On the optimization of recursive relational queries.

DIG Seminar

Louis Jachiet

CRIStAL
Centre de Recherche en Informatique,
Signal et Automatique de Lille

The relational algebra

Generalities

The relational algebra [Cod70]

- a set of base relations

  the tables in SQL

- combined through operators

  union, projection, filter, join, etc.

- operates on named tuples

Syntax

φ	$::=$		term
	$\mid$	X	relation variable
	$\mid$	$\lvert c \to v \rvert$	constant
	$\mid$	$\emptyset$	empty set
	$\mid$	$\varphi_1 \cup \varphi_2$	union
	$\mid$	$\varphi_1 \bowtie \varphi_2$	join
	$\mid$	$\varphi_1 \triangleright \varphi_2$	antijoin
	$\mid$	$\sigma_{filter}(\varphi)$	filter
	$\mid$	$\rho_a^b(\varphi)$	rename
	$\mid$	$\pi_P(\varphi)$	projection

Figure 1: Syntax of the relational algebra

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\pi_{to}(T)$

to
Paris
Saclay
Grenoble

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\rho_{to}^{step}(T)$

from	step
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\rho_{to}^{step}(T) \bowtie \rho_{from}^{step}(T)$

from	step	to
Lille	Paris	Saclay
Lille	Paris	Grenoble
Lille	Saclay	Grenoble
Paris	Saclay	Grenoble

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\sigma_{\text{from}=\text{Lille}}(T)$

from	to
Lille	Paris
Lille	Saclay

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\sigma_{\text{from=Paris}}(T) \cup \sigma_{\text{from=Lille}}(T)$

from	to
Lille	Paris
Lille	Saclay
Paris	Saclay
Paris	Grenoble

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\pi_{to}(T) \triangleright \sigma_{\text{from=Lille}}(T)$

to
Grenoble

Recursive relational algebra

Syntax

φ	$::=$		term
	$\mid$	X	relation variable
	$\mid$	$\lvert c \to v \rvert$	constant
	$\mid$	$\emptyset$	empty set
	$\mid$	$\varphi_1 \cup \varphi_2$	union
	$\mid$	$\varphi_1 \bowtie \varphi_2$	join
	$\mid$	$\varphi_1 \triangleright \varphi_2$	antijoin
	$\mid$	$\sigma_{filter}(\varphi)$	filtering
	$\mid$	$\rho_a^b(\varphi)$	rename
	$\mid$	$\pi_{c_1,\ldots,c_n}(\varphi)$	projection

Figure 2: Syntax of our relational algebra

Syntax

φ	$::=$		term
	$\mid$	X	relation variable
	$\mid$	$\lvert c \rightarrow v \rvert$	constant
	$\mid$	$\emptyset$	empty set
	$\mid$	$\varphi_1 \cup \varphi_2$	union
	$\mid$	$\varphi_1 \bowtie \varphi_2$	join
	$\mid$	$\varphi_1 \triangleright \varphi_2$	antijoin
	$\mid$	$\sigma_{filter}(\varphi)$	filtering
	$\mid$	$\rho_a^b(\varphi)$	rename
	$\mid$	$\tilde{\pi}_c(\varphi)$	anti-projection

Figure 2: Syntax of our relational algebra

Syntax

$$
\begin{array}{rcll}
\varphi & ::= & & \text{term} \\
& | & X & \text{relation variable} \\
& | & |c \to v| & \text{constant} \\
& | & \emptyset & \text{empty set} \\
& | & \varphi_1 \cup \varphi_2 & \text{union} \\
& | & \varphi_1 \bowtie \varphi_2 & \text{join} \\
& | & \varphi_1 \triangleright \varphi_2 & \text{antijoin} \\
& | & \sigma_{filter}(\varphi) & \text{filtering} \\
& | & \rho_a^b(\varphi) & \text{rename} \\
& | & \tilde{\pi}_c(\varphi) & \text{anti-projection} \\
& | & \beta_a^b(\varphi) & \text{duplication}
\end{array}
$$

Figure 2: Syntax of our relational algebra

Syntax

φ	$::=$		term
	$\mid$	X	relation variable
	$\mid$	$\lvert c \to v \rvert$	constant
	$\mid$	$\emptyset$	empty set
	$\mid$	$\varphi_1 \cup \varphi_2$	union
	$\mid$	$\varphi_1 \bowtie \varphi_2$	join
	$\mid$	$\varphi_1 \triangleright \varphi_2$	antijoin
	$\mid$	$\sigma_{filter}(\varphi)$	filtering
	$\mid$	$\rho_a^b(\varphi)$	rename
	$\mid$	$\tilde{\pi}_c(\varphi)$	anti-projection
	$\mid$	$\beta_a^b(\varphi)$	duplication
	$\mid$	$\mu(X = \varphi)$	fixpoint

Figure 2: Syntax of our relational algebra

Differences

Anti-projection
Remove a column

$$\tilde{\pi}_{d_1}\left(\dots\tilde{\pi}_{d_k}\left(\varphi\right)\dots\right) = \pi_{c_1\dots c_n}\left(\varphi\right)$$

Differences

Anti-projection
Remove a column

$$\tilde{\pi}_{d_1}\left(\ldots\tilde{\pi}_{d_k}\left(\varphi\right)\ldots\right) = \pi_{c_1\ldots c_n}\left(\varphi\right)$$

Duplication
Copy a column

$$\beta_a^b\left(\varphi\right) = \sigma_{a=b}\left(\varphi \bowtie \rho_a^b\left(\varphi\right)\right)$$

Differences

Anti-projection
Remove a column

$$\tilde{\pi}_{d_1}\left(\ldots \tilde{\pi}_{d_k}\left(\varphi\right)\ldots\right) = \pi_{c_1 \ldots c_n}\left(\varphi\right)$$

Duplication
Copy a column

$$\beta_a^b\left(\varphi\right) = \sigma_{a=b}\left(\varphi \bowtie \rho_a^b\left(\varphi\right)\right)$$

Fixpoints
Compute the least fixpoint of a function $S \to \varphi[X/S]$

$$[\![\mu(X = \varphi)]\!]_V = lim_{n \to \infty} U_n \qquad U_0 = \emptyset$$
$$U_{n+1} = U_n \cup [\![\varphi]\!]_{V[X/U_n]}$$

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\tilde{\pi}_{from}(T)$

to
Paris
Saclay
Grenoble

Examples

T

from	to
Lille	Paris
Lille	Saclay
Paris	Grenoble
Paris	Saclay
Saclay	Grenoble

$\beta^{to}_{from}(\tilde{\pi}_{to}(T))$

from	to
Lille	Lille
Saclay	Saclay
Paris	Paris

Examples

$$T^{+} = \mu(X = T \cup T/X)$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

from	to

Examples

$$T^+ = \mu(X = T \cup \tilde{\pi}_s\,(\rho^s_{to}\,(X) \bowtie \rho^s_{from}\,(T)))$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

Examples

$$T^{+} = \mu(X = T \cup \tilde{\pi}_s\,(\rho^s_{to}\,(X) \bowtie \rho^s_{from}\,(T)))$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble
Lille	Saclay
Paris	Lyon
Saclay	Grenoble

Examples

$$T^{+} = \mu(X = T \cup \tilde{\pi}_s\,(\rho^s_{to}\,(X) \bowtie \rho^s_{from}\,(T)))$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble
Lille	Saclay
Paris	Lyon
Saclay	Grenoble
Lille	Lyon
Paris	Grenoble

Examples

$$T^+ = \mu(X = T \cup \tilde{\pi}_s\,(\rho^s_{to}\,(X) \bowtie \rho^s_{from}\,(T)))$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble
Lille	Saclay
Paris	Lyon
Saclay	Grenoble
Lille	Lyon
Paris	Grenoble
Lille	Grenoble

Limitations

Limitations on fixpoints

- recursive variables must appear positively

$$\textit{No } \mu(X = R \triangleright X)$$

Limitations

Limitations on fixpoints

- recursive variables must appear positively

$$\textit{No } \mu(X = R \triangleright X)$$

- no join between recursive terms

$$\textit{No } \mu(X = X \bowtie X)$$

Limitations

Limitations on fixpoints

- recursive variables must appear positively

$$\textit{No } \mu(X = R \triangleright X)$$

- no join between recursive terms

$$\textit{No } \mu(X = X \bowtie X)$$

- no mutually recursive fixpoints

$$\textit{No } \mu(X = \mu(Y = X \cup Y))$$

Limitations

Limitations on fixpoints

- recursive variables must appear positively

$$\textit{No } \mu(X = R \triangleright X)$$

- no join between recursive terms

$$\textit{No } \mu(X = X \bowtie X)$$

- no mutually recursive fixpoints

$$\textit{No } \mu(X = \mu(Y = X \cup Y))$$

→ *corresponds to linear datalog!*

→ *superset of `WITH RECURSIVE` in SQL!*

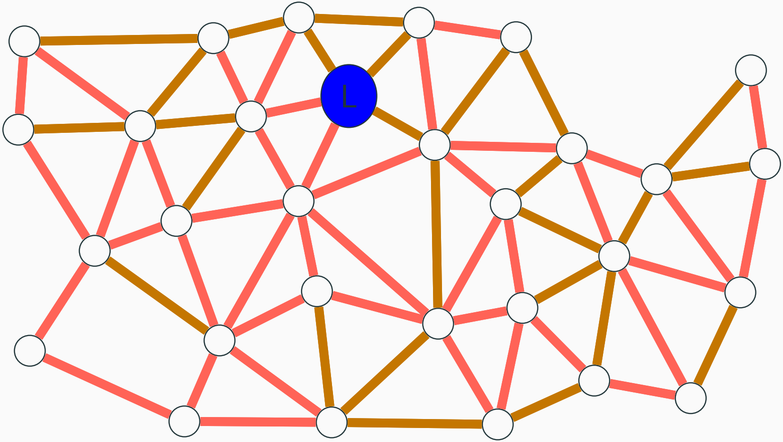
Performance of recursive queries

An example

:Lille `:TGV`/`:Bus*` `?o`

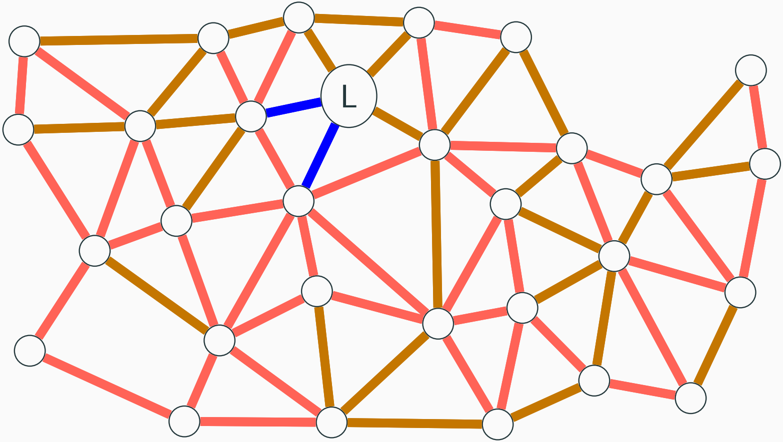
An example

:Lille `:TGV`/`:Bus*` `?o`



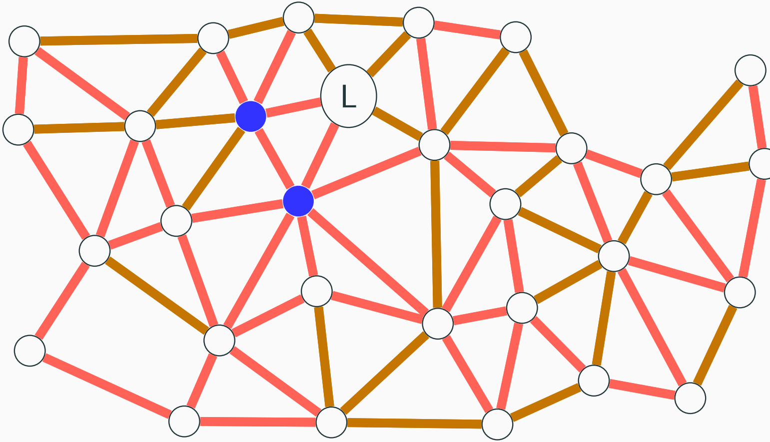
An example

:Lille :TGV/:Bus* ?o



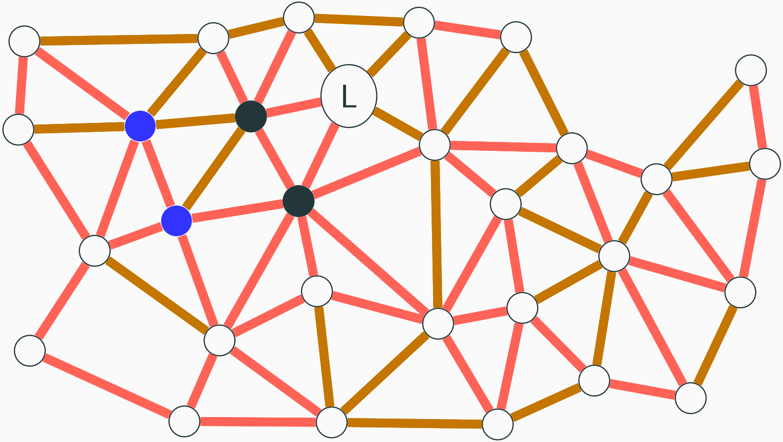
An example

:Lille :TGV/:Bus* ?o



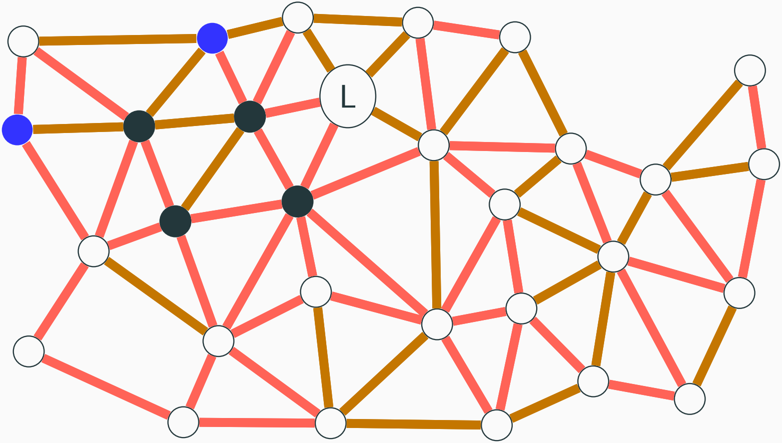
An example

:Lille :TGV/:Bus* ?o



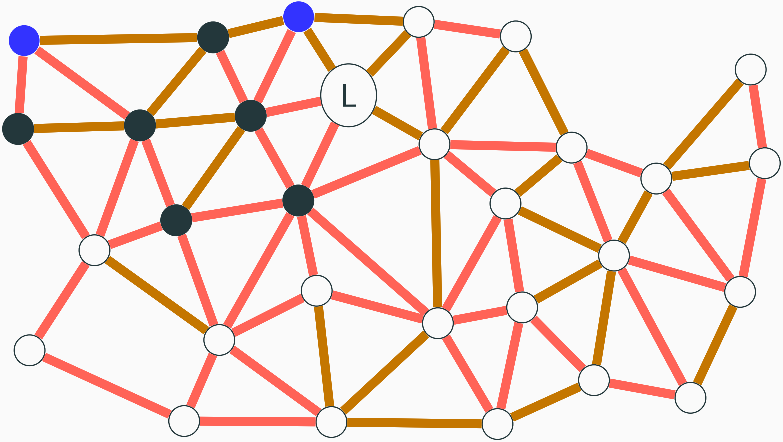
An example

:Lille :TGV/:Bus* ?o



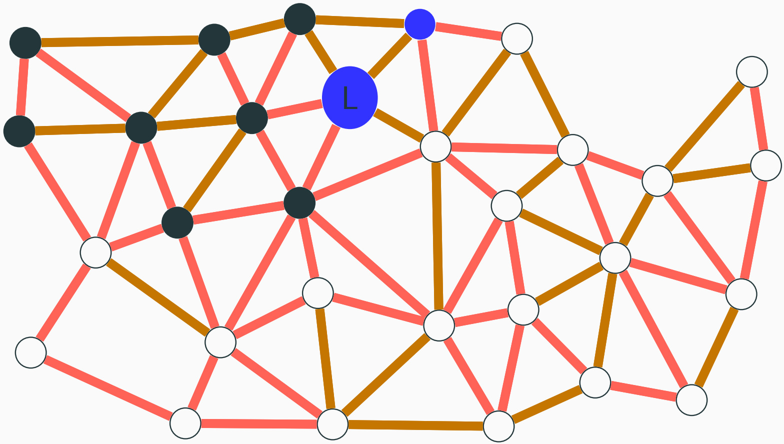
An example

:Lille :TGV/:Bus* ?o



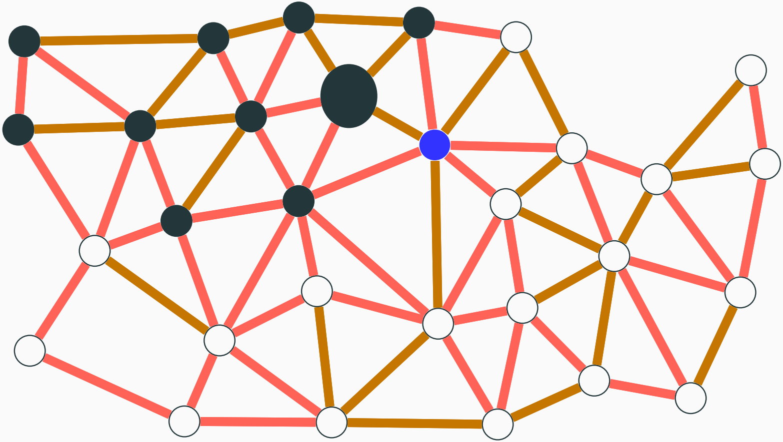
An example

:Lille :TGV/:Bus* ?o



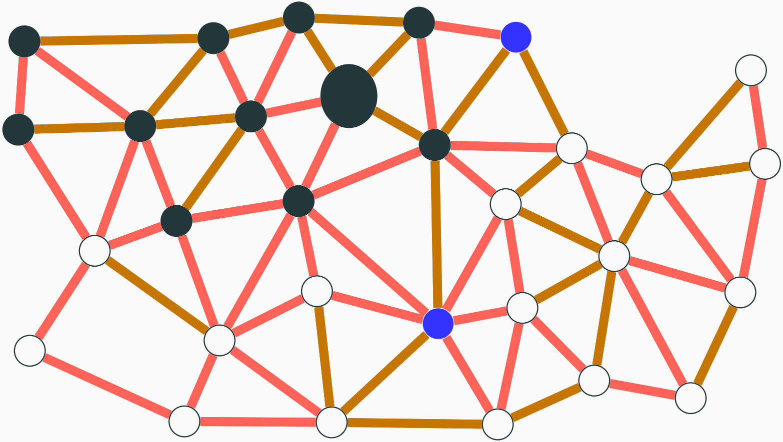
An example

:Lille :TGV/:Bus* ?o



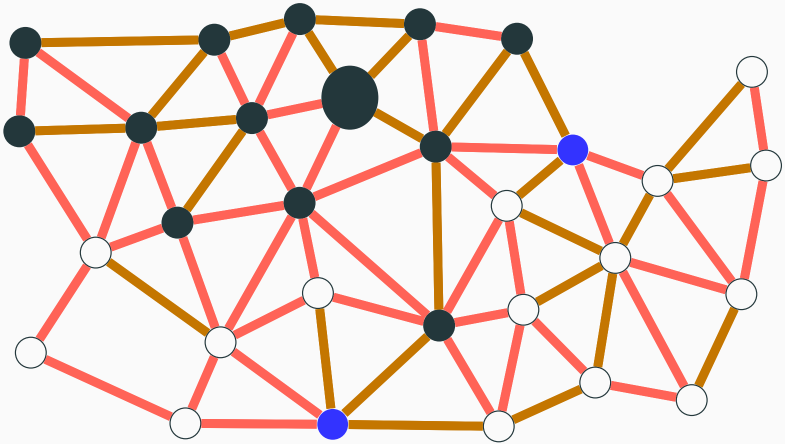
An example

:Lille :TGV/:Bus* ?o



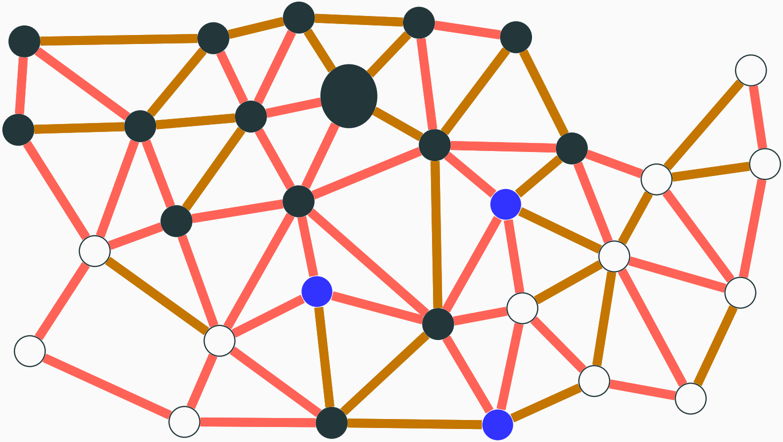
An example

:Lille :TGV/:Bus* ?o



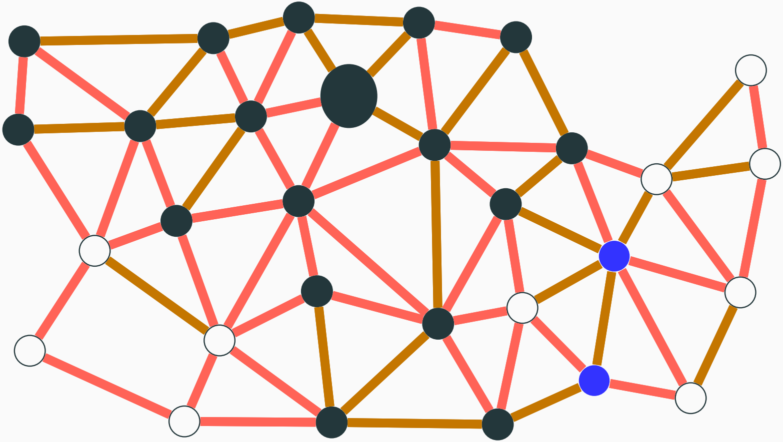
An example

:Lille :TGV/:Bus* ?o



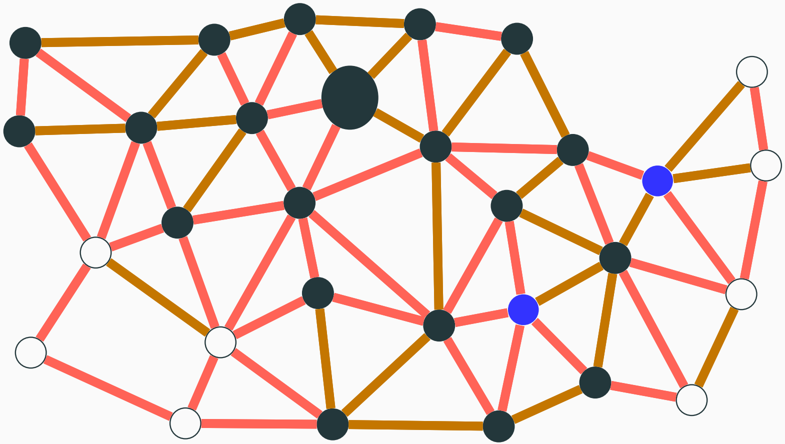
An example

:Lille :TGV/:Bus* ?o



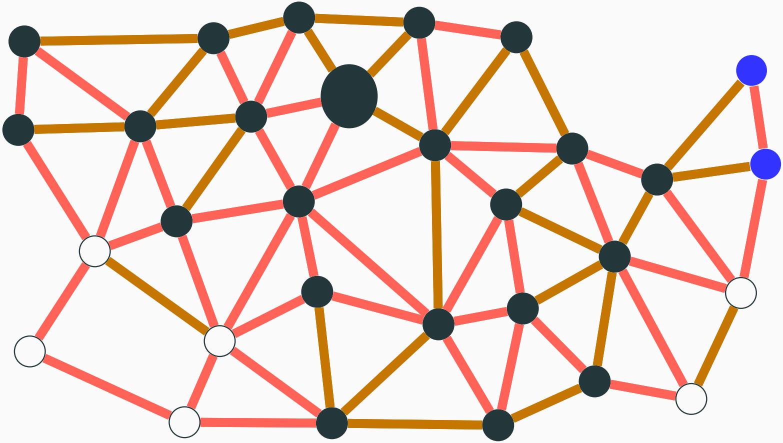
An example

:Lille :TGV/:Bus* ?o



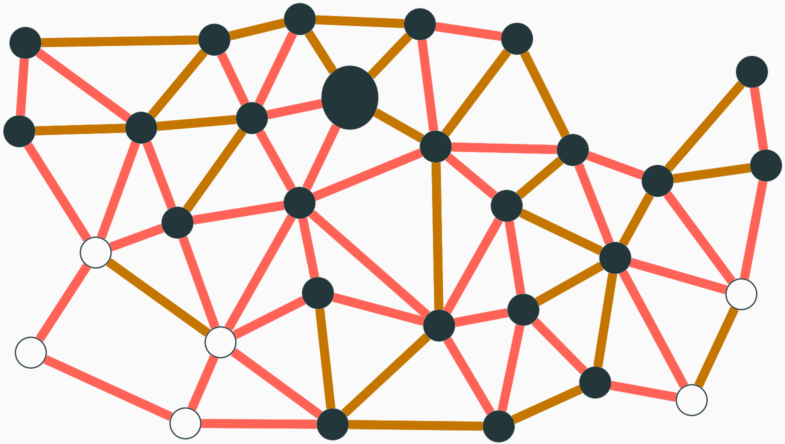
An example

:Lille `:TGV/:Bus* ?o`



An example

:Lille :TGV/:Bus* ?o



An example

`:L :TGV/:Bus* ?o`

An example

```
:L :TGV/:Bus* ?o
```

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathsf{TGV}/\mu(X = \beta_s^o\left(AllNodes\right) \cup X/:\mathsf{Bus})\right)\right)$$

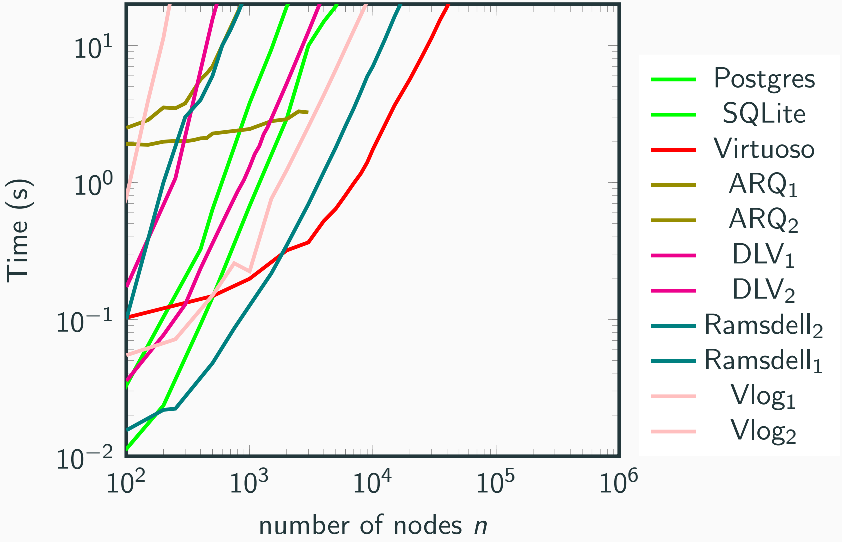
An example

:L `:TGV`/`:Bus`* ?o

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathrm{TGV}/\mu(X = \beta^{o}_{s}\left(AllNodes\right) \cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathrm{TGV}/\mu(X = \beta^{o}_{s}\left(AllNodes\right) \cup :\mathrm{Bus}/X)\right)\right)$$

Benchmarking



Benchmarking

Logicblox

- Materialization of all intermediate predicate
- ... except for “on demand” predicate
- therefore manual optimization

Rewrite rules

Rewrite rules for fixpoints

- pushing filters?

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

Rewrite rules

Rewrite rules for fixpoints

- pushing filters?

$$\sigma_{\mathit{filter}}\,(\mu(X = \varphi)) \stackrel{?}{=} \mu(X = \sigma_{\mathit{filter}}\,(\varphi))$$

- pushing joins?

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

Rewrite rules

Rewrite rules for fixpoints

- pushing filters?

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

- pushing joins?

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

- pushing antijoins?

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

Rewrite rules

Rewrite rules for fixpoints

- pushing filters?

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

- pushing joins?

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

- pushing antijoins?

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

- pushing anti-projections?

$$\tilde{\pi}_p(\mu(X = \varphi)) \stackrel{?}{=} \mu(X = \tilde{\pi}_p(\varphi))$$

Rewrite rules

Rewrite rules for fixpoints

- pushing filters?

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

- pushing joins?

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

- pushing antijoins?

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

- pushing anti-projections?

$$\tilde{\pi}_p\,(\mu(X = \varphi)) \stackrel{?}{=} \mu(X = \tilde{\pi}_p\,(\varphi))$$

- combine fixpoints?

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

Rewrite rules for fixpoints

- Reverse fixpoints?

An example

:L :TGV/:Bus* ?o

An example

:L `:TGV`/`:Bus`* ?o

$$\tilde{\pi}_{?s}\,(\sigma_{?s=:L}\,(:\mathsf{TGV}/\mu(X = \beta_{s}^{o}\,(AllNodes) \cup X/:\mathsf{Bus})))$$

An example

:L `:TGV/:Bus* ?o`

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:TGV/\mu(X = \beta_s^o\left(AllNodes\right) \cup X/:Bus)\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X = :TGV/\beta_s^o\left(AllNodes\right) \cup X/:Bus)\right)\right)$$

An example

:L :TGV/:Bus* ?o

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathrm{TGV}/\mu(X=\beta_s^o\left(AllNodes\right)\cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X=:\mathrm{TGV}/\beta_s^o\left(AllNodes\right)\cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X=:\mathrm{TGV}\cup X/:\mathrm{Bus})\right)\right)$$

An example

:L `:TGV`/`:Bus`* ?o

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:TGV/\mu(X = \beta_s^o\left(AllNodes\right) \cup X/:Bus)\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X = :TGV/\beta_s^o\left(AllNodes\right) \cup X/:Bus)\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X = :TGV \cup X/:Bus)\right)\right)$$

$$\tilde{\pi}_{?s}\left(\mu(X = \sigma_{?s=:L}\left(:TGV\right) \cup X/:Bus)\right)$$

An example

:L :TGV/:Bus* ?o

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathrm{TGV}/\mu(X = \beta_s^o\left(AllNodes\right) \cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X = :\mathrm{TGV}/\beta_s^o\left(AllNodes\right) \cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(\mu(X = :\mathrm{TGV} \cup X/:\mathrm{Bus})\right)\right)$$

$$\tilde{\pi}_{?s}\left(\mu(X = \sigma_{?s=:L}\left(:\mathrm{TGV}\right) \cup X/:\mathrm{Bus})\right)$$

$$\mu(X = \tilde{\pi}_{?s}\left(\sigma_{?s=:L}\left(:\mathrm{TGV}\right)\right) \cup X/:\mathrm{Bus})$$

Other methods of optimization

Datalog?
No combination of fixpoints

Automata based techniques?
Step by Step and no conjunction

$(a/b/c)^+ \quad vs \quad ((a/b)/c)^+ \quad vs \quad ((a/b/c))^+$

Special joins (RDF-3X ferari)?
Compute efficiently $A \bowtie (B)*$ but same problem...

Waveguide
Efficient on a single RPQ but cannot optimize across RPQ.

Theoretical framework

Decomposed fixpoints
Given a fixpoint $\mu(X = \varphi)$ it can be rewritten to $\mu(X = \varphi_{con} \cup \varphi_{rec})$ with:

- φ_{con} constant, *i.e.* $[\![\varphi_{con}]\!]_{V[X/\emptyset]} = [\![\varphi_{con}]\!]_{V[X/S]}$
- φ_{rec} recursive, *i.e.* $[\![\varphi_{con}]\!]_{V[X/\emptyset]} = \emptyset$

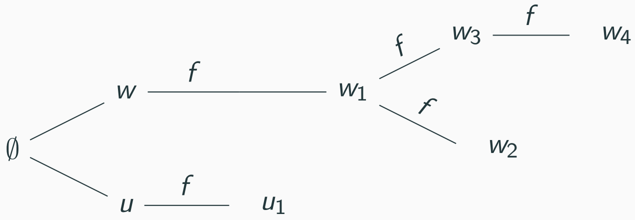
Linearity of fixpoints
Given a fixpoint $\mu(X = \varphi)$:

$$[\![\varphi]\!]_{V[X/S]} = [\![\varphi]\!]_{V[X/\emptyset]} \bigcup_{w \in S} [\![\varphi]\!]_{V[X/\{w\}]}$$

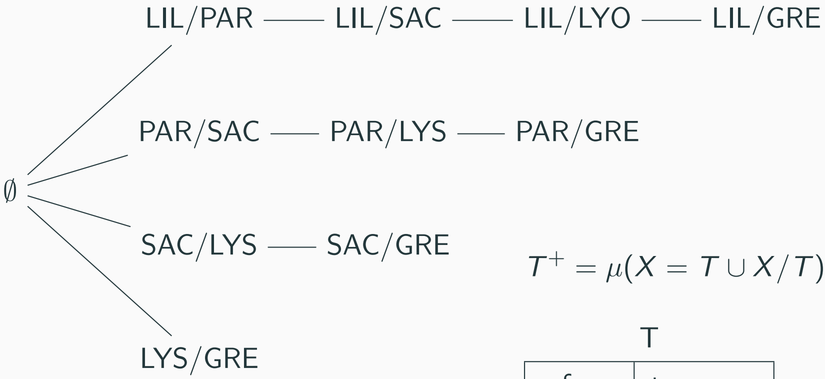
Linearity of fixpoints
Given a fixpoint $\mu(X = \varphi)$:

$$[\![\varphi]\!]_{V[X/S]} = [\![\varphi]\!]_{V[X/\emptyset]} \bigcup_{w \in S} [\![\varphi]\!]_{V[X/\{w\}]}$$

Lineage
For each $m \in U_{i+1} \setminus U_i$ we can find $w \in U_i$ such that $m \in f(w)$ with $f(w) = [\![\varphi]\!]_{V[X/\{w\}]} \setminus [\![\varphi]\!]_{V[X/\emptyset]}$.



Examples



$$T^+ = \mu(X = T \cup X/T)$$

T

from	to
Lille	Paris
Paris	Saclay
Saclay	Lyon
Lyon	Grenoble

Invariant

How the elements of $f(w)$ depend on w?

Invariant

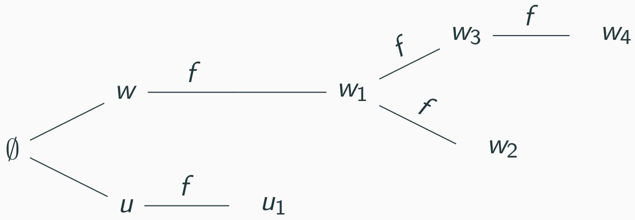
How the elements of $f(w)$ depend on w?

Stabilizers
For each w, $m \in f(w)$ and $c \in stab(\varphi)$: $m(c) = w(c)$.

Invariant

How the elements of $f(w)$ depend on w?

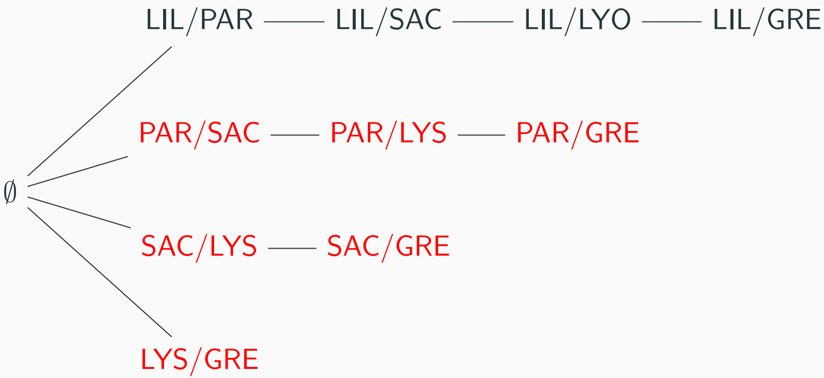
Stabilizers
For each w, $m \in f(w)$ and $c \in stab(\varphi)$: $m(c) = w(c)$.



$$\sigma_{filter}(\mu(X = \varphi)) = \mu(X = \sigma_{filter}(\varphi))$$

when *filter* operates on $stab(\varphi)$

Examples



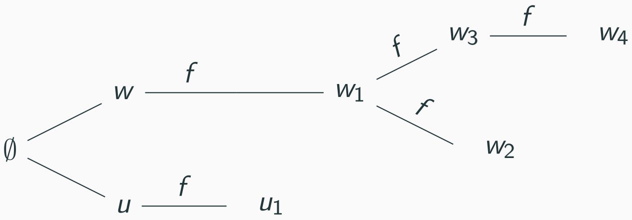
$$\sigma_{from=Lille}\left(T^{+}\right) = \mu(X = \sigma_{from=Lille}(T) \cup X/T)$$

$$\sigma_{to=Lille}\left(T^{+}\right) \neq \mu(X = \sigma_{from=Lille}(T) \cup X/T)$$

Invariant

How the elements of $f(w)$ depend on w?

Added columns
For each $c \in add(\varphi)$: $f(w) \bowtie |c \to v| = f(w \bowtie |c \to v|)$



$$\psi \bowtie \mu(X = \varphi) = \mu(X = \psi \bowtie \varphi)$$

when $sort(\psi) \subseteq stab(\varphi)$

and $sort(\psi) \subseteq add(\varphi) \cup sort(\mu(X = \varphi))$

Rewrite rules

Rewrite rules for fixpoints

- pushing filters

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

- pushing joins

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

- pushing antijoins

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

- pushing anti-projections

$$\tilde{\pi}_p\,(\mu(X=\varphi)) \stackrel{?}{=} \mu(X=\tilde{\pi}_p\,(\varphi))$$

- combine fixpoints

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

Streams

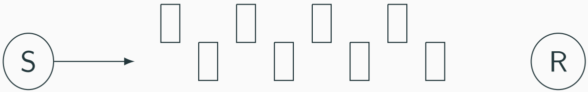
Streams are one-way communication channels

Streams

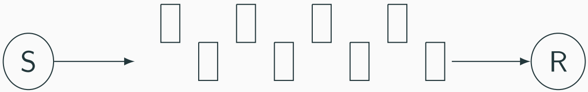
Streams are one-way communication channels

Streams

Streams are one-way communication channels

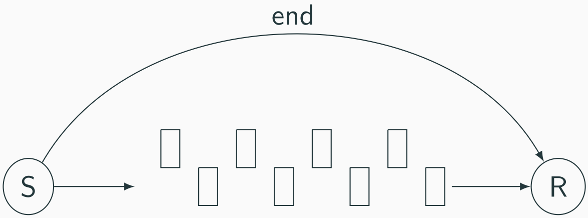


Streams are one-way communication channels



Streams

Streams are one-way communication channels



Streams: a a good abstraction for iterative distributed execution

- no order of messages

Streams: a a good abstraction for iterative distributed execution

- no order of messages
- not necessarily a DAG

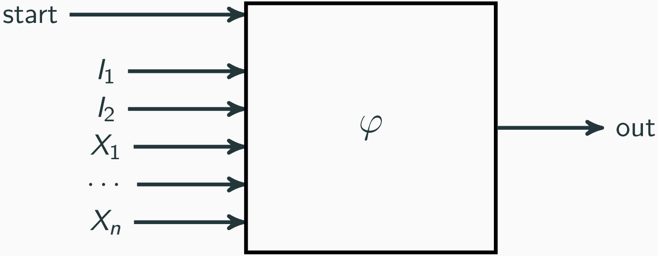
Streams: a a good abstraction for iterative distributed execution

- no order of messages
- not necessarily a DAG
- fast single machine communication and slow inter-machine communication

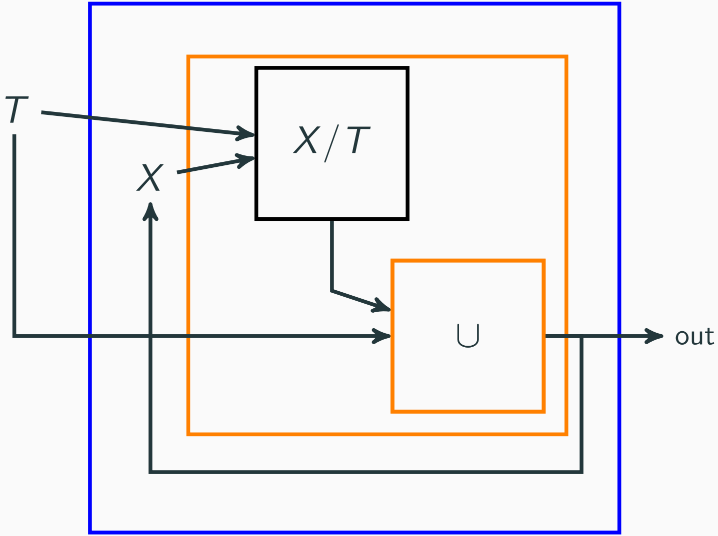
Streams: a a good abstraction for iterative distributed execution

- no order of messages
- not necessarily a DAG
- fast single machine communication and slow inter-machine communication
- partial typing of message content (for fast serialization)

Execution of μ-algebra terms with streams



Streams for $\mu(X = X/T \cup T)$



Benchmarking

Q2

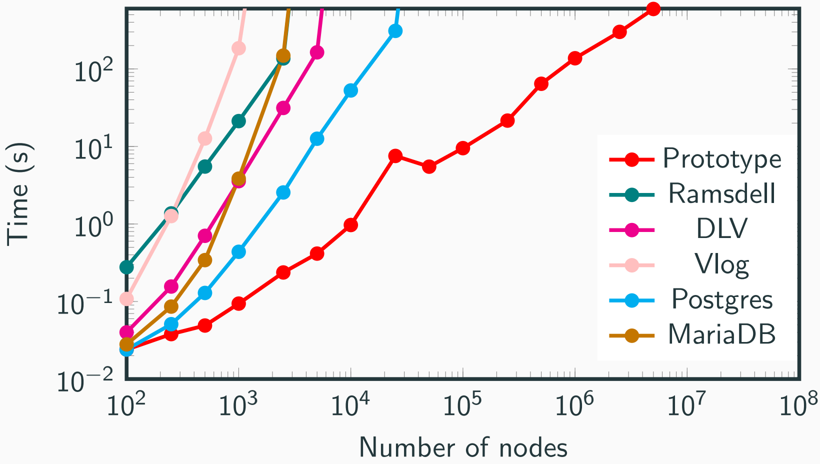


Figure 4: $?a\ \ (P1+)/(P5+)\ \ ?b$.

Q3

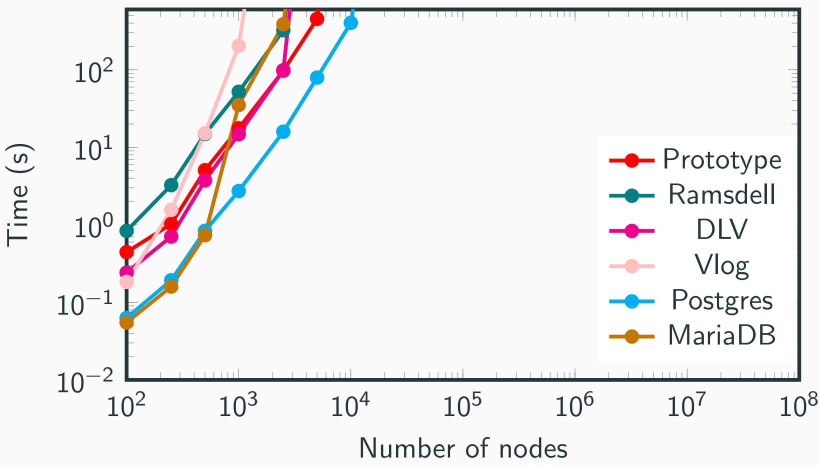


Figure 5: $?a\ \ (P1+)/P2\ \ ?b\ .\ ?b\ \ P3+\ \ ?c.$

Q7

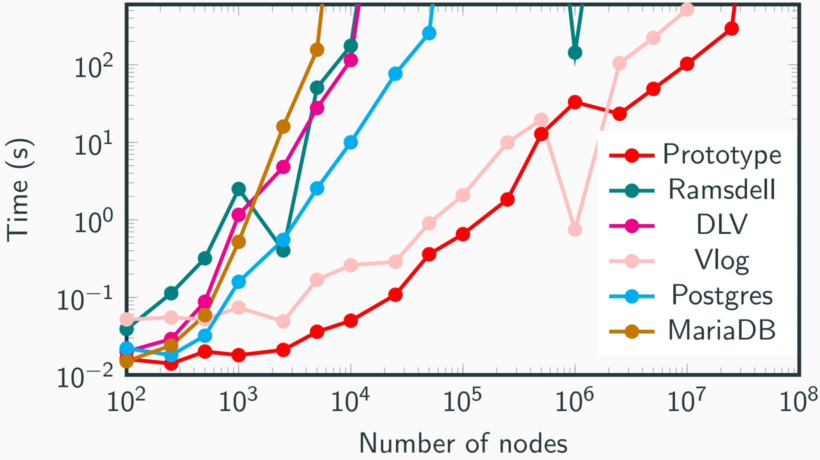


Figure 6: $N0\ P1/(P2+)\ ?a$

Q10

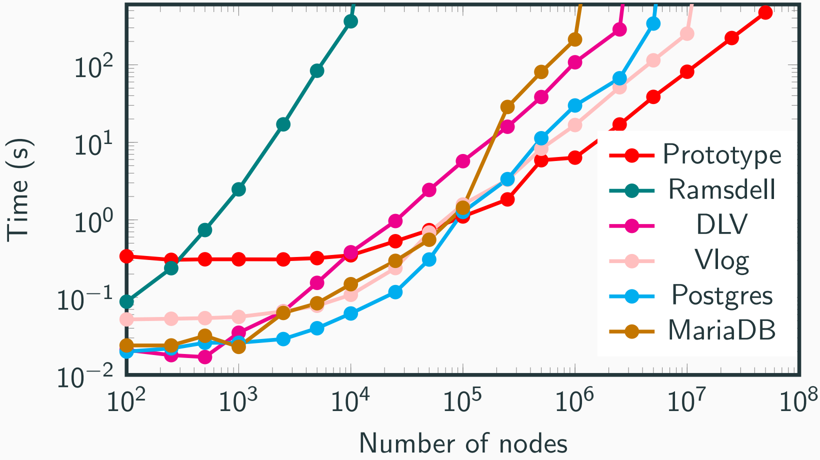
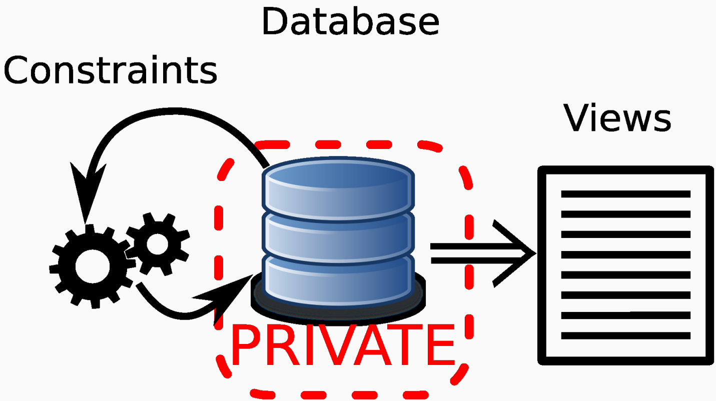


Figure 7: $?a\ \ (P4+)/(P5+)/(P3+)\ \ ?b$

Why recursive queries?

- Evaluate property paths
- Evaluate general recursive queries
- OBDA without rewriting nor materialization

What am I doing now?



Questions?

Edgar F Codd.
A relational model of data for large shared data banks.
Communications of the ACM, 13(6):377–387, 1970.

Semantics

Semantics

$$
\begin{aligned}
[\![\varphi_1 \bowtie \varphi_2]\!]_V &= \{m_1 + m_2 \mid m_1 \in [\![\varphi_1]\!]_V \wedge m_2 \in [\![\varphi_2]\!]_V \wedge m_1 \sim m_2\} \\
[\![\varphi_1 \cup \varphi_2]\!]_V &= [\![\varphi_1]\!]_V \cup [\![\varphi_2]\!]_V \\
[\![\varphi_1 \triangleright \varphi_2]\!]_V &= \{m \in [\![\varphi_1]\!]_V \mid \forall m' \in [\![\varphi_2]\!]_V \ \neg(m' \sim m)\} \\
[\![\tilde{\pi}_a(\varphi)]\!]_V &= \left\{\{c \to v \in m \mid c \neq a\} \;\middle|\; m \in [\![\varphi]\!]_V\right\} \\
[\![X]\!]_V &= V(X) \\
[\![\beta_a^b(\varphi)]\!]_V &= \Big\{\{c \to v \in m \mid c \neq b\} \cup \{b \to v \mid a \to v \in m\} \\
&\qquad \Big| \; m \in [\![\varphi]\!]_V\Big\} \\
[\![\sigma_{filter}(\varphi)]\!]_V &= \{m \mid m \in [\![\varphi]\!]_V \wedge filter(m) = \top\} \\
[\![\mu(X = \varphi)]\!]_V &= [\![X]\!]_{V[X/U_\infty]}, \ U_0 = \emptyset, \ U_{i+1} = U_i \cup [\![\varphi]\!]_{V[X/U_i]}
\end{aligned}
$$