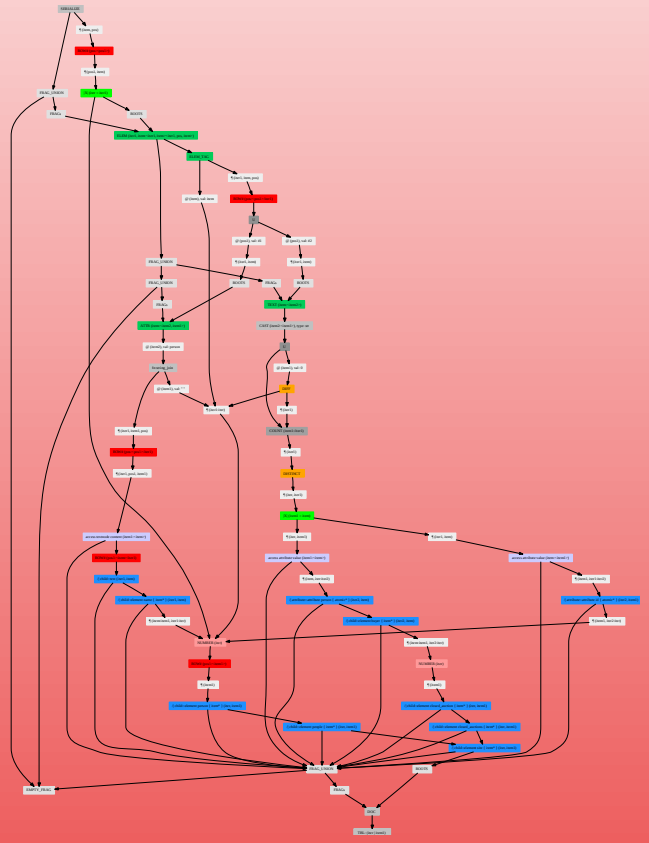


Pathfinder: A Relational Query Optimizer Explores XQuery Terrain

Torsten Grust · Jan Rittinger · Jens Teubner · TU München

Source Language: XQuery Core

Pathfinder consumes XQuery with arbitrary expression nestings ...

literals	`42`, `"foo"`, `()`, ...
arithmetics	e_1 `+` e_2, e_1 `-` e_2, ...
built-in functions	`fn:sum(`e`)`, `fn:count(`e`)`, `fn:doc(`uri`)`, ...
variable bindings	`let $`v `:=` e_1 `return` e_2
iteration	`for $`v `at $`p `in` e_1 `return` e_2
conditionals	`if` p `then` e_1 `else` e_2
sequence construction	e_1`,` e_2
user-defd. functions	f `(`e_1`,` e_2`, ...,` e_n`)`
element construction	`element` e_1 `{` e_2 `}`
XPath steps	$e/\alpha::\nu$ (full axis feature)
⋮	⋮

Source Language: XQuery Core

Pathfinder consumes XQuery with arbitrary expression nestings . . .

literals	`42`, `"foo"`, `()`, ...
arithmetics	e_1 `+` e_2, e_1 `-` e_2, ...
built-in functions	`fn:sum(`e`)`, `fn:count(`e`)`, `fn:doc(`uri`)`, ...
variable bindings	`let $`v `:=` e_1 `return` e_2
iteration	`for $`v `at $`p `in` e_1 `return` e_2
conditionals	`if` p `then` e_1 `else` e_2
sequence construction	e_1, e_2
user-defd. functions	f`(`e_1`,` e_2`, ...,` e_n`)`
element construction	`element` e_1 `{` e_2 `}`
XPath steps	e/α`::`ν (full axis feature)
⋮	⋮

FULL XQUERY SUPPORT

Target Language: Relational Algebra

. . . and compiles them into plans of a standard relational algebra.

π	column projection, renaming
σ	row selection
$\bowtie$	equi-join
$\times$	Cartesian product
$\dot{\cup}$, $\setminus$	disjoint union, difference
δ	duplicate elimination
ϱ	row numbering
$\lrcorner$	staircase join[1]
ε, τ	element/text node construction[1]
$\circledast$	arithmetic/comparison/Boolean operator $*$

- Operates on node (not tree!) level and 1NF relations.

[1] Syntactic sugar; expressible by remaining operators.

Target Language: Relational Algebra

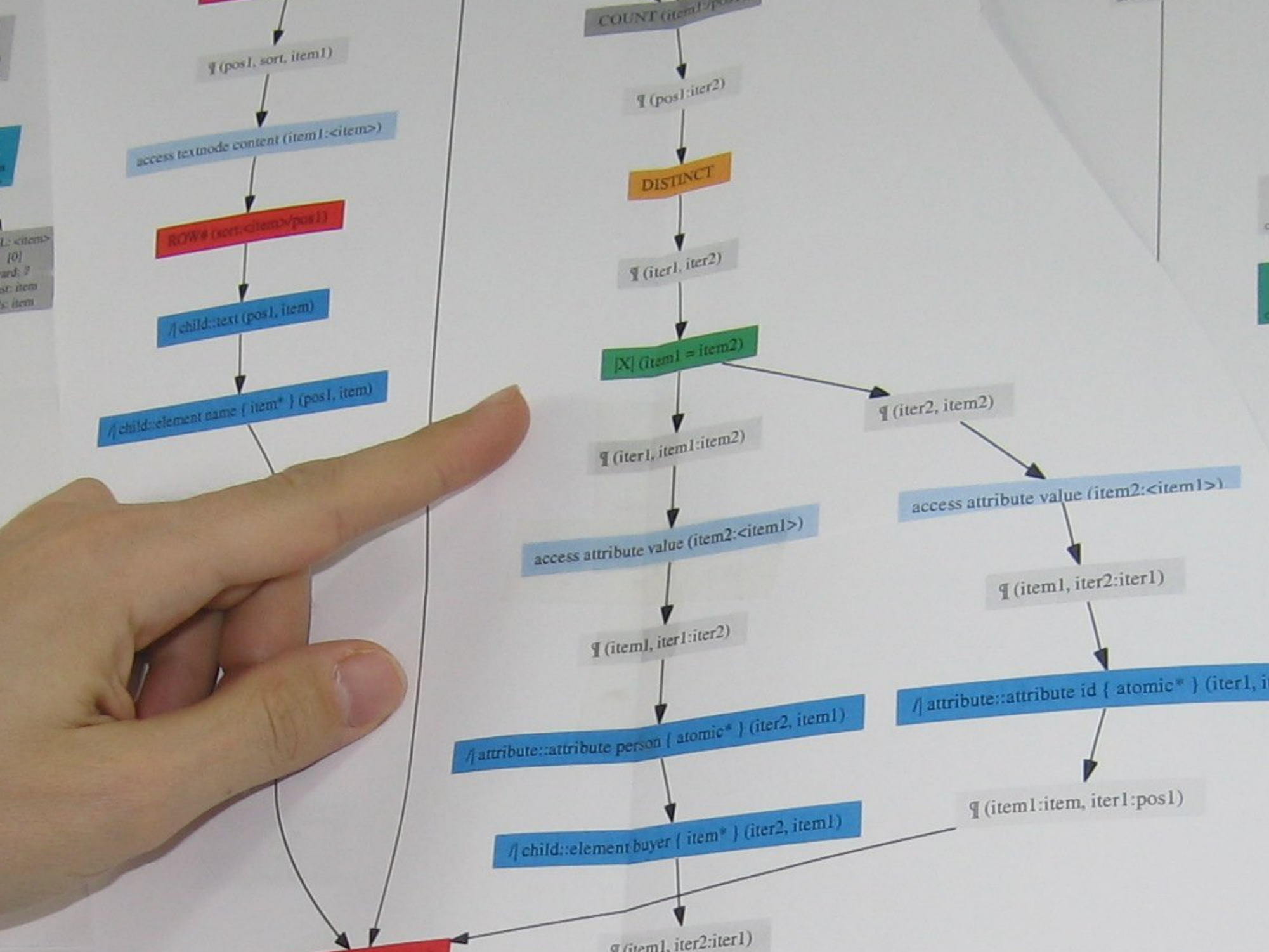
. . . and compiles them into plans of a standard relational algebra.

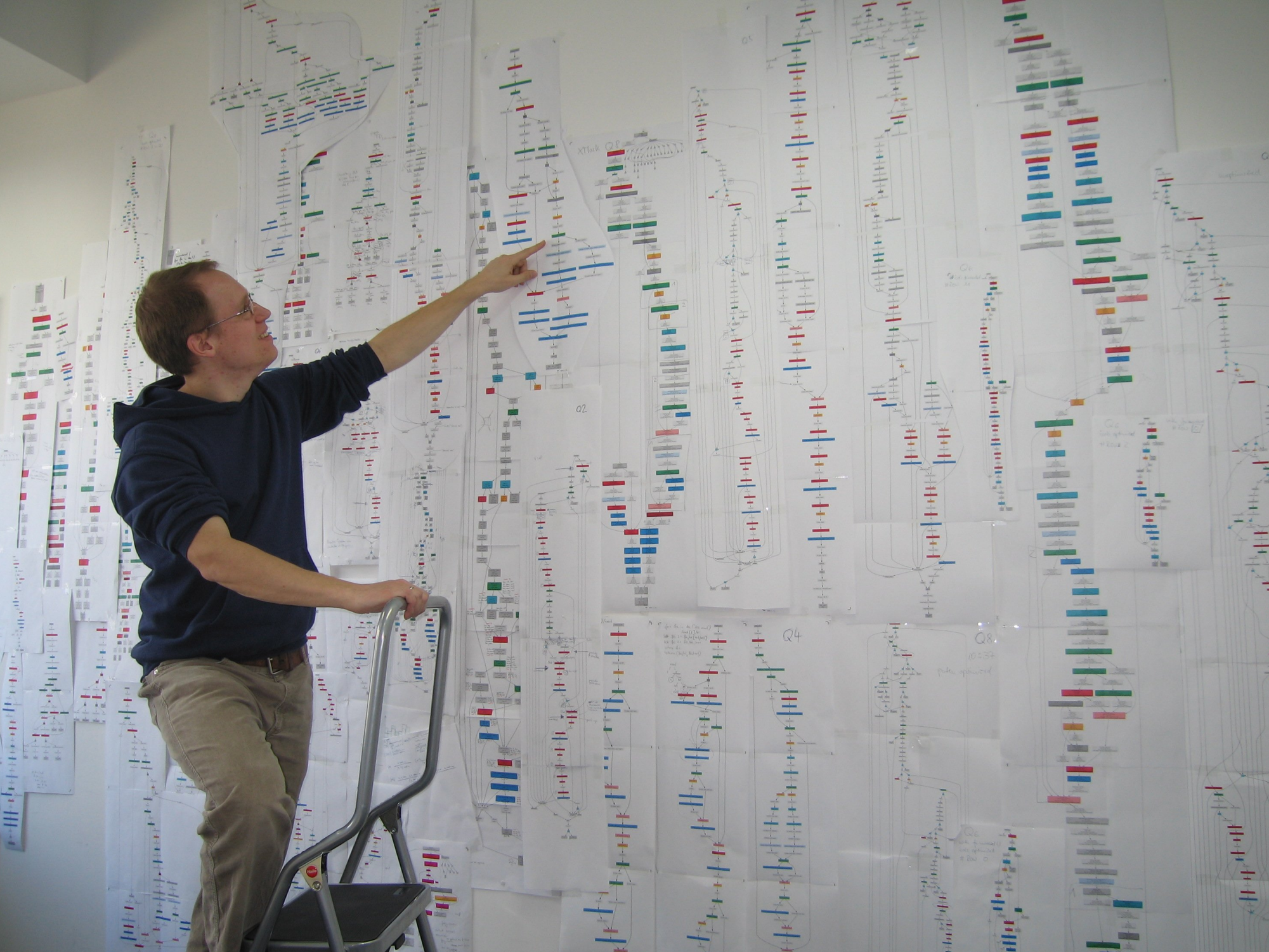
π	column projection, renaming
σ	row selection
$\bowtie$	equi-join
$\times$	Cartesian product
$\dot{\cup}$, $\setminus$	disjoint union, difference
δ	duplicate elimination
ϱ	row numbering
$\triangleright$	staircase join[1]
ε, τ	element/text node construction[1]
$\circledast$	arithmetic/comparison/Boolean operator $*$

EFFICIENTLY IMPLEMENTABLE ON SQL 99 HOSTS

- Operates on node (not tree!) level and 1NF relations.

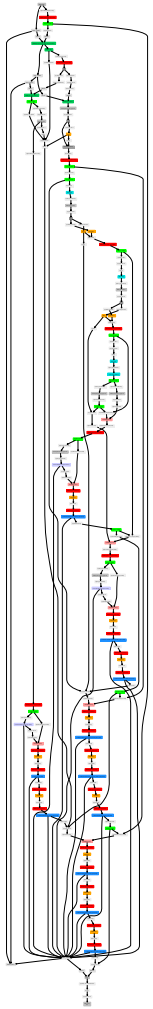
[1] Syntactic sugar; expressible by remaining operators.


¶ (pos1, sort, item1)
access textnode content (item1:<item>)
ROW# (sort:<item>/pos1)
/| child::text (pos1, item)
/| child::element name { item* } (pos1, item)
COUNT (item1/pos1)
¶ (pos1:iter2)
DISTINCT
¶ (iter1, iter2)
[X] (item1 = item2)
¶ (iter1, item1:item2)
access attribute value (item2:<item1>)
¶ (item1, iter1:iter2)
/| attribute::attribute person { atomic* } (iter2, item1)
/| child::element buyer { item* } (iter2, item1)
¶ (iter2, item2)
access attribute value (item2:<item1>)
¶ (item1, iter2:iter1)
/| attribute::attribute id { atomic* } (iter1, i
¶ (item1:item, iter1:pos1)
¶ (item1, iter2:iter1)


Q2
Q4
Q8

Relational Query Optimization for XQuery

To combat the resulting plan sizes, Pathfinder uses



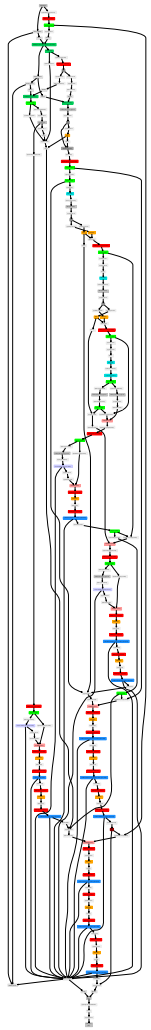
XMark Q8 · 34 MB · Intel Xeon 3.2 GHz · 8 GB RAM

Resulting query runtime (MonetDB/XQuery):

Relational Query Optimization for XQuery

To combat the resulting plan sizes, Pathfinder uses

1. **constant propagation**,



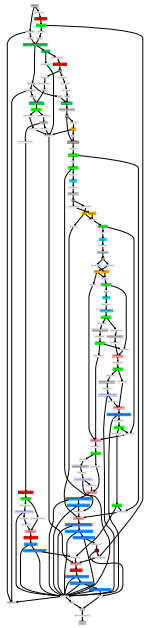
XMark Q8 · 34 MB · Intel Xeon 3.2 GHz · 8 GB RAM

Resulting query runtime (MonetDB/XQuery):

Relational Query Optimization for XQuery

To combat the resulting plan sizes, Pathfinder uses

1. **constant propagation**,
2. **projection pushdown**,



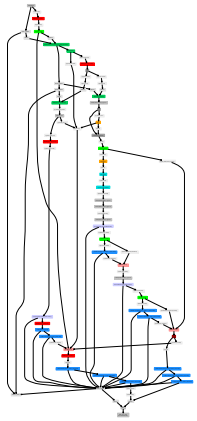
XMark Q8 · 34 MB · Intel Xeon 3.2 GHz · 8 GB RAM

Resulting query runtime (MonetDB/XQuery):

Relational Query Optimization for XQuery

To combat the resulting plan sizes, Pathfinder uses

1. **constant propagation**,
2. **projection pushdown**,
3. **functional dependency** and **data flow analyses**, and



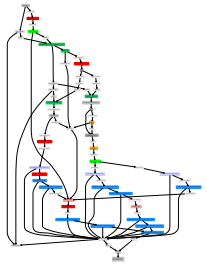
XMark Q8 · 34 MB · Intel Xeon 3.2 GHz · 8 GB RAM

Resulting query runtime (MonetDB/XQuery):

Relational Query Optimization for XQuery

To combat the resulting plan sizes, Pathfinder uses

1. **constant propagation**,
2. **projection pushdown**,
3. **functional dependency** and **data flow analyses**, and
4. algebraic **join detection**.



XMark Q8 · 34 MB · Intel Xeon 3.2 GHz · 8 GB RAM

Resulting query runtime (MonetDB/XQuery):

Get your copy of Pathfinder today:

`http://www.pathfinder-xquery.org/`

Pathfinder is developed at TU München by

- **Torsten Grust, Jan Rittinger, and Jens Teubner**.

Thanks to our long-term partners

- Peter Boncz, CWI Amsterdam
- Jan Flokstra, U Twente
- Maurice van Keulen, U Twente
- Martin Kersten, CWI Amsterdam
- Stefan Manegold, CWI Amsterdam
- Sjoerd Mullender, CWI Amsterdam
- Ying Zhang, CWI Amsterdam

Pathfinder is supported by the German Research Foundation DFG.