Towards GQL v1

A Property Graph Query Language Standard

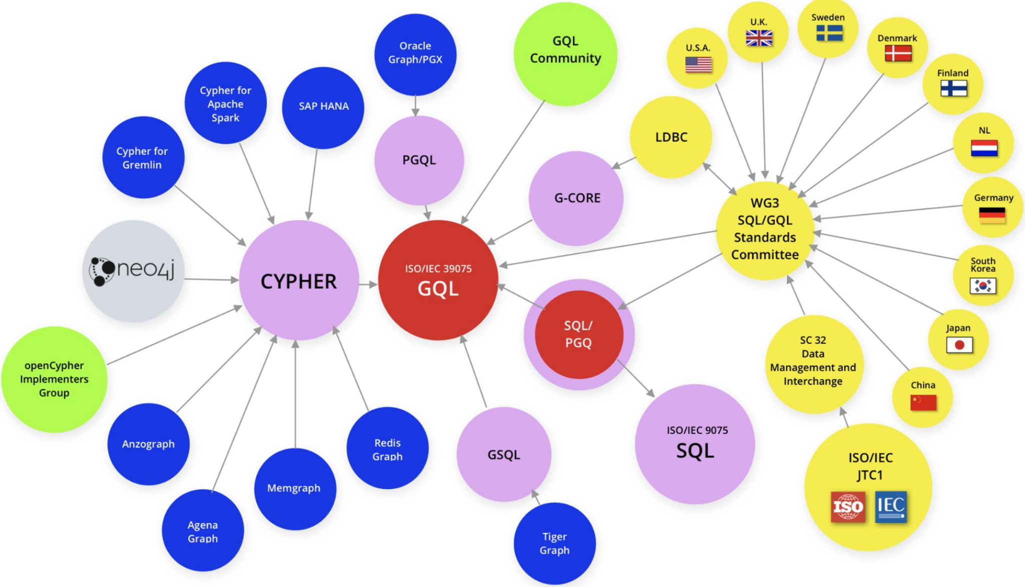
15th LDBC Technical User Committee (TUC) Meeting
18 June, 2022

Petra Selmer, Query Language Standards & Research, Neo4j

petra.selmer@neo4j.com

Topics

- What is GQL?
- How is GQL produced?
- What does GQL look like?



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.

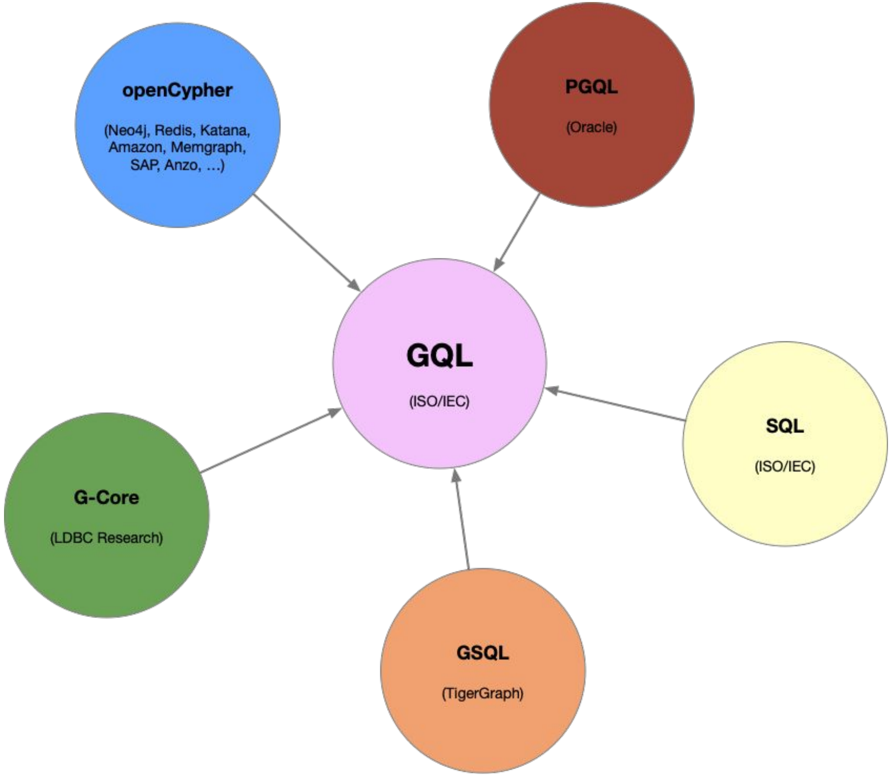
What is GQL?

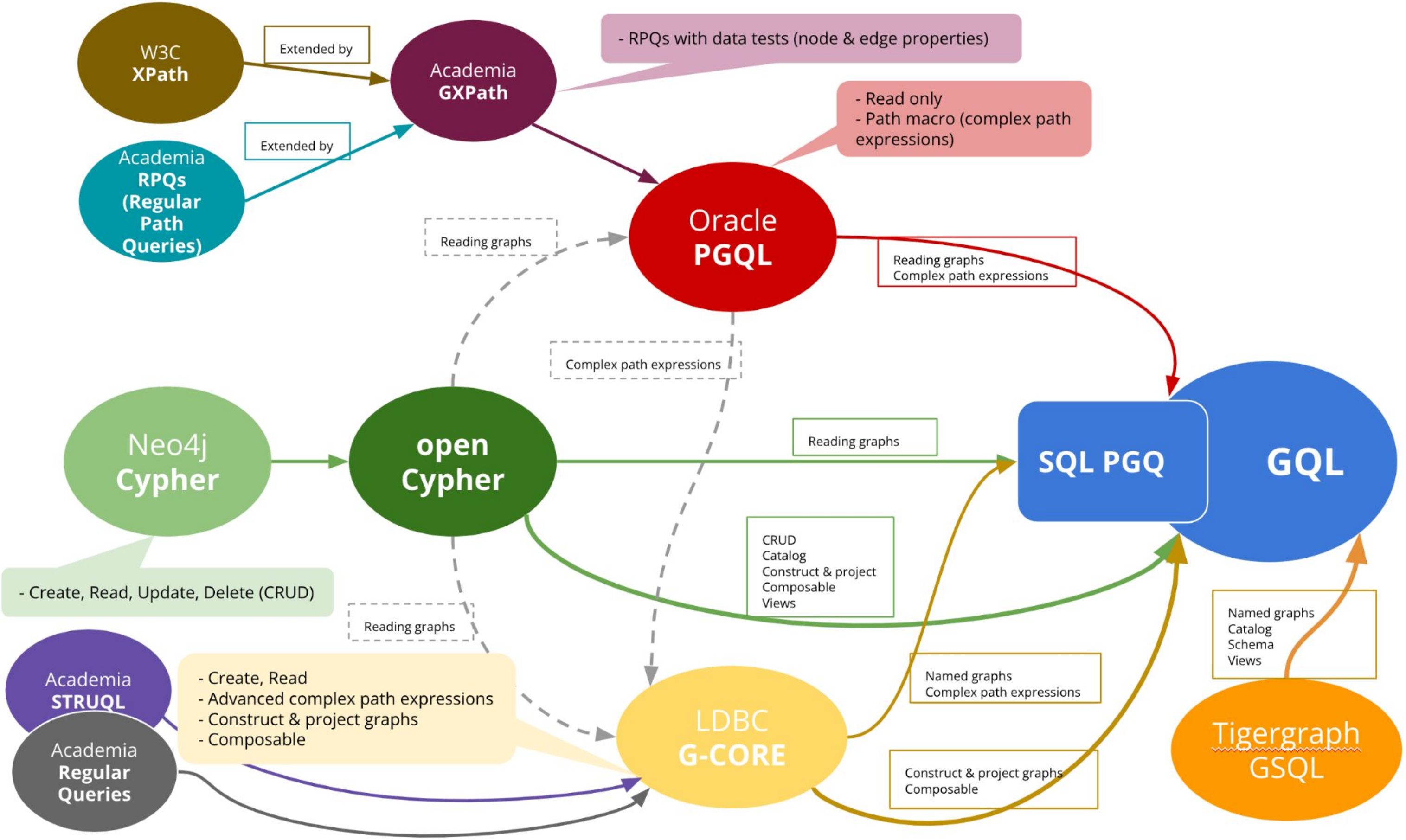
What is GQL?

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)





W3C **XPath**

Extended by

Academia **GXPath**

- RPQs with data tests (node & edge properties)

Academia **RPQs (Regular Path Queries)**

Extended by

- Read only
- Path macro (complex path expressions)

Oracle **PGQL**

Reading graphs

Reading graphs
Complex path expressions

Complex path expressions

Neo4j **Cypher**

open Cypher

Reading graphs

SQL PGQ

GQL

- Create, Read, Update, Delete (CRUD)

CRUD
Catalog
Construct & project
Composable
Views

Reading graphs

Named graphs
Catalog
Schema
Views

Academia **STRUQL**

Academia **Regular Queries**

- Create, Read
- Advanced complex path expressions
- Construct & project graphs
- Composable

LDBC **G-CORE**

Named graphs
Complex path expressions

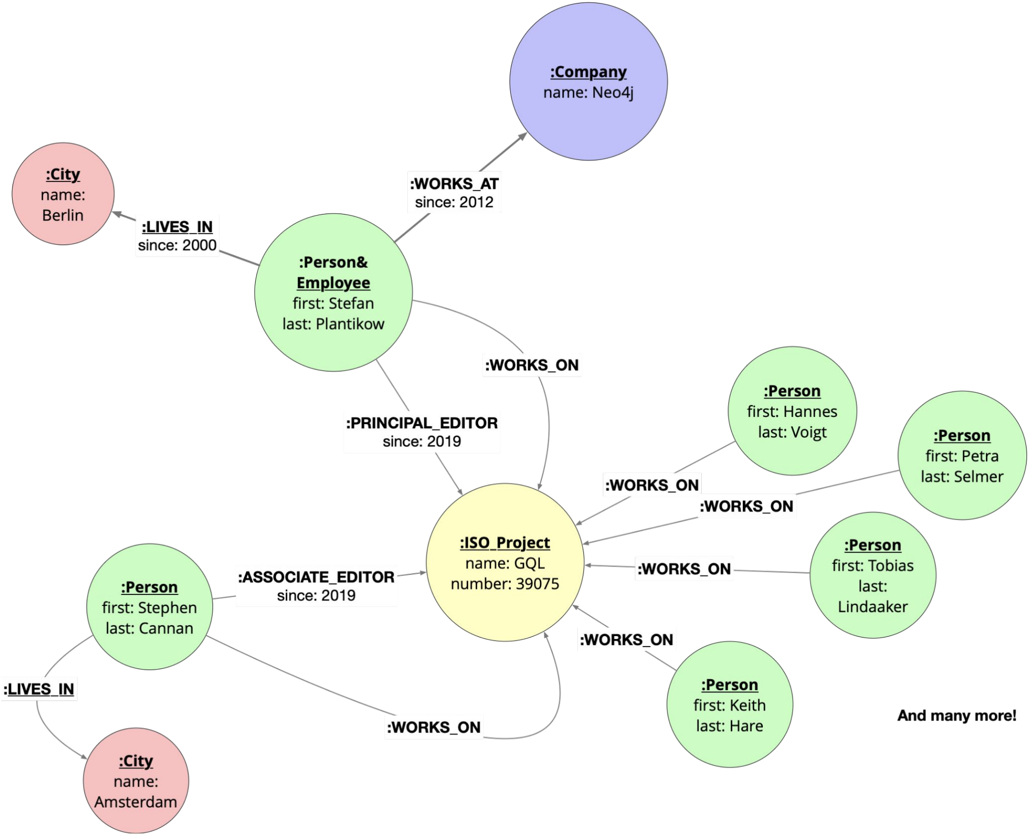
Construct & project graphs
Composable

Tigergraph GSQL

GQL Lineage

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 labels
  - 0..n properties

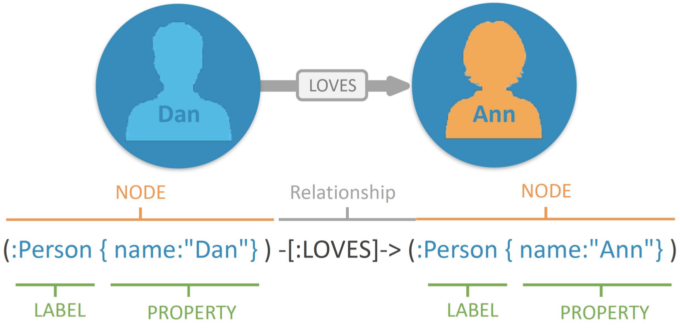


And many more!

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



```
(:Person { name:"Dan"} ) -[:LOVES]-> (:Person { name:"Ann"} )
```

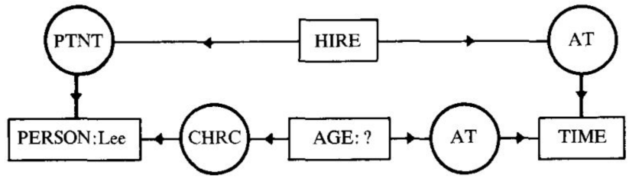


Figure 20 Query graph for "What was Lee's age when hired?"

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

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**.

Property Graph Standard GQL

GQL

- Full DB language
  - DQL - **Graph pattern matching** queries
  - DML – **CRUD** (Create, read, update, and delete) on graph elements and their labels and properties
  - DDL – Create graphs, **graph types**, etc. in a **global hierarchical database catalog**
- Optional extensions to the property graph model: multiple edge types, undirected edges
- Leverages common foundation from SQL and property graph languages, incl. **session and transaction control**
- Supports **schema-fixed** and **schema-optional** variants

How is GQL produced?

Property Graph Standards – SQL/PGQ and GQL

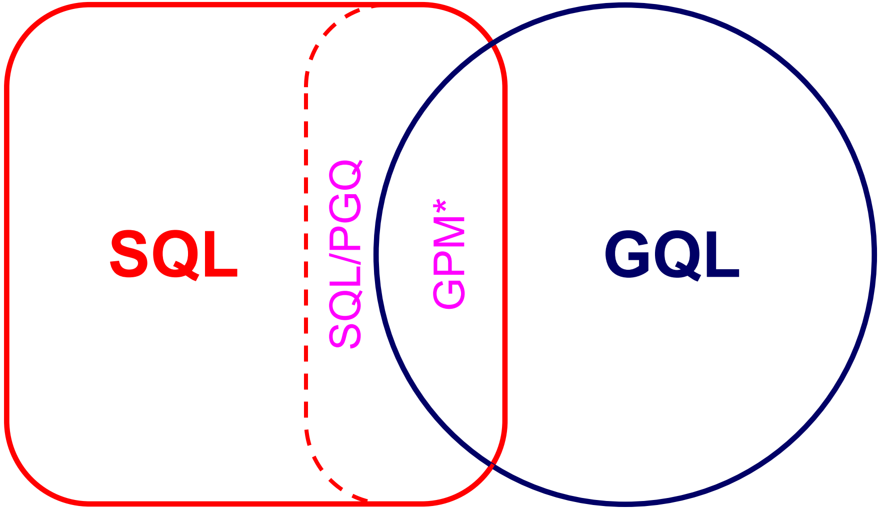
SQL/PGQ

- Property Graph views of SQL tables
- Graph Pattern Matching queries
  - GRAPH_TABLE() in SQL FROM
  - Supports Reads
- Common foundation with SQL and graph query languages
- Does not support schema-optional graphs

GQL

- Full DB language
  - DML – Create, Read, Update, Delete
  - DDL – Create Type, Create Graph
- Graph Pattern Matching queries
- Leverages common foundation from SQL and property graph languages
- Supports schema-fixed and schema-optional variants

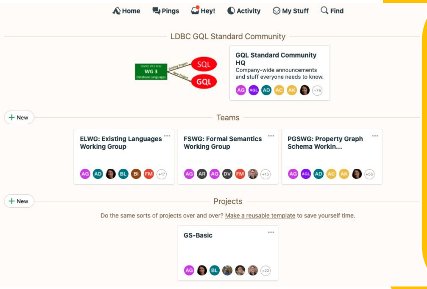
Property Graph Standards – SQL/PGQ and GQL



* Graph Pattern Matching

GQL community work

gqlstandards.org



LDBC GQL Standard Community

GQL Standard Community HQ
Company-wide announcements and stuff everyone needs to know.

Teams

ELWG: Existing Languages Working Group

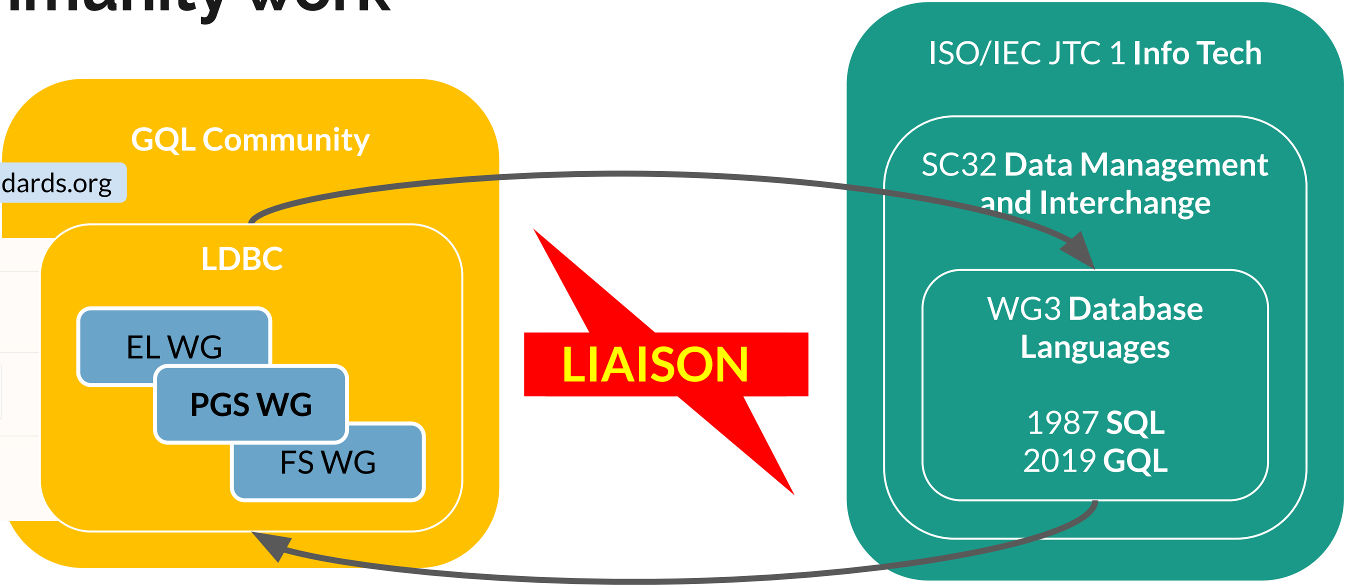
FSWG: Formal Semantics Working Group

PGSWG: Property Graph Schema Workin...

Projects

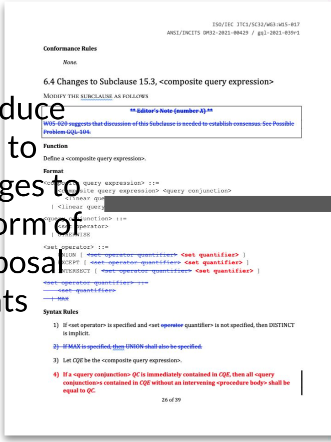
Do the same sorts of projects over and over? Make a reusable template to save yourself time.

GS-Basic



Work on the GQL draft between ballots

GQL Expert Group*



```
ISO/IEC JTC1/SC32/WG3:W15-017
ANSI/INCITS DM32-2021-00429 / gql-2021-039r1
```

Conformance Rules

None.

6.4 Changes to Subclause 15.3, <composite query expression>

MODIFY THE SUBCLAUSE AS FOLLOWS

~~**** Editor's Note (number *X*) ****~~

~~W05-020 suggests that discussion of this Subclause is needed to establish consensus. See Possible Problem GQL-104.~~

Function

Define a <composite query expression>.

Format

```
<composite query expression> ::=
    <composite query expression> <query conjunction>
      <linear que
  | <linear query
<query conjunction> ::=
    <set operator>
  | OTHERWISE
<set operator> ::=
    UNION [ ~~<set operator quantifier>~~ <set quantifier> ]
  | EXCEPT [ ~~<set operator quantifier>~~ <set quantifier> ]
  | INTERSECT [ ~~<set operator quantifier>~~ <set quantifier> ]
~~<set operator quantifier> ::=~~
~~    <set quantifier>~~
~~  | MAX~~
```

Syntax Rules

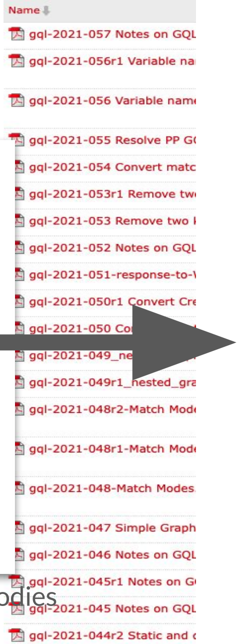
1) If <set operator> is specified and <set ~~operator~~ quantifier> is not specified, then DISTINCT is implicit.

~~2) If MAX is specified, then UNION shall also be specified.~~

3) Let *CQE* be the <composite query expression>.

4) **If a <query conjunction> *QC* is immediately contained in *CQE*, then all <query conjunction>s contained in *CQE* without an intervening <procedure body> shall be equal to *QC*.**

26 of 39



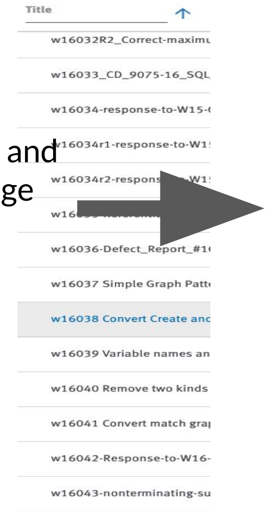
Name

- gql-2021-057 Notes on GQL
- gql-2021-056r1 Variable na
- gql-2021-056 Variable name
- gql-2021-055 Resolve PP G
- gql-2021-054 Convert matc
- gql-2021-053r1 Remove tw
- gql-2021-053 Remove two
- gql-2021-052 Notes on GQL
- gql-2021-051-response-to-
- gql-2021-050r1 Convert Cre
- gql-2021-050 Co
- gql-2021-049_ne
- gql-2021-049r1_nested_gra
- gql-2021-048r2-Match Mod
- gql-2021-048r1-Match Mod
- gql-2021-048-Match Modes
- gql-2021-047 Simple Graph
- gql-2021-046 Notes on GQL
- gql-2021-045r1 Notes on G
- gql-2021-045 Notes on GQL
- gql-2021-044r2 Static and

Experts produce and agree to specific changes to the draft in form of change proposal documents

WG 3

Database Languages

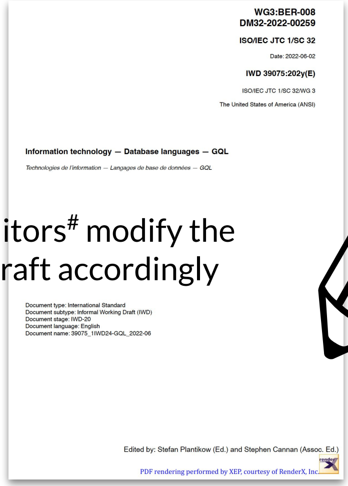


Title

- w16032R2_Correct-maximu
- w16033_CD_9075-16_SQL
- w16034-response-to-W15-
- w16034r1-response-to-W1
- w16034r2-respons -W1
- w16
- w16036-Defect_Report_#1
- w16037 Simple Graph Patt
- w16038 Convert Create and
- w16039 Variable names an
- w16040 Remove two kinds
- w16041 Convert match gra
- w16042-Response-to-W16-
- w16043-nonterminating-su

Experts debate and agree to change proposal documents

Editors# modify the draft accordingly



WG3:BER-008
DM32-2022-00259

ISO/IEC JTC 1/SC 32

Date: 2022-06-02

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: IWD-20
Document language: English
Document name: 39075_1IWD24-GQL_2022-06

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

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

* Or experts in other national bodies or a liaison

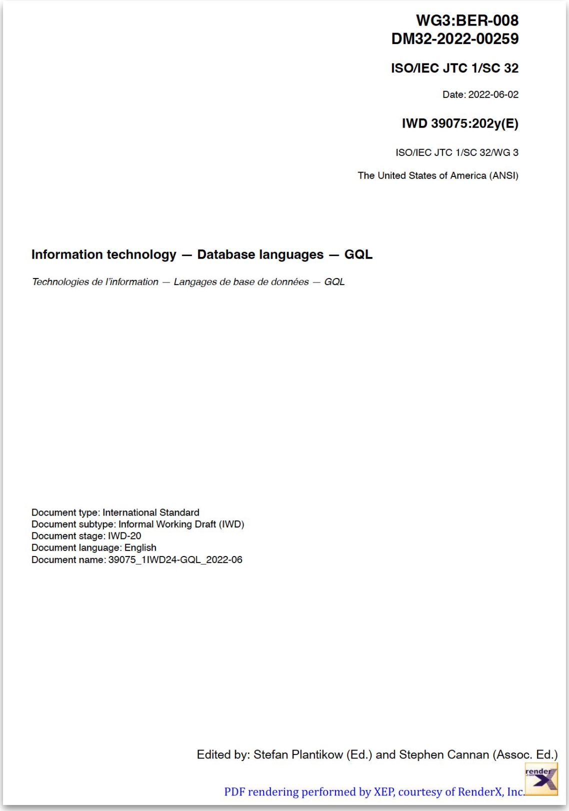
# Stefan Plantikow and Stephen Cannan

GQL Progress

- 505 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, DML and DQL statements
- Ongoing: Type system
- Ongoing: Resolving issues and comments
- Ongoing: Reducing size and scope



WG3:BER-008
DM32-2022-00259

ISO/IEC JTC 1/SC 32

Date: 2022-06-02

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: IWD-20
Document language: English
Document name: 39075_1IWD24-GQL_2022-06

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

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

GQL v1

- Go-to language for all new (and existing) property graph vendors
- We want to ensure adoption is as widespread as possible
- v1: focus on the core minimum
- Reduce feature set size => punt these to v2 and beyond

Start small => get big over multiple versions

What does GQL look like?

A taste of GQL: 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
```

(2) Pattern matching syntax extensions

- Selecting nodes and relationships with **label expressions** (and, or , not, etc.), e.g `:Person&(Employee|Intern)`
- **Path pattern union** `MATCH ( (a)-[:KNOWS]->(b)  |  (a)<-[:LOVES]-(b) )`

  Multiset alternation `MATCH ( (a)-[:KNOWS]->(b) |+| (a)<-[:LOVES]-(b) )`
- **Quantified path patterns**

  Simple `MATCH (a:Boss)-->(b:Sales))+`

  Filtering `MATCH ((a:Boss)-[r]->(b:Sales) WHERE r.age>5)+`

  Union `MATCH ((a)-[:KNOWS]->(b)  |  (a)<-[:LOVES]-(b)){2, 6}`

  Alternation `MATCH ((a)-[:KNOWS]->(b) |+| (a)<-[:LOVES]-(b)){2, 6}`

  Aggregation
  ```
  MATCH (a) ((:A)-[r:L]->(:C|D)){1,5} (b)
  WHERE a.height < AVG(r.weight) AND AVG(r.weight) < b.height
  ```

  + any combination
- Matching **walks**, **trails**, **simple paths**, **top k shortest/paths/path groups**, ...

Graph schema as a graph

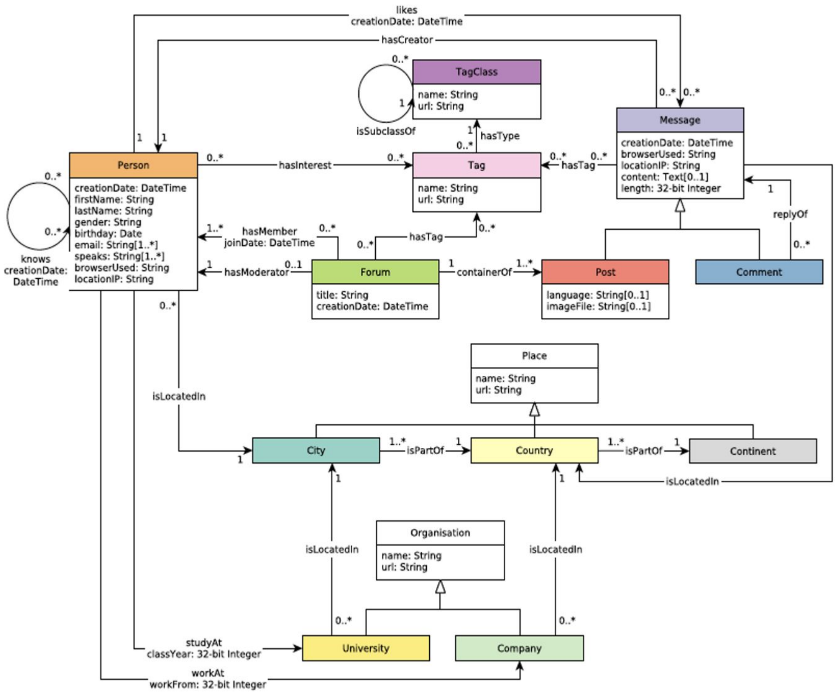


Figure 2.1: The LDBC SNB data schema

```
// 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
...
```

Labels

Schema

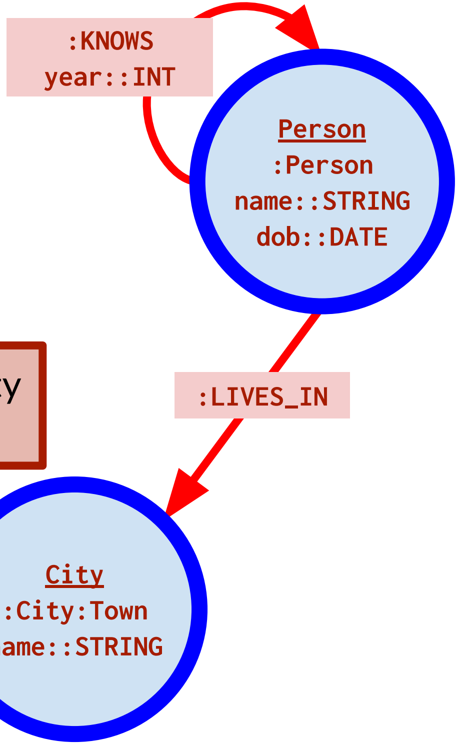
Node types

```
CREATE GRAPH TYPE SocialNetwork {
    (Person :Person {name STRING, dob DATE}),
    (City :City:Town {name STRING}),
    (Person)-[:LIVES_IN]->(City),
    (Person)-[:KNOWS {year INT}]->(Person)
}
```

Property types

Type names: not necessary, but helpful for more complex schemas

Relationship types



A taste of GQL: DML

```
INSERT ()-[r:S]->()
SET r = { a: 20, b: "West", c: 0.937 }
RETURN r.a, r.b, r.c // 20, "West", 0.937
```

```
MATCH ()-[r { a: 20 }]->()
SET r.b = "West"
RETURN r.a, r.b // 20, "West"
```

A taste of GQL: 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
```

"Getting Ready for GQL", Stefan Plantikow, Neo4j, Graph Day Austin, 2020

GQL v1

Property Graph Query Language Standardization

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

-
  ```
  FROM isoIecJtc1graph
  MATCH (:ISO_WG {num: 3})-[:WORKS_ON]->(gql:Standard {num: 39075})
  RETURN gql
  ```
- **Standards are hard and take a while**