StatiX: Making XML Count

Juliana Freire*  
Prasan Roy  
Jerome Simeon  
Bell Labs - Lucent Technologies

Jayant Haritsa  
Maya Ramanath  
Indian Institute of Science
Motivation

- Statistics to estimate cardinality of queries
  - Query optimization
  - Exploratory queries
  - Cost-based storage design for XML data

- Relational
  - Fixed and flat structure
  - Need information about distribution of values

- XML
  - Flexible and nested structure
  - Need information about both values and structure
Example: IMDB Schema

```haskell
type Show =
    show [ title [String],
         year [Integer],
         review [String] *,
         Aka *,
         ( box_office [Integer] |
           ( season [Integer],
             Episode * ) ) ]

type Aka = aka [String]

type Episode = episode [ Aka {0,2},
                         guest_dir [String]? ]
```

![Diagram of the IMDB schema]
IMDB Sample Data

<show>
<title> Fugitive, The </title>
<year> 1993 </year>
<review>
    best action movie of the decade...
</review>
<review>
    Ford and Jones at their best...
</review>
<review>
    top notch action thriller...
</review>
<review>
    Solid action and great suspense...
</review>
<aka> Auf der Flucht </aka>
<br>box office> 183752965 </box office>
</show>

<show>
<title> Seinfeld </title>
<year> 1990 </year>
<review>
    The best comedy series ever!
</review>
<seasons> 9 </season>
<episode>
    <aka> The Soup Nazi </aka>
    <aka> The Soup </aka>
</episode>
<episode>
    <aka> Good Samaritan, The </aka>
    <guest_director> Alexander, Jason </guest_director>
</episode>
<episode>
    <aka> Gum, The </aka>
</episode>
</show>
Selectivity of XML Queries

List all akas for show
/show/aka

List all akas for episodes
/show/episode/aka

List all akas and reviews for movies released after 1991
FOR $s in
document ("imdb.xml")/show
$a in $s/aka, $r in $s/review
WHERE $s/year > 1991
RETURN $s/title, $s/box_office, $a, $r

- How to **concisely** capture structural and value skew?
- How to generate **accurate** estimates?

Capture structure of tags

Capture distribution of values over structure
Related Work

- McHugh & Widom, VLDB 1999: all subpaths of length up to \( k \)
  - For “longer” queries, combine info about subpaths

- Chen et al, ICDE 2001: capture correlation between paths
  - Use set hash signatures to represent paths
  - Compute selectivity by combining signatures

- Aboulnaga et al, VLDB 2001: compress path information
  - Focus: small summaries
  - Coalesce/delete elements with low frequency

- Polyzotis & Garofalakis, SIGMOD 2002: estimates for graph-structured data

- Wu et al, EDBT 2002: use histograms to capture ancestor/descendent relations
## Related Work

<table>
<thead>
<tr>
<th></th>
<th>Summary</th>
<th>Stats</th>
<th>Supported Queries</th>
<th>Estimate computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen et al ICDE 2001</td>
<td>Data</td>
<td>Paths + correlations</td>
<td>Tree pattern, no values</td>
<td>Specialized</td>
</tr>
<tr>
<td>Aboulnaga et al VLDB 2001</td>
<td>Data</td>
<td>Paths</td>
<td>Simple path, no values</td>
<td>Specialized</td>
</tr>
<tr>
<td>StatiX</td>
<td>Schema + Data</td>
<td>Schema types</td>
<td>Tree pattern, values</td>
<td>Classical histogram multiplication</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Related Work

- Overview
- Architecture
- Gathering statistics
- Transformations: flexible granularity
- Applying StatiX to XML storage
- Experiments
- Summary and Future Work
The StatiX Framework

- **Schema-based stats**
  - Use XML Schema to identify sources of skew

- **Gather stats about types**
  - Piggyback on validation
  - Concise summaries
  - Flexible granularity

- **Histograms for summarizing information about distribution of values and structure**
  - Well-studied and effective
  - Scalable: adjust to memory budgets
  - Symmetric mechanism for capturing structural and value skews
  - Easy to migrate functionality into relational backend

- **Queries**
  - Tree patterns, value-based selection and joins
Architecture

StatiX

XML Schema

Transformations

Schema validator

XML Schema

Control
granularity

StatiX Summary

Collect
stats

XML Data
Validation and Statistics Collection

- **Validation**
  - Check document against schema
  - Result: assignment of types to nodes
- **Extend validation for statistics collection**
- **Count occurrences of types and their parents**
  - Assign a unique id to each type
  - Keep one counter and a parent set per type
  - For each type instance
    - Assign a global id: id.counter
    - Add the id of the parent to the parent set
## Gathering Statistics

### StatiX Summary

**Show**: card = 2, id = [11, 12]

- parent_histogram:
  - [11, 12] = 4, [31] = 1

**Aka**: card = 4, id = [21, 25]

- value_histogram:
  - [1979, 2002] = 2

**Year**: card = 2, id = [41, 42]

- parent_histogram:
  - [11, 12] = 2

<table>
<thead>
<tr>
<th>Type</th>
<th>ID</th>
<th>Parent Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Aka</td>
<td>2</td>
<td>1.1 {2}, 1.2 {2}, 3.1 {1}</td>
</tr>
<tr>
<td>Episode</td>
<td>3</td>
<td>1.2 {1}</td>
</tr>
<tr>
<td>Year</td>
<td>4</td>
<td>1.1 {1}, 1.2 {1}, 1979 {1}, 2002 {1}</td>
</tr>
</tbody>
</table>

### Summarization

Statix - SIGMOD, 2002

Juliana Freire
Estimate Computation

FOR $s$ in
document ("imdb.xml")/show,
$a$ in $s$/aka, $r$ in $s$/review
WHERE $s$/year > 1991
RETURN $s$/title, $a$, $r$

SELECT title, aka, review
FROM Show, Aka, Review
WHERE Show.year > 1991 AND
  Show.ID = Aka.ParID AND
  Show.ID = Review.ParID

Histogram multiplication
Architecture

StatiX

XML Schema

Schema Transformations

XML Schema

Schema validator

XML Data

Control granularity

StatiX Summary
A given document can be validated by different XML Schemas
- Different but equivalent regular expressions can be used to define an element
- The presence or absence of a type name does not change the semantics of an XML Schema

Control stat collection by:
- Creating type names for potential sources of skew, i.e., Unions, repetitions, optional elements
- Additional transformations (LegoDB, ICDE 2002)
Schema Transformations

```haskell
type Show = show [ title [String], year [Integer],
  review[String]*, Aka *,
  ( box_office [Integer] |
    ( season [Integer], Episode * ) ) ]

type Aka = aka [String]

type Episode = episode [ Aka{0,2},
  guest_dir [String]? ]

type Review = review[String]
...
```

Distribution of reviews over shows
More Transformations...

type Show = show [title [String], year [Integer],
Review* , Aka *, 
(Movie|TVShow ) ]

type Aka = aka [String]

type Episode = episode [ Aka{0,2},
GuestDirector? ]

type Review = review[String]
...

type Show1 = show [title [String], year [Integer],
Review*, Aka *, Movie ]

distribution of reviews over shows that are movies

type Show2 = show [title [String], year [Integer],
Review*, Aka *, TVShow ]

distribution of reviews over shows that are tv shows

type Aka = aka [String]

type Episode = episode [ Aka{0,2},
GuestDirector? ]

type Review = review[String]

union distribution
Summary Sizes

\[ |\text{summary}| = \sum_i (\text{ctypes}_i \times \text{nbuckets}_i \times \text{bsize}_i) + \sum_j (\text{btypes}_j \times \text{nbuckets}_j \times \text{bsize}_j) \]

- Size is proportional to:
  - Number of incoming edges into types (ctypes)
  - Number of basic types (btypes)
  - Number of buckets

- **100MB XMark, 30 buckets/hist**
  - 125 types: 25KB
  - 1887 types: 200KB (finest grain)

- Limit the number of types during transformation
- Limit the number of buckets
Applying StatiX
StatiX and Cost-Based Storage Design

- LegoDB (ICDE2002): cost-based XML-to-relational mapping
- Given a schema, query workload, and data sample
  - Explore a set of alternative mappings
  - Select the mapping that leads to lowest cost
- Not practical to gather stats for each configuration
  - Need to derive precise stats for each configuration
- Use StatiX to generate fine-grained stats
Experiments
Statistics Collection: Overheads

- Stat gathering overhead depends on number of types
- Overheads can be reduced
  - Tune code + Sampling
- One-time only procedure

125 types

1887 types
Accuracy: XMark

- Queries involving value-based and structure-based joins (2-8 joins)
- Uniform distribution leads to large errors
- Chen et al. using a summary twice as big lead to 53% error for X7
Accuracy: IMDB

![Bar chart showing result size for IMDB queries Q1 and Q2 for different methods: Actual, Uniform, Statix (Struct), StatiX (Struct + Val).]
Summary and Future Work

◆ StatiX: leverage XML Schema data model, type transformations, and histograms to generate **concise, flexible, and accurate** summaries of XML documents

◆ Differentiators:
  - Handles larger class of queries
  - Localized impact of updates
  - Constructive approach
  - Easy to integrate into relational backend

◆ Future:
  - Support for recursion and aggregates
  - Ambiguity in validation (union distribution)