EC 구축을 위한 효율적인 XML 데이터서버 기술
Efficient XML DataServer Technology
for EC Implementation

박시형
데어텍㈜ XML솔루션 사업부 부장
shpark@datec.co.kr
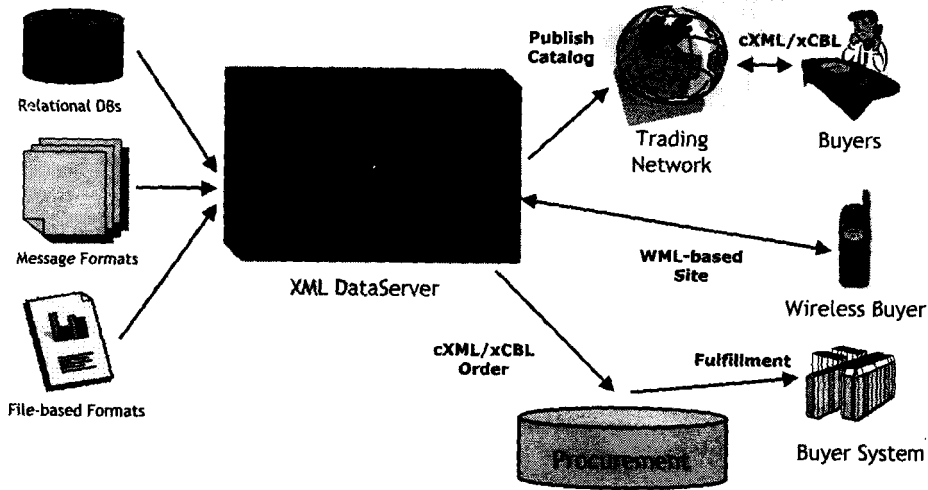
강성수
한국산업기술대학교 컴퓨터공학과 교수
kss@kpu.ac.kr

목차

1 XML Overview

* A sub-set of SGML
  (Standard Generalized Markup Language)

* A family of standards

* XML 1.0 is the specification that defines what tags,
  attributes and other XML nodes are

* License-free, platform-independent and well-
  supported

1.1 Why XML ?

* **Simple**
  ▶ Human-readable, easy to use

* **Extensible**
  ▶ A dynamic data model

* **Flexible**
  ▶ For handling complex data

* **Portable**
  ▶ For cross-platform data exchange

* **Standard**
  ▶ Easy to integrate, widely adopted

1.2 XML@E-Commerce

2. XML DataServer 기능

* Collecting all information as XML format
  ▶ Web page, ERP, RDB, File...

* Process XML data for EC
  ▶ Catalog process
  ▶ Order process
  ▶ Payment process
  ▶ Logistics process

* Exchange data between businesses
  ▶ Different Application (ERP, DB...)
  ▶ Different platform

2.1 Business Requirements for XML DataServer

* Ease of adoption
* Data integrity
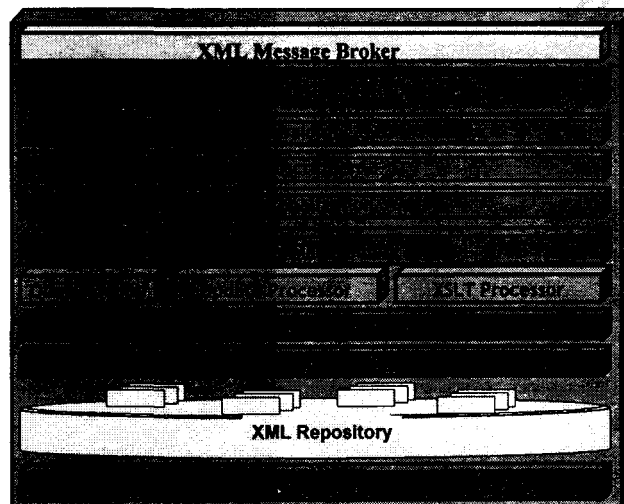* Integration of data from disparate sources
* Scalable performance

2.2 Technical Requirements for XML DataServer

* **XML Storage Engine**
    * Store XML data and other data format as well
    * Organize XML data to support W3C standards
    * Valid-formed and Well-formed document support
* **XML Update Interface**
    * Easy interface to update XML contents
* **XML Query Interface**
    * XPath Query, Index
* **XSL Translation (Format Conversion)**
    * XML to XML
    * XML to HTML (Browser independent)
    * XML to TEXT, XML to any format

2.2 Technical Requirements for XML DataServer

- **XML Cache**
  - ▶ Distribute process and data
  - ▶ Load balancing

- **XML Application Server**
  - ▶ Reduce process overhead
  - ▶ Scalable performance

- **Connectivity to Existing Systems**
  - ▶ RDB, ISAM, ASCII, Excel
  - ▶ ERP, Enterprise Applications

-9-

2.3 Components of the XML DataServer

-10-

3. XML Persistence

◆ **Efficiently storing arbitrary hierarchies like XML requires a different storage system whose requirements are :**

  ▶ Allows granular access to XML element

  ▶ Supports extensibility

  ▶ Provides indexing for fast retrieval

  ▶ Provides in-memory performance (fast XML caches)

  ▶ Supports XML standard APIs

  ▶ Support both structured and semi-structured data

3.1 XML Persistence Solutions : Flat Files

◆ **Solution**

  ▶ Store XML in flat files

◆ **Pros**

  ▶ Cheap

  ▶ Supports extensibility

  ▶ Supports semi-structured data

◆ **Cons**

  ▶ No data integrity

  ▶ No granular reuse of element across users

  ▶ Not scalable

3.2 XML Persistence Solutions : R-DB(Table)

- **Solution**
  - ▶ Map XML into table
- **Pros**
  - ▶ Relatively cheap (if you own the database already)
  - ▶ Data Integrity
  - ▶ Granular Access
- **Cons**
  - ▶ No (easy) extensibility
  - ▶ Need Schema in advance

3.3 XML Persistence Solutions : R-DB(BLOB)

- **Solution**
  - ▶ Store XML as BLOB(Binary Large Object)
  - ▶ Create full text index
- **Pros**
  - ▶ Supports extensibility
- **Cons**
  - ▶ No granular access/reuse
  - ▶ Data redundancy to make full text index
  - ▶ Bad concurrency on updates
  - ▶ Poor performance

3.4 XML Persistence Solutions : Native XML

* **Solution**
  * Store the XML parsed tree
* **Pros**
  * No need to re-parse
  * No need to reload the entire tree to manipulate it (DOM)
  * No joins, no mapping
  * Fully extensible
  * Supports both structured data and semi-structured data
  * Granular access
  * Data Integrity

---

4. Efficient XML DataServer for EC

* **W3C XML Standard Support**
  * DOM
  * XPath
  * XSL, XSLT
  * XML Update Language
  * XLink, XPointer ...

* **Valid-formed, and Well-formed XML Support**

* **Distributed Cache, Index**

* **Store Parsed Tree**