XML-Based Ship Data Modeling and its Application for Maritime Field

石，佳弘

(Degree)
博士（工学）

(Date of Degree)
2009-03-25

(Date of Publication)
2012-12-21

(Resource Type)
doctoral thesis

(Report Number)
甲4684

(URL)
https://hdl.handle.net/20.500.14094/D1004684

Doctoral Thesis

XML-Based Ship Data Modeling and Its Application for Maritime Field

（XML ベースによる船舶データモデルリング及び海事分野における応用）

January, 2009

Graduate School of Science and Technology

Kobe University

Chia-Hung SHIH

Acknowledgement

Four years have passed since I have started the initial research on this study. I could not complete this study without many supports during the period.

First, I want to give many special thanks to my Academic Supervisor, Dr. Nobukazu Wakabayashi, Graduate School of Maritime Sciences, Kobe University for his support, encouragement and patience throughout this entire Ph.D. process. Thank you for accepting me into this program and for helping me makes it through every gate. Also, I would like to give my thanks to the doctoral degree committee members, Dr. Shigeaki Shiotani, Dr. Hiroshi Ishida and Dr. Eiichi Kobayashi. Thank you for spending time reading my dissertation and giving me your valuable opinions to improve its quality helpfully.

I would like to express the appreciation to Dr. Saburo Yamamura, Graduate School of Maritime Sciences, Kobe University for participating in my committee and taking the time to read the dissertation and provide useful comments. Your direction throughout my studies and research has always been accurate and meaningful.

I would like to give my special thanks to Dr. Cheng-Tung Yang and Dr. Tso-Chung Sung, Department of Systems Engineering and Naval Architecture, National Taiwan Ocean University for making allowances to attend the Kobe University. In addition, the great experiences in the Computer Information Management Lab. allow me to make more progress. Also, I would like to give my appreciation to Dr. Deng-Fwu Hwang, Department of Food Science, National Taiwan Ocean University for encouraging me to attend the international exchange student between the Kobe University and the National Taiwan Ocean University.

I also truly appreciate for the very valuable assistance provided by Ms. Juri Harada, Mr. Kenichi Mori and other graduated members of our laboratory, Wakabayashi Lab., for their willingness, committed job and for supporting me in many situations.

My gratitude embraces a large number of friends who affectionately supported me nowhither you are in Japan, Taiwan or other countries for now. Especially Dr. Serdar KUM, Istanbul Technical University Maritime Faculty. Thank you for accompanying with me during the hardest time here. There will be always a place in my heart for each of you and I will always consider you as my friends.

Finally, I would like to include an acknowledgement for my family. I express my sincere admiration and love for their understanding and unique support for what they give me in all times, as well as their unwavering interest in my personal and professional improvement.

Abstract

Along with the evolution of digital generation, in order to raise the competitive advantage, every industry has to process certain computerization to improve its working efficiency. Maritime field also cannot be an exception in this tendency. Take the maritime field of Japan as an example; recently Japanese government had introduced EDI (Electronic Data Interchange) and Sea-NACCS Systems (Sea Nippon Automated Cargo Clearance System) to simplify the complexities of workflow. In the future, the unification of these systems will be processed to unify various ongoing services into one single window through the internet structure.

There are various kinds of the Internet structures, the most popularized mean nowadays is integrating Web interface with relational database to store information within central database, and then users can access and modify information through client-server mechanism. Relational database system has been popularized and implemented for many years because of their advantages. But different from overland industries for the maritime field, a vessel cannot access the Internet at anytime due to they are active units while sailing. Although this issue could be solved by satellite communication technology, however its cost still doesn’t reach a reasonable price for popularization.

Distributed information storage is a more reasonable mean compared with centralized data management for the maritime field. In contrast with traditional relational database, XML is the burgeoning data format for information storage and exchange recently. In this research, I focused on the tree structure and self-describable characteristic of XML and proposed to implement it through the maritime field. In data storage level, the property of data node is used to establish the data models for ships. Under the close intranet structure, XML is treated as metadata to establish context model for services. In data querying level, I proposed an efficient data restructuration mechanism to improve the redundancy of XML essence. The objective of this research is to propose a more suitable information model than relational database under the considerations of the singularities of maritime field.

Table of Contents

Acknowledgement ---- *I*

Abstract ---- *III*

Chapter 1 Introduction ---- *1*

Chapter 2 Background Concepts ---- *10*

2.1 Extensible Markup Language (XML) ---- **11**

2.1.1 XML Features ---- 12

2.1.2 Well-Formed XML Document ---- 13

2.1.3 Valid XML Document ---- 15

2.2 Processing XML ---- **19**

2.2.1 Simple API for XML (SAX) ---- 21

2.2.2 Document Object Model (DOM) ---- 23

2.3 XML Path Language (XPath) ---- **26**

2.4 Extensible Stylesheet Language (XSL) ---- **29**

2.5 Comparison of XML and Relational Database ---- **33**

Chapter 3 XML Data Modeling ---- *36*

3.1 Data Storage Method ---- **38**

3.1.1 Fleet Management Mode ---- 39

3.1.2 Ship Data Modeling ---- 40

3.2 Data Integration ---- **43**

3.2.1 Data Update During Sailing ---- 43

3.2.2 Synchronization when Arriving in Port ---- 45

3.3 Potential Implementation for Port Administration ---- **47**

Chapter 4 Context Model for Services ---- *51*

4.1 Web Service Life Cycle ---- **52**

4.2 System Architecture ---- **55**

4.2.1 Context Model ---- 57

4.2.2 User Model ---- 60

4.2.3 Providing Information Passively ---- 62

4.2.4 Providing Information Actively ---- 64

4.3 System Implementation within Passenger Ship Scenario ---- 67

Chapter 5 Efficient Data Structure ---- 72

5.1 Description of Container Operation ---- 73

5.2 XML Data Model for Container Operation ---- 82

5.2.1 Container XML Data Model Definition ---- 83

5.2.2 Restructuring Container XML Data Model ---- 85

5.3 Performance Evaluation ---- 89

Chapter 6 Research Contributions and Future Research ---- 102

References ---- 110

List of Published Papers ---- 121

List of Tables

Table 2-1 XPath Axes definition ---- 28

Table 4-1 Context model architecture ---- 58

Table 4-2 User model architecture ---- 60

Table 5-1 Desired container attributes ---- 83

Table 5-2 Index attribute candidates ---- 87

Table 5-3 Common XPath constructions ---- 89

Table 5-4 Sample queries for data sets ---- 92

Table 5-5 Characteristics of test datasets ---- 98

List of Figures

Figure 2-1 An example of XML document structure ---- *11*

Figure 2-2 A DTD sample ---- *16*

Figure 2-3 A sample for XML Schema ---- *18*

Figure 2-4 DOM and SAX parsers ---- *20*

Figure 2-5 A sample for SAX ---- *22*

Figure 2-6 An XML node-tree sample ---- *23*

Figure 2-7 hierarchical relationship of XML nodes ---- *25*

Figure 2-8 XPath evaluation model for a single location step ---- *26*

Figure 2-9 Transformation from XML to XSL Formatting Semantics ---- *30*

Figure 2-10 An XSL transformation sample ---- *32*

Figure 3-1 Separable XML tree architecture ---- *39*

Figure 3-2 Ship data model ---- *41*

Figure 3-3 Corresponding sub-tree according to duty ---- *42*

Figure 3-4 Upward parent node chain update algorithm ---- *44*

Figure 3-5 Image of upward parent node chain update ---- *44*

Figure 3-6 Flowchart of Data synchronization ---- *46*

Figure 3-7 Kobe port on-line wharf site application form ---- *49*

Figure 3-8 Corresponding XML Schema definition ---- *50*

Figure 4-1 Web service state ---- *54*

Figure 4-2 Architecture overview of proposed system ---- *56*

Figure 4-3 Design diagram of context model schema ---- *59*

Figure 4-4 User model schema design diagram ---- *61*

Figure 4-5 Update Automatically with XSL Transformation ---- *63*

Figure 4-6 User active access to get extra information ---- *64*

Figure 4-7 Time trigger event diagram ---- *65*

Figure 4-8 Time-dependent exception handle policy algorithm ---- 66

Figure 4-9 Sample of the Context model ---- 68

Figure 4-10 Sample of the User model ---- 69

Figure 4-11 Example scenario information providing ---- 70

Figure 4-12 Example scenario exception handle flowchart ---- 71

Figure 5-1 Sample of Cargo delivery order ---- 75

Figure 5-2 Sample of Bill of lading ---- 76

Figure 5-3 Sample of Dangerous cargo list ---- 77

Figure 5-4 Sample of Cargo boat note ---- 78

Figure 5-5 Sample of Equipment receipt ---- 79

Figure 5-6 Sample of Container load plan ---- 80

Figure 5-7 Sample of Container move-in form ---- 81

Figure 5-8 Original container XML data model ---- 82

Figure 5-9 Original container XML data model Schema ---- 84

Figure 5-10 Sample of original container XML document ---- 84

Figure 5-11 Attributes grouping target ---- 85

Figure 5-12 Sample of XML document restructuration ---- 86

Figure 5-13 Sample of Indexed container XML document ---- 87

Figure 5-14 XML files capacity comparison ---- 88

Figure 5-15 An example of XPath query ---- 90

Figure 5-16 XPath querying time comparison while total container number=1,000 ---- 93

Figure 5-17 XPath querying time comparison while total container number=3,000 ---- 93

Figure 5-18 XPath querying time comparison while total container number=5,000 ---- 94

Figure 5-19 XPath querying time comparison while total container number=8,000 ---- 94

Figure 5-20 XPath querying time comparison while total container number=10,000 ---- 95

Figure 5-21 XPath querying time comparison while total container number=30,000 ---- 95

Figure 5-22 XPath querying time comparison while total container number=50,000 ---- 96

Figure 5-23 XPath querying time comparison while total container number=80,000 ---------- 96

Figure 5-24 XPath querying time comparison while total container number=100,000 ---------- 97

Figure 5-25 Query time reducing rate ---------- 97

Figure 5-26 File capacity comparison with reverse order and index attribute number -------- 100

Figure 5-27 XPath querying time comparison with reverse order and index attribute number 100

Chapter 1 Introduction

Along with the evolution of the Internet, various transactions between enterprises are recently processed through the Internet structure. In order to gain the business information, many famous international enterprises and organizations have already started to adopt the Extensible Markup Language (XML) as the standard format of data exchange (such as the Biztalk structure of Microsoft and the ebXML standard proposed by OASIS and UN/CEFACT). Simultaneously, various related technologies surrounding XML were developed, such as the analysis and parsing of XML documents (DOM and SAX), the definition for XML documents (DTD and XML Schema), the display and transformation for XML documents (XSLT), the designation within XML documents (XPath), the hyperlinks out of the XML documents (XPointer and XLink) and the query for XML documents (XQuery). XML is a self-describable language and its semantic characteristic extends to various research fields (e.g. the definitions and generations of XML documents, the communications depending on its semantic structure, the query algorithms for its tree-structure, the implementations within distributed structure environments, etc.). The semantic structure of XML provides the automatic communication facility between servers. This characteristic allows XML to play an important role within the growing ubiquitous technologies. Considering the format of data exchange for establishing the context model, the concept of agent also propagates extensive research issues. Following the multi implementations of XML, the Internet services are no longer user interfaces simply, but they are treated as one kind of object. The management and implementations of services can be achieved by XML. Furthermore, the issues about to access control and security are extended. Huge amount of researches have been proposed and described in the following:

- XML essence

  As the most popularized technology of data storage, relational database is still extensively used. As a result of this, the combinations of XML and relational database are surveyed a lot such as; reflecting updates of external XML documents into the loaded XML data in a relational database system [1]. An XML database engine can be easily built on the top of any existing Relational Database Management System (an open source is also proposed). It has been designed for the large databases and it can be partitioned, stored in multiple mobile devices. So, it cannot be a single source [2]. A method of merging XML

files into a relational database for permanent storage is also proposed [3], it takes advantage of relational database management system's search and query features to locate certain record. HiMASS-x was investigated [4]. It is an XML-centered suite of software applications. In their research, XML is used as an open, cross-platform, and extendable file format for the description of hierarchical simulation models, and it included the graphical representations, initial model conditions, and model execution algorithms. By combining both XML and secure PDF documents, an enterprise electronic contract management system can provide the capabilities and functionalities for contract creation, negotiation, execution and data mining. The structured XML document allows an easy and flexible way to create, negotiate, revise and finalize a contract when the PDF document permits adding signature as watermark on a secure signed contract [5]. XPEN is a scalable vector graphics format for visualization and a programming language via the DOM application programming interface was used for developing XPEN [6]. Xregion is a structure-based approach that can store XML data in relational databases. Xregion firstly partitions an XML document into several disjoint regions according to the cardinality of element nodes, and then maps these regions into separate relations [7]. XML can also be used as a non-conventional digital signature techniques and it queries over encrypted data [8]. Benchmarks belong to the very standard repertory of tools deployed in database development. Assessing the capabilities of a system, analyzing actual and potential bottlenecks, and comparing the pros and cons of different systems architectures have become naturally indispensable tasks as databases management systems grow in complexity and capacity. In the development of XML databases, a benchmark framework necessity has become more and more evident: a great many different ways to store XML data have been suggested in the literature. Each of them are needed to be carefully considered with its genuine advantages, disadvantages and consequences that propagate through the layers of a complex database system. The different storage schemes render the query characteristics of the data variably [9].

- XML Schema

  An XML schema provides a means by which an XML processor can validate the syntax and some of the semantics of an XML document. However, the use of schemas is optional; an XML document does not need to reference a schema, even if one exists. The resulting document can be processed more quickly when

the cost of some loss of confidence exists in the document quality. Some toolkits, can serve as the universal code generation toolkit for distributed frameworks, are proposed [10]. By defining a solid XML Schema for the real time domain attributes and applications from diverse platforms can be based on the data definition and create instances of XML documents to exchange data [11]. Researches of using flexible XML schema on the Internet are also discussed [12] for reducing costs and ultimately helping to generate new market opportunities for today's businesses. As the definition of XML documents, faults revealing is a very important issue, and fault classes defined for the XML schemas have been tested to validate them [13]. The Schema Component XML Syntax (SCX) is a representation which attempts to map schema components to the XML structures as faithfully possible. SCX serves the starting point for applications which need to access schema components and do so using standardized and widely available XML technologies [14].

- XML Generator

  By following the pre-defined XML Schema, XML documents can be generated automatically. An XML-based Partition Testing (XPT) approach for the automatic generation of XML instances from an XML schema is proposed, it is inspired by the well-known “Category Partition” method for black-box testing [15]. A method has been developed for creating functional test suites, in which a test engineer analyzes the system specification, writes a series of formal test specifications, and then uses a generator tool to produce test descriptions from which test scripts are written [16]. Using schema to correct sub trees are also discussed [17]. The dynamic generation of XML translators could be the future of the inter-operation among systems (B2B) and among devices (like cellular phones and machines) [18]. Automatically generating mappings between elements in the sources and the mediated schema are also available [19].

- Semantic structure

  One of the important characteristics of XML is semantic tags. A semantic management framework can provide the ubiquitous application field independent interoperability [20]. Semantic group mapping enables group node representation form on mobile devices and the corresponding operation sending scheme that can provide on demand detailed group information and maintain the consistency of pattern replicas on mobile site [21]. A simple Resource Description Framework

(RDF)-based ontological representation scheme is used for only structural relations among independently-managed XML schema from different institutes or domains. The semantic integration of scientific data uses XML schema and RDF-based schema mapping [22].

- Query efficiency

  When we use XML as data storage mean, there is an issue about how to query it efficiently. One way is to make a concise structure summary for XML documents [23]. The approaches of querying large numbers of text documents efficient using parallel processing methods are proposed [2]. The concepts of "weighted term frequency" and "inverted element frequency" are also proposed, where the weight of a term depends on its frequency and location within an XML element as well as its popularity among similar elements in an XML dataset. In which system users can configure appropriate index types for XML tags and text contents. Based on users' index configurations, the system transforms XML structures into a compact tree representation, Ctree, and indexes XML text contents [24].

- Distributed environment

  XML is available to access external documents, this characteristics makes it suitable to implement a distributed environment. A distributed telecommunication management system (DTMS) uses an object-oriented model to describe the networked voice communication system (VCS) to be managed. In order to allow the robust system evolution and maintenance, strong coherence of concern is achieved by strict encapsulation of the VCS model [25, 26]. A distributed query processing method for large XML data by partitioning and distributing XML data to multiple computation nodes is proposed [27, 28]. Parallel and distributed secure updates to XML documents are also discussed. Based on the use of a security region-object parallel flow (S-RPF) graph protocol, it is particularly suited for all environments that requiring cooperative updates to XML documents [29]. Applications communicate via an XML-based distributed virtual shared information space extends some common ideas on XML language binding frameworks by a dedicated "merge logic", that pervasive devices can share their information with low overhead [30]. Specifying consistency rules for distributed partial specifications with overlapping contents present a classification for different types of consistency rules related to various types of inconsistencies and that shows how to express these consistency rules [31]. An XML stream filtering

system that uses a large number of subscribers' profiles, can filter XML streams, and then publish the filtered data in real-time [32, 33]. A framework for the use of hierarchical federation communities as a tool for distributed simulation is proposed [34, 35]. An XML-based representation for distributed models, can be shared across the federation of simulations, is also discussed. Specifically, three different model components that form the basis of a design framework for distributed simulation models - object model (for simulation constructs), logic model (represents information flow within a model and its interactions) and the connectivity model (represents the links among the simulation constructs) [36].

- Ubiquitous environment

  Ubiquitous is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. Ubiquitous network combines the advantages of both disconnection operation and system virtualization in order to provide 7 days /24 hours services to mobile users [37]. Various ubicomp compliant objects provide uniqueness in scalable object identity, object categorization and standard format for the data being exchanged [38]. Context information is important in ubiquitous environment, quality dimensions and measurement methods should be considered according to characteristics of context information [39].

- Context model

  Context-awareness is an important concept for ubiquitous environment. Development of the context model and context acquisition can formally describe and acquire contextual information [40], and also provides an analysis of the runtime processes [41]. A Lightweight Context-Aware Service-Oriented Architecture (LCASOA) based on OSGi (Open Services Gateway initiative), is proposed to support context acquisition, discovery and reasoning [42]. "Hydrogen Context-Framework" proposes a software framework which can support context-awareness [43]. Context-Aware Service Middleware (CASM) provides a middleware-level support for building and rapid prototyping of context-aware services [44, 45].

- Implementation

  Ubiquitous environments are applied in various fields [46]. An exhibition reminiscent system for generating tour summary in a ubiquitous environment is

proposed [47]. “Blue Network”, a water recycling project, is processed to take advantage of U-technology [48]. XNMP, an XML-based network management protocol, can greatly simplify the overhead needed to design data structures and algorithms in VoIP setting [49]. The healthcare professionals can access patient information stored in heterogeneous autonomous information systems from a set of formal aggregates of health data based on the healthcare record architecture ENV13606 from CEN/TC251 [50]. The XML-based “Clinical Document Architecture” (CDA) for document exchange defines a three-level document architecture with each higher level by adding more specificity to the markup of the document. Such a layered architecture also meets the requirement for semantic processing of hierarchically structured clinical documents [51]. A framework based upon the use of semi-structured database management systems would provide an integrated interface for the collection, storage and retrieval of biological data from existing repositories and biological information generated by existing analysis programs [52]. The concept of ubiquitous classroom and its implementation enable us to expand “the awareness among faculty and students in classroom” [53, 54]. An alternative set of potential design principles for educational ubiquitous computing is proposed, it can stress values such as expressiveness, creative control, and aesthetics [55]. Methods that can observe a learner’s behaviors by a ubiquitous environment are also discussed [56]. An Austrian research project makes educational material accessible for students with disabilities using XML standard and transformational algorithms [57].

- Ubiquitous services

  As the “Service Oriented Architecture” (SOA) principles have gained importance, an emerging need has appeared for methodologies to locate desired services that provide access to their capability descriptions. These services must be typically assembled into short-term service collections together with code execution services that are combined into a meta-application to perform a particular task. Context-awareness makes services more flexibly. A personal space model called VPW (Virtual Personal World) supports more delicate user-centric adaptation, use of multiple devices spread over locations, and harmony between multiple users and multiple services [58]. Storing location and time, frequency information of often used service among various services can reduce using and searching time of service through technique [59]. A consistent hashing-based catalog service engine can map keys onto the nodes in an identifier ring with load

balance and speed the key location [60, 61].

- Access control

  Context-awareness services are normally combined with user identification to classify the permissions, so, access control issues should be discussed together [62]. A context roles extended from current RBAC can control the accesses to privacy in ubiquitous environment. It employs multi-policy to constrain privacy and role-data objects [63]. An experimental middleware infrastructure named Gaia provides the user-oriented interfaces for the physical spaces populated with network-enabled computing resources [64]. Access control models can be built upon the concept of context as the first-class design principle to rule access to resources, this allows to associate access control permissions with contexts where users operate and users acquire/lose their permissions when entering/leaving a specific context [65]. For an active space, an access control system that automates the creation and enforcement of access control policies for different configurations is proposed [66]. UbiCOSM, a context-centric access control middleware, dynamically determines the contexts of mobile users and effectively rules the access to them by taking into account different types of metadata (user profiles and system/user-level authorization policies) [67]. A reusable access control layer for Web service software is also proposed. The layer is designed as an independent software component which is separated from the application components of Web applications and services. It applies expert system type of rules using an inference engine to determine security rules and access rights [68]. A context aware access control architecture called “Anonymous Context Aware Access Control Architecture” (ACA2) is proposed. The basic idea of this architecture is based on an analogy to the public telephone service. Anonymous users can use services supported by their context information through preregistered software components called proxies. The main features of the architecture include anonymity, access suspension caused by context changes, and active context certificates with stream verification [69]. An approach to add contextual information is the distributed Role Based on Access Control (dRBAC) model to support spontaneous coalition is also proposed [70]. A family of Coalition-Based Access Control (CBAC) models provides a range of expressivity with an accompanying range of implementation complexity, which defines the protection state of a system, which provides the semantics of CBAC-based access policies [71]. A neural network algorithm for ubiquitous applications extends the

role based access control with context constraints [72, 73]. PAPI is a system for providing access control to the restricted information resources across the Internet. The authentication mechanisms are designed to be as flexible as possible by allowing each organization to use its own authentication schema, keeping user privacy, and offering information providers data enough for statistics. Access control mechanisms are transparent for user and compatible with the most commonly employed Web browsers and any operating system [74]. A content-based authorization model suitable for a digital library environment provides both content-dependent and content-independent access control to the digital library objects [75]. Languages for the specification of access restrictions use standard notations and concepts, together with a description of a system architecture for access control enforcement are discussed [76, 77]. Combining role-based access control found in the “Role Graph Model” are also discussed [78]. An access control system reduces the complexity involved in defining authorization permissions, particularly in structured documents such as XML where the user may be granted restricted access, is proposed [79, 80].

Compared with relational database, XML might be a more suitable option for the maritime field. The objective of this research is to focus on the possibility of the implementation of XML for the maritime field. Three issues are discussed in this study, the first one is the data modeling for fleet management and its potential possibility of implementation on port administration; the second is the service integrated modeling under the close intranet structure within the ship; and the third is to propose an efficient data structure for XML to reduce the disadvantage caused by XML essential. The other parts of this thesis are organized and described as follows:

In chapter 2, the basic concepts of XML related technologies including “Document Object Model” (DOM), Simple API for XML (SAX), XML Path Language (XPath) and “Extensible Stylesheet Language” (XSL) are introduced. A simple comparison of XML and relational database is given at the end of this chapter,.

In chapter 3, I propose to use XML for constructing ship data management model, so then it could be applied for fleet management and future port administration. The data model can be composed of many kinds of information, such as; hardware (ship name, ship type, net tonnage, etc.), software (schedule, service, passenger, etc.) and freight (freight kind, freight charge, deliver country, etc.). Multi-level data model can provide the ability for fleet inter-management and communicating function to government related departments for port administrations.

In chapter 4, I introduce XML as metadata to establish context models for services and user model for identification. They are used to describe the behaviors of ordinary operations and tried to provide the desired information users need by separating the attribute connection within models. There are a lot of researches related to ubiquitous technology that have been studied, but most of them are focused on electronic commerce level. In this research, I focused on sailing management and the efficiency of typical sailing operation is improved by using the advantage of ubiquitous environment. In a passenger ship, many people work and travel (e.g. there should be various necessary information needed to be transmitted and shared). But in this case, the information should be classified and separated by its attributes. Real time information sharing and selection issue should be considered to solve the complex relationship between context and user group.

In chapter 5, I discuss the efficiency of querying XML documents and propose an efficient data structure. Compared with relational database, XML has the semantic advantage. This characteristic allows it to communicate with servers and to achieve the objective of automatic operations. On the other hand, compared with the binary structure of database, XML is built in pure text format so that it takes more data storage space and has obvious backward query efficiency. In this research, in order to reduce the disadvantage of XML essence, the data storage structure for ship data modeling was investigated from query efficiency view and this data model is applied on the container operation as a sample to verify the improvement of query efficiency.

Chapter 2 Background Concepts

Relational Database systems (RDBS) are well-known for consistent storage, retrieval, and manipulation of data. At the same time, the Extensible Markup Language (XML) is generally accepted as data description language for both Web-based information systems and electronic data interchange among different organizations.

Recently, XML becomes a standard for data communication over the Internet. Like HTML, XML is a markup language, but it supports a rich set of features, such as; user-defined tags that allow both data and descriptive information about data to be represented within a single document. Simultaneously, presentation aspects remain decoupled from data representation.

The flexibility of XML allows it to serve as a meta-language for defining other markup languages specialized in a specific context. A document type definition (DTD or XML Schema) describes the tags documents can be used, and it customized to the specific semantic requirements of the application context in the rules connecting tags with their contents. These capabilities make XML to be a common data format for data interchange among computer systems and applications.

XML and database seem like an odd couple (two very different concepts driven by two different communities with different expectations and requirements). The selection of adoption from them should be considered the characteristics and properties of the implemented targets.

Data model heterogeneity is focused in this chapter, and it provides a comparison of concepts available in RDBS and XML schema specification languages, comprising XML DTD and XML Schema.

2.1 Extensible Markup Language (XML)

The use of information technology and the Internet have changed how the business was done in today's global environment. Consistent with the evolution of these business processes, the use of data for decision-making is also increasing at a rapid pace. This increased demand has led to the use of several types of Markup Languages for better and faster information. The term Markup is generally applied in any set of codes or tags added to the contents of a document, in order to indicate its meaning or presentation. The best example of this is using Hypertext Markup Language (HTML) on Web pages. Through the use of the software program called Web browser, users are able to view pages of information that are presented using a series of tags. The tags tell the software (Web browser) how to interpret the data and display the information on the screen.

The way of using HTML has changed the mechanisms of business and how information is accessed via the Internet. An extension of this concept is the Extensible Markup Language (XML) [81, 82, 83, 84] that originated in September, 1998 by the World Wide Web Consortium (W3C) [85]. The design of XML was created within the concept of transferring data embedded into its definition. It is an independent, global way to express any kind of information by using constructs that can be accommodated to fit particular needs.

XML is a subset of the Standard Generalized Markup Language (SGML) designed to provide the flexibility and power of SGML, while capitalizing on the popularity of the Hypertext Markup Language (HTML). XML is a markup language similar to HTML. Conversely to HTML, HTML was designed to display data using a predefined set of tags, whilst XML was designed as a structure to store and carry data within markup tags defined by the user. Figure 2-1 shows an example of XML tags and its document structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<employee id="1026">
        <name>John Smith</name>
        <department>IT</department>
        <address>
                <apt>230 City Towers</apt>
                <street>Newport Avenue</street>
                <city>Newark</city>
                <state>New Jersey</state>
        </address>
</employee>
```

Figure 2-1 An example of XML document structure

Similarly with HTML, XML makes use of tags (words bracketed by “<” and “>”) and attributes (of the form name= "value") to mark up text.

The second line in Figure 2-1 is a tag with employee as the tag name. It has an attribute id (identification) by the value “1026”. <employee> is a begin tag. </employee> in the last line is its corresponding end tag.

The free text that occurs outside the tags is character data, or content. “John Smith” in Line 3 of Figure 2-1 is an example of this character data.

Unlike in HTML, XML allows the use of one’s own tags for the markup.

2.1.1 XML Features

XML is a simple, very flexible markup language. Originally designed to meet the challenges of large-scale electronic publishing, its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the internet.

The characteristics of XML can be described as follows:

- Tree architecture: XML can structure data and display it with a tree-structure. Also, the unstructured data that relational database is difficult to handle can be translated to the Semi-structured data [86] by definitions of DTD (Document Type Definition) or XML Schema. Then, the unstructured data could be much easy to be handled by applications.
- Normal text format: pure text store format allows users to use various free text editor software to edit a XML document.
- Unicode support: for the international era of nowadays, the support of multi-language is much more important than before. This feature is especially useful for an international enterprise like Ship Company did.
- Self-define tag: this feature provides the extensibility. XML provides a markup structured data construction; it allows users to define specific professional field tags by themselves.
- Data validation: precise data structure and grammar promise that all XML documents could be checked by XML parser and make sure of that the names, sequences and layers of elements analyzed by different XML parsers will receive

the same results.

- Independency of data structure: data is separated from display format. XML is a markup language for describing data; it does not define the display format within itself. The display format will be handled by XSL (Extensible Style Language) [87]. Because of the content and display format are separated, it is much easier to read and maintain for both human beings and computers.

XML could be a more suitable data management model than RDBMS due to its characteristics that described above.

2.1.2 Well-Formed XML Document

XML enforces strict rules for a document to be well-formed. A well-formed XML document is one that corresponds to the XML 1.0 grammar specified by W3C. A well-formed XML document has exactly one root element. <employee> is the root element of the XML document in Figure 2-1.

Each of beginning element (beginning tag) in XML should have a corresponding end element (ending tag). The elements should be nested within another. The tags and nesting rules allow XML to represent information in a hierarchical manner.

Following the definition established by W3C, a Well-Formed XML document should follow the principles described as follows:

- To be taken as a whole. It matches the production labeled document.
- It meets all the well-formedness constraints given in W3C's specification.
- Each of the parsed entities is referenced directly or indirectly within the document and is well-formed.

XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree. The only indispensable syntactical requirement is that the document has exactly one root element. This means that the text must be enclosed between a root start-tag and a corresponding end-tag.

The root element can be preceded by an optional XML declaration. This element states which version of XML was used (normally 1.0); it may also contain information about character encoding and external dependencies (e.g. the encoding is “UTF-8” in Figure 2-1).

Comments can be placed anywhere in the tree, including in the text if the content of the element is text or “#PCDATA”.

XML comments start with “<!-- “ and end with “-->”. Two dashes “--" may not appear anywhere in the text of the comment.

In XML, a well-formed document must conform to the following rules:

- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an empty-element (self-closing) tag, such as “<IAmEmpty />”. This is equal to “<IAmEmpty></IAmEmpty>”.
- All attribute values are quoted with either single (') or double (") quotes. Single quotes must close a single quote and double quotes must close a double quote.
- Tags may be nested, but should not be overlapped. Each non-root element must be completely contained in another element.
- The document complies with its declared character encoding. The encoding may be declared or implied externally, such as in "Content-Type" headers when a document is transported via HTTP, or internally using explicit markup at the beginning of document. When there is no such declaration, a Unicode encoding is assumed, as defined by a Unicode Byte Order Mark before the document's first character. If the mark does not exist, UTF-8 encoding is assumed.

XML is also more structured than HTML since XML documents are required to conform strictly to XML syntax as defined by W3C’s XML specification. XML documents that conform to the XML syntax are referred to as well-formed. However, being well-formed does not guarantee that a document is free of errors. One way of checking an XML document for content or structure that is not valid is to use an XML schema.

2.1.3 Valid XML Document

XML allows users to specify the structure of an XML document by a customizable schema or Document Type Definition (DTD). XML provides a syntactic foundation ensures that all XML-aware software can at least read and understand the relative arrangement of information within them. The schema merely supplements the syntax rules within a set of constraints. Schema typically restrict element and attribute names and their allowable containment hierarchies, such as; only allowing an element named “birthday” to contain one element named as “month” and one element named as “day”, each of them has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the “month” element's character data may be defined as being a month according to a particular schema language's conventions; perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it is some other type of data.

An XML schema is an XML document that is used to define and restrict the structure and content of the XML document, and sometimes referred as an XML language. XML schema provide for the use of primitive, generated, and user-defined types. Primitive types consist of string, boolean, byte, long, etc. Generated types are predefined types that build upon existing primitive types to form new types, e.g. date, time, integer. Users can also define their own types using primitive, generated, and other user-defined types. Restrictions can also be placed on the various types. For example, a type might restrict itself to integers in the range from 1 to 10. Therefore, a valid entry for an element or attribute of this type would only consist of the integers from 1 to 10. XML documents that comply with the rules of the schema document are referred to as instances of this schema. The process of verifying that an instance of the schema conforms to the schema language is referred to as validation. An XML document that conforms to its schema is therefore a “valid” XML document for the schema.

There are two kinds of schemas for XML for now; DTD and XML Schema.

- DTD

The Document Type Definition (DTD) is the oldest schema format for XML, inherited from SGML. DTD support is ubiquitous, due to its inclusion in the XML 1.0 standard and it is seen as limited because of the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressiveness. Certain formal aspects of an XML document cannot be captured in a DTD.
- It uses custom non-XML syntax, inherited from SGML, to describe the schema.

Figure 2-2 shows the DTD as a sample XML document from the sample in Figure 2-1.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!ELEMENT employee (name, dept, address)>
<!ATTLIST employee id CDATA #REQUIRED >
<!ELEMENT name (#PCDATA)>
<!ELEMENT dept (#PCDATA)>
<!ELEMENT address(apt, street, city, state)>
<!ELEMENT apt (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
```

Figure 2-2 A DTD sample

DTD is still used in many applications because it is considered the easiest to read and write.

- XML Schema

A newer XML schema language, described by the W3C as the successor of DTD, is XML Schema, or more informally referred as the initialism for XML Schema instances, XSD (XML Schema Definition). XSD are more powerful than DTD in describing XML languages. They use a rich datatyping system, allow more detailed constraints in an XML document's logical structure, and must be processed in a more robust validation framework. XSD also use an XML-based format which makes it possible to use ordinary XML tools and to help process, although XSD implementations require much more than just the ability to read XML.

Criticisms of XSD include the following concepts:

- The specification is too large, which makes it difficult to understand and implement.
- The XML-based syntax leads to verbosity in schema descriptions; this makes XSDs to be hard for reading and writing.
- Schema validation can be an expensive addition to XML parsing, especially for high volume systems.
- The modeling capabilities are very limited; with no ability allow attributes to influence content models.
- The type derivation model is very limited; in particular that derivation by extension is rarely useful.
- Database-related data transfer is supported with arcane ideas such as null ability, but the requirements of industrial publishing are under-supported.

Figure 2-3 shows the XML Schema as a sample XML document for the sample in Figure 2-1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="employee">
      <xs:annotation>
         <xs:documentation>Comment describing your root element</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="department" type="xs:string"/>
            <xs:element name="address">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="apt" type="xs:string"/>
                     <xs:element name="street" type="xs:string"/>
                     <xs:element name="city" type="xs:string"/>
                     <xs:element name="state" type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:attribute name="id" type="xs:integer"/>
</xs:schema>
```

Figure 2-3　A sample for XML Schema

XML Schema is a more powerful tool than DTD; its XML structure gives the unity for XML documents management. Nowadays, more and more applications adopt XML Schema as the definition for their documents.

2.2 Processing XML

In order to read, update, create and manipulate an XML document; an XML parser is required. In this respect, a parser reads the structure and content of XML document and provides an application based on this structure and content for further manipulation. Parser is also required to check an XML document for well-formedness.

Parsers can be validating or non-validating. Parsers that use an XML schema or DTD to validate the document are referred to validating parsers.

Typically XML processing involves the following steps:

- Parse - Scanning through the XML document processing elements and attributes, and possibly build an in-memory tree (for Document Object Model parsers). Parsing is a pre-requisite for any processing of XML document.
- Access - Extracting the data from the elements and attributes of parts of the document into the application program. For example, to consider be given an XML document for an invoice, the application might want to retrieve the prices for each item in the invoice.
- Modify - Changing the textual content of elements or attributes, and possibly also the structure of the document by inserting or deleting elements. This does not apply to streaming parsers. As an example, an application might wish to update the price of some items in an invoice or may want to insert or delete some of them.
- Serialize - Converting the in-memory tree representation to a textual form; that is written to a disk file or forwarded to a network stream. This makes sense only for tree building parsers, and is necessary for the cases where the XML document has been modified in memory.

The Simple API for XML (SAX) and the Document Object Model (DOM) are two common techniques to access and process the information content of an XML document. SAX is serialized based on events and DOM produces a complete tree-like structure as output. These two parsers can be implemented as shown in Figure 2-4, and their details will be described in the following sections.

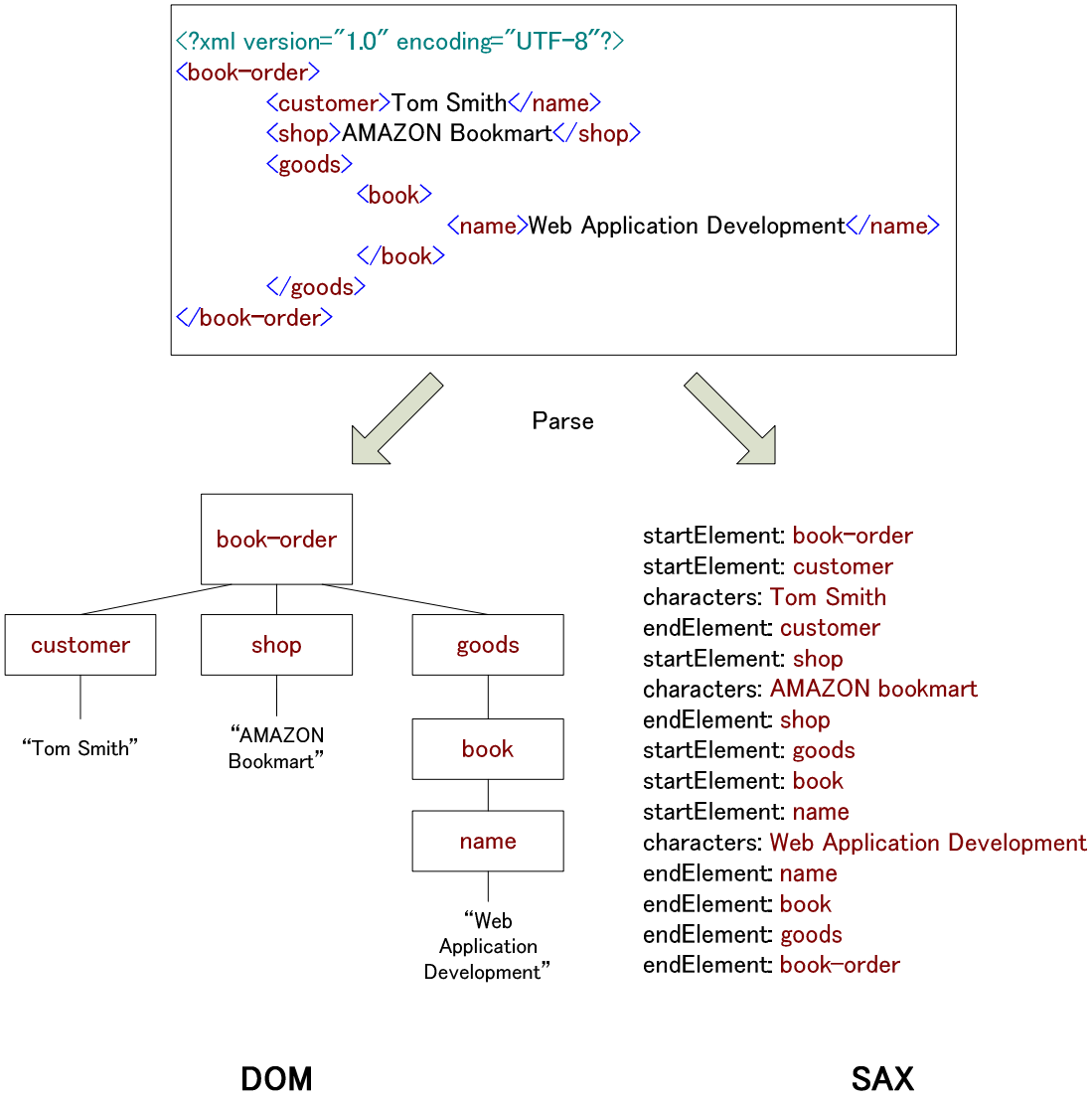


Figure 2-4 DOM and SAX parsers

2.2.1 Simple API for XML (SAX)

David Megginson, principal of Megginson Technologies, led development of the Simple API for XML (SAX) [88, 89, 90], a widely-used specification that describes how XML parsers can pass information efficiently from XML documents to software applications. SAX was originally implemented in Java, but is now supported by nearly all major programming languages.

Unlike most XML-related specifications, SAX does not come from a formal committee at a standards body or industry consortium. David putted together a proof-of-concept implementation during the holidays in December 1997, and he presented it to the xml-dev mailing list in January 1998. Through collaborative online discussion, the xml-dev members developed the proof-of-concept into SAX 1.0 which immediately received industry-wide acceptance from both commercial and free software developers. SAX, although not an official W3C recommendation, has become an industry standard for parsing XML in practice, due to its simplicity and processing speed.

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled with the same way, no matter where they occur in the document.

A parser which implements SAX (i.e., a SAX Parser) functions as a stream parser with an event-driven API. The user defines a number of callback methods that will be called when events occur during parsing. The SAX events include the follows:

- XML Text nodes
- XML Element nodes
- XML Processing instructions
- XML Comments

Events are fired, when each of these XML features are encountered, and again when the end of them is encountered. XML attributes are provided as part of the data passed to element events.

SAX parsing is unidirectional; previously parsed data cannot be re-read without starting the parsing operation again. Figure 2-5 shows the XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<RootElement param="value">
        <FirstElement>
            Some Text
        </FirstElement>
        <SecondElement param2="something">
            <Inline>
                Inlined text
            </Inline>
        </SecondElement>
</RootElement>
```

Figure 2-5 A sample for SAX

This XML document, when passed through a SAX parser, will generate a sequence of events as follows:

- XML Processing instruction, named xml, with attributes version equal to "1.0" and encoding equal to "UTF-8"
- XML Element start, named RootElement, with an attribute param equal to "value"
- XML Element start, named FirstElement
- XML Text node, with data equal to "Some Text" (note: text processing, with regard to spaces, can be changed)
- XML Element end, named FirstElement
- XML Element start, named SecondElement, with an attribute param2 equal to "something"
- XML Element start, named Inline
- XML Text node, with data equal to "Inlined text"
- XML Element end, named Inline
- XML Element end, named SecondElement
- XML Element end, named RootElement

2.2.2 Document Object Model (DOM)

DOM (its official name is W3C Document Object Model) is a string-based Application Programming Interface (API) for manipulating XML data and structures.

According to the definition of W3C:

- The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.

DOM creates a representation of the XML document in a tree-like structure consisting of parent and child nodes. This representation is held within memory and can be manipulated by the DOM API to add, delete or modify data, and can also be used within a parser application to assist in the validation of XML document against a related Document Type Definition (DTD) or XML schema. One key feature of DOM is its ease of use. However, a major drawback to DOM is its need to load the entire XML document structure into memory. This can be problematic when large document sizes are used. The XML DOM views an XML document as a tree-structure. The tree structure is called a node-tree. Figure 2-6 shows a sample of XML node-tree structure.

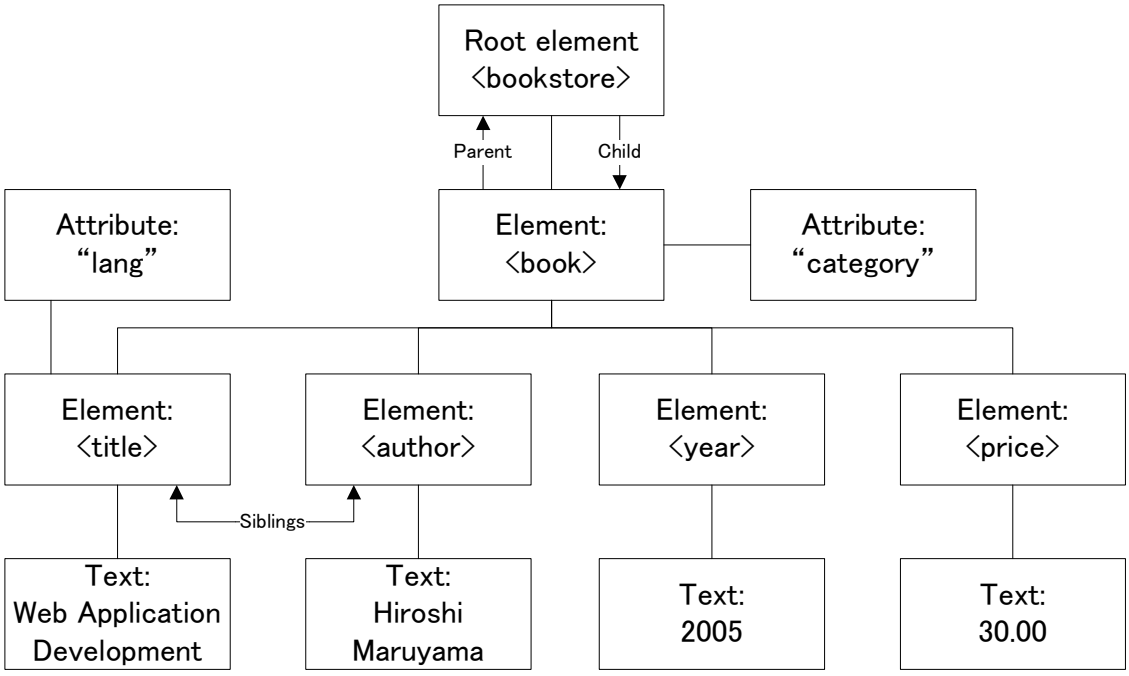


Figure 2-6 An XML node-tree sample

The DOM parser performs three functions—it represents the XML document in memory as a tree, provides methods to manipulate the in-memory tree, and provides methods to create new XML elements.

According to the DOM, everything in an XML document is a node.

- The entire document is a document node.
- Every XML element is an element node.
- The texts in XML elements are text nodes.
- Every attribute is an attribute node.
- Comments are comment nodes.

The XML DOM defines the objects and properties of all XML elements, and the methods (interface) to access them. In other words; The XML DOM is a standard for how to get, change, add, or delete XML elements.

The nodes in the node tree have a hierarchical relationship to each other. The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).

- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

Figure 2-7 illustrates a part of the node tree and the relationship among the nodes:

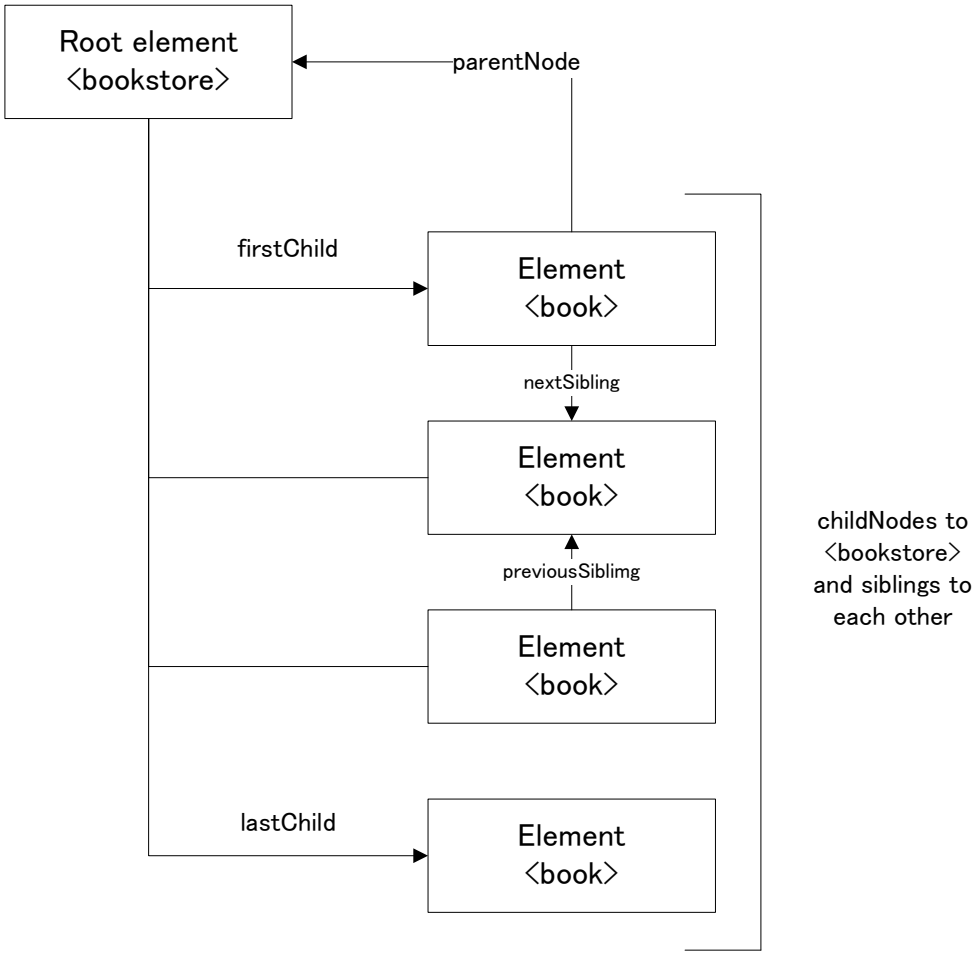


Figure 2-7　hierarchical relationship of XML nodes

Because the XML data is structured in a tree form, it can be traversed without knowing the exact structure of the tree and without knowing the type of data contained within.

The primary benefit of DOM over SAX is that the entire document information is available in memory. Hence arbitrary processing and transformation of elements is possible. On the other hand, a drawback of this approach is that application processing has to wait till the entire document is parsed and the document tree constructed in memory.

2.3 XML Path Language (XPath)

The XML Path Language (XPath) [91] is a language for selecting or pointing out parts of XML documents. It may be used in many contexts. Nowadays, the best known is XPointer which is again used by XLink and XInclude to refer to fragments of other documents. XPath is the primitive that allows pointing to the internals of documents, and not just their wholes. XPath is restricted to operate on XML Namespaces compliant documents, and of course, is operated under the XPath data model. The core of XPath expressions are the location paths, those are used for selecting nodes in XML data.

A location path consists of a series of location steps, each consecutively producing a set of nodes working from the set produced by the previous step. The final node-set is the result of the location path evaluation. A node-set is one of the four types of values in the XPath language, and it is simply an unordered collection of references to nodes in some XML document (see Figure 2-8 that is a visualization of the evaluation model for one location step).

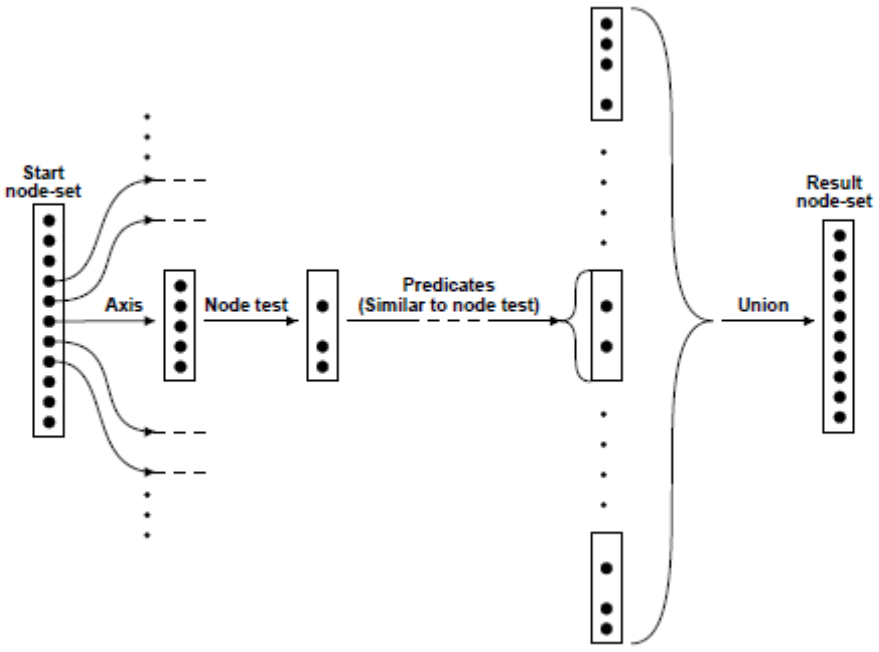


Figure 2-8 XPath evaluation model for a single location step

A location step is comprised of three parts:

- An axis; doing an initial rough selection of nodes by their structural relationships to the context node (the nodes in the source set), e.g. the child axis, selecting all children of the context node.
- A node test; refining what kind of nodes we are interested in. This is often a specific element type; such as chapter, leaving only elements with the name chapter in the set.
- Any number of predicates; which are essentially Boolean expressions that filter out all nodes for which the expression is not true.

The XPath axes are: child, attribute, namespace, descendant, self, descendant-or-self, parent, ancestor, ancestor-or-self, preceding, following, preceding-sibling and following-sibling. Each axis is either a forward or reverse axis. From the specification:

> "An axis is that only ever contains the context node or nodes that are after the context node in document order is a forward axis. An axis is that only ever contains the context node or nodes, are before the context node in document order is a reverse axis."

Thus the reverse axes are: parent, ancestor, ancestor-or-self, preceding and preceding-sibling. The self axis can also referred as "directionless" and it falls into both categories, but the ordering makes no difference since selects a single node. Similarly for the parent axis: it only ever selects one node or none at all (for the root node).

In addition, every axis has a principal node type which is element nodes for all the axes that can select elements. This means that most of the axes, except for the attribute and namespace axes, for which the principal node type is the attribute and namespace nodes respectively.

An axis defines a node-set relative to the current node. The details are given in Table 2-1.

Table 2-1 XPath Axes definition

Axis Name	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node.
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself.
attribute	Selects all attributes of the current node.
child	Selects all children of the current node.
descendant	Selects all descendants (children, grandchildren, etc.) of the current node.
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself.
following	Selects everything in the document after the closing tag of the current node.
following-sibling	Selects all siblings after the current node.
namespace	Selects all namespace nodes of the current node.
parent	Selects the parent of the current node.
preceding	Selects everything in the document that is before the start tag of the current node.
preceding-sibling	Selects all siblings before the current node.
self	Selects the current node.

XPath includes a full expression language with Booleans, numbers and strings as well as the node-sets which the location paths yield. These expressions allow quite complex filtering in the predicates. A common predicate is testing the proximity position of nodes through the position () function, e.g. selecting every other node for given type (the proximity position will be according to the document order for a forward axis; axis it will be the reverse document order for a reverse).

2.4 Extensible Stylesheet Language (XSL)

The Extensible Stylesheet Language (XSL) is made up of three related languages:

- XSLT: Extensible Stylesheet Language for transformations.
- XPath: a language used to access the documents.
- XSL-FO: a formatting language.

Data represented within XML can be manipulated by using XSL to provide different text formats of data. The most common transformation is XML to HTML/XHTML for presentation of data within websites. This capability is the result of XML's separation of data from its presentation. The strength of XSL is that the same data can be represented in different ways. Data can be transformed from one format to another. Data that is not required for a particular view can be omitted, and data that is missing can be added through subsequent applications of an XSLT to the processed data.

XSLT is the most important part of XSL.

XSLT is used to transform an XML document into another XML document, or another type of document recognized by a browser, such as HTML and XHTML. Normally XSLT does this process by transforming each XML element into an (X) HTML element.

XSLT allow users to add/remove elements and attributes to or from the output file. The users can also rearrange and sort elements, perform tests and make decisions about which elements should be hided and displayed, and a lot more.

A common way to describe the transformation process is to say that XSLT transforms an XML source-tree into an XML result-tree.

XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents.

In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

The Extensible Stylesheet Language Formatting Objects (XSLFO) is a pagination markup language describing a rendering vocabulary by capturing the semantics of formatting information for paginated presentation.

Formatting is the process of turning the result of an XSL transformation into a tangible form for the reader or listener. This process comprises several steps; some of them depend on others in a non-sequential way. The model for formatting will be the construction of an area tree, which is ordered tree containing geometric information for the placement of every glyph, shape, and image in the document, together with information embodying spacing constraints and other rendering information.

When consequent tree in an XSLT process is specified to utilize the XSLFO pagination vocabulary, the normative behavior of an XSLFO processor incorporating an XSLT processor is to interpret as the consequent tree. This interpretation reifies the semantics expressed in constructs of the consequent tree to some medium; e.g. pixels on a screen, dots on paper, and sound through a synthesis device as shown in Figure 2-9.

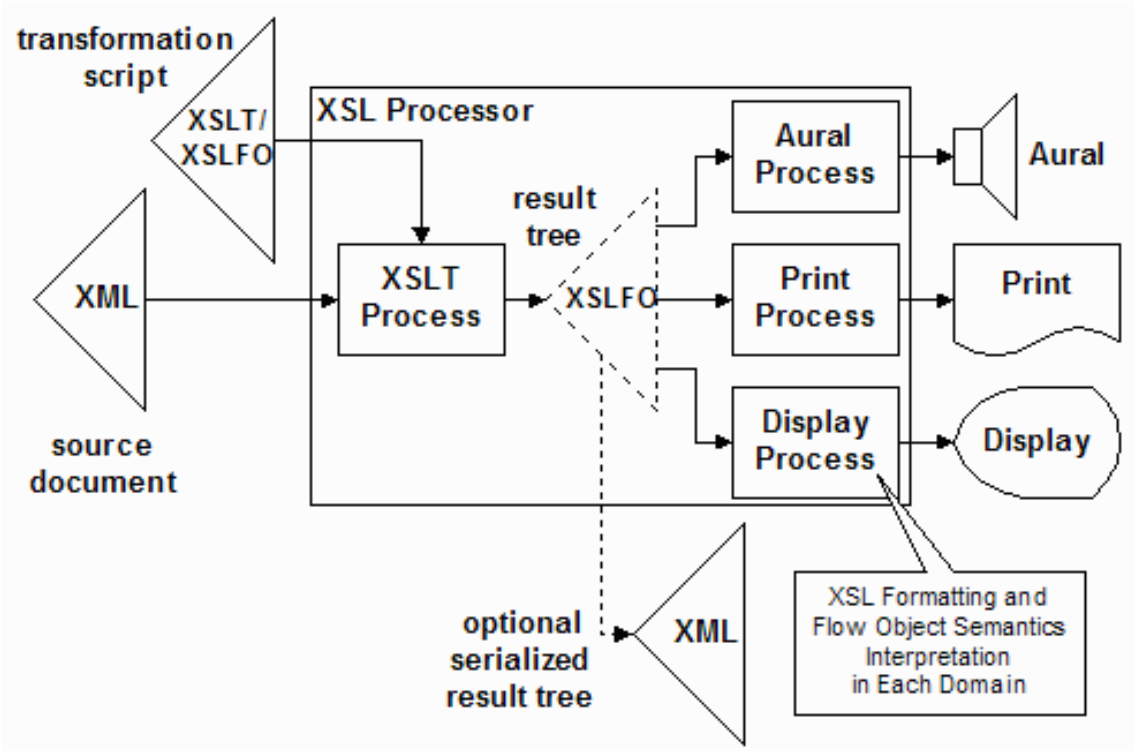


Figure 2-9 Transformation from XML to XSL Formatting Semantics

An XSL stylesheet is a well-formed XML document. The basic building block of an XSL style sheet is a template rule. A template rule consists of two parts:

- A pattern that identifies an element in the XML document.
- An action that specifies the transformation and rendering of the identified node.

The root element that declares the document to be an XSL style sheet is "<xsl:stylesheet>" or "<xsl:transform>". To get access to the XSLT elements, attributes and features, the XSLT namespace must be declared at the top of document.

XSLT allows elements to be identified (matched) in various ways as follows;

- By name: <xsl:template match="employee" >
- By ancestry: <xsl:template match = "employee//state" >
- By attributes: <xsl:template match = "employee[@id="1234"]" >
- By children: <xsl:template match = "employee[name]" >
- By position: <xsl:template match = "employee[lst-of-type()]" >
- By matching several element names: <xsl:template match="employee|manager">

A key feature of XSLT is its recursive nature. Starting from the root element of XML document, each node is inspected to see if there is an applicable template rule. In case of more than one rule being applicable to a particular node, XSLT allows priorities to be specified for transformation. The relevant rule is invoked, and the inspection then continues on the children of the new node. <xsl:apply-templates> is used to apply this recursive action.

XSLT allows conditional processing of elements with xsl:if and xsl:choose constructs. <xsl:apply-templates select="xxx"/> can be used to process selected children alone of a matched element. <xsl:for-each> construct can be used to process each of repetitive elements. Sorting and numbering of elements are other manipulations that can be performed.

Figure 2-10 shows a sample of XSL transformation. The example XML book data transforms into the output HTML data by the XSL document defined by the user. The right side of Figure 2-10 shows the output of displayed in Web browser before and after the transformation.

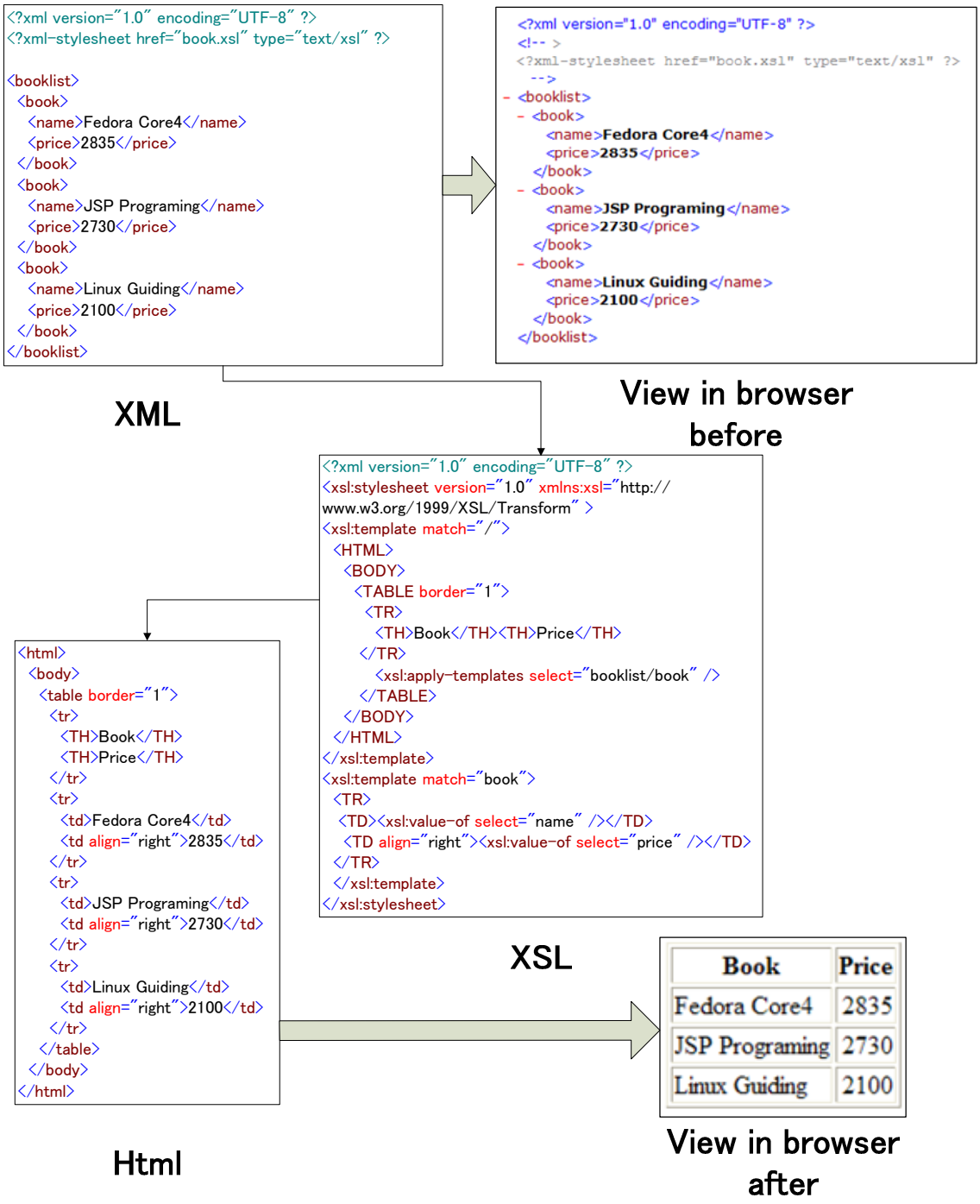


Figure 2-10 An XSL transformation sample

2.5 Comparison of XML and Relational Database

Database systems are well-known for consistent storage, retrieval, and manipulation of data. Similarly, XML is generally accepted as data description language for both Web-based information systems and electronic data interchange for different organizations [92].

When the users design our databases, they need to decide whether their data is better suited to the XML or the relational model. While this discussion explains some main differences between the models and factors applied to each, there are numerous factors can determine the most suitable choice for the implementation.

The major differences between XML data and relational data are:

- XML data is hierarchical; relational data is represented in a model of logical relationships.

  An XML document contains information about the relationship of data items to each other in the hierarchical form. With the relational model, the only types of relationships that can be defined are parent table and dependent table relationships.

- XML data is self-describing; relational data is not.

  An XML document contains not only the data, but also tagging for the data that explains what it is. A single document can have different types of data. With the relational model, the content of data is defined by its column definition. All data in a column must have the same type of data.

- XML data has inherent ordering; relational data does not.

  For an XML document, the order in which data items are specified is assumed to be the order of data in the document. There is often no other way to specify order within the document. For relational data, the order of rows is not guaranteed unless we specify an ORDER BY clause on one or more columns.

Sometimes the nature of data dictates a way in which the users store it. For example, if the data is naturally hierarchical and self-describing, they might store it as XML data. However, other factors might influence our decision about which model to use.

Some of those factors are:

- Maximum flexibility of data is needed.

  Relational tables are fairly rigid. For example, it would be very difficult to normalize one table into many or de-normalize many tables into one. If the data design is changed very frequently, it is better to represent it as XML data.

- Maximum performance for data retrieval is needed.

  Some expense is associated with serializing and interpreting XML data. If performance is more of an issue than flexibility, relational data might be the better choice.

- The data is processed later as relational data.

  If subsequent processing of data depends on the data being stored in a relational database, it might be appropriate to store some parts of data as relational by using decomposition. An example of this situation is; when online analytical processing (OLAP) is applied to the data in a data warehouse. Also, if other processing is required on the XML document as a whole, then storing some of the data as relational (as well as storing the entire XML document) might be a suitable approach in this case.

- The data components have meaning outside a hierarchy.

  Data might be inherently hierarchical in nature, but the child components do not need parents to provide value (e.g. a purchase order might contain part numbers). The purchase orders with part numbers might be best representation as XML documents. However, each part number has a part description associated with it. It might be better to include the part descriptions in a relational table, because of the relationship between part numbers and part descriptions is logically independent from the purchase orders in which the part numbers are used.

- The data attributes apply to all data, or to only a small subset of data.

  Some sets of data have a large number of possible attributes, but only a small number of those attributes apply to any particular data value. For example, in a retail catalog, there are many possible data attributes, such as; size, color, weight, material, style, weave, power requirements or fuel requirements. For any given item in the catalog, only a subset of those attributes is relevant (power requirements are meaningful for a table saw, but not for a coat). This type of data is difficult to be represented and searched with a relational model, but relatively easy to be represented and searched with XML model.

- The ratio of data complexity to volume is high.

  Many situations involve highly structured information in very small quantities. Representation of that data with a relational model can involve complex star schemas in which each dimension table is joined to many more dimension tables, and most of these tables have only a few rows. A better way to represent this data is to use a single table with an XML column, and to create views on that table, where each view represents a dimension.

- Referential integrity is required.

  XML columns cannot be defined as part of referential constraints. Therefore, if values in XML documents need to participate in referential constraints, the data should be stored as relational data.

- The data needs to be often updated.

  Currently, the users can update XML data in an XML column only by replacing full documents. If they need to frequently update small fragments of very large documents for a large number of rows, it can be more efficient to store data in non-XML columns. If, however, the users are updating small documents and only a few documents at a time, storing as XML can be efficient as well.

Chapter 3 XML Data Modeling

Maritime transportation industry is a high-complex industry; a fleet management company has to manage various ships with different characteristics divided by their duties (such as merchant ship, passenger ship, tanker ship, cargo ship, etc.). Moreover, within the ship, it should be divided into various departments (such as engine, deck, information, etc.) depend on their functions. According to these reasons, introduction of the distributed management mechanism is necessary. Unfortunately, the construction of ship industry is different from ordinary electronic business enterprise; the biggest difference can be analyzed as follows.

Though every activity under electronic commerce management mode is dynamic, but considering departments; each activity belongs to are still static. On the other hand in fleet management mode, ships can be treated as department within the whole construction. In this case, ships are not static, but need to launch out in indefinite alternation by actual requirements. While a ship is sailing offshore, the communication ways with overland are limited into quite a small range, it can be just used radio wave as usual communicate mean; recently, because of the advancement of technology, communicate with satellite is also available, but this technology is not widely popularized yet, and the cost is very expensive. So, it just can be treated as an emergency communication mean. Under these circumstances, using a traditional data construction such as relational database system to store and manage information, the data would be difficult to divide into a separate ship; because of the relationships between tables in relational database are complex. And also, it could cause the discontinuity and inconsistency of data.

Since the Extensible Markup Language (XML) is adopted as a Recommendation standard by World Wide Web Consortium (W3C) at 1998, it has become a very import emerging standard for E-commerce because of its flexibility and universality. Many software designers are actively developing new systems to handle information in XML formats. Some researches in navigation field have used it as a data exchange format for applications [93, 94], but they have not considered about the integration of data between ship and Ship Company.

In this chapter, I proposed to establish a ship data model by XML. By the characteristic of XML's tree-structure, a ship can be treated as a child node under the management of fleet management company, and each child node has its own independent XML sub-tree structure. Then, during the sailing, it can be managed by its own data structure independently.

Another important characteristic of XML is that elements and their attributes can be defined by user. In this research, the classified resources and Role-Based Access Control model [95] are combined to provide users appropriate resource access permissions. When a ship arrived in the port, an efficient data synchronization mechanism between the ship and the fleet management company is provided.

The rest of this chapter is organized as follows: First, Section 3.1 describes the characteristics of XML and presents how to establish the ship data model. Section 3.2 describes the data-update mechanism and synchronization procedure for fleet management. Finally, Section 3.3 describes the convenience and possibility of using XML Schema to replace recent online application form in the port administration level.

3.1 Data Storage Method

Web-based information systems no longer aim at purely providing read-only access to their content, which is simply represented in terms of Web pages stored in the Web server's directory. Nowadays, the employment of databases to store the content of a Web site turns out to be worthwhile due to new requirements emerging from several application areas such as; electronic commerce, etc.

In recent electronic business modes, the amount of users from various countries have been huge grown because of the popularization of the Internet. In order to handle with the diversification of information and the necessity of simultaneous access from huge users, the international enterprises mostly use relational database to store their data. The advantages of relational database can be described as follows:

- Reduce the duplication of data; data is independent and easy to establish applications to manage it.
- RDBS can provide the maintenance for the security and the consistency of data. It is also more efficient than simple file management.
- RDBS provides an index function for data access. When the capacity of data becomes very huge, the speed of data access could be much faster than directly accessing to files.

Database systems are well-known for consistent storage, retrieval, and manipulation of data. At the same time, XML is generally accepted as data description language for both Web-based information systems and electronic data interchange among different organizations.

Although RDBMS has many advantages, it is not so suitable for fleet management. The main reason is that different from ordinary enterprises, the ships under the management of fleet management company (could be treated as the departments within a general enterprise) are not fixed, but have the necessity of launching out and arriving in. While ships are sailing offshore, it is difficult to get the information from centric database through general Internet construction. If the user wants to allot related information to a ship before it launches out, it would be also difficult to achieve. Because the tables of relational database are related with each other and they are difficult to divide. In order to solve this problem, I proposed to establish the ship XML data model for fleet management and the detail will be described in next sub-section.

3.1.1 Fleet Management Mode

Fleet management can be seen as a tree-structure with fleet management company on the top, and various ships below under this management. So, its structure can be displayed visually by XML. The proposed data model treats the fleet management company as the root node, ships under management as child nodes; and each child node has its own affiliated data model. When ships berth in the port, the data model is a complete XML tree architecture. But, when a ship launches out, the partial tree data belong to the ship should be divided from the original XML tree. In addition, while this ship is sailing, it should upgrades to be a temporary complete XML tree and has its own root element (as shown in Figure 3-1). After this ship comes back to the port, based on the policies predefined, the synchronization between fleet management company and ships can be processed by exchanging corresponding child nodes data easily.

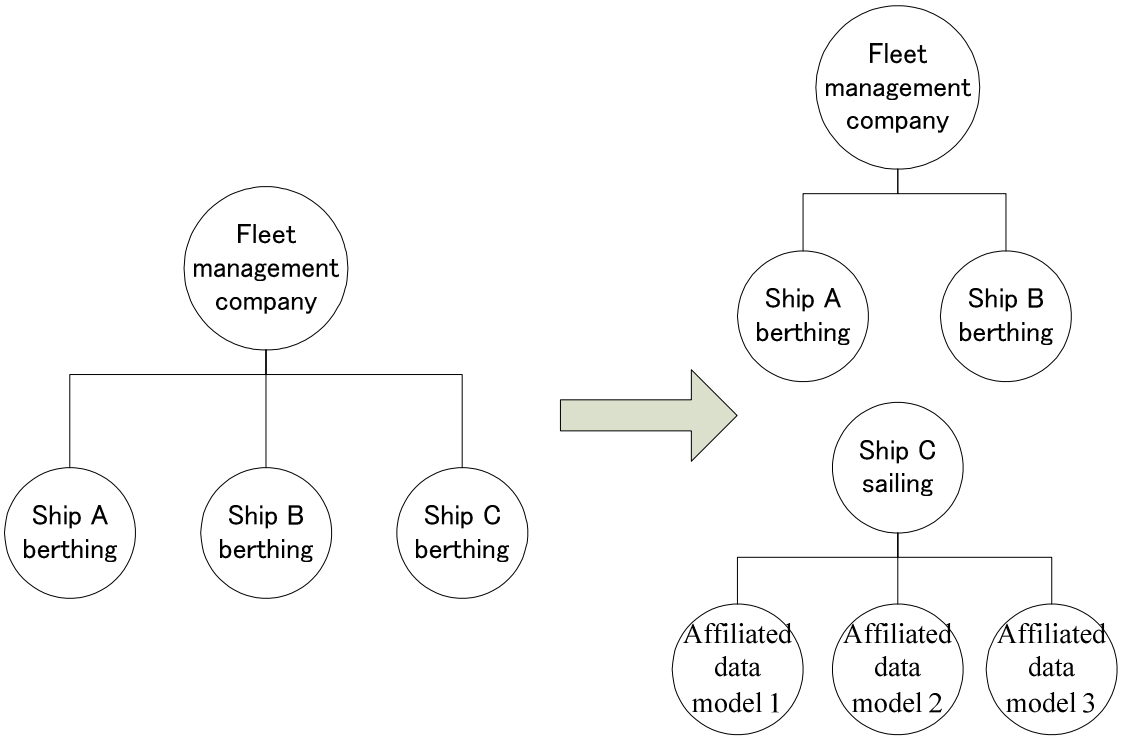


Figure 3-1 Separable XML tree architecture

3.1.2 Ship Data Modeling

In this research, I focused on cargo ships and passenger ships, the image figure of this ship data model is illustrated in Figure 3-2. In order to implement this model (under the various objectives, such as; procedures of arriving in, cargo clearance, immigration declaration, etc.), the data structure should include the following elements:

- Ship basic information: use to identify ship itself. Including name, description, flag, official number, length, draft, agent, operator, owner, net tonnage, etc.
- Navigation data: the navigation record data of all kinds of navigation instruments within the ship; including radar, gyro compass, GPS, speedlog, chronometer, etc.
- Schedule data: the schedule information of ship; including port of provenance, date of departure, port of call, date of arrival, time and date of entrance, berth, final port of destination, etc.
- Service metadata: the metadata of applications applied within the ship, such as watch sign-in management service.
- Cargo data: the information of cargo transported by ship; including marks and Nos., number and kind of packages, description of goods, gross weight, place of shipment, place of destination, etc.
- Passenger data: the information of passengers in ship; including name, nationality, date and place of birth, port of embarkation, port of disembarkation, passport number, etc.
- Crew data: the crew information; including name, nationality, date of birth, sex, number of seamen's book, rank, etc. It is similar with the passenger information except the information of duty.

The usability of XML is raised hugely because it is able to define elements and their attributes within the document. In this research, in order to classify the data and raise the efficiency of synchronization, two attributes were defined as property and update. The sample can be described as:

<Element property="public" update="true">

In the sample above, the “public” value of property means everybody is allowed to access to this resource. The “true” value of update means this element has been updated before.

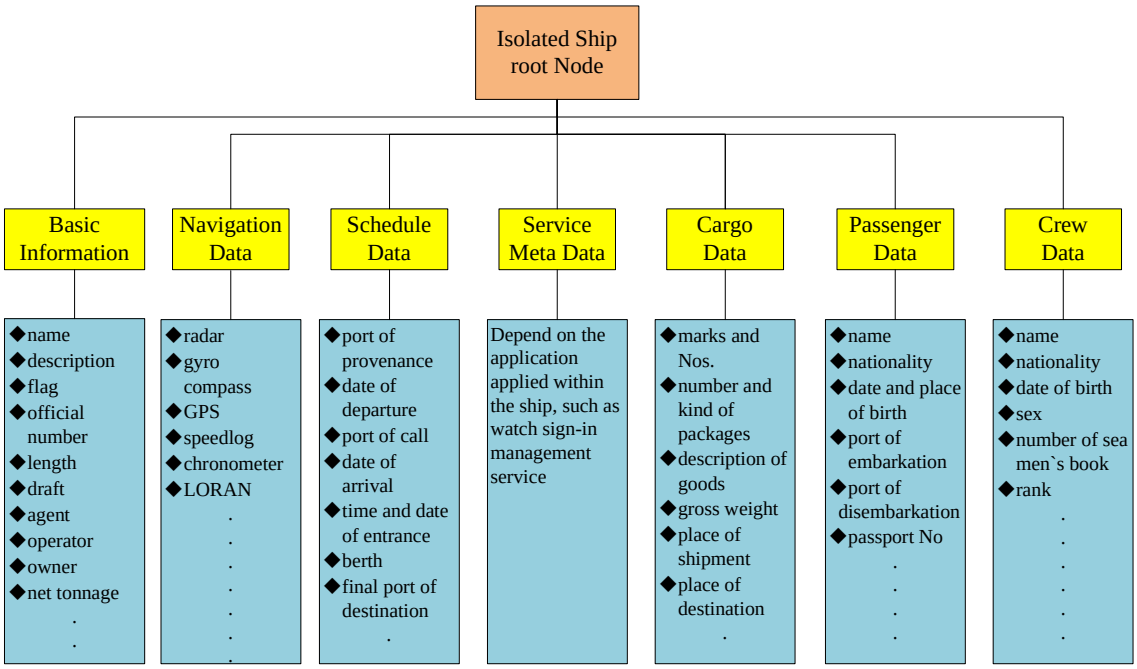


Figure 3-2 Ship data model

The number of code line in XML data model could reach several thousand due to the essence of XML construction depending on the actual situation. So then, it is not so efficient to read the whole file. In order to solve this problem and consider the security at the same time, Role-Based Access Control (RBAC) model [95] is proposed. The essence of RBAC is that permissions are assigned to roles rather than to individual users. Roles are created for various job functions, and users are assigned to roles based on their qualifications and responsibility. With this way, the task of specifying user authorization is divided into two logically independent parts: one of them assigns users to roles and another assigns access rights for objects to roles. Users can be easily reassigned from one role to another without modifying the underlying access structure. RBAC is thus more scalable than user-based security specifications, and greatly reduces the cost and administration overhead associated with fine-grained security administration at the level of individual users, objects, or permissions. RBAC model can be simply expressed as the form (s, o, a), where “s” is a subject (user), “o” is an object (resource) and “a” is an access right (permission).

By using the RBAC model and the property defined in this ship data model, we can achieve the objective of distributed management. For example, passengers are only allowed to access the general public information, such as; the schedule data within the ship data model. This can not only maintain the security, but also can raise the efficiency by reducing the capacity of data reading. By controlling the permissions of writing in data, we can also make the information belong to each department be in charge by the chief of this department. Such as the chief engineer can renew the operation information of all machines in the engine room, but the chief officer can only read this information. When a ship arrives in the port, if we allow the agents of government departments accessing to the ship data model, we can also assign the corresponding information by their duties to them, such as; a custom agent can access the related cargo's information to apply the cargo clearance procedures, an immigration agent can access the related passengers' information to apply the immigration declaration procedures. The image figure is shown in Figure 3-3.

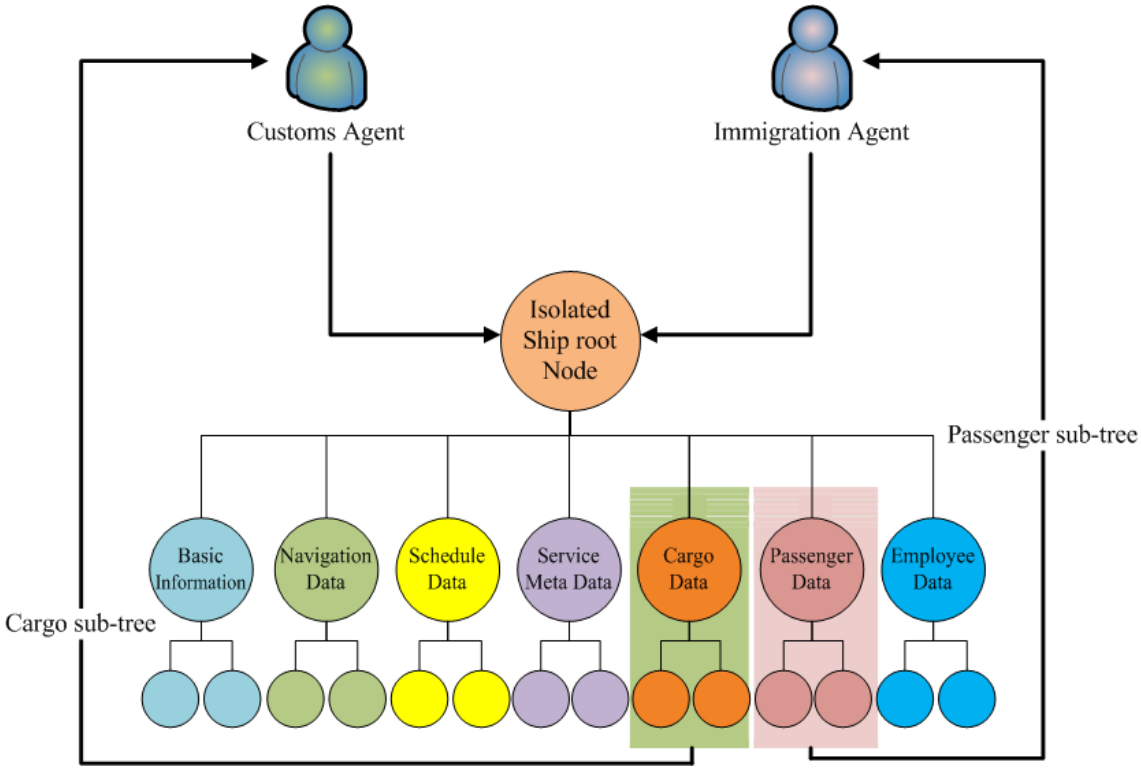


Figure 3-3 Corresponding sub-tree according to duty

3.2 Data Integration

During the sailing of ships, the data (such as sailing schedule, navigation instruments records) could have the necessity that should be renewed because of various unpredictable factors (e.g. weather, hydrographic condition, etc.). In this research, the ship is treated as an independent individual. While the ship is sailing offshore, it should manage its data by itself, and processes the data integration with fleet management company when it arrives in the port. The old ship data that fleet management company has should be read only during the ship is sailing to prevent from destroying the consistency of data. Because XML is adopted to establish data model and treated the ship as a node, the most simple synchronization mechanism is replacing the old node data of fleet management company with the renewed ship node data. But, this mean is not efficient when the capacity of data is huge, so I propose a more efficient data synchronization mechanism in this section.

The proposed data integration mechanism should be divided into two parts. The first part is the data update during the sailing of ship; the second part is the synchronization mechanism when the ship arrives in the port as explained in the following sections.

3.2.1 Data Update During Sailing

During the sailing of ship, there is a possibility that data could be change because of various unpredictable situations (such as the change of weather or hydrographic condition, engine problem, human factors, etc). In order to record the path of data updating and achieve an efficient synchronization later, an update attribute is defined for XML document elements. Within the XML data model I proposed, if any information of node $N_{i, j}$ is updated (i denotes the i^{th} layer of the whole XML tree, j denotes the j^{th} node of the current layer), then the update attribute of $N_{i, j}$ is set to be “true”, and at the same time pass the value of “true” to the update attribute of its parent node $N_{i-1, parent}$ (the direct parent node of $N_{i, j}$ within $i\text{-}1^{th}$ layer of the whole XML tree). If the parent node $N_{i-1, parent}$ is not the root node of the ship data model, then repeat the processes above. In this section, I call this mechanism as “upward parent node chain update”. The algorithm of it is shown in Figure 3-4; the image is also illustrated in Figure 3-5.

Upward parent node chain update algorithm

```
If ( Node i,j .update == true ) {
        While ( Node i-1,parent != Node root ) {
                Node i-1,parent . Update = true ;
                i = i -1 ;
        }
}
Node root .update = true ;
```

Figure 3-4 Upward parent node chain update algorithm

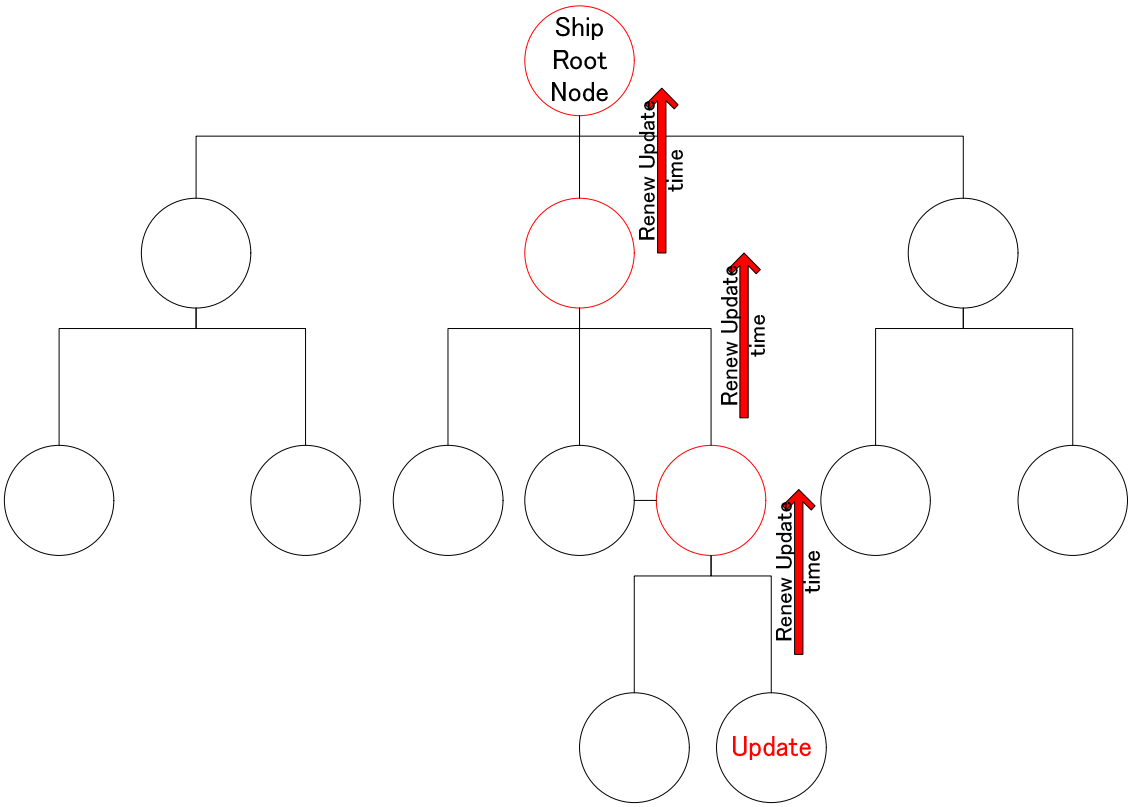


Figure 3-5 Image of upward parent node chain update

3.2.2 Synchronization when Arriving in Port

When the ship arrives in the port, in order to maintain the consistency of data, the synchronization with fleet management company should be applied. The procedures of the proposed synchronization mechanism are described as following:

- Step.1:

  Read in XML Schema to get the structure of entire XML document, mostly the relationship among nodes in different levels.

- Step.2:

  Compare the data that fleet management company has with the data of ship just arriving in.

- Step.3:

  If the data of ship has been updated, then start to search the child nodes of the first sub-layer of the ship root node.

- Step.4:

  If the j^{th} node of the current layer is not updated, then ignore all sub-trees below it and move to $j+1^{th}$ node.

- Step.5:

  If the j^{th} node of current layer is updated, then start to search its child nodes of its first sub-layer.

- Step.6:

  Repeat Step.4 to Step.5 till the whole XML-tree has been searched completely.

According to these procedures, the data synchronization flowchart is illustrated as shown in Figure 3-6.

By combining those two procedures of Section 3.2.1 and Section 3.2.2, an efficient XML data synchronization mechanism is proposed. In fleet management level, the ships can manage their own data structures while they are sailing offshore; and when they arrive in the port, the synchronizations with the fleet management company would be applied to maintain the consistency of data.

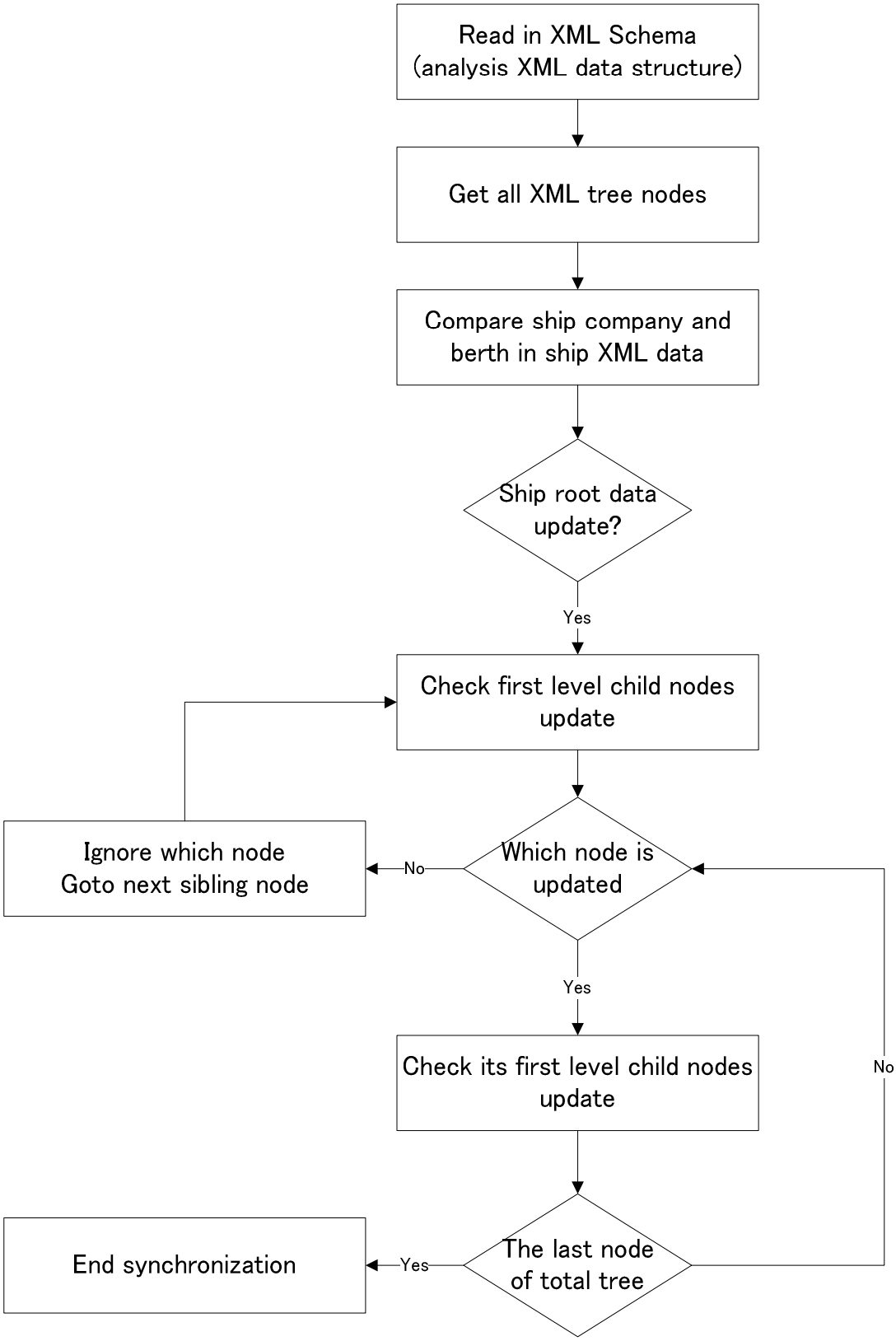


Figure 3-6 Flowchart of Data synchronization

3.3 Potential Implementation for Port Administration

When a ship arrives in a port, it should apply many complex procedures with various government departments. The main procedures of arriving in include the cargo clearance of Custom, the immigration declaration, the quarantine of the Ministry of health, Labour and Welfare, the clearance inward of each port, etc. Furthermore, each procedure need to fill in various applying tables, such as; the cargo clearance need to fill in general declaration, cargo declaration, ship's stores declaration, passenger list, crew manifest, crew list, etc. In order to simplify these complex procedures, Japanese government are processing the optimization project [96] of the import, export, and clearance inward procedures. The objective is to combine the ongoing multi-systems (such as Nippon Automated Cargo And Port Consolidated System, port Electronic Data Interchange, etc.) to complete the procedures, such as; clearance inward that has to apply to multi-government departments with one single window and one time data input/transmission for international ships. Additionally, it would be much more convenient for users and also it can reduce the costs and applying time. For now, these application procedures can basically be processed to input various data with Web browser as the input interface through the Internet. Comparing with old human operation era, we can say that it is already a very convenient way. But, if we can use the characteristics of the authentication of XML document, there are still some potential to further simplify human inputting processes. The possibility of implementation of the XML Schema (an important XML document authentication mechanism) for port administration is described as follows.

XML schema is a standard that was approved as a World Wide Web (W3C) Recommendation on 2 May 2001. It can be used to express a schema: a set of rules to which an XML document must conform in order to be considered validation according to that schema.

XML schema express shared vocabularies and allow machines to carry out rules made by people. They provide a mechanism for defining the structure, content and semantics of XML documents. The XML Schema definitions for XML documents include:

- The vocabulary: element and attribute names.
- The content model: relationships and structure.
- The data types: data entity format (such as string, Boolean, integer, etc.).

If an XML document can be valid by a specific XML Schema document, then it can be said that this XML document conform every restriction for our purpose, and it is a valid XML document.

Take the Kobe Port EDI system as an example here. When a ship arrives in the port, it could fill in corresponding tables through the Internet to apply various services (such as licensing of berthing facilities, watering application, licensing of wharf site, etc.). Each column in the electronic form has its own format restriction to prevent for errors of storing data into database from user miss inputting.

As an implementation sample of XML Schema, take the online electronic applying form of Kobe Port EDI system wharf site license for the example (as shown in Figure 3-7). According to the format restriction of each column, we can establish the corresponding XML Schema that defines the data format of every column in the form (as shown in Figure 3-8); and then we can establish a corresponding XML document with this XML Schema document.

The format restriction of XML Schema is very regular, so when user transmits the applying XML document to corresponding government department, if it can pass the check mechanism of XML Schema, then the government department can consider that the content of this document is correct. There is no need for human rechecking and the data of this document can be directly storing into the database.

This method can reduce the waste of the Internet resource in one side; and in other side it is possible to simplify applying procedures. Cooperating with the proposed ship XML data model, it could be considered as a potential implementation for port administration.

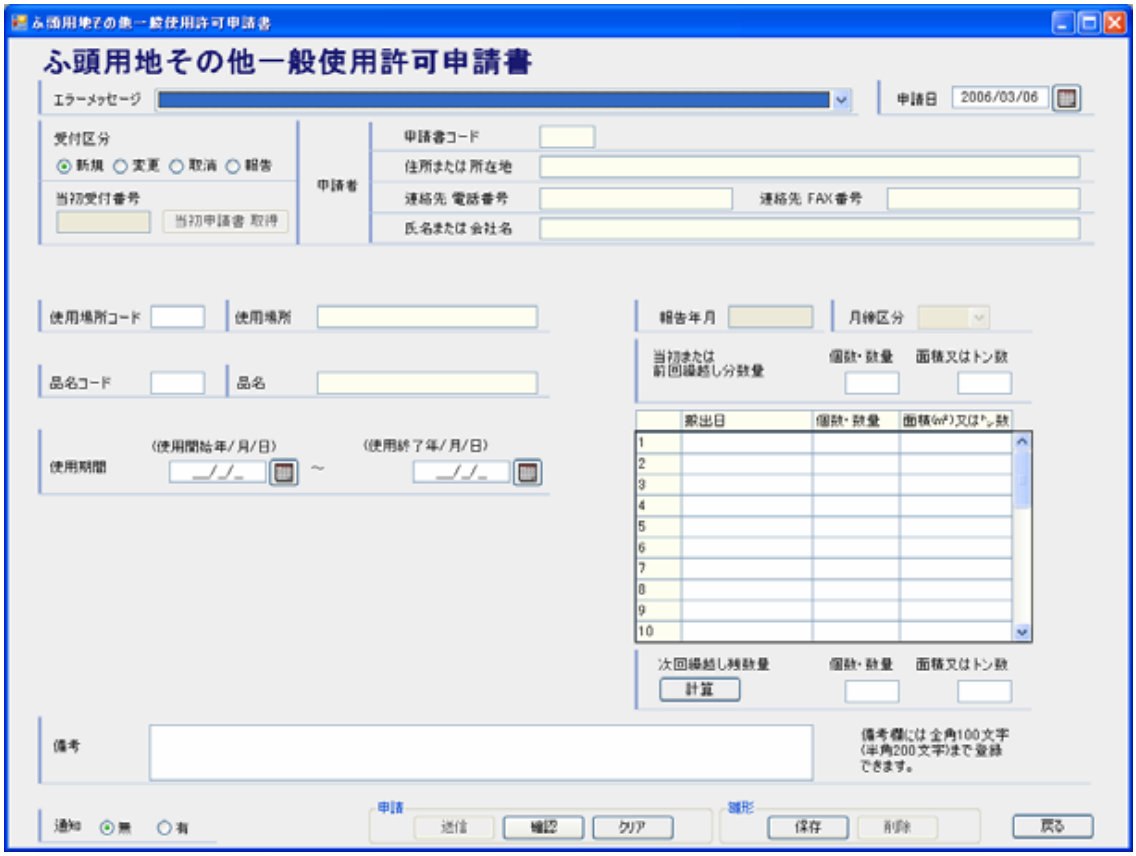


Figure 3-7　Kobe port on-line wharf site application form

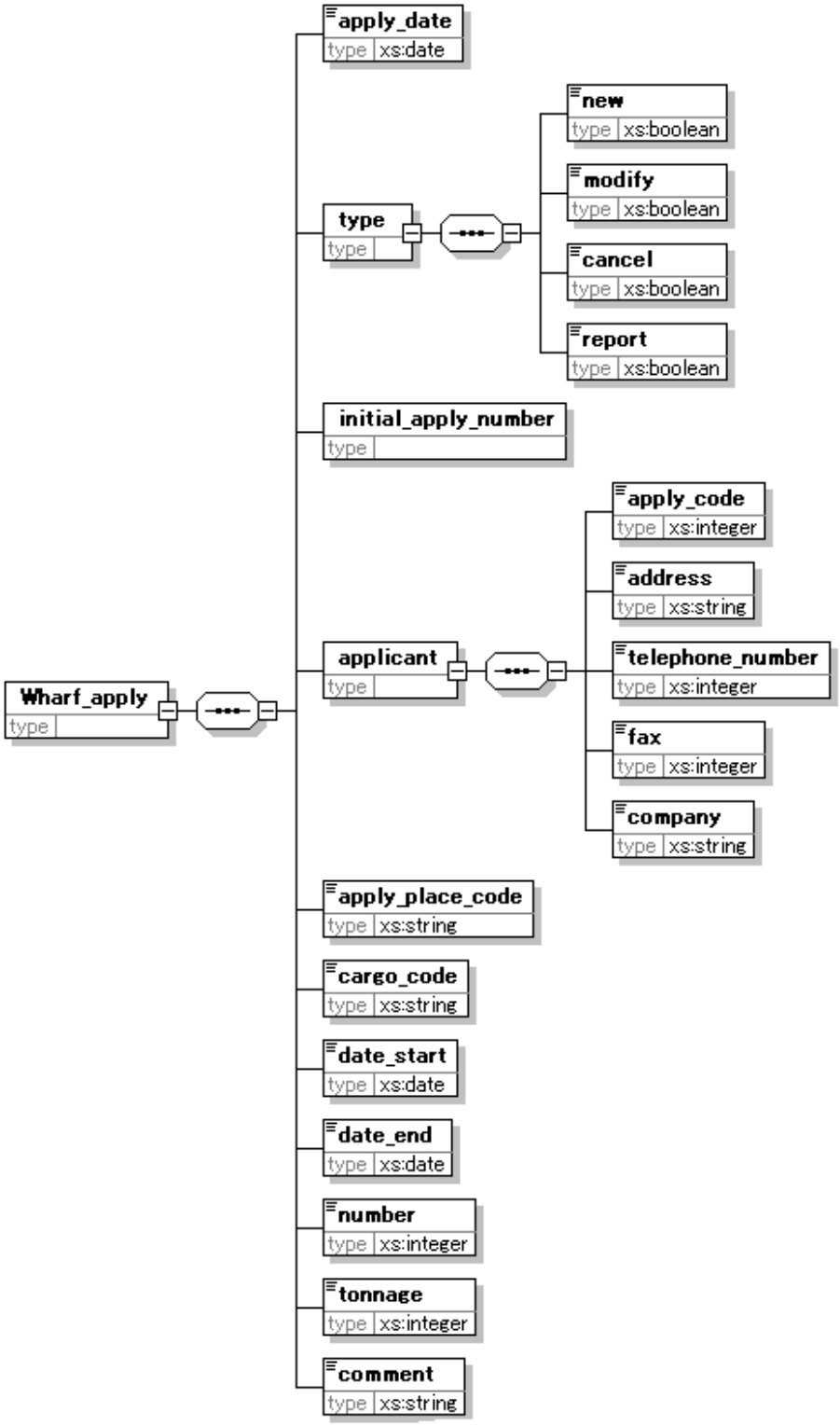


Figure 3-8 Corresponding XML Schema definition

Chapter 4 Context Model for Services

Context aware ubiquitous content access is characterized by providing intuitive ways for accessing Web content based on users surrounding context. Context is referred to as any information that can be used to characterize the situation of an entity where an entity can be a person, place or a physical/computational object [97], and the information can be where and when for Web users are, as well as what the Web content are available nearby, and etc.

There are many research efforts for the development of context-awareness toolkits including HP's Cooltown project [98], Dey's the Context Toolkit [99], the CB-SeC framework [100] and the Gaia middleware [101]. These toolkits either provide functionalities to help service requesters obtaining services based on their contexts or enable content adaptations with user's contextual information. In order to promote context aware service providing, Kouadri et al. [102] proposed a formal definition of service context to model service's contextual information. Lemlouma et al. [103] proposed a framework to assert metadata information of Web contents. They used Composite Capabilities/Preferences Profiles (CC/PP) as interoperable context representation to enable communication and negotiation of device capabilities and user preference in terms of a service invocation. Besides, several OWL-based context models are presented [104, 105, 106] to provide high-quality results of service discoveries beyond the expressive limitations of CC/PP. They utilized the ontology to describe contextual information including location, time, device, preference and network, etc. By combining semantic contextual information with inductive or deductive techniques, they can perform matches against both user and service's context semantically.

In the maritime field, after a ship launches out of the port, it can be seen as an independent unit separating with land. Everything happened within it must be taken care by cruises themselves. Under this kind of situation, real time information transportation and appropriate service management are very necessary. In this chapter, I use XML to define the context model and the user model, and then use them to establish the proposed system. The context can be simple information or a specific service, each of them limits that only specific users/groups are allowed to access it. The user model defines the group that users belong to, and use it to grant corresponding permissions by identifying specific individual. In the passive providing information level, firstly the

public part of description session is captured in the context, and then translated them into general HTML webpage by XSL Transformations (XSLT) [107], finally presented it to all users. In addition, in order to transmit real time information automatically, the above procedures are repeated in a fixed time interval (5 minutes period is used here). A user can also access the context to get extra information, but under his permission limit by authenticating himself. In the active providing information level, we can trigger services to transmit specific information (such as operation initial time or the schedule of next day) to related users/groups by real time, and then we can prevent operations be delayed or even more serious situations happen.

4.1 Web Service Life Cycle

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

A Web service is viewed as an abstract notion that must be implemented by a concrete agent. The agent is the concrete entity (a piece of software) that sends and receives messages, while the service is the abstract set of functionality that is provided.

While a Web service is applied, its activation is the process of creating or making a Web service instance active to serve a Web service request. The way in which the activation process controlled is called as the activation policy. The process of activation enables a Web service instance hosted by Cape Clear Server to hold session state.

Following are the supported activation policies and the main attributes of each policy:

- Application

  One instance of the service is created to handle all requests. If the timeout is set to zero, the same instance will always be in memory. If the timeout period is set to some non-zero value, the instance will be removed from memory if it has not been used for a period of time equal to this timeout period. When a request is received and the instance has expired, a new instance is created. The service implementation must be thread-safe. This model should be used where only application state or no state is required, and performance is important.

- User

  An instance of service is created to handle all requests for a particular user. The client must first be authenticated with the server to use this policy type. The instance is removed from memory when no requests have been received from the user for a period of time equal to the timeout. A user may be logged into the server more than once simultaneously, but effectively the user has only one active session. Each request for that user will always use the same instance. The user activation policy uses the user name principal as a basis for allocating Web service instances. Use this model when only one instance is to process requests for a particular authenticated user.

- Session

  An instance of the service is created to handle all requests for a particular user session. This does not require a client to authenticate with the server first. A user can have many sessions in active simultaneously. If a request has no session ID, a new session ID is created and used to associate the instance with the session. On response, for the HTTP transport, the session ID is stored as a cookie on the client. All future requests from the client will send the session ID cookie, ensuring that the correct instance is invoked on the server. Use this model if general session behavior is required and the service must maintain state, or to increase performance by allocating a single instance to a particular session.

- Request

  An instance is created to handle each request. After handling the request, the instance is discarded. Use this model if no state is required, the implementation is not thread-safe, and performance is not critical. This policy is the most expensive, due to it causes more objects to be created (and ultimately garbage-collected), and each instance must be initialized on every request. This is the default policy assigned to Web services.

I would like to introduce the concept of service life cycle in this chapter. When a service is used, it can be realized by a concrete provider agent. The state of a service is shown in Figure 4-1.

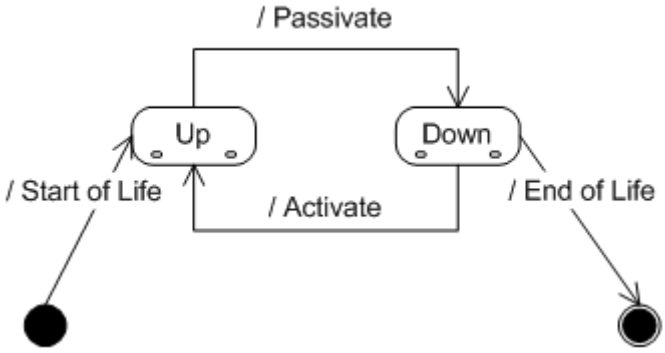


Figure 4-1 Web service state

The definitions of state up and down are described in the following:

- UP

  The provider agent is capable of accepting and processing requests (i.e. the service is available).

- DOWN

  The provider agent is not capable of accepting any requests (i.e. the service is not available).

In this chapter, I use XML data model as the metadata of service, a time trigger as the agent of service to apply a time-dependent information providing system.

4.2 System Architecture

The concept of metadata is wide spread in computer science. A general definition is “data about data”, i.e. data that describes the original data. It improves the value of operational data by giving applications and users additional information on the data’s origin, its precision or staleness.

Context models store data on the user’s context: location, surroundings (what is around him and close-by), intentions, activities, etc. So far, most context models focused on providing only the data that reflects the current situation in the real world. In this chapter, I go one step further and extend the scope of context models to also contain data about data, so called metadata.

Metadata helps applications to assess the data’s quality and to select the desired data in a fine-grained fashion. Storing the metadata explicitly in the context model increases its flexibility and facilitates the integration of many local context models into a global one. This in turn enables many applications and data providers to share a lot of data, which offers applications a larger-than-ever pool of data and cuts down deployment times and costs.

The objective of this research is to provide a contextual architecture with time event driven facility. Context model is used to describe the behaviors of services, time trigger to manipulate specific operations, user model to deal with user/group permissions, and exception handler to deal with unexpected situations. I used not only user model to manage the permission for each context, but also extended context model with the ability of dealing with some timely operations by trigger services in real time.

In the security level, although authentication is an important step toward authorization, the specific authentication mechanisms are not the focus of this chapter. Any authentication mechanism that securely transmits the credentials to the user can be adopted. In such a manner, these credentials cannot be tampered with stolen, or forged. Given these properties, the access control can rely on the credentials to identify the requester.

Figure 4-2 illustrates the overview of the system. It is composed of context model, user model, automatic update, time trigger, and exception handler.

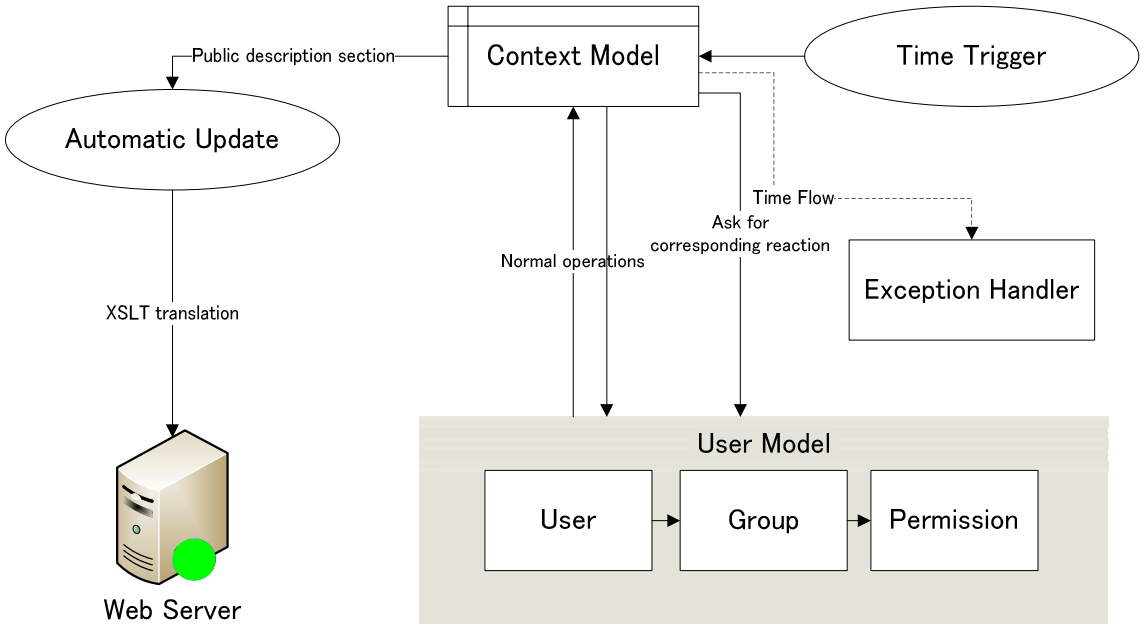


Figure 4-2 Architecture overview of proposed system

Through the user model I proposed, a person can be authenticated as a regular user, and get the permissions according to the group which belongs to. Normally, users are free to connect to the resources under their permission at any time and any place. Also, through a Web server, public information updates itself automatically to ensure all users receive the newest information.

At the same time, when a context's initial condition matches with real time, time trigger will initial corresponding services and ask related users to take reaction default designed in system. If the corresponding users could not accomplish the operation on time, the exception handler will be initiated to remind them continually.

4.2.1 Context Model

Users are now living in a ubiquitous computing environment. They are able to access information or use computing services while they are doing their everyday life activities. Researchers have tried to solve the problem by improving user interaction by exploiting information relating to users, devices and environments, through the notion of context awareness. In order to use context in developing context aware applications, the notion of context has to be well understood. Different researchers have different points of view of context; I would like to discuss the implementation for services under intranet structure environment to apply within a sailing ship.

The context model of this research is used to provide information in real time. It can be divided into two levels: simple information or information attached by service/operation. The biggest characteristic of the context model I proposed is that it extends time features from normal context models, so we can apply a background time trigger according to real time and deal with some time-dependent exceptions.

The proposed model requires three key elements.

- A User is assigned belonging to a group with specific permission to access resources under his privilege limit within system.
- Contexts can be triggered automatically by comparing with real time.
- Contexts require specific users/groups with permission to handle with.

By following above principles, each user belongs to one user group. Similarly, each resource will assign a permission subset from the entire permission set to each group that has a privilege to access it.

Table 4-1 illustrates the features of the context model: name, entity, description, owner, physical location, attribute, property, current state, relevance and time. Time factor is used to trigger contexts and provide real time information, including operation initial time, exception initial time, exception handle time interval and destruction time.

Table 4-1 Context model architecture

Contextual Element		Content
Name		Context (service/information) name.
Entity		If the context is a service, this should address to where the entity of service actual exist.
Description		Context content description, it could be used to announce necessary information to users/groups.
Owner		The owner (The highest permission user) of this context.
Physical Location Limitation		Limitation of where this context can be accessed. If there is no limitation, then the value should be "null".
Attribute		Simple information can set attribute as "information". Specific service can set attribute as "service", which can be triggered by real time.
Property		Define the property, it could be "public" or "private", if the context is private, it could be further divided to the groups.
Current State		It shows this context is available or unavailable.
Relevance		The context could be related to specific users/groups, only legal users/groups under permission limitation are allowed to access it.
Time	Inform time	Before context initial, the time that inform related member.
	Operation initial time	The start time of a service.
	Exception initial time	The time of exception handler start to be applied.
	Exception handle time interval	The time interval of re-check service situation.
	Destruction time	The time of this context fails.

In the proposed context model for service, I extended the time factors, then we could manipulate the service and detect exception situation by real time, also this context model is able to describe the ordinary behaviors of the system. The Schema design of context model is shown in Figure 4-3.

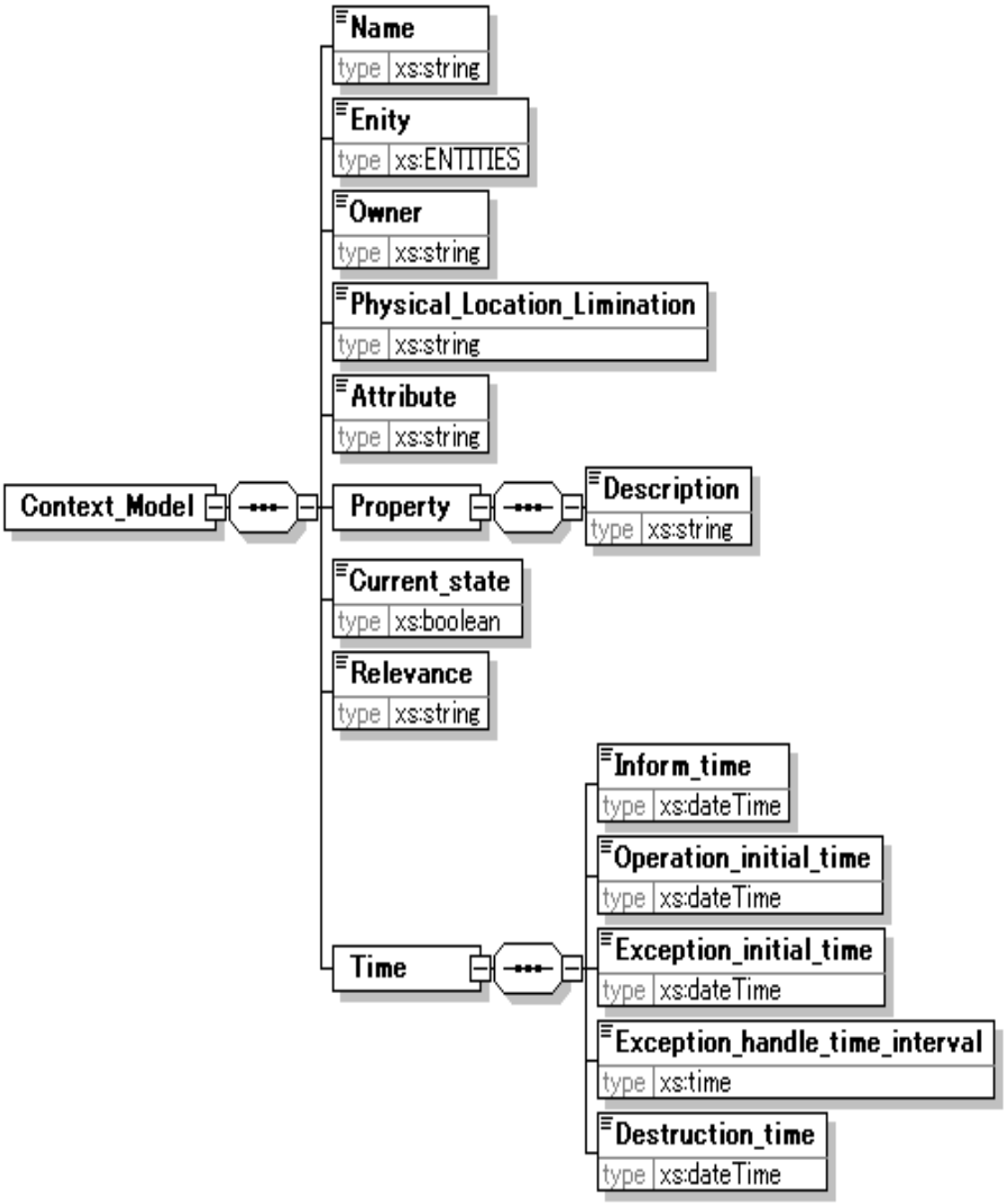


Figure 4-3 Design diagram of context model schema

4.2.2 User Model

The definition of user model can be given as follows [108]:

- A user model is a knowledge source in a natural-language dialog system which contains explicit assumptions on all aspects of user; this may be relevant to the dialog behavior of the system. These assumptions must be separable from the rest of the system's knowledge.
- A user modeling component is a part of dialog system, whose function is to incrementally construct a user model; to store, update and delete entries; to maintain the consistency of model; and to supply other system components with assumptions about the user.

In this research, I establish the user model for passengers and crews within the ship. The focus was the identification of users, and its modification was out of the objectives in this chapter.

In order to identify the user, the user model includes information relevant to the group, personal information and surrounding environment (physical environment, software and hardware environment). When a user access to the context, the system can identify him/her and determine if a desired context is available to the specific user by the permission of the group he/she belongs to.

The Schema design of user model is shown in Figure 4-4 and the details are shown in Table 4-2.

Table 4-2 User model architecture

Element	Content
Group	Decide which group the user belong to
Name	User name
ID	User identification number
Title	Duty title
PhoneExt	Office phone number
IP address	The IP address that user use to connect to internet/intranet
Email	User mail address
Physical Environment	Location
Hard environment	Device capabilities, bandwidth, processor speed, storage capacity, resolution, etc.
Software environment	Operating System

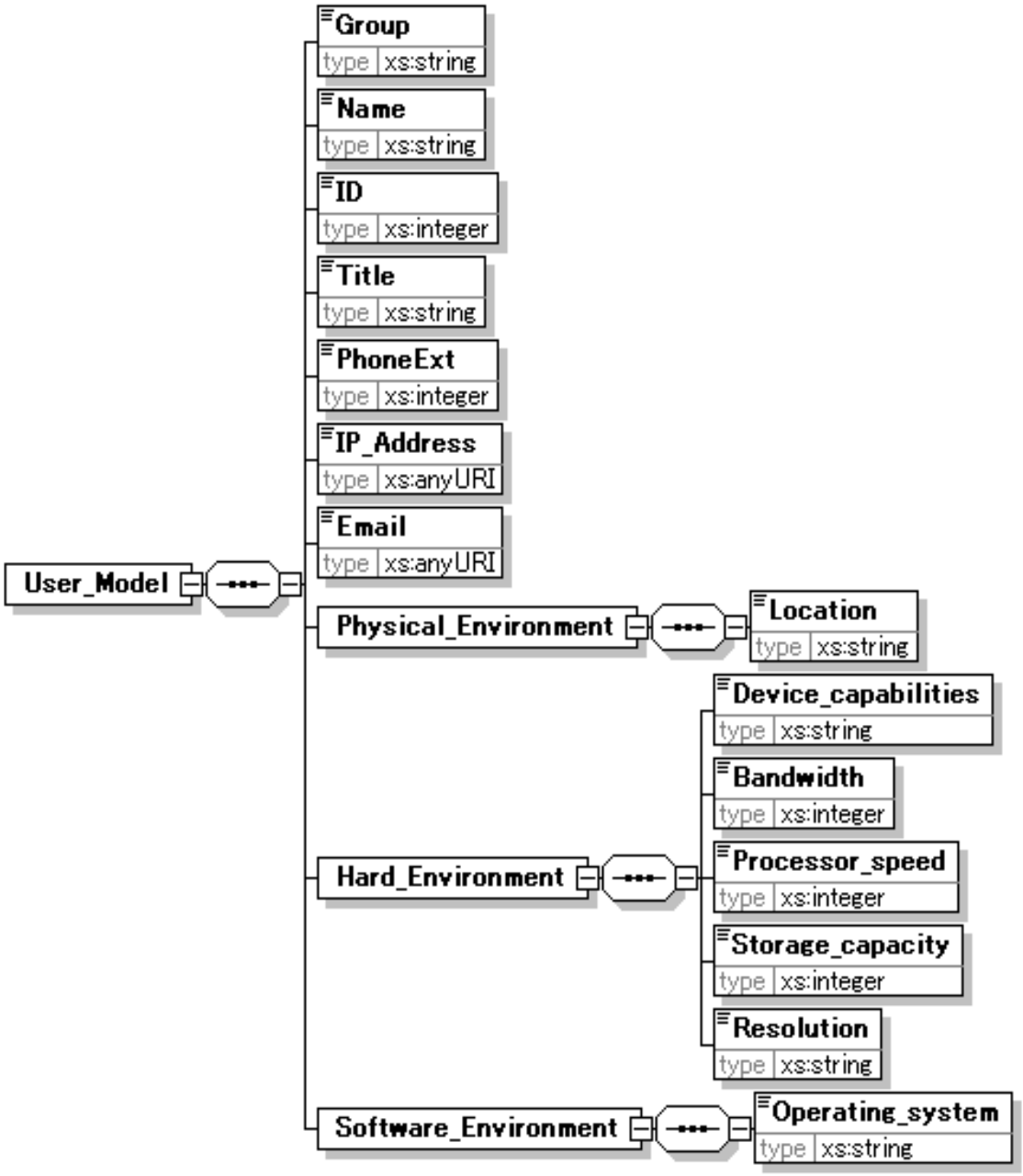


Figure 4-4 User model schema design diagram

4.2.3 Providing Information Passively

In this chapter, two ways to provide information passively are proposed; one is present with Web server and the other is user access to the context himself. The details will be described below:

- Automatic webpage updating

  When we think of the Web, the first thought is the Hypertext Markup Language (HTML). It is so popularized because HTML document is easy to create and use although its structure is lack of preciseness. But, in order to go further into the Business-to-Business generation, XML standard was established. XML document has the advantage of the advanced interaction between servers, but contrarily it is not so easy to read by human being. To get the full benefit of XML, the W3C started developing a standard language for presenting information held in XML in 1998. This language was named Extendible Stylesheet Language (XSL). The goal of XSL was to develop a stylesheet that could overcome the limitations of Cascading Style Sheets (CSS) to restructure information and add things as heading to a page.

  One extension of XSL defines how to transform from any XML-based markup language into another markup language (or into a plain text). This language is known as XSL Transformation, or XSLT. An XML document that is written in XSLT is commonly known as an XSLT stylesheet. Each XSLT describes how a set of XML documents (the source documents) should be converted into other documents (the result documents, such as HTML). Usually, an XSLT stylesheet will take source documents written in one particular markup language and produce a result in another markup language that can be used by a specific application, such as HTML for presentation in a browser.

  Within this context model I proposed, the description session and public property are set, so then the private/secured information can be kept safe, but only public information is captured to be published. After the partial XML document is captured, the XSL Transformation is used to convert it into general HTML webpage and published it to all users with Web server. In order to provide the newest information, the system set a fixed time interval to recapture the context model for converting and publishing, and then information can be provided in real time. The architecture of automatic updating is shown in Figure 4-5.

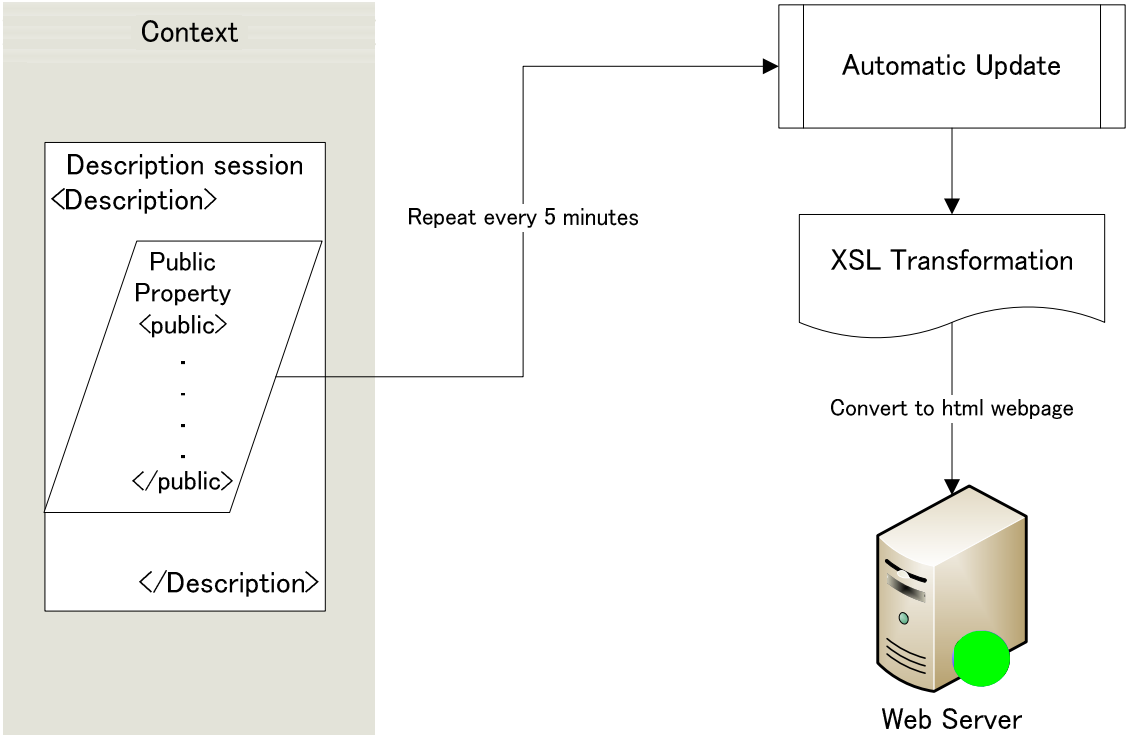


Figure 4-5 Update Automatically with XSL Transformation

- User active access

  Within the user model, each user is assigned to a specific group. On the other hand, within the context model, each context is designed to allow specific groups accessing to it. As a result of this, the objective of access control by granting users/groups appropriate permissions could be archived.

  In addition, the attributes/properties are assigned within the context model, a user under authenticated can only be allowed to access the public information and the information under his permission limit in his group, so then the security of context can be protected. The architecture of user active access is shown in Figure 4-6.

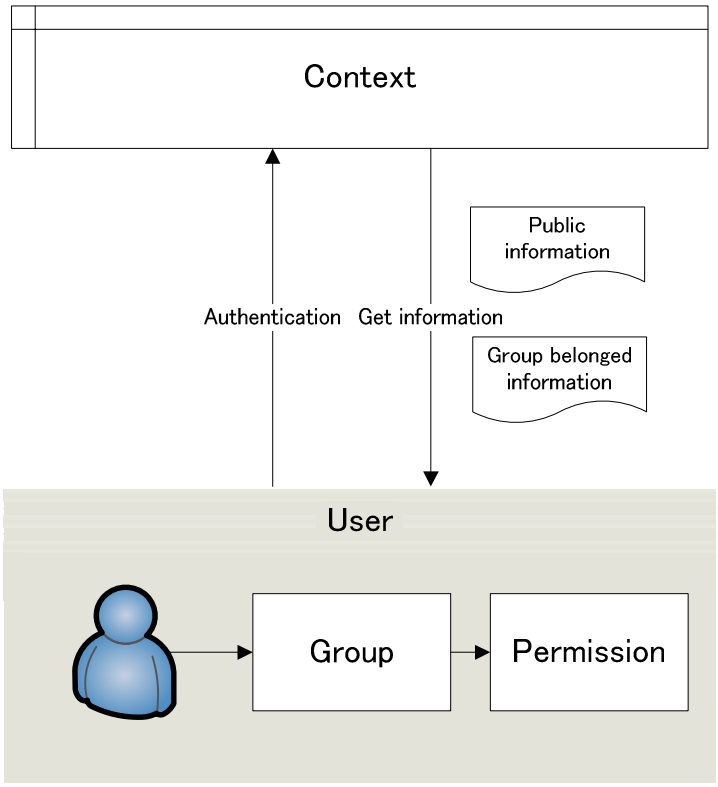


Figure 4-6 User active access to get extra information

4.2.4 Providing Information Actively

The context event could be triggered in real time due to the time attributes in the context model have been set. The facilities of time factor elements are described as follows:

- Operation initial time: the start time of a critical operation.
- Exception initial time: after a critical operation initiates, the time of exception handler initiates if the operation is not handled on time.
- Exception handle time interval: the time interval of different exception policies being applied. Along with time passing, if a critical operation is still under an abnormal situation, the corresponding exception policies will be applied by different time interval.
- Destruction time: the time that an operation fails. If the ongoing real time crosses this destruction time, this situation would be treated as a serious error in the system. All services in the system would be temporary paused, and the system administrator would be informed to deal with the error immediately.

The time trigger event flow is shown in Figure 4-7 and the details are described below. Also, in order to ensure that the operation is manipulated correctly, a time-dependent exception handle policy is established. The algorithm of the exception handle policy is described in Figure 4-8.

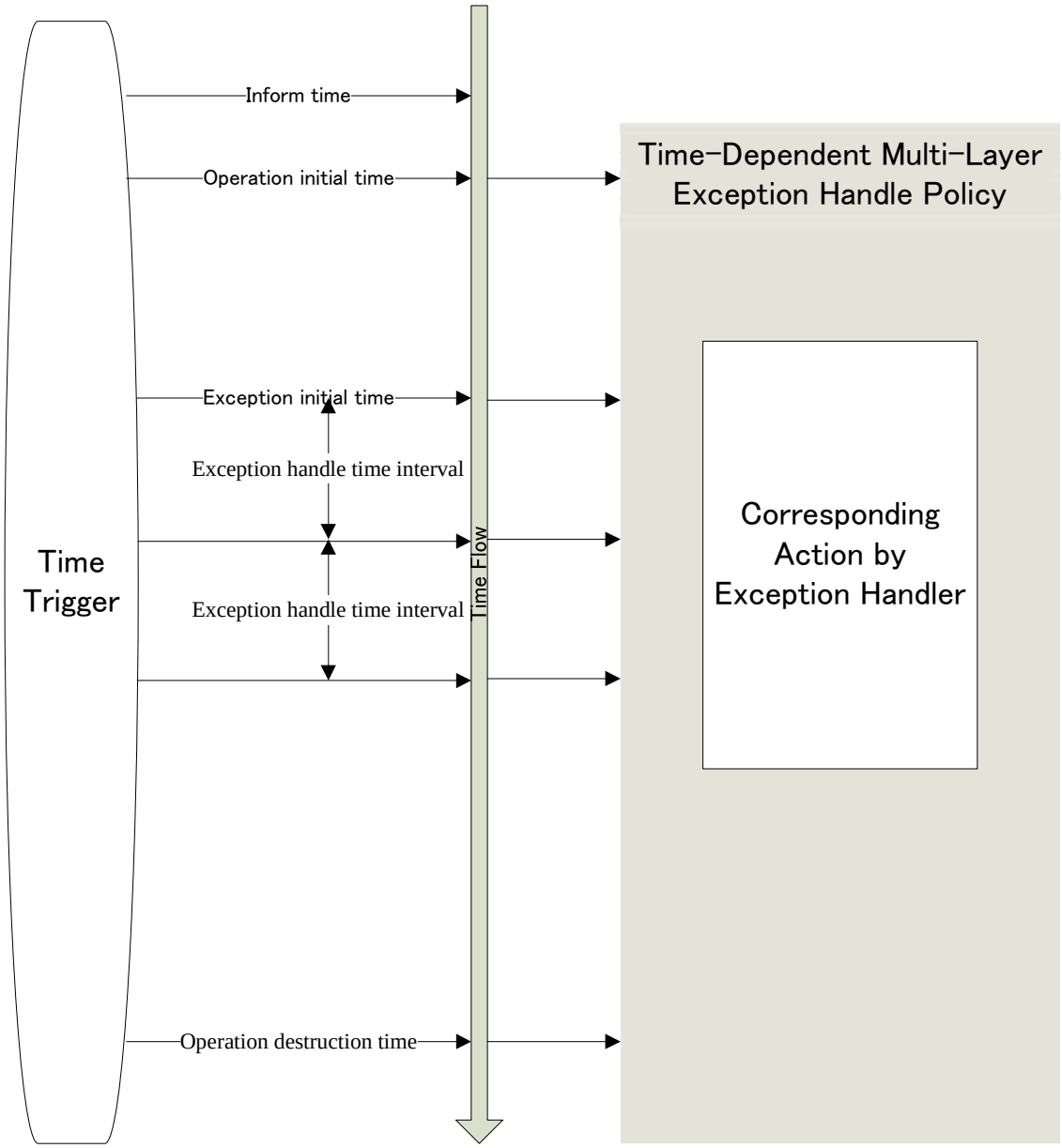


Figure 4-7 Time trigger event diagram

```
operation initial time matches with real time {
   operation initiate;
}
   If ( operation destruction time matches with real time &&
        system.current_state == abnormal ) {
   system.false==true;
   inform system.administrator;
   break;
   }
   exception initial time matches with real time{
      if (corresponding reaction is taken ) {
         system.current_state = normal;
         break;
      } else {
         system.current_state = abnormal;
         exception policy. apply ();
         next exception check time = exception initial time +
         exception handle time interval;

         while ( system.current_state != normal ) {
            if ( next exception check time matches with real time ) {
               if (corresponding reaction is taken ) {
                  system.current_state = normal;
                  break;
               } else {
                  exception policy level . apply();
                  next exception check time = exception initial time +
                  exception handle time interval;
         }
      }
   }
```

Figure 4-8 Time-dependent exception handle policy algorithm

If the context is simple information, the system can actively send information to related users/groups, and also users of different groups will receive corresponding information. For example, within a passenger ship, general passengers can receive the time/schedule information of the next event held with ship, and the employees can receive the deployment information of this event.

If the context is a specific service, first of all, system will inform related users/groups the initial time of the operation at the assigned inform time, and then initiate service for users to operate it at the initial time of operation. Some important operations in whole system may need to be handled by specific reaction (such as scheduled inspection operation). If the system could not get the corresponding respond even after the default operation exception initial time, it will send an instant message to the user and ask for corresponding reactions. Along with time passing, if system is still in an abnormal situation, it will send the warning message to the related users by fixed exception handle time interval. If the real time has cross the default operation destruction time, then the service will be closed and inform system administrator to deal with this abnormal situation.

4.3 System Implementation within Passenger Ship Scenario

A prototype system that has a sign in service had been implemented to simulate a scenario within a passenger ship and described the behaviors of model. In the prototype system, the following components are implemented: passive information providing, active information providing, and exception handling time triggered event.

Herein the scenario; It was assumed that there was a sailing passenger ship with many passengers and employees inside. Time now was PM 02:00, there was going to be a speech held by captain at PM 03:30, and the speech would be held in hall, three employees should start preparing at PM 03:00 before the speech began, all three of them were asked to sign in the service to make sure everything was on schedule. The information about this speech was set to inform everyone at PM 02:30. The operation destruction time was set to be PM 03:15.

The context model and user model are established by following situations, the sample models are illustrated in Figure 4-9 and Figure 4-10. Figure 4-9 shows the architecture of sign in service, including the information that should be published and the information should be kept inside, related members, location and exception handling time. Figure 4-10 shows the detail of specific user, including various kinds of personal information. It is used to identify the user and also connect with him/she.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context_Model xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <Name>Sign_in</Name>
   <Enity>/service/sign_in</Enity>
   <Owner>Administrator</Owner>
   <Physics_Location_Limitation>Hall</Physics_Location_Limitation>
   <Attribute>service</Attribute>
   <Property>
      <Description>
         <public>
            Title:Welcome speech
            abstract:.....
         </public>
         <private>
            <employee>
               intra-schedule:.....
               deployment:.....
            </employee>
      </Description>
   </Property>
   <Current_State>true</Current_State>
   <Relevance>captain,employee1,employee2,employee3</Relevance>
   <Time>
      <Inform_time>2007-08-18T14:30:00</Inform_time>
      <Operation_initial_time>2007-08-18T15:00:00</Operation_initial_time>
      <Exception_initial_time>2007-08-18T15:05:00</Exception_initial_time>

   <Exception_handle_time_interval>05:00</Exception_handle_time_interval>
      <Destruction_time>2007-08-18T15:15:00</Destruction_time>
   </Time>
</Context_Model>
```

Figure 4-9 Sample of the Context model

```
<?xml version="1.0" encoding="UTF-8"?>
<User_Model xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <Group>navigation</Group>
   <Name>employee1</Name>
   <ID>ID_026</ID>
   <Title>salor</Title>
   <PhoneExt>202</PhoneExt>
   <IP_Address>192.168.1.25</IP_Address>
   <Email>employee1@domin</Email>
   <Physical_Environment>
      <Location>bridge</Location>
   </Physical_Environment>
   <Hard_environment>
      <Device_capabilities>Zaurus</Device_capabilities>
      <bandwidth>10M</bandwidth>
      <processor_speed>250Hz</processor_speed>
      <storage_capacity>2GB</storage_capacity>
      <resolution>640*480</resolution>
   </Hard_environment>
   <Software_environment>
      <Operating_System>Linux</Operating_System>
   </Software_environment>
</User_Model>
```

Figure 4-10 Sample of the User model

In order to raise the compatibility, I used java language to establish the time trigger engine and parsed XML data by Document Object Model (DOM) API. By combining the models and the trigger engine, the system was implemented under a general wireless internet environment.

With the models, the behaviors of prototype system can be described as follows:

- Passive information providing level:

  Everyone (include passengers and employees) in this ship can check the public information about the speech (such as time, location, speech subject, etc.) from the Web server by using the terminals deployed within this passenger ship or their own PDA that has equipped with this Web browser. For the consideration of security, passengers can only access the public information, but employees with proper authentication can not only access public information, but also can receive more detail information about the speech (such as; people deployment, necessary

equipment, intra-schedule, etc.).

- Active information providing level:

  At PM 02:30, everyone should automatically receive an instant message (public information) from the terminal deployed in the passenger room or the employee PDA. Furthermore, the three employees that are assigned to prepare the speech would receive more extra detail information. The image diagram is shown in Figure 4-11.

- Exception handling time triggered event:

  At PM 03:00, if three of employees assigned to prepare the speech have not all sign in the service; then system will send an instant warning message to the PDA of missing employees to remind them. Along with time passing, if this abnormal situation is still existing, system will continually send warning message. Finally, at PM 03:15, if there is still a problem exists, then system will close this sign in service and inform system administrator to deal with this exception. The flowchart of exception handling is shown in Figure 4-12.

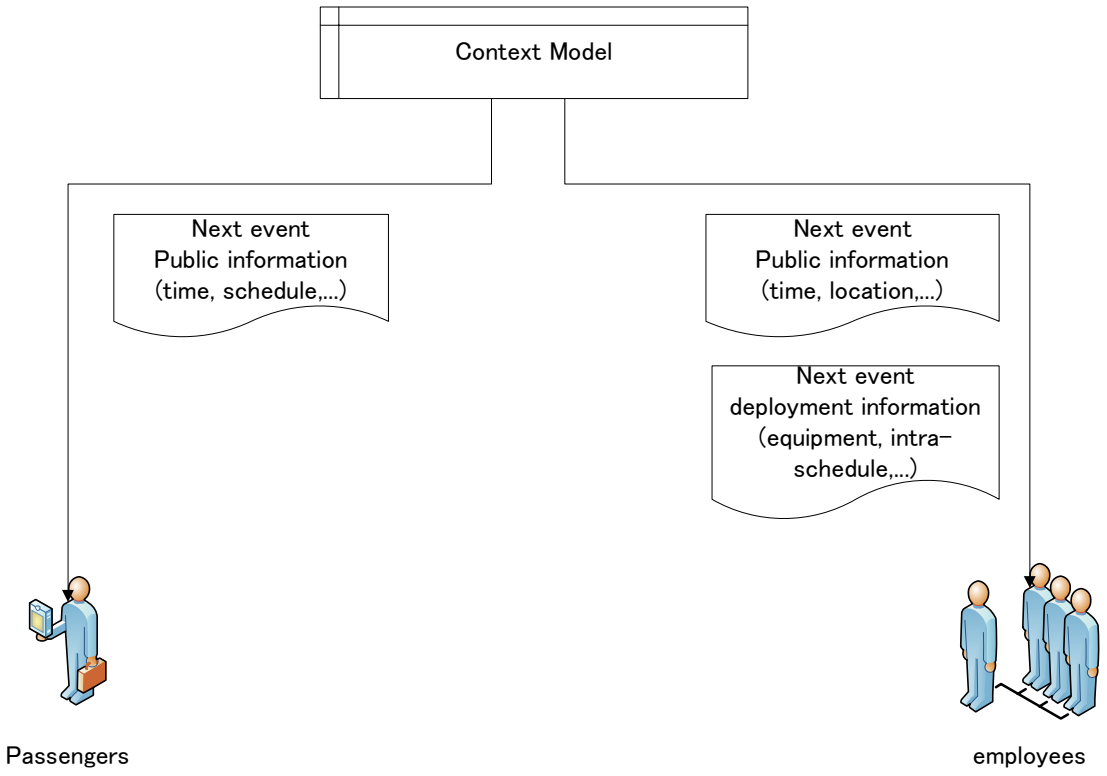


Figure 4-11 Example scenario information providing

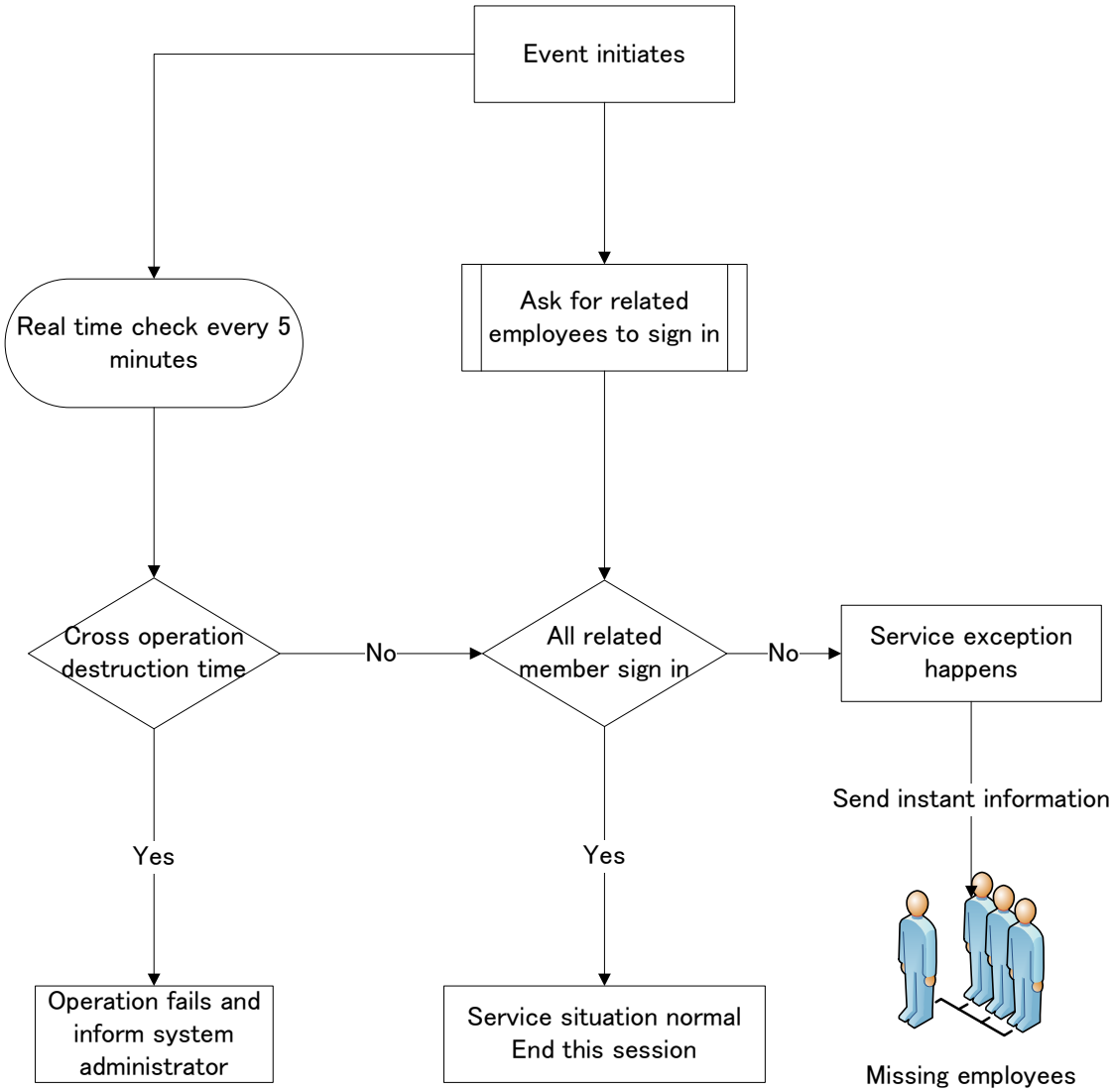


Figure 4-12 Example scenario exception handle flowchart

Chapter 5 Efficient Data Structure

In Chapter 3, an XML ship data model is proposed for fleet management. This data model can be operated independently within the ship, and when the ship arrives at port, it can be easily integrated with ship company data. On the other hand, compared with relational database, the efficiency of processing query for XML document is a serious issue. There are various researches surveyed about efficient XML query processing; they can be classified into the following three categories: 1) Path indexing [109], creates a path summary from XML data. Path indexing speeds up the evaluation of single path queries. 2) Node indexing [110], indexes each data node by some numbering schema. 3) Sequence-based indexing [111], transforms both XML documents and queries into sequence, and evaluates queries based on sequence matching. Although, there are so many efficient querying methods, the basic XML document deployment is not discussed. If the duplication of data set is not properly deployed, its performing efficiency could be reduced to a lower state. In order to overcome this disadvantage, a proper redesigned data structure should be considered.

In the maritime field, the industry is involved with several different government levels (such as; procedures of arriving in, cargo clearance, immigration declaration, etc.), and each department has its own specific requirement for desired information, so finally the needed total data storage capacity could be very huge. For example, the same or similar content of containers carried by a container ship could be deployed in different blocks within the ship or the container yard. In this case, most of cargo data set could be duplicated except for deploying locations. This kind of data set with high duplication could seriously affect its query efficiency if the data structure is not properly restructured.

For the application of the restructuring data structure, the XML data model is applied on containers. Because of the high duplication of this data set, an original XML document could be restructured to the more proper indexed XML document, and then XML Path Language [91] is employed to verify that the querying efficiency is improved.

5.1 Description of Container Operation

On the 26th of April 1956, a crane lifted fifty-eight aluminum truck bodies aboard an aging tanker ship moored in Newark, New Jersey. Five days later, the Ideal-X sailed into Houston, where fifty-eight trucks waited to take on the metal boxes and haul them to their destinations. Such was the beginning of a revolution [112].

Before the container steps in, transporting goods was so expensive that it did not pay to ship many things halfway across the country, much less halfway around the world.

What is it about the container that is so important? The value of this utilitarian object lies not in what the container is, but in how it is used. The container is at the core of a highly automated system for moving goods from anywhere to anywhere, with a minimum of cost and complication on the way.

Containers represent one of the most commonly used and diverse forms of units for transportation. Compared to tanks or surface impoundments, containers are less expensive and generally less difficult to manage. Containers are also mobile, allowing an owner or operator to use only one unit for storage, transportation, and disposal.

Container terminal operators support a very intense communication with external parties like shipping lines, agents, forwarders, truck and rail companies, governmental authorities like customs, waterway police and others. Every change of container status is communicated among the respective parties. From the point of view of operating terminal, the most important messages are: the container loading and discharging lists which specify every container to be loaded or unloaded to/from a ship with specific data; the ‘bayplan’ which contains all containers of a ship with their precise data and position within the ship (it is communicated before arrival in the port); the ‘stowage instruction’ which describes the positions where export containers have to be located in a ship and which is the base for the stowage plan of the terminal; container pre-advices for delivery by train and truck, and the schedule and loading instruction for trains.

Container operation is the Unit Load System which uses containers as units for unifying cargo transportation. It became an important subject for port transportation Since Container Operation was introduced in 1957. The merits can be given as below:

- Cargo loading rationalization by exclusive equipments: the cargo loading efficiency can be greatly improved by using exclusive equipments. This can also reduce the number of port operators and ship berthing time.
- Simply cargo freight: container based fee simply cargo freight system so then it is easy to calculate the total transportation freight.
- Enable of raining transportation: loading operation can be applied without considering weather condition, it allows accurate loading plan to be applied.
- Keep away from cargo damage and robbery: cargos can be transported directly from owners to ships, which can avoid unexpected situations to happen.
- Integration with overland transportation: container based transportation can integrate overland and sea transportation system.
- Usage of FCL and LCL cargo: Full Container Load Cargo (FLC) and Less than Container Load Cargo (LCL) can be applied at the same time, which gives more flexibility for port transportation.
- Keep cargo in good condition: cargo condition can be well protected by choosing proper type of container to maintain its temperature, moisture or ventilation.

In this chapter, I focused on the implementation of XML data model in the Container Yard. Take the container import as example, the container operations have to be composed of several procedures, such as; getting the Manifest from ship company, establishing the container cargo list, submitting the Unloading container list and boating note to the Custom, establishing the container unloading plan, ensuring the Import Declaration, Overland Transport and Delivery Order, taking out cargo from CFS container and rearranging them, registering the Duty Exempted Container, establishing the operation schedule of Inbound Botanical and Zoological cargo, etc. [113, 114, 115].

There are many kinds of official documents have to be submitted while the container import procedure, such as; the Cargo delivery order (Figure 5-1), Bill of lading (Figure 5-2), Dangerous cargo list (Figure. 5-3), Cargo boat note (Figure 5-4), Equipment receipt (Figure 5-5), Container load plan (Figure. 5-6) and Container move-in form (Figure 5-7).

I focused on the container related part of application documents listed above and draw out the desired information for the implementation of container node within the entire XML data model.

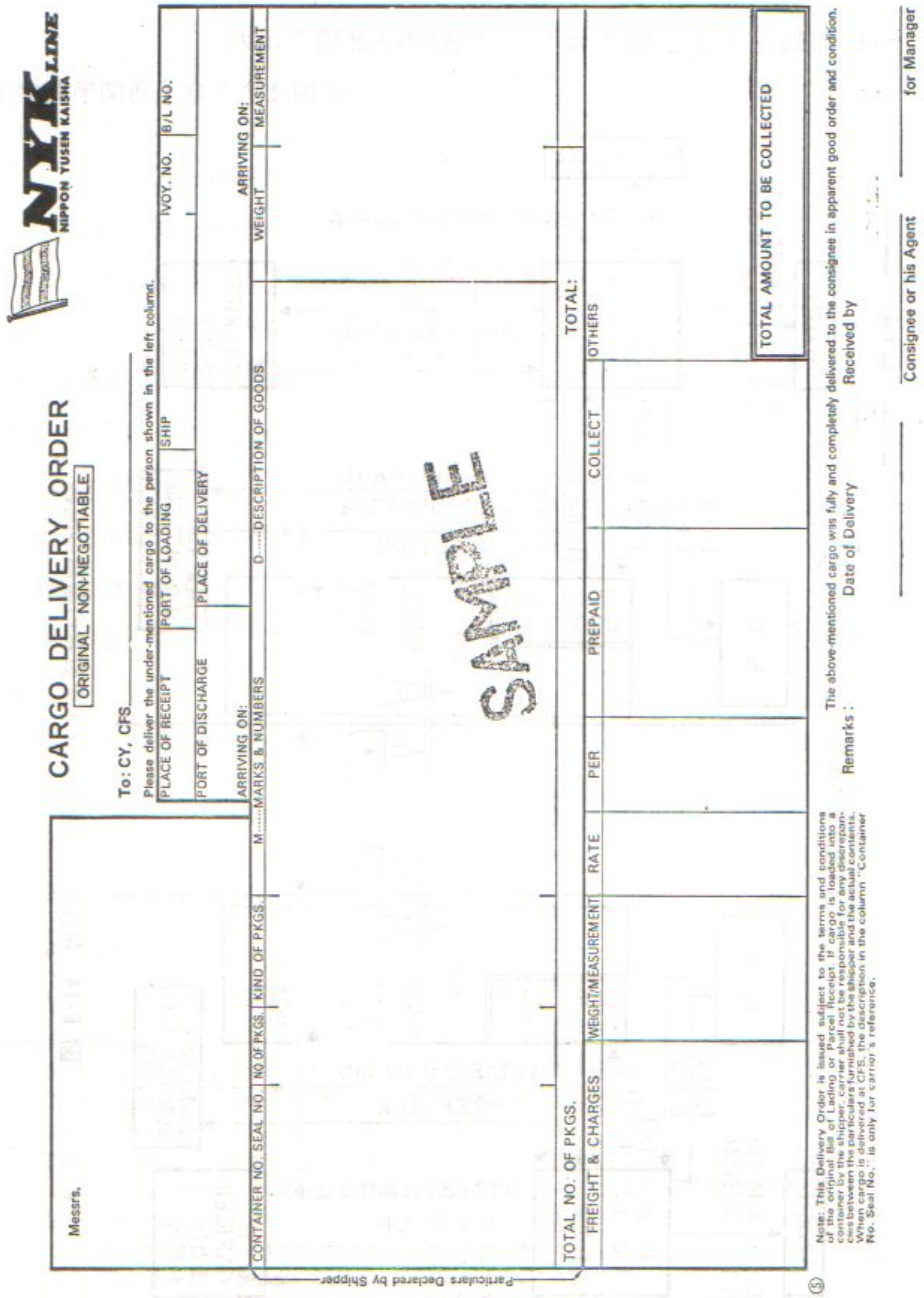


Figure 5-1 Sample of Cargo delivery order [113]

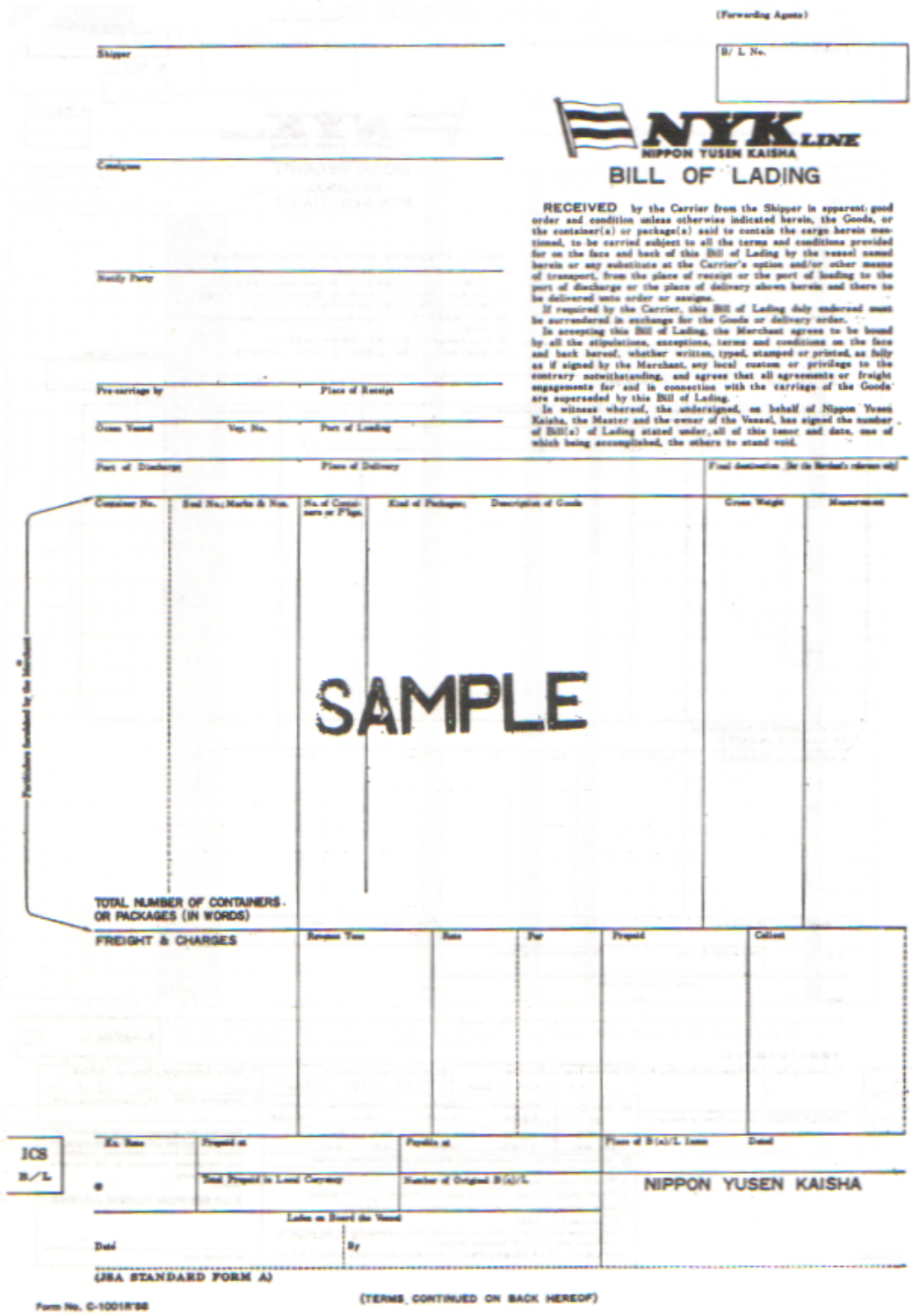


Figure 5-2 Sample of Bill of lading [113]

DANGEROUS CARGO LIST

PAGE: ____

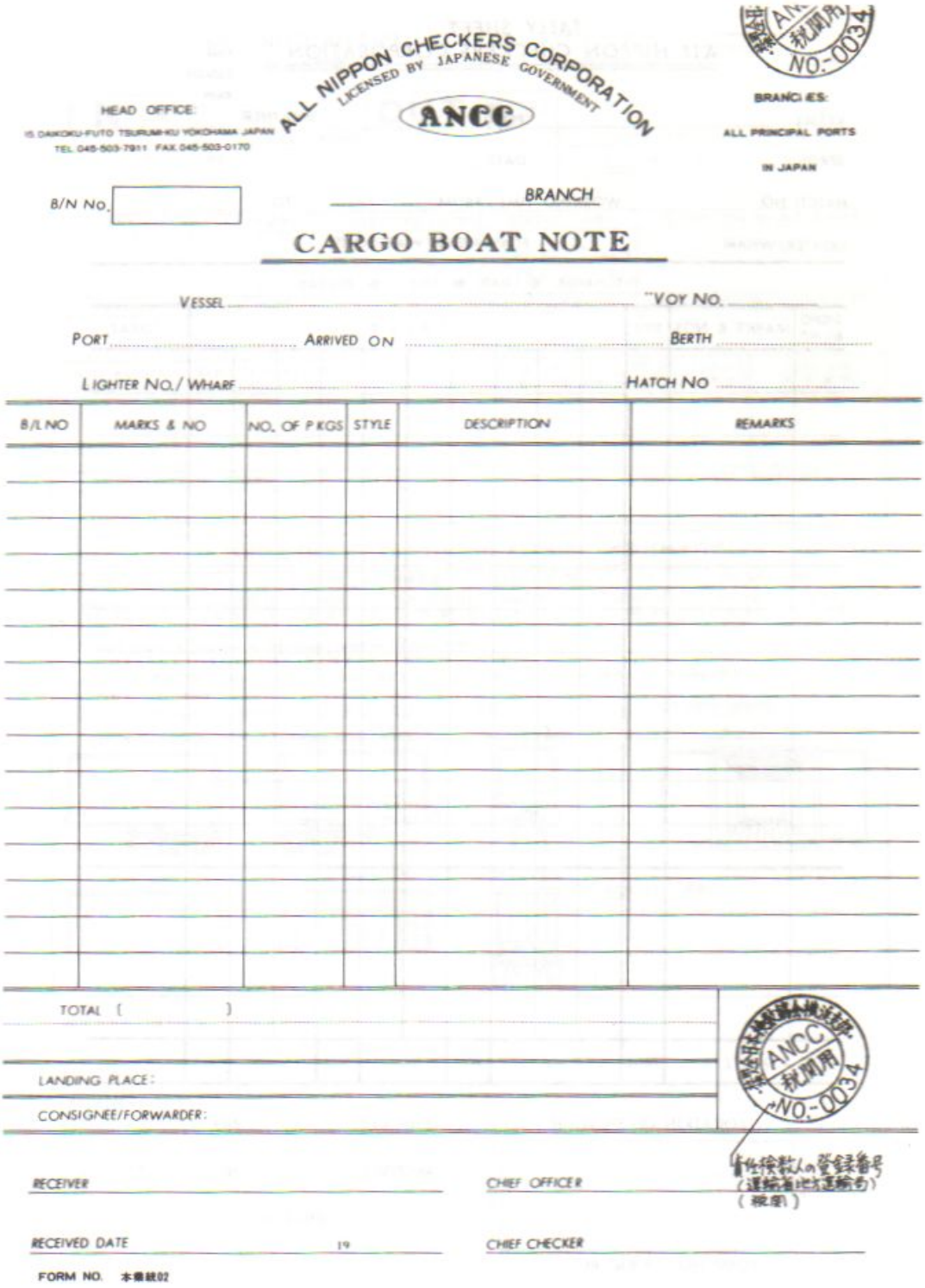
SHIP'S NAME: _____ NATIONALITY: _____ REGISTERED NO: _____ KIND OF SHIP: _____

VOYAGE NO: _____ NAME OF MASTER: _____

CONTAINER NO. / B/L NO.	PORT OF LOADING / PORT OF DISCHARGE	DATE D/M/Y / DATE D/M/Y	SERVICE CY / CFS	QUANTITY NO. & KIND OF PACKAGES / WEIGHT (GROSS/NET)	LOCATION OF GOODS	DESCRIPTION OF GOODS ARTICLE	MARKS AND NOS ITEM	CLASSIFICATION LABEL	SHIPPER'S ADDRESS NAME & / CONSIGNEE'S ADDRESS NAME &	REMARKS

NOTE : Flash point for dangerous cargo must be shown in remarks column

Figure 5-3 Sample of Dangerous cargo list [113]



ALL NIPPON CHECKERS CORPORATION
LICENSED BY JAPANESE GOVERNMENT
ANCC

HEAD OFFICE:
15, DAIKOKU-FUTO TSURUMI-KU YOKOHAMA JAPAN
TEL 045-503-7911 FAX 045-503-0170

BRANCHES:
ALL PRINCIPAL PORTS
IN JAPAN

B/N No. _____ BRANCH _____

CARGO BOAT NOTE

VESSEL _____ VOY NO. _____

PORT _____ ARRIVED ON _____ BERTH _____

LIGHTER NO./ WHARF _____ HATCH NO _____

B/L NO	MARKS & NO	NO. OF PKGS	STYLE	DESCRIPTION	REMARKS

TOTAL ( )

LANDING PLACE:

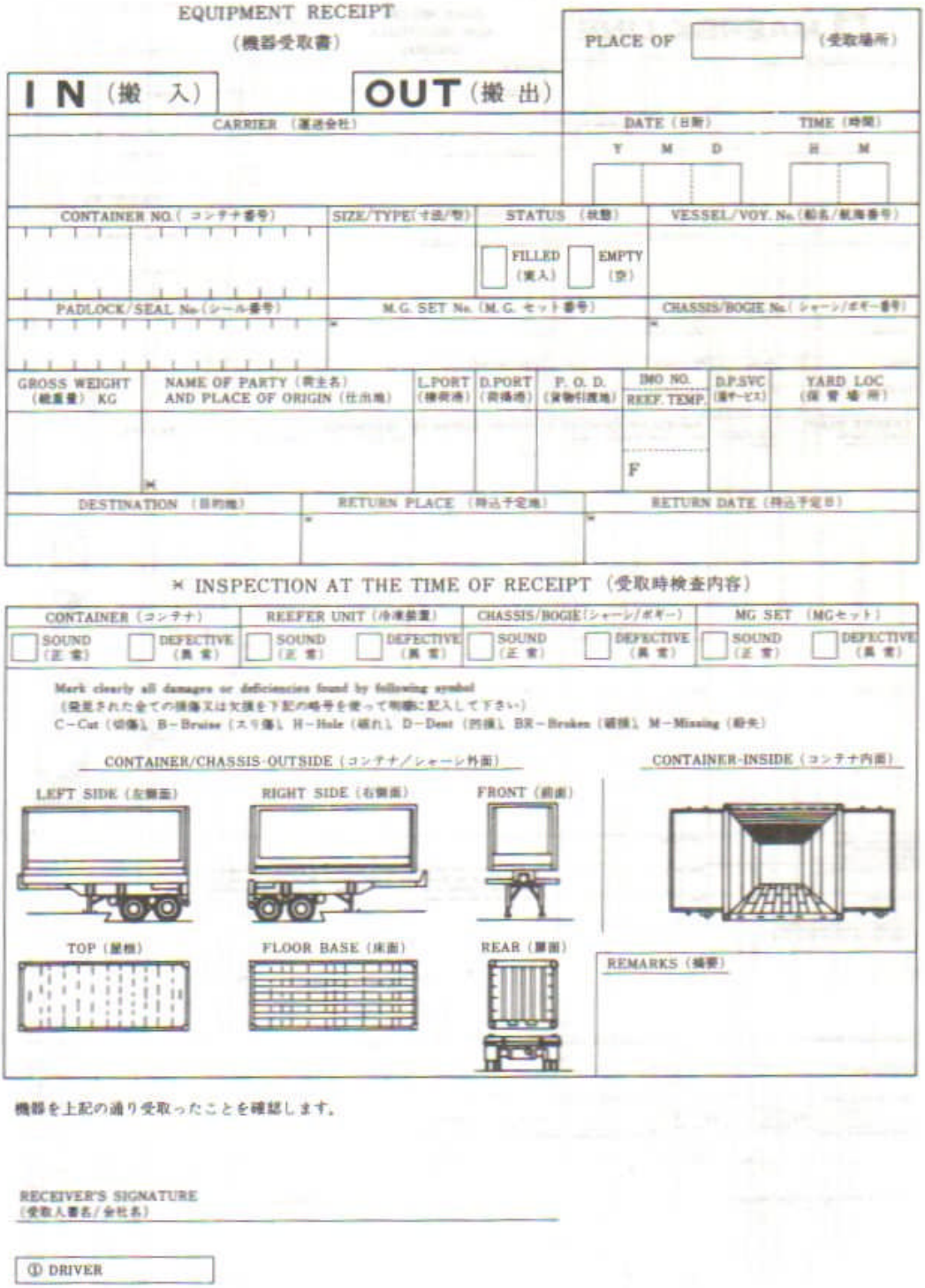
CONSIGNEE/FORWARDER:

RECEIVER _____ CHIEF OFFICER _____

RECEIVED DATE _____ 19 _____ CHIEF CHECKER _____

FORM NO. 本機統02

Figure 5-4 Sample of Cargo boat note [114]



EQUIPMENT RECEIPT
(機器受取書)

PLACE OF (受取場所)

I N (搬 入) OUT (搬 出)

CARRIER (運送会社) | DATE (日附) Y M D | TIME (時間) H M

CONTAINER NO. (コンテナ番号) | SIZE/TYPE (寸法/型) | STATUS (状態) FILLED (実入) EMPTY (空) | VESSEL/VOY. No. (船名/航海番号)

PADLOCK/SEAL No. (シール番号) | M.G. SET No. (M.G. セット番号) | CHASSIS/BOGIE No. (シャーシ/ボギー番号)

GROSS WEIGHT (総重量) KG | NAME OF PARTY (荷主名) AND PLACE OF ORIGIN (仕出地) | L.PORT (揚荷港) | D.PORT (荷積港) | P. O. D. (貨物引渡地) | IMO NO. / REEF. TEMP. F | D.P.SVC (港サービス) | YARD LOC (保管場所)

DESTINATION (目的地) | RETURN PLACE (持込予定地) | RETURN DATE (持込予定日)

※ INSPECTION AT THE TIME OF RECEIPT (受取時検査内容)

CONTAINER (コンテナ)	REEFER UNIT (冷凍装置)	CHASSIS/BOGIE (シャーシ/ボギー)	MG SET (MGセット)
SOUND (正常) DEFECTIVE (異常)	SOUND (正常) DEFECTIVE (異常)	SOUND (正常) DEFECTIVE (異常)	SOUND (正常) DEFECTIVE (異常)

Mark clearly all damages or deficiencies found by following symbol
(発見された全ての損傷又は欠損を下記の略号を使って明瞭に記入して下さい)
C－Cut (切傷), B－Bruise (スリ傷), H－Hole (穴れ), D－Dent (凹損), BR－Broken (破損), M－Missing (紛失)

CONTAINER/CHASSIS OUTSIDE (コンテナ/シャーシ外面)

LEFT SIDE (左側面) RIGHT SIDE (右側面) FRONT (前面)

TOP (屋根) FLOOR BASE (床面) REAR (扉面)

CONTAINER-INSIDE (コンテナ内面)

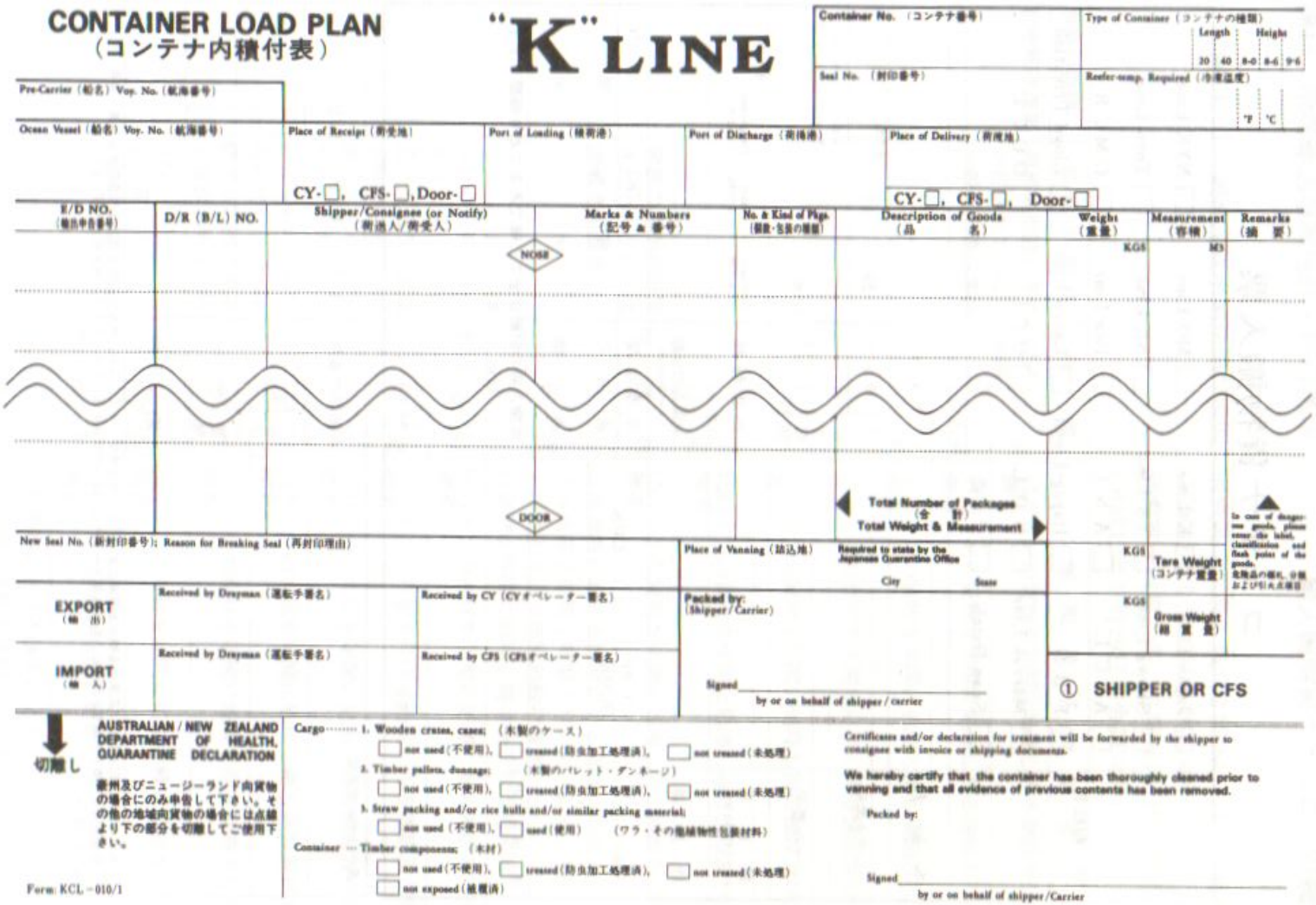
REMARKS (摘要)

機器を上記の通り受取ったことを確認します。

RECEIVER'S SIGNATURE
(受取人署名/会社名)

① DRIVER

Figure 5-5 Sample of Equipment receipt [114]



CONTAINER LOAD PLAN
（コンテナ内積付表）

"K" LINE

Container No. （コンテナ番号）
Type of Container （コンテナの種類）
Length: 20 40 — Height: 8-0 8-6 9-6
Seal No. （封印番号）
Reefer-temp. Required （冷凍温度） °F °C
Pre-Carrier （船名） Voy. No. （航海番号）
Ocean Vessel （船名） Voy. No. （航海番号）
Place of Receipt （荷受地）
CY-☐, CFS-☐, Door-☐
Port of Loading （積荷港）
Port of Discharge （荷揚港）
Place of Delivery （荷渡地）
CY-☐, CFS-☐, Door-☐

E/D NO. （輸出申告番号）	D/R (B/L) NO.	Shipper/Consignee (or Notify) （荷送人/荷受人）	Marks & Numbers （記号 & 番号）	No. & Kind of Pkgs. （個数・包装の種類）	Description of Goods （品名）	Weight （重量）	Measurement （容積）	Remarks （摘要）
			NOSE			KGS	M3	
			DOOR		Total Number of Packages （合計） Total Weight & Measurement			

In case of dangerous goods, please enter the label, classification and flash point of the goods.
危険品の場合、分類および引火点項目

New Seal No. （新封印番号）; Reason for Breaking Seal （再封印理由）
Place of Vanning （詰込地）
Required to state by the Japanese Quarantine Office
City State
KGS Tare Weight （コンテナ重量）
KGS Gross Weight （総重量）

EXPORT （輸出）
Received by Drayman （運転手署名）
Received by CY （CYオペレーター署名）

IMPORT （輸入）
Received by Drayman （運転手署名）
Received by CFS （CFSオペレーター署名）

Packed by: (Shipper/Carrier)
Signed
by or on behalf of shipper / carrier

① SHIPPER OR CFS

切離し

AUSTRALIAN / NEW ZEALAND DEPARTMENT OF HEALTH, QUARANTINE DECLARATION
豪州及びニュージーランド向貨物の場合にのみ申告して下さい。その他の地域向貨物の場合には点線より下の部分を切離してご使用下さい。

Form: KCL－010/1

Cargo…… 1. Wooden crates, cases; （木製のケース）
☐ not used (不使用), ☐ treated (防虫加工処理済), ☐ not treated (未処理)
2. Timber pallets, dunnage; （木製のパレット・ダンネージ）
☐ not used (不使用), ☐ treated (防虫加工処理済), ☐ not treated (未処理)
3. Straw packing and/or rice hulls and/or similar packing material; （ワラ・その他植物性包装材料）
☐ not used (不使用), ☐ used (使用)
Container … Timber components; （木材）
☐ not used (不使用), ☐ treated (防虫加工処理済), ☐ not treated (未処理)
☐ not exposed (被覆済)

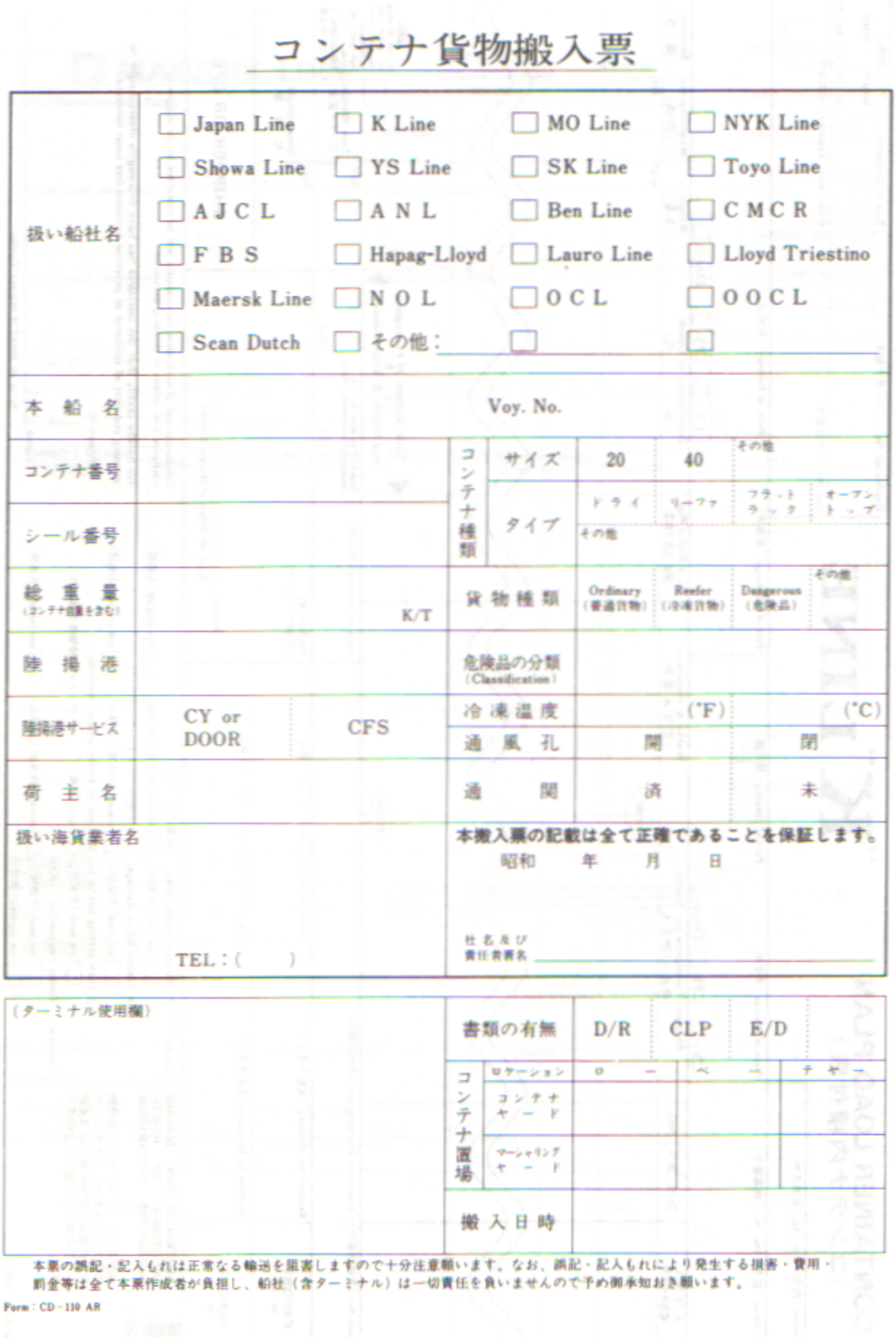
Certificates and/or declaration for treatment will be forwarded by the shipper to consignee with invoice or shipping documents.

We hereby certify that the container has been thoroughly cleaned prior to vanning and that all evidence of previous contents has been removed.

Packed by:

Signed
by or on behalf of shipper/Carrier

Figure 5-6 Sample of Container load plan [115]



コンテナ貨物搬入票

扱い船社名
☐ Japan Line ☐ K Line ☐ MO Line ☐ NYK Line
☐ Showa Line ☐ YS Line ☐ SK Line ☐ Toyo Line
☐ A J C L ☐ A N L ☐ Ben Line ☐ C M C R
☐ F B S ☐ Hapag-Lloyd ☐ Lauro Line ☐ Lloyd Triestino
☐ Maersk Line ☐ N O L ☐ O C L ☐ O O C L
☐ Scan Dutch ☐ その他： ☐ ☐

本船名 Voy. No.

コンテナ番号
シール番号
総重量（コンテナ自重を含む） K/T
陸揚港
陸揚港サービス CY or DOOR CFS
荷主名

コンテナ種類 サイズ 20 40 その他
タイプ ドライ リーファ フラットラック オープントップ その他
貨物種類 Ordinary（普通貨物） Reefer（冷凍貨物） Dangerous（危険品） その他
危険品の分類（Classification）
冷凍温度 (°F) (°C)
通風孔 開 閉
通関 済 未

扱い海貨業者名
TEL：( )

本搬入票の記載は全て正確であることを保証します。
昭和 年 月 日
社名及び責任者署名

（ターミナル使用欄）

書類の有無 D/R CLP E/D
コンテナ置場 ロケーション ロー ベー テヤー
コンテナヤード
マーシャリングヤード
搬入日時

本票の誤記・記入もれは正常なる輸送を阻害しますので十分注意願います。なお、誤記・記入もれにより発生する損害・費用・罰金等は全て本票作成者が負担し、船社（含ターミナル）は一切責任を負いませんので予め御承知おき願います。

Form：CD－110 AR

Figure 5-7 Sample of Container move-in form [115]

5.2 XML Data Model for Container Operation

In this section, I focused on the port container operation and established cargo XML data model for container management by treating a single container as an XML data node.

There are many requests for managing containers from multi ship companies in the Container Yard, loading and unloading operations have to be handled simultaneously while ships arrive in the port. Containers are divided into several types by the cargo nature; different types of cargo also have different requests (such as; specific equipments for unloading, quarantine operations for botanical and zoological cargo, the reefer containers have to be kept in a low temperature, etc.). However, the operations are very complicated, the schedules and procedures have to be planned and managed in advance. Normally, a Container Yard accepts the orders from several fixed ship companies to handle the related import and export flow. Under this kind of situation, the original container data structure of each single container could be seen as under respective ship company, also each single container owns its independent details (attributes).

If we display the container data with XML tree structure, then the containers could be treated as the elements belongs to the lowest layer, each container node has several leaf elements to store different attributes. Each container belongs to a specific ship company; finally each ship company belongs to the container yard node (the root element of XML data model). This container XML data model is called as original container XML data model and its structure is illustrated in Figure. 5-8. Detail of container XML data model will be given in next section.

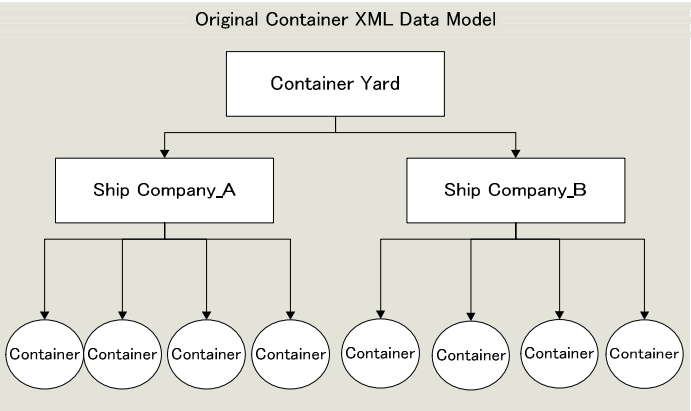


Figure 5-8 Original container XML data model

5.2.1 Container XML Data Model Definition

In order to apply the container yard management, the data node for single container should be composed of multiple-information. Based upon ongoing Customs' clearance procedures, the desired information from related application forms (as shown form Figure 5-1 to Figure 5-7) are collected as the attributes for containers shown in Table 5-1. This data model is built for verifying the efficiency of querying experiments, but not proposing the actual container operation implementation. There are still many attributes not included in this container model, but this research is focus on data structure, so just some important attributes are imported from the entire cargo transportation system for the implementation.

Table 5-1 Desired container attributes

Attribute	Description
Serial number	Identification ruled by ISO standard for each container.
Cargo context	Classification of Full Container Load Cargo (FLC) and Less than Container Load Cargo (LCL).
Ship company	The ship company having responsibility for cargo transportation.
Gross weight	Cargo total weight (including of container tare weight).
Port of departure	The port that containers were loaded.
Port of destination	The port that container will be unloaded.
Dangerous cargo	Cargo which, because of its dangerous properties, is subject to special regulations for its transport.
Place of origin	The place of cargo production.
Name of party	The name of cargo owner.
Kind of package	The classification of cargo.
Description	Description of cargo context.
Container type	Containers can be classified as several types by their purposes (dry, flat, opentop, bulk, reefer, tank, ventilated and pen).

According to proposed definition for container node, the XML Schema for original container XML data model can be illustrated in Figure 5-9. Each node under container node is a leaf node which can carries text or numeric information. Figure 5-10 shows some simple entity of the complete original XML data model.

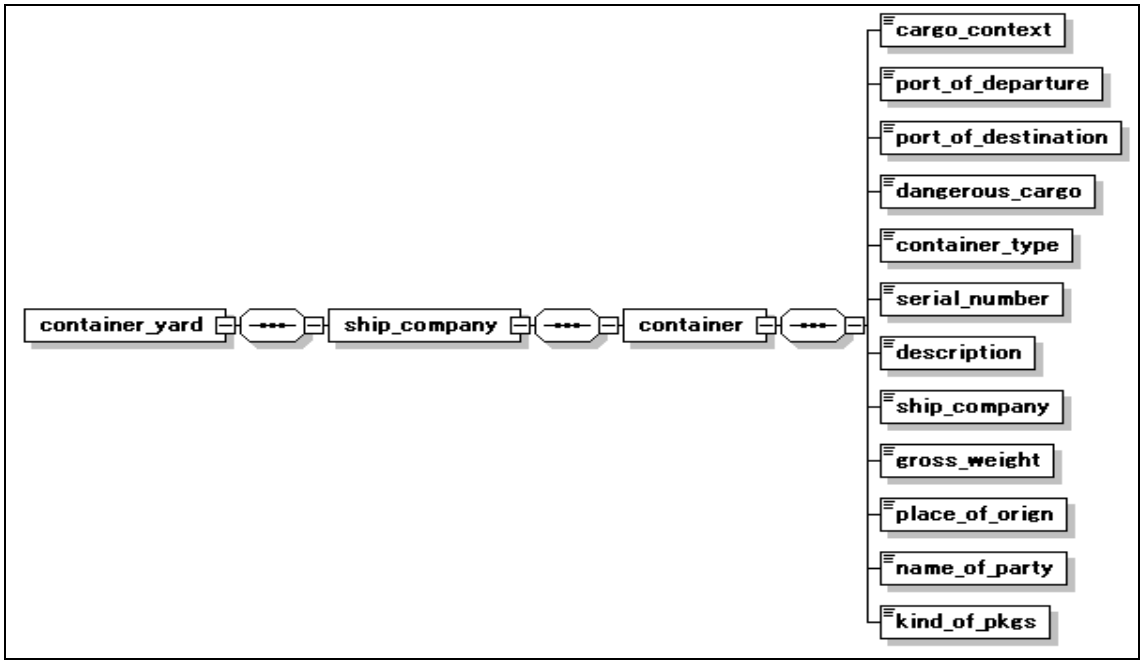


Figure 5-9 Original container XML data model Schema

```
<?xml version='1.0'?>
<container_yard>
   <ship_company>
      <container>
         <cargo_context>LCL</cargo_context>
         <dangerous_cargo>no</dangerous_cargo>
         <port_of_departure>OSAKA</port_of_departure>
         <port_of_destination>KOBE</port_of_destination>
         <container_type>bulk</container_type>
         <serial_number>JCAU 1234567</serial_number>
         <description>INSTANT FOOD</description>
         <ship_company>ship_company1</ship_company>
         <gross_weight>100</gross_weight>
         <place_of_origin>KYOTO</place_of_origin>
         <name_of_party>OWNER1</name_of_party>
         <kind_of_pkgs>FOOD</kind_of_pkgs>
      </container>
                        ......
   </ship_company>
</container_yard>
```

Figure 5-10 Sample of original container XML document

5.2.2 Restructuring Container XML Data Model

The XML standard defines that an XML document has to be a nested structure and only the leaf elements (the lowest layer elements) can store strings or numeric information. XML documents have more redundant context and need more storage space than relational database because of this essential limitation. This fact directly affects the efficiency of querying.

For one container, it has various attributes and each of those attributes has the one-to-one relationship with the container. But, if we see the entire data structure, there are still some attributes with high duplication which can be grouped. Considering the “dangerous cargo” attribute of the container data model as an example, its value can be just “yes” or “no”. On the other hand, once the “dangerous cargo” attribute is defined within the XML schema, the instance of it has to occur repeatedly within each single container node of XML document. So, if we can group the attributes with highly duplicated within XML documents, then we can reduce the data storage space effectively. The image figure of grouping attributes is shown in Figure 5-11.

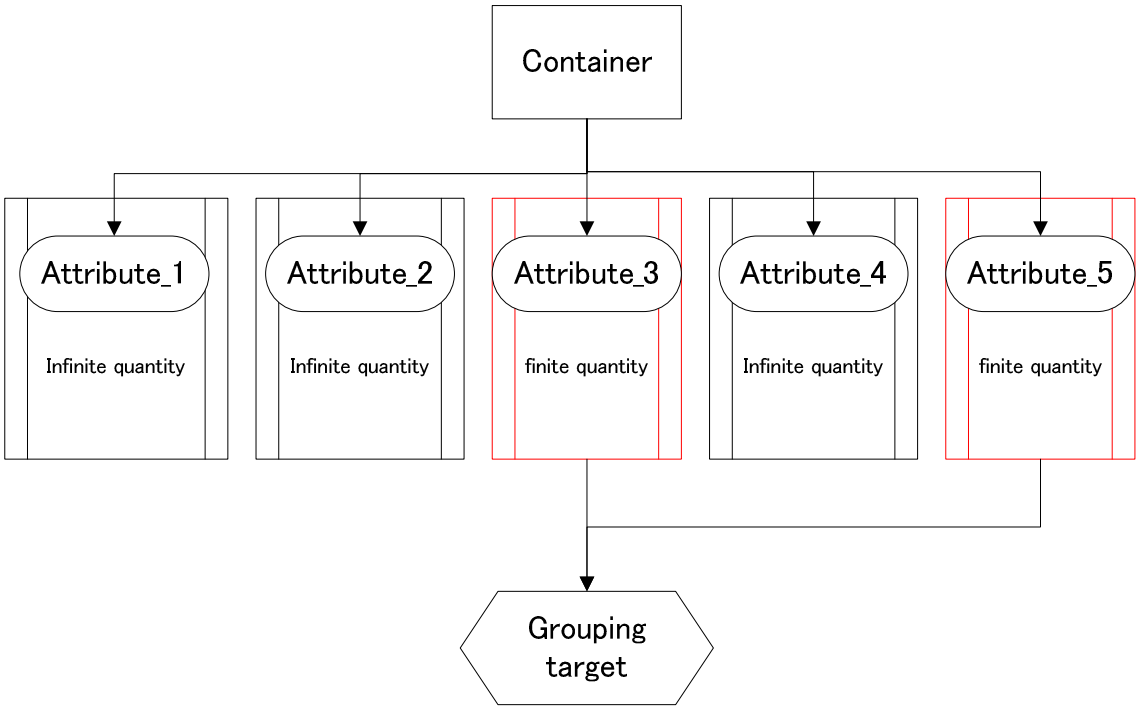


Figure 5-11 Attributes grouping target

Although the elements of XML document (except leaf elements) cannot store information, it is available to self-define multi attributes for each element. By using this function, we can translate highly duplicated attributes with finite quantity of leaf elements into the self-defined attribute values of upper layer elements. A translation sample is given in Figure 5-12.

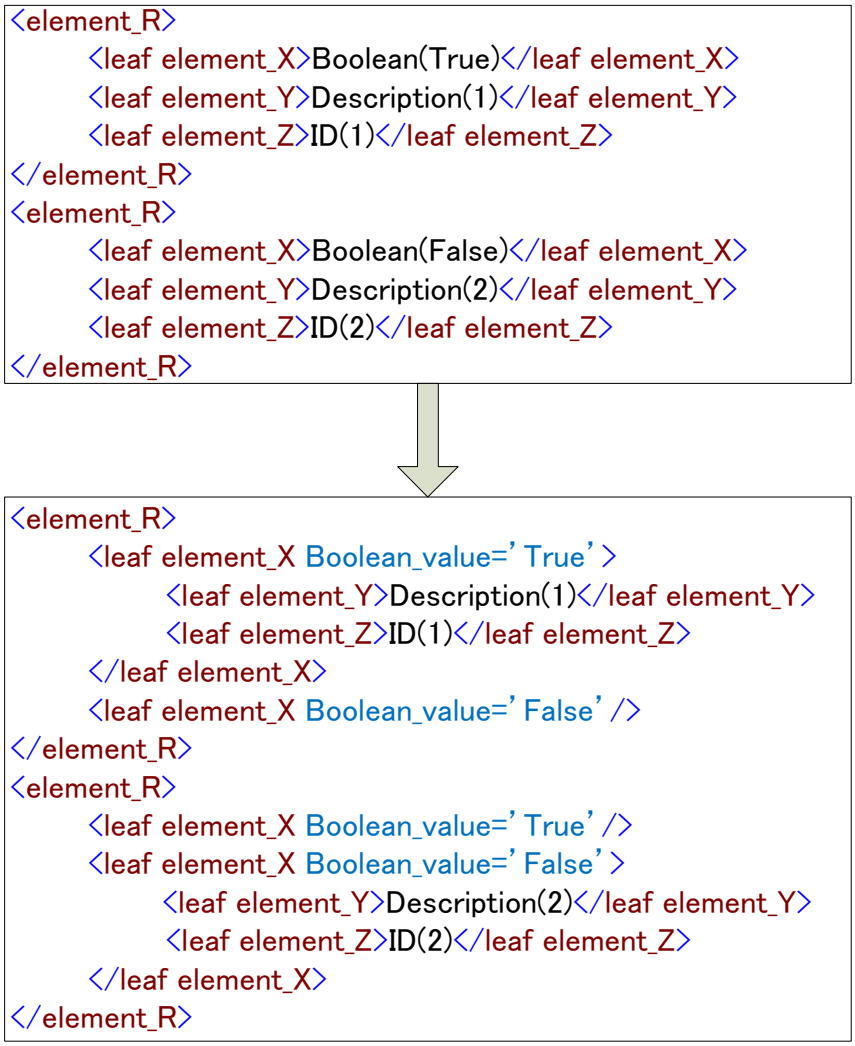


Figure 5-12 Sample of XML document restructuration

For the original container XML data model, the attributes with high duplication can be collected as index candidates and restructured the original XML document into an indexed XML document. The detail of index candidates is given in Table 5-2.

Table 5-2 Index attribute candidates

Attributes	Possible values	Quantity
Cargo context	FCL, LCL	2
Dangerous cargo	Yes, no	2
Port of departure	PL1, PL2, PL3,PL4, PL5	5
Port of destination	PD1, PD2, PD3,PD4, PD5, PD6	6
Container type	dry, flat, opentop, bulk, reefer, tank, ventilated, pen	8

In order to improve the querying efficiency, the indexed XML document is restructured by increasing order of attribute quantity as shown in Table 5-2.

A sample of the indexed container XML document is illustrated as Figure 5-13.

```
<?xml version='1.0'?>
<container_yard>
   <cargo_context value=' LCL' >
      <dangerous_cargo value=' no' >
         <port_of_departure value=' OSAKA' >
            <port_of_destination value=' KOBE' >
               <container_type value=' bluk' >
                  <container>
                     <serial_number>JCAU 1234567</serial_number>
                     <description>INSTANT FOOD</description>
                     <ship_company>ship_company1</ship_company>
                     <gross_weight>100</gross_weight>
                     <place_of_origin>KYOTO</place_of_origin>
                     <name_of_party>OWNER1</name_of_party>
                     <kind_of_pkgs>FOOD</kind_of_pkgs>
                  </container>
               </container_type>
            <port_of_destination>
         <port_of_departure>
      <dangerous_cargo>
   <cargo_context>
                              ......
</container_yard>
```

Figure 5-13 Sample of Indexed container XML document

Each container node within Original XML container document has 12 leaf elements. In the test case, 5 leaf elements are adopted from them as the index attributes to process the grouping procedure and each container within the Indexed container XML document only has 7 leaf elements. So every time when a new container is added, the data difference between the Original XML container document and the Index container XML document will increase 5. On the other hand, the subject varieties of the index group are fixed according to the number of index attributes at the first place. Therefore, the data capacity will totally decrease along with the increase of the number of containers.

In order to verify the effect of data storage space, several cases composed of different total container number from 1,000 to 500,000 are established for both original and indexed XML documents. For the first 1,000 container case, the file storage space reducing percentage is 83.4%. And, along with the growth of container number, finally the file storage space reducing percentage reaches to 74.0%. The relationship between total container number and file capacity is shown in Figure 5-14.

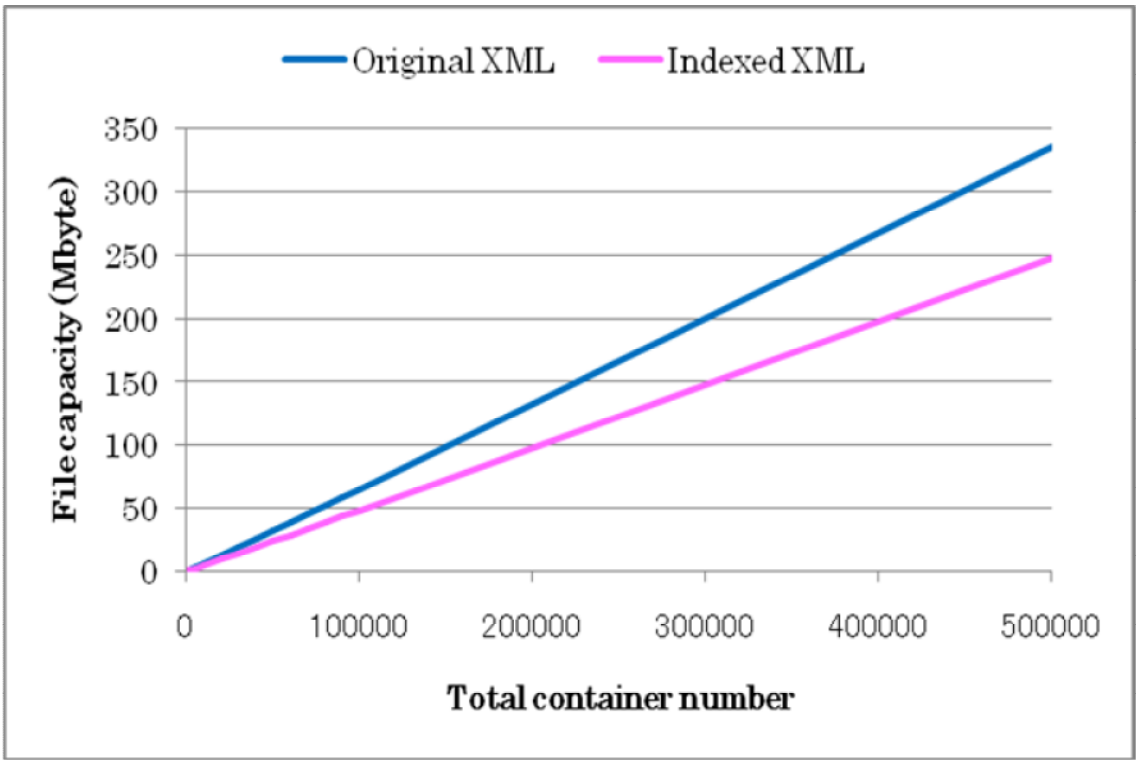


Figure 5-14 XML files capacity comparison

5.3 Performance Evaluation

In this section, XPath constructions are used to verify the efficiency for querying the original and restructured XML documents.

XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by the following path (or steps). However, the full syntax for XPath is somewhat cumbersome, so some syntactic simplifications have been introduced. For example, if no axis is specified, the child axis is used as default. This leads to a syntax very similar to directory paths of modern operating systems, so long as you restrict yourself to only the child axis. Certainly, the child axis is the most used of the axes.

This context node is provided by the parent language in which XPath is used, e.g. by XPointer. On the other hand, an absolute path always starts from the root node.

The common XPath constructions are given in Table 5-3.

Table 5-3 Common XPath constructions

Full construction	Simplified construction	Description
child::	Ignored	Child node of construction node
attribute::	@	Attribute node of construction node
/descendant-or-self:: node()/	//	Construction itself and its descendant nodes
self::node()	.	Construction node itself
parent::node()	..	Parent node of construction node
Child::text()	text()	All text belong to the child nodes of construction node

For example, with the root node as context node, the location path "News/Item[@date]" selects all news elements with a date attribute. An example of XPath query is shown in Fig 5-15. In this example, The XPath construction as “/News/Item[@date=’080521’]” is used to query the XML document. The query result returned would be a node set composed of node (Headline_c) and node (Headline_d), then we can use various node set functions to get the contexts of the result node set.

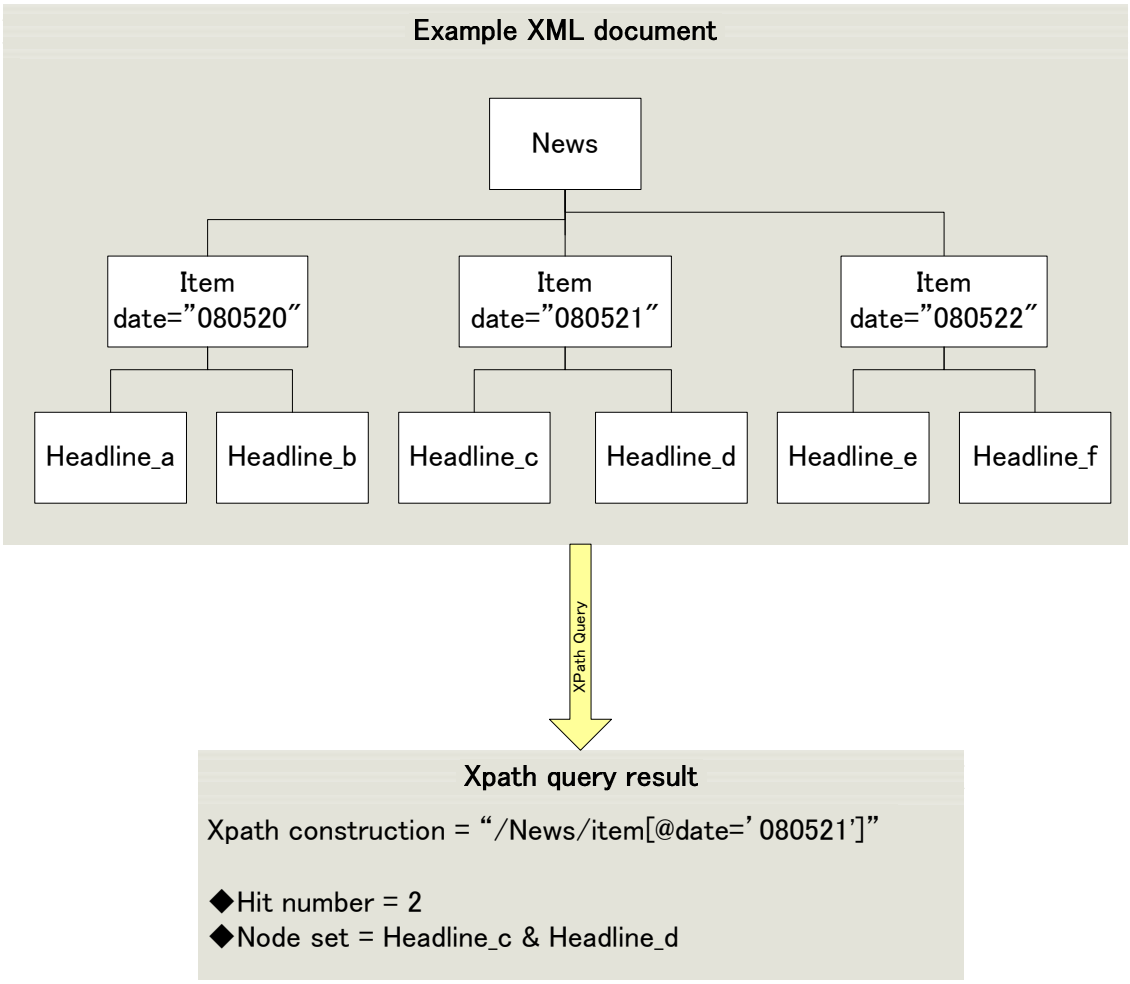


Figure 5-15 An example of XPath query

The query result of XPath returns a node set including node “Headline_c” and node “Headline_d”. Then, method count () can be used to get the total node number from the result node set, and method text () to output the content of each node.

In order to verify the efficiency of the XML data structure, visual basic.net 2008 express version is used to build an XML document generator and XPath query application.

The XML document generator can establish random original and restructured XML documents by designating total number of container. The important index attributes of each container are assigned within the default candidates. For example, the value of container type can only be one of dry, flat, opentop, bulk, reefer, tank, ventilated and pen. The other attributes are randomly given such as; strings, integers or their combination. The original and indexed container XML datasets with 100,000 containers could be built within one minute.

The XPath query function can process single path query (direct input XPath) and scheduled query procedures (10 constructions are designed as default). Each query procedure runs 20 times to get their average and ensure that querying results can be more accurate.

Performance experiments were run on a 2 GHz PC-compatible machine with RAM of 1GB running Windows XP.

Three control factors in the evaluation are used: total container number, the number of index attribute candidates, sequence and order of the index attribute candidates. The evaluation is separated into two steps.

- First step

  In order to verify the improvement of XPath querying efficiency, the used number of index attributes is fixed to 5 (all index attribute candidates in Table. 5-2); the order of index attributes is increasing sequence. Under these conditions, several test cases that total container number is from 1,000 to 100,000 are executed.

- Second step

  The effects of the number and order of index attributes are discussed in the second step. In this step, the number of index attributes is changed to 3, the first three index attribute candidates (Cargo context, Dangerous cargo and Port of departure); the order changes to decreasing sequence. Then I compare them with the result of first step.

10 query missions are designed for evaluation test with different purposes for the data sets as shown in Table 5-4.

Table 5-4 Sample queries for data sets

	Description
Q1	Original: //container
	Indexed: //container
Q2	Original: //container[serial_number='986']
	Indexed: //container[serial_number='986']
Q3	Original: sum(//container/gross_weight/text())
	Indexed: sum(//container/gross_weight/text())
Q4	Original: //container[cargo_context='LCL']
	Indexed: //cargo_context[@value='LCL']//container
Q5	Original: //container[port_of_departure='PL3']
	Indexed: //port_of_departure[@value='PL3']//container
Q6	Original: //container[port_of_destination='PD4']
	Indexed: //port_of_destination[@value='PD4']//container
Q7	Original: //container[container_type='reefer']
	Indexed: //container_type[@value='reefer']//container
Q8	Original: //container[dangerous_cargo='yes']
	Indexed: //dangerous_cargo[@value='yes']//container
Q9	Original: //container[container_type='ventilated' and cargo_context='FCL']
	Indexed: //cargo_context[@value='FCL']//container_type[@value='ventilated']//container
Q10	Original: //container[dangerous_cargo='yes' and port_of_destination='PD6' and port_of_departure='PL2']
	Indexed: //dangerous_cargo[@value='yes']/port_of_departure[@value='PL2']/port_of_destination[@value='PD6']//container

The performance of the querying procedures is shown in Figure 5-16 through Figure 5-24.

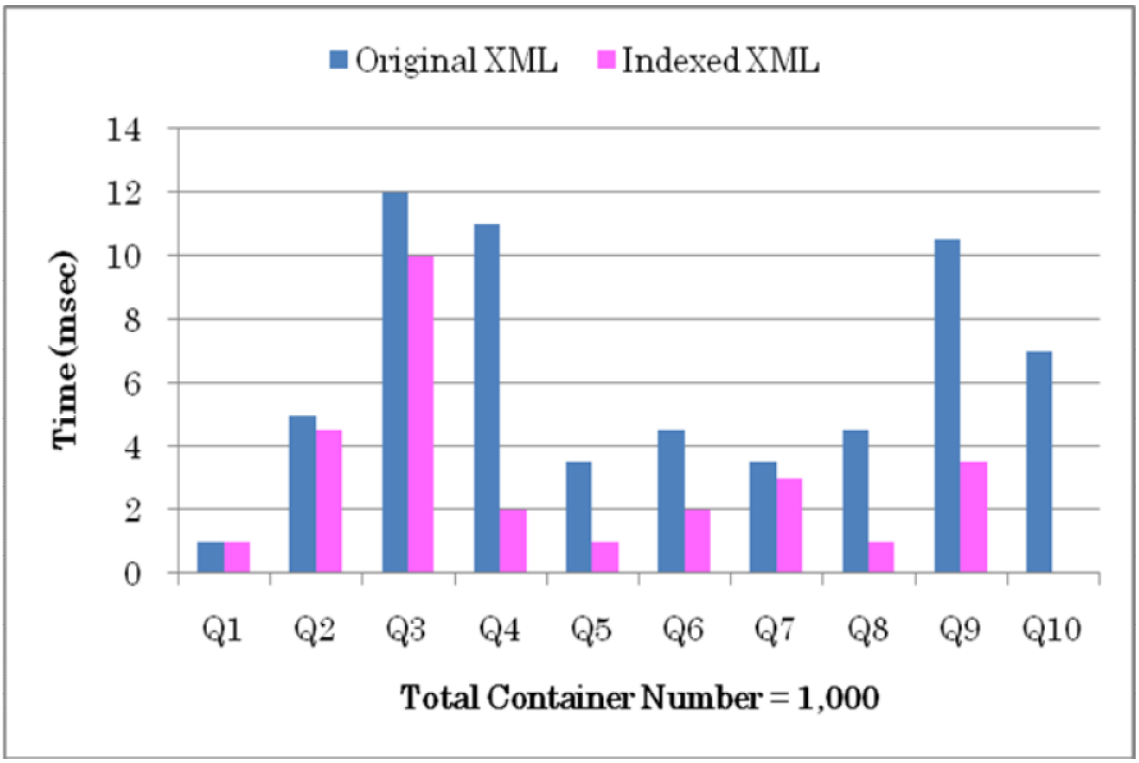


Figure 5-16 XPath querying time comparison while total container number=1,000

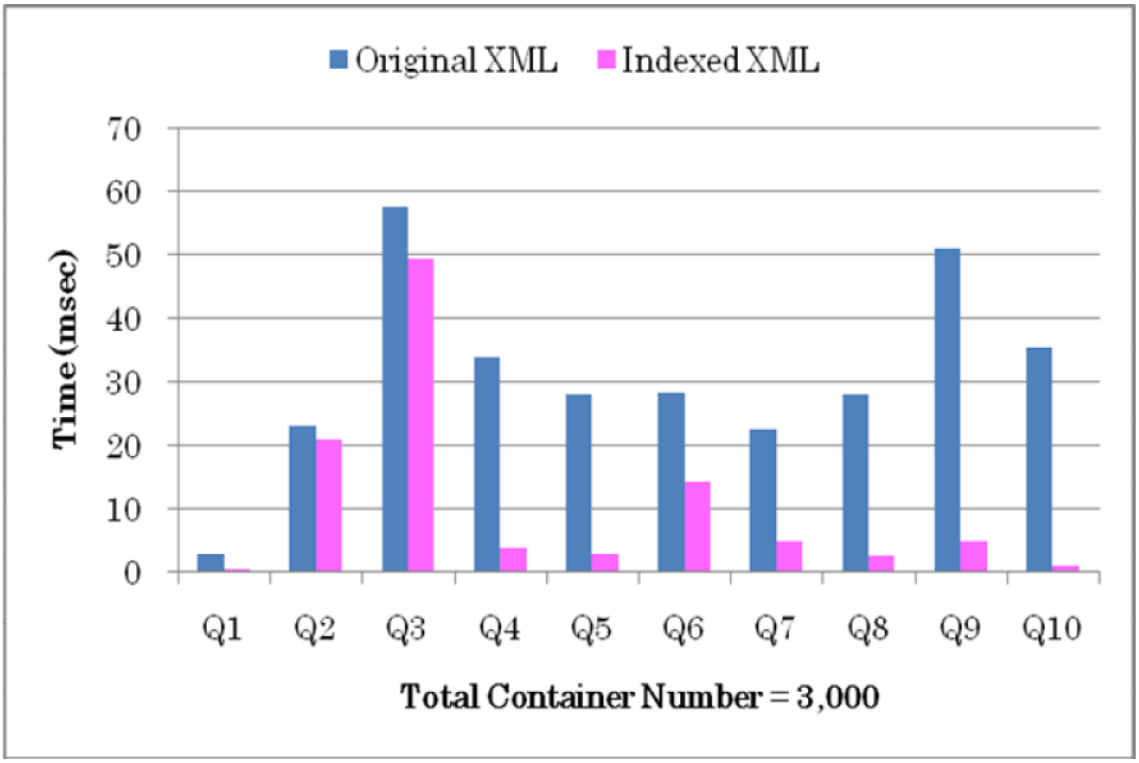


Figure 5-17 XPath querying time comparison while total container number=3,000

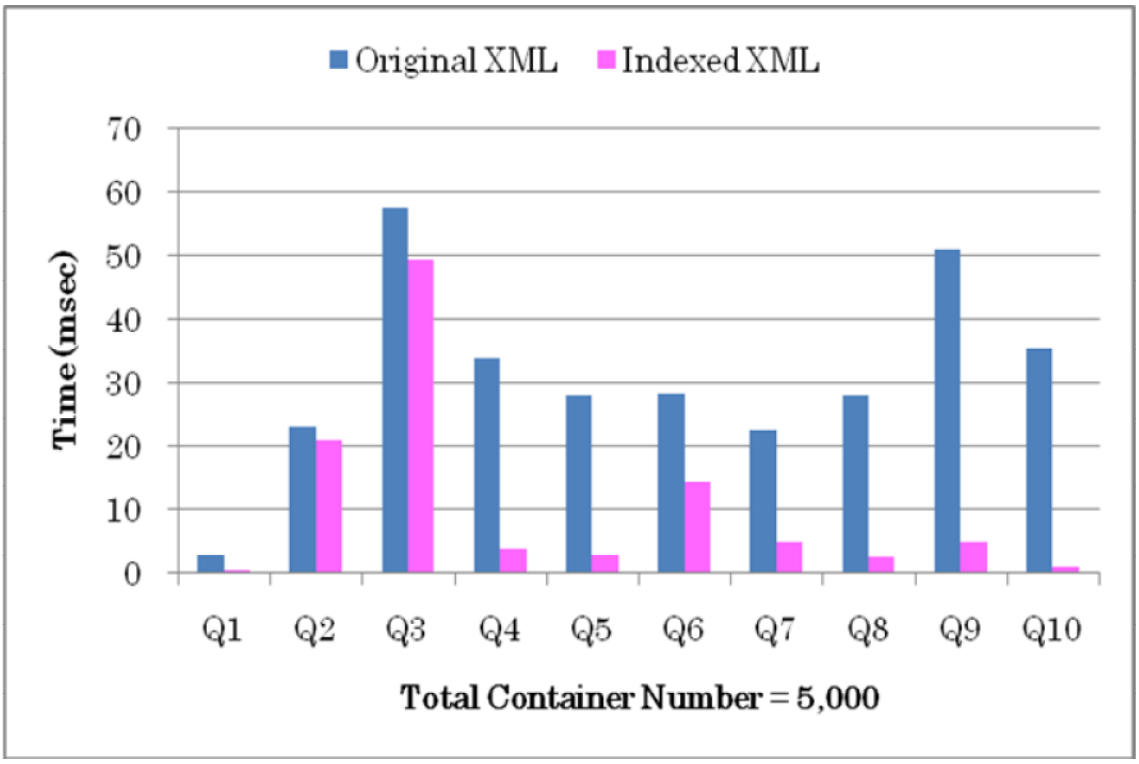


Figure 5-18 XPath querying time comparison while total container number=5,000

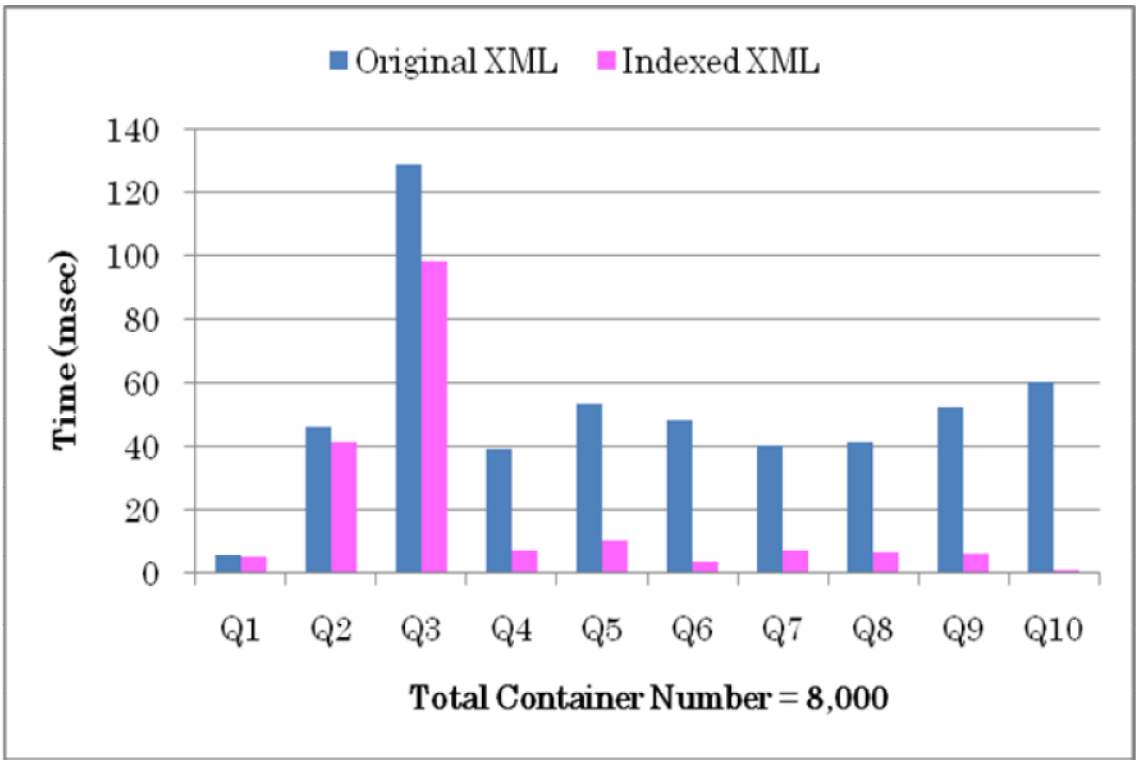


Figure 5-19 XPath querying time comparison while total container number=8,000

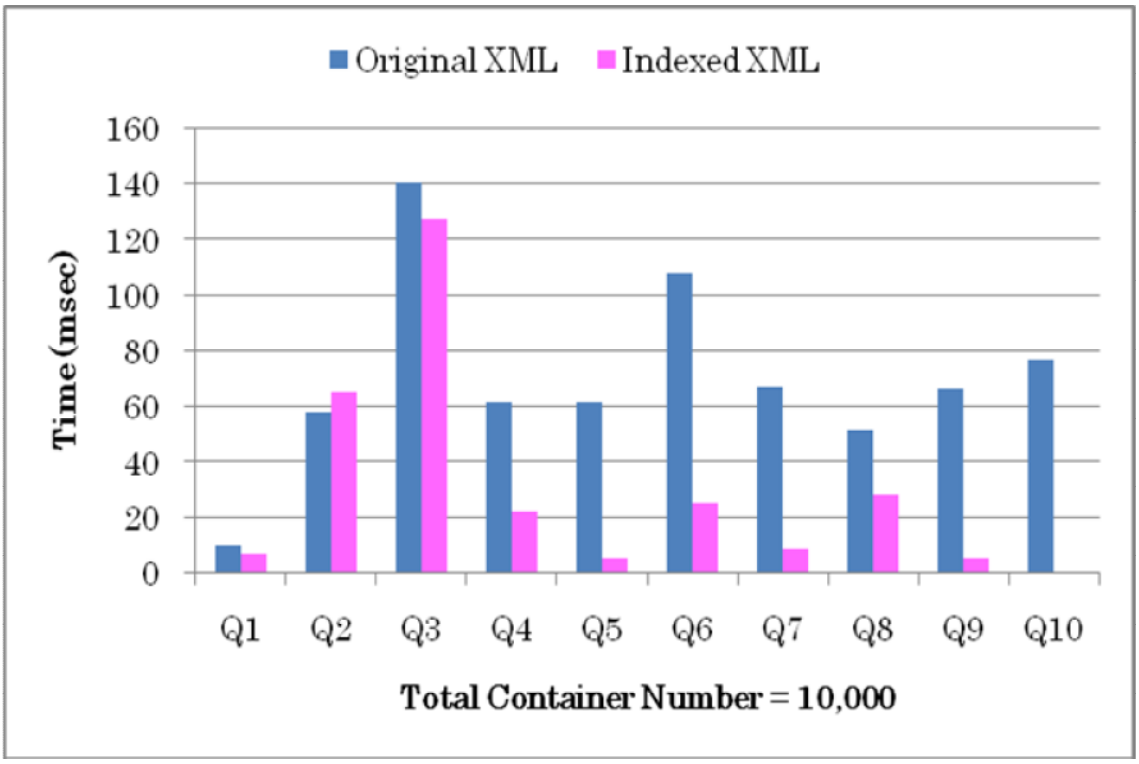


Figure 5-20 XPath querying time comparison while total container number=10,000

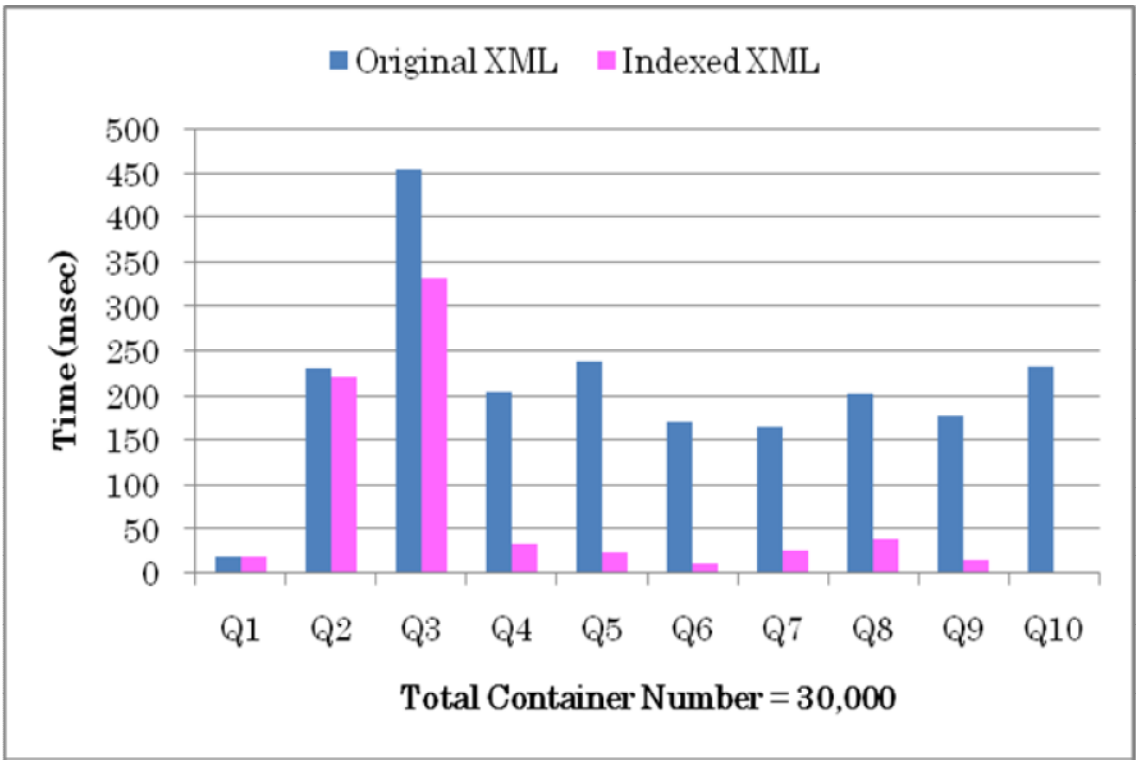


Figure 5-21 XPath querying time comparison while total container number=30,000

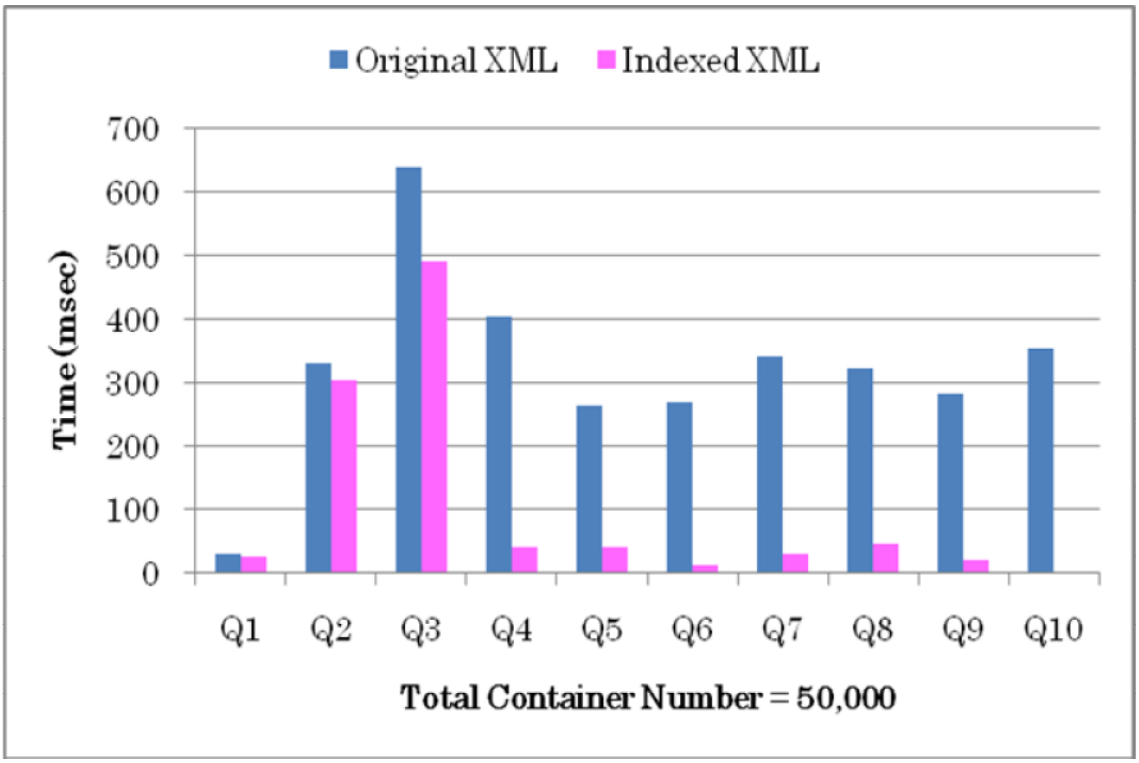


Figure 5-22 XPath querying time comparison while total container number=50,000

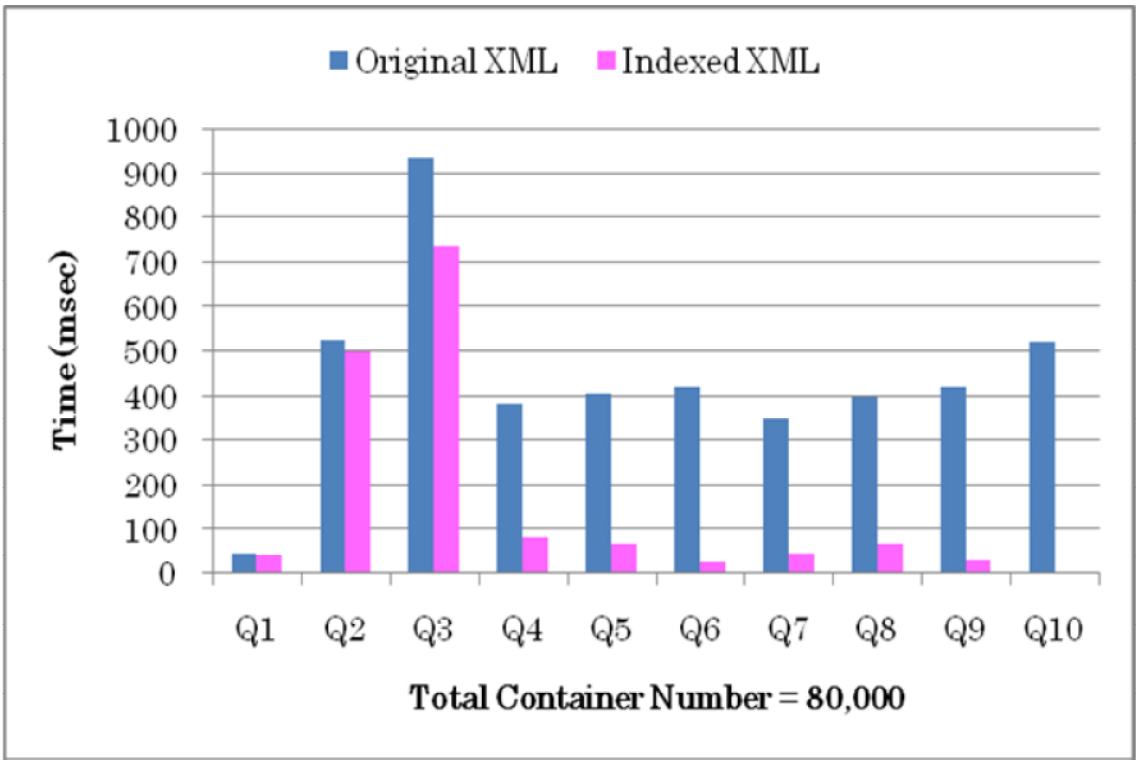


Figure 5-23 XPath querying time comparison while total container number=80,000

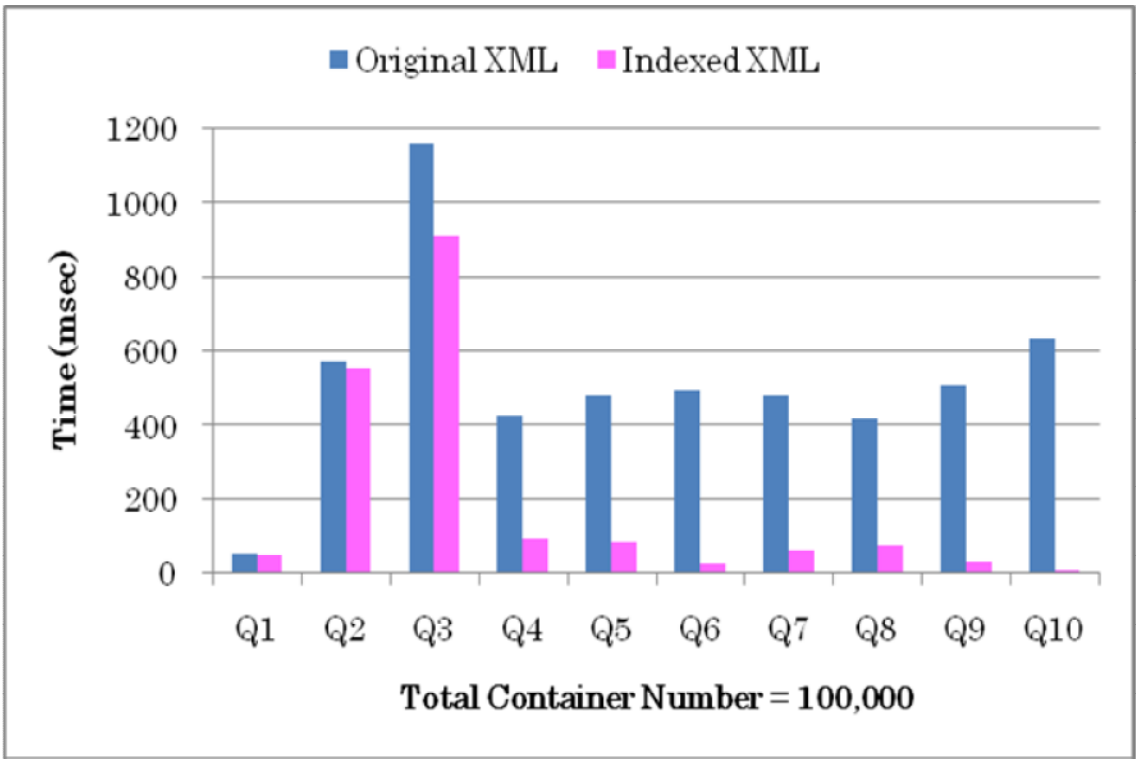


Figure 5-24 XPath querying time comparison while total container number=100,000

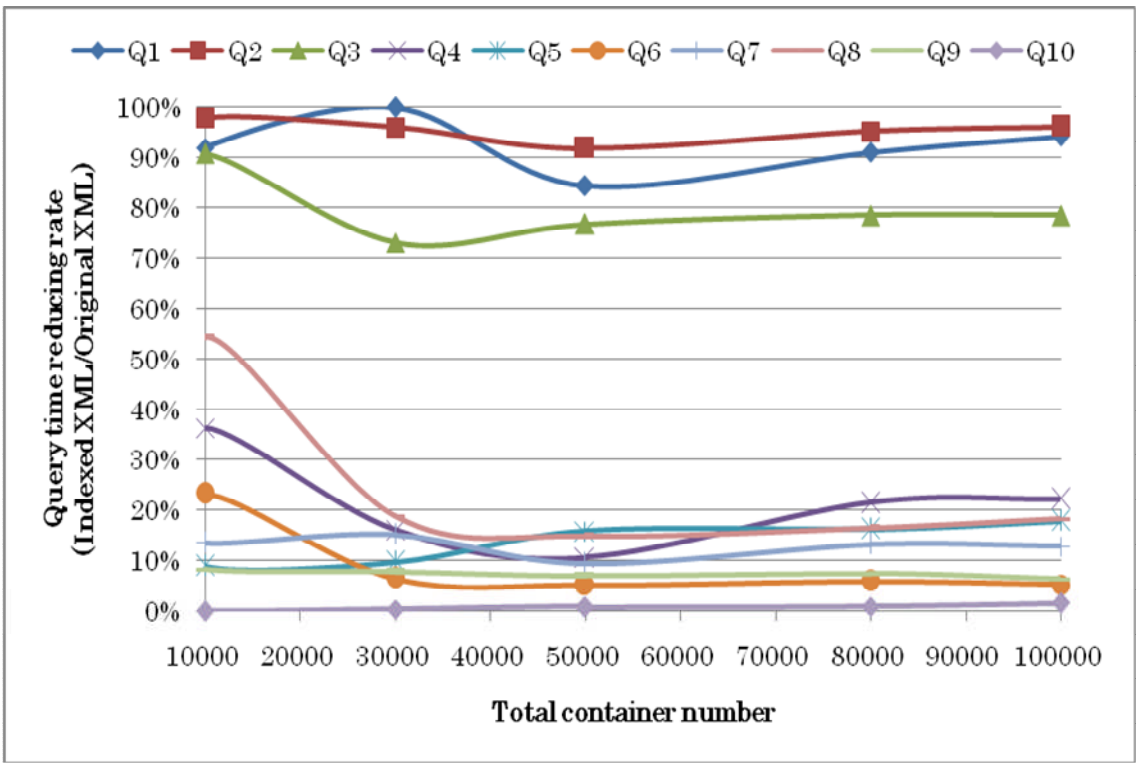


Figure 5-25 Query time reducing rate

From Figure 5-16 to Figure 5-24 show the query time under different total container number. Along with the increasing of total container number, query time also increase. According to Figure 5-25, the query time reducing rate (indexed XML / original XML) of each query mission doesn't show an obvious change along with the increasing of total container number, but mostly depending on the property of each query mission. When the total container number is under 30,000; the reducing rate seems not be stable due to the query time is too short to show the precise results. Along with the increasing of total container number, the reducing rate comes to be a fixed percentage.

Despite of increasing of the query time along with the total container number, the difference would be more obvious when the total container number is high. Figure 5-24 shows an example to explain the evaluation result; the total container number is set as 100,000 in the test datasets. Table 5-5 shows the properties of two datasets, original and indexed container XML documents. After the restructuring procedure, the dataset capacity, line number and element number reduces by 73.7%, 70.7% and 68.8% respectively. We can see that the restructuring procedure effectively reduce the storage space from the results.

Table 5-5 Characteristics of test datasets

	Original XML	Indexed XML
Size (MB)	63.2	46.6
Max depth	3	7
Line#	1,700,033	1,202,216
Element#	1,600,016	1,101,107

The results of each query mission are summarized as follows:

- Q1: query for the total container number.

  Q1 only search until the element "container". For the original XML dataset, it would be the third layer. And for the indexed XML dataset, it would be the seventh layer. But, both of them stop searching for the below various attributes, so this query doesn't spend so much time. Because the indexed XML dataset has a smaller capacity than original XML dataset, so its efficiency is still a little better than original XML dataset.

- Q2: query for a certain container among all containers.

  As Q2 designates a specific container serial number, this query has to search until the lowest layer to check the serial number attribute value. For original XML dataset, this query would need to check about 1.2 million elements. That's why; when compared Q1with Q2; Q1 spent ten times of time for querying. According to the result, indexed XML dataset is still better than original XML dataset.

- Q3: request for the sum of all containers' gross weight.

  Similar to Q2, it has to search all attributes under element "container". But compared with Q2, indexed XML dataset outperform original XML dataset much better than Q2. The reason is the difference of leaf elements' number between these two datasets. The leaf element number under original XML dataset is 12, and indexed XML dataset is 7. This difference effectively reduces the querying time.

- Q4 through Q8: query for the containers with construction contains one index attribute.

  In these cases, the efficiencies of indexed XML datasets are greatly outperform original XML datasets.

- Q9: query for containers with construction contains two index attributes; Q10: query for containers with construction contains three index attributes.

  According to the results, their performance improvements are even better than Q4 through Q8.

From the experimental results, the performances of indexed XML dataset I proposed significantly outperform the original XML dataset. If the request construction contains more grouped index attributes simultaneously, the indexed method will be able to achieve more performance to improve querying efficiency.

The effects of index attribute number and sequence order are discussed in future steps. I changed the number of index attribute into 3, the order into decreasing sequence; the total container number is set to be 100,000. The result of file capacity comparison is shown in Figure 5-26, XPath querying time comparison is also shown in Figure 5-27.

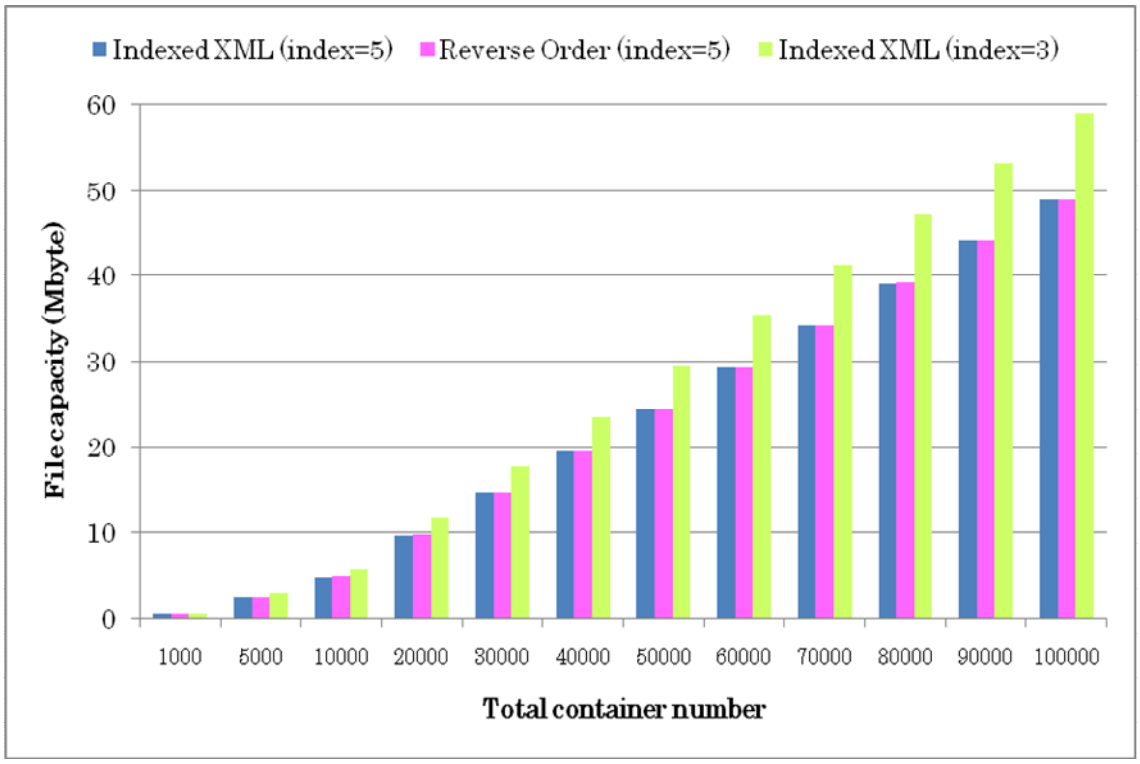


Figure 5-26 File capacity comparison with reverse order and index attribute number

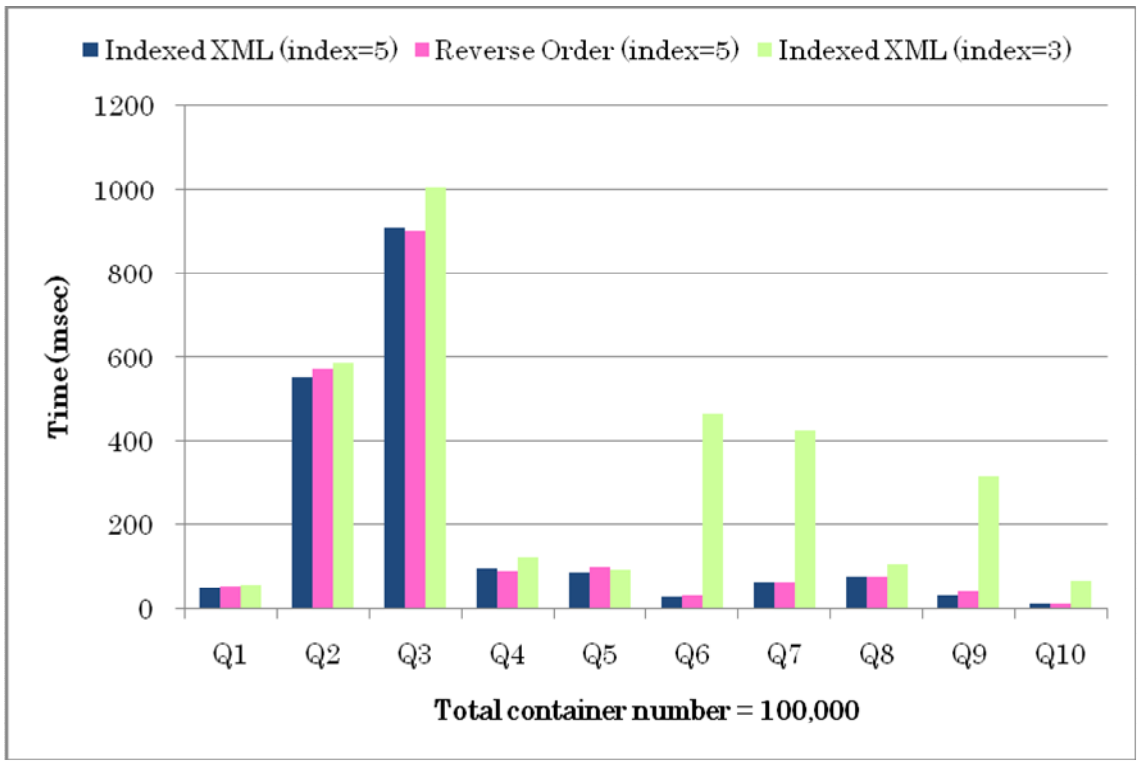


Figure 5-27 XPath querying time comparison with reverse order and index attribute number

The effects of sequence order and index attribute number are described as follows:

- Sequence order

  According to Figure 5-26, the sequence order almost doesn't affect the file capacity at all. Also, compared with previous propose increasing order case with the same index attribute number, their querying time are very close according to Figure 5-27. So, the sequence order is not an important affecting factor to the proposed XML data structure for querying it by XPath. This result could be reasonable because the essence of XPath is to collect the nodes that match the path within XML document. So nowhither how we arrange the sequence of the index attributes, the result node set would has a similar structure.

- Index attribute number

  According to Figure 5-26, less index attribute number obviously increases the file capacity than other two cases. For the querying time level, the less index attribute number cases are a little worse than others; from Q1 to Q5 and Q8, under the same condition (the XPath construction contains the same amount of index attribute number, 1 or 0). On the other hand, the query efficiencies are greatly drawback in Q6, Q7, Q9 and Q10 due to the last two candidates of index attributes are canceled; this makes that their XPath constructions contain less index attributes than other cases, and directly affects the querying efficiencies.

In this chapter, I proposed an index mechanism to group attributes within XML document. The restructuring procedure can effectively reduce XPath querying time and file capacity. In addition, the effects of the number and sequence order of index attributes are considered. The improvement of efficiency is greatly related to the number of index attributes and the properties of XPath constructions. If we can group the frequent used attributes for actual requirements as many as possible, the query efficiency would be improved more effectively.

Chapter 6 Research Contributions and Future Research

Although relational database is a very powerful and popularized method for data storage model, XML is recently growing rapidly because of its various characteristics that relational database doesn’t have. In the Business to Business (B2B) generation, automatic communication among servers is a more important issue than efficiency. By sacrificing some efficiency, XML can achieve various objectives with its semantic structures that relational database is not able to achieve.

In this research, I proposed to adopt XML in maritime field and discussed it with three themes: (1) data storage model, (2) metadata model and (3) efficient XML data structure.

- Data storage model

  The relational database is implemented to achieve the objectives of add, delete or query of datasets by defining and controlling structured data exhaustively. The data are saved by the format of rows based on precise definitions of columns data type, and the query results are the complex combinations through the related data stored in different data tables.

  On the other hand, XML adopts hierarchical tree structure as its storage format, uses element and attribute containing data and XML schema to maintain its correctness. This structural characteristic allows XML to add new columns arbitrarily as long as it matches the definition of XML schema. Under the same situation, relational database has to consider a lot of conditions including the relationship of related data tables, for example the impacts to old data, setup of primary keys, etc. This character makes relational database not as easy to be modified as XML.

  XML is also suitable for handling unstructured data which relational database cannot handle directly but has to convert them into several corresponding data tables. Unfortunately, data required or produced by enterprises are not always structured data.

  In the technical implementation level, nowadays, electronic business enterprises

mostly use relational database to store and manage their data due to relational database technology is already very mature and easy to be implemented in distributed environment. Although the data belong to ships have the necessity of independent self-management themselves for fleet management because ships cannot access to centric database through general internet construction while offshore. The relational database cannot completely satisfy this objective due to it is not easy to be divided.

Although nowadays ships sailing offshore can communicate with overland by using the satellite communication technology; it is still not popularized yet due to the development of the Internet in recent years is just focus on wireless technology. Furthermore, within the computer science field, the technology has not reached a huge market scale is difficult to lead to cost down.

Take JSAT Corporation (the biggest satellite communication supply company in the Pacific Ocean area of Asia) as an example, which provides a SPACE IP service for enterprise clients. The SPACE IP service is composed of two kinds of plans; light plan and standard plan. The light pan provides 5Mbps download speed bandwidth and 1Mbps upload speed bandwidth, the monthly cost is 100,000 Japanese Yen. The standard plan provides 10Mbps of download speed bandwidth and 2Mbps upload speed bandwidth, the monthly cost is 200,000 Japanese Yen. Before starting to use the satellite communication, enterprises also have to pay at least 400,000 Japanese Yen for the construction of equipments. In addition, enterprises have to pay about 30,000 Japanese Yen for renting the communication equipments for the monthly cost.

If we compare the light plan with general internet services (such as ADSL or cable modem), satellite communication is about twenty times higher than general ones for the monthly cost, but the communication speed is only 0.05% among them. Depends on the conditions listed above; the satellite communication technology is considered still not a useful communication way for the maritime field. In this case, a self-manageable data model for ship is a valuable issue that should be discussed.

In chapter 3, I adopted XML to construct the ship data model for fleet management and classify data model by XML tree-structure characteristic. The ship data model treats ships as separable units, and each ship is a child node of the Ship Company. This data model makes ships handle its own data while

sailing independently. Also, two attributes (property and update) for elements are defined in this data model to provide the function of resource classification and the index of data updating. Cooperating with appropriate user permission management, we can achieve the objective of access control and efficient data synchronization. Instead of ongoing Web application interfaces of port administration, an alternative mean to process such kinds of applications is introduced by exchanging XML data that could be validated through pre-defined XML Schemas.

It has been ten years since XML became to be a recommended standard proposed by W3C. Although XML still couldn't replace HTML completely, the wave of XML standardizing for various industries had never stopped. This high speed global internet generation is asking for a more complex and flexible data exchange mechanism now, and obviously XML is the alternative solution after HTML nowadays. For the real cases, the Organization for the Advancement of Structured Information Standards (OASIS) has joined with consortia from the insurance, publishing and human resources industries to develop and promote XML standards particular to each industry. By establishing agreed-upon XML document formats, businesses should be able to more easily exchange information with and between partners or disparate computing systems.

An irreplaceable advantage as data exchange format for XML is its semantic tag mechanism. The semantic structure provides the automatic communicating functions between different machines through enterprises or even industries. This mechanism is based on the common agreements. It's not easy to establish a unified industry standard especially for a traditional industry like maritime business field. But we have already learned the importance of unification from the container operations; the worldwide container standard had improved the productivity and competition dramatically. The XML standard for maritime industry could be another great impact to change the whole sea transportation business activities, and the benefits brought by automatic data exchange will improve the efficiency greatly and is necessary as considered.

- Metadata model

  The definition of metadata is "data about data" in information technology field, which describes the data itself. Metadata is also defined as "structure data of data", which describes the characteristics of a resource. A metadata record

consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values.

Along with intercommunication need growth within the Internet, the scope and role of metadata has been changed. Metadata nowadays should satisfy three needs:

- Not only provide the data abstraction but also improve system intercommunications.
- Allow computers to access and analyze metadata resources automatically.
- Maintain the metadata correctness by confirming with the original data resources periodically.

For metadata attached to Web pages, the standard encoding scheme is HTML. And its inheritor XML supports multiple metadata schemes even much better. The advantages of using XML can be listed as follows:

- Separates data management from data presentation, making both processes more efficient.
- Can handle multiple metadata schemas in the one record.
- Easier for computers to understand.
- Can group elements.
- Supports complex values.
- Gives multilingual support.

Ubiquitous is an important subject of the information technology recently because of the progress of wireless network technology and the miniaturization of handheld equipment; people can access information they need in everywhere and at anytime. However, it causes some problems that plenty of relevant information is not integrated efficiently. In order to solve these problems, using metadata to build the data models is an efficient mean that should be considered.

In chapter 4, considering the benefits of modeling metadata into context model, an object-based context model is designed which can describe simple information or service/operation and a user model both based on XML standard. This modeling cannot only be used for resource finding and enhanced data selection,

but also improve the security of system by preventing users to access resources directly.

In the context model, various attributes are defined to divide information into several levels. User model is used to identify users and achieve the objective of access control. Also, an example scenario was performed to describe behavior patterns of different users/groups within a passenger ship.

A sailing passenger ship can be treated as an isolated unit. Crisis handling mechanism is very important because it cannot expect support from the land. And, ship management is a combination of various complicated operations; a simple carelessness could cause huge crisis or tragedy. So, it is necessary to apply operations on schedule within a sailing passenger ship.

The most important control factor in this research is the time factor in the context model. By defining the inform time, operational initial time, exceptional initial time, exception handle time interval and destruction time, we can passively or actively provide the newest information in simple information level. In service level, we can trigger service by real time and reduce the possibility of unexpected situation by the exception handling mechanism proposed in this chapter.

The proposed models allow public information updating automatically in real time. It can not only passively provide the shared information, but also actively send announces to the related members to make sure that everything is on schedule.

In the real world, even the computer technology has already been developed highly, however there still will be some needs for artificial operations. In this research, an extended context model for services which can be driven by time factor to deal with critical operations is proposed. According to the time trigger mean and corresponding exception handle policy divided by time, missions with specific importance can be handled correctly by its emergency level and eventually maintain the integration and high security of whole system. However this model dose limit some level of the degree of freedom, therefore only some operations with high necessity of security or essential are suggested to be set as critical operations and general operations can still be operated by RBAC model to maintain the flexibility of whole system. Considering about some special high-secured necessity, the implementation of this model is considered valuable.

- Efficient XML data structure

  In the maritime industry, marine operations are not heavily depending on information technologies, so the query efficiency issues have not been paid much attention on. But, compared with the capability of cargo tracking for overland logistic systems, sea transportation systems still cannot achieve such kind of operation precision. Manual management is no longer the gist in B2B generation, but most of complex processes should be automatically handled by servers. Along with the growth of implemented applications in the maritime field, data storage capacity and single query efficiency will affect entire system performance explicitly.

  This study proposed to adopt XML for the maritime field. At the same time, we have to face the serious issues of the inefficient data structure of XML. XML is pure text format; it can be edited by any general text editor without installing any dedicated applications or systems.

  Efficiency is a serious issue for XML compared with relational database. This issue is mostly derived from the differences of stored data type; relational database stores data with binary format and XML uses pure text format. The poor efficiency of XML query could be caused by the unstructured data compared with the normalized data in relational database. Also, the necessary file storage space for the same data would be generally bigger in XML.

  In chapter 5, a new mechanism is also proposed to group the XML document elements by using attributes as leaf element value. By this mean, the storage space could be effectively reduced for XML document. Simultaneously as a result of this, more efficient performance could be gained along with the increasing of duplication of dataset.

  In this research, the test cases composed of 100,000 containers is assumed that the maximum quantity of containers could be handled by single container yard. According to the Japanese main ports analysis report 2008, the quantity of containers handled by Japanese main ports are between 2,309,820 TEU (Osaka Port) and 4,124,140 TEU (Tokyo Port) in one year, so the average handling quantity of containers everyday would be between 6,328 and 11,300 TEU. Although the handling quantity could change following the objectives of each container yard, the test cases are considered to be able to cover the actual implementation in real world for the daily use.

In this research, XPath is adopted as XML document query tool. Compared with SQL (Structured Query Language) for relational database; XPath defines the hierarchical relationship within XML document instead of relational tables. The degree of structuring for the same data would affect the query performance between relational database and XML.

The performance of XML could be affected by various factors such as the size of the XML documents, the amount of script code required to process the documents, the amount of output generated, etc. In addition, the performance of query is application dependent. For example, major variables that can affect the performance of MSXML include:

- The kind of XML Data
- The ratio of tags to text
- The ratio of attributes to elements
- The amount of discarded white space

If the data needed to be stored are mostly structured and static, then relational database would be a better solution as a pure data storage method than XML because of its high efficiency. Otherwise, if the data definitions need to be modified frequently or there are needs for data exchange, XML would have more advantages than relational database schemas. Although XML could be the next generation internet language standard, its adoption should consider the actual needs depending on the considerations of efficiency, security, presentation, convenience, and so on.

Although XML is a powerful tool, it still has several drawbacks listed as follows because of its essence:

- Lack of efficient data storage ability.
- Document index definition.
- Document security issue.
- Multi users access ability.
- Query across multi documents.

For the future work, subjects working for overcoming or avoiding these disadvantages are expected. As the extension of this research, there are several possible research issues as follows:

- XML document automatic generation

  To develop a system that can automatically generate corresponding XML documents by predefined XML schema according to various application forms of on-line port administration for government departments. By using the XML data model structured for the ship, it could be more efficient by reducing the time of artificial inputting and the waste of internet resource. In this way, the port application procedures can be further simplified.

  The bottleneck would appear on the appropriate nodes selection from existed XML document, nodes satisfy XML schema condition could exist at the same layer within the document. The complete automatic XML document generation might not be able to be achieved directly but has to rely on an extra mechanism judging the surrounding environment.

- Ubiquitous services

  For the proposed context model, the time factor plays a very important role. By applying the exception policy, service can react to specific situation for preventing unexpected events. The further analysis of time factor could be analyzed more specifically to search for new possibilities of various implementations. Multi-layer exception policies issue is also another issue that could be followed.

  The biggest issue of ubiquitous services is how to detect the contexts surrounding users such as positions, activities, available resources, etc. Also, if there are multi users and/or resources, the priority and collision issues should be considered.

- Cargo transportation and tracing

  An efficient data structure was proposed for querying XML document. Some operations that highly affected by query efficiency (such as cargo transportation and tracing) should be investigated. Compared with the overland transportation, sea cargo transportation still cannot achieve such kind of automation and precision, so there are a large number of issues to be investigated and discussed within this field.

References

[1] Xin Zhang, Elke A. Rundensteiner, Gail Mitchell and Wang-Chien Lee, “Clock: Synchronizing Internal Relational Storage with External XML Documents”, 11th International Workshop on research Issues in Data Engineering, pp. 111-118, 2001.

[2] Czejdo, B., Miller, R., Taylor, M. and Rusinkiewicz, M., “Distributed processing of queries for XML documents in an agentbased information retrieval system”, Digital Libraries: Research and Practice, 2000 Kyoto, International Conference”, pp. 246-253, 2000.

[3] Sungchul Hong and Yeong-Tae Song, “Efficient XML query using Relational Data Model”, Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference, pp. 1095-1100, 2007.

[4] Daum, T.S. and Sargent, R.G., ” A Web-ready HiMASS: facilitating collaborative, reusable, and distributed modeling and execution of simulation models with XML”, Simulation Conference, 2002. Proceedings of the Winter, pp. 634- 640, 2002.

[5] Thomas Kwok and Thao Nguyen, “An Enterprise Electronic Contract Management System using Dual XML and Secure PDF Documents”, pp. 57-60, 2006.

[6] Lenaghan, A.P. and Malyan, R.R., “XPEN: an XML based format for distributed online handwriting recognition”, Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference, pp. 1270-1274, 2003.

[7] Li-Yan Yuan and Meng Xue, “Xregion: A structure-based approach to Storing XML Data in Relational Databases”, Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference, pp. 544-551, 2007.

[8] Barbara Carminati, Elena Ferrari and Elisa Bertino, “Securing XML data in third-party distribution systems”, Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 99-106, 2005.

[9] Albrecht Schmidt, Florian Waas, Martin Kersten, Daniela Florescu, Michael J. Carey, Ioana Manolescu and Ralph Busse, “Why and how to benchmark XML databases”, ACM SIGMOD Record, pp. 27-32, 2001.

[10] Govindaraju, M., “XML Schemas Based Flexible Distributed Code Generation Framework”, Web Services, 2007. ICWS 2007. IEEE International Conference, pp. 1212-1213, 2007.

[11] Polly M. S. Poon, Tharam S. Dillon and Elizabeth Chang, “XML as a basis for interoperability in Real Time Distributed Systems”, Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS'04), pp. 90-96, 2004.

[12] J. Tucker, W. Alcorn and K. Kaplan, “Development of XML industry standards for information exchange and commerce”, Proceedings of the International Symposium on Electronics and the Environment, pp. 281-286, 2004.

[13] Maria Claudia F. P. Emer, Silvia Regina Vergilio and Mario Jino, “A Testing Approach for XML Schemas”, Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 2 - Volume 02, pp. 57-62, 2005.

[14] Erik Wilde and Felix Michel, “XML-based XML schema access”, Proceedings of the 16th international conference on World Wide Web, pp.1351-1352, 2007.

[15] Antonia Bertolino, Jinghua Gao, Eda Marchetti and Andrea Polini, “Automatic Test Data Generation for XML Schema-based Partition Testing”, Automation of Software Test , 2007. AST '07. Second International Workshop, pp. 4-10, 2007.

[16] T. J. Ostrand and M. J. Balcer, “The category-partition method for specifying and generating functional tests”, Communications of the ACM Volume 31, Issue 6, pp. 676-686, 1988.

[17] Beatrice Bouchou, Ahmed Cheriat, Myrian Halfeld Ferrari and Agata Savary, “XML Document Correction: Incremental Approach Activated by Schema Validation”, Database Engineering and Applications Symposium, 2006. IDEAS '06. 10th International, pp. 228-238, 2006.

[18] Bendeck, F., “Automation of XML documents translators’ generation”, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001. WET ICE 2001. Proceedings. Tenth IEEE International Workshops, pp. 37-38, 2001.

[19] Reynaud, C., Sirot, J.-P. and Vodislav, D., “Semantic integration of XML heterogeneous data sources”, Database Engineering & Applications, 2001 International Symposium, pp. 199-208, 2001.

[20] Dongwon Jeong , Yixin Jing, Jinhyung Kim and Doo-Kwon Baik, “A Framework for Application-Independent Semantic Management for Ubiquitous Computing Environment”, Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, pp. 417-422, 2006.

[21] Bo Jiang and Guozheng Wang, “Remote Awareness of complicated Pattern Group in Ubiquitous Collaborative Graphics Editing System”, Proceeding of the 16th International Conference on Artificial Reality and Telexistence Workshops (ICAT’06), pp. 35-38, 2006.

[22] Dongkwang Kim, Karpjoo Jeong, Hyoseop Shin and Suntae Hwang, “An XML Schema-based Semantic Data Integration”, Proceedings of the Fifth International Conference on Grid and Cooperative Computing, pp. 522-525, 2006.

[23] Davood Rafiei, Daniel L. Moise and Dabo Sun, “Finding Semantic Similarities Between Xml Documents”, Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA’06), pp. 68-72, 2006.

[24] Shaorong Liu, Qinghua Zou and Wesley W. Chu, “Configurable indexing and ranking for XML information retrieval”, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 88-95, 2004.

[25] Goeschka, K.M., Reis, H. and Smeikal, R., “XML based robust client-server communication for a distributed telecommunication management system”, System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference, pp. 10-19, 2003.

[26] Jagannathan, V. and Fuchs, M., “Workshop report on integrating xml & distributed object technologies”, Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings. IEEE 8th International Workshops, pp. 291-296, 1999.

[27] Hiroto Kurita, Kenji Hatano, Jun Miyazaki and Shunsuke Uemura, “Efficient Query Processing for Large XML Data in Distributed Environments”, Proceedings of the 21st International Conference on Advanced Networking and Applications, pp. 317-322, 2007.

[28] Roussev, V., Dewan, P., Koorakula, N. and Sellappa, S., “Integrating XML and object-based programming for distributed collaboration”, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000. (WET ICE 2000). Proeedings. IEEE 9th International Workshops, pp. 254-259, 2000.

[29] Koglin, Y., Mella, G., Bertino, E. and Ferrari, E., “An Update Protocol for XML Documents in Distributed and Cooperative Systems”, Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference, pp. 314-323, 2005.

[30] Luttenberger, N., Reuter, F. and Koberstein, J., “XML language binding support for pervasive communication in distributed virtual shared information spaces”,

Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference, pp. 181-186, 2004.

[31] Zisman, A., Emmerich, W. and Finkelstein, A., “Using XML to build consistency rules for distributed specifications”, Software Specification and Design, 2000. Tenth International Workshop, pp. 141-148, 2000.

[32] Hiroyuki Uchiyama, Makoto Onizuka and Takashi Honishi, “Distributed XML Stream Filtering System with High Scalability”, 21st International Conference on Data Engineering (ICDE'05), pp. 968-977, 2005.

[33] Cheung, D., Lee, S.D., Lee, T., Song, W. and Tan, C.J., “Distributed and scalable XML document processing architecture for E-commerce systems”, Advanced Issues of E-Commerce and Web-Based Information Systems, 2000. WECWIS 2000. Second International Workshop, pp. 152-157, 2000.

[34] Melbourne, Victoria and Australia, “Employing hierarchical federation communities in the virtual ship architecture”, Proceedings of the twenty-fifth Australasian conference on Computer science - Volume 4, pp. 41-49, 2002.

[35] Junqi Zhang, Vijay Varadharajan and Yi Mu, “Securing XML Document Sources and Their Distribution”, Proceedings of the 18th International Conference on Advanced Information Networking and Applications - Volume 2, pp. 562-567, 2004.

[36] Srinon, R. and Ramakrishnan, S., “Distributed simulation modeling for manufacturing systems design using XML”, Systems Engineering, 2005. ICSEng 2005. 18th International Conference, pp. 395-400, 2005.

[37] Jinsuo Zhang and Sumi Helal, “UbiNet: A Generic and Ubiquitous Service Provider Framework”, Software Engineering Advances, International Conference, pp. 11-15, 2006.

[38] U. P. Kulkarni, J. V. Vadavi, S. M. Joshi and A. R. Yardi, “Ubiquitous Object Categorization and Identity”, Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce, pp. 81-86, 2006.

[39] Younghee Kim and Keumsuk Lee, “A Quality Measurement Method of Context Information in Ubiquitous Environments”, Proceedings of the 2006 International Conference on Hybrid Information Technology - Volume 02, pp. 576-581, 2006.

[40] Stephen J.H. Yang, Angus F.M. Huang, Rick Chen, Shian-Shyong Tseng and Yen-Shih Shen, “Context Model and Context Acquisition for Ubiquitous Content Access in ULearning Environments”, IEEE International Conference on Sensor

Networks, Ubiquitous, and Trustworthy Computing - Vol 2 - Workshops, pp. 78-83, 2006.

[41] Iacob, Sorin M., Bargh and Mortaza S., “Key Issues in the Optimization of Runtime Processes for Dynamic Provisioning of Ubiquitous Services”, Computing in the Global Information Technology, ICCGI '06. International Multi-Conference, pp. 23-31, 2006.

[42] Qiu-Sheng He, Xun Chen and Shi-Liang Tu, “Research of a Goal-Driven Architecture in Ubiquitous Environments”, Computer and Information Technology, 2006. CIT '06. The Sixth IEEE International Conference, pp. 237-242, 2006.

[43] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann and Werner Retschitzegger, “Context-Awareness on Mobile Devices - the Hydrogen Approach”, Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9, pp. 292-301, 2003.

[44] Nam-Shik Park, Kang-Woo Lee and Hyun Kim, “A Middleware for Supporting Context-Aware Services in Mobile and Ubiquitous Environment”, Proceedings of the International Conference on Mobile Business, pp. 694-697, 2005.

[45] Jinwoo Park, Jong-Kwon Lee, YoungSang Paik, MyungChul Lee and Chandra Narayanaswami, “Device Context Discovery System for Context-Aware Services in Ubiquitous Device Collaboration Environment”, Proceedings of the International Workshop on System Support for Future Mobile Computing Applications, pp. 25-34, 2006.

[46] Mizuho Iwaihara, Somchai Chatvichienchai, Takayuki Shiga and Yahiko Kambayashi, “XML Access Control and e-Commerce Technologies for Advanced Information Distribution”, Proceedings of the International Conference on Informatics Research for Development of Knowledge Society Infrastructure, pp. 107-115, 2004.

[47] JeongWon Cho and EenJun Hwang, “An Exhibition Reminiscent System for Ubiquitous Environment”, Proceedings of the Sixth IEEE International Conference on Computer and Information Technology, pp. 241-246, 2006.

[48] Sung-Je Park, Soon-Young Bae, Jae-Hoon Jin, Jong-hwan Suh and Sang-Chan Park, “Ubiquitous Water Recycle Management Service Proposal in Ubiquitous City”, Artificial Reality and Telexistence--Workshops, 2006. ICAT '06. 16th International Conference, pp. 558-561, 2006.

[49] Donwook Shin and Choon Shim, “XNMP - An XML Based Network

Management Protocol over VoIP", Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, pp. 208-213, 2005.

[50] Maldonado, J.A., Robles, M. and Cano, C., "Integration of distributed healthcare information systems: application of CEN/TC251 ENV13606", Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE, pp. 3731-3734. 2001.

[51] Paterson, G.I., Shepherd, M., Xiaoli Wang, Watters, C., and Zitner, D., "Using the XML-based Clinical Document Architecture for exchange of structured discharge summaries", System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference, pp. 1200-1209, 2002.

[52] William M. Shui, Raymond K. Wong, Stephen C. Graham, Lawrence K. Lee and W. Bret Church, "Integrating, Managing and Analyzing Protein Structures with XML Databases", Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03), pp. 319-326, 2003.

[53] Shoji Kajita and Kenji Mase, "uClassroom: Expanding Awareness in Classroom to Ubiquitous Teaching and Learning", Fourth IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education (ICHIT'06), pp. 161-169, 2006.

[54] Jihen Malek, Mona Laroussi and Alain Derycke, "A Multi-Layer Ubiquitous Middleware for Bijective Adaptation between Context and Activity in a Mobile and Collaborative learning", Proceedings of the International Conference on Systems and Networks Communication, pp. 39-44, 2006.

[55] Michael Eisenberg, Ann Eisenberg, Leah Buechley and Nwanua Elumeze, "Invisibility Considered Harmful: Revisiting Traditional Principles of Ubiquitous Computing in the Context of Education", Proceedings of the Fourth IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education, pp. 103-110, 2006.

[56] Shengguo Sun, Zixue Cheng, Lei Jing, Tongjun Huang and Hui-Huang Hsu," A Ubiquitous Learning System for Improving Quality of Learner's Behaviors Based on Shaping Principle", Computer and Information Technology, 2006. CIT '06. The Sixth IEEE International Conference, pp. 216-221, 2006.

[57] Reinhard Ruemer and Klaus Miesenberger, "Using XML for Publishing on Demand in Different Output Formats", Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. AXMEDIS '06. Second

International Conference, pp. 153-156, 2006.

[58] Kyung-Lang Park, Joo-Kyoung Park, Chang-Deok Kang, Hoon-Ki Lee, Eui-Hyun Baek and Shin-Dug Kim, “VPW: An Effective Personal Space Model for Providing Ubiquitous”, Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, pp. 428-435, 2006.

[59] Changbok Jang, Sunghun Cho, Moohun Lee and Euiin Choi, “A DUPS on Ubiquitous Computing”, Hybrid Information Technology, 2006. ICHIT'06. Vol 2. International Conference, pp. 489-496, 2006.

[60] Ying Yang and Jia-jin Le, “Catalog service engine for XML data sources in distributed systems”, Distributed Frameworks for Multimedia Applications, 2005. DFMA '05. First International Conference, pp. 8-14, 2005.

[61] Grabs, T., Bohm, K. and Schek, H.-J., “Scalable distributed query and update service implementations for XML document elements”, Research Issues in Data Engineering, 2001. Proceedings. Eleventh International Workshop, pp. 35-42, 2001.

[62] Bertino, E. and Sandhu, R., “Database security - concepts, approaches, and challenges”, Dependable and Secure Computing, IEEE Transactions, pp. 2-19, 2005

[63] Kyu-il Kim, Hyun-Sik Hwang, Hyuk-Jin Ko, Hae-Kyung Lee and Ung-mo Kim, “Multi-Policy Access control considering Privacy in Ubiquitous Environment”, Hybrid Information Technology, 2006. ICHIT '06. Vol1. International Conference, pp. 216-222. 2006.

[64] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell and Klara Nahrstedt, “Gaia: a middleware platform for active spaces”, ACM SIGMOBILE Mobile Computing and Communications Review, pp. 65-79, 2002.

[65] Antonio Corradi, Rebecca Montanari and Daniela Tibaldi, “Context-Based Access Control for Ubiquitous Service Provisioning”, 28th Annual International Computer Software and Applications Conference (COMPSAC'04), pp. 444-451, 2004.

[66] Sampemane, G., Naldurg and P., Campbell, R.H., “Access control for Active Spaces”, Computer Security Applications Conference, 2002. Proceedings. 18th Annual, pp. 343- 352, 2002.

[67] Antonio Corradi, Rebecca Montanari and Daniela Tibaldi, “Context-Based Access Control Management in Ubiquitous Environments”, Network Computing and

Applications, Third IEEE International Symposium on (NCA'04), pp. 253-260, 2004.

[68] Weider D. Yu, “An Intelligent Access Control for Web Services Based on Service Oriented Architecture Platform”, Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06) - Volume 00, pp. 190-198, 2006.

[69] Shigetoshi Yokoyama, Eiji Kamioka and Shigeki Yamada, “An Anonymous Context Aware Access Control Architecture For Ubiquitous Services”, Proceedings of the 7th International Conference on Mobile Data Management, pp. 74-81, 2006.

[70] Ramiro Liscano and Kaining Wang, “A SIP-based architecture model for contextual coalition access control for ubiquitous computing”, Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference, pp. 384- 392,

[71] Eve Cohen, Roshan K. Thomas, William Winsborough and Deborah Shands, “Models for coalition-based access control (CBAC)”, Proceedings of the seventh ACM symposium on Access control models and technologies, pp. 97-106, 2002.

[72] Tae-Hun Lim and Sang-Uk Shin, “Intelligent Access Control Mechanism for Ubiquitous Applications”, Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference, pp. 955-960, 2007.

[73] Vincent C. Hu, D. Richard Kuhn and David F. Ferraiolo, “The Computational Complexity of Enforceability Validation for Generic Access Control Rules”, Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06) - Volume 01, pp. 260-267, 2006.

[74] Lopez, D.R. and Castro-Rojo, R., “Ubiquitous Internet access control: the PAPI system”, Database and Expert Systems Applications, 2002. Proceedings. 13th International Workshop, pp. 441-445, 2002.

[75] N.R. Adam, V. Atluri, E. Bertino and E. Ferrari, “A Content-Based Authorization Model for Digital Libraries”, IEEE Transactions on Knowledge and Data Engineering March/April 2002 (Vol. 14, No. 2), pp. 296-315, 2002.

[76] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi and Pierangela Samarati, “A fine-grained access control system for XML documents”, ACM Transactions on Information and System Security (TISSEC), pp. 169-202, 2002.

[77] Mizuho Iwaihara, “Access Control of XML Documents and Business Rule Processing for Advanced Information Exchange”, Proceedings of the Second International Conference on Informatics Research for Development of Knowledge Society Infrastructure, pp. 177-184, 2007.
[78] Jingzhu Wang and Sylvia L. Osborn, “A role-based approach to access control for XML databases”, Proceedings of the ninth ACM symposium on Access control models and technologies, pp. 70-77, 2004.
[79] Brian Shields and Owen Molloy, “Using Description Logic and Rules to Determine XML Access Control”, Proceedings of the 18th International Conference on Database and Expert Systems Applications, pp. 718-724, 2007.
[80] Jason Crampton, “Applying hierarchical and role-based access control to XML documents”, Proceedings of the 2004 workshop on Secure web service, pp. 37-46, 2004.
[81] XML - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/XML#Well-formed_and_valid_XML_documents.
[82] Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, Makato Murata, Andy Clark, Yuichi Nakamura, Ryo Neyama, Kazuya Kosaku and Satoshi Hada, “XML and Java, Second Edition: Developing Web Application”, Person Education Japan, 2002.
[83] W3Schools Online Web Tutorials, http://www.w3schools.com.
[84] XML.com: XML From the Inside Out, http://www.xml.com/index.csp
[85] The Official W3C XML Site, http://www.w3.org/XML
[86] Apache Xindice 1.0, http://www.apache/org/xindice, 2002.
[87] Extensible Stylesheet Language (XSL) Version 1.0, W3C Recommendation, October 2001, http://www.w3.org/TR/2001/REC-xsl-20011015/.
[88] The Official SAX 2.0 Site, http://www.saxproject.org/.
[89] Megginson Technologies: Simple API for XML , http://www.megginson.com/downloads/SAX/
[90] Simple API for XML, http://en.wikipedia.org/wiki/Simple_API_for_XML.
[91] XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xpath.
[92] DB2 Database for Linux, UNIX, and Windows, IBM, http://www.ibm.com/software/data/db2/udb/support/.
[93] Kouhei Hirono, “A study on the computer-based agent to assist watch-keeping tasks”, The journal of Japan institute of navigation, pp. 21-28, 2002.3.
[94] Kouhei Hirono, “Development of the agent system to assist a navigator onboard

and collect operational logs for shore based process management", The journal of Japan institute of navigation, pp. 1-10, 2003.3.

[95] M.J. Moyer and M. Abamad, "Generalized Role-Based Access Control", Proc. 21st International Conference on Distributed Computing System, pp. 391-398, 2001.

[96] "The optimization project of import/export and port/air port related procedure operation system" (in Japanese), 16th Chief Information Officer meeting conclusion, 2005.12.28.

[97] B. Schilit, N. Adams and R. Want, "Context-Aware Computing Applications", Proc. of the Workshop on Mobile Computing System and Applications, pp. 85-90, 1994.

[98] H. Packard, Cooltown Project, http://www.cooltown.com/cooltown/index.asp, 2004.

[99] A.Dey, "The Context Toolkit", http://www.cs.berkeley.edu/~dey/context.html, 2004.

[100] S.K Mostefaoui, A.T. Bouzid and B. Hirsbrunner, "Using Context Information for Service Discovery and Composition", Proc. of 5th Conference on Information Integration and Web-based Applications and Services, pp. 129-138, 2003.

[101] M. Roman et al, "A Middleware Infrastructure to Enable Active Spaces", IEEE Pervasive Computing, Vol.1, No.4, pp. 74-83, 2002.

[102] S. Kouadri et al, "Context Aware Service Provisioning", Proc. of the IEEE International Conference on Pervasive Services, pp. 71-80, 2004.

[103] T. Lemlouma et al, "The Negotiation of Multimedia Content Services in Heterogeneous Environments", Proc. of 8th International Conference on Multimedia Modeling, pp. 187-206, 2001.

[104] T. Broens et al, "Context-aware, Ontology-based, Service Discovery", Proc. of 2nd European Symposium on Ambient Intelligence, pp. 72-83, 2004.

[105] M. Khedr and A. Karmouck, "Negotiating Context Information in Context-Aware Systems", IEEE Intelligent Systems, Vol.19, No.6, pp. 21-29, 2004.

[106] M. Khedr, "A Semantic-Based, Context-Aware Approach for Service-Oriented Infrastructures", Proc. of 2nd IFIP International Conference on Wireless and Optical Communications Networks, pp. 584-588, 2005.

[107] XSL Transformations (XSLT) Version 2.0, W3C Recommendation 23 January 2007, http://www.w3.org/TR/xslt20/#xslt-mime-definition.

[108] Alfred Kobsa and Wolfgang Wahlster, "User models in dialog systems", Springer-Verlag New York, Inc., 1990.

[109] R. Goldman and J. Widom, “Dataguides: Enabling query formulation and optimization in semistructured databases”, VLDB, 1997.
[110] Q. Li and B. Moon, “Indexing and querying XML data for regular path expression”, VLDB, 2001.
[111] Qinghua Zou, Shaorong Liu, and Wesley W.Chu, “Ctree: A Compact Tree for Indexing XML Data”, Workshop On Web Information And Data Management, pp. 39-46, 2004.
[112] Marc Levinson, “The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger”, Princeton University, 2006.
[113] “Container Handbook” (in Japanese), Foundation Offshore Container Community, 1995.3.
[114] Otohei Amada, “Know How of Port Transportation” (in Japanese), Seizando bookstore, 2002.3.
[115] Ikuo Tamura, “Explanation of Port Transportation Business” (in Japanese), Seizando bookstore, 2004.11.

List of Published Papers

(1) Cheng-Tung Yang, Chia-Hung Shih, Yan-Liang Chen and Tso-Chung Sung: “The Display of working process for ship structures through Internet, Chinese Society of Mechanical Engineers”, Vol.21, pp.183-191, 2004.11. (in Chinese)

(2) Cheng-Tung Yang, Yan-Liang Chen and Chia-Hung Shih: “Solving Scheduling Problem with SVG technology and Distributed Framework based on Internet Environment”, Journal of Taiwan Society of Naval Architects and Marine Engineers, Vol.17, pp.79-84, 2005.03. (in Chinese)

(3) Chia-Hung Shih, Nobukazu Wakabayashi and Youhei Tamura: “Improvement of Search Procedure for Course Line Generation System using Course Line Parts”, The Journal of Japan Institute of Navigation, Vol. 114, pp.73-80, 2006.3. (in Japanese)

(4) Nobukazu Wakabayashi, Juri Harada, Chia-Hung Shih and koji urai： “Discussion of Functions, Usability and Providing Information for Ship Navigation System”, The Journal of Japan Institute of Navigation, Vol. 117, pp.49-58, 2007.9. (in Japanese)

(5) Chia-Hung Shih, Nobukazu Wakabayashi and Saburo Yamamura: “Construction of Context and User Models under Ubiquitous Environment and its Application for Onboard Information Management”, Asia Navigation Conference 2007, pp.260-268, 2007.11.

(6) Chia-Hung Shih, Nobukazu Wakabayashi and Saburo Yamamura: “A Distributed Data Model for Port Administration and Onboard Information & Service Management”, The Journal of Japan Institute of Navigation, Vol. 118, pp.65-72, 2008.3.

(7) Chia-Hung Shih, Nobukazu Wakabayashi, Saburo Yamamura and Juri Harada: “An Efficient Data Structure for Querying XML Document in Maritime Field”, The Journal of Japan Institute of Navigation, Vol. 119, pp.89-89, 2008.9.