Dependable Cardinality Forecasts for XQuery

Jens Teubner, ETH (formerly IBM Research)
Torsten Grust, U Tübingen (formerly TUM)
Sebastian Maneth, UNSW and NICTA
Sherif Sakr, UNSW and NICTA

Cardinality Estimation for XQuery

The feature-richness and semantics of the language make cardinality estimation for XQuery notoriously hard.

```
for $d in doc("forecast.xml")/descendant::day
let $day := $d/@t
let $ppcp := data($d/descendant::ppcp)
return
  if ($ppcp > 50)
    then ("rain likely on", $day,
          "chance of precipitation:", $ppcp)
    else ("no rain on", $day)
```

- `for` iteration
- conditionals (`if-then-else`)
- existential quantification
- sequence construction
- ...

(XPath is **not** a focus of this work.)

Cardinality Estimation for XQuery

The feature-richness and semantics of the language make cardinality estimation for XQuery notoriously hard.

```
for $d in doc("forecast.xml")/descendant::day
let $day := $d/@t
let $ppcp := data($d/descendant::ppcp)
return
  if ($ppcp > 50)
    then ("rain likely on", $day,
          "chance of precipitation:", $ppcp)
    else ("no rain on", $day)
```

$card_1 = ?$

$card_2 = ?$

$card_3 = ?$

$card_4 = ?$

- `for` iteration
- conditionals (`if-then-else`)
- existential quantification
- sequence construction
- …

(XPath is **not** a focus of this work.)

→ **Goal:** Compute subexpression-level cardinalities $card_i$.

Idea: Perform cardinality estimation on relational plan equivalents for XQuery.

relational cardinality estimation (System R)
\+ existing work on XPath estimation
\+ histograms for value predicates

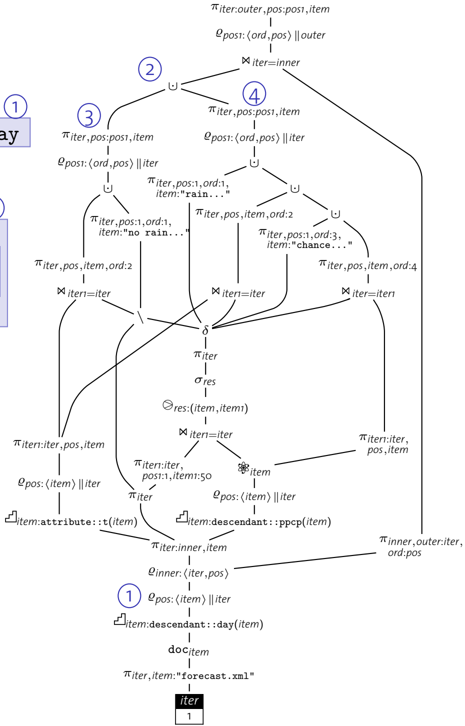
= cardinality estimation for XQuery

- Build on Pathfinder's XQuery-to-relational algebra compiler.[1]
  - $\longrightarrow$ tuple count $\equiv$ XQuery item count
- Cardinality information for **each** subexpression.

[1] http://www.pathfinder-xquery.org/

Example (Plan details not of interest today.)

```
for $d in doc("forecast.xml")/descendant::day
let $day := $d/@t
let $ppcp := data($d/descendant::ppcp)
return
  if ($ppcp > 50)
    then ("rain likely on", $day,
          "chance of precipitation:", $ppcp)
    else ("no rain on", $day)
```



- Pathfinder compiles XQuery with arbitrary nesting.
- Maintain correspondence to original query if back-end is not relational.
- Derive estimates based on an **inference rule set**.

Relational XQuery Cardinality Estimation

Apply System R-style estimation to relational XQuery plans, *e.g.*,

Disjoint union:

$$|q_1 \cupdot q_2| = |q_1| + |q_2|$$

Cartesian product:

$$|q_1 \times q_2| = |q_1| \cdot |q_2|$$

Equi-join:

$$|q_1 \underset{a=b}{\bowtie} q_2| = \begin{cases} \dfrac{|q_1| \cdot |q_2|}{\max\{|a|_{\text{idx}}, |b|_{\text{idx}}\}} & \text{if there are indexes on both join columns,} \\ \dfrac{|q_1| \cdot |q_2|}{|a|_{\text{idx}}} & \text{if there is only an index on column } a, \\ |q_1| \cdot |q_2| \cdot 1/10 & \text{otherwise} \end{cases}$$

$|c|_{\text{idx}}$: Number of unique values in index on column *c*.

Relational XQuery Cardinality Estimation

Apply System R-style estimation to relational XQuery plans, *e.g.*,

Disjoint union:

$$|q_1 \mathbin{\dot\cup} q_2| = |q_1| + |q_2|$$

Cartesian product:

$$|q_1 \times q_2| = |q_1| \cdot |q_2|$$

Equi-join:

$$|q_1 \underset{a=b}{\bowtie} q_2| = \begin{cases} \dfrac{|q_1| \cdot |q_2|}{\max\{|a|_{\text{idx}}, |b|_{\text{idx}}\}} & \text{if there are indexes on both join columns,} \quad ? \\ \dfrac{|q_1| \cdot |q_2|}{|a|_{\text{idx}}} & \text{if there is only an index on column } a, \quad ? \\ |q_1| \cdot |q_2| \cdot 1/10 & \text{otherwise} \end{cases}$$

- Our joins typically operate over **computed** relations.

$|c|_{\text{idx}}$: Number of unique values in index on column c.

Abstract Domain Identifiers

A simple form of **data flow analysis** provides the information needed.

- Introduce **abstract domain identifiers** $\alpha, \beta, \ldots$ as placeholders for the **active runtime domain** for each column *c*.
(Read c^α as "column *c* contains values from domain α.")
- Estimate the **size** $\|\alpha\|$ of each domain α, *e.g.*,[2]

$$dom\left(\varrho_{a:\langle b_1,\ldots,b_n\rangle}(q)\right) \supseteq dom\,(q) \cup \left\{a^\alpha \wedge \|\alpha\| \stackrel{!}{=} |q|\right\} \quad .$$

- Identify **inclusion relationships** $\alpha \sqsubseteq \beta$ between domains, *e.g.*,

$$a^\alpha \in dom\,(q) \wedge a^\beta \in dom\,(\sigma_{\ldots}(q)) \Rightarrow \beta \sqsubseteq \alpha \quad .$$

[2] Operator $\varrho_{a:\langle b_1,\ldots,b_n\rangle}$ introduces a new key column (holding row numbers).

Abstract Domain Identifiers

Use abstract domain information for cardinality estimation.

E.g., "foreign key" join:

$$\frac{a^{\alpha} \in dom(q_1) \qquad b^{\beta} \in dom(q_2) \qquad \alpha \sqsubseteq \beta}{|q_1 \underset{a=b}{\bowtie} q_2| = \dfrac{|q_1| \cdot |q_2|}{\|\beta\|}}$$

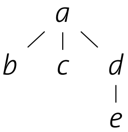
- Domain inclusion guarantees that each tuple in q_1 finds (at least one) join partner in q_2.

Other examples:

- $|q_1 \setminus q_2| = |q_1| - |q_2|$ if q_2 is a subset of q_1.
- $|q_1 \setminus q_2| = 0$ if q_1 is a subset of q_2.
- $|q_1 \setminus q_2| = |q_1|$ if q_1 and q_2 are disjoint.

Interfacing with XPath—Projection Paths

Track XPath navigation by means of **projection paths**[3]



a	b
1	γ_a
2	γ_b
3	γ_d

q_1

$$\xrightarrow{⎍_{c:\texttt{child::*}(b)}}$$

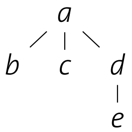
a	b	c
1	γ_a	γ_b
1	γ_a	γ_c
1	γ_a	γ_d
3	γ_d	γ_e

q_2

- $b^{\Rightarrow p} \in path\,(q_1) \quad \Rightarrow \quad c^{\Rightarrow p/\texttt{child::*}} \in path\,(q_2)$
- Step operator ⎍ makes XPath navigation explicit in relational plans (compiles to join on SQL back-ends).

[3] A. Marian and J. Siméon. Projecting XML Documents. *VLDB 2003*.

Interfacing with XPath—Cardinality Inference



b^{α} ↓

a	b
1	γ_a
2	γ_b
3	γ_d

q_1

$\xrightarrow{\quad c{:}\texttt{child::*}(b) \quad}$

$b^{\alpha'}$ ↓

a	b	c
1	γ_a	γ_b
1	γ_a	γ_c
1	γ_a	γ_d
3	γ_d	γ_e

q_2

$\alpha' \sqsubseteq \alpha$

Cardinality:

$$|q_2| = |q_1| \cdot \frac{\texttt{fn:count}\,(p/\texttt{child::*})}{\texttt{fn:count}\,(p)} = |q_1| \cdot \underbrace{\Pr_{\texttt{child::*}}(p)}_{\textbf{fanout}}$$

fanout (here: 4/3)

Domain Sizes:

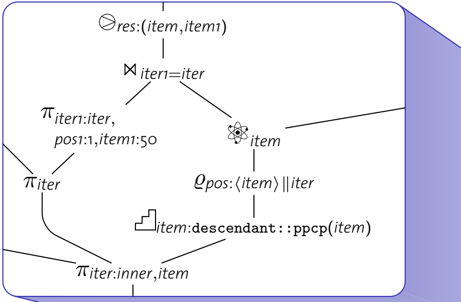
$$\|\alpha'\| = \|\alpha\| \cdot \frac{\texttt{fn:count}\,(p\,[\,\texttt{child::*}\,])}{\texttt{fn:count}\,(p)} = \|\alpha\| \cdot \underbrace{\Pr_{[\texttt{child::*}]}(p)}_{\textbf{selectivity}}$$

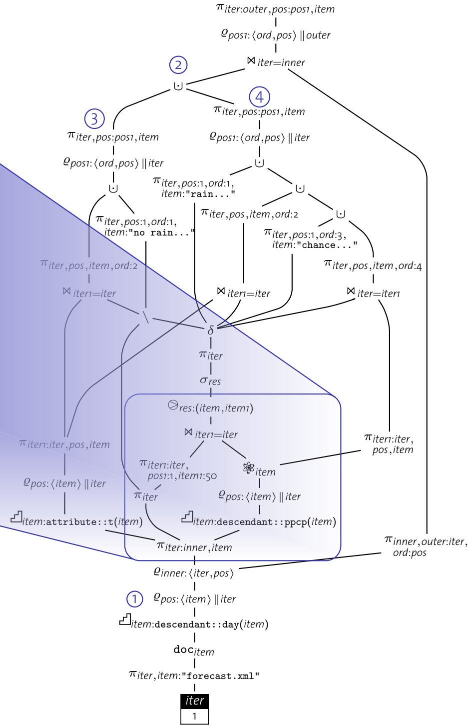
selectivity (here: 2/3)

Any XPath estimator that provides $\Pr_{p_2}(p_1)$ and $\Pr_{[p_2]}(p_1)$ will do.

- Our prototype uses a simple Data Guide-based implementation.

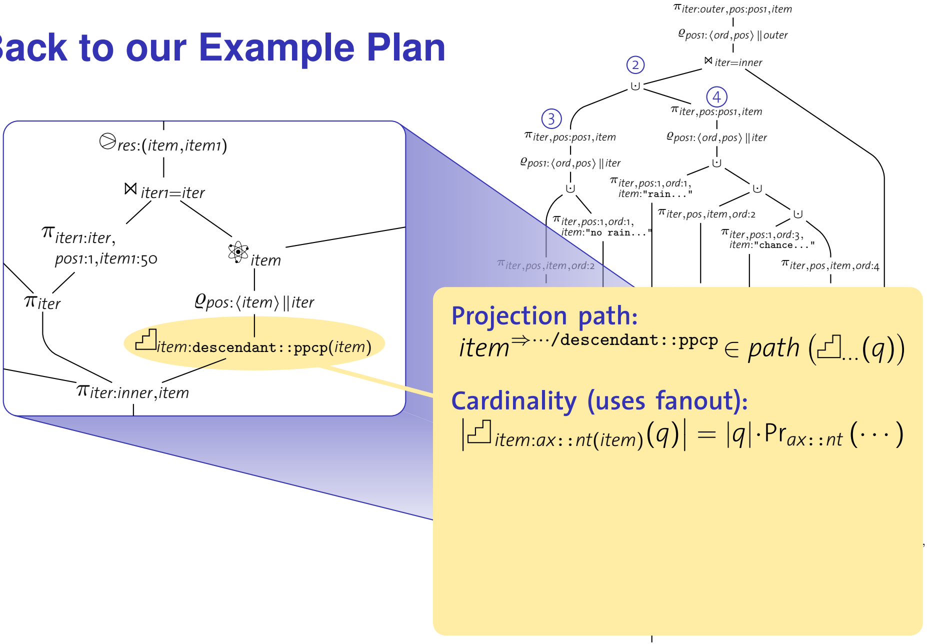
Back to our Example Plan





$⚛_a$: Retrieve **typed values** for node identifiers in column *a* (atomization).

Back to our Example Plan



Projection path:

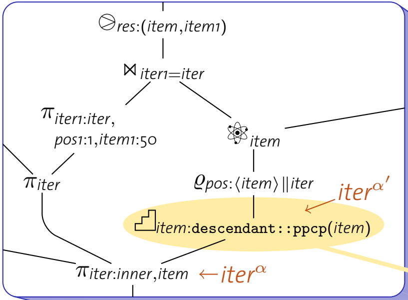
$$item^{\Rightarrow \cdots/\texttt{descendant::ppcp}} \in path\,(\ldots(q))$$

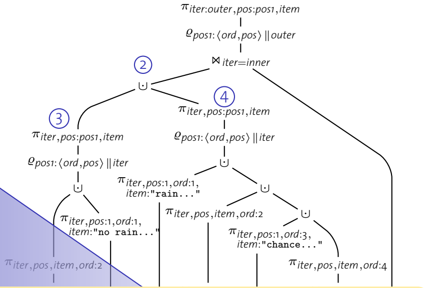
Cardinality (uses fanout):

$$\left|\,_{item:ax::nt(item)}(q)\right| = |q| \cdot \Pr_{ax::nt}(\cdots)$$

$_a$: Retrieve **typed values** for node identifiers in column a (atomization).

Back to our Example Plan





Projection path:

$$item^{\Rightarrow \cdots/\texttt{descendant::ppcp}} \in path\,(\sqcup_{\ldots}(q))$$

Cardinality (uses fanout):

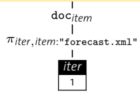
$$\left|\sqcup_{item:ax::nt(item)}(q)\right| = |q| \cdot \Pr_{ax::nt}(\cdots)$$

Domain inclusion:

$$iter^{\alpha'} \in dom\,(\sqcup_{\ldots}(q));\ \alpha' \sqsubseteq \alpha$$

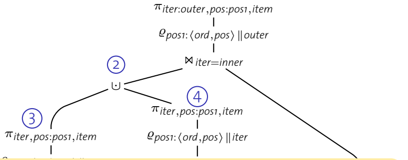
Domain size (uses step selectivity):

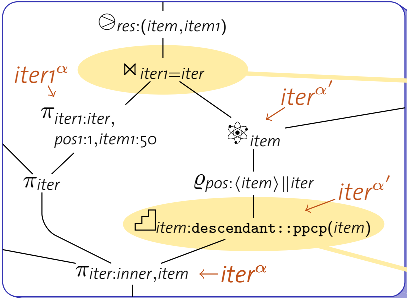
$$\|\alpha'\| = \|\alpha\| \cdot \Pr_{[ax::nt]}(\cdots)$$



$_a$: Retrieve **typed values** for node identifiers in column *a* (atomization).

Back to our Example Plan





"Foreign key" join:

$$\left|q_1 \underset{iter1=iter}{\bowtie} q_2\right| = \frac{|q_1|\cdot|q_2|}{\|\alpha\|} \quad (\text{since } \alpha' \sqsubseteq \alpha)$$

Projection path:

$$item \Rightarrow \cdots\texttt{/descendant::ppcp} \in path\left(\text{⏋}_{\ldots}(q)\right)$$

Cardinality (uses fanout):

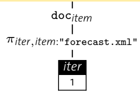
$$\left|\text{⏋}_{item:ax::nt(item)}(q)\right| = |q| \cdot \Pr_{ax::nt}(\cdots)$$

Domain inclusion:

$$iter^{\alpha'} \in dom\left(\text{⏋}_{\ldots}(q)\right);\ \alpha' \sqsubseteq \alpha$$

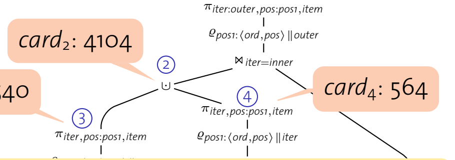
Domain size (uses step selectivity):

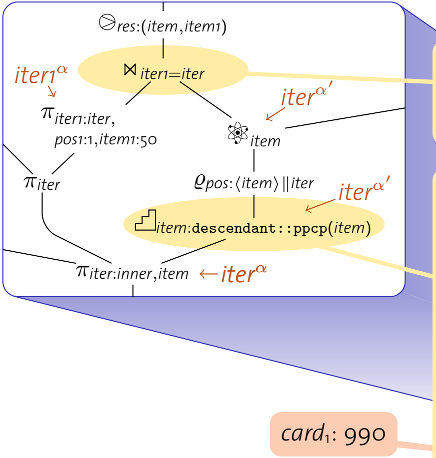
$$\|\alpha'\| = \|\alpha\| \cdot \Pr_{[ax::nt]}(\cdots)$$



$_a$: Retrieve **typed values** for node identifiers in column a (atomization).

Back to our Example Plan





"Foreign key" join:

$$\Bigl|q_1 \underset{iter1=iter}{\bowtie} q_2\Bigr| = \frac{|q_1|\cdot|q_2|}{\|\alpha\|} \quad (\text{since } \alpha' \sqsubseteq \alpha)$$

Projection path:

$$item \Rightarrow \cdots/\texttt{descendant::ppcp} \in path\,(\text{⏍}_{\ldots}(q))$$

Cardinality (uses fanout):

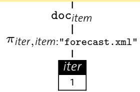
$$\bigl|\text{⏍}_{item:ax::nt(item)}(q)\bigr| = |q|\cdot \Pr_{ax::nt}(\cdots)$$

Domain inclusion:

$$iter^{\alpha'} \in dom\,(\text{⏍}_{\ldots}(q));\ \alpha' \sqsubseteq \alpha$$

Domain size (uses step selectivity):

$$\|\alpha'\| = \|\alpha\| \cdot \Pr_{[ax::nt]}(\cdots)$$



$\circledast_a$: Retrieve **typed values** for node identifiers in column *a* (atomization).

Forecasting New Zealand's Weather

The obtained cardinalities can be mapped back to predict item counts for corresponding XQuery expressions:[4]

```
for $d in doc("forecast.xml")/descendant::day
let $day := $d/@t
let $ppcp := data($d/descendant::ppcp)
return
  if ($ppcp > 50)
    then ("rain likely on", $day,
           "chance of precipitation:", $ppcp)
    else ("no rain on", $day)
```

$card_1 = 990/990$

$card_2 = 4104/3402$

$card_3 = 3540/2370$

$card_4 = 564/1032$

- Value statistics based on histograms (↗ paper)
- Inaccuracy is mainly due to correlations in the data.
  - Rain in the morning likely means rain in the afternoon, too.

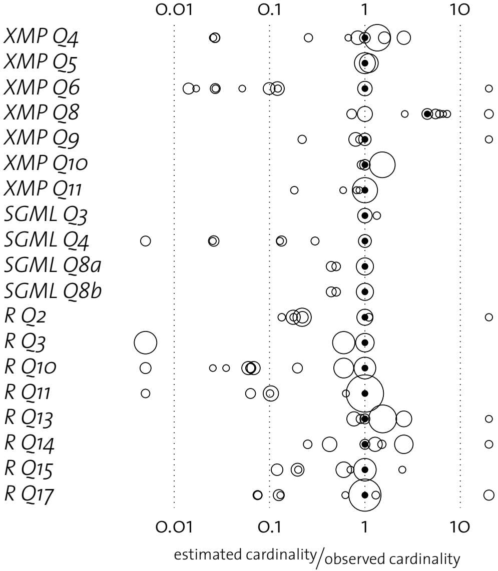
[4] estimated/observed, based on data taken for New Zealand two weeks ago.

More Realistic Queries: W3C XQuery Use Cases

- Prototype implementation based on Pathfinder
- For each subexpression:

$$\frac{\text{estimated cardinality}}{\text{observed cardinality}}$$

- Diameter indicates data point "stacking"
- Plan root: filled circle •
- Applicable to "real" queries
- Recovery from intermediate mis-estimations
    - *e.g.*, existential semantics



Wrap-Up

- Cardinality estimation framework for XQuery
- **Subexpression-level** estimates for **arbitrary** XQuery expressions
- Based on Pathfinder's XQuery-to-relational algebra compiler

relational cardinality estimation (System R)
\+ existing work on XPath estimation
\+ histograms for value predicates

= cardinality estimation for XQuery

- High-quality estimates for realistic XQuery workloads
  - **Robust** with respect to intermediate errors
- **Pluggable** and **extensible**
  - *e.g.*, XPath estimation subsystem, positional predicates