Query languages for property graphs:
RPQs in theory and practice

Victor MARSAULT

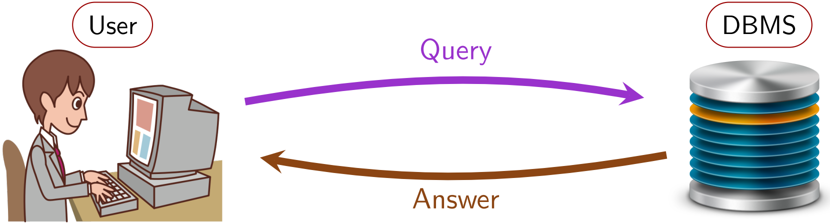
Université Gustave-Eiffel, CNRS, LIGM

BAAM/ADA/MOA Seminar

2022-11-29

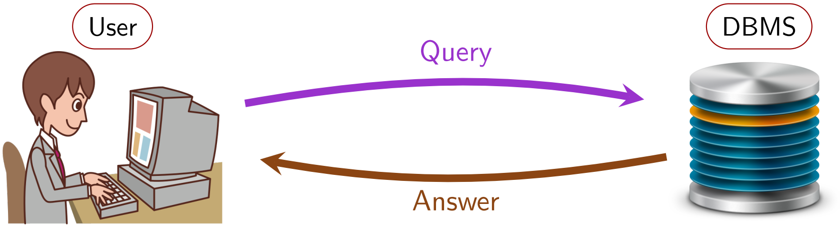
Introduction

General setting



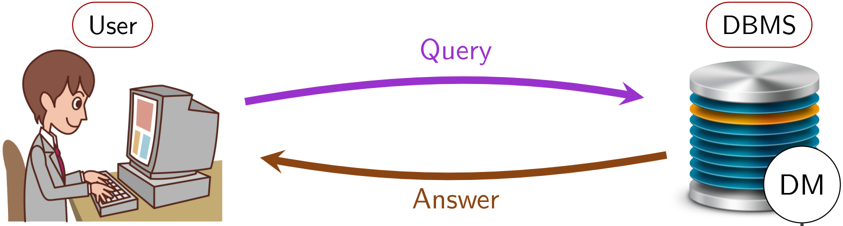
General setting

- DBMS = **D**ata**B**ase **M**anagement **S**ystem



General setting

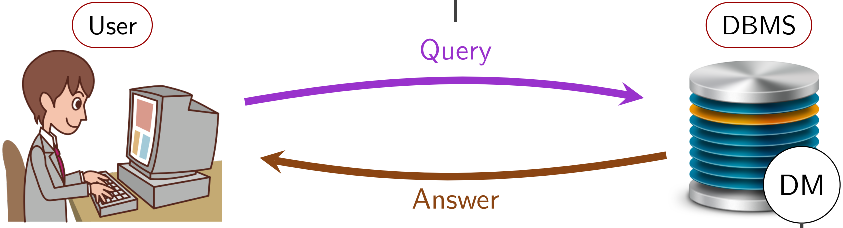
- DBMS = **D**ata**B**ase **M**anagement **S**ystem



- DM = **D**ata **M**odel = "*The way data is structured*"
  - Relational ? XML ? Property graph ? RDF ? etc.

General setting

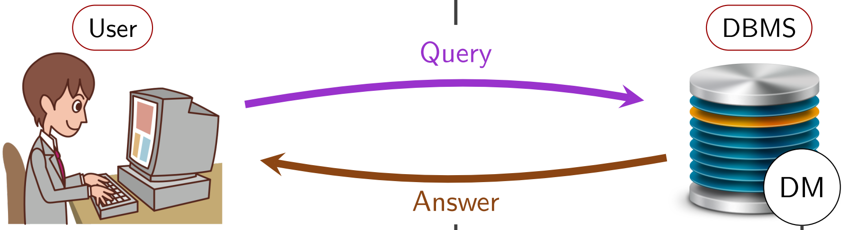
- DBMS = **D**ata**B**ase **M**anagement **S**ystem
- Query language
  - "*What can users ask for?*"



- DM = **D**ata **M**odel = "*The way data is structured*"
  - Relational ? XML ? Property graph ? RDF ? etc.

General setting

- DBMS = **D**ata**B**ase **M**anagement **S**ystem
- Query language
  - "*What can users ask for?*"



- Semantics of query
  - "*What does the query mean?*"
- DM = **D**ata **M**odel = "*The way data is structured*"
  - Relational ? XML ? Property graph ? RDF ? etc.

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Order table

id	buyer	date
0	Alice	01-11-1865
1	Bob	07-07-2022

Order-content table

order_id	product
0	Rabbit
0	Pocket Watch
1	Sponge
1	Broom

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Order table

id	buyer	date
0	Alice	01-11-1865
1	Bob	07-07-2022

Order-content table

order_id	product
0	Rabbit
0	Pocket Watch
1	Sponge
1	Broom

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Order table

id	buyer	date
0	Alice	01-11-1865
1	Bob	07-07-2022

Order-content table

order_id	product
0	Rabbit
0	Pocket Watch
1	Sponge
1	Broom

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Order table

id	buyer	date
0	Alice	01-11-1865
1	Bob	07-07-2022

Order-content table

order_id	product
0	Rabbit
0	Pocket Watch
1	Sponge
1	Broom

Relational DBMS = tables with cross-references

Example: DB for a small store

Client table

name	address
Alice	Wonderland
Bob	124 Conch St.
Charlie	1593 Broadway

Product table

name	price
Sponge	1€
Broom	5€
Rabbit	0€
Pocket Watch	100€

Order table

id	buyer	date
0	Alice	01-11-1865
1	Bob	07-07-2022

Order-content table

order_id	product
0	Rabbit
0	Pocket Watch
1	Sponge
1	Broom

Vast majority of DMBS's are relational, not graph

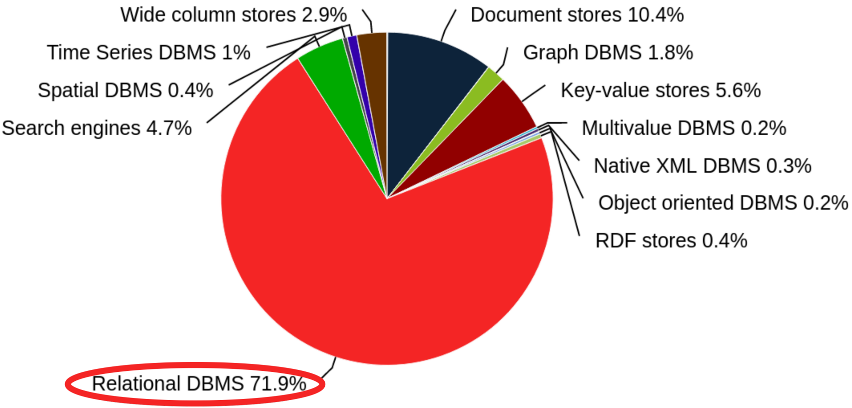


Figure and data from `db-engines.com`, June 2022

Vast majority of DMBS's are relational, not graph

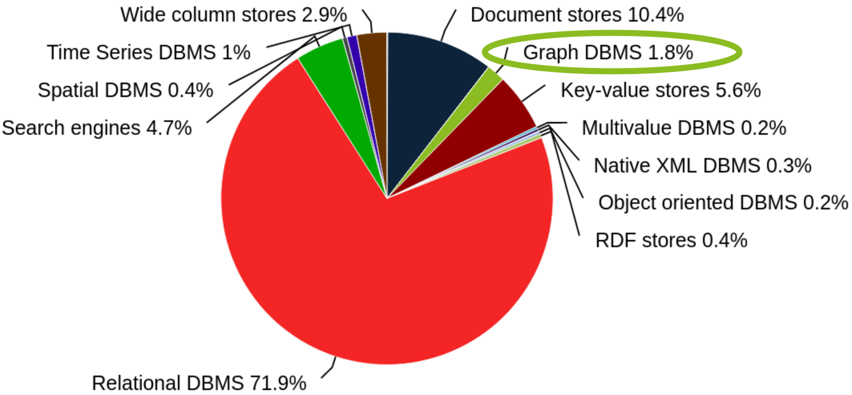


Figure and data from `db-engines.com`, June 2022

Graph DBMS is growing in popularity

+25% per year since 2013

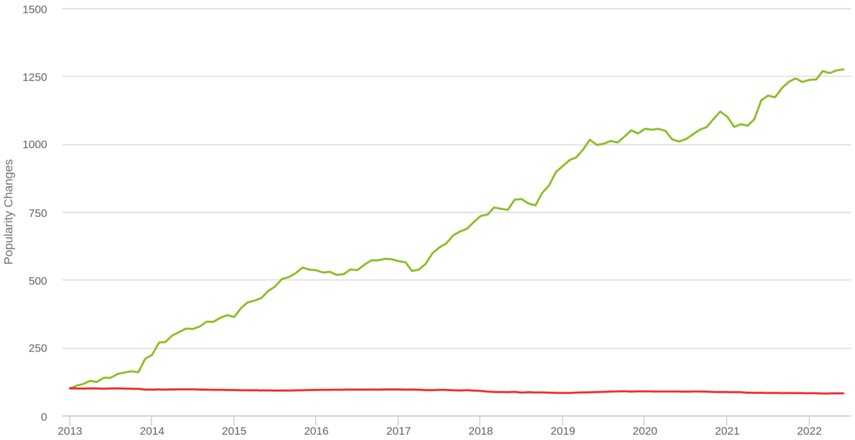
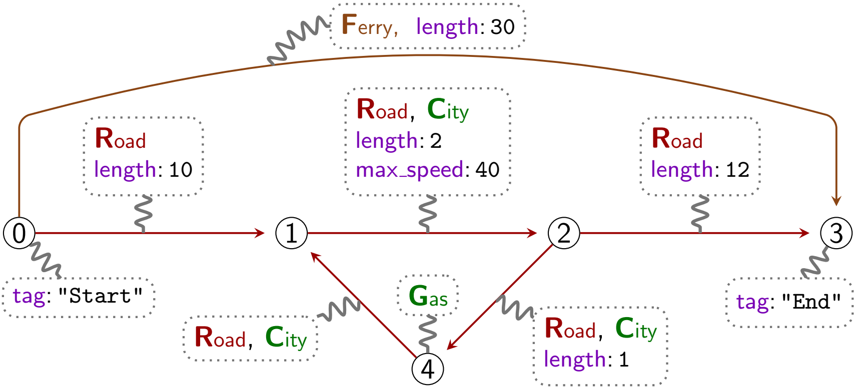


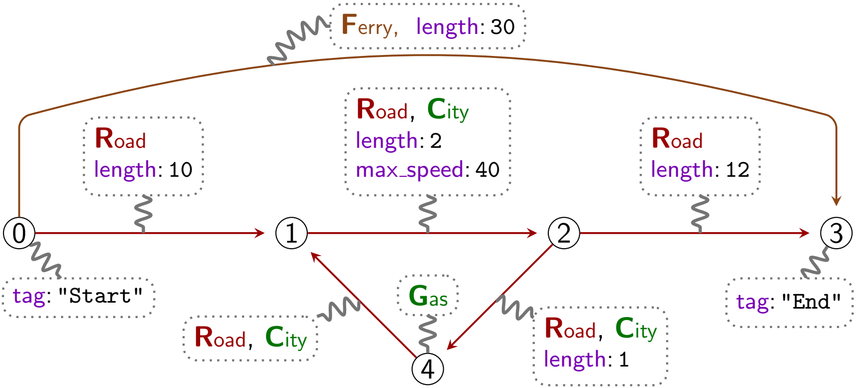
Figure and data from `db-engines.com`, June 2022

Why use graph databases ?

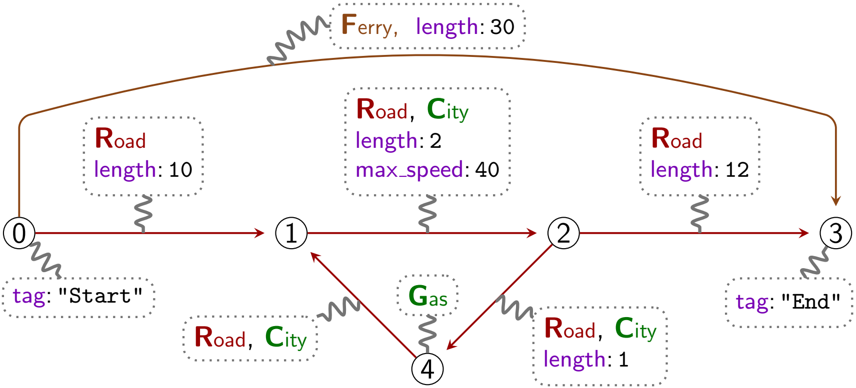
Some data have intrinsically the structure of graphs (e.g. networks)



Why not store graphs in tables?



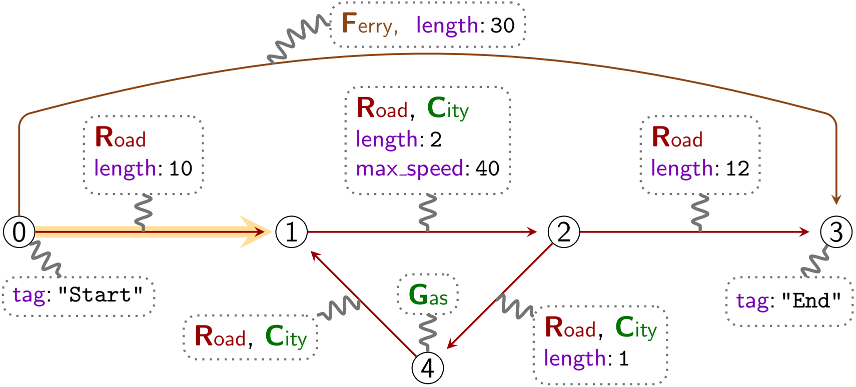
Why not store graphs in tables?



id	source_id	target_id	Road	Ferry	City	length	max_speed
e_{01}	0	1	true	false	false	10	
e_{12}	1	2	true	false	true	10	40
e_{41}	4	1	true	false	true		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Graph — Labels — Properties

Why not store graphs in tables?

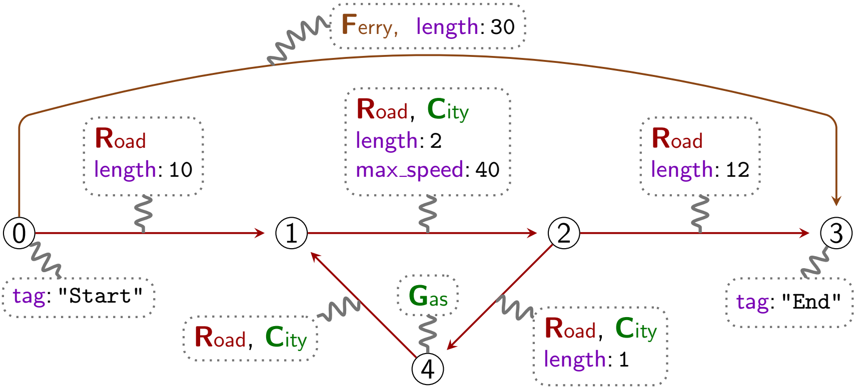


	id	source_id	target_id	Road	Ferry	City	length	max_speed	
▶	e_{01}	0	1	true	false	false	10		◀
	e_{12}	1	2	true	false	true	10	40	
	e_{41}	4	1	true	false	true			
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Graph — Labels — Properties

Why not store graphs in tables?

→ Model restriction allows specific algorithms and intuitive visualisation



id	source_id	target_id	Road	Ferry	City	length	max_speed
e_{01}	0	1	true	false	false	10	
e_{12}	1	2	true	false	true	10	40
e_{41}	4	1	true	false	true		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	Graph			Labels		Properties	

History of query languages for property graphs

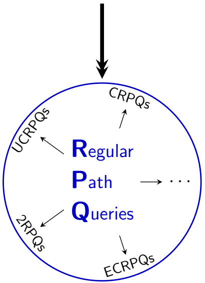
From RPQs to GQL: history and actors

1987 – RPQs are invented [Cruz-Mendelzon-Wood, 1987]

Academia

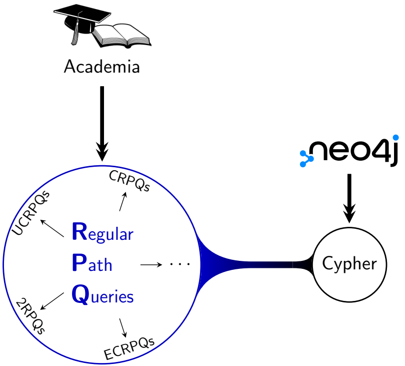
Regular
Path
Queries

Since 1990's – RPQs are extended and studied in academia



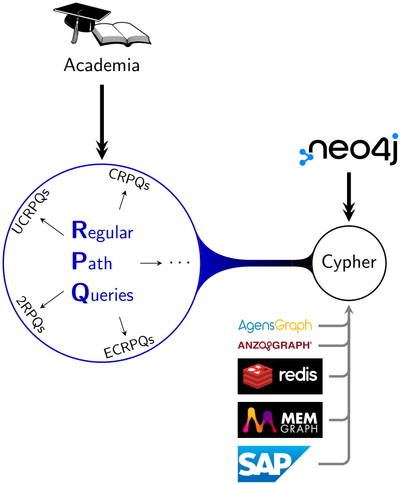
From RPQs to GQL: history and actors

2011 – Cypher is designed by Neo4j



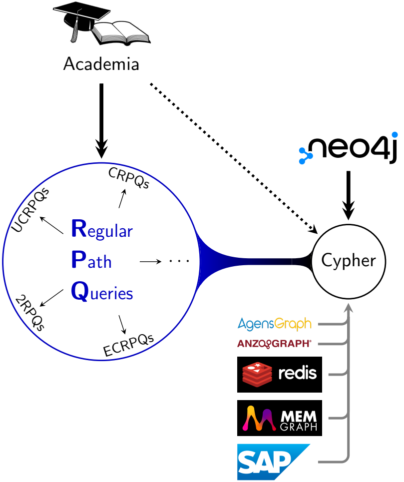
From RPQs to GQL: history and actors

mid 2010's – Cypher is becoming a standard de facto. Standardize Cypher?



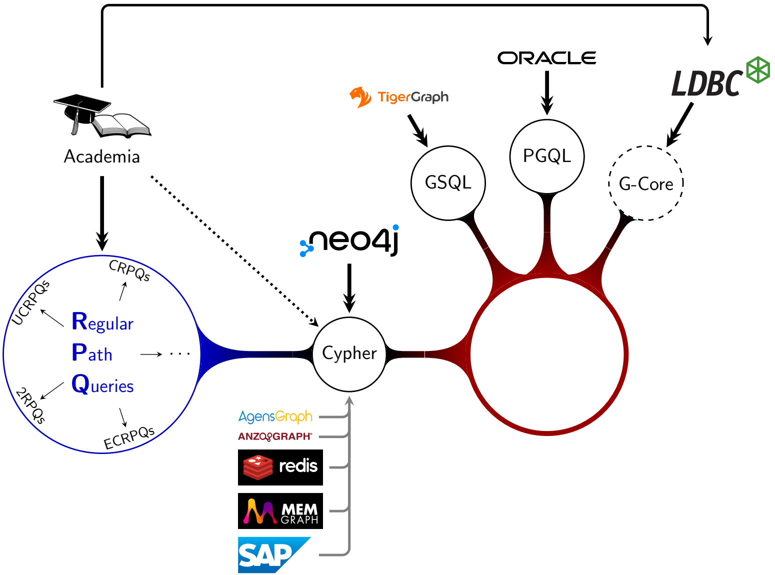
From RPQs to GQL: history and actors

mid 2010's – Cypher is becoming a standard de facto. Standardize Cypher?



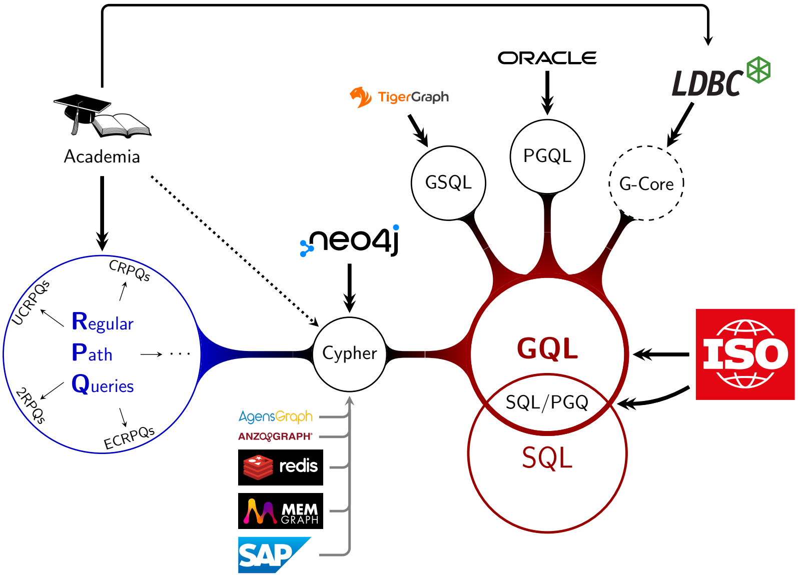
From RPQs to GQL: history and actors

late 2010's – Merge all existing languages instead of standardizing Cypher?



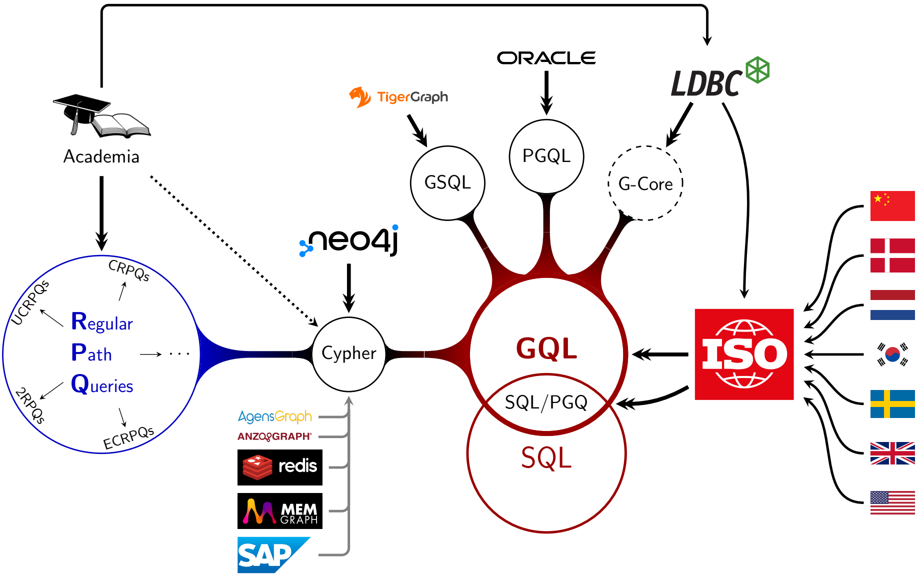
From RPQs to GQL: history and actors

2019-2021 – Two ISO projects: GQL [39075] and SQL/PGQ [9075-16.2]



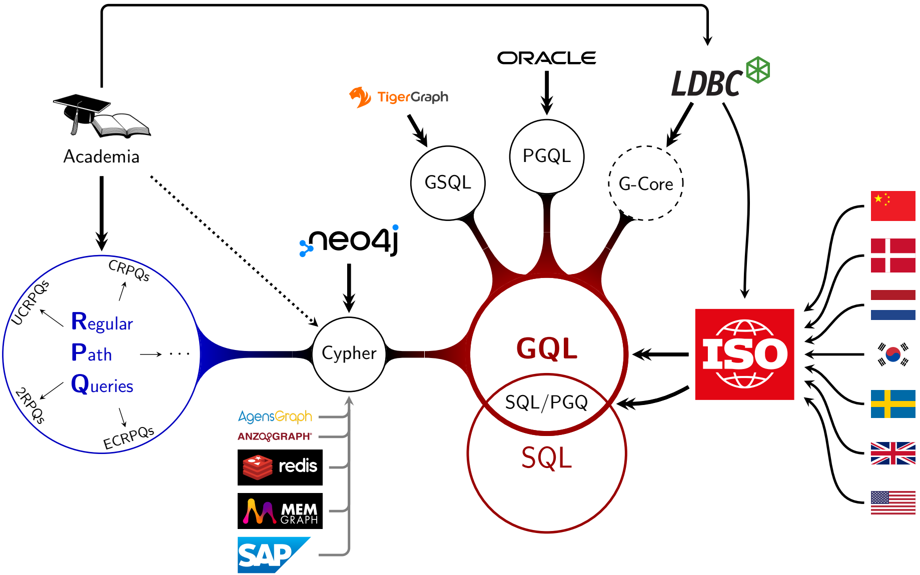
From RPQs to GQL: history and actors

2019-2021 – Two ISO projects: GQL [39075] and SQL/PGQ [9075-16.2]



From RPQs to GQL: history and actors

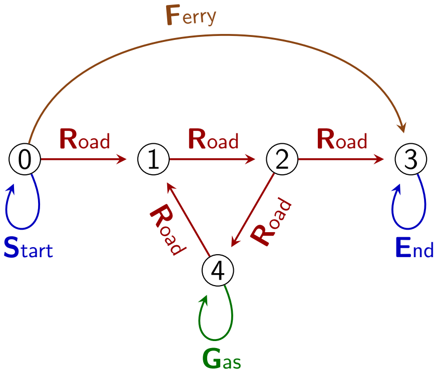
2023 (expected) – Publication of version 1 of GQL



Foundation of querying graph databases: RPQs

RPQs operates on labelled graphs

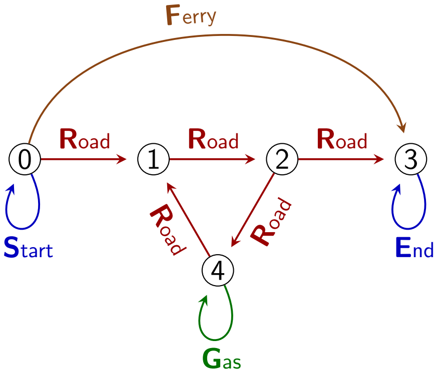
A graph consists of ...



RPQs operates on labelled graphs

A graph consists of ...

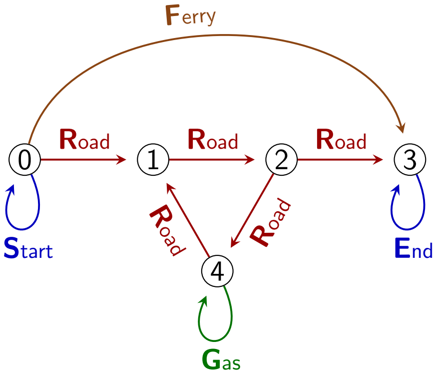
- Vertices (or Nodes)



RPQs operates on labelled graphs

A graph consists of ...

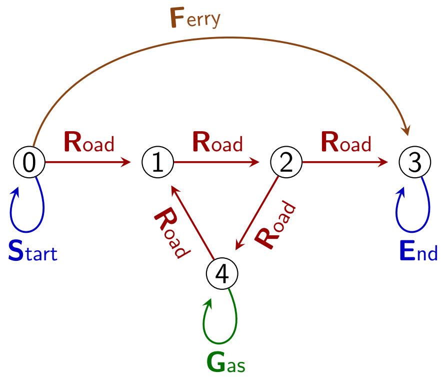
- Vertices (or Nodes)
- Edges (or Relationships)



RPQs operates on labelled graphs

A graph consists of ...

- Vertices (or Nodes)
- Edges (or Relationships)
- Edge labels: {**R**, **F**, **G**, **S**, **E**}



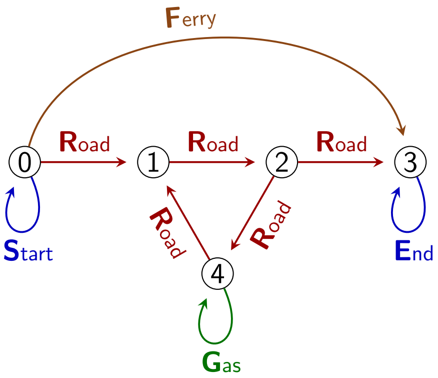
RPQs operates on labelled graphs

A graph consists of ...

- Vertices (or Nodes)
- Edges (or Relationships)
- Edge labels: {**R**, **F**, **G**, **S**, **E**}

Walk

- a.k.a. Path
- Sequence of edges
- Can reuse vertices and edges



RPQs operates on labelled graphs

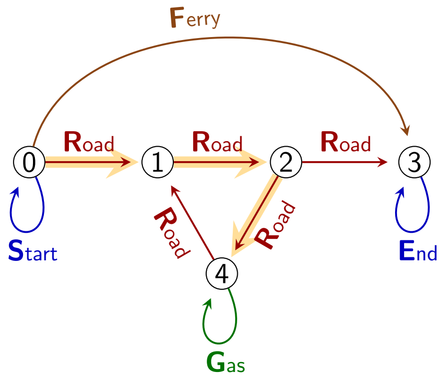
A graph consists of ...

- Vertices (or Nodes)
- Edges (or Relationships)
- Edge labels: {**R**, **F**, **G**, **S**, **E**}

Walk

- a.k.a. Path
- Sequence of edges
- Can reuse vertices and edges

$0 \to 1 \to 2 \to 4$



RPQs operates on labelled graphs

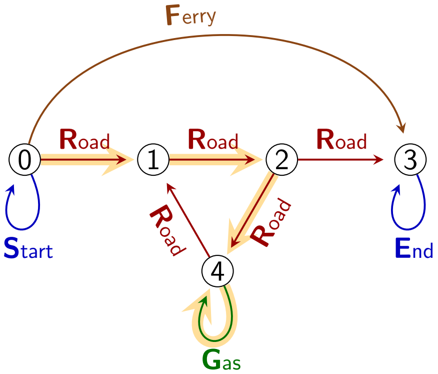
A graph consists of ...

- Vertices (or Nodes)
- Edges (or Relationships)
- Edge labels: {**R**, **F**, **G**, **S**, **E**}

Walk

- a.k.a. Path
- Sequence of edges
- Can reuse vertices and edges

$0 \to 1 \to 2 \to 4 \to 4$



RPQs operates on labelled graphs

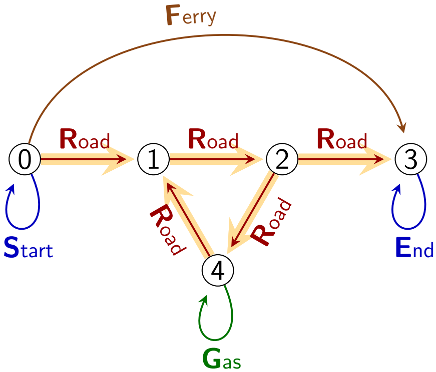
A graph consists of ...

- Vertices (or Nodes)
- Edges (or Relationships)
- Edge labels: {**R**, **F**, **G**, **S**, **E**}

Walk

- a.k.a. Path
- Sequence of edges
- Can reuse vertices and edges

$0 \to 1 \to 2 \to 4 \to 1 \to 2 \to 3$



RPQ = Regular expression

$$\mathcal{Q} ::= \quad \mathbf{A}$$
$$\mathcal{Q}\mathcal{Q}$$
$$\mathcal{Q} + \mathcal{Q}$$
$$\mathcal{Q}^*$$

where **A** is a label in the graph.

An RPQ denotes a set of words

$\mathbf{S}\,(\mathbf{F}+\mathbf{R})^*\mathbf{E}$ denotes the words of the shape **S**`<something>`**E**

Any number of **F** and **R**, in any order

RPQ = Regular expression

$$\mathcal{Q} ::= \quad \mathbf{A}$$
$$\mathcal{Q}\mathcal{Q}$$
$$\mathcal{Q} + \mathcal{Q}$$
$$\mathcal{Q}^*$$

where **A** is a label in the graph.

An RPQ denotes a set of words

$\mathbf{S}\,(\mathbf{F}+\mathbf{R})^*\mathbf{E}$ denotes the words of the shape **S**`<something>`**E**

Any number of **F** and **R**, in any order

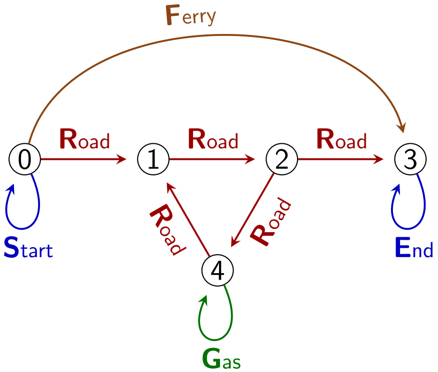
Matches

A match for $\mathcal{Q}$ is any walk w such that $\mathcal{Q}$ denotes the label of w

First example

Query RR matches...

...walks of two **R**oad-edges



First example

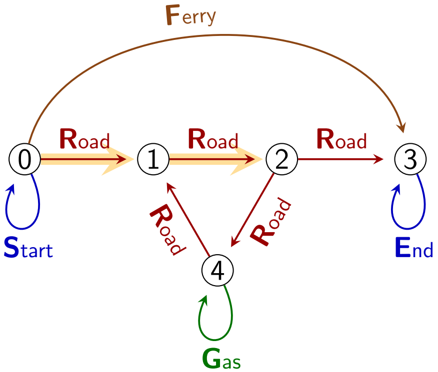
Query RR matches...

...walks of two **R**oad-edges

$0 \to 1 \to 2$ $\quad$ $2 \to 4 \to 1$

$1 \to 2 \to 3$ $\quad$ $4 \to 1 \to 2$

$1 \to 2 \to 3$

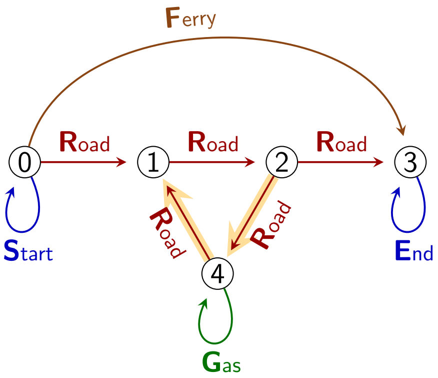


First example

Query RR matches...

...walks of two **R**oad-edges

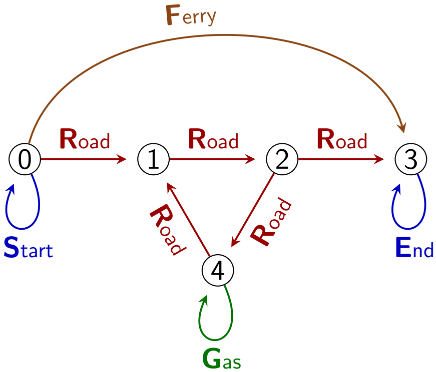
$0 \to 1 \to 2$	$2 \to 4 \to 1$
$1 \to 2 \to 3$	$4 \to 1 \to 2$
$1 \to 2 \to 3$	



The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

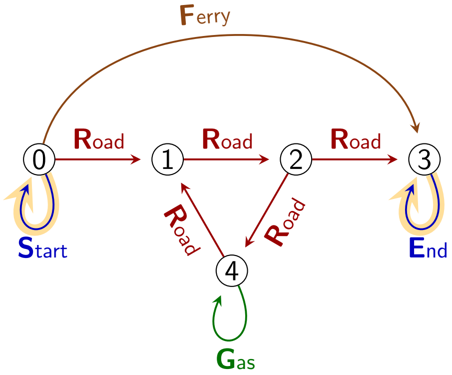
$\mathcal{Q}_1$ matches...



The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

$\mathcal{Q}_1 = \mathbf{S}(\mathbf{R}+\mathbf{F})^*\mathbf{E}$

$\mathcal{Q}_1$ matches...

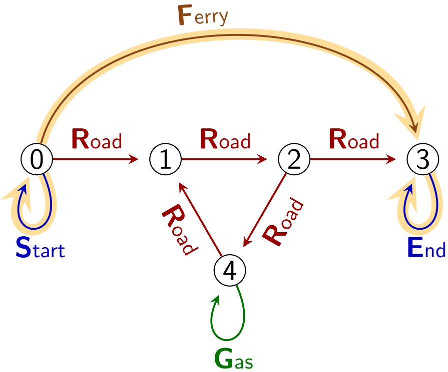


The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_1$ matches...

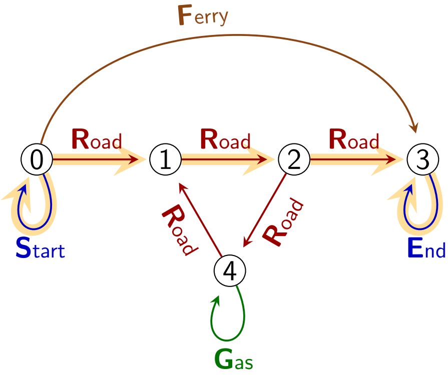
- The ferry



$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_1$ matches...

- The ferry
- The direct road

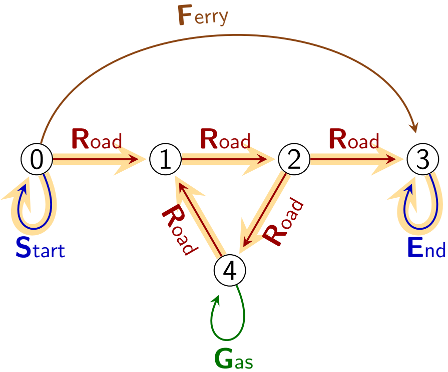


The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_1$ matches...

- The ferry
- The direct road
- Roads with laps in the circuit



The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

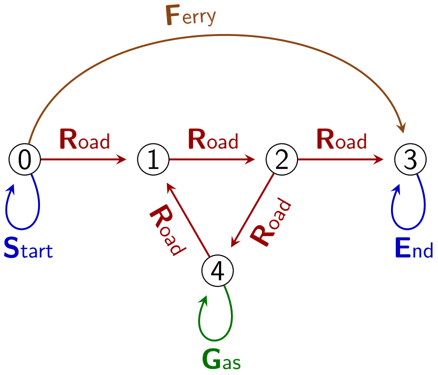
$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_1$ matches...

- The ferry
- The direct road
- Roads with laps in the circuit

$\mathcal{Q}_2 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{G}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_2$ matches...



The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

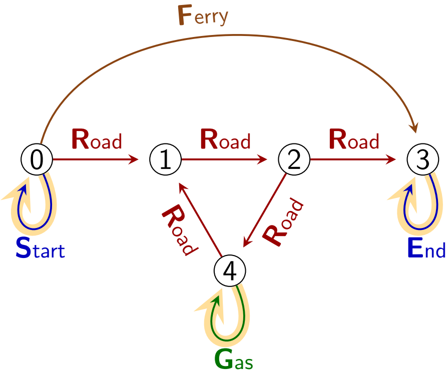
$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_1$ matches...

- The ferry
- The direct road
- Roads with laps in the circuit

$\mathcal{Q}_2 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{G}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_2$ matches...



The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$

$\mathcal{Q}_1 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

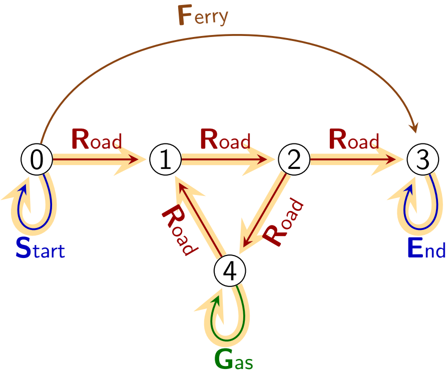
$\mathcal{Q}_1$ matches...

- The ferry
- The direct road
- Roads with laps in the circuit

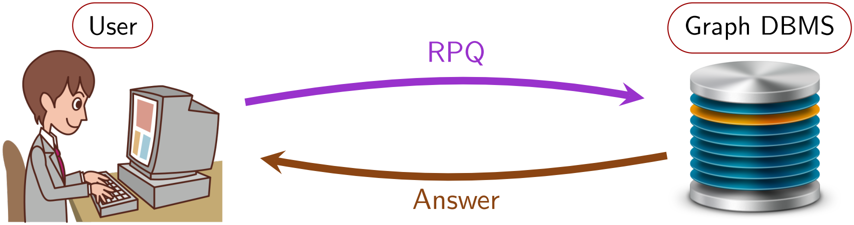
$\mathcal{Q}_2 = \mathbf{S}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{G}\,(\mathbf{R}+\mathbf{F})^*\,\mathbf{E}$

$\mathcal{Q}_2$ matches...

- Roads with laps in the circuit



Fundamental challenge with RPQs



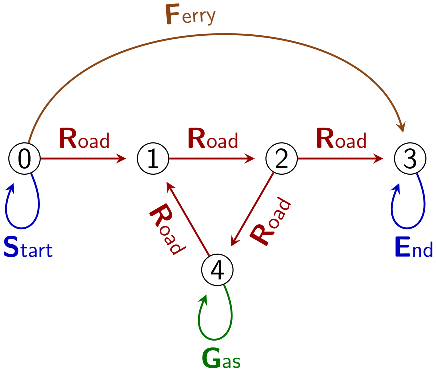
⚠ Infinitely many matches but finite answer ⚠

Homomorphism semantics

Main theoretical semantics [Angles et al. 2017] (used in SparQL)

Definition

- Returns the endpoints of matches



Homomorphism semantics

Main theoretical semantics [Angles et al. 2017] (used in SparQL)

Definition

- Returns the endpoints of matches

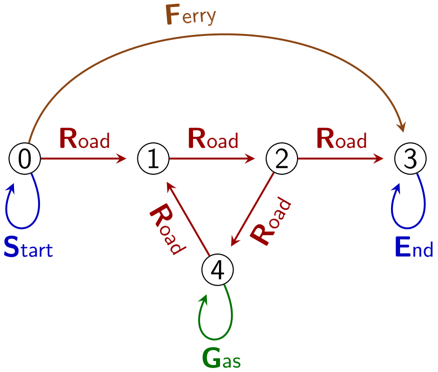
$\mathcal{Q}_1 =$ **S** (**R**+**F**)* **E**

$\mathcal{Q}_2 =$ **S** (**R**+**F**)* **G** (**R**+**F**)* **E**

- All matches are of the form:

$0 \rightarrow \cdots \rightarrow 3$

$\Rightarrow \mathcal{Q}_1$ and $\mathcal{Q}_2$ return $\{(0,3)\}$



Homomorphism semantics (2)

Pros and cons

Pros

- Efficient algorithms
- Well grounded theory

Homomorphism semantics (2)

Pros and cons

Pros

- Efficient algorithms
- Well grounded theory

Cons

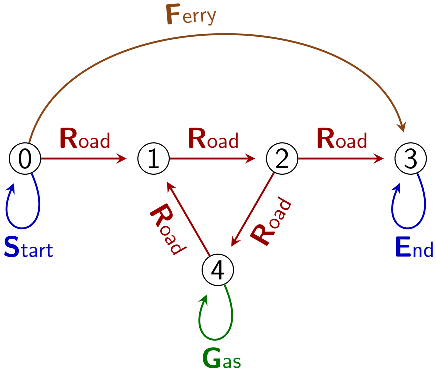
- Very limited information in the answer
  - User: *"I want to go from Paris to Lyon by car"*
  - Database: *"Yes you can"*

Shortest-walk semantics

Used in PGQL (Oracle), in GSQL (TigerGraph), in G-core [Angles et al. 2018]

Definition

- Return the walk with the least number of edges



Shortest-walk semantics

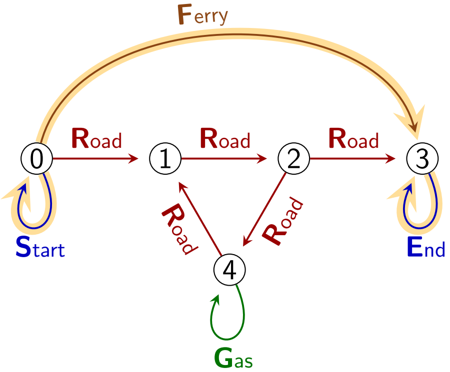
Used in PGQL (Oracle), in GSQL (TigerGraph), in G-core [Angles et al. 2018]

Definition

- Return the walk with the least number of edges

$\mathcal{Q}_1 =$ **S** (**R**+**F**)* **E**

- $\mathcal{Q}_1$ returns 1 walk
  - the ferry
- Walks taking the road have more edges



Shortest-walk semantics

Used in PGQL (Oracle), in GSQL (TigerGraph), in G-core [Angles et al. 2018]

Definition

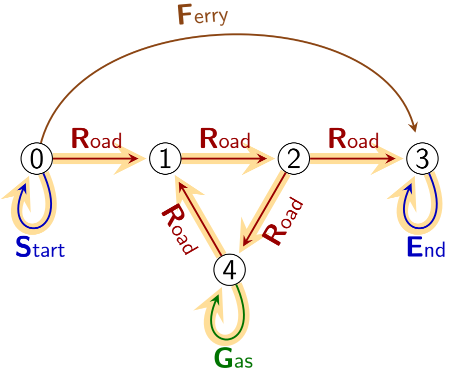
- Return the walk with the least number of edges

$\mathcal{Q}_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_1$ returns 1 walk
  - the ferry
- Walks taking the road have more edges

$\mathcal{Q}_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_1$ returns 1 walk
  - the one-lap road



Shortest walk semantics (2)

Pros and cons

Pros

- Returns walks
- Efficient algorithms
- Horizontal post-processing
  - Horizontal = along the walk
  - *"Is there a gas station on the way?"*
  - *"What is the length of the walk?"*

Shortest walk semantics (2)

Pros

- Returns walks
- Efficient algorithms
- Horizontal post-processing
  - Horizontal = along the walk
  - *"Is there a gas station on the way?"*
  - *"What is the length of the walk?"*

Cons

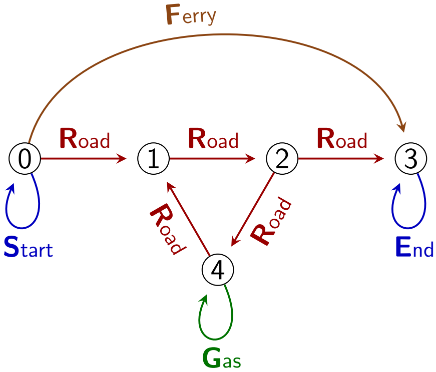
- No vertical post-processing
  - Vertical = accross the walks with the same endpoints
  - *"What is the shortest walk in time?"*
  - *"What is the connectedness level?"*
- No coverage of the space of matches

Trail semantics

Used in Cypher (Neo4j) [Francis et al. 2018] [Green et al. 2019]

Definition

- Return walks
- Forbid to repeat edges



Trail semantics

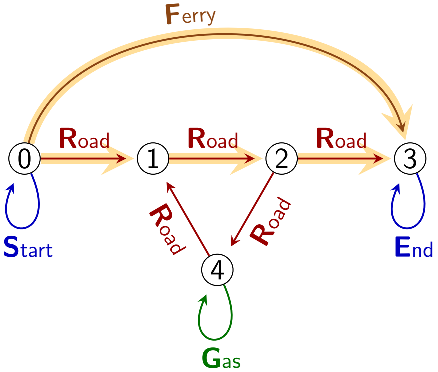
Used in Cypher (Neo4j) [Francis et al. 2018] [Green et al. 2019]

Definition

- Return walks
- Forbid to repeat edges

$\mathcal{Q}_1 = \; \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_1$ returns 2 walks
  - the ferry
  - the straight road



Trail semantics

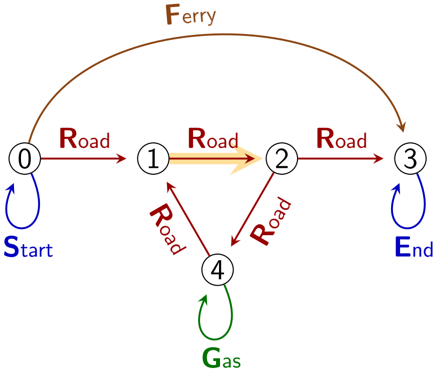
Used in Cypher (Neo4j) [Francis et al. 2018] [Green et al. 2019]

Definition

- Return walks
- Forbid to repeat edges

$\mathcal{Q}_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_1$ returns 2 walks
  - the ferry
  - the straight road
- Walks with circuit laps repeat the middle edge

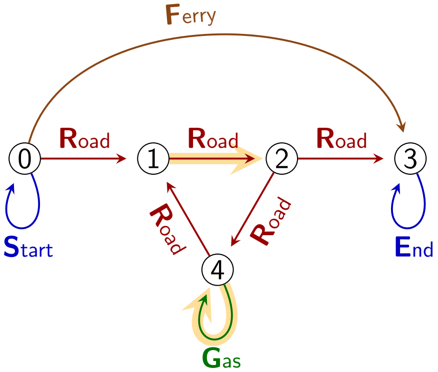


Trail semantics

Used in Cypher (Neo4j) [Francis et al. 2018] [Green et al. 2019]

Definition

- Return walks
- Forbid to repeat edges



$\mathcal{Q}_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_1$ returns 2 walks
  - the ferry
  - the straight road
- Walks with circuit laps repeat the middle edge

$\mathcal{Q}_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- $\mathcal{Q}_2$ returns nothing

Trail semantics (2)

Pros and cons

Pros

- Returns walks
- Counting matches is possible
- Horizontal and vertical post-processing
- Some coverage of the space of matches

Trail semantics (2)

Pros and cons

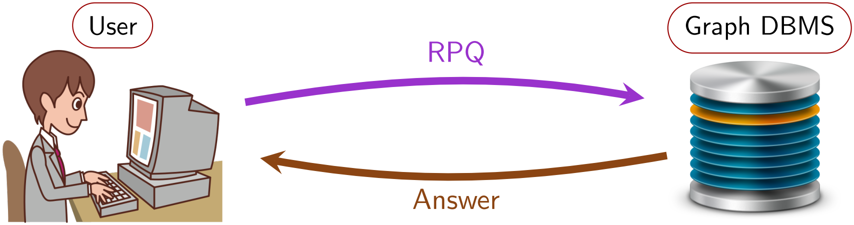
Pros

- Returns walks
- Counting matches is possible
- Horizontal and vertical post-processing
- Some coverage of the space of matches

Cons

- All problems are computationally hard [Martens et al. 2020]
  - Counting, enumeration, existence
  - Checking whether Q_2 returns anything $\rightarrow$ Already NP-hard
- Part of the space of matches might be uncovered

Fundamental challenge with RPQs



⚠ Infinitely many matches but finite answer ⚠

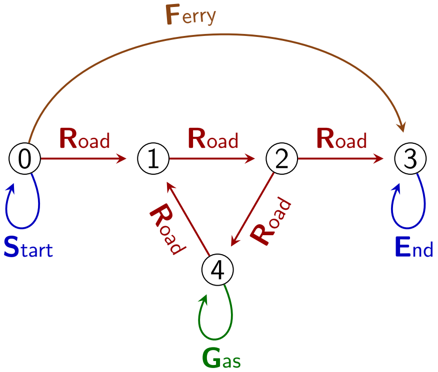
- Several way to ensure finiteness
  - Homomorphism $\rightarrow$ Filters out most information
  - Shortest-walk $\rightarrow$ Bad coverage of the space of matches
  - Trail $\rightarrow$ Computationally hard
  - Other variants have similar issues.
- No solution is clearly superior

Run-based semantics

New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once



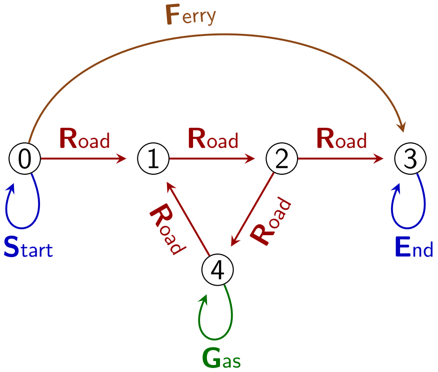
Run-based semantics

New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$$\mathcal{Q}_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$



Run-based semantics

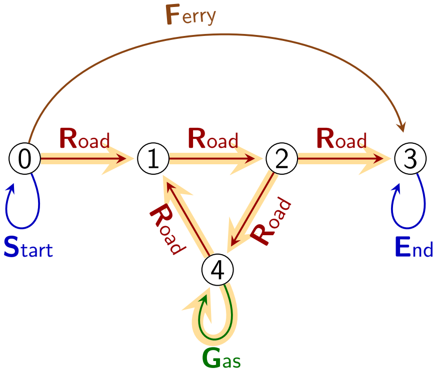
New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$\mathcal{Q}_2 =$ **S** (**R**+**F**)* **G** (**R**+**F**)* **E**

- Returns the 1-lap road only



Run-based semantics

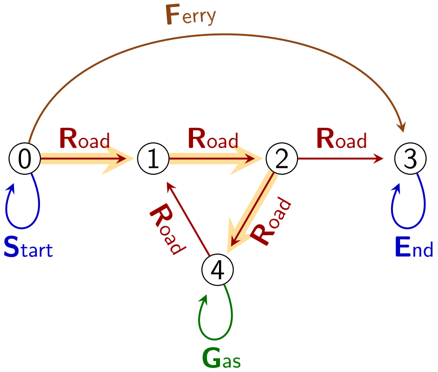
New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$\mathcal{Q}_2 = \quad \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns the 1-lap road only
  - Before **G** $\rightarrow$ use the left **R**



Run-based semantics

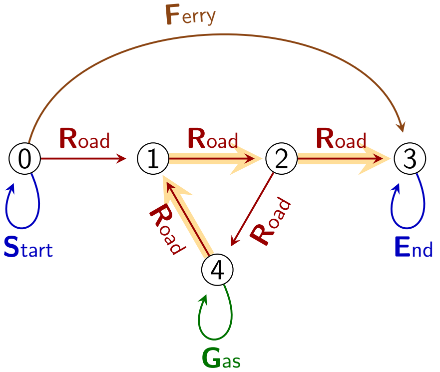
New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$\mathcal{Q}_2 =$ **S** (**R**+**F**)* **G** (**R**+**F**)* **E**

- Returns the 1-lap road only
  - Before **G** → use the left **R**
  - After **G** → use the right **R**



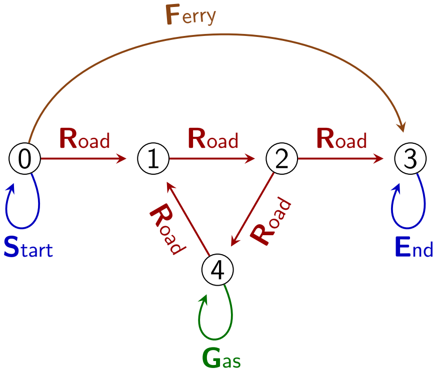
New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$\mathcal{Q}_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns the 1-lap road only
  - Before **G** → use the left **R**
  - After **G** → use the right **R**
- >1 circuit lap ⇒ some edge use the same atom twice



Run-based semantics

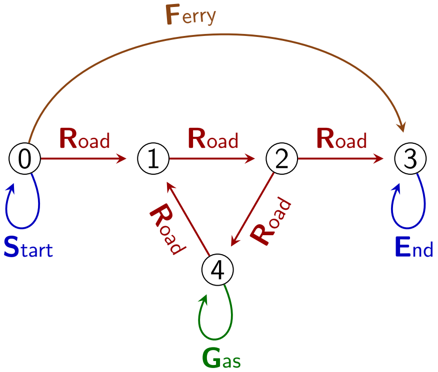
New theoretical compromise [David-Francis-Marsault 202?]

Definition

- Returns walks
- Each edge may match each atom only once

$\mathcal{Q}_2 =$ **S** (**R**+**F**)* **G** (**R**+**F**)* **E**

- Returns the 1-lap road only
  - Before **G** → use the left **R**
  - After **G** → use the right **R**
- > 1 circuit lap ⇒ some edge use the same atom twice



$\mathcal{Q}_1 =$ **S** (**R**+**F**)* **E**

- Returns the ferry and the straight road

Run-based semantics (2)

Pros and cons

Pros

- Returns walks
- Horizontal and vertical post-processing
- "Reasonable" coverage of the space of matches
- Counting results is possible
- Emptyness and Enumeration are efficient

Run-based semantics (2)

Pros and cons

Pros

- Returns walks
- Horizontal and vertical post-processing
- "Reasonable" coverage of the space of matches
- Counting results is possible
- Emptyness and Enumeration are efficient

Cons

- Counting results is computationally hard
- Answer depends on the way the query is written
  - $\mathbf{R}^*$ allows no lap in the circuit
  - $(\mathbf{R} + \mathbf{R})^*$ allows 1 lap in the circuit

Run-based semantics (2)

Pros and cons

Pros

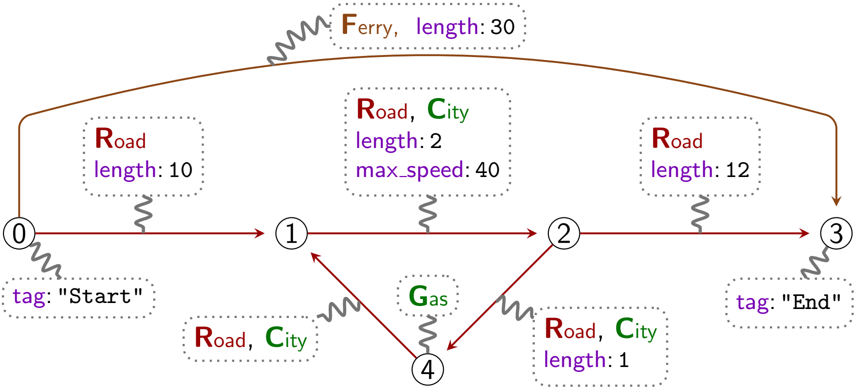
- Returns walks
- Horizontal and vertical post-processing
- "Reasonable" coverage of the space of matches
- Counting results is possible
- Emptyness and Enumeration are efficient
- Gives some expressivity to the user

Cons

- Counting results is computationally hard
- Answer depends on the way the query is written
  - $\mathbf{R}^*$ allows no lap in the circuit
  - $(\mathbf{R} + \mathbf{R})^*$ allows 1 lap in the circuit

Property graphs and real query languages

Back to our example property graph



Vertices and edges may bear:

- zero or more labels
- zero or more properties

- Property = key-value pair
- Key = string
- Value = `bool`, `int`, `str`, ...

Cypher features

Cypher features

- Trail semantics

Cypher features

- Trail semantics
- Restricted RPQs (in fact UC2RPQs) with the following restrictions:
  - Under a Kleene star, only unions of atoms are allowed:

- Trail semantics
- Restricted RPQs (in fact UC2RPQs) with the following restrictions:
  - Under a Kleene star, only unions of atoms are allowed:
- Backward atoms are allowed e.g. $\mathbf{F}\text{erry}\left(\mathbf{R}\text{oad}^{-1}\right)^*$

Cypher features

- Trail semantics
- Restricted RPQs (in fact UC2RPQs) with the following restrictions:
  - Under a Kleene star, only unions of atoms are allowed:
- Backward atoms are allowed e.g. $\mathbf{F}\text{erry}\left(\mathbf{R}\text{oad}^{-1}\right)^{*}$
- ASCII-art syntax

Cypher features

- Trail semantics
- Restricted RPQs (in fact UC2RPQs) with the following restrictions:
  - Under a Kleene star, only unions of atoms are allowed:
- Backward atoms are allowed e.g. $\mathbf{F}\text{erry}\left(\mathbf{R}\text{oad}^{-1}\right)^*$
- ASCII-art syntax
- Cypher is graph-to-tables
- Chaining of clauses

ASCII-art syntax

- Vertices: `MATCH (:Gas)`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`
- Disjunction: `MATCH ()-[:Road|Ferry]->()`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`
- Disjunction: `MATCH ()-[:Road|Ferry]->()`
- Kleene star: `MATCH ()-[:Road*]->()`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`
- Disjunction: `MATCH ()-[:Road|Ferry]->()`
- Kleene star: `MATCH ()-[:Road*]->()`

- Variables: `MATCH ()-[:Road]->(x)-[:Road]->()`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`
- Disjunction: `MATCH ()-[:Road|Ferry]->()`
- Kleene star: `MATCH ()-[:Road*]->()`

- Variables: `MATCH ()-[:Road]->(x)-[:Road]->()`
- Implicit join: `MATCH (x)-[:Road*]->(x)`

ASCII-art syntax

- Vertices: `MATCH (:Gas)` `MATCH tag:"Start"`
- Edges: `MATCH -[:Road]->`

- Concatenation: `MATCH ()-[:Road]->(:Gas)-[:Road]->()`
- Disjunction: `MATCH ()-[:Road|Ferry]->()`
- Kleene star: `MATCH ()-[:Road*]->()`

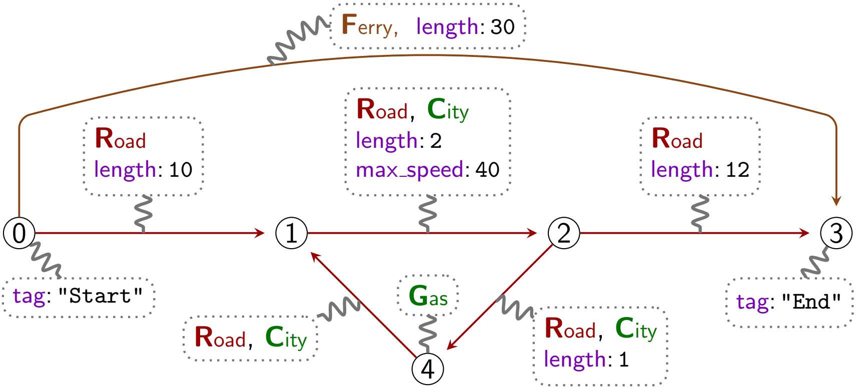
- Variables: `MATCH ()-[:Road]->(x)-[:Road]->()`
- Implicit join: `MATCH (x)-[:Road*]->(x)`

Cypher queries for $\mathcal{Q}_1$ and $\mathcal{Q}_2$

```
MATCH ({tag:"Start"})-[:Road|Ferry*]->({tag:"End"})
```

```
MATCH ({tag:"Start"})-[:Road|Ferry*]->
                    (:Gas)-[:Road|Ferry*]->({tag:"End"})
```

Cypher returns a table...



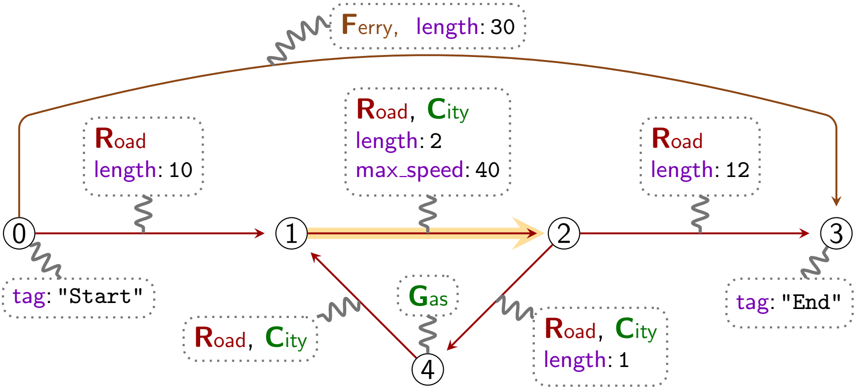
Query

```
MATCH (s)-[:City]->(t)
```

Result

s	t
1	2
2	4
4	1

Cypher returns a table...



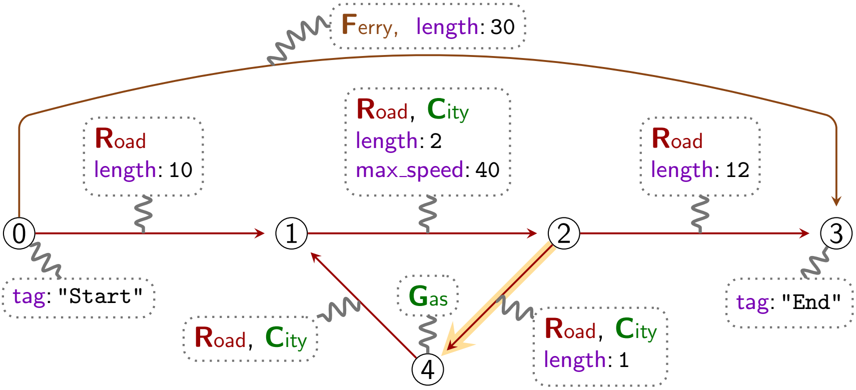
Query

```
MATCH (s)-[:City]->(t)
```

Result

s	t
1	2
2	4
4	1

Cypher returns a table...



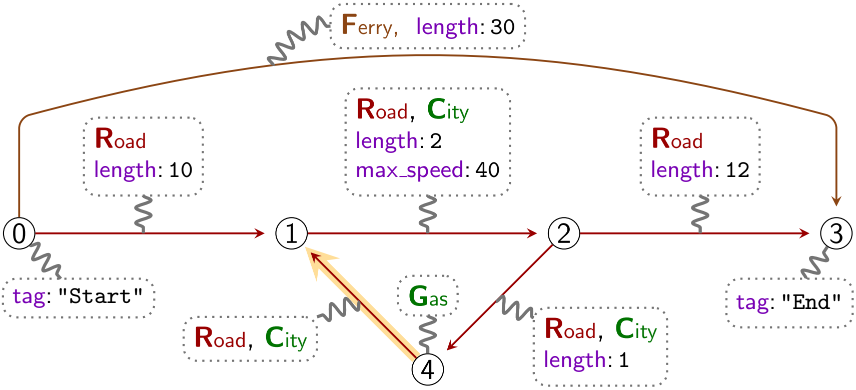
Query

```
MATCH (s)-[:City]->(t)
```

Result

s	t
1	2
2	4
4	1

Cypher returns a table...

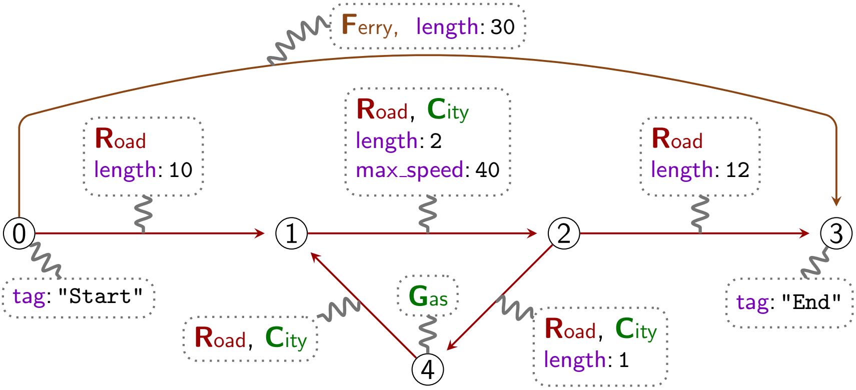


Query

```
MATCH (s)-[:City]->(t)
```

Result

s	t
1	2
2	4
4	1



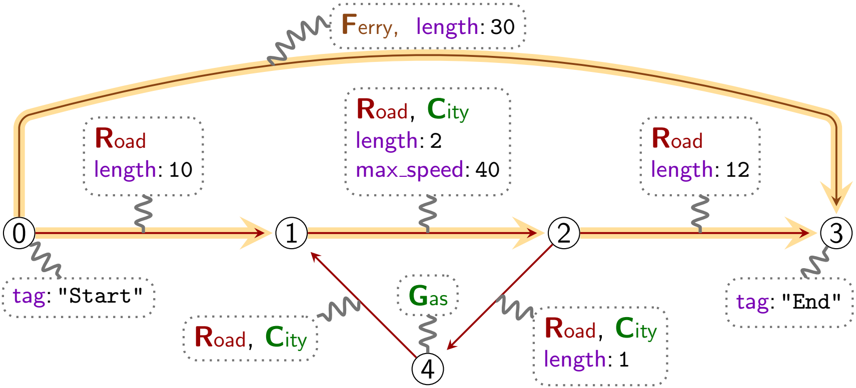
Query

```
MATCH
    (s {tag:"Start"})
        -[:Road|Ferry*]->
            (t {tag:"End"})
```

Result

s	t
0	3
0	3

Cypher returns a table... but computes walks



Query

```
MATCH
    (s {tag:"Start"})
        -[:Road|Ferry*]->
            (t {tag:"End"})
```

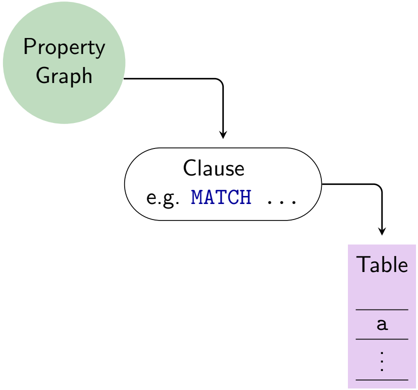
Result

s	t	
0	3	← The ferry
0	3	← The direct road

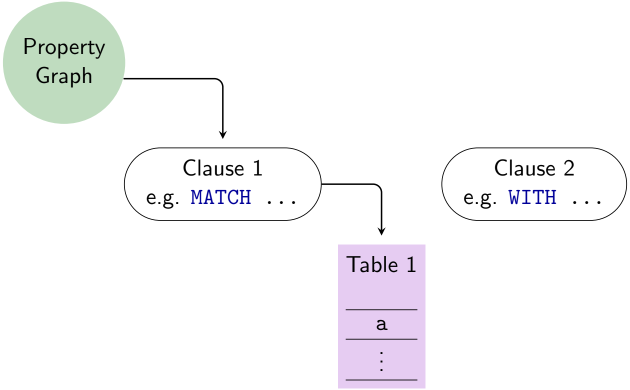
Other Cypher features

- `ORDER BY`: orders row
- `WHERE`: filters row
- `WITH` or `RETURN`:
  - adds/renames columns
  - horizontal aggregation (e.g. with keyword `reduce`)
  - vertical aggregation (e.g. with keyword `count`, `max`)
- `CREATE`/`DELETE`/`SET`: updates the property graph

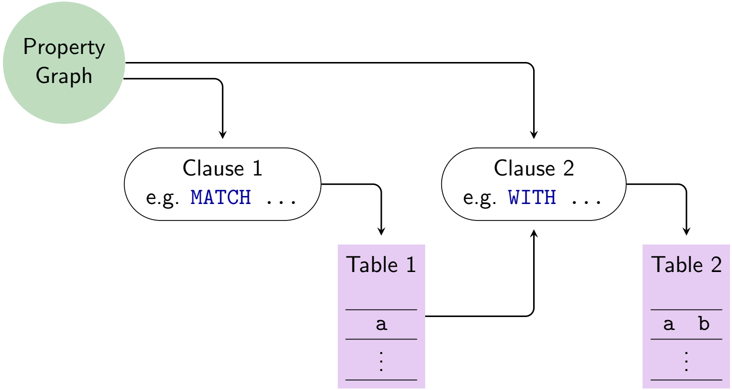
A Cypher query actually chain clauses



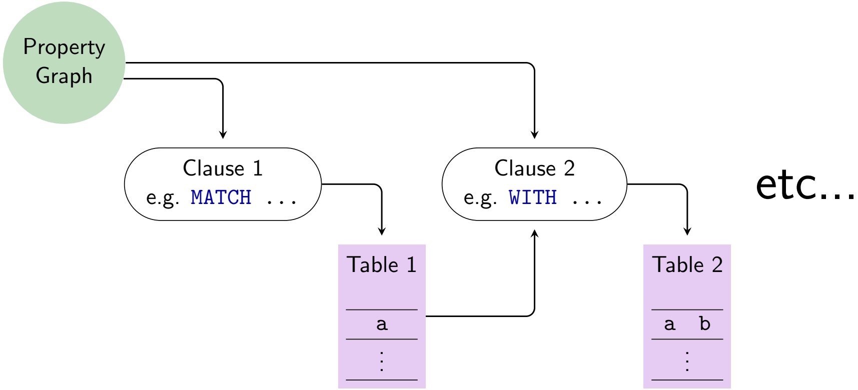
A Cypher query actually chain clauses



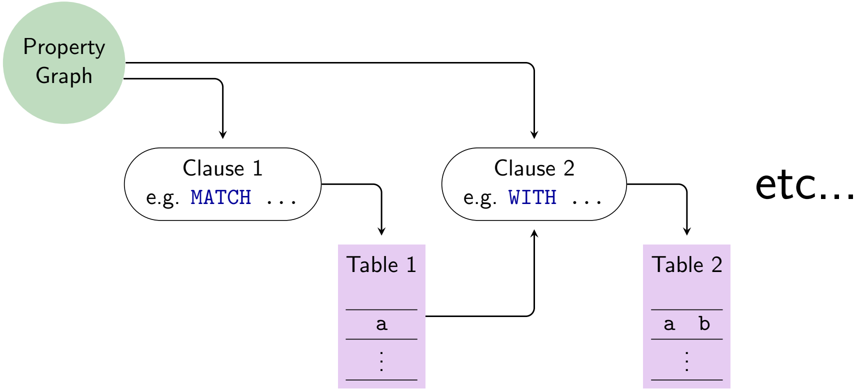
A Cypher query actually chain clauses



A Cypher query actually chain clauses



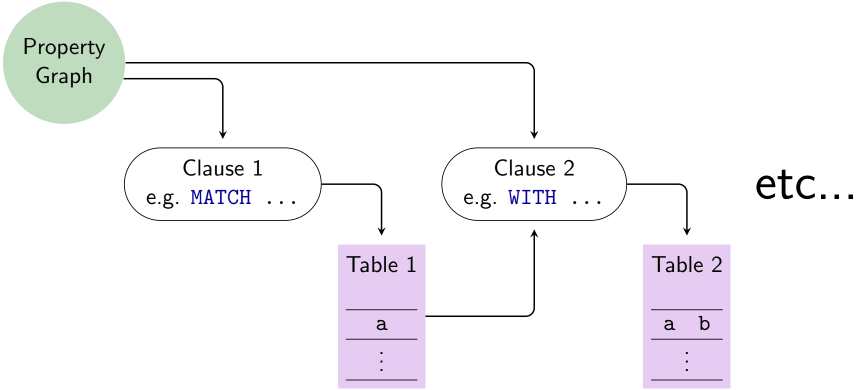
A Cypher query actually chain clauses



Example

- Clause 1 makes some pattern matching
- Clause 2 aggregates over the result of Clause 1

A Cypher query actually chain clauses



Example

- Clause 1 makes some pattern matching
- Clause 2 aggregates over the result of Clause 1

$\Rightarrow$ Trail semantics (rich post-processing at the cost of efficiency)

GQL, standard query language for property graphs

Features inherited from Cypher

- ASCII-art syntax
- Graph-to-tables
- Chaining of clauses
- No nested Kleene stars

GQL, standard query language for property graphs

Features inherited from Cypher

- ASCII-art syntax
- Graph-to-tables
- Chaining of clauses
- No nested Kleene stars

New features

- Arbitrary union under star
- Undirected edges
- Query multiple database at the same time
- Subqueries

GQL, standard query language for property graphs

Features inherited from Cypher

- ASCII-art syntax
- Graph-to-tables
- Chaining of clauses
- No nested Kleene stars

New features

- Arbitrary union under star
- Undirected edges
- Query multiple database at the same time
- Subqueries
- Deduplication based on "binding path"

GQL and the fundamental challenge with RPQs

An RPQ may have infinitely many matches

- GQL has to ensure finiteness of answer
- No solution is clearly superior

GQL and the fundamental challenge with RPQs

An RPQ may have infinitely many matches

- GQL has to ensure finiteness of answer
- No solution is clearly superior

GQL does not choose

- Trail semantics → keyword `TRAIL`
- Shortest-walk semantics → keyword `SHORTEST`
- Syntax restriction → keyword `WALK`
- Others

GQL and the fundamental challenge with RPQs

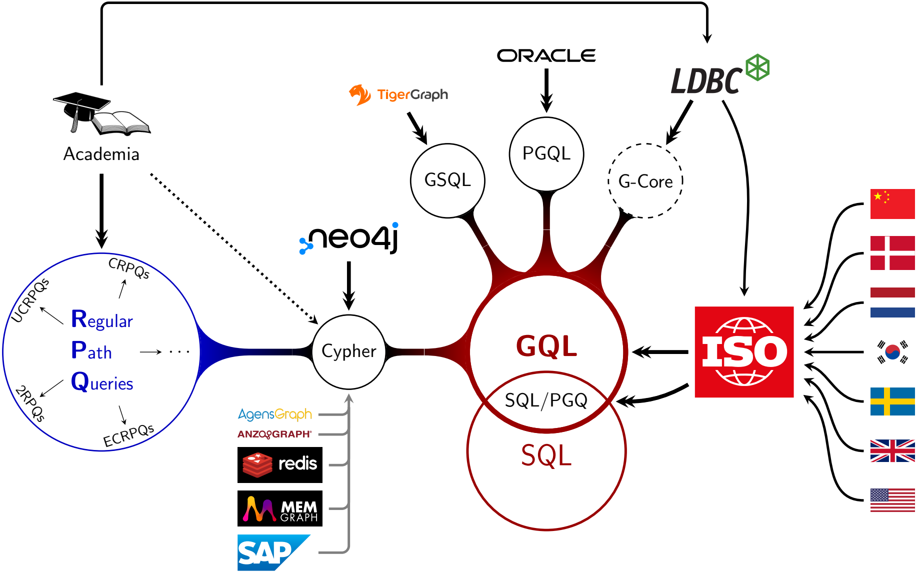
An RPQ may have infinitely many matches

- GQL has to ensure finiteness of answer
- No solution is clearly superior

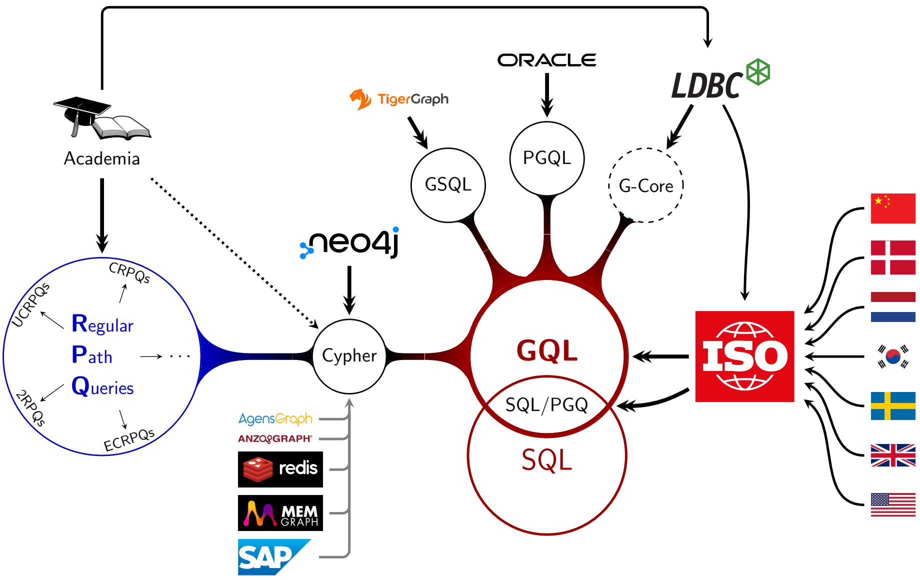
GQL does not choose

- Trail semantics → keyword `TRAIL`
- Shortest-walk semantics → keyword `SHORTEST`
- Syntax restriction → keyword `WALK`
- Others

→ Could we add run-based semantics in GQL 2.0?



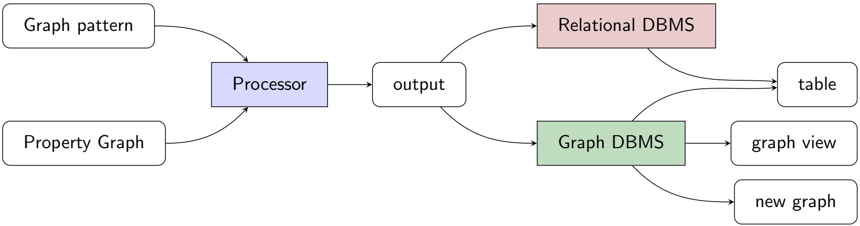
Academia
CRPQs
UCRPQs
Regular
Path
Queries
2RPQs
ECRPQs
TigerGraph
ORACLE
LDBC
GSQL
PGQL
G-Core
neo4j
Cypher
GQL
SQL/PGQ
SQL
ISO
AgensGraph
ANZOGRAPH
redis
MEM GRAPH
SAP



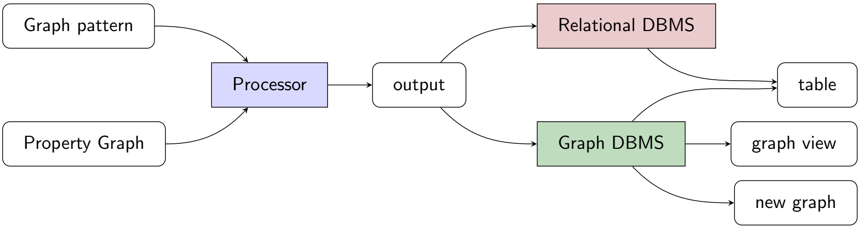
Thank you for your attention!

Navigable outline

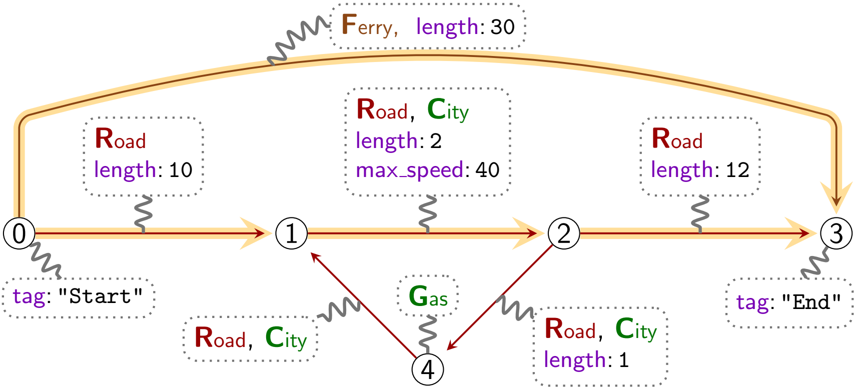
GQL will be usable from SQL



GQL will be usable from SQL

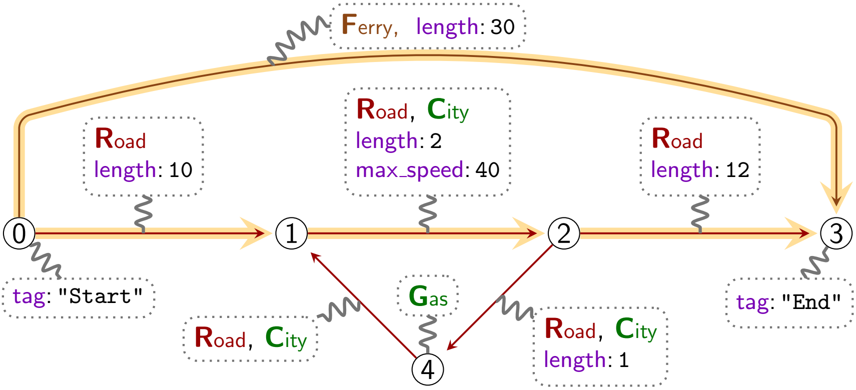


Output of GQL: set of path bindings
Path binding = walk annotated with variables




```
MATCH TRAIL (a WHERE a.tag="Start")
          [ -[r:Road]-> | -[c:City]-> ]* (b WHERE b.tag="End")
```

GQL path-bindings in one slide




```
MATCH TRAIL (a WHERE a.tag="Start")
         [ -[r:Road]-> | -[c:City]-> ]* (b WHERE b.tag="End")
```

0	→	1	→	2	→	3		
a	r		r		r	b		

0	→	1	→	2	→	3		
a	r		c		r	b		