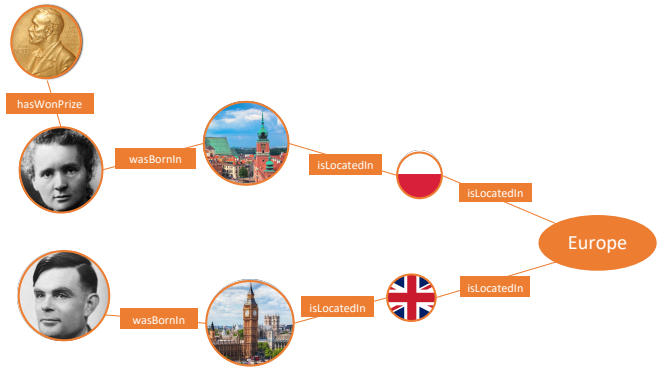
Graph Querying

Pierre Genevès
CNRS

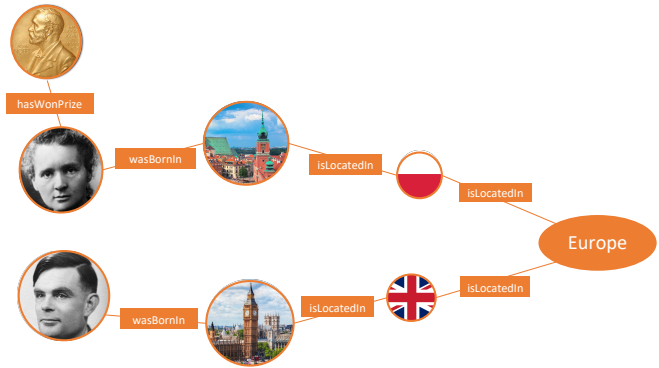
Declarative Graph Querying



Reknown scientists born in Europe:

`?x, ?y` ⟵ `?x hasWonPrize ?y, ?x wasBornIn/isLocatedIn`$^{+}$ `Europe`

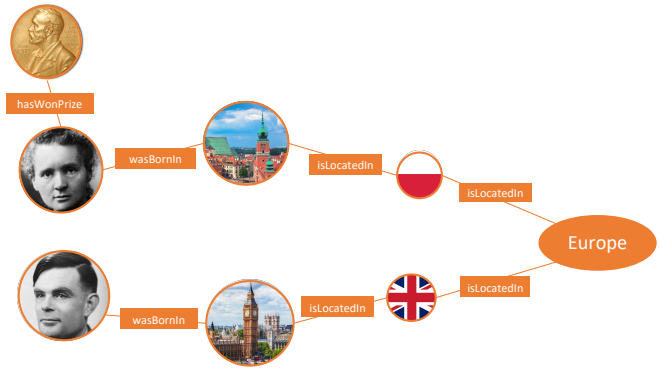
Declarative Graph Querying



Reknown scientists born in Europe:

$$\underbrace{\texttt{?x, ?y}}_{\text{head}} \longleftarrow \underbrace{\texttt{?x hasWonPrize ?y, ?x wasBornIn/isLocatedIn}^{+}\ \texttt{Europe}}_{\text{body}}$$

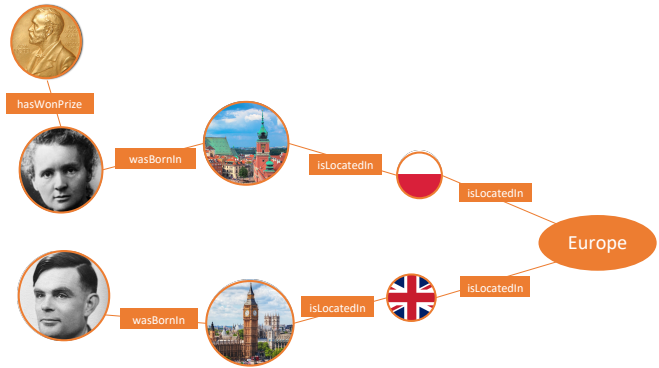
Declarative Graph Querying



Reknown scientists born in Europe:

$$\texttt{?x, ?y} \longleftarrow \underbrace{\texttt{?x hasWonPrize ?y}}_{\text{conjunct}}, \underbrace{\texttt{?x wasBornIn/isLocatedIn}^{+}\ \texttt{Europe}}_{\text{conjunct}}$$

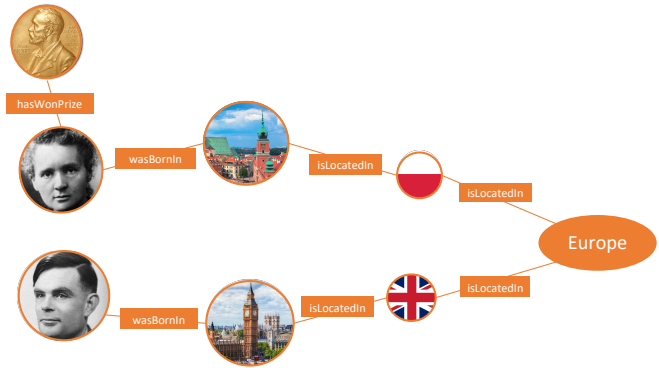
Declarative Graph Querying



Reknown scientists born in Europe:

$$\texttt{?x, ?y} \longleftarrow \texttt{?x}\ \underbrace{\texttt{hasWonPrize}}_{\text{simple relation}}\ \texttt{?y, ?x}\ \underbrace{\texttt{wasBornIn/isLocatedIn}^{+}}_{\text{path relation}}\ \texttt{Europe}$$

Declarative Graph Querying



Reknown scientists born in Europe:

`?x, ?y` ⟵ `?x` **`hasWonPrize`** `?y, ?x` **`wasBornIn/isLocatedIn`**$^{+}$ `Europe`

simple relation

path relation with transitive closure (recursion)

Recursion

- Essential mechanism to enable deep navigation in a graph
- Major feature for extracting valuable information from a linked data structure
- Navigational paths (regular path queries)
- PageRank, shortest paths, connected components, etc.

More Examples

All ancestors of Alice:

$$\texttt{?y} \longleftarrow \texttt{Alice (father|mother)}^{+}\ \texttt{?y}$$

Pairs of stops connected by tram lines A and B:

$$\texttt{?x, ?y} \longleftarrow \texttt{?x nextStopA}^{+}\ \texttt{?y, ?x nextStopB}^{+}\ \texttt{?y}$$

People with a certain skill that Bob knows, either directly or indirectly:

$$\texttt{?x} \longleftarrow \texttt{Bob knows}^{+}\ \texttt{?x:Person, ?x hasSkill skill1}$$

Analysis of drug interactions, etc.

Recursive graph query evaluation today

- In practice: on large knowledge graphs (e.g. Yago, wikidata), certain recursive queries can be difficult to evaluate... or even hardly feasible

Recursive graph query evaluation today

- In practice: on large knowledge graphs (e.g. Yago, wikidata), certain recursive queries can be difficult to evaluate... or even hardly feasible

- This depends on graph instance (size, topology) and **query shape**

Recursive graph query evaluation today

- In practice: on large knowledge graphs (e.g. Yago, wikidata), certain recursive queries can be difficult to evaluate... or even hardly feasible

- This depends on graph instance (size, topology) and **query shape**

Recursive subqueries may result in intermediate results whose size is bigger than the overall graph size

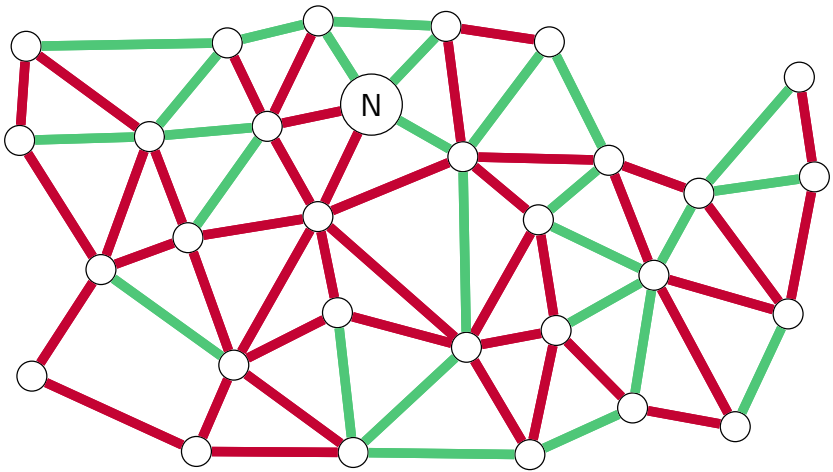
An example

N Red/Green* ?t

This query retrieves nodes reachable from node N through a given (recursive) path along edges

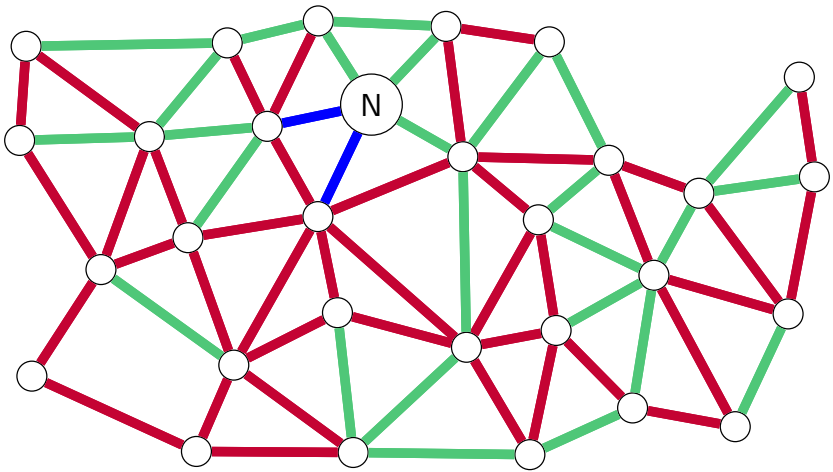
An example

N Red/Green* ?t



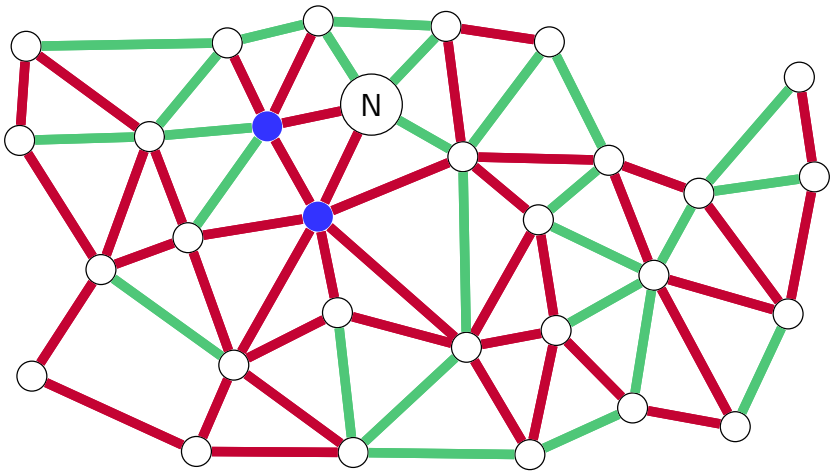
An example

N Red/Green* ?t



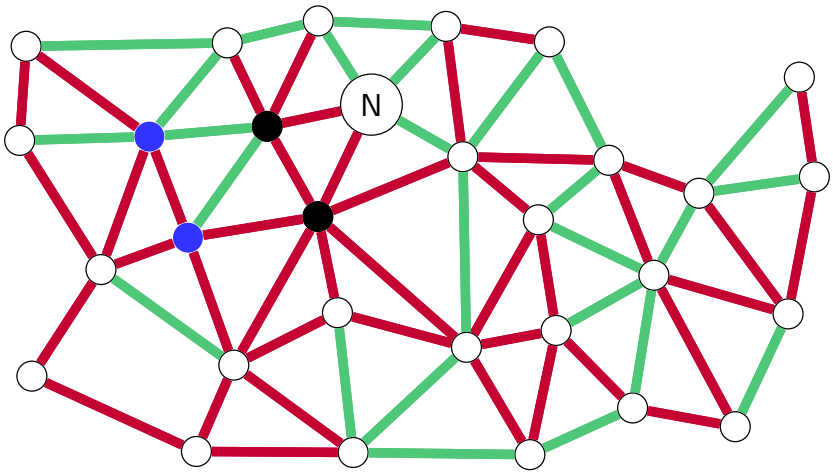
An example

N Red/Green* ?t



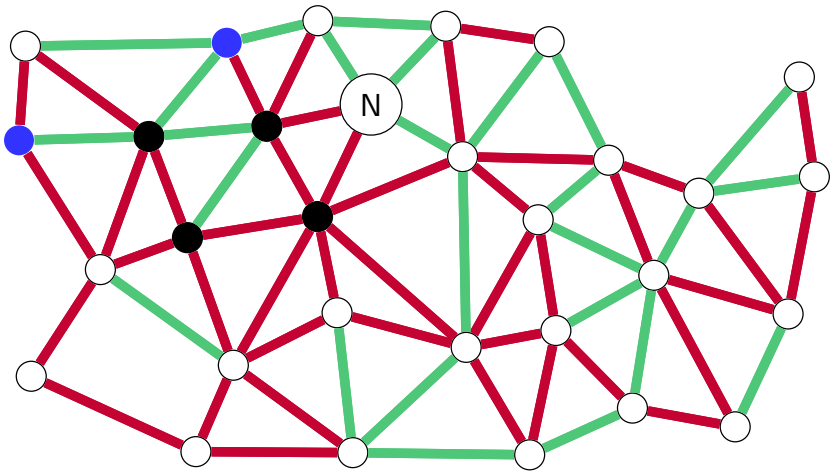
An example

N Red/Green* ?t



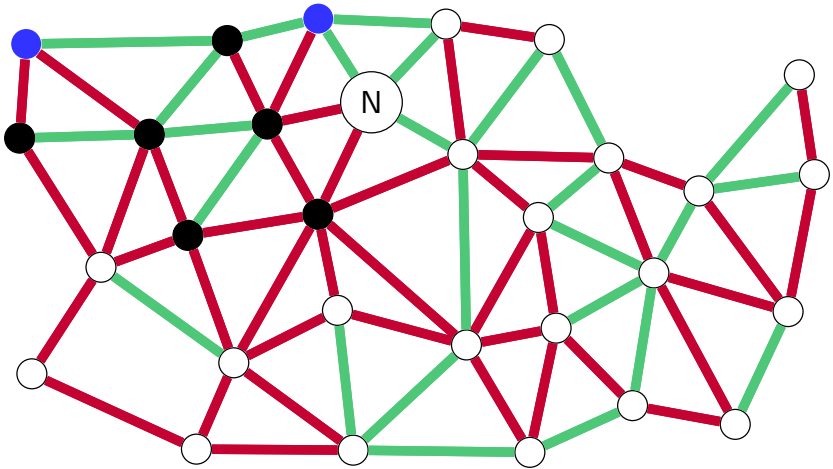
An example

N Red/Green* ?t



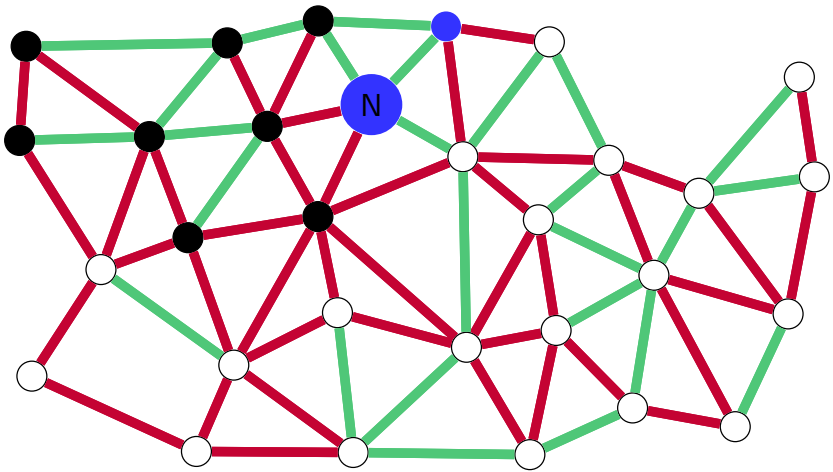
An example

N Red/Green* ?t



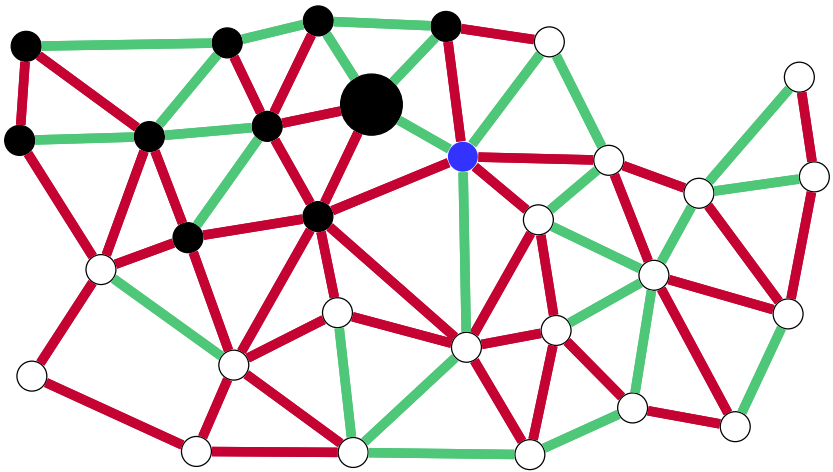
An example

N Red/Green* ?t



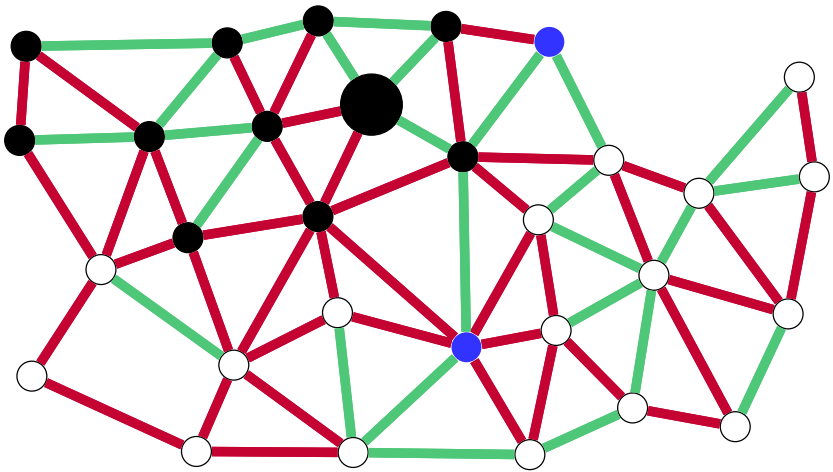
An example

N Red/Green* ?t



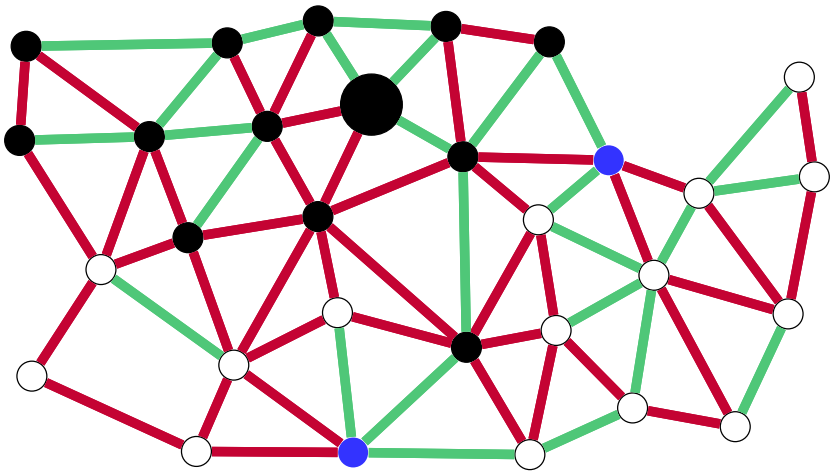
An example

N Red/Green* ?t



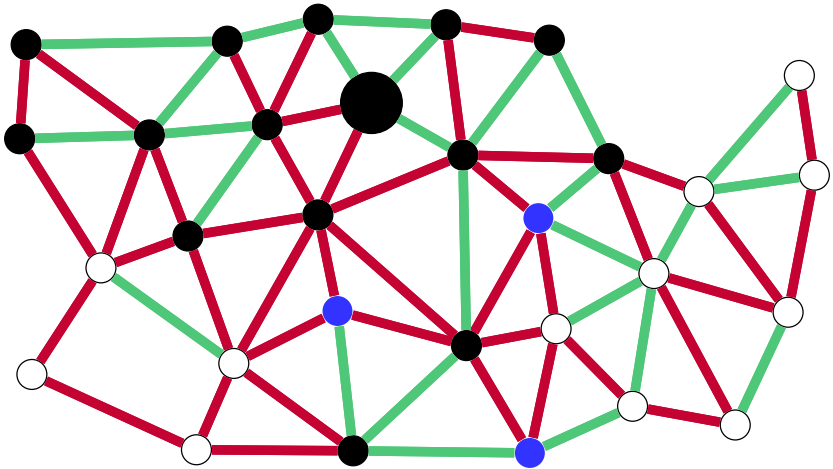
An example

N Red/Green* ?t



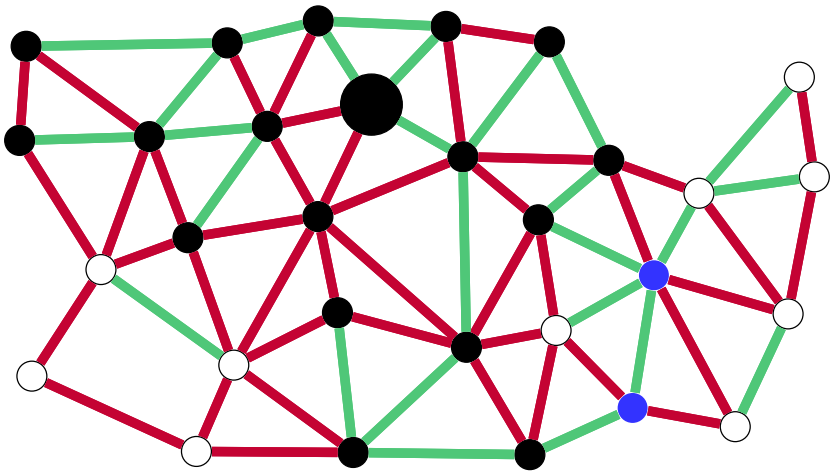
An example

N Red/Green* ?t



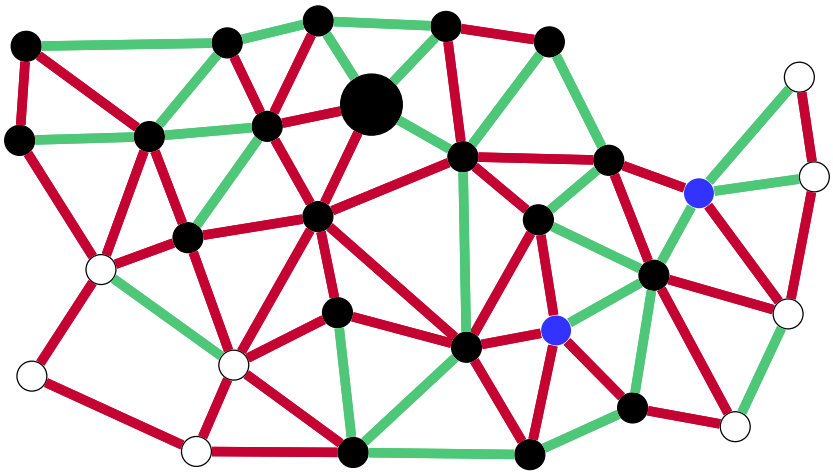
An example

N Red/Green* ?t



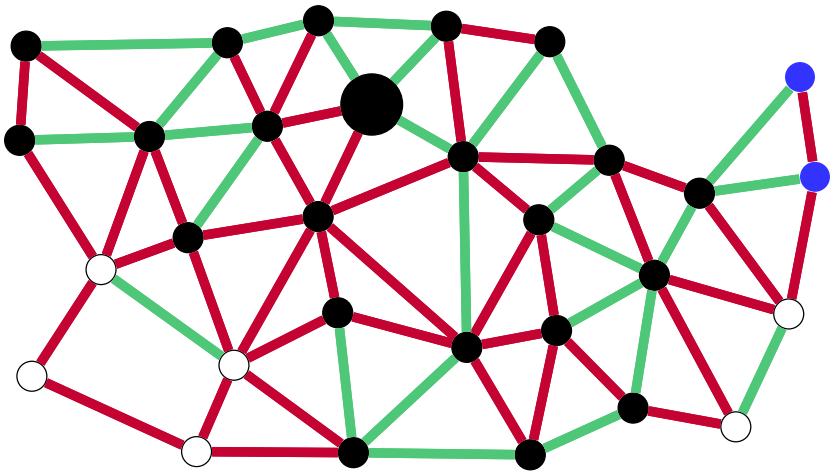
An example

N Red/Green* ?t



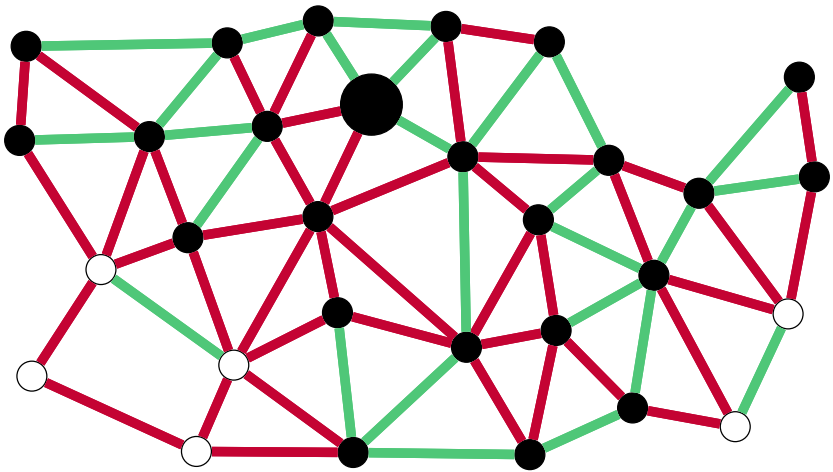
An example

N Red/Green* ?t



An example

N Red/Green* ?t



Observations

N Red/Green* ?t

Computing Green* might be very costly (or even unfeasible) on certain graphs.

Observations

N Red/Green* ?t

Computing Green* might be very costly (or even unfeasible) on certain graphs.

Retrieving the query results without computing the full relation Green* is possible

Observations

N Red/Green* ?t

Computing Green* might be very costly (or even unfeasible) on certain graphs.

Retrieving the query results without computing the full relation Green* is possible

More generally

The evaluation strategy – the *query execution plan* – can have a **huge** impact on performance.

Observations

N Red/Green* ?t

Computing Green* might be very costly (or even unfeasible) on certain graphs.

Retrieving the query results without computing the full relation Green* is possible

More generally

The evaluation strategy – the *query execution plan* – can have a **huge** impact on performance.

Query answering might be feasible... or not. Even for graphs of moderate size.

Observations

N Red/Green* ?t

Computing Green* might be very costly (or even unfeasible) on certain graphs.

Retrieving the query results without computing the full relation Green* is possible

More generally

The evaluation strategy – the *query execution plan* – can have a **huge** impact on performance.

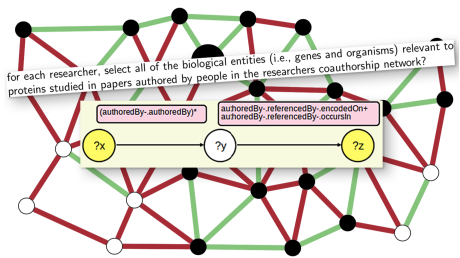
Query answering might be feasible... or not. Even for graphs of moderate size.

Query execution plan is crucial.

Other examples

What if...

- the query contains more than one variable?
  ?x Red*/Green ?y
- more general forms of navigation?
  ?x (Red | Green)* N
- Several recursions? Which exploration should start first?



Fundamental Problem

Given a recursive graph query*, how to generate an efficient evaluation plan?

(*) Well-known recursive query language fragments include e.g.: RPQ, C2RPQ, UCRPQ, etc.

What is the state of the theory?

The Theory – Brief (and partial) Recap

1970: Codd's relational algebra (RA), established the domain of databases (multi $G market in 2022)
key ideas: separation of the query language (SQL) from the RA; tables (no recursion)

1979-1990s: attempts at extending RA with recursion, moderate success (limited forms of recursion); research on the Datalog (logic-programming view) side

1999: SQL supports recursive queries (but seen as optimization barriers for optimizers)

2000-2015: boom of NoSQL research (in particular for trees)
key idea: preserve the native data structure (try not to split into tables)

Late 201x: a myriad of graph DB systems (many with poor/no support of recursion). Some with recursion increasingly inspired by Datalog or SQL.

From 2020: RA extended with a more general form of recursion (inspired from tree logics), with application to graphs.

NoSQL
↓
Not only SQL
↓
NewSQL

Relational Data Model (Codd, 1970)

- Data organized in tables (relations)
- 1 database: 1 set of relations
- Type of a relation: set of (possibly typed) column names
- Data seen as tuples (or mappings of column names and values)

author

authId	name	birthDate
1	Edgar F. Codd	19/08/1923
2	Marie Curie	07/11/1867
3	Alain Turing	23/06/1912

article

arId	authId	title
1	1	A relational model of data for large shared data banks
2	1	Understanding relations
3	2	Rayon semis par les composes de l'Uranium et du Thorium
4	3	Computing machinery and intelligence

Relational operators: filter

author

authId	name	birthDate
1	Edgar F. Codd	19/08/1923
2	Marie Curie	07/11/1867
3	Alain Turing	23/06/1912

article

arId	authId	title
1	1	A relational model of data for large shared data banks
2	1	Understanding relations
3	2	Rayon semis par les composes de l'Uranium et du Thorium
4	3	Computing machinery and intelligence

$\sigma_{name=Marie\ Curie}$

authId	name	birthDate
2	Marie Curie	07/11/1867

Relational operators: projection

author

authId	name	birthDate
1	Edgar F. Codd	19/08/1923
2	Marie Curie	07/11/1867
3	Alain Turing	23/06/1912

π_{name}

name
Edgar F. Codd
Marie Curie
Alain Turing

article

arId	authId	title
1	1	A relational model of data for large shared data banks
2	1	Understanding relations
3	2	Rayon semis par les composes de l'Uranium et du Thorium
4	3	Computing machinery and intelligence

Relational operators: natural join

author

authId	name	birthDate
1	Edgar F. Codd	19/08/1923
2	Marie Curie	07/11/1867
3	Alain Turing	23/06/1912

⋈

article

arId	authId	title
1	1	A relational model of data for large shared data banks
2	1	Understanding relations
3	2	Rayon semis par les composes de l'Uranium et du Thorium
4	3	Computing machinery and intelligence

arId	authId	name	birthDate	title
1	1	Edgar F. Codd	19/08/1923	A relational model of data for large shared data banks
2	1	Edgar F. Codd	19/08/1923	Understanding relations
3	2	Marie Curie	07/11/1867	Rayon semis par les composes de l'Uranium et du Thorium

Other relational operators

- Column renaming $\rho_a^b(author)$: renames column a into column b in relation *author*
- Union $\cup$ of two relations (preferrably of same type!)
- ...
- No recursion in Codd's original relational algebra.

Graph Data

Possible representation: a table keeps track of source and target nodes connected by each relation

For instance, for a social network:

follows

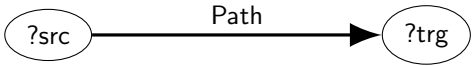
src	trg
Anna	Tom
Tom	Ryan
Ryan	Jane

Overview of some recent results in the area

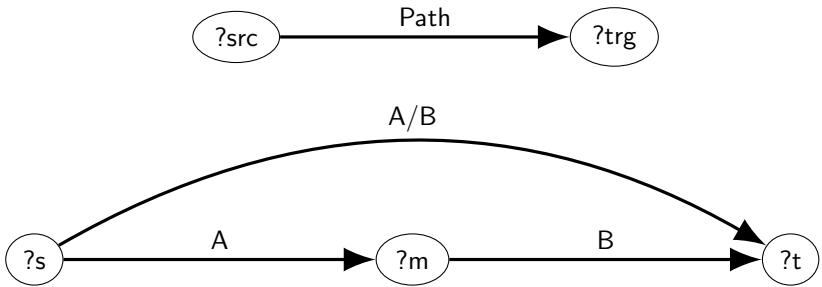
Extended Relational Algebra (μ-RA, 2020)

φ	$::=$		term
	$\mid$	X	relation variable
	$\mid$	$\lvert c_1 \to v_1, \ldots, c_n \to v_n \rvert$	constant (tuple)
	$\mid$	$\sigma_{filter}(\varphi)$	selection
	$\mid$	$\rho_a^b(\varphi)$	column renaming
	$\mid$	$\widetilde{\pi}_a(\varphi)$	column dropping (antiprojection)
	$\mid$	$\beta_a^b(\varphi)$	column duplication
	$\mid$	$\varphi_1 \cup \varphi_2$	union
	$\mid$	$\varphi_1 \bowtie \varphi_2$	natural join
	$\mid$	$\mu(X = \varphi)$	fixpoint

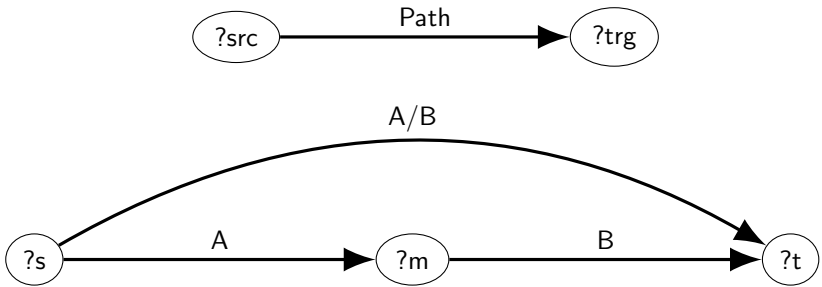
Translation of Path Queries



Translation of Path Queries



Translation of Path Queries



$$Tr(A/B) \quad = \quad \widetilde{\pi}_m \left(\rho_t^m \left(Tr(A) \right) \bowtie \rho_s^m \left(Tr(B) \right) \right)$$

Translation

$$A^* = \text{Empty Path } \textit{or } \underbrace{\text{Path of one or more edges labeled with A}}_{A^+ = A^* / A}$$

Translation

$$A^* = \text{Empty Path } \textit{or } \underbrace{\text{Path of one or more edges labeled with A}}_{A^+ = A^*/A}$$

$$Tr(A^*) \quad = \quad \textit{EmptyPath} \quad \cup \quad Tr(A^*)/A$$

Translation

$$A^* = \text{Empty Path } \textit{or } \underbrace{\text{Path of one or more edges labeled with A}}_{A^+ = A^*/A}$$

$$\begin{aligned} Tr(A^*) &= & \textit{EmptyPath} \quad \cup & \quad Tr(A^*)/A \\ &= \mu\Big(X = & \textit{EmptyPath} \quad \cup & \quad X/A\Big) \end{aligned}$$

Translation

$$A^* = \text{Empty Path } \textit{or } \underbrace{\text{Path of one or more edges labeled with A}}_{A^+ = A^*/A}$$

$$\begin{aligned}
Tr(A^*) &= \quad\quad\quad EmptyPath \quad \cup \quad\quad\quad Tr(A^*)/A \\
&= \mu\Big(X = \quad EmptyPath \quad \cup \quad\quad\quad X/A\Big) \\
&= \mu\Big(X = \quad \beta_t^s\,(\text{AllNodes}) \quad \cup \quad \widetilde{\pi}_m\,(\rho_t^m\,(X) \bowtie \rho_s^m\,(Tr(A)))\Big)
\end{aligned}$$

New Rewrite Rules

Algebraic transformations of fixpoints

New Rewrite Rules

Algebraic transformations of fixpoints

- pushing filters?

$$\sigma_{filter}\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \sigma_{filter}\left(\varphi\right))$$

New Rewrite Rules

Algebraic transformations of fixpoints

- pushing filters?

$$\sigma_{filter}\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \sigma_{filter}\left(\varphi\right))$$

- pushing joins?

$$\psi \bowtie \mu(X = \varphi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi)$$

New Rewrite Rules

Algebraic transformations of fixpoints

- pushing filters?
- pushing joins?
- pushing projections?

$$\sigma_{filter}\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \sigma_{filter}\left(\varphi\right))$$

$$\psi \bowtie \mu(X = \varphi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi)$$

$$\widetilde{\pi}_p\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \widetilde{\pi}_p\left(\varphi\right))$$

New Rewrite Rules

Algebraic transformations of fixpoints

- pushing filters?

$$\sigma_{filter}\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \sigma_{filter}\left(\varphi\right))$$

- pushing joins?

$$\psi \bowtie \mu(X = \varphi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi)$$

- pushing projections?

$$\widetilde{\pi}_p\left(\mu(X = \varphi)\right) \stackrel{?}{=} \mu(X = \widetilde{\pi}_p\left(\varphi\right))$$

- NEW merging fixpoints?

$$\mu(X = \psi \cup \kappa) \bowtie \mu(X = \varphi \cup \xi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi \cup \xi \cup \kappa)$$

New Rewrite Rules

Algebraic transformations of fixpoints

- pushing filters?

$$\sigma_{filter}\left(\mu(X = \varphi)\right) \overset{?}{=} \mu(X = \sigma_{filter}\left(\varphi\right))$$

- pushing joins?

$$\psi \bowtie \mu(X = \varphi) \overset{?}{=} \mu(X = \psi \bowtie \varphi)$$

- pushing projections?

$$\widetilde{\pi}_p\left(\mu(X = \varphi)\right) \overset{?}{=} \mu(X = \widetilde{\pi}_p\left(\varphi\right))$$

- NEW merging fixpoints?

$$\mu(X = \psi \cup \kappa) \bowtie \mu(X = \varphi \cup \xi) \overset{?}{=} \mu(X = \psi \bowtie \varphi \cup \xi \cup \kappa)$$

All rules are semantics-preserving
? : decidable criteria [SIGMOD'20]

Back to simple example

N Red/Green* ?t

Back to simple example

N Red/Green* ?t

$\widetilde{\pi}_{?s}\left(\sigma_{?s=N}\left(\text{Red}/\mu(X = \beta_s^t\left(\text{AllNodes}\right) \cup X/\text{Green}\right)\right)\right)$

Back to simple example

N Red/Green* ?t

$\widetilde{\pi}_{?s}\left(\sigma_{?s=N}\left(\mathsf{Red}/\mu(X = \beta^{t}_{s}\left(\mathsf{AllNodes}\right) \cup X/\mathsf{Green}\right)\right)\right)$

$\widetilde{\pi}_{?s}\left(\sigma_{?s=N}\left(\mu(X = \mathsf{Red}/\beta^{t}_{s}\left(\mathsf{AllNodes}\right) \cup X/\mathsf{Green}\right)\right)\right)$

Back to simple example

N Red/Green* ?t

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mathsf{Red}/\mu(X = \beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red}/\beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red} \cup X/\mathsf{Green})))$

Back to simple example

N Red/Green* ?t

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mathsf{Red}/\mu(X = \beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red}/\beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red} \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\mu(X = \sigma_{?s=N}\,(\mathsf{Red}) \cup X/\mathsf{Green}))$

Back to simple example

N Red/Green* ?t

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mathsf{Red}/\mu(X = \beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red}/\beta^{t}_{s}\,(\mathsf{AllNodes}) \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mu(X = \mathsf{Red} \cup X/\mathsf{Green})))$

$\widetilde{\pi}_{?s}\,(\mu(X = \sigma_{?s=N}\,(\mathsf{Red}) \cup X/\mathsf{Green}))$

$\mu(X = \widetilde{\pi}_{?s}\,(\sigma_{?s=N}\,(\mathsf{Red})) \cup X/\mathsf{Green})$

Intuition of merged fixpoints

Sample query of the form: ?x $\text{Red}^+/\text{Green}^+$?y

plan 1 : unfold Green^+ from right to left, ...

Intuition of merged fixpoints

Sample query of the form: ?x $\text{Red}^+/\text{Green}^+$?y

plan 1 : unfold Green^+ from right to left, ...
plan 2 : start with Red^+, left to right ...

Intuition of merged fixpoints

Sample query of the form: ?x $\text{Red}^+/\text{Green}^+$?y

plan 1	:	unfold Green^+ from right to left, ...
plan 2	:	start with Red^+, left to right ...
plan 3	:	start with Red^+, right to left ...
⋮		⋮

Intuition of merged fixpoints

Sample query of the form: ?x $\text{Red}^+/\text{Green}^+$?y

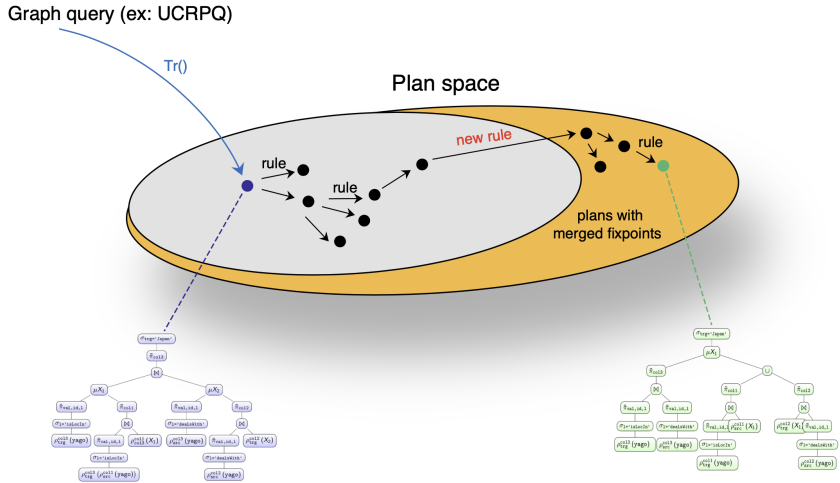
plan 1 : unfold Green^+ from right to left, ...
plan 2 : start with Red^+, left to right ...
plan 3 : start with Red^+, right to left ...
⋮ ⋮
plan i : merged fixpoint:

$$\mu(X = \text{Red}/\text{Green} \cup \text{Red}/X \cup X/\text{Green})$$

start from Red/Green, then hop on left by Red^*, and on the right by Green^*

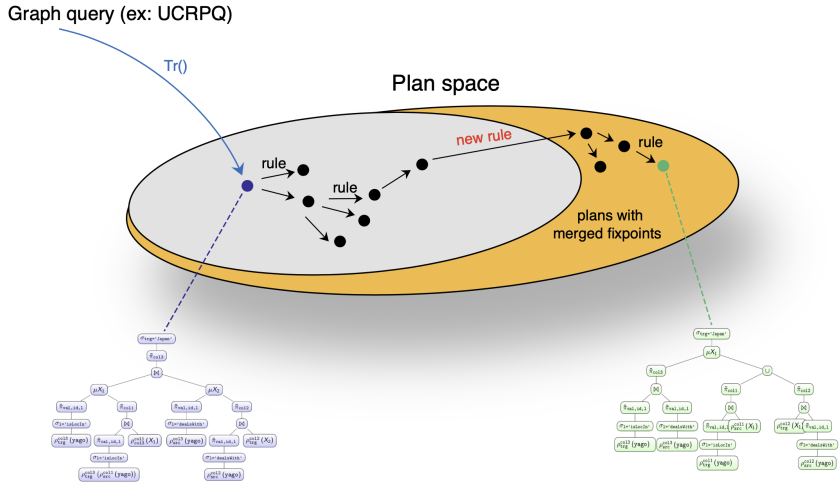
⋮ ⋮

More generally: new broader plan space



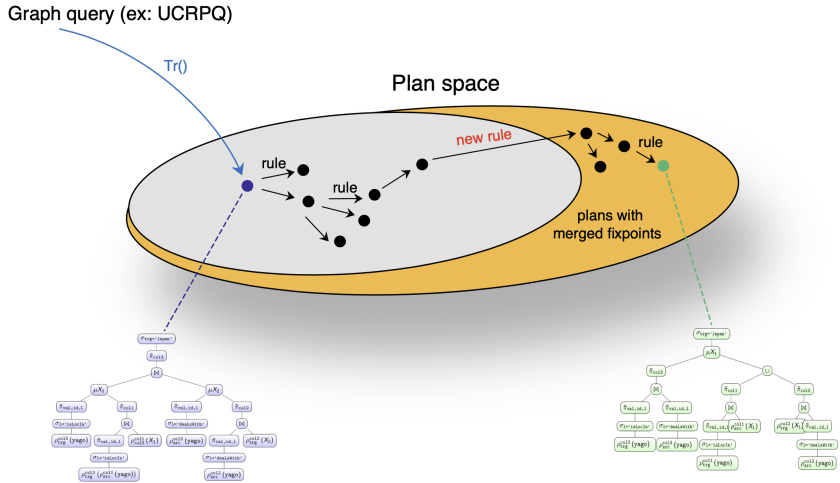
1. A graph query is translated into μ-RA

More generally: new broader plan space



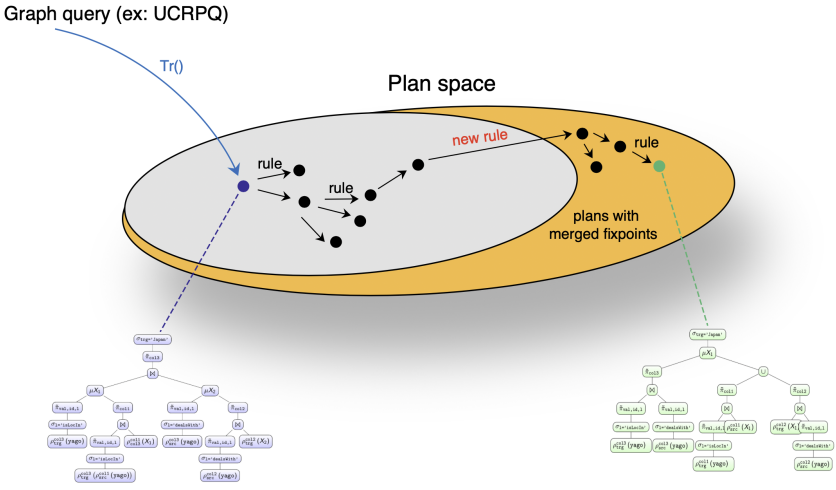
2. With transformation rules, we obtain a space of evaluation plans

More generally: new broader plan space



3. An estimated most efficient plan is selected using a cost estimation and data statistics

More generally: new broader plan space

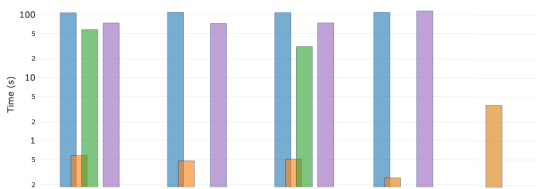


μ-RA plan space is richer compared to previous approaches (e.g. fixpoints can be merged)

Application for Graph Querying

In practice, on centralized systems

- Implementation on top of PostgreSQL (a popular relational database management system)
- Evaluation of queries on knowledge graphs:



Time (log. scale) for third-party queries on `YagoDB` (from `Wikipedia`: 62M edges, 42M nodes)

- Performance gains (orders of magnitude, feasibility) due to new algebraic evaluation plans

Concluding Remarks

A glance at some hot research activity and recent results:

- Powerful recursive graph queries
- Extension of relational algebra with recursion [SIGMOD'20]
- Query plan enumeration (research results to be presented in 2024)
- Extension to neuro-symbolic graph queries...

References

Louis Jachiet, Pierre Genevès, Nils Gesbert and Nabil Layaïda,

On the Optimization of Recursive Relational Queries: Application to Graph Queries.

In Proceedings of the *ACM SIGMOD International Conference on Management of data, 2020* (SIGMOD'20)

Amela Fejza, Pierre Genevès, Nabil Layaïda, Sarah Chlyah

The μ-RA System for Recursive Path Queries over Graphs.

In Proceedings of the *32nd ACM International Conference on Information and Knowledge Management, 2023* (CIKM'23)

Amela Fejza, Pierre Genevès and Nabil Layaïda,

Efficient Enumeration of Recursive Plans in Transformation-based Query Optimizers.

Preprint, 2023: https://inria.hal.science/hal-03692274/file/rlqdag.pdf (to appear in 2024).