TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

A Semantics of GQL; a New Query Language for Property Graphs Formalized

Formalizing read statements on property graphs

Olof Hendrik Morra, Data Science in Engineering

Department of Computer Science and Mathematics

Who am I?

- Student at Eindhoven University of Technology
- Data Science in Engineering Master
- Database group

Current field in database systems

Relational vs. property graph databases

Relational databases

- Table as underlying data structure
- Relations defined between tables
- SQL
- Oracle, MySQL, Microsoft SQL Server, PostgreSQL, etc.

Property graph databases

- Property graph as underlying data structure
- Relations defined between records
- Cypher, G-CORE, Gremlin, PGQL
- Amazon Neptune, Neo4j, Oracle, OrientDB, etc.

What is GQL?

- Graph Query Language
- ISO Standard draft [1]
- Composable declarative database language (like SQL)
- Property graphs
- CRUD operations
- Regular path queries

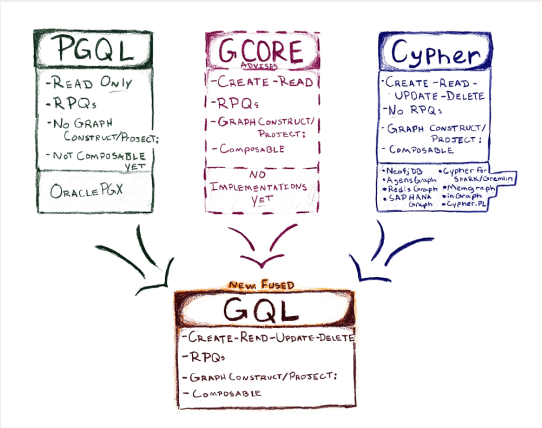


Figure: Combination of three languages into GQL [3]

Goal

- Explore capabilities of GQL
  - Read statements (**FROM-MATCH-RETURN-WHERE**)
- Define data model
- Clarify syntax
- Provide semantics
- Create a database engine
- Provide algebraic representation for input queries

Data model

- Values
  - Identifiers ($\mathcal{E}$ and $\mathcal{N}$)
  - Real numbers ($\mathbb{R}$)
  - Floating point numbers ($\mathbb{F}$)
  - Strings (Σ^*)
  - Truth values
  - set(), multiset(), map(), path()
- Property graph (G)
- Graph database (D)
  - Collection of named property graphs
- Binding table (T)

Property graphs

Based on property graph definition of Angles [2]

A property graph is a tuple $G = (N, E, \rho, \varepsilon, \lambda_N, \lambda_E, \pi)$ where:

- Sets N, E, L and P
- $\rho : E \to (N \times N)$
- $\varepsilon : E \to \{0, 1\}$ (0 = undirected, 1 = directed)
- $\lambda_N : N \to \emptyset$ or $\lambda_N : N \to \mathrm{SET}^1(L)$ or $\lambda_N : N \to \mathrm{SET}^+(L)$ or a combination
- $\lambda_E : E \to \emptyset$ or $\lambda_E : E \to \mathrm{SET}^1(L)$ or $\lambda_E : E \to \mathrm{SET}^+(L)$ or a combination
- $\pi : (N \cup E) \to \mathrm{SET}^+(P) \cup \{\emptyset\}$

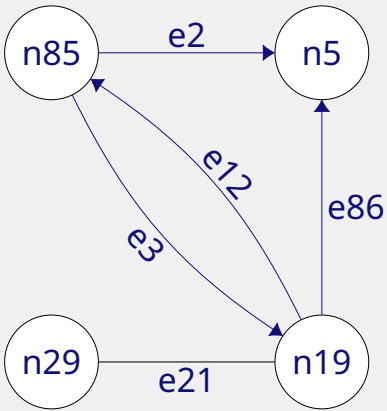
Implementation choices for property graphs

GQL ISO Standard draft allows for differences across implementation for property graphs

- Maximum cardinality of labels
- Maximum cardinality of properties
- Allowing undirected edges

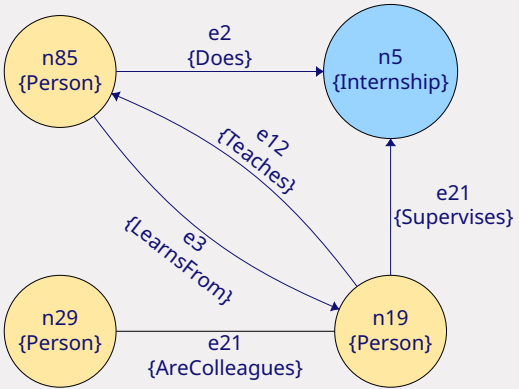
Property graph examples (1)

Most basic property graph



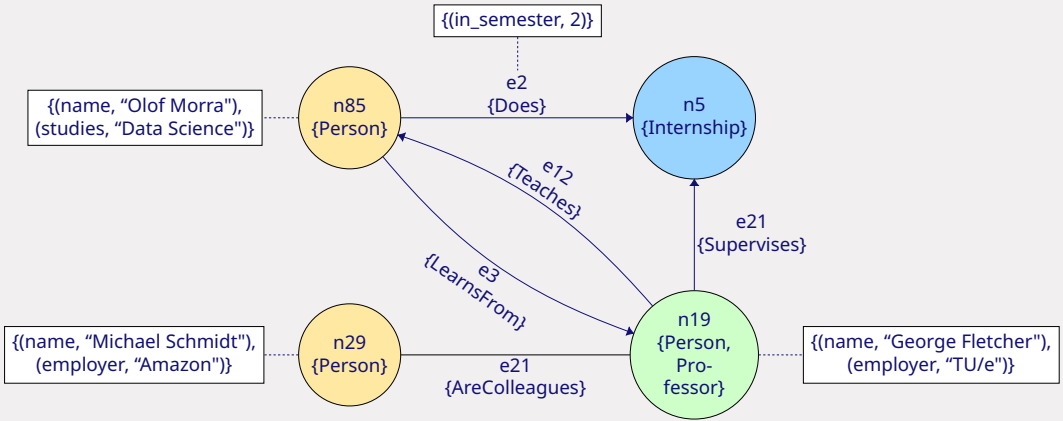
Property graph examples (2)

Property graph with singleton label sets



Property graph examples (3)

Property graph with unbounded label and property sets



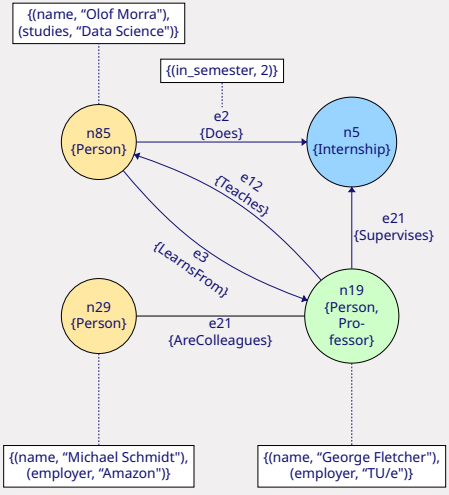
Binding table

A binding table is a tuple $T = (A_T, R, \sigma)$ where:

- A_T is a subset of full names $\mathcal{A}^2$;
- R is a set of records
- $\sigma : \{A_T\} \to \{1, \ldots, |A_T|\}$

Binding table example

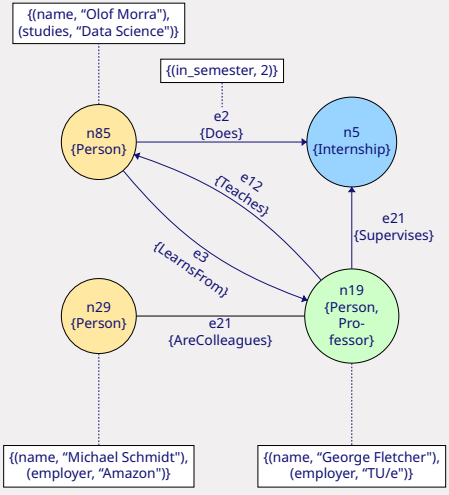
FROM g3
MATCH (x:Person)
RETURN *



Binding table example

FROM g3
MATCH (x:Person)
RETURN *

$$A_T = \{g3.x\}$$
$$R = \{(g3.x : n19), (g3.x : n29), (g3.x : n85)\}$$
$$\sigma = \{g3.x \mapsto 1\}$$
$$T = (A_T, R, \sigma)$$



Binding table example

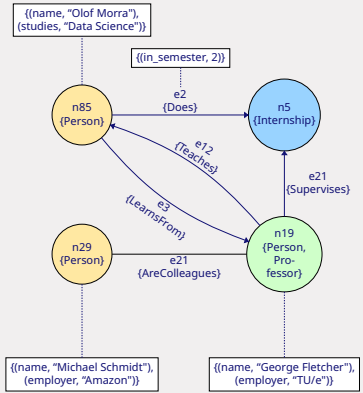
FROM g3
MATCH (x:Person)
RETURN *

$A_T = \{g3.x\}$

$R = \{(g3.x : n19), (g3.x : n29), (g3.x : n85)\}$

$\sigma = \{g3.x \mapsto 1\}$

$T = (A_T, R, \sigma)$



	x
1	*n*19
2	*n*29
3	*n*85

GQL Syntax

Mandatory match with distinct return

From graph *g3* match nodes and return all the distinct names of the nodes, if no match is found throw an error

FROM g3
MANDATORY MATCH (x)
RETURN DISTINCT x.name **AS** name

GQL Syntax

Optional match with return all

Match a node *x* that is connected to an undirected edge and return true if the name is "Michael Schmidt"

OPTIONAL MATCH TRAIL (x)~[]~()
RETURN ALL
x.name = "Michael Schmidt" **AS** is_michael

GQL Syntax

Where clause returning everything

Match a source node *x* that has the name "Olof Morra" and id n85, return the full nodes

MATCH WALK (x)-[]->()
WHERE x.name = "Olof Morra" **AND** x.id = "n85"
RETURN *

GQL Syntax

Label conjunction and disjunction

From graph *g3* match a node that is a person and student, or is only a person, return the node

FROM g3
MATCH SIMPLE
(a:Person & Student | Person)
RETURN a

GQL Syntax

Wildcard label with edge quantifier

From graph *g3* match a path with two edges going right that contain at least one label, return the last target node

FROM g3
MATCH SIMPLE ()-[:%]->{2}(a)
RETURN a

GQL Syntax

Property syntax

From graph *g3* match an edge that has the property *in_semester* equal to 2 and return all edges

```
FROM g3
MATCH ACYCLIC
()<-[b {in_semester: 2}]-()
RETURN b
```

GQL Syntax

Query conjunction

From graph *g3* match all nodes except those who are a person, return the nodes

```
FROM g3
MATCH (x)
RETURN *
EXCEPT ALL
MATCH (y:Person)
RETURN y AS x
```

Formal semantics

Why do we need formal semantics?

- Guide new implementations
- Untangle GQL ISO Standard draft
- Formal data model
- Equivalence of queries
- Prove correctness of optimizations
- Discover optimizations

Pattern matching

1. Node patterns
2. Edge patterns
3. Path patterns
4. Multiple path patterns

Rigid patterns only (patterns of fixed length)

Node patterns

A node pattern χ is a triple (a, L, P) where:

- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is a set of node labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$

Node patterns

A node pattern χ is a triple (a, L, P) where:

- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is a set of node labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$

(x:(Person&Student)|Person {name: "Olof Morra"})
$(x, \{\{Person, Student\}, \{Person\}\}, \{(name, \text{Olof Morra})\}$

Node patterns

A node pattern χ is a triple (a, L, P) where:

- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is a set of node labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$

()
$(\text{nil}, \emptyset, \emptyset)$

Edge patterns

An edge pattern ρ is a tuple (d, a, L, P, I) where:

- $d \in \{\rightarrow, \leftarrow, -\}$ specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is set of edge labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$
- I is (m, n) with $m, n \in \mathbb{N}^+$ where $m = n$

Edge patterns

An edge pattern ρ is a tuple (d, a, L, P, I) where:

- $d \in \{\rightarrow, \leftarrow, -\}$ specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is set of edge labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$
- I is (m, n) with $m, n \in \mathbb{N}^+$ where $m = n$

-[y:Does {in_semester:2}]->
$(\rightarrow, y, \{\{Does\}\}, \{(in\_semester, 2)\}, (1, 1))$

Edge patterns

An edge pattern ρ is a tuple (d, a, L, P, I) where:

- $d \in \{\rightarrow, \leftarrow, -\}$ specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\textbf{nil}\}$
- $L \subset \emptyset \cup \{L_1, \ldots, L_n\}$ for which each $L_i \subset \mathcal{L} \cup \{\%\}$ is set of edge labels
- P is a set of key-value pairs of the form (k, v) where $k \in \mathcal{K}$ and $v \in V$
- I is (m, n) with $m, n \in \mathbb{N}^+$ where $m = n$

~[**IS** AreColleagues]~{2}
$(-, \text{nil}, \{\{\textit{AreColleagues}\}\}, \emptyset, (2, 2))$

Path patterns

A path pattern π is an alternating sequence of node and edge patterns

$$\chi_1 \rho_1 \chi_2 \cdots \rho_{n-1} \chi_n$$

Path patterns

A path pattern π is an alternating sequence of node and edge patterns

$$\chi_1 \rho_1 \chi_2 \cdots \rho_{n-1} \chi_n$$

And can be matched in four modes:

Evaluation mode	Relation	Morphism
WALK	$\models_W$	Homomorphism
TRAIL	$\models_T$	Edge-isomorphism
ACYCLIC	$\models_A$	Node-isomorphism
SIMPLE	$\models_S$	Partial node-isomorphism

Pattern matching

General

A node n from a graph G matches a node pattern $\chi = (a, L, P)$ if:

- A label set in L is contained in n or $L = \emptyset$
- P is contained in n

Pattern matching

General

An edge e from a graph G matches an edge pattern $\rho = (d, a, L, P, I)$ partly if:

- $d = 1$ and e is directed, or $d = 0$ and e is undirected,
- A label set in L is contained in e or $L = \emptyset$
- P is contained in e
- The source and target node of e correspond to the node patterns before and after ρ

Pattern matching

General

A path p in a graph G matches a path pattern π if each node and edge in p correspond to the node and edge pattern at the same place

Pattern matching

Evaluation modes

The general definition holds for $\models_W$.

For $\models_T$ each edge in p must be distinct.

For $\models_A$ each node in p must be distinct.

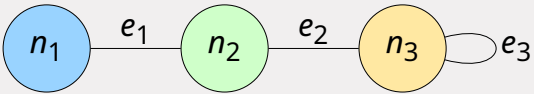
For $\models_S$ each node in p must be distinct, except for the first and last node.

$$\text{match}(\pi, G, u, \models) = \biguplus_{p \text{ in } G} \left\{ u' \;\middle|\; \begin{array}{l} \text{dom}(u') = \text{free}(\pi) - \text{dom}(u) \\ \text{and } (p, G, u \cdot u') \models \pi \end{array} \right\}$$

Pattern Matching

Example $\vDash_W$

	x	y	z
1	n_1	n_2	n_1
2	n_1	n_2	n_3
3	n_2	n_1	n_2
4	n_2	n_3	n_2
5	n_2	n_3	n_3
6	n_3	n_2	n_1
7	n_3	n_2	n_3
8	n_3	n_3	n_2
9	n_3	n_3	n_3



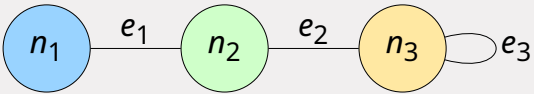
MATCH WALK (x)~[]~(y)~[]~(z)

$$\pi = \underbrace{(x, \emptyset, \emptyset)}_{\chi_1}\underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_1}\underbrace{(y, \emptyset, \emptyset)}_{\chi_2}\underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_2}\underbrace{(z, \emptyset, \emptyset)}_{\chi_3}$$

Pattern Matching

Example $\vDash_T$

	x	y	z
1	n_1	n_2	n_3
2	n_2	n_3	n_3
3	n_3	n_2	n_1
4	n_3	n_3	n_2



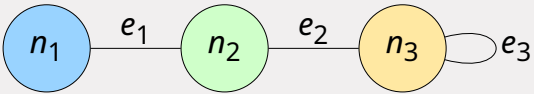
MATCH TRAIL (x)~[]~(y)~[]~(z)

$$\pi = \underbrace{(x, \emptyset, \emptyset)}_{\chi_1} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_1} \underbrace{(y, \emptyset, \emptyset)}_{\chi_2} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_2} \underbrace{(z, \emptyset, \emptyset)}_{\chi_3}$$

Pattern Matching

Example $\vDash_A$

	x	y	z
1	n_1	n_2	n_3
2	n_3	n_2	n_1



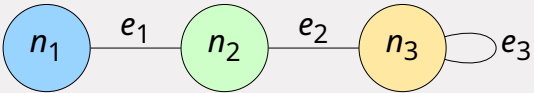
MATCH ACYCLIC (x)~[]~(y)~[]~(z)

$$\pi = \underbrace{(x, \emptyset, \emptyset)}_{\chi_1} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_1} \underbrace{(y, \emptyset, \emptyset)}_{\chi_2} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_2} \underbrace{(z, \emptyset, \emptyset)}_{\chi_3}$$

Pattern Matching

Example $\models_S$

	x	y	z
1	n_1	n_2	n_1
2	n_1	n_2	n_3
3	n_2	n_1	n_2
4	n_2	n_3	n_2
5	n_3	n_2	n_1
6	n_3	n_2	n_3



MATCH SIMPLE (x)~[]~(y)~[]~(z)

$$\pi = \underbrace{(x, \emptyset, \emptyset)}_{\chi_1} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_1} \underbrace{(y, \emptyset, \emptyset)}_{\chi_2} \underbrace{(-, \text{nil}, \emptyset, \emptyset, (1,1))}_{\rho_2} \underbrace{(z, \emptyset, \emptyset)}_{\chi_3}$$

Matching multiple patterns

Obtain a tuple of patterns $\bar{\pi} = (\pi_1, \ldots, \pi_n)$ with their corresponding evaluation modes $\vDash_\pi = (\vDash_1, \ldots, \vDash_n)$

Let $M_i = \mathsf{match}(\pi_i, G, u_i, \vDash_i)$ for each $i \in \{1, \ldots, n\}$

The output is $M = M_1 \times \cdots \times M_n$, i.e. the cross product combining the domains

Semantic function

Semantic function $[\![\cdot]\!]$

Defined on graph database D and current working graph G

Denoted by $[\![\cdot]\!]_{D,G}$

Also defined on assignment $u = (a_1 : v_1, \ldots, a_n : v_n)$

Denoted by $[\![\cdot]\!]_{D,G,u}$

Semantics of expressions

Values, variables and property reference

- $[\![v]\!]_{D,G,u} = v$
- $[\![a]\!]_{D,G,u} = u(a)$
- $$[\![e.k]\!]_{D,G,u} = \begin{cases} v & \text{if } [\![e]\!]_{D,G,u} \in N \cup E \text{ and } (k,v) \in \phi([\![e]\!]_{D,G,u}) \\ \textbf{unknown} & \text{if } [\![e]\!]_{D,G,u} \in N \cup E \text{ and } (k,v) \notin \phi([\![e]\!]_{D,G,u}) \\ \text{Not valid} & \text{otherwise} \end{cases}$$

Semantics of expressions

Logic

- $[\![e \texttt{ OR } e']\!]_{D,G,u} = \begin{cases} \texttt{true} & \text{if } [\![e]\!]_{D,G,u} = \texttt{true} \text{ or } [\![e']\!]_{D,G,u} = \texttt{true} \\ \texttt{false} & \text{if } [\![e]\!]_{D,G,u} = [\![e']\!]_{D,G,u} = \texttt{false} \\ \texttt{unknown} & \text{otherwise} \end{cases}$
- Similar for **AND**, **XOR**, and **NOT**

Semantics of expressions

Numbers and strings

- $[\![e = e']\!]_{D,G,u} = \begin{cases} \texttt{true} & \text{if } [\![e]\!]_{D,G,u} = [\![e']\!]_{D,G,u} \\ \texttt{false} & \text{otherwise} \end{cases}$
- $[\![e <> e']\!]_{D,G,u} = \begin{cases} \texttt{false} & \text{if } [\![e]\!]_{D,G,u} = [\![e']\!]_{D,G,u} \\ \texttt{true} & \text{otherwise} \end{cases}$
- Other comparisons based on their exact value, IEEE754 Standard, or lexographic ordering

Semantics of queries

Return statement

- $[\![\textbf{RETURN ALL } *]\!]_{D,G}(T) = T$
- $[\![\textbf{RETURN ALL } e_1 \textbf{ AS } a_1], \ldots, e_m \textbf{ AS } a_m]]\!]_{D,G}(T) =$
$$\biguplus_{u \in T} \{(a'_1 : [\![e_1]\!]_{D,G,u}, \ldots, a'_m : [\![e_1]\!]_{D,G,u})\}$$
- $[\![\textbf{RETURN DISTINCT} \ldots]\!]_{D,G}(T) = \delta([\![\textbf{RETURN ALL } \ldots]\!]_{D,G}(T))$

Semantics of clauses

Where and from clause

- $[\![\textbf{WHERE}\ \ e]\!]_{D,G}(T) = \{u \in T \mid [\![e]\!]_{D,G,u} = \textbf{true}\}$
- $[\![\textbf{MATCH}\ \ \bar{\pi}\ \ \textbf{WHERE}\ \ e]\!]_{D,G}(T) = [\![\textbf{WHERE}\ \ e]\!]_{D,G}\Big([\![\textbf{MATCH}\ \ \bar{\pi}]\!]_{D,G}(T)\Big)$
- $[\![\textbf{FROM}\ \ s\ \text{match\_clause}]\!]_{D,G} = [\![\text{match\_clause}]\!]_{D,D(s)}$, i.e. set the current working graph to $D(s)$

Semantics of clauses

Match clause

- $[\![\textbf{MATCH}\ \bar{\pi}]\!]_{D,G}(T) = \biguplus_{u \in T} \{u \times u' \mid u' \in \text{match}(\bar{\pi}, G', u, \vDash)\}$
- $[\![\textbf{OPTIONAL MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) =$

$$\begin{cases} [\![\textbf{MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) & \text{if } [\![\textbf{MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) \neq \emptyset \\ \biguplus_{u \in T} (u, (\text{free}(u, \bar{\pi}) : \textbf{null})) & \text{otherwise} \end{cases}$$

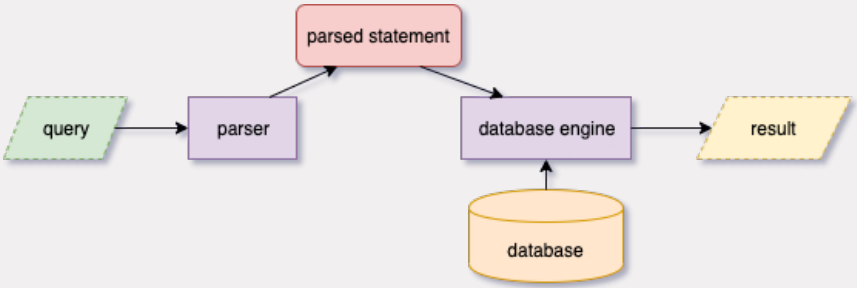
- $[\![\textbf{MANDATORY MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) =$

$$\begin{cases} [\![\textbf{MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) & \text{if } [\![\textbf{MATCH}\ \bar{\pi}\ \textbf{WHERE}\ e]\!]_{D,G}(T) \neq \emptyset \\ \text{Not valid} & \text{otherwise} \end{cases}$$

Interpreter

Process GQL queries

Output algebra formalizations



Implementation choices for GQL

GQL ISO Standard draft allows for differences across implementation for property graphs

- Evaluation order if not defined by parentheses
- Length/precision of base types
- Access to identifiers
- Ordering of rows if not defined by query

Demo

GitHub repository of GQL parser

Latex output from demo

$$G = g3$$
$$\chi_1 = (x, \{\{Person\}\}, \emptyset)$$
$$\pi_1 = \chi_1$$
$$\vDash_1 = \vDash_W$$
$$\bar{\pi} = (\pi_1)$$
$$\vDash = (\vDash_1)$$
$$T = \{()\}$$
$$M_1 = [\![\textbf{MATCH}\ \ \bar{\pi}]\!]_{D,G}(T) = \biguplus_{u \in T} \{u \times u' \mid u' \in \mathsf{match}(\bar{\pi}, G, u, \vDash)\}$$
$$Q_1 = [\![\textbf{RETURN}\ \ \textbf{ALL}\ \ *]\!]_{D,G}(M_1) = M_1$$

FROM g3
MATCH (x:Person)
RETURN *

Limitations

Not all of GQL is formalized

- Only read statements considered

Interpretation of the GQL ISO Standard draft

- Mathematical representation of verbal definition

Experimental implementation

- First implementation out there
- Naive query planner

Data input format

- JSON only now

Further research

Discuss interpretation

- GQL ISO Standard draft not always straightforward

Three-valued logic

- Guagliardo et al. [4] proof that SQL has the same expressiveness without null

Extend syntax, semantics and interpreter

Further research

Query optimization

- Proof empty result of entire query based on output of one query

Graph serialization

- Different evaluation modes might thrive under different graph serializations

Graph indexing

- How to index data for searching in a property graph under different evaluation modes

Thank you

Questions?

Bibliography I

- ISO/IEC 39075:202y: Information technology - Database languages - GQL (draft), 2021-02-02.
- R. Angles.
  The Property Graph Database Model.
  In *AMW*, 2018.
- A. Green.
  GQL Manifesto, 2018.
- P. Guagliardo and L. Libkin.
  A formal semantics of SQL queries, its validation, and applications.
  Proceedings of the VLDB Endowment, 11(1):27–39, 2017.

GQL vs. Cypher

1. Different morphisms
2. Directed and undirected edges
3. Undirected means no direction
4. Ternary logic

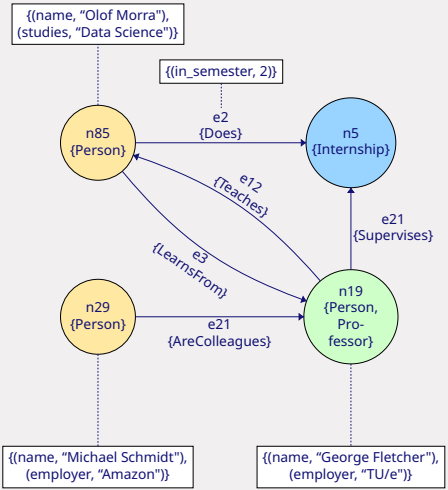
1. Edge isomorphism
2. Only directed edges
3. Undirected (at query time) means going left or right
4. Ternary logic

GQL vs. Cypher

Example GQL vs. Cypher

GQL:
MATCH TRAIL (x)~[]~()
RETURN *
Returns nothing

Cypher:
MATCH (x)-[]-()
RETURN *
Returns each starting and end node of an edge

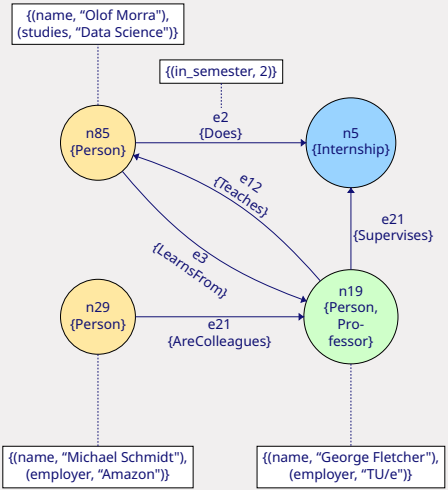


GQL vs. Cypher

Example GQL vs. Cypher

GQL:
MATCH TRAIL (x)-[]->{3}()
RETURN *
Equal queries

Cypher:
MATCH (x)-[*3]->()
RETURN *

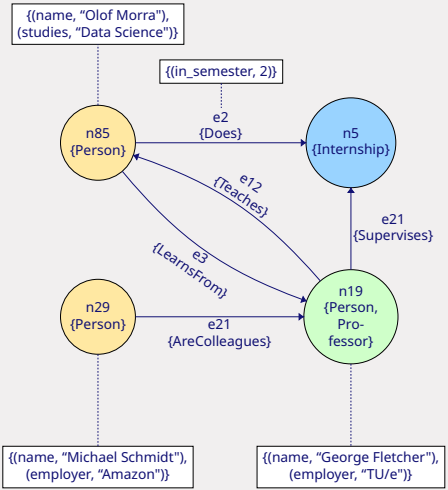


GQL vs. Cypher

Example GQL vs. Cypher

GQL:
MATCH (x)-[]->{3}(y)
RETURN *
Outputs *n*85 for x and *n*19 for y

Cypher:
MATCH (x)-[*3]->()
RETURN *
Does not output the above, as *e*3 cannot be used twice



Formal pattern matching

General

Node n of graph G matches node pattern $\chi = (a, L, P)$ with assignment u, $(n, G, u) \vDash_W \chi$, if all of the following hold:

- $a = \text{nil}$ or $u(a) = n$,
- $L = \emptyset$ or $\exists L_i \in L : L_i \subseteq \lambda_N(n) \vee (\% \in L_i \wedge \lambda_N(n) \neq \emptyset)$,
- $P \subseteq \phi(n)$.

Formal pattern matching

General

By induction on m, let χ be a node pattern, p a path with starting node n_p, π a rigid path pattern and $\rho = (d, a, L, P, (m, m))$.
Then, $(n \cdot p, G, u) \vDash_W \chi\rho\pi$ for $m = 0$ if:

- $a = \text{nil}$ or $u(a) = n$,
- $(n, G, u) \vDash_W \chi$,
- $(p, G, u) \vDash_W \pi$,
- $n = n_p$.

Formal pattern matching

General

For $m \geq 1$, we have that $(n_1 e_1 \cdots e_m n_{m+1} \cdot p, G, u) \vDash_W \chi \rho \pi$ if all of the following hold:

- $a = \text{nil}$ or $u(a) = \text{set}(e_1, \ldots, e_m)$,
- $(n_1, G, u) \vDash_W \chi$,
- $(p, G, u) \vDash_W \pi$,
- $n_{m+1} = n_p$;

Formal pattern matching

General

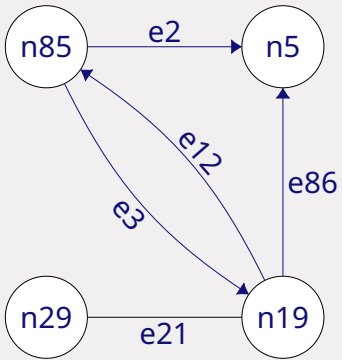
For every edge e_i with $i \in \{1, \ldots, m\}$ if all the following hold:

- $\varepsilon(e_i) = \begin{cases} 1 & \text{if } d \text{ is } \rightarrow \text{ or } \leftarrow \\ 0 & \text{if } d \text{ is } - \end{cases}$;
- $L = \emptyset$ or $\exists L_i \in L : L_i \subseteq \lambda_E(e_i) \vee (\% \in L_i \wedge \lambda_E(e_i) \neq \emptyset)$;
- $P \subseteq \phi(e_i)$;
- $\eta(e_i) \in \begin{cases} \{(n_i, n_{i+1})\} & \text{if } d \text{ is } \rightarrow \\ \{(n_{i+1}, n_i)\} & \text{if } d \text{ is } \leftarrow \\ \{(n_i, n_{i+1}), (n_{i+1}, n_i)\} & \text{if } d \text{ is } - \end{cases}$.

GQL query example

Different graph implementations

FROM g
MATCH (p:Person)
WHERE p.name = "Olof Morra"
RETURN p.studies

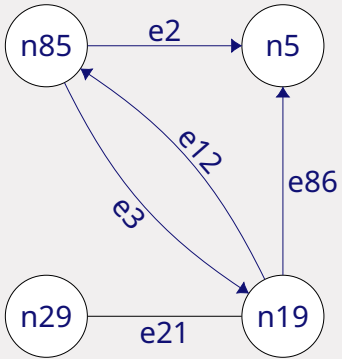


GQL query example

Different graph implementations

FROM g
MATCH (p:Person)
WHERE p.name = "Olof Morra"
RETURN p.studies

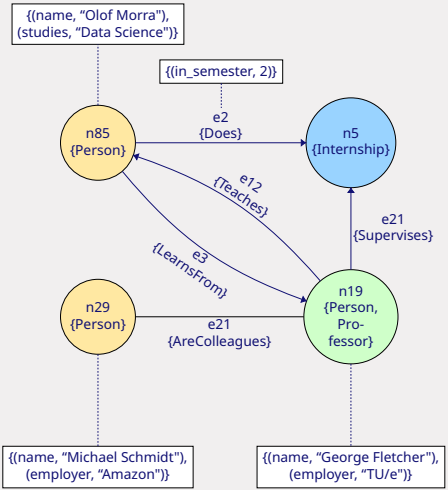
Error if labels are not allowed, no matches found if labels are allowed



GQL query example

Different graph implementations

```
FROM g3
MATCH (p:Person)
WHERE p.name = "Olof Morra"
RETURN p.studies
```



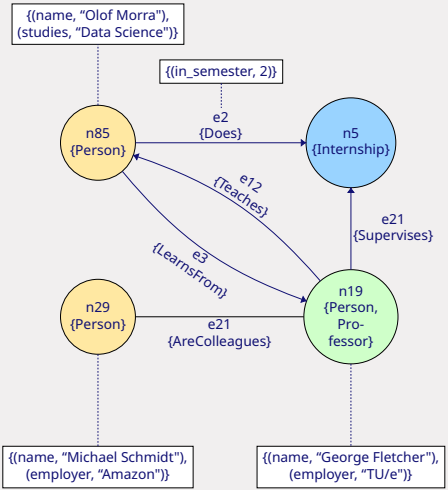
GQL query example

Different graph implementations

FROM g3
MATCH (p:Person)
WHERE p.name = "Olof Morra"
RETURN p.studies

Returns:

	x
1	"Data Science"



Semantics extra

Nulls and truth values

- $[\![e \text{ comp\_op } e']\!]_{D,G,u} = \textbf{unknown}$ if either $[\![e]\!]_{D,G,u} = \textbf{null}$ or $[\![e']\!]_{D,G,u} = \textbf{null}$, for comp_op $\in \{<, \leq, \geq, >, =, <>\}$.
- $[\![e \texttt{ IS UNKNOWN}]\!]_{D,G,u} = \begin{cases} \textbf{true} & \text{if } [\![e]\!]_{D,G,u} = \textbf{unknown} \\ \textbf{false} & \text{if } [\![e]\!]_{D,G,u} \neq \textbf{unknown} \end{cases}$
- $[\![e \texttt{ IS NOT UNKNOWN}]\!]_{D,G,u} = \begin{cases} \textbf{true} & \text{if } [\![e]\!]_{D,G,u} \neq \textbf{unknown} \\ \textbf{false} & \text{if } [\![e]\!]_{D,G,u} = \textbf{unknown} \end{cases}$
- Latter two similar for **TRUE** and **FALSE**

Semantics extra

Query conjunctions

- $[\![Q_1 \ \texttt{UNION ALL} \ Q_2]\!]_{D,G}(T) = [\![Q_1]\!]_{D,G}(T) \uplus [\![Q_2]\!]_{D,G}(T)$
- $[\![Q_1 \ \texttt{UNION DISTINCT} \ Q_2]\!]_{D,G}(T) = \delta([\![Q_1 \ \texttt{UNION ALL} \ Q_2]\!]_{D,G}(T))$
- $[\![Q_1 \ \texttt{UNION MAX} \ Q_2]\!]_{D,G}(T) =$
  $[\![Q_1 \ \texttt{UNION ALL} \ Q_2]\!]_{D,G}(T) - [\![Q_1 \ \texttt{INTERSECT ALL} \ Q_2]\!]_{D,G}(T)$
- Also for intersection, difference, and otherwise