Towards GQL 1

Status report on the upcoming ISO/IEC graph query language standard

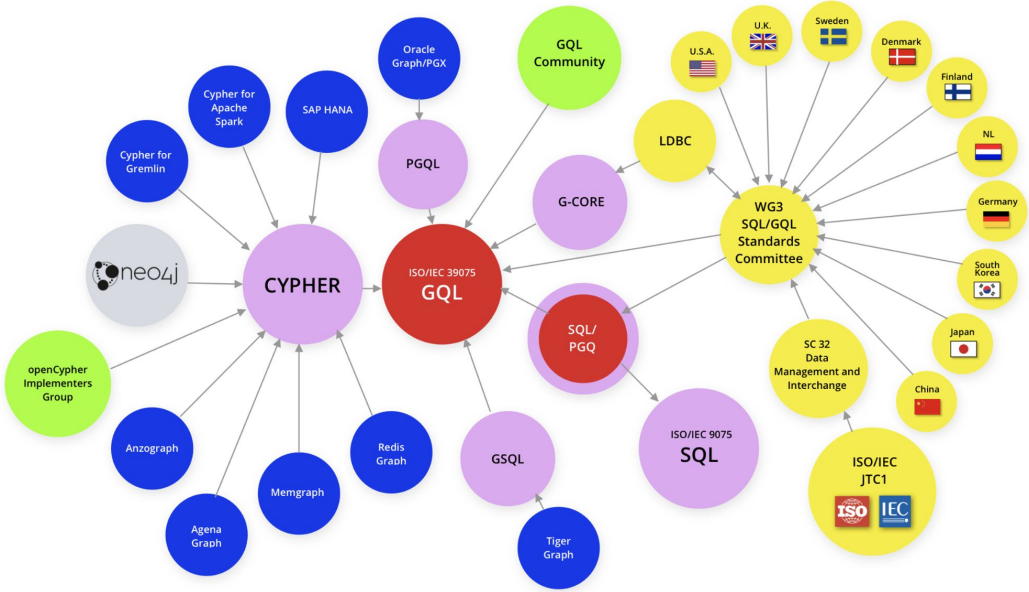
LDBC TUC, August 2021

Stefan Plantikow, Principal GQL Editor, Neo4j

Topics

- What is GQL?
- A taste of GQL
- Progress

Timelines and process are covered in the talk by Keith Hare.



Attention

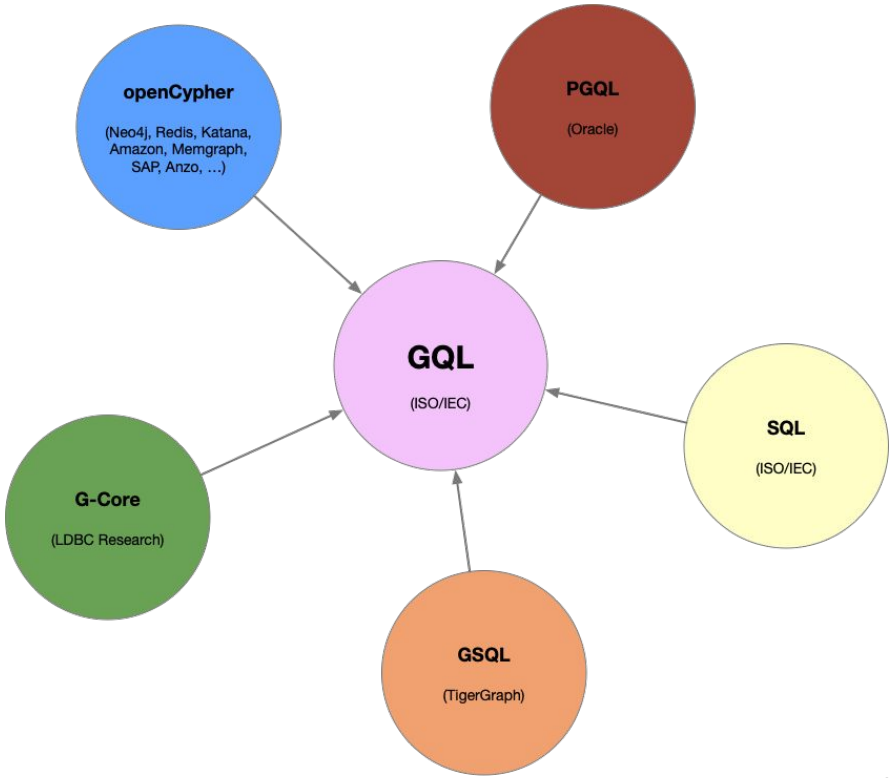
- **GQL is still under development and not final**
  Features may be *changed*, *dropped*, *or moved to a future version*.
- **ISO database standards are ”featurized”**
  Implementations are considered conforming as long as they don’t violate the standard but it’s up to them which optional features they choose to implement.
- **Safe harbour statement**
  Nothing in this talk, the slides, or the accompanying discussion represents a commitment by Neo4j (or any other vendor) to implement GQL or any of its features.

What is GQL?

Standardization effort by "The SQL Committee" for a new graph query language.

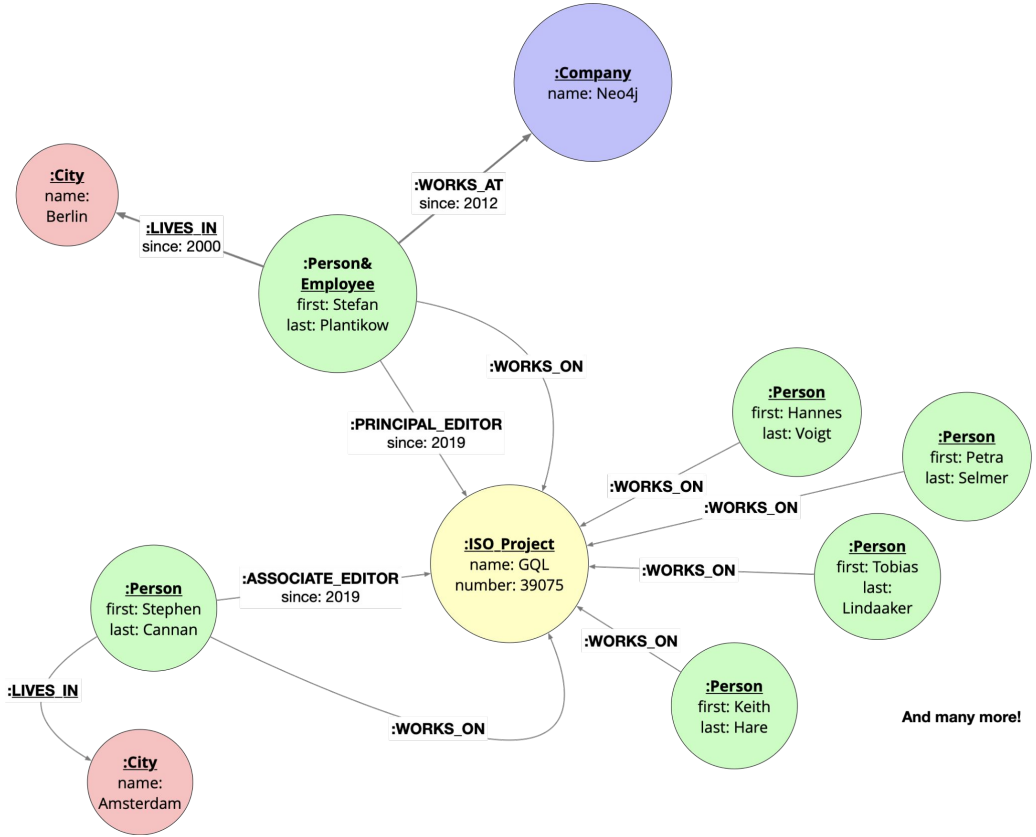
Motivated by growing adoption of property graphs (fastest growing database segment by far) and **commonalities across languages.**

Initiated by A. Green's "The GQL-manifesto": open letter to database industry: "*Let's build a next generation, declarative, composable, compatible, modern, intuitive International Standard for a Property Graph Database Language*" (Votes: 95 % of ca. 4000 votes: YES)



Property Graph Data Model

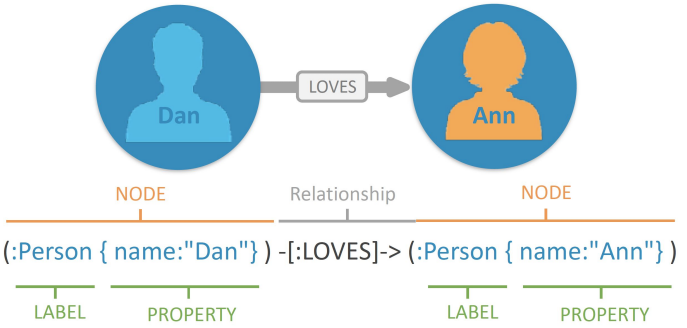
- Nodes (vertices) and relationships (edges) have
  - synthetic identity
  - 0..n labels
  - 0..n properties
- Edges are directed or undirected
- Graphs have 0..n graph properties



Visual Graph Pattern Syntax

```
MATCH (a:Person)-[:KNOWS*{1,2}]->(b:Person)
RETURN *
```

- Visual highly intuitive "Ascii-Art" syntax
- Use for property graph matching
  originally pioneered by Neo4j
- Idea adopted by
  openCypher, G-CORE, GSQL, PGQL
- "Best syntax for describing joins ever invented"
- Applicable in DQL, DML, DDL, Serialization



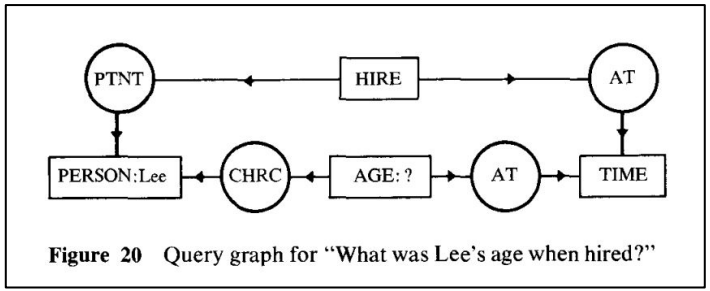


Figure 20 Query graph for "What was Lee's age when hired?"

Conceptual Graphs for a
Data Base Interface. J. F. Sowa. 1976.

Graph schema as a graph

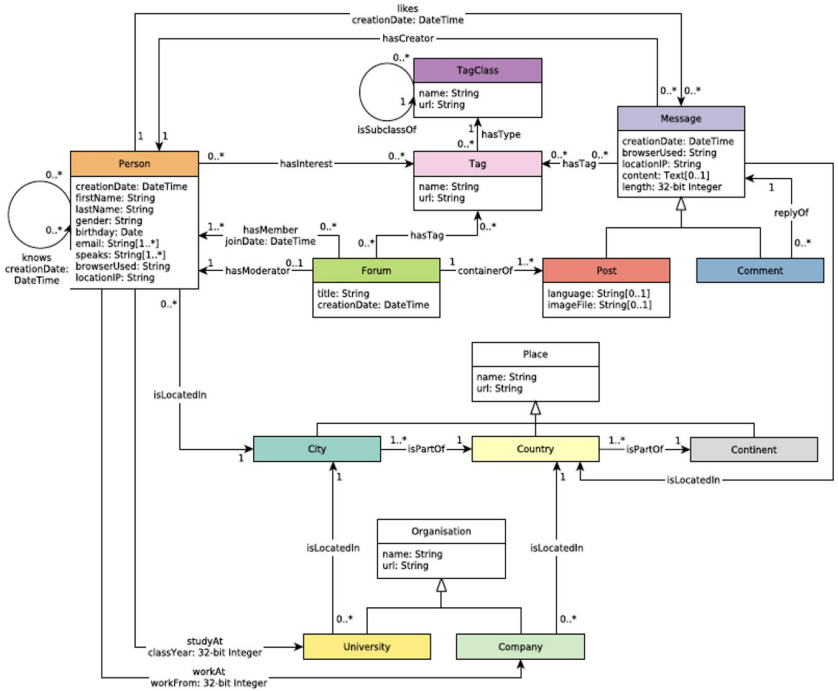


Figure 2.1: The LDBC SNB data schema

```
// Graph type describing graph schema
(:Person { gender STRING, birthday DATE } ),
(:Message { creationDate DATETIME, context TEXT }),
(:Tag { name STRING, url STRING }),
...


(:Person)-[:LIKES { creationDate DATETIME }]->(Message),
(:Message)-[:HAS_TAG]->(:Tag),
(:Person)-[:HAS_INTEREST]->(:Tag),
...

// Not yet defined
```

- Schema constraints
- Key constraints
- Cardinality constraints

```
...
```

GQL Goals

1. **Industry effort** informed by research and by community requirements.
2. **Universal property graph query language** that users can depend on to access graph databases, enabling skills reuse, vendor interoperability, and data longevity.
3. Establish **graphs as primary data model**, raising the level of abstraction and thereby enabling graph views and transformation.
4. **Backwards compatible** with existing languages, applications, and skills.
   No idle variation from proven syntax & semantics.
5. Query **language for all**: graph experts, SQL users, programmers, and data analysts.
6. **Grow the property graph space** to enable use of connected data by modern organizations.
7. **Integrate into modern technology stacks**: Unicode, IEEE Floats, ISO 8601 Temporal data, …
8. Standard that is **easy to learn, use, teach, implement, and evolve**.

GQL Features

1. Executed in simple request-response model in flat ACID transactions bound to a session.
2. Graph pattern matching supporting: joins, disjunction, nesting (with predicates), variable length patterns, shortest path patterns, path bindings, different matching modes, label expressions, ...
3. Data-querying (DQL) and data-modifying (DML) operations.
4. Hierarchical catalog for accessing multiple graphs in the same query.
5. Support for graphs with and without a graph type/schema.
6. Composeable language with support for user-defined procedures written in GQL.
7. SQL-compatible predefined atomic types, expressions, and standards mechanics, support for path, collections, and record types.
8. Scalar and tabular results.

Bonus: Graph composition, subgraph views, and returning graphs.

A taste of GQL (1): RETURN + DML

```
①  SESSION SET $myParam /* session parameters */
②  START TRANSACTION /* transaction demarcation */

③  GQL runtime=gpu /* optional implementation-specific preamble */
   FROM socialGraph
   MATCH (p:Person)-[:FRIEND]->()-[:FRIEND]->(f:friend)
     WHERE p.age < f.age AND f.country = $country     /* request parameter */
   INSERT (p)-[:FOAF]->(f)                          /* INSERT instead of CREATE */
   RETURN count(*) AS edges_added                   /* SELECT is supported, too */

④  COMMIT /* transaction demarcation */
⑤  END /* session demarcation */
```

A taste of GQL (2): SELECT

```
SELECT t.name AS team, avg(p.age) AS avgAge, count(p) AS numPlayers
FROM SportsGraph
MATCH (t:BasketballTeam)->(p:Player) WHERE t.level = 'pro'
GROUP BY t HAVING numPlayers > 5
ORDER BY avgAge DESC
LIMIT 5
```

A taste of GQL (3): Multigraph query

```
CALL {
  FROM /socNet/twitter
  MATCH (f:Follower)
  RETURN f, "twitter" AS kind
  UNION
  FROM /socNet/instagram
  MATCH (f:Follower)
  RETURN f, "insta" AS kind
}
MATCH (c:Customers) WHERE c.email = f.email
RETURN c.name AS name, kind
```

Pattern matching algebra in GQL

Pattern matching syntax and operator semantics shared by GQL and SQL/PGQ

- Selecting nodes and relationships with **complex label expression syntax** (conjunction, disjunction, nesting, negation, e.g `:Person&(Employee|Intern)`
- **Path pattern union** (a form of disjunction) e.g.
  `(a) (-[:KNOWS]- | -[:WROTE]->()<-[:WROTE]- | -[:WORKS_AT]->()<-[:WORKS_AT]-) (b)`
- Join, e.g. `(a)->(b), (a)->(x)`
- Transitive closure (Kleene star), with optional lower and upper bounds, e.g. `()-[*{1,2}]->()`
- **Nesting with optional predicates** and **sub-aggregation**
  `(a) ( ()-[:X]->(r)-[:Y]->() WHERE r.score > 0.5 )* (b)`
- Path binding, e.g. `p=()->()`
- Modifiers for controlling match semantics:
  Shortest path, **cheapest path**, different nodes/edges (aka node/edge-isomorphism)

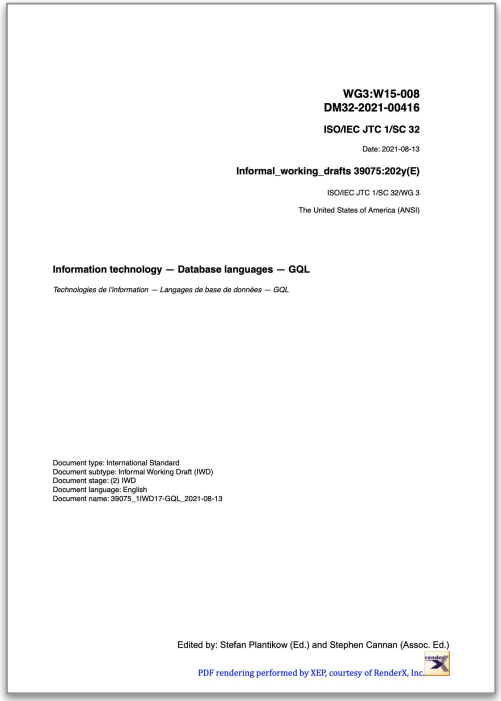
GQL Progress

- 525 pages with annexes, indexes, notes, released monthly
- Editorially drafted, currently reviewing/reworking features

- Pattern matching functionality
- Execution model of the standard
- GQL-Environment and GQL- Catalog, data model, and basic graph schema
- Predefined data types

- Ongoing: Query structure and DQL statements
- Ongoing: Type system

- Next: DML, DDL statements, resolve issues, review expressions, reduce size, ...



WG3:W15-008
DM32-2021-00416

ISO/IEC JTC 1/SC 32

Date: 2021-08-13

Informal_working_drafts 39075:202y(E)

ISO/IEC JTC 1/SC 32/WG 3

The United States of America (ANSI)

Information technology — Database languages — GQL

Technologies de l'information — Langages de base de données — GQL

Document type: International Standard
Document subtype: Informal Working Draft (IWD)
Document stage: (2) IWD
Document language: English
Document name: 39075_1IWD17-GQL_2021-08-13

Edited by: Stefan Plantikow (Ed.) and Stephen Cannan (Assoc. Ed.)

PDF rendering performed by XEP, courtesy of RenderX, Inc.

Future: Graph projection

```
CALL {
  FROM /socNet/twitter
  MATCH (f:Follower) RETURN f
  UNION
  FROM /socNet/instagram
  MATCH (f:Follower) RETURN f
}
MATCH (c:Customers)
  WHERE c.email = f.email
CONSTRUCT
MERGE (p:Person {email: c.email})
MERGE (c)-[:IS]->(p)<-[:IS_PERSON]-(f)
```