

# GQL Scope and Features

(incl. GQL Process Update)

*Stefan Plantikow for the Neo4j Query Languages Team, LDBC TUC Meeting, Amsterdam, Netherlands, 2019*

# Information technology – Database languages – GQL

- Next Generation Query Language for Property Graphs by ISO/IEC JTC 1/SC 32/WG 3 (Convenor: Keith Hare)
- Goal: ISO Standard
- Status: Ballot Proposal for New Work Item Proposal (NWIP) started (until: Sep 2019)

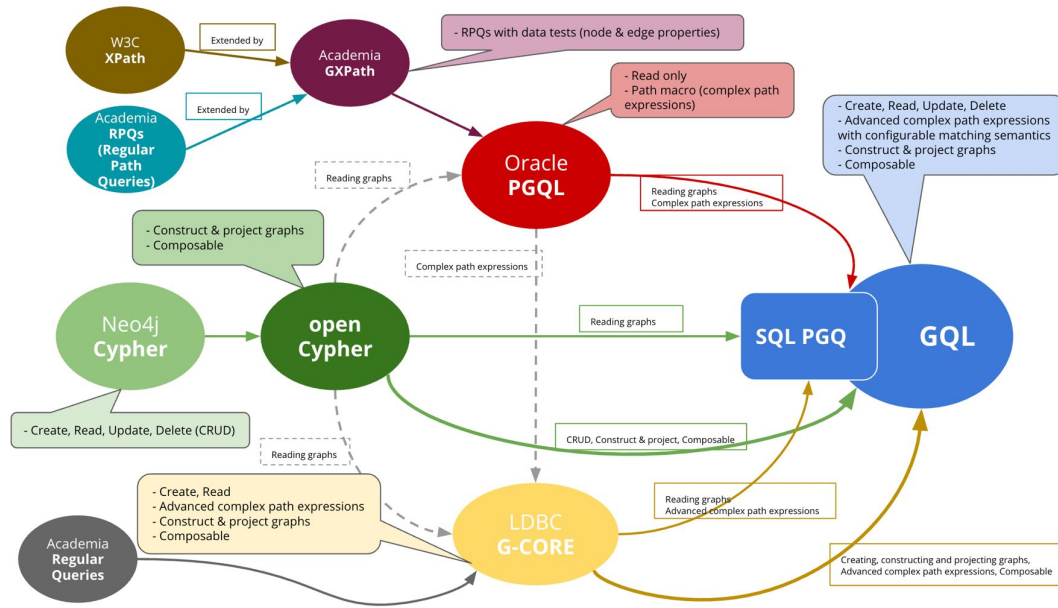
## **RESOLUTION 03-003 – New Work Item Ballot Circulation**

SC 32 requests its Committee Manager to circulate the following proposal for a **12-week** letter ballot and to JTC 1 for concurrent review.

<b>Designation</b>	<b>Title</b>	<b>Project Editor</b>
SC 32 N3002 (WG3:JCJ-011, WG3:JCJ-012r2)	Information Technology – Database Language GQL	Stefan Plantikow Stephen Cannan

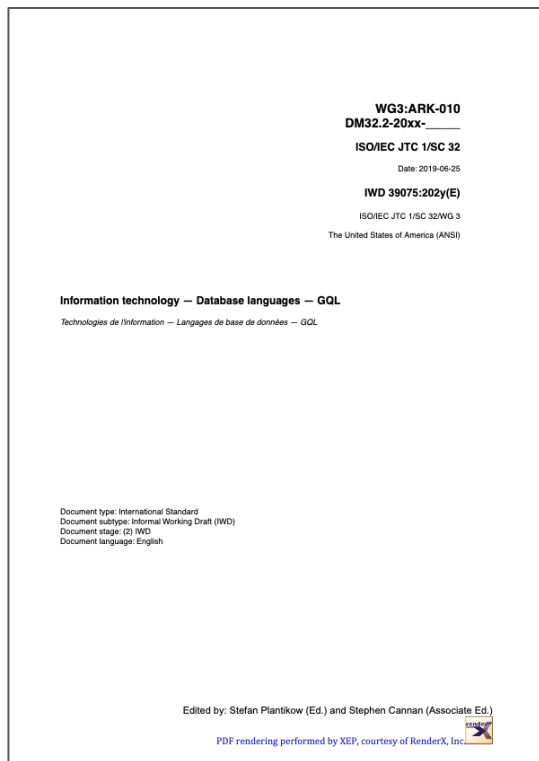
Note: ISO/IEC 39075 has been reserved for this project.

# Motivation



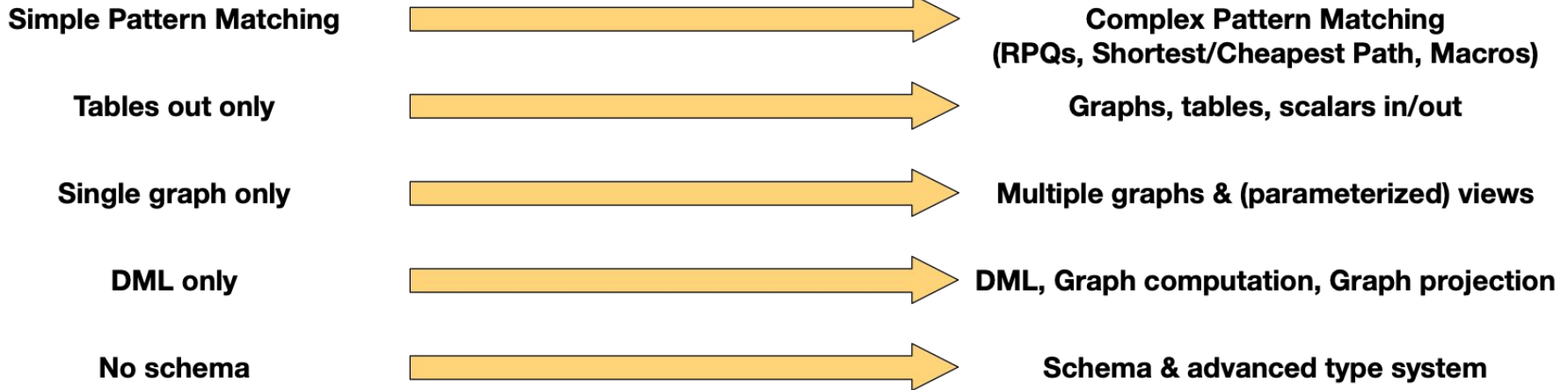
- ① Lead and consolidate existing demand for such a language
- ② Address the specific needs of graph use cases
- ③ Increase the utility of property graph querying
- ④ Drive adoption of graph database systems

# GQL Early Working Draft



- Currently being written
  - Sharing foundational aspects with SQL
  - Using same conventions as SQL and tooling developed by Jim Melton and other SQL editors
- Available to standards process participants only at this time
- Starting point that will see many changes
- Based on
  - *GQL Scope and Features (this talk)*
  - Other WG3 contributions

# From Cypher, PGQL, GSQL, SQL/PGQ to **GQL**



All aligned with basic data types, infrastructure, and expressions of the SQL database

Support for basic tabular manipulation (projection, sorting, grouping etc)

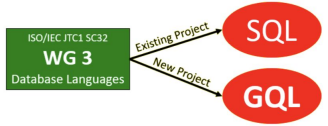
More features are discussed (Indexing)

<http://tiny.cc/gql-scope-and-features>

# BNE-023: GQL Scope and Features

- ① Introduction  
*Inputs, motivation, orientation*
- ② References  
*Related material, incl. designs from openCypher*
- ③ Discussion  
*Overarching design principles, language overview*
- ④ Proposal  
*Project scope, definitions, language features*
- ⑤ Grammar  
*Sketch of proposed syntax - to show structure*

ISO/IEC SC32/WG3/BNE-023  
ANSI INCITS DM32.2-2018-00196  
ANSI INCITS sql-pg-2018-0046r3



The diagram shows a green box on the left labeled "ISO/IEC JTC1 SC32 WG3 Database Languages". Two arrows originate from this box: one labeled "Existing Project" points to a red oval labeled "SQL", and another labeled "New Project" points to a red oval labeled "GQL".

Fig. 1: GQL image (Source: Keith Hare)

### GQL Scope and Features

**Title:** GQL Scope and Features  
**Authors:** Neo4j Query Languages Standards and Research Team<sup>1</sup>  
**Status:** Discussion Paper

**Revisions:** Revision 3, December 14, 2018  
Subeditorial corrections: Added document numbers

Revision 2, November 29, 2018  
Subeditorial corrections: Clarifications in 1.2 Summary of scope; Added 3.6 Combinators; Additions to 4.2 Definitions; Corrections in 3 Discussion, 4.4 Data types; Include tables from [REF-038] for 1.4 Concordances

Revision 1, November 12, 2018  
Subeditorial corrections, including adding of references and related changes, and exchanged order of 4.7 and 4.8; Clarifications in 3.8 Design principles, 3.9 Motivation, 4.2 Definitions, 4.3 Type system, 4.6 Statements for graph pattern matching, 4.7 Statements for modifying graphs, 4.10.1 Nested procedures

Original, October 31, 2018

Copyright © 2018, Neo4j Inc. Please see last page of this document for Apache 2.0 licence grant.

<sup>1</sup> Current members of the Neo4j Query Languages Standards and Research Team are: Alastair Green, Peter Furniss, Tobias Lindacker, Petra Selmer, Hannes Voigt, Stefan Plantikow

1

Publicly available at <http://tiny.cc/gql-scope-and-features>

# BNE-023: GQL Design Principles and Scope [3.9, 4.1]

- ① **New and independent property graph query language**  
*Follow the tradition of existing languages like Cypher, PGQL and SQL/PGQ*  
*Support both standalone implementation or extension of existing SQL-based systems*
- ② **Declarative language**  
*Emphasize what over how (in particular by using pattern matching) to support different implementation strategies*
- ③ **Composable language**  
*Compose procedures from nested sub-procedures and statement sequences in a language closed under graphs and tables*
- ④ **Compatible language**  
*Ensure compatibility with established and widely used features of SQL and avoid idle variation to existing syntax*
- ⑤ **Modern language**  
*Introduce next-generation language features using established designs from existing languages where available*
- ⑥ **Intuitive language**  
*Follow consistent, "whiteboard-friendly", visual syntax (in particular by using "ascii-art" patterns)*

***Covering the full spectrum of features of an industry-grade database query language!***

# BNE-023: Example query [3.1]

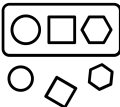
```
// from graph or view "friends" in the catalog  
FROM friends  
  
// match persons a and b that travelled together  
MATCH (a IS Person)-[IS TRAVELLED_TOGETHER]-(b IS Person)  
WHERE a.age = b.age AND a.country = $country AND b.country = $country  
  
// from view parameterized by country  
FROM census($country)  
  
// find out if a and b at some point moved to or where born in a place p  
MATCH SHORTEST (a) ((-)[IS BORN_IN|MOVED_TO]->())* (p)  
                  ((<-)[IS BORN_IN|MOVED_TO]-())* (b)  
  
// that is located in a city c  
MATCH (p)-[IS LOCATED_IN]->(c IS City)  
  
// aggregate number of such pairs per city and age cohort  
RETURN a.age AS age, c.name AS city, count(*) AS pairs GROUP BY age
```



# BNE-023: Topics

① Graphs & Pattern matching 

② Tables & Expressions 

③ Type system & Schema 

④ Modifying and Projecting graphs 

⑤ Query composition & Views 

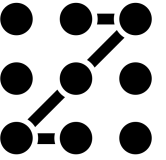
⑥ Schema & Catalog

⑦ Interoperability

⑧ Error handling model

⑨ Security model

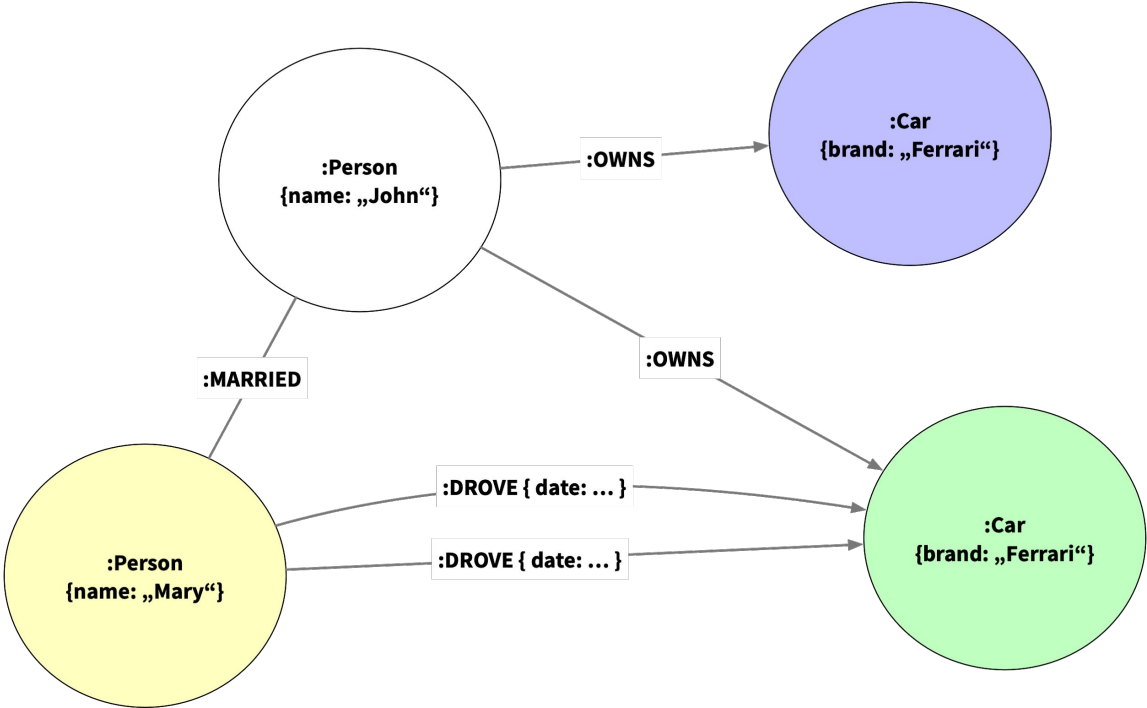
⑩ User defined procedures & functions



# BNE-023: Graphs & Pattern matching [3.4, 4.6]

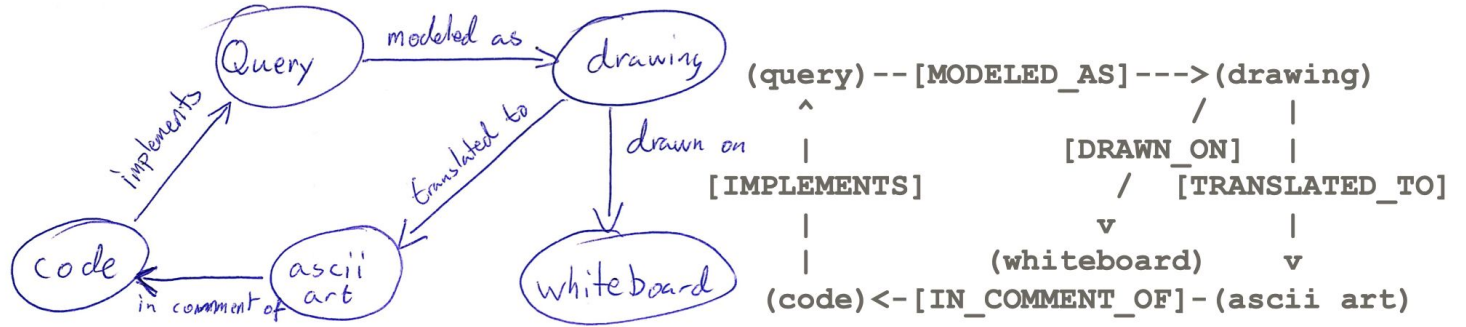
- ① Property graph model
- ② Nodes, Edges, and Paths [4.4.3]
- ③ Patterns [ERF-035, BNE-034]
- ④ Match and path modifiers [4.6.1-4.6.4]
- ⑤ Working with paths [4.6.5]

# BNE-023: Intrinsic Identity



BNE-023:

# Graph Patterns



```
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
```



# BNE-023: Pattern matching modifiers

<path modifiers> for controlling  
path matching semantics

[**ALL**] **SHORTEST** - for shortest path patterns

[**ALL**] **CHEAPEST** - for cheapest path patterns

(both with **TOP** <k>, **MAX** <k> qualifiers, and  
supporting **WITH TIES**)

**REACHES** - unique end nodes with  $\geq 1$  matching path

**ALL** - all paths

**SIMPLE** - may not contain repeated nodes

**TRAIL** - may not contain repeated edges

**ACYCLIC** - may not repeat nodes,  
except allowing the *first* and *last* node to be the same

```
FROM twitter
```

```
MATCH SIMPLE (a) (()-[IS Knows]->())* (b),
```

```
TRAIL (a)-[IS Lives_At]->()
```

```
(()-[IS Bus|Train|Plane]->())*
```

```
()<-[IS Lives_At]-(b)
```

# BNE-023: Pattern matching structure

```
[FROM <graph>]  
MATCH <pattern> {<comma> <pattern> ...}
```

+ optional modifiers to **MATCH**  
for controlling pattern matching behaviour

**OPTIONAL MATCH** - outer join, binds nulls if nothing matches

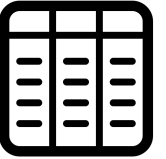
**MANDATORY MATCH** - query fails if nothing matches

**MATCH ...**

- **DIFFERENT (VERTICES | NODES)** - vertex isomorphism
- **DIFFERENT (EDGES | RELATIONSHIPS)** - edge isomorphism
- **UNCONSTRAINED** - homomorphism

```
FROM twitter  
MATCH (a)-[IS Follows]->(b)
```

```
OPTIONAL MATCH (  
  (b)-[p IS Posted]->(m)  
  WHERE p.date > three_days_ago  
)
```



# BNE-023: Tables & Expressions [4.5, 4.9]

- ① Basic table operations (selection, projection, ordering, filtering, slicing) [4.9]
- ② Aggregation and grouping [4.5]
- ③ Tabular set operations (**UNION** [**ALL**]) [4.9]
- ④ Graph element expressions [4.5]
- ⑤ Collection and dictionary expressions [4.5]
- ⑥ Relationship to SQL

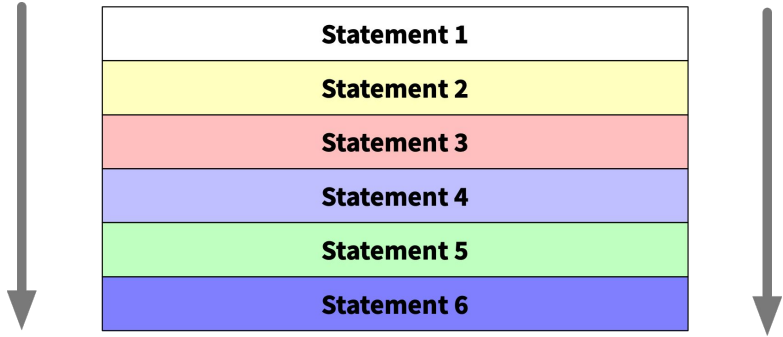
# BNE-023: Why tabular operations in GQL?

- (A)** Pattern matching => (Multi) set of bindings (=> Table)  
=> *Tabular result transformation useful to avoid client-side processing*
- (B)** Bindings main input into graph modifying operations (DML)  
=> *Supported by tabular result transformation and combination*
- (C)** Bindings main input into graph construction operators  
=> *Supported by tabular result transformation and combination*

**Not needed:** Features focussed on tables as a base data model like e.g. referential integrity via foreign key constraints



# BNE-023: Linear statement composition [3.10.3, 4.3.4.3]



- Top-Down flow
- Combined using lateral join
- Statements are update horizons

## Benefits

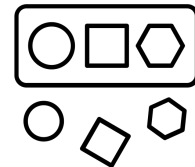
- Natural, linear order used in programming
- Allows query-aggregate-query without (named) nested subqueries
- Allows mixing reading and writing (e.g. returning modified data)
- Solvable using subquery unnesting (maps on "apply" operator)
- **RETURN** has been very positively received by PGM users

# BNE-023: Graph element expressions and functions

- Element access: `n.prop`, `labels(n)`, `properties(n)`, `handle(n)`
- Dynamic label tests
- Element operators: `allDifferent(<elts>)`, `=`, `<>`
- Element functions: `source(e)`, `target(e)`, `(in|out)degree(v)`
- Path functions: `nodes(p)`, `edges(p)`, concatenation

# BNE-023: Collection and dictionary expressions

- Collection literals: `[ a, b, c, ... ]`
- Dictionary literals: `{ alpha: some(a), beta: b+c, ... }`
- Indexing and lookup: `coll[1], dict['alpha']`
- Map comprehensions
- List comprehension
- Functions



# BNE-023: Type system & Schema [3.3, 4.4]

- ① Selected scalar data types from SQL [4.4.1]
- ② Nested data and collections [4.4.2]
- ③ Graph-related data types [4.4.3]
  - Nodes and Edges - with intrinsic identity
  - Paths
  - Graphs
- ④ Advanced type system features [3.3, 4.4.4]
- ⑤ Static and dynamic typing [4.4.5]

# BNE-023: Advanced types

## Heterogeneous types

**MATCH** (n) **RETURN** n.status may give conflicting types (esp. in a large schema)

*Possible type system extension: Union types, e.g. A | B | NULL*

## Complex object types

Support the typing of complex objects like graphs and documents

*Possible type system extension: Graph types, structural types, recursive document type*

## Incomplete type information

Data access (e.g in a big data file system) with runtime metadata discovery

*Possible type system extension: Gradual type for "value of unknown type" ?*

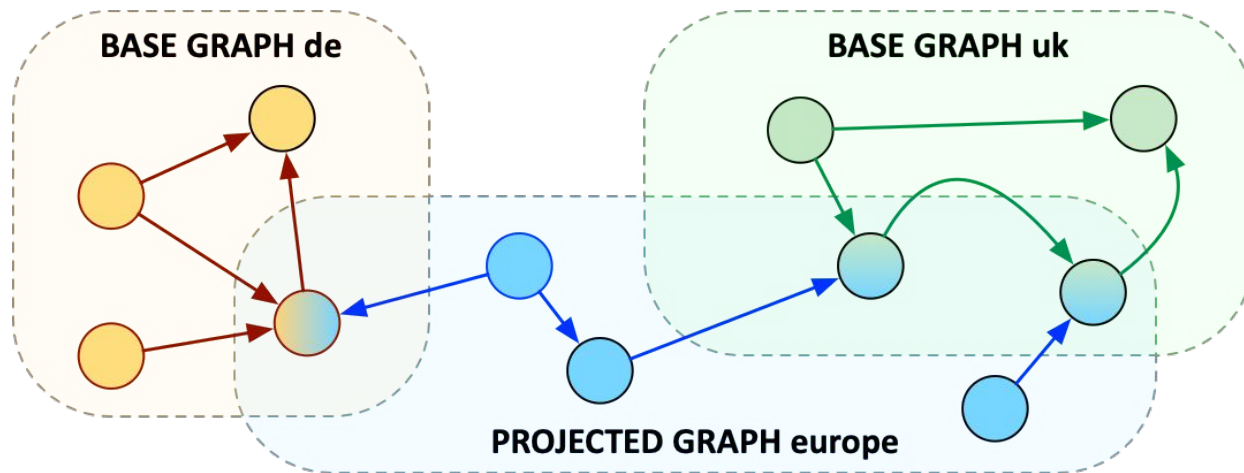


# BNE-023: Modifying and Projecting graphs

- ① Modifying graphs using patterns [3.5, **4.7**]
- ② Graph projection [**4.8**]
- ③ Element sharing [**4.8**]
- ④ Graph combinators (**UNION**, **INTERSECTION**, ...) [**4.8**]



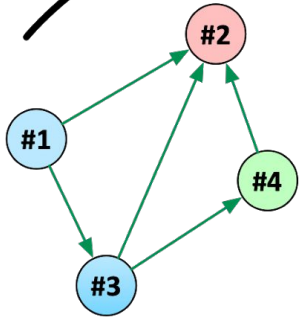
# BNE-023: Graph projection



- Deriving identical elements in the projected graph ("sharing")
- Deriving new elements in the projecte graph
- Shared edges always point to the same (shared) endpoints in the projected graph

# BNE-023: Graph Projection is inverse pattern matching

GRAPH MATCHING



ORIGINAL GRAPH

(#1)->(#2)

(#1)->(#3)

(#3)->(#2)

(#3)->(#4)

(#4)->(#2)

SUBGRAPH MATCHES

a: #1, b: #2

a: #1, b: #3

a: #3, b: #2

a: #3, b: #4

a: #4, b: #2

DRIVING TABLE

NEW ENTITIES

(#1)<-[#5]-(#2)

(#1)<-[#6]-(#1)

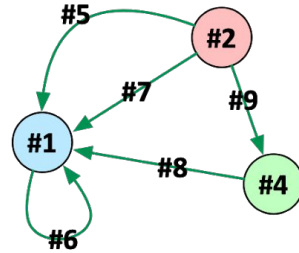
(#1)<-[#7]-(#2)

(#1)<-[#8]-(#4)

(#4)<-[#9]-(#2)

GRAPH CONSTRUCTION  
WITH GROUPING

NEW GRAPH



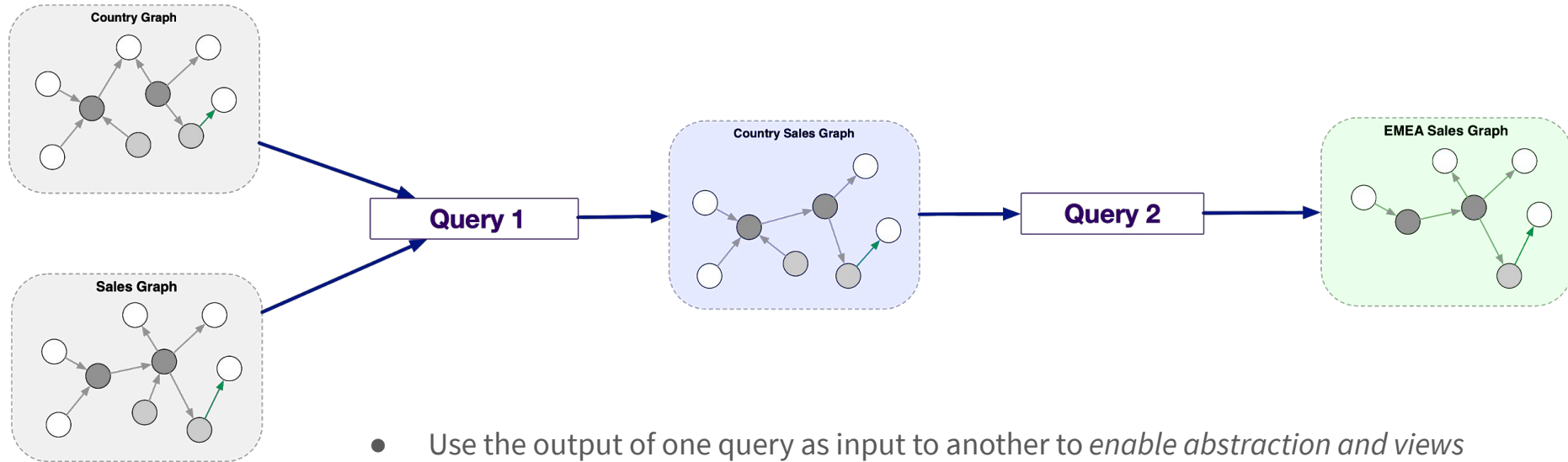


# BNE-023: Query composition & Views



- ① Composable graph procedures [4.3.3-4.3.5]
- ② Parameters and results [4.3.2, 4.3.4.1]
- ③ Linear statement composition [3.10.3, 4.3.4.3]
- ④ Graph views model [3.7]
- ⑤ Updatable views and graph augmentation
- ⑥ Provenance tracking

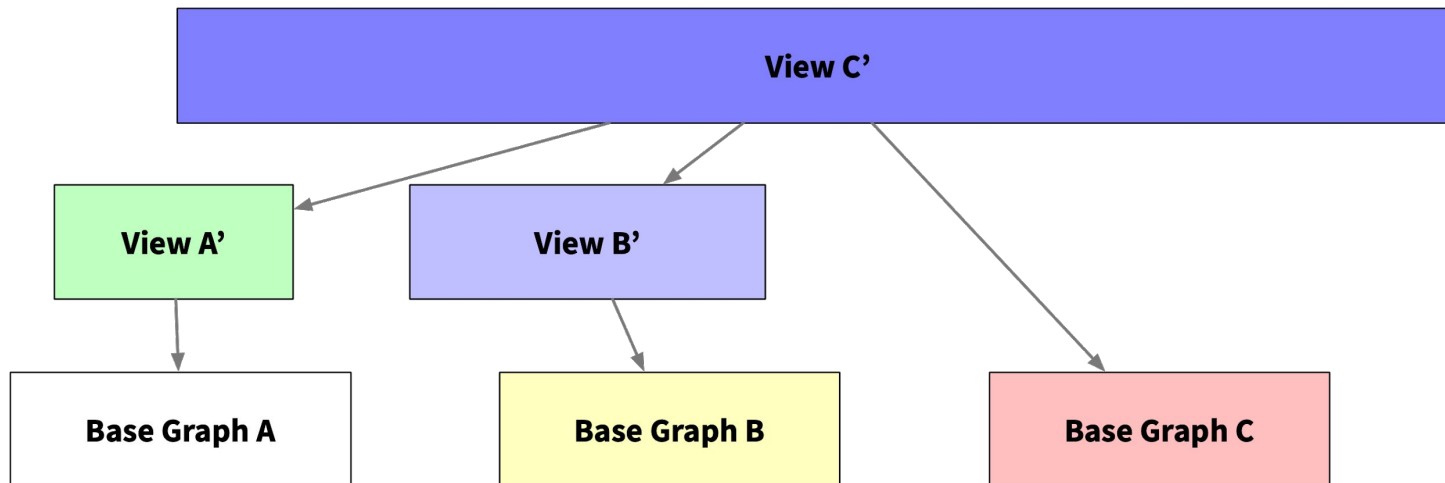
# BNE-023: Queries are procedures [4.3]



- Use the output of one query as input to another to *enable abstraction and views*
- Both for queries with *tabular* output and *graph* output
- Nested queries and procedures [4.10]
- Simple linear composition of tabular output of one query as input to another [3.10.3]



## BNE-023: Views [3.7, 4.12]



- Graph elements in views are derived from other graphs (which may again be views)
- Graph elements are "owned" by their base graph or introducing views
- Derivation graph must form a DAG
- Updates reverse transformation

# BNE-023: Views [3.7, 4.12]

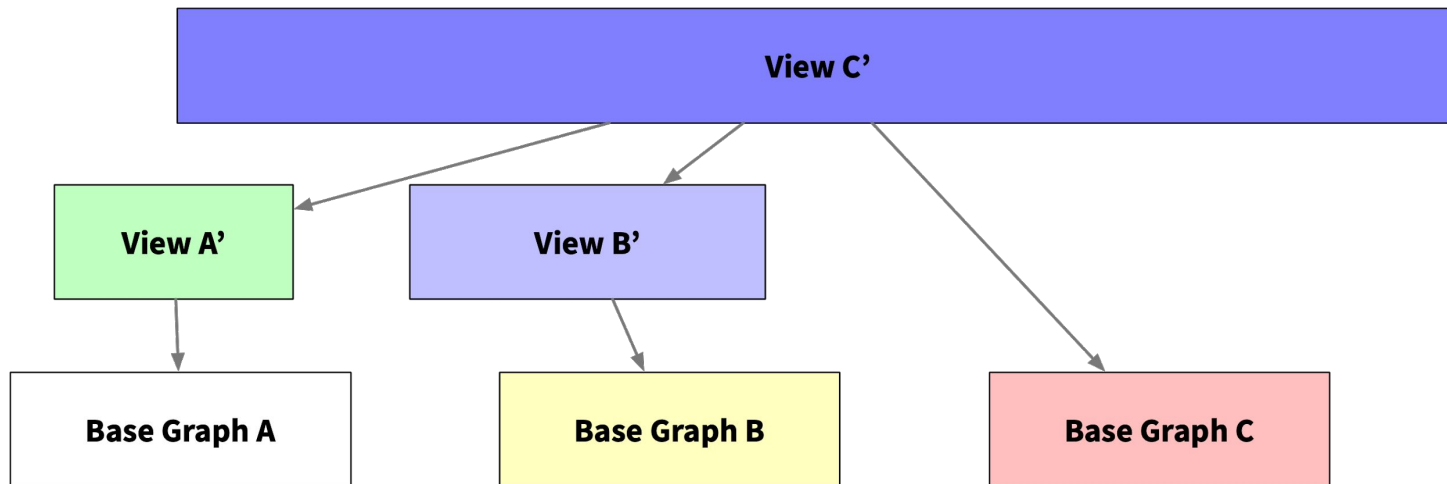
- A (graph) view is a query<sup>†</sup> that returns a graph
  - GQL could also support tabular views
- A view can be used as if it was a graph
  - a tabular view can be used as if it was a table
- Queries (incl. views) can be parameterized
  - allowing the application of the same transformation over compatible graphs

```
CREATE QUERY foaf($input SocialGraph) AS {  
  FROM $input  
  MATCH (a)-[IS FRIEND]-()-[IS FRIEND]-(b)  
  CONSTRUCT (a)-[IS FOAF]-(b)  
}
```

```
FROM foaf(facebook) MATCH ...  
FROM foaf(twitter) MATCH ...
```



# BNE-023: Graph Augmentation



Views behave as if conceptually computed on the fly, including shared graph elements but what if one wants to explicitly express persistently shared graph elements?

Graph augmentation: Allow explicit persistent layered graphs with derived graph elements

Many open questions (e.g. deletion semantics, security model implications)

# BNE-023: GQL Scope and Features

*A new and independent*

*Declarative,*

*Composable,*

*Compatible,*

*Modern,*

*Intuitive*

*Property Graph Query Language*

<http://tiny.cc/gql-scope-and-features>

<http://tiny.cc/gql-scope-digest>

ISO/IEC SC32/WG3-DBR-023  
ANSI INCITS DM32.2-2018-00196  
ANSI INCITS sql-pg-2018-0046x3

Fig. 1: GQL image (Source: Keith Hare)

### GQL Scope and Features

**Title:** GQL Scope and Features  
**Authors:** Neo4j Query Languages Standards and Research Team<sup>1</sup>  
**Status:** Discussion Paper

**Revisions:**

- Revision 3, December 14, 2018  
Subeditorial corrections; Added document numbers
- Revision 2, November 29, 2018  
Subeditorial corrections; Clarifications in 1.2 Summary of scope; Added 3.6 Combinators; Additions to 4.2 Definitions; Corrections in 3 Discussion, 4.4 Data types; Include tables from [ERE-038] for 1.4 Concordances
- Revision 1, November 12, 2018  
Subeditorial corrections, including adding of references and related changes, and exchanged order of 4.7 and 4.8; Clarifications in 3.8 Design principles, 3.9 Motivation, 4.2 Definitions, 4.3 Type system, 4.6 Statements for graph pattern matching, 4.7 Statements for modifying graphs, 4.10.1 Nested procedures
- Original, October 31, 2018

Copyright © 2018, Neo4j Inc. Please see last page of this document for Apache 2.0 licence grant.

<sup>1</sup> Current members of the Neo4j Query Languages Standards and Research Team are: Alastair Green, Peter Furniss, Tobias Lindacker, Petra Selmer, Hannes Voigt, Stefan Plantikow

1