High Performance XML Data Retrieval

Mark V. Scardina  
Group Product Manager & XML Evangelist  
Oracle Corporation

Jinyu Wang  
Senior Product Manager  
Oracle Corporation
Agenda

- Why XPath for Data Retrieval?
- Current XML Data Retrieval Strategies and Issues
- High Performance XPath Requirements
- Design of Extractor for XPath
- Extractor Use Cases
Why XPath for Data Retrieval?

- W3C Standard for XML Document Navigation since 2001
- Support for XML Schema Data Types in 2.0
- Support for Functions and Operators in 2.0
- Underlies XSLT, XQuery, DOM, XForms, XPointer
Current Standards-based Data Retrieval Strategies

- Document Object Model (DOM) Parsing
- Simple API for XML Parsing (SAX)
- Java API for XML Parsing (JAXP)
- Streaming API for XML Parsing (StAX)
Data Retrieval Using DOM Parsing

- **Advantages**
  - Dynamic random access to entire document
  - Supports XPath 1.0

- **Disadvantages**
  - DOM In-memory footprint up to 10x doc size
  - No planned support for XPath 2.0
  - Redundant node traversals for multiple XPaths
DOM-based XPath Data Retrieval

1. /A/B/C
2. /A/B/C/D
Data Retrieval using SAX/StAX Parsing

• Advantages
  – Stream-based processing for managed memory
  – Broadcast events for multicasting (SAX)
  – Pull parsing model for ease of programming and control (StAX)

• Disadvantages
  – No maintenance of hierarchical structure
  – No XPath Support either 1.0 or 2.0
High Performance Requirements

- Retrieve XML data with managed memory resources
- Support for documents of all sizes
- Handle multiple XPaths with minimum node traversals
- Support DTD and Schema-based XML documents
Extractor for XPath

- Stream-based processing utilizing SAX
- Support for DTDs and XML Schemas
- Implements Publish/Subscribe model for scalability
- Handles multiple XPaths simultaneously
- Supports XPath 1.0; extended to 2.0
Extractor’s Publish/Subscribe Processing Model

Extractor

SAX

XPath Expressions & Content Handlers

Outputs
Extractor’s Function Blocks

- **Initialization**: registration of XPaths/Handlers
- **XPath Compilation**: compiles and builds index graphs
- **XPath Tracking**: maintains XPath state and matches doc XPaths with the indexed XPaths
- **Output**: sends matching XPath start/stop events along with the XML data
Extractor’s Function Blocks

**Initialization**
- Register XPath expressions and the instances of content handlers
- Compile XPath expressions and build Index graph
- Track XPath and perform XPath matching when receiving SAX events

**Output**
- XMLSequenceBuilder
- XMLSAXSerializer
- Registered Handler A
- Registered Handler Z

**XPath Expressions & Content Handlers**

**DTD/XSD**

**XML**

**SAX**

**XML Sequences**

**Files**
Initialization

- Registration of absolute XPaths
- Support for XML Namespaces for differentiation
- Registration of execution handlers using XContentHandler()
- Built-in Handlers for ease of use
  - XMLSequenceBuilder()
  - XMLSAXSerializer()
XPath Compilation

- XPath streamability evaluation
- Streamable isAll=true/false option
  - True: Process only streamable XPaths
  - False: Buffer data as needed
- Build XPath Predicate Table
- Build Index Tree
  - XPath Dependency Tree (w/o DTD/XSD)
  - Data Model Tree (w/ DTD/XSD) using existing Validation engine
Compilation of Each XPath Predicate

<table>
<thead>
<tr>
<th>Location XPath -1</th>
<th>Related to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate XPath -1</td>
<td>Value</td>
</tr>
<tr>
<td>Predicate XPath -2</td>
<td>Predicate XPath-3</td>
</tr>
<tr>
<td>Predicate XPath-4</td>
<td>Value</td>
</tr>
</tbody>
</table>

Predicate Table (not-matched, matched, yet-to-be-matched)

Also Used for isAll = True condition
Runtime XPath Matching

- State Machine tokenizes and tracks
  - In-scope Namespaces
  - Current Element Name
  - Current Element Attributes
  - Node Position Relative to Siblings
  - Number of Child Elements

- Implemented as a Stack
XPath Index Tree (w/o DTD/XSD)

XPath Expressions

1. /A/B/C
2. /A/B/C/D
3. /A/B/..P
4. /..P
5. /..P/..Q

XPath Dependency Tree
Dependency Tree Traversal

XPath Stack | Matched Node
---|---
P | X'' X'
D | X'' X'
C | X'' X'
B | X''
A | X''

XPath Stack:
1. /A/B/C
2. /A/B/C/D
3. /A/B///P
4. ///P
5. ///P/*/Q

XPath Dependency Tree:
- Fake Node for /*
- Fake Node for //

1. /A/B/C
2. /A/B/C/D
3. /A/B///P
4. ///P
5. ///P/*/Q
Synchronous Data Model Traversal
Extractor Output

- **XContentHandler()**
  - Execution of Registered Content Handlers
- **XMLSequenceBuilder()**
  - Built-in Handler
  - Presents Result Set as XMLSequence Object
  - Contains a Linked List of XMLItems
- **XMLSAXSerializer**
  - Built-in Handler
  - Serializes output to Printwriter or OutputStream
Extractor Use Cases

- Content Management
- Web-Services
- XSLT/XQuery Implementation
Extractor Content Management Use Case

SAX Parser

Pre-Process

XPath Expressions:
- /a/b
- /a/b/@c

Retrieve XPath Expressions

system/public ID or
XML Schema Location URL

DTD/XML Schema: XPaths Table

META Data Tables

Oracle Database

Connection Pool

Extractors

Registered Handler A

Registered Handler Y

Registered Handler Z

Process Data using Content Handlers

JDBC
Extractor Web Service Use Case

Web Services

Service Broker

Web Service Client

Register XPath/Content Handlers

SAX Parser

Extractor

"invoke" WS SOAP

WS response SOAP

Register Data Subscription using XPath

XML

Client 1

Client 2

Client 3

Clients Subscribing and Receiving the Data
Oracle XML Resources

Oracle Technology Network
- http://otn.oracle.com
- Downloads, Demos, Samples, Papers
- XML Support Forum

Oracle Database 10g XML & SQL
Design, Build, & Manage XML Applications in Java, C, C++, & PL/SQL
- Covers all of Oracle XML technology
- BetaBook Forum on OTN
- Available in May from Bookstores