The GQL Standard

The upcoming ISO/IEC International Standard for a property graph query language

Stefan Plantikow, GQL Editor, Neo4j

The GQL Standard

Stefan Plantikow, Principal GQL Editor, Neo4j

The upcoming ISO/IEC International Standard for a property graph query language

KGC The Knowledge Graph Conference

The GQL Standard

1

GQL Story

From openCypher to ISO Project

2

Envisaging GQL

Features and syntax

3

GQL Process

Standardization and timeline

Stefan Plantikow, GQL Editor, Neo4j, May 2021

Stefan Plantikow, GQL Editor, Neo4j, May 2021

Caveats

- **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 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**

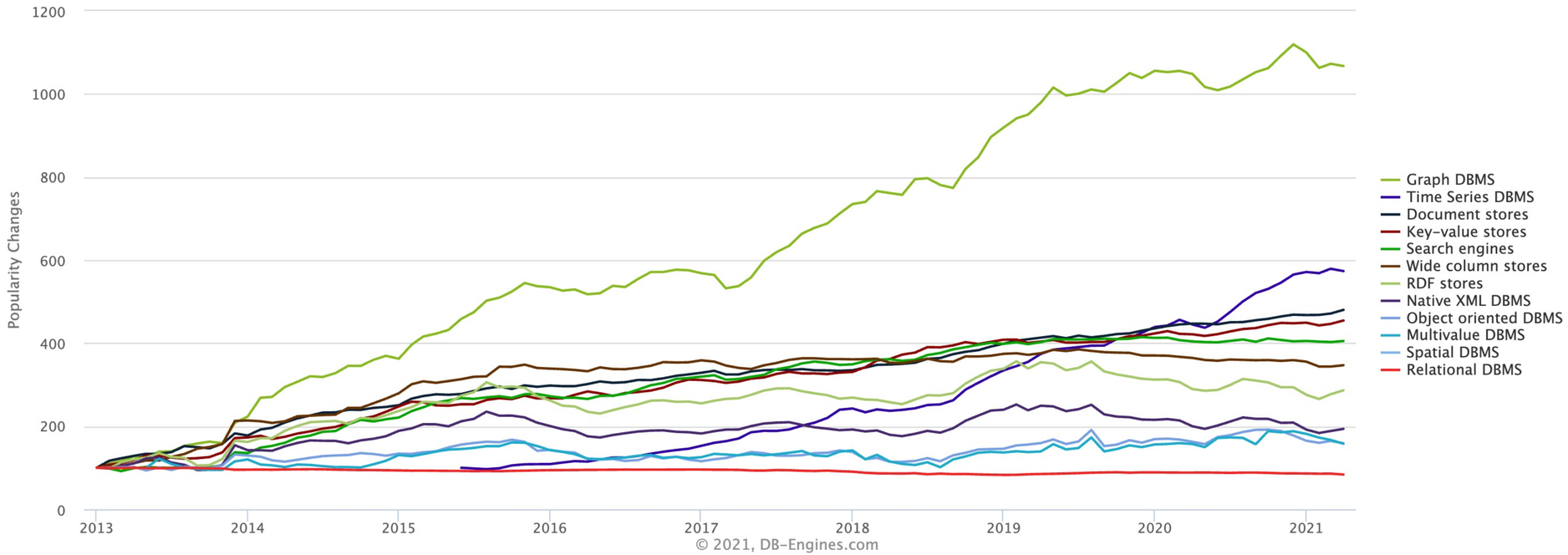
  Nothing in this talk, the slides, or the accompanying discussion represents a commitment by Neo4j
  (or any other vendor) to implement GQL or any of its features.

GQL Story

From openCypher to ISO project

(Property) graphs are everywhere

Complete trend, starting with January 2013



Neo4j share of Graph DBMS:

20 000 000+ Direct downloads,
100 000 000+ DockerHub downloads,
50.000+ Trained developers

3/top 5 airlines
25%+ Fortune 500,
7/top 10 retailers

20/top 25 financial services
3/top 5 hotels
3/top 5 airplane manufacturers
7/top 10 software companies

4/top 5 telecommunication firms

KGC The Knowledge Graph Conference

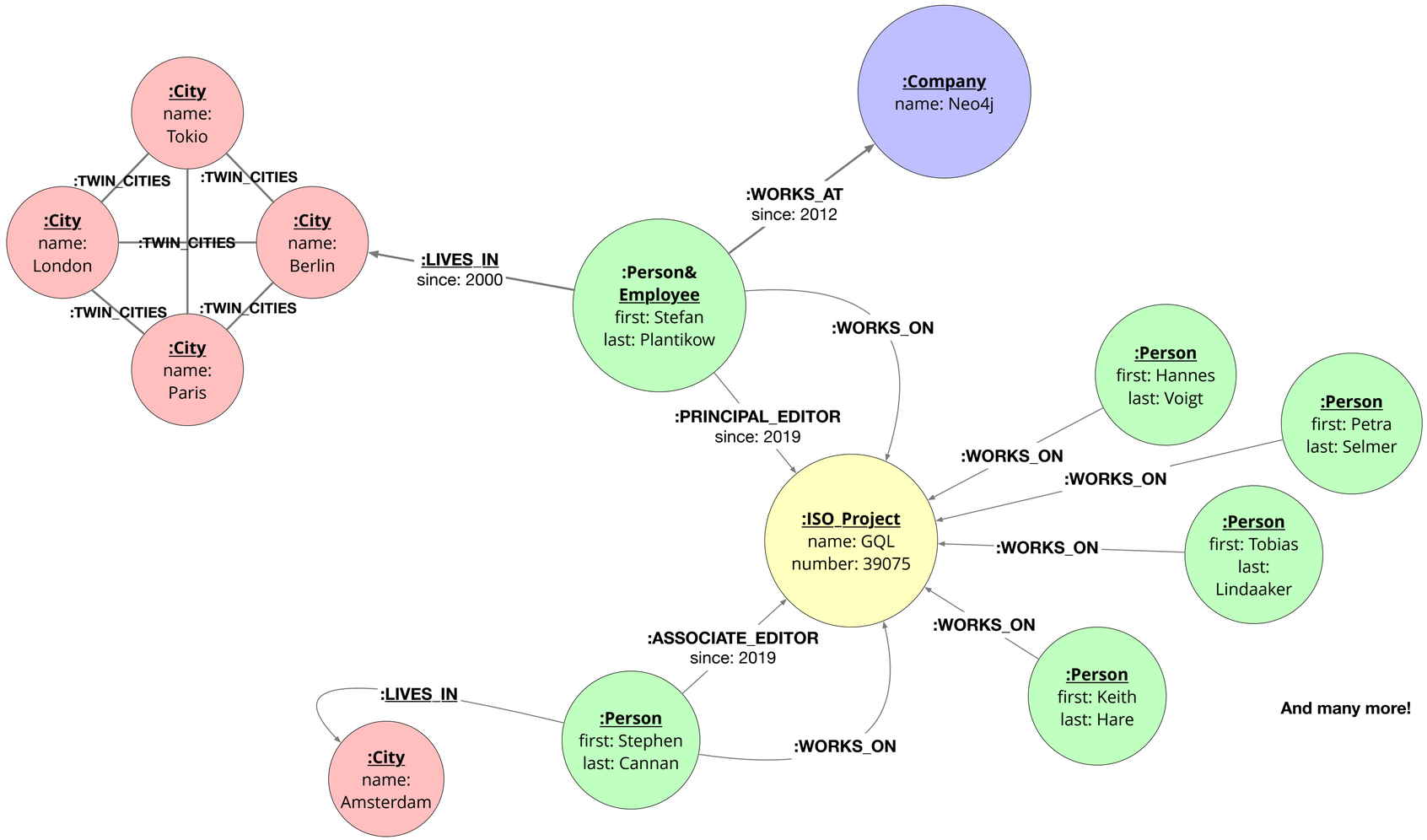
Data model

Property graphs

Nodes (vertices) and relationships (edges) with

- Synthetic (unique) identity
- 0..n Labels
- 0..n Properties
- Edges may be directed or undirected

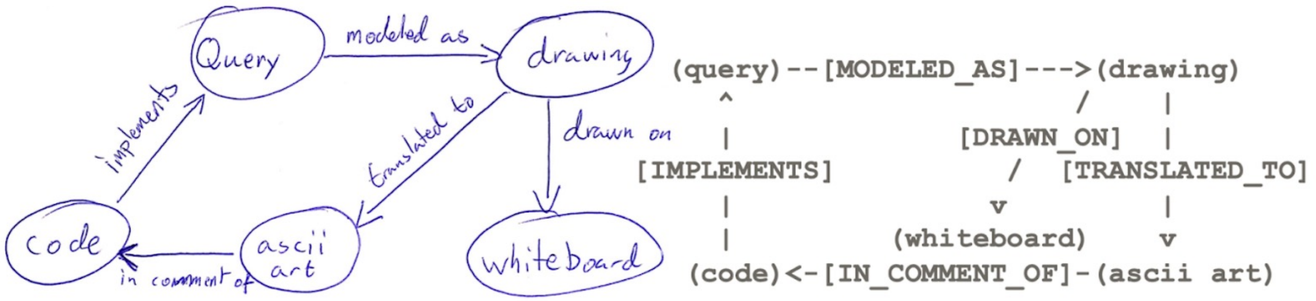
+ 0..n graph properties



Cypher Query Language

Declarative **graph pattern matching** language

- Intuitive:
  Graph patterns are very easily expressed
- Recursive (e.g. variable-length) path queries
- Querying and returning paths
- Modern data types and conventions
- Top-Down



SQL-like and pattern-based syntax and features

- **DQL** for reads (focus of this talk)
- **DML** for updates
- **DDL** for creating constraints and indexes

```
MATCH (query)-[:MODELED_AS]->(drawing),
      (code)-[:IMPLEMENTS]->(query),
      (drawing)-[:TRANSLATED_TO]->(ascii_art),
      (ascii_art)-[:IN_COMMENT_OF]->(code),
      (drawing)-[:DRAWN_ON]->(whiteboard)
WHERE query.id = $query_id
RETURN code.source
```

Cypher's linear vs SQL's nested composition

```
FROM myGraph
MATCH (a:B) WHERE a.foo%42 = 0
LET id=a.id, foo=a.foo, bar=a.bar
FILTER bar > 3
RETURN id, foo, count(*) as X
THEN
FILTER X > 100
RETURN foo+" "+id
```

Linear
Top-Down

```
SELECT foo+" "+id
FROM {
  SELECT id, foo, count(*) as X
  FROM {
    SELECT a.id AS id, a.foo AS foo, a.bar AS bar
    FROM myGraph
    MATCH (a:B) WHERE a.foo%42 = 0
  }
  WHERE bar > 3
  GROUP BY id, foo
}
WHERE X > 100
```

Bottom-up
Nested

- Form of nested procedure composition: Compose reading, aggregation, and updating steps in linear, top-down ("GQL-style") order
- (GQL: work with multiple graphs)

- Form of nested procedure composition: Compose reading and aggregation step in nested, bottom-up („SQL-style") order
- (GQL: work with multiple graphs)

KGC The Knowledge Graph Conference

openCypher

Multivendor effort to evolve Cypher

- Started 2015, informal
- Implementers, researchers, academics
- Markdown-based specification via addendums to Cypher 9 reference

Implementations

- 9 commercial, 4 research
- Cypher for Apache Spark
- Cypher for Gremlin

Artifacts and tooling

- Language Front-End (Parser, Syntax Analysis)
- ANTLR and EBNF Grammars
- Formal semantics
- Technology Compatibility Kit (TCK) - Cucumber test suite
- Style guide

Research

Updating Graph Databases with Cypher.
VLDB. A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Schuster, P. Selmer, H. Voigt. 2019

G-CORE: A Core for Future Graph Query Languages.
SIGMOD. R. Angles, M. Arenas, P. Barcelo, P. Boncz, G. Fletcher, C. Gutierrez, T. Lindaaker, M. Paradies, S. Plantikow, J. Sequeda,
O. van Rest, and H. Voigt. 2018.

Cypher: An Evolving Query Language for Property Graphs.
SIGMOD. N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. 2018.

Graph querying

Choices…

Declarative or imperative

Platform API, language, or protocol

…and commonalities

Move towards *declarative languages*

Pattern matching, pioneered by Neo4j Cypher

Common property graph model

Use-case centric

openCypher
(Neo4j, Redis, Katana Graph, Memgraph, Agens Graph, SAP/HANA, CAPS, …)

```
MATCH (n:Person)-[:KNOWS*..2]->(m:Person)
WHERE n <> m
RETURN n, count(DISTINCT m) AS num_foafs
```

PGQL *(Oracle PGX)*

```
SELECT n, count(DISTINCT m) AS num_foafs
FROM MATCH (n:Person)-[:KNOWS{,2}]->(m:Person)
WHERE n <> m
GROUP BY n
```

GSQL (*TigerGraph*)

```
SELECT n, count(DISTINCT m) AS num_foafs
FROM Person:n-(Knows>:e*..2)-Person:m
WHERE n <> m
GROUP BY n
```

Gremlin (Apache)

```
// alt: Cypher-for-gremlin
g.V().hasLabel('person').as('n')
     .outE('knows').optional(_.in().outE('knows'))
     .in().hasLabel('person).as('m')
     .select('n', 'm').group().by('n').dedup().count()
```

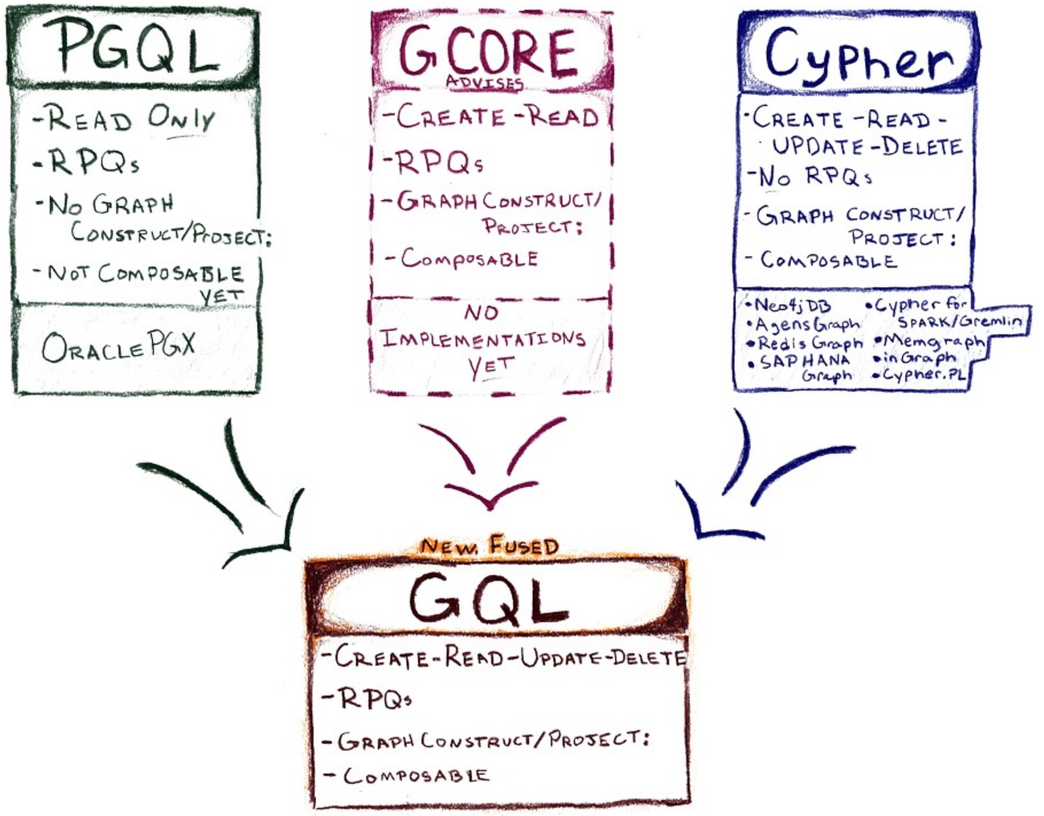
The GQL Manifesto

Open letter to the database industry

"Let's build a next generation, declarative, composable, compatible, modern, intuitive International Standard for a Property Graph Database Language"

(Alastair Green, @ Neo4j in 2019)

95 % of ca. 4000 votes: YES



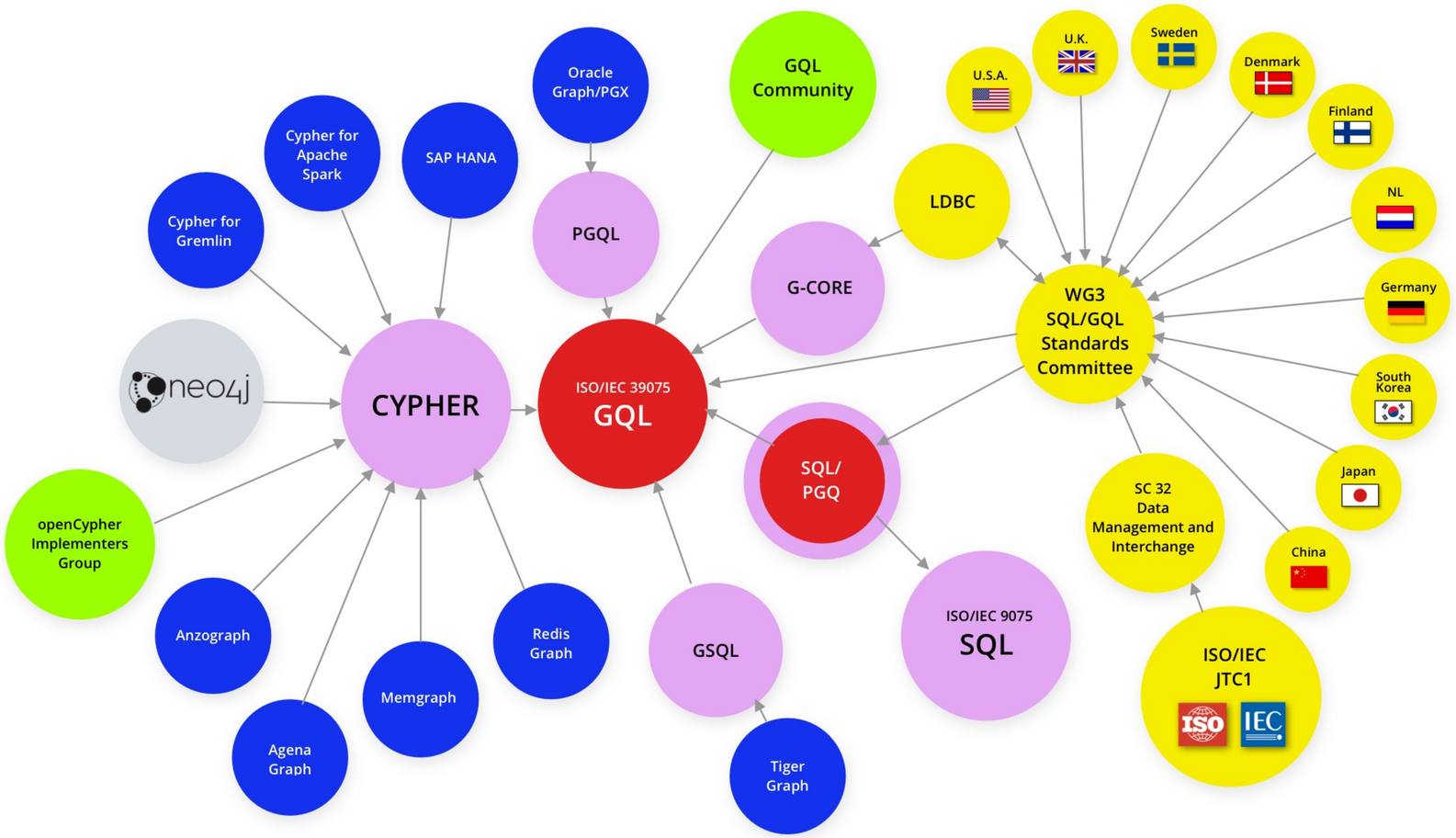
GQL Project

Property Graph World
Cypher, PGQL, GSQL

SQL
Data types, Expressions

Community
openCypher, LDBC

Academia
G-Core, Data graphs, ...



Goals

- Universal property graph query language that users can depend on to access graph databases, enabling skills reuse and vendor interoperability.
- Establish graphs as primary data model, raising the level of abstraction and enabling graph views and transformation.
- Compatibility with existing languages, applications, and skills. No idle variation from proven syntax & semantics.
- Query language for graph experts, SQL users, programmers, and data analysts.
- Grow the property graph space to enable use of connected data by modern organizations.
- Support modern technology stacks: Unicode, IEEE Floats, ISO 8601, …
- Standard that is easy to learn, use, teach, implement, and evolve.

Envisaging GQL

Features and syntax

A taste of what GQL might look like

```
①  SESSION SET VALUE $age = 20      /* session parameters */
   START TRANSACTION                /* transaction demarcation */


②  GQL runtime=gpu                                          /* optional preamble */
③  USE myGraph
④  MATCH (p:Person)-[:FRIEND]->()-[:FRIEND]->(friend)
   WHERE friend.age > $age AND p.country = $country         /* procedure parameter */
⑤  RETURN count(*) AS edges_added                           /* SELECT is supported, too */


⑥  COMMIT  /* transaction demarcation */
   END     /* session demarcation */
```

Language basics

Common ground between SQL and property graph world.

- Human-readable „Clause syntax“ following SQL, openCypher, PGQL, GSQL, ...
- Linear top-down (RETURN) and nested bottom-up (SELECT) composition
- Expressions from all input languages (openCypher, SQL, GSQL, ...)
- Hierarchical catalog: Vendor-adaptable directory structure with schema at leaves.

Modern data types

- **Standard scalar types:** Unicode strings, numbers incl. IEEE 754/ISO 60559 floats, ISO 8601 temporal types, booleans, ...
- **Collection data types:** Maps `{ name: "GQL", type: "language", age: 0 }`, arrays `[1, 2, 3]`, sets `SET { a, b }`, ...
- **Graph-related data types:** Node, edge, and graph references, as well as paths, ...
- **Other schema objects** (Next slide)

Hierarchical Catalog

Vendor-adaptable directory structure with schema at leaves. Schema contains primary objects such as

- Graphs
- Graph types
- Stored procedures
- Global constants
- Path pattern macros
- …

```
/sales                                  (root directory)
  /2019                                 (schema)
    • Total2019                         (graph)
    • Quarterly2019Q4                   (graph)
    • Quarterly2019Q3                   (graph)
  /2018                                 (schema)
    • Total2018                         (graph)
/HR                                     (directory)
  /open                                 (schema)
    /Staff                              (graph)
      • Persons                         (subgraph view)
      • Departments                     (subgraph view)
    /Sensitive                          (graph)
      • Persons                         (subgraph view)
      • Departments                     (subgraph view)
      • Salaries                        (graph)
```

(Example from H. Voigt)

Graph schema is a graph

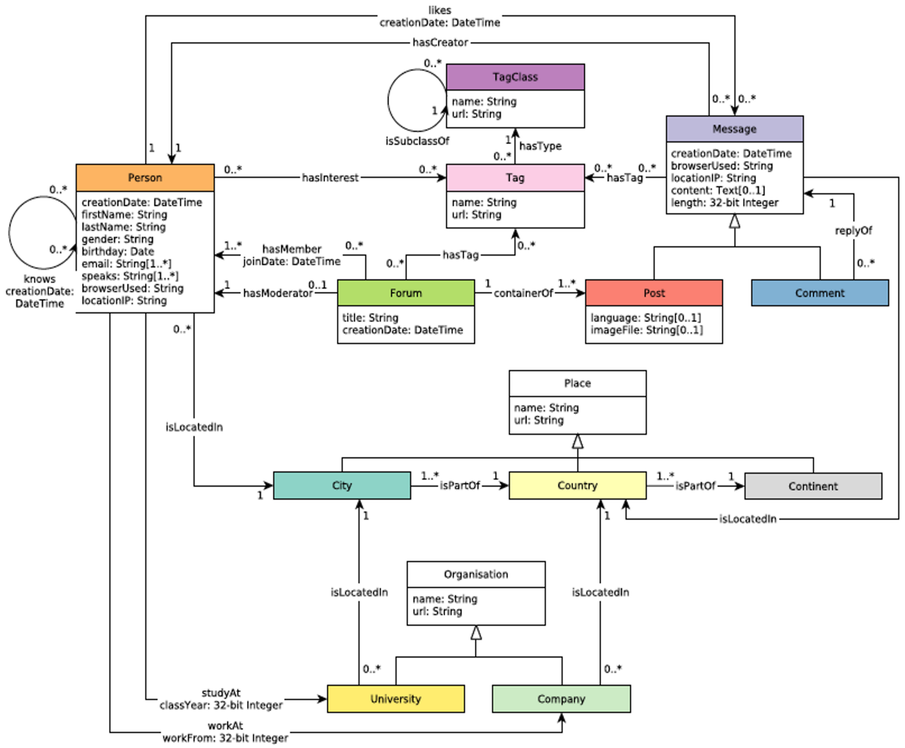


Figure 2.1: The LDBC SNB data schema

```
// 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),
...
```

+ Schema constraints

- Key constraints
- Cardinality constraints
- ...

New graph patterns

1. Selecting elements with **complex label expressions.**

```
MATCH (n:Person&(Employee|Intern))
```

2. Predicates on properties along a path:

```
MATCH (start) [ (p1:Person)-[r:KNOWS]-(p2:Person) WHERE r.since < date("2001-09-11"]* (end)
```

3. Bounded repetition.

```
MATCH (me) [ (:Person)-[:KNOWS]->(:Person) ]{2,5} (you)
```

4. **Path pattern union** (a form of disjunction).

```
MATCH (a) (-[:KNOWS]- | -[:WROTE]->()<-[:WROTE]- | -[:WORKS_AT]->()<-[:WORKS_AT]-) (b)
```

5. Configurable pattern-matching semantics: Node isomorphism, edge isomorphism, homomorphism.

6. Path pattern output modifiers.

```
MATCH ALL | ANY SHORTEST | ALL SHORTEST ...
```

(Examples from P. Selmer)

Updating graphs

The power of visual pattern syntax re-used:

```
USE myGraph
INSERT (:Language {name: "GQL"})-[:PRESENTED_AT]->(:Place {name: "Austin"})

// or
INSERT (place:Place {name: "Austin"})
INSERT (lang:Language {name: "GQL"})
INSERT (lang)-[:PRESENTED_AT]->(place)
```

More modifying statements:

```
DELETE n                        // delete graph elements
DETACH DELETE n                 // delete nodes and their relationships
SET/REMOVE n:Label              // set or remove labels
SET/REMOVE n.prop               // set or remove properties
MERGE (:Book { isbn: ... })     // upsert (under discussion)
```

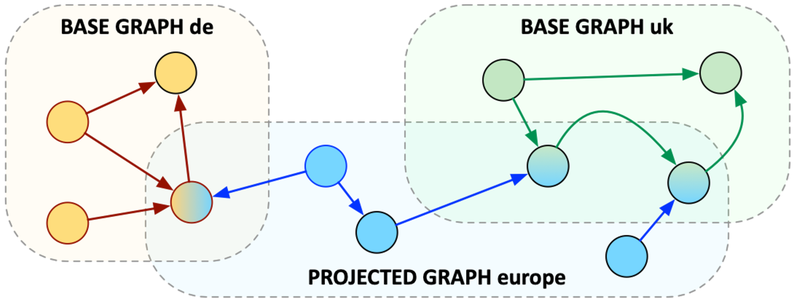
KGC The Knowledge Graph Conference

Multigraph and graph projection

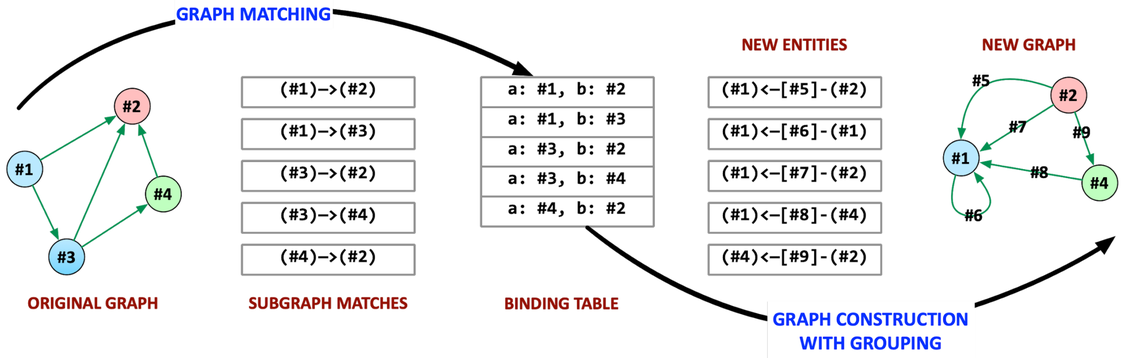
Queries may use multiple graphs.
Graphs are either persistent or constructed by

- Sharing existing elements
- Deriving new elements

(Shared edges always point to the same (shared) endpoints in all graphs)



Graph projection is
the inverse of pattern matching.



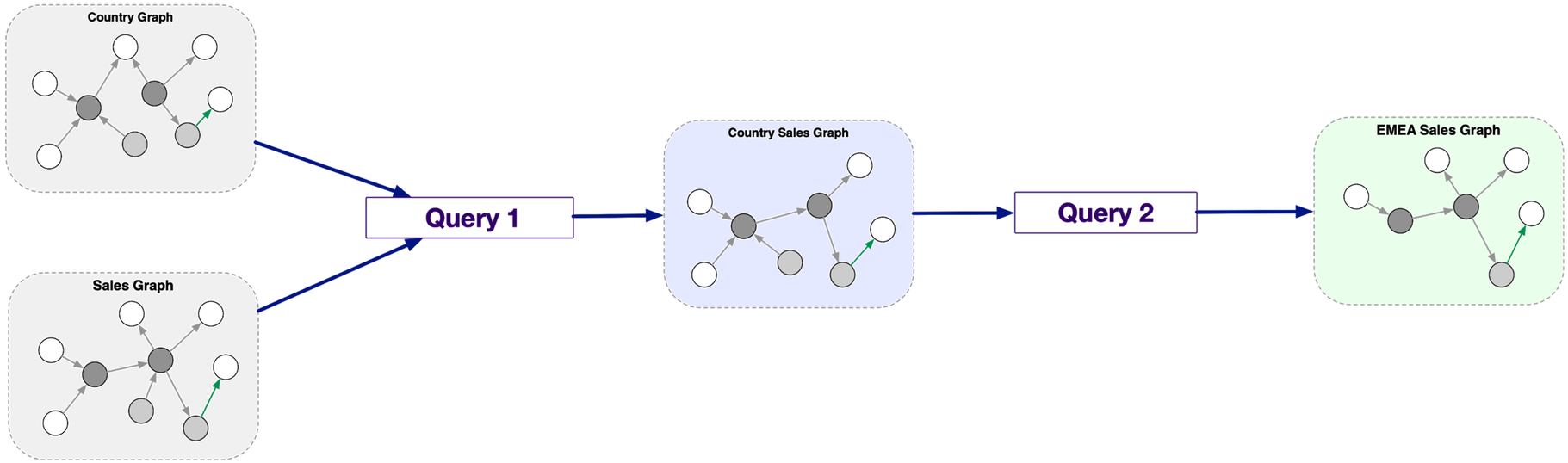
Procedure composition

Returning

- Graphs
- Binding tables
- Regular values,
  including nodes, relationships, paths etc.

Enabling

- Abstraction and re-use
- Complex processing chains
- Parameterized views (e.g. for inference)



GQL Process

Standardization and timeline

Standardizing GQL

Goal Let's build a next generation, declarative, composable, compatible, modern, intuitive International Standard for a Property Graph Database Language!

Where? ISO/IEC **JTC 1/SC 32/WG 3** aka "The SQL Standards Committee" (Convenor: Keith Hare)

How?

1. **National standards bodies** representing countries (e.g. ANSI) or **liasion organizations** join SC 32 as P-members or establish a liasion relationship (e.g. LDBC)

2. Each such eligible organization **delegates experts** (mostly from major vendors) to WG3 to submit **discussion papers** and **change proposals** and generally attend the WG3 meetings discussion those papers.

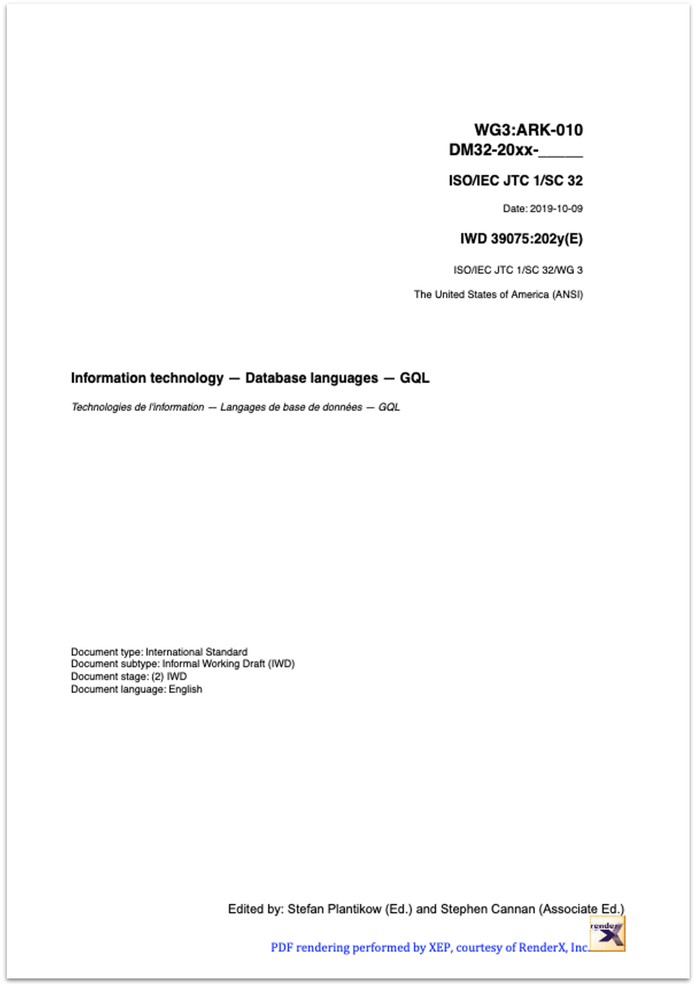
3. Editors modify the **GQL draft** to reflect consensus on submissions by WG3 and conform to ISO requirements.

4. The GQL draft is gradually refined in stages to become an **International Standard.**
   Formal progress requires voting by SC32 P-Members.

More information: Please contact Keith Hare, Stefan Plantikow, or Stephen Cannan.

KGC The Knowledge Graph Conference

GQL Draft



WG3:ARK-010
DM32-20xx-____

ISO/IEC JTC 1/SC 32

Date: 2019-10-09

IWD 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

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

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

- Initially drafted by editors (S. Plantikow, S. Cannan)

  Re-using existing text from SQL where appropriate and in line with ISO requirements, using proven tooling of the SQL Standard

- Featurized to help adoption (“Small GQL core")

- Currently reviewed and reworked by standardization process involving experts from the national standards bodies of the US, Germany, Netherlands, UK, Japan, South Korea, Sweden, Finland, Canada, China and liaison organizations such as the LDBC

- Developed in tandem with SQL/PGQ sibling standard

- 498 pages

Timeline and summary

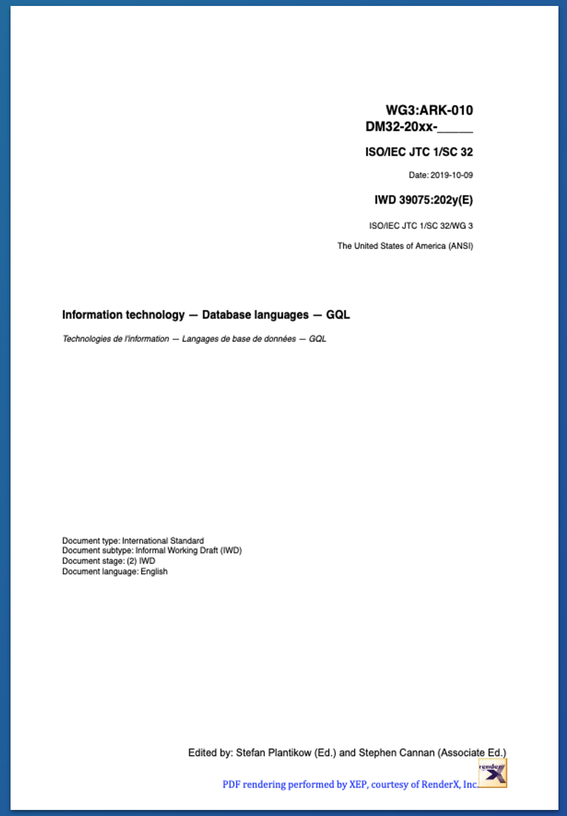
Working Draft	Since early 2021
Committee Draft	Towards end of 2021
Draft Intl. Standard	Polishing
Final Draft Intl. Standard	More polishing
International Standard	Publication by ISO, ~ 2022

- **GQL** is being standardized based on the success of property graph query languages.
- This is just the first version. More to come.
- Established languages, like **openCypher**, and their implementations (e.g. Neo4j) today provide the gateway to **GQL** tomorrow.

gqlstandards.org

#KGC2021

Join the Conversation



WG3:ARK-010
DM32-20xx-_____

ISO/IEC JTC 1/SC 32

Date: 2019-10-09

IWD 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

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

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

@KGConference @boggle

linkedin.com/company/the-knowldge-graph-conference/

youtube.com/playlist?list=PLAiy7NYe9U2Gjg-600CTV1HGypiF95d_D

KGC The Knowledge Graph Conference