQL/Foundation with respect to basic datatypes, expressions, ternary logic etc., to maximize adoption and interoperability.

3. SQL PGQ pure graph querying features should be viewed as a subset of GQL (whereas features that interface SQL to graph queries are better viewed as a part of SQL, and not of a stand-alone language).

5. An ITI 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.