



# 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

# 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
- Composable declarative database language (like SQL)
- Property graphs
- CRUD operations
- Regular path queries

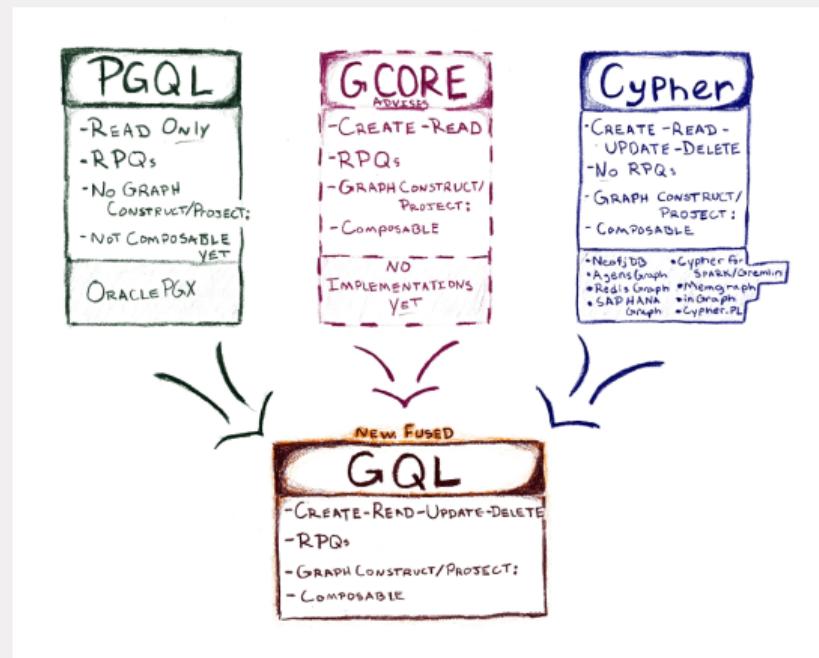


Figure: Combination of three languages into GQL [2]

# 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 ( $\Sigma^*$ )
  - ▶ 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 [1]

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

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

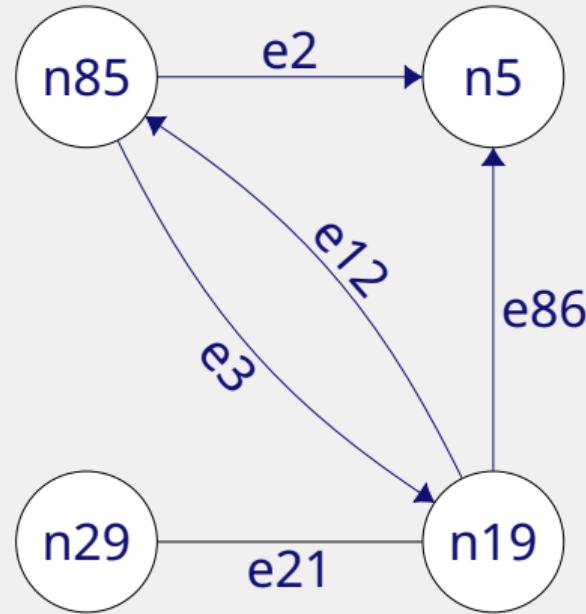
## Implementation choices for property graphs

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

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

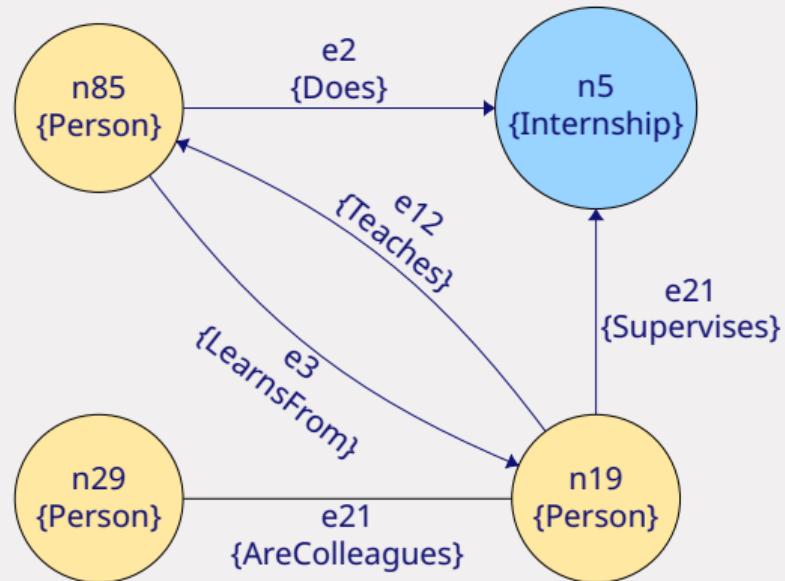
# Property graph examples (1)

Most basic property graph



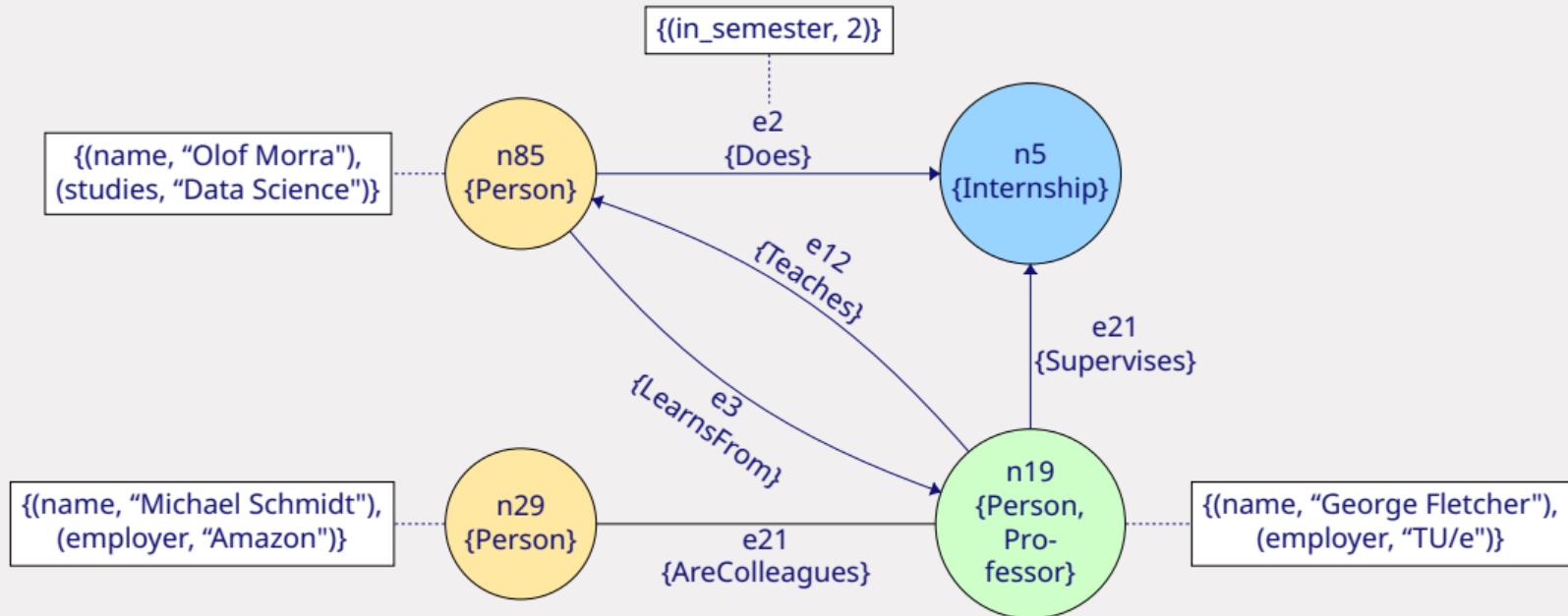
## Property graph examples (2)

Property graph with singleton label sets



# Property graph examples (3)

Property graph with unbounded label and property sets



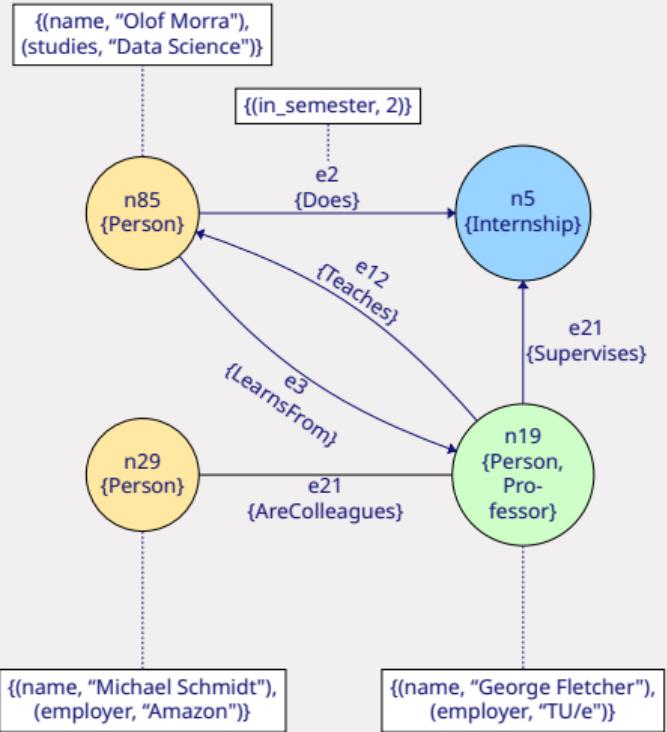
## Binding table

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
- $\sigma : \{A_T\} \rightarrow \{1, \dots, |A_T|\}$

# Binding table example

```
FROM g3
MATCH (x:Person)
RETURN *
```



# Binding table example

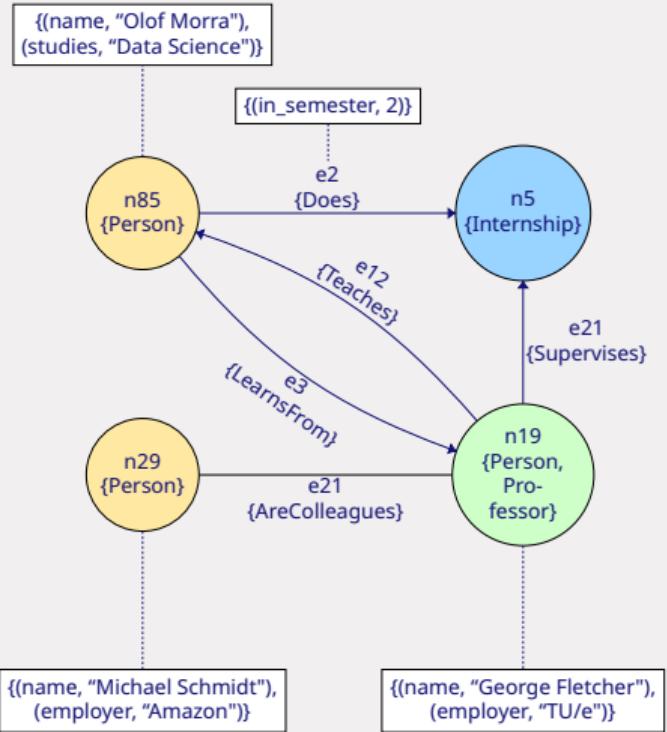
```
FROM g3  
MATCH (x:Person)  
RETURN *
```

$$A_T = \{g3.x\}$$

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

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

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



# Binding table example

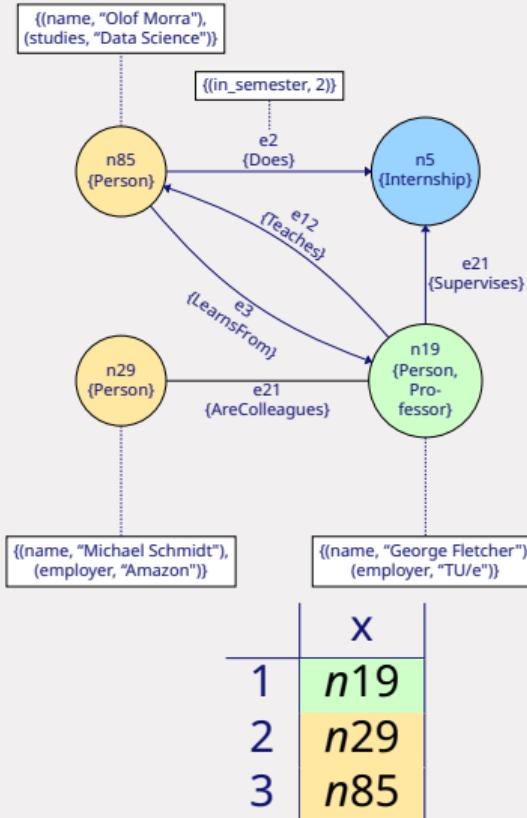
```
FROM g3  
MATCH (x:Person)  
RETURN *
```

$$A_T = \{g3.x\}$$

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

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

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



# 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** ( $x$ ) $\sim[]\sim()$

**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  $\chi$  is a triple  $(a, L, P)$  where:

- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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  $\chi$  is a triple  $(a, L, P)$  where:

- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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  $v \in V$

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

$(x, \{\{\text{Person}, \text{Student}\}, \{\text{Person}\}\}, \{(name, \text{Olof Morra})\})$

## Node patterns

A node pattern  $\chi$  is a triple  $(a, L, P)$  where:

- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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  $v \in V$

$\textcolor{blue}{()}$

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

## Edge patterns

An edge pattern  $\rho$  is a tuple  $(d, a, L, P, I)$  where:

- $d \in \{\rightarrow, \leftarrow, -\}$  specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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  $\rho$  is a tuple  $(d, a, L, P, I)$  where:

- $d \in \{\rightarrow, \leftarrow, -\}$  specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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}]->  
 $(\rightarrow, y, \{\{\text{Does}\}\}, \{(in\_semester, 2)\}, (1, 1))$

## Edge patterns

An edge pattern  $\rho$  is a tuple  $(d, a, L, P, I)$  where:

- $d \in \{\rightarrow, \leftarrow, -\}$  specifies the direction of the edge
- $a \in \mathcal{A} \cup \{\text{nil}\}$
- $L \subset \emptyset \cup \{L_1, \dots, L_n\}$  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  $m = n$

$\sim[\text{IS AreColleagues}]^{\sim}\{2\}$

$(-, \text{nil}, \{\{\text{AreColleagues}\}\}, \emptyset, (2, 2))$

## Path patterns

A path pattern  $\pi$  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  $\pi$  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
<b>WALK</b>	$\models_W$	Homomorphism
<b>TRAIL</b>	$\models_T$	Edge-isomorphism
<b>ACYCLIC</b>	$\models_A$	Node-isomorphism
<b>SIMPLE</b>	$\models_S$	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  $\rho$

# Pattern matching

## General

A path  $p$  in a graph  $G$  matches a path pattern  $\pi$  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.

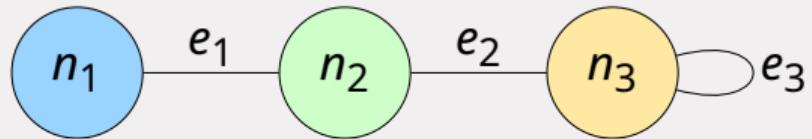
For  $\models_S$  each node in  $p$  must be distinct, except for the first and last node.

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

# 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	$n_3$	$n_3$	$n_3$



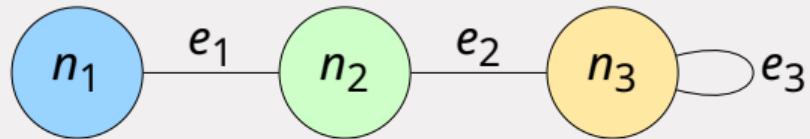
**MATCH WALK** ( $x \sim [] \sim (y \sim [] \sim (z \sim [] \sim []))$ )

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

# Pattern Matching

Example  $\models_T$

	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	$n_3$	$n_3$	$n_2$



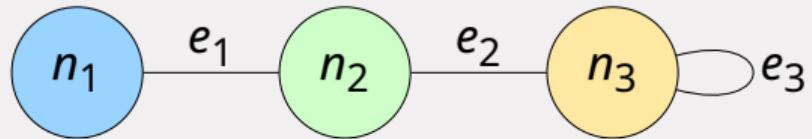
**MATCH TRAIL**  $(x) \sim [] \sim (y) \sim [] \sim (z)$

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

# Pattern Matching

Example  $\vdash_A$

	x	y	z
1	$n_1$	$n_2$	$n_3$
2	$n_3$	$n_2$	$n_1$



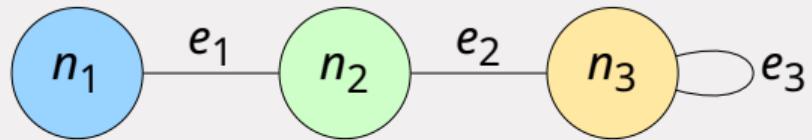
**MATCH ACYCLIC (x)~[]~(y)~[]~(z)**

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

# Pattern Matching

Example  $\vdash_S$

	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**  $(x) \sim [] \sim (y) \sim [] \sim (z)$

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

## Matching multiple patterns

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

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

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

## Semantic function

Semantic function  $\llbracket \cdot \rrbracket$

Defined on graph database  $D$  and current working graph  $G$

Denoted by  $\llbracket \cdot \rrbracket_{D,G}$

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

Denoted by  $\llbracket \cdot \rrbracket_{D,G,u}$

# Semantics of expressions

## Values, variables and property reference

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

# Semantics of expressions

## Logic

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

# Semantics of expressions

## Numbers and strings

- $\llbracket e = e' \rrbracket_{D,G,u} = \begin{cases} \text{true} & \text{if } \llbracket e \rrbracket_{D,G,u} = \llbracket e' \rrbracket_{D,G,u} \\ \text{false} & \text{otherwise} \end{cases}$
- $\llbracket e <> e' \rrbracket_{D,G,u} = \begin{cases} \text{false} & \text{if } \llbracket e \rrbracket_{D,G,u} = \llbracket e' \rrbracket_{D,G,u} \\ \text{true} & \text{otherwise} \end{cases}$
- Other comparisons based on their exact value, IEEE754 Standard, or lexicographic ordering

# Semantics of queries

## Return statement

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

# Semantics of clauses

## Where and from clause

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

# Semantics of clauses

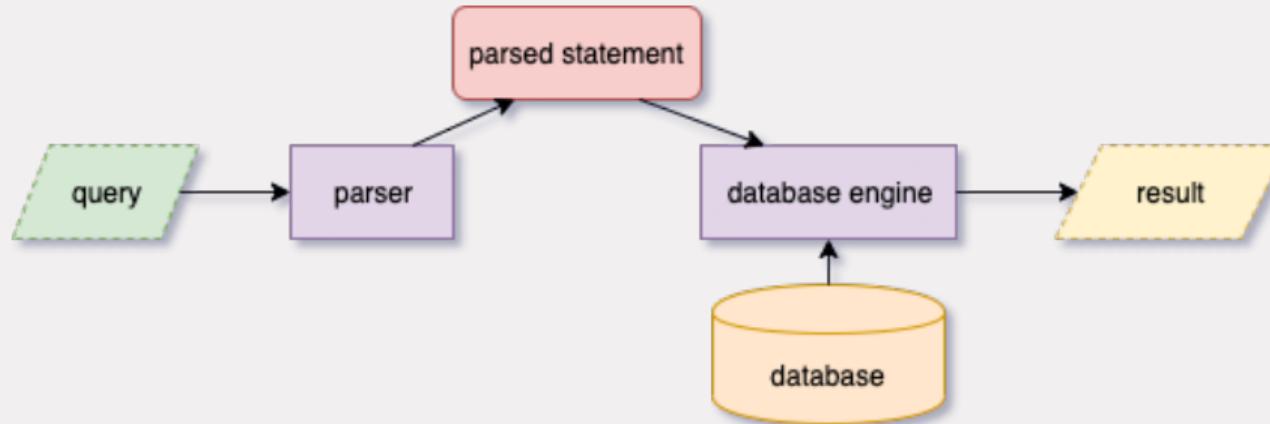
## Match clause

- $\llbracket \text{MATCH } \bar{\pi} \rrbracket_{D,G}(T) = \biguplus_{u \in T} \{u \times u' \mid u' \in \text{match}(\bar{\pi}, G', u, \models)\}$
- $\llbracket \text{OPTIONAL MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) = \begin{cases} \llbracket \text{MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) & \text{if } \llbracket \text{MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) \neq \emptyset \\ \biguplus_{u \in T} (u, (\text{free}(u, \bar{\pi}) : \text{null})) & \text{otherwise} \end{cases}$
- $\llbracket \text{MANDATORY MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) = \begin{cases} \llbracket \text{MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) & \text{if } \llbracket \text{MATCH } \bar{\pi} \text{ WHERE } e \rrbracket_{D,G}(T) \neq \emptyset \\ \text{Not valid} & \text{otherwise} \end{cases}$

# 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

$G = g3$

$\chi_1 = (x, \{\{Person\}\}, \emptyset)$

$\pi_1 = \chi_1$

$\models_1 = \models_W$

$\bar{\pi} = (\pi_1)$

$\models = (\models_1)$

$T = \{()\}$

$M_1 = [\![\text{MATCH } \bar{\pi}]\!]_{D,G}(T) = \biguplus_{u \in T} \{u \times u' \mid u' \in \text{match}(\bar{\pi}, G, u, \models)\}$

$Q_1 = [\![\text{RETURN ALL } *]\!]_{D,G}(M_1) = M_1$

```
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. [3] 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

-  R. Angles.  
The property graph database model.  
In *AMW*, 2018.
-  A. Green.  
*GQL Manifesto*, 2018.  
Accessed November 24, 2020.
-  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 \***

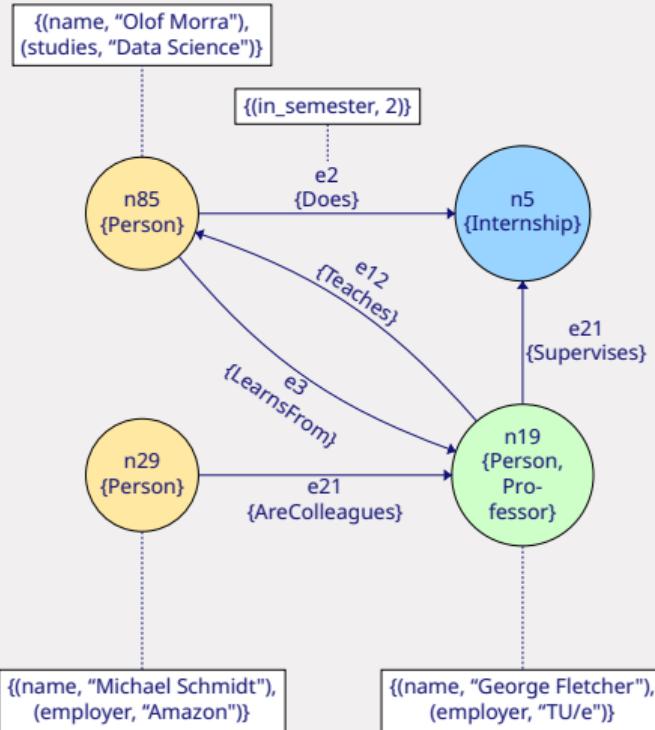
Returns nothing

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}()**

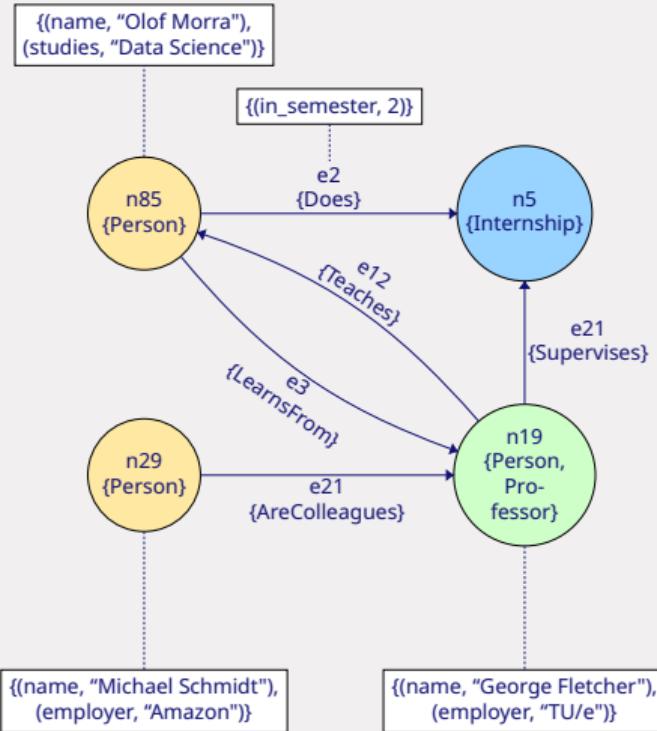
**RETURN \***

Equal queries

Cypher:

**MATCH (x)-[\*3]->()**

**RETURN \***



# GQL vs. Cypher

## Example GQL vs. Cypher

GQL:

**MATCH** (x)-[]->{3}(y)

**RETURN** \*

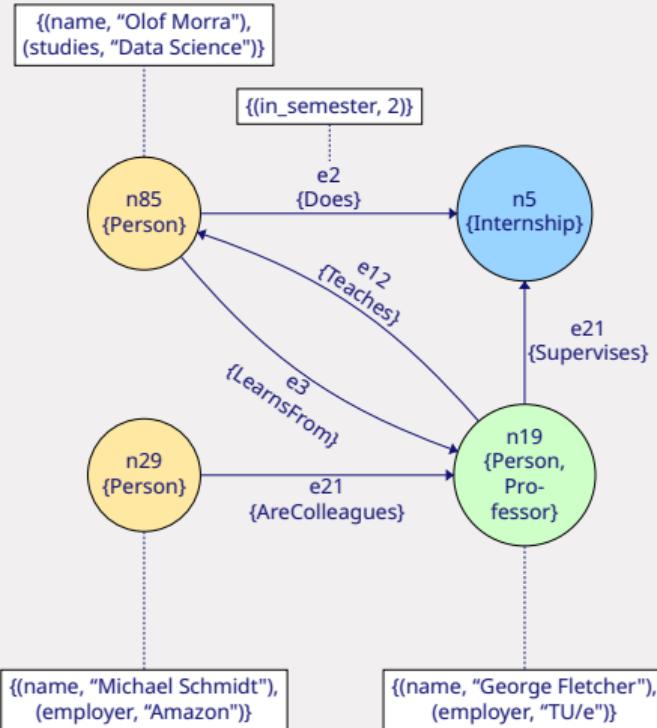
Outputs *n85* for x and *n19* for y

Cypher:

**MATCH** (x)-[\*3]->()

**RETURN** \*

Does not output the above, as *e3* cannot be used twice



# Formal pattern matching

## General

Node  $n$  of graph  $G$  matches node pattern  $\chi = (a, L, P)$  with assignment  $u$ ,  
 $(n, G, u) \models_W \chi$ , 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  $\chi$  be a node pattern,  $p$  a path with starting node  $n_p$ ,  $\pi$  a rigid path pattern and  $\rho = (d, a, L, P, (m, m))$ .

Then,  $(n \cdot p, G, u) \models_W \chi\rho\pi$  for  $m = 0$  if:

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

# Formal pattern matching

## General

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

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

# Formal pattern matching

## General

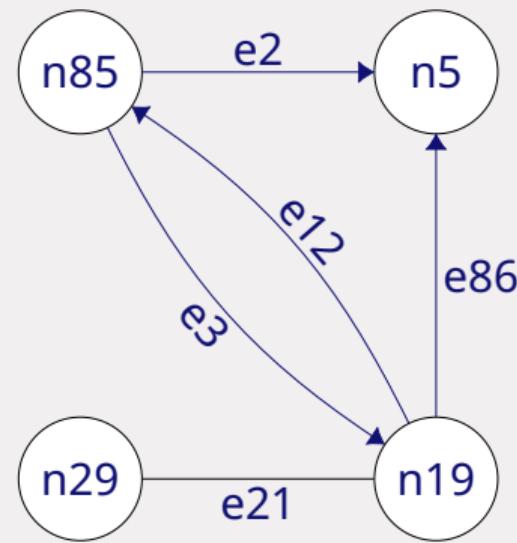
For every edge  $e_i$  with  $i \in \{1, \dots, m\}$  if all the following hold:

- $\varepsilon(e_i) = \begin{cases} 1 & \text{if } d \text{ is } \rightarrow \text{ or } \leftarrow; \\ 0 & \text{if } d \text{ is } - \end{cases}$
- $L = \emptyset$  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
```

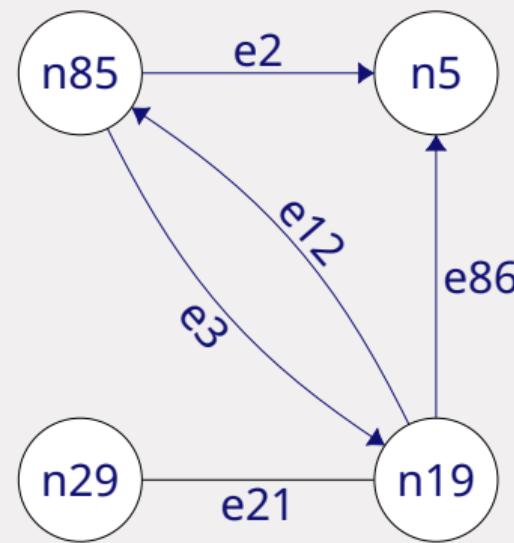


# GQL query example

## Different graph implementations

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

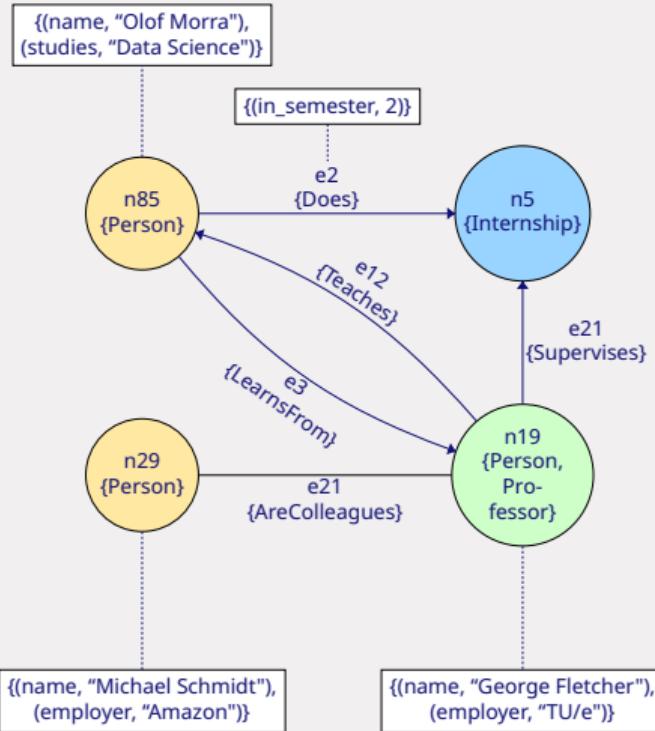
Error if labels are not allowed, no matches found if labels are allowed



# GQL query example

## Different graph implementations

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



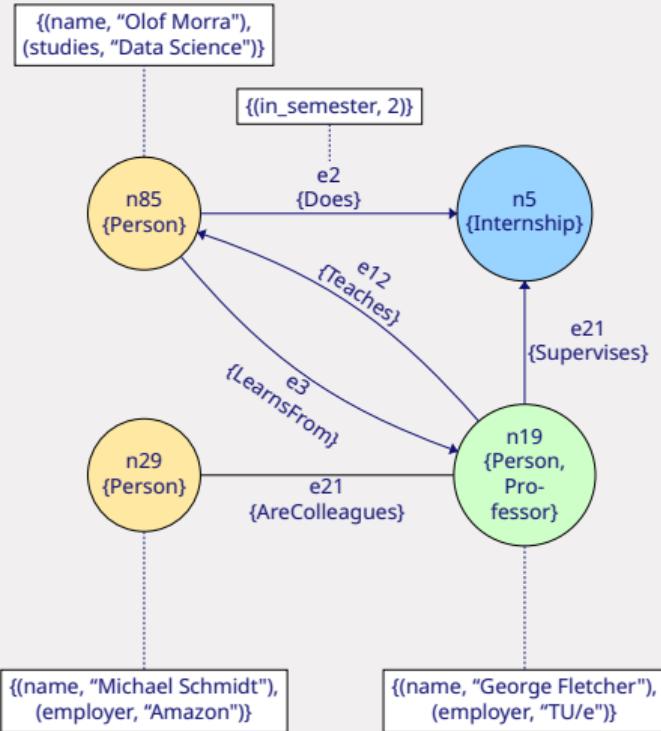
# GQL query example

## 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

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

# Semantics extra

## Query conjunctions

- $\llbracket Q_1 \text{ UNION ALL } Q_2 \rrbracket_{D,G}(T) = \llbracket Q_1 \rrbracket_{D,G}(T) \uplus \llbracket Q_2 \rrbracket_{D,G}(T)$
- $\llbracket Q_1 \text{ UNION DISTINCT } Q_2 \rrbracket_{D,G}(T) = \delta(\llbracket Q_1 \text{ UNION ALL } Q_2 \rrbracket_{D,G}(T))$
- $\llbracket Q_1 \text{ UNION MAX } Q_2 \rrbracket_{D,G}(T) =$   
 $\llbracket Q_1 \text{ UNION ALL } Q_2 \rrbracket_{D,G}(T) - \llbracket Q_1 \text{ INTERSECT ALL } Q_2 \rrbracket_{D,G}(T)$
- Also for intersection, difference, and otherwise