openCypher, a path to GQL

Valerio Malenchino, PM for database languages, Neo4j

Prelude

- openCypher project launched in 2015
  - To give the graph database user's community a common language
- In a few years, several implementations are released
- In 2019, ISO announces a new project for GQL, with strong support and participation from openCypher community
  - evolution of openCypher language in practice stops
  - but implementations of openCypher keep growing
- In April 2024, GQL becomes an official ISO standard
- … and here we are

Cypher and GQL

Due to the common and intertwined history, they are pretty similar

The poster Cypher query:

```
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie)
WHERE a.name = 'Tom Hanks'
RETURN m.title
```

could be the poster GQL query!

But there are differences too.

Looking at GQL with openCypher glasses

- From openCypher point of view, features can broadly be classified into:
  - features already in openCypher
  - syntactic/minor variations (mostly) of SQL origin (e.g INSERT vs CREATE)
  - additions to existing openCypher features
    - *minor*, like simple CASE statement
    - *major* improvements, like Graph Pattern Matching extensions

- openCypher also has features that are not in GQL (yet?)

A road to GQL

- The future of property graph languages is GQL

- To make that the present, GQL needs
  - good implementations **ready for production**
  - keen users
- openCypher has both

- Helping the openCypher ecosystem to transition to GQL will help making GQL a success.

Make openCypher ecosystem excited about GQL

- For implementers, it is important to:
  - support new use cases
    - complex/expensive extensions must be justified by users benefits.
  - preserve existing development investment
- For users, it is important to:
  - address use cases that are hard/impossible in openCypher
  - preserve investment in existing queries and skills

- For example, extensions to Graph Pattern Matching are a better motivator than renaming `toUpper()` to `upper()`. We need to do both, but ordering/mix is important.

In practice: openCypher new role

- Stop being a forum for language discussions
  - the industry now has a better forum: ISO
  - ISO is where the language standard decisions are taking place

- Become a GQL implementation project
  - make it easy for Cypher users to start using GQL (in production)
  - make it easy for openCypher implementers to support GQL

In practice: GitHub branches

- Branch the openCypher GitHub repo

- 'openCypher 9.1' will be the existing openCypher
  - Implementers can stick to it if they want to

- 'main' will be the migration of openCypher towards GQL: starts from openCypher 9.1 and adds GQL feature in an 'openCypher-friendly' way

In practice: Artifacts

- New openCypher improvement proposals(CIPs)
  - translation from GQL specs
  - try to keep implementers and users excited about GQL with right mix/order
  - no new non-GQL features in openCypher!
- New openCypher grammar
  - non-GQL existing features clearly marked as Cypher extensions
  - openCypher features with a suitable GQL replacement will be marked
    - to help implementing a 'GQL-strict' mode.
- TCK starting from openCypher scenarios and extended
- Periodic releases
  - about every six months
  - versioned using calendar versioning

In practice: Timeframe

- First release coming soon (Sep/Oct):
  - text BNF Grammar that includes Cypher 9.1 + 2-3 GQL features (initial focus on adding pattern matching and trivial synonyms such as INSERT)
  - CIPs for new inclusions
  - initial version of TCK (initially just Cypher 9)
- Steady stream of additions: est. 2-3 CIPs per quarter
- TCK scenarios for new inclusion will be done later
- Participation invitation:
  - review for conformance
  - TCK scenarios for released grammar

Thank you!