SCHEMA EXPORTATION AND INTEGRATION FOR ACHIEVING INFORMATION SHARING IN A TRANSNATIONAL SETTING

By

MANJIRI PANDURANG PATIL

A THESIS PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2005
Copyright 2005

by

Manjiri Pandurang Patil
I dedicate this thesis to my beloved parents.
ACKNOWLEDGMENTS

Research for this Transnational Digital Government project is supported by grant EIA-0131886 from the National Science Foundation. I would like to thank Dr. Stanley Y. W. Su (my supervisory committee chair) for giving me an opportunity to work on this thesis topic and for his valuable guidance and support in this research effort. I would also like to express my gratitude to Dr. Jose Fortes (supervisory committee member) for his feedback and guidance, during the design and implementation phases of my research work. I would like to thank Dr. Herman Lam for serving on my supervisory committee.

I would like to extend my gratitude to Mauricio Tsugawa and Andrea Matsunaga, Ph.D. students from the Advanced Computing and Information Systems Laboratory (ACIS Lab) for their help in integrating our system with the Machine Translation System and the Conversational Interface System. I am also grateful to all of my friends for their help and support.

Most of all, I would like to thank my beloved family for their love, support, constant encouragement, and blessings. They made this thesis possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td></td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background and Motivation</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1.2 Challenges and Approach Taken</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>2 OVERALL SYSTEM ARCHITECTURE</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2.1 Participating Sites</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2.2 Databases</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2.3 Export Schema Tool</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>2.4 Schema Integration Tool and the Generation of a Global Schema</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>2.5 Event Server</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>2.6 Event Trigger Rule Server (ETR)</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>2.7 Distributed Query Processor</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>2.8 Integration with the Language Translation System</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>2.9 Integration with the Conversational Interface System</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>2.10 Authorization of Agents in the Watch-List Scenario</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>2.11 Short Message Service Center</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>3 DETAILED DESIGN</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3.1 Export Schema Tool</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.2 Schema Integration Tool to Generate Global Schema</td>
<td></td>
<td>23</td>
</tr>
</tbody>
</table>
4 DISTRIBUTED QUERY PROCESSOR AND WATCH-LIST SCENARIO ..........27
  4.1 Global Query Processor (GQP) ..............................................................27
    4.1.1 Global Schema Files ........................................................................27
    4.1.2 Country Information ........................................................................29
    4.1.3 Tomcat Authentication and User Profile Information .........................29
    4.1.4 Global Search Form .........................................................................31
  4.2 Local Query Processor (LQP) .................................................................33
    4.2.1 Web Service Interface for LQP ..........................................................34
    4.2.2 The Wrapper Associated with the LQP ..............................................34
  4.3 Watch-List Scenario ...............................................................................35

5 IMPLEMENTATION DETAILS ........................................................................38
  5.1 Export Schema Tool ................................................................................38
  5.2 Integrate the Local Schemas to Generate the Global Schema .................41
  5.3 Distributed Query Processor .................................................................44
    5.3.1 Global Query Processing Component (GQPC) ..................................44
    5.3.2 Local Query Processing Component (LQPC) ....................................46
  5.4 Watch-List Scenario ...............................................................................47
  5.5 Translation System ..................................................................................49
  5.6 Conversational Interface System ............................................................50

6 TECHNOLOGIES USED .................................................................................51
  6.1 The Events, Triggers, and Rules (ETR) Technology ..................................51
    6.1.1 Events, Triggers, and Rules ...............................................................52
    6.1.2 Event Manager ................................................................................52
    6.1.3 The ETR Server ...............................................................................53
    6.1.4 Knowledge Profile Manager (KPM) ...............................................53
    6.1.5 Persistent Object Manager (POM) ....................................................53
  6.2 The Java Technologies .............................................................................54
    6.2.1 Java Servlet Technology ..................................................................54
    6.2.2 Java Server Pages (JSP) Technology ..............................................54
    6.2.3 Tomcat ............................................................................................55
  6.3 The XML-Related Technologies ...............................................................55
    6.3.1 Extensible Markup Language (XML) ..............................................55
    6.3.2 Xerces: XML Parsers in Java ..........................................................55
  6.4 Web Services ..........................................................................................56
  6.5 Simple Object Access Protocol (SOAP) ..................................................56
  6.6 Apache Axis ............................................................................................57
  6.7 The MySQL Database .............................................................................57
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1.</td>
<td>Dominican Republic’s sample exported schema</td>
<td>21</td>
</tr>
<tr>
<td>3-2.</td>
<td>Belize’s sample exported schema</td>
<td>22</td>
</tr>
<tr>
<td>3-3.</td>
<td>Attribute mappings in the global schema</td>
<td>26</td>
</tr>
<tr>
<td>4-1.</td>
<td>Role list based on access privileges</td>
<td>30</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1. Virtual collaboration grids</td>
<td>2</td>
</tr>
<tr>
<td>2-1. Overall system architecture of the prototype system</td>
<td>9</td>
</tr>
<tr>
<td>3-1. Export schema and integrate exported schemas</td>
<td>15</td>
</tr>
<tr>
<td>3-2. The export schema flow chart</td>
<td>18</td>
</tr>
<tr>
<td>3-3. The add attributes to export schema flow chart</td>
<td>19</td>
</tr>
<tr>
<td>3-4. Generate global schema by mapping the exported attributes</td>
<td>25</td>
</tr>
<tr>
<td>4-1. Architecture of the Distributed Query Processor</td>
<td>28</td>
</tr>
<tr>
<td>4-2. Global search form at the Belize site</td>
<td>31</td>
</tr>
<tr>
<td>4-3. The Watch-List Scenario flow chart</td>
<td>36</td>
</tr>
<tr>
<td>5-1. Format of the exported_attributes.xml file</td>
<td>39</td>
</tr>
<tr>
<td>5-2. Export schema page</td>
<td>40</td>
</tr>
<tr>
<td>5-3. Add attribute to export schema page</td>
<td>41</td>
</tr>
<tr>
<td>5-4. Published schemas page</td>
<td>42</td>
</tr>
<tr>
<td>5-5. Mapping page for exported attributes</td>
<td>43</td>
</tr>
<tr>
<td>5-6. Sample XML query</td>
<td>465</td>
</tr>
<tr>
<td>5-7. Sample XML result</td>
<td>46</td>
</tr>
<tr>
<td>5-8. Query results displayed at the Belize site</td>
<td>46</td>
</tr>
<tr>
<td>5-9. Create authorized agents’ list page</td>
<td>48</td>
</tr>
<tr>
<td>5-10. Arrival form at the port of entry in a Dominican Republic border station</td>
<td>49</td>
</tr>
</tbody>
</table>
There is an urgent need for collaborations among governments of various countries to tackle global problems such as drug trafficking, disease control, immigration and border control, and terrorism. The Transnational Digital Government project (funded by the National Science Foundation) aims at collaborating, integrating, and sharing information among governments/agencies using information technologies. The transnational government collaboration faces many challenges, because individual countries differ in their languages, laws, policies and regulations, infrastructures, and other resources. Our study focused on the development and use of advanced information technologies for the collection, processing, exchange, and integration of the information needed in a transnational digital government setting.

We developed distributed database technologies to support the needs of the project. We designed and implemented an Export Schema Tool for participating countries to define the data that they are willing to share with other countries/agencies (i.e., to define
export schemas). We also designed and implemented a Schema Integration Tool for correlating (mapping) and integrating the data entities and attributes specified in different natural languages and stored in different databases. The exported schemas were used to generate a Global Search Form. Data mapping information and the integrated schema were used by a Distributed Query Processor to query and retrieve data from heterogeneous database sources. Our system was integrated with a language translation system developed at the Carnegie Mellon University (Pittsburgh, Pennsylvania) and a conversational interface system developed at the University of Colorado (Boulder, Colorado) to achieve international collaboration. Interoperation among all of these system components was achieved through a Web Services infrastructure.

We also demonstrated the use of events, triggers, and rules to enforce government policies and security constraints; and to facilitate event filtering and notification (in a sample scenario called the Watch-List scenario). Contributions of this work are design and development of the Export Schema Tool and the Schema Integration Tool; and integration of these tools with an enhanced Distributed Query Processor, a language translation system, a conversational interface system, and an Event-Trigger-Rule Server.
1.1 Background and Motivation

Countries all over the world are facing global problems such as drug trafficking, immigration and border control, disease detection and control, global education, terrorism. These problems can be solved through information sharing and close communication, coordination, and collaboration among government agencies in various countries. There is an urgent need for developing and integrating advanced information technologies to enable government agencies within a country (as well as across national boundaries) to share information and to work together.

The Transnational Digital Government (TDG) project is a research project funded by the National Science Foundation (NSF) of the United States. It aims to develop and apply advanced information technologies to address global or regional problems. Under this project, researchers from seven universities (Carnegie Mellon University, University of Belize, University of Colorado, University of Florida, North Carolina State University, University of Massachusetts and Pontificia Universidad Católica Madre y Maestra of the Dominican Republic) and experts from agencies in three countries (the Organization of American States (OAS) of the United States, the National Drug Abuse Control Council of Belize’s Ministry of Health, and the National Drug Council of the Dominican Republic) are developing information technologies to enable information sharing, integration, and coordination among agencies of the collaborating countries. The developed information
technologies will enable transnational resource sharing and inter-government and inter-organizational collaboration over virtual collaboration grids (Figure 1-1).

Figure 1-1. Virtual collaboration grids

To build the transnational prototype system, the participants teamed up with two small countries (Belize and the Dominican Republic), and jointly identified immigration and border control as the transnational problems to tackle. The idea was to share information across countries and agencies, for tracking movements of people entering and leaving these countries. Thus the goal of the initial system is to allow government agencies of collaborating countries to

- Enter and share immigration information (arrival and departure information of travelers)
- Integrate the shared information to generate a global view of the distributed data, to facilitate querying
- Access distributed data to identify suspicious individuals
• Support and coordinate inter-government and inter-organizational activities by secured data access, event notification, and policy enforcement

• Deliver useful information to the right people and organizations, at the right time, using different modes of communication

Information technologies being developed by the researchers in this project include a conversational interface system, a language translation system, a collaborative information management system, Internet portals and services, and a network support for collaboration grids. These technologies can be used to solve many other transnational problems similar to the immigration and border control problems. The Research and Development focus of our study was the collaborative information management system.

1.2 Challenges and Approach Taken

Solving the complex problems in the transnational setting presents many new technological challenges [1] as described below.

1. **Data heterogeneity.** Data gathered by the agencies at the ports of entry in both Belize and the Dominican Republic is stored in different formats, structures, and schemas. An integrated, global schema (Section 4.1.1) is needed to give users a uniform view of the distributed data. For this, we designed and implemented a system for data sharing by the two countries, and provided techniques for data mediation and integration. The distributed query processing system is developed for accessing this shared data. The global schema is presented in the different natural languages used by users (i.e., English for Belize users and Spanish for users in the Dominican Republic).

2. **Language heterogeneity.** The collaborating countries (Belize and Dominican Republic) use different natural languages. The Schema Integration tool is needed to specify the equivalence relationships between the data entities and attributes used in the heterogeneous databases of these countries. The language translation system is needed to translate sentences recorded in the comment field of the port-of-entry forms. For this, we integrated our system with the language translation system (Section 2.8) being developed at the Carnegie Mellon University.

3. **Heterogeneity in government policies, and security and privacy rules.** Each country may have its own policies, regulations, constraints, and rules regarding what information can be accessed by whom; and when and how information can be used. These policies and regulations may change with time. We provide a system to define and execute such rules. For instance, a country may have a rule that a tourist
official, while querying for some visitor’s arrival/departure information through the arrival/departure form, will have access to the visitor’s tourism data or arrival data, but will not have access to the departure data. To achieve this functionality, we use the Knowledge Web Server [2], developed at the Database Systems Research and Development Center, University of Florida. The Knowledge Web Server provides advanced event-filtering and rule-processing capabilities; and tools and software components for defining and processing events, triggers, and rules (Section 6.1.1).

4. **Difficulties in inter-agency and inter-government communication and coordination.** Communication and coordination are vital among collaborating countries. Collaborating countries can inform others of important events (e.g., the outbreak of a disease, or a terrorist’s movements), by automatically sending notifications and delivering relevant information on the occurrence of important events. To achieve this, we provide tools and mechanisms for supporting event publication, subscription, filtering and notification, and for performing event and rule-based triggering of operations and processes.

5. **Heterogeneity in working environments and computing platforms.** Government agents may have varying access to some of the computing facilities; or access to the Internet may be unreliable, or missing. Our system provides different means of communication and notification for such users (e.g., communication by emails and short messages via cell phones). Different government agencies worldwide use dissimilar hardware, software, operating systems, database-management systems, and application systems to perform their functions. There is a need for a common, standard-based infrastructure for accessing and interoperating these resources over a wide-area network like the Internet. Our system uses the Web Services model (Section 6.4) to achieve software resource sharing and interoperation of heterogeneous application systems. The Simple Object Access Protocol (SOAP) (Section 6.5) was used to invoke the Web Services. These Web Services can be accessed over the Internet via HTTP.

To show the transnational scenarios decided by the participants of this project and to show the approaches taken to solve the challenges explained above, we developed a Transnational Digital Government (TDG) prototype system at the University of Florida. Design and implementation of this prototype system was the purpose of our study. The system comprises

- Tool for participating countries/agencies to specify those and only those data that they are willing to share with others (i.e., for defining export schemas)
- Tool for integrating and correlating the exported information
- Distributed query-processing system for accessing the shared data
• Knowledge Web Server comprising an event server, an event-trigger-rule server, and knowledge profile manager.

  All of these components were developed at the University of Florida. They were integrated with a language translation system developed at the Carnegie Mellon University, and a conversational interface system developed at the University of Colorado. A Web Services infrastructure was jointly implemented by the collaborating universities to achieve the interoperability of these system components.

To test and demonstrate the developed technologies, the project’s initial focus was on the information-sharing and process-coordination problems related to border control against illegal immigration and drug trafficking. Our system (executed in Belize and the Dominican Republic) focuses on connecting border stations between these two countries, but the technologies can be used in other problem domains to enhance international cooperation.

**Limitations of the former prototype system:** The Transnational Information Sharing and Event Notification System [3] was developed only for processing distributed port-of-entry and exit data. It cannot be extended to other categories. It was built on a fixed set of database schemas, and was not built to handle join queries and queries that contain aggregate functions. The system can only handle a single user request at a time (i.e., queries issued by multiple users are not processed concurrently, but instead in a sequential order). Thus, there was a need to make this system more robust and extensible so that

• Multiple authorized users can query the system concurrently
• Code change will not be necessary in case there is a change in the export schema defined by a participating country or agency
• Same system can be used for information sharing in other problem domains such as agriculture inspection and protection, disease control, and homeland security

To overcome the limitations of the initial system, we have developed an Export Schema Tool (Section 2.3), which facilitates any agency in a country to define an export schema in any application category that the agency is willing to share with others. This tool is replicated and installed at the sites of all the participating countries, and the user interface of this tool allows users to select the natural languages that they desire for communicating with the tool. There is also a need for a tool to define new application categories for which schemas can be exported by participating countries or agencies. The exported schemas can then be integrated at a host site to generate a global schema, for querying purposes. To meet this need, we have developed a Schema Integration Tool (Section 2.4), which is installed at the host site. A person who knows the languages of the participating countries can log-in to this tool and perform data mappings (i.e., to specify the equivalence relationships) between data entities and attributes given in the exported schemas of an application category, as a way to integrate them and to generate the global schema.

The Distributed Query Processor (DQP) (Section 2.7) has been enhanced to use this global, integrated schema and also to process join and aggregate queries on the distributed, heterogeneous databases. It has been further enhanced to use the language translation system developed at the Carnegie Mellon University to mediate the language heterogeneities and display the results of the issued query in the user’s own natural language. Another extension added to the DQP was its integration with the conversational interface developed at the University of Colorado. The queries issued by the conversational interface are processed by the Distributed Query Processor, and the
query results are sent back to the conversational interface. The interoperability between all these system components is achieved through the Web Services infrastructure.

We also integrated our Distributed Query Processing System with the Knowledge Web Server, to demonstrate the enforcement of policies and regulations using events, triggers, and rules. The Watch-list scenario (Section 2.10) was added to create an authorized list of the immigration agents by some supervisor and to mark selected agents as ‘under suspicion’. When a traveler enters the country, the port-of-entry event occurs and a rule is triggered to check if the traveler is in the watch-list. If so, a notification (email and/or cell phone) is sent to the subscribers of the event. Along with this, an alert message is shown to only those agents not ‘under suspicion’, to warn them that the traveler is in the watch-list. An agent who is under suspicion of collaborating with the traveler will not get the alert message, but the notification of the traveler’s arrival will be sent to all relevant agencies.

1.3 Thesis Organization

This section describes the organization of the thesis in the following chapters. In Chapter 2, we explain the overall architecture of the TDG prototype system and briefly describe the functions of its system components. In Chapter 3, we explain the Export Schema Tool, developed for exporting the shared information to the host site and the Schema Integration Tool, developed for mapping the exported schemas to generate a Global Search Form. In Chapter 4, we describe the enhanced Distributed Query Processing system and its various components. DQP uses the Global Search Form for accessing the shared information. In this chapter we also describe the Watch-list scenario, which makes use of the Knowledge Web Server. Chapter 5 provides the implementation details of the system. In Chapter 6, we describe the existing technologies used to
implement the system. In Chapter 7, we give the performance evaluation of our system, and in Chapter 8, we give a conclusion of this work and propose some problems for future work.
CHAPTER 2
OVERALL SYSTEM ARCHITECTURE

This chapter provides an overview of the system architecture of our prototype system. The various sections describe the components, which include the tool for exporting and integrating the schemas, the distributed query processor, the watch-list scenario, and the integration with the language translation system and the conversational interface system. In this chapter, we shall also discuss the participating sites, the databases used, the Short Message Service Center, the ETR Server, and the Event Server.

![Overall system architecture of the prototype system](image)

Figure 2-1. Overall system architecture of the prototype system
The overall system architecture is shown in Figure 2-1. This system prototype is developed for processing distributed immigration data (i.e., port of entry and exit data), but it can be extended and used in other application domains such as disease control, agriculture security, etc. The various components of the system are described below.

2.1 Participating Sites

There are three participating sites in this prototype.

- Host Site
- Belize Site
- Dominican Republic Site

The Host Site has a collaboration portal and provides the facility for generating the global schema, event registration, and subscription. The two participating sites, one in Belize and the other in the Dominican Republic (DR) represent the agencies in the participating countries. The developed software components are extensible and they can accommodate a larger number of participating countries and agencies. The users of this system are authorized users at the host site and the sites of the participating countries. They include agents at the border stations and government agencies related to immigration.

2.2 Databases

Figure 2-1 shows a local database system at each of the participating countries’ sites. Each country may have databases from different vendors, and also the structures and schemas of these database systems may be different. Our system provides a tool to export the sharable entities and attributes of these local databases. The export schemas are integrated to produce a global schema, which represents the view of the distributed data as seen by the “global users” of the prototype system. Here, global users are those who have the right to query for distributed data. A global user can thus issue a query
against the generated global schema. The query once issued will be sent to the local
databases of the participating countries. It will then be processed by the local database
systems to extract relevant data from the local databases, and return the retrieved data to
the user.

2.3 Export Schema Tool

The Export Schema Tool is deployed at the local site of each participating country
as shown in Figure 2-1. This tool allows an agency of a participating country to define
those and only those data entities and attributes that it is willing to share with others. The
data defined in an export schema can thus be queried by legitimate users through a
Global Search Form, which will be explained in Section 4.1.4.

2.4 Schema Integration Tool and the Generation of a Global Schema

This tool, installed at the host site, is used to establish the semantic and language
equivalence relationships between data entities and attributes defined in the exported
schemas. It allows an authorized user at the host site (an IT personnel who knows
different languages of the participating countries) to establish mappings between two sets
of entities and attributes, which are exported by the participating nations and defined in
different natural languages. The result of this data mapping process is a set of data
mapping tables and a global, integrated schema. They are stored as global schema files
(Section 4.1.1) at the host site and sent to the participating countries’ sites by invoking
their Web services.

2.5 Event Server

The event server handles event registration, event notification, and also
communicates with the local ETR server, to activate rules triggered by those events. We
use the Event Server in our system in the Watch-list scenario to identify suspicious individuals and send event notifications to the subscribers of this event.

2.6 Event Trigger Rule Server (ETR)

The ETR server (Section 6.1.3) handles the installation and processing of rules at each site. Whenever the ETR Server receives an event notification from the Event Server, it identifies the proper triggers and rules to be executed.

2.7 Distributed Query Processor

This module is also deployed at the local site of each participating country. It includes a Global Search Form, which is dynamically generated using the global schema files. The Global Search Form includes entities and attributes that are the union of all the entities and attributes shared by the participating countries. If a participating country makes changes to the shared entities and attributes, those changes will be reflected in this form. A new country can easily become a part of the Global Search Form by sharing its database entities and attributes without involving any changes to the underlying code. Authorized users of the participating countries use this form to issue queries to access data stored in the local databases of these countries. The queries can be simple queries, join queries, or queries that contain aggregate functions like max, min, sum, average, and count.

2.8 Integration with the Language Translation System

The language translation system is developed at the Carnegie Mellon University. The integration of the language translation system with our system is required since the participating countries may use different natural languages. In that case, there will be a need to translate some of the data into the language of the logged-in user. For example, in the prototype system, Belize uses English language whereas the Dominican Republic
uses Spanish. There are several instances in our system where we invoke the language translation system through a Web Service interface that it provides.

2.9 Integration with the Conversational Interface System

The conversational interface system is developed at the University of Colorado. This system demonstrates the use of natural language to query the global database, and display the result to the user. The natural language query is translated into a query that is processed by the Distributed Query Processor to retrieve data stored in the database systems of the participating countries. The query is sent to the Distributed Query Processor in an XML format, and the retrieved data is also sent back to the conversational interface in a predefined XML format. The communication between these two system components is achieved through Web Services.

2.10 Authorization of Agents in the Watch-List Scenario

This module is installed at the local sites of each of the participating countries. The purpose of this component is to allow a supervisor to authorize the agents at the border stations, to use the system and also to mark some agents as ‘under suspicion’ of collaborating with some people in a watch-list. In this scenario, if some suspicious individual enters the country, the watch-list database will be checked to see if the traveler is present in the watch-list. If he/she is in the watch-list, then a warning (alert message) will be shown to only those agents, which are not ‘under suspicion’. Event notification will also be sent to all the subscribers of this watch-list event (e.g., security agencies, military, and law enforcement organizations). Thus, an agent under suspicion will not receive the alert signal and will not know that the traveler will be watched by relevant agencies.
2.11 Short Message Service Center

The event notification can be sent to the subscribers of an event using emails and/or cell phone notifications based on the options the subscribers selected at the time of event subscription. The cell phone notification is routed through the Short Message Service Center (SMSC). During event notification, the event server looks up the cell phone numbers and the network of the subscribers, and then sends the message with SMTP to phone@messaging.cell-network.com. The SMSC routes this message to the users’ cell phones as an SMS message.

The communication between the various components of our system deployed at different sites is through the Internet. In most cases, the services of these software components are invoked using the Web Service technology.
CHAPTER 3
DETAILED DESIGN

This chapter presents the detailed design of the tools for exporting and integrating the database schemas (Figure 3-1). Section 3.1 explains the design of the Export Schema Tool. This tool includes an interface to define new export schemas and an interface to modify a schema that was already created. Section 3.2 describes the Schema Integration Tool used to correlate all the exported schemas and to generate the global schema.

Figure 3-1. Export schema and integrate exported schemas
The IT personnel of the participating countries will use the Export Schema interface to define the database entities and attributes stored in their local databases that they are willing to share. The ExportSchema Web Service, deployed at the host site will be invoked in order to save the exported schema at the host site. Once all the participating countries have exported their schemas of an application category to the host site, the exported schemas will be integrated using the Schema Integration Tool installed at the host site, to generate the global schema. The host site will invoke the WriteGlobalSchema Web Service at the local site of each participating country. This Web Service will send the generated global schema files, which includes the global, integrated schema and the data mapping tables to the local site. The global schema will be used to generate the Global Search Form.

### 3.1 Export Schema Tool

The Export Schema Tool is used by the participating nations to specify their local database schema entities and attributes to be shared with other nations. It consists of the following interfaces: a form to select the display language so that the user can view the interfaces in his/her natural language, a form to select an application category for which the schema is to be exported, a form to add new data entities, a form to add attributes that are to be exported, and an interface to modify an existing schema. The steps followed by the user while exporting the schema are shown in Figure 3-2.

1. Authorized user logs in to the Export Schema Tool
2. User selects the language for displaying the user interface in that language. The choice of languages is restricted to the languages used by the participating countries
3. User enters the application category for which he/she wants to export the schema. For example, the user can enter a category name such as immigration, or agriculture
4. Next, the user can add new data entities, delete unwanted data entities, or skip this step. All the data entities, whose database attributes are to be exported should be added in this step.

5. Next the user gets to view the attributes, if any, that have already been added for exporting. The user can add more attributes by selecting the Add New Attributes (Figure 3-3) option or the user can delete an already added attribute, which is not to be exported.

6. Once the user is done with adding all the attributes to the form for exporting, he/she can select the Export Schema option to export the schema.

Given below is the sequence of steps executed by the user while adding new attributes for exporting (Figure 3-3)

1. On the add attributes page, the user selects the data entity to which the attribute belongs.

2. User will enter the attribute name for the new attribute.

3. User selects the type of the attribute (String, Integer, Boolean, or Date).

4. Next, the user will select the display type for the attribute (text box, radio button, or list box). This information is used later to generate the Global Search Form.

5. If the user wants to insert the new attribute before a particular attribute on the Export Schema page, he/she can enter that particular attribute’s name in the “Insert Before Attribute” text box.

6. If the display type of the new attribute is radio button or list box, user has to give the default values for this attribute, which will be displayed on the Global Search Form.

7. As a result, the new attribute can be appended at the end of the list of attributes that were already added, or it can be inserted before a particular attribute.
Figure 3-2. The export schema flow chart
Figure 3-3. The add attributes to export schema flow chart
Table 3-1 and Table 3-2 show two example schemas that are exported from the Dominican Republic and Belize site, respectively. Each exported schema includes the local database attribute names, the database relation to which the attribute belongs, the attribute types, and the display types of the attributes. Table 3-3 shows the components of the integrated global schema, which includes the pairs of attributes that are mapped from the two sites and the internal names assigned to each of these pairs.
Table 3-1. Dominican Republic’s sample exported schema

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Relation Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>docviajenumero</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Numerocedula</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Apellidos</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Nombres</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Sexo</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Fechallegada</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Puertoembarque</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Fechapartida</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Partidanumerovuelo</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Puertoembarque</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Fechapartida</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Lugarnacimiento</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Ciudadnacimiento</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Nacionalidad</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Ocupacion</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Estadocivil</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Calle</td>
<td>SALIDA</td>
</tr>
<tr>
<td>No</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Ciudadparaje</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Provinciaestado</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Pais</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Numerovuelo</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Motivoviaje</td>
<td>SALIDA</td>
</tr>
<tr>
<td>Comentariogeneral</td>
<td>SALIDA</td>
</tr>
<tr>
<td>id_pais</td>
<td>PAIS</td>
</tr>
<tr>
<td>nombre_pais</td>
<td>PAIS</td>
</tr>
</tbody>
</table>

Attribute Type: String, Display Type: Textbox
Table 3-2. Belize’s sample exported schema

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Relation Name</th>
<th>Attribute Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passportnum</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Passportdate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Passportstate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Passportcountry</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Lname</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Fname</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Middlei</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Gender</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Entrydate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Portofembcity</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Portofembcountry</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Departuredate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Portofdisembcity</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Portofdisembcountry</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Birthdate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Birthcountry</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Nationality</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Occupation</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrstreet</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrnumber</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrcity</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrstate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrcountry</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Paddrzip</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Vehiclenumber</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Baddrstreet</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Baddrnumber</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Baddrcity</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Baddrstate</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Intendedstaylength</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Purposeoftrip</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Visitnum</td>
<td>MAIN</td>
<td>Integer</td>
</tr>
<tr>
<td>Comments</td>
<td>MAIN</td>
<td>String</td>
</tr>
<tr>
<td>Passportnum</td>
<td>TOUR</td>
<td>String</td>
</tr>
<tr>
<td>Visitedbefore</td>
<td>TOUR</td>
<td>String</td>
</tr>
<tr>
<td>Lodging</td>
<td>TOUR</td>
<td>String</td>
</tr>
<tr>
<td>Interests</td>
<td>TOUR</td>
<td>String</td>
</tr>
</tbody>
</table>

Display Type: Textbox
3.2 Schema Integration Tool to Generate Global Schema

This module is installed at the host site. It is used to establish the semantic and language equivalence relationships between the exported data entities and attributes. In this process, pairs of data entities and attributes displayed in different natural languages will be mapped (correlated) to generate the global schema. This mapping is required to mediate the schematic and semantic heterogeneities that exist between the databases of the participating countries. During attribute mapping, one internal name is given to each set of mapped attributes and they together are added to the global schema at the host site. Internal attribute names are neutral representations of attribute names used in different databases. When the integrated global schema is saved, the global schema files and the data mapping files are also sent to the local sites of each of the participating countries. These files are used to generate the Global Search Form in the language used by the user.

This module includes the following form interfaces: a form to add new application categories, a form to map the exported attributes defined by different countries, and a form to assign internal name to a pair of mapped attributes and to add default values for some attributes. The sequence of steps executed during the integration of the exported schema attributes is as follows (Figure 3-4).

1. An authorized user (IT personnel), who knows all the languages of the participating countries logs in to the system at the host site to map the schemas.

2. The user can view a list of the available application categories.

3. He/she can add more category names and descriptions, so that the participating countries can export the schemas for this new category also.

4. The user can select a category, to view the schemas that have already been exported for that category.

5. The user will select a correlated pair of attributes from the exported schemas (those attributes, which he/she wants to map). He/she has to provide an internal name to
the pair of mapped attributes and then add it to the global schema. If an attribute is not present in one of the countries’ local database, then the user has to provide a name for that attribute in the countries’ natural language so that it can be displayed in the Global Search Form to be generated for the users of that country.

6. The user will repeat the above steps to map all the attributes that are to be exported and finally save the global schema.
Show available categories

Add new category

Enter category name & description

Y

Select a category

Show the schemas exported for selected category

Map a pair of attributes from Belize and DR

Add the attribute to global schema

Map more attributes?

Y

N

Save global schema

Figure 3-4. Map Exported Attributes
<table>
<thead>
<tr>
<th>Attribute Name in Belize</th>
<th>Attribute Name in DR</th>
<th>Internal Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passportnum</td>
<td>docviajenumero</td>
<td>passportno</td>
</tr>
<tr>
<td>Passportdate</td>
<td>fechademision</td>
<td>dateofissue</td>
</tr>
<tr>
<td>Passportstate</td>
<td>lugar-de-emision-estado</td>
<td>placeofissue-state</td>
</tr>
<tr>
<td>Passportcountry</td>
<td>lugar-de-emision-pais</td>
<td>placeofissue-country</td>
</tr>
<tr>
<td>Idno</td>
<td>Numerocedula</td>
<td>idno</td>
</tr>
<tr>
<td>Lname</td>
<td>Apellidos</td>
<td>Lastname</td>
</tr>
<tr>
<td>Fname</td>
<td>Nombres</td>
<td>firstname</td>
</tr>
<tr>
<td>Middlei</td>
<td>segundo-inicial</td>
<td>mi</td>
</tr>
<tr>
<td>Gender</td>
<td>Sexo</td>
<td>sex</td>
</tr>
<tr>
<td>Entrydate</td>
<td>Fechalambrada</td>
<td>dateofentry</td>
</tr>
<tr>
<td>Portofembarkationcode</td>
<td>Puertoembarque</td>
<td>portofembarkationcode</td>
</tr>
<tr>
<td>Portofembcity</td>
<td>puertoembarque-ciudad</td>
<td>portofembarkationcity</td>
</tr>
<tr>
<td>Portofembcountry</td>
<td>puertoembarque-pais</td>
<td>portofembarkationcountry</td>
</tr>
<tr>
<td>Departuredate</td>
<td>Fechapartida</td>
<td>dateofdeparture</td>
</tr>
<tr>
<td>Portofdisembarkationcode</td>
<td>puertodesembarque</td>
<td>portofdisembarkationcode</td>
</tr>
<tr>
<td>Portofdisembcity</td>
<td>puertodesembarque-ciudad</td>
<td>portofdisembarkationcity</td>
</tr>
<tr>
<td>Portofdisembcountry</td>
<td>puertodesembarque-ciudad</td>
<td>portofdisembarkationcountry</td>
</tr>
<tr>
<td>Birthdate</td>
<td>Fechanacimiento</td>
<td>dateofbirth</td>
</tr>
<tr>
<td>Birthcountry</td>
<td>paisnacimiento</td>
<td>placeofbirth</td>
</tr>
<tr>
<td>Nationality</td>
<td>Nacionalidad</td>
<td>nationality</td>
</tr>
<tr>
<td>Occupation</td>
<td>Ocupacion</td>
<td>occupation</td>
</tr>
<tr>
<td>Maritalstatus</td>
<td>Estadocivil</td>
<td>Maritalstatus</td>
</tr>
<tr>
<td>Paddrstreet</td>
<td>Calle</td>
<td>permanentaddress-street</td>
</tr>
<tr>
<td>Paddrnumber</td>
<td>No</td>
<td>permanentaddress-number</td>
</tr>
<tr>
<td>Paddrcity</td>
<td>Ciudadparaje</td>
<td>permanentaddress-city</td>
</tr>
<tr>
<td>Paddrstate</td>
<td>Provinciaestado</td>
<td>permanentaddress-state</td>
</tr>
<tr>
<td>Paddrcountry</td>
<td>Pais</td>
<td>permanentaddress-country</td>
</tr>
<tr>
<td>Vehiclenumber</td>
<td>Numerovuelo</td>
<td>airline-vehicle-vesselno</td>
</tr>
<tr>
<td>Baddrstreet</td>
<td>Direccion-destinada-calle</td>
<td>intendedaddress-street</td>
</tr>
<tr>
<td>Baddrnumber</td>
<td>Direccion-destinada-no</td>
<td>intendedaddress-number</td>
</tr>
<tr>
<td>Baddrcity</td>
<td>Direccion-destinada-ciudad</td>
<td>intendedaddress-city</td>
</tr>
<tr>
<td>Baddrstate</td>
<td>Direccion-destinada-estado</td>
<td>intendedaddress-state</td>
</tr>
<tr>
<td>Purposeoftrip</td>
<td>Motivoviaje</td>
<td>purpose-of-trip</td>
</tr>
<tr>
<td>Visitnum</td>
<td>visite el número</td>
<td>visitno</td>
</tr>
<tr>
<td>Comments</td>
<td>comentariogeneral</td>
<td>comments</td>
</tr>
<tr>
<td>Passportnum</td>
<td>docviagenumero</td>
<td>passportnumber</td>
</tr>
<tr>
<td>Visitedbefore</td>
<td>visitadoantes</td>
<td>visitedbefore</td>
</tr>
<tr>
<td>Lodging</td>
<td>accomodation-destinado</td>
<td>intended-accomodation</td>
</tr>
<tr>
<td>Interests</td>
<td>intereses-especiales</td>
<td>special-interests</td>
</tr>
</tbody>
</table>
This chapter describes the architectural design and functionality of the Distributed Query Processor (Figure 4-1) and the Watch-list scenario. The components of the Distributed Query Processor are: the Global Query Processing component (GQP) described in Section 4.1 and the Local Query Processing component (LQP) explained in Section 4.2. Section 4.3 explains the watch-list module.

### 4.1 Global Query Processor (GQP)

GQP makes use of the global schema files, country information, and user profile information to generate a Global Search Form in the natural language used by the user. The global schema files include form_info.txt, and mapCountryname.txt. The country information is stored in country_info1.txt file. The user profile information is stored in tomcat-users.xml file, and it is used for authentication and authorization of the logged-in user. We use Tomcat’s User Authentication facility to authenticate the users [4] [5].

#### 4.1.1 Global Schema Files

These files are generated after mapping the exported schemas from different countries at the host site. As explained in Section 3.2, the global schema files are saved at the local site of each participating country by invoking the WriteGlobalSchema Web Service. The format and use of each of the global schema files is explained below.

**The form_info.txt file:** This file stores information like the internal name for an attribute, the display type of the attribute, the number of default values associated with
that attribute, and the country codes of all those countries, which contain this attribute in their local databases. This is one of the files used to generate the Global Search Form.

Figure 4-1. Architecture of the Distributed Query Processor

The mapCountryname.txt file: A separate mapCountryname.txt file is generated for each of the participating countries. For example, the file sent to the Belize site when the “Generate Global Schema” button is clicked by the user will be named mapBelize.txt. This file includes information like the internal attribute name, the corresponding name for
the attribute in the country’s local database, the database relation to which the attribute belongs, the data type of the attribute, and the number of default values associated with the attribute. The actual default values will also be stored in the <VALUE> tags. If the attribute is not present in the local database of a country, then that field is replaced by the attribute name in the country’s local language and the database relation name will be set to “none”. This file is used for mapping the global (internal) attribute names to the countries’ local database attribute names, when the query is sent to the local site, and vice versa when the query results are generated and returned. This file together with the form_info.txt file is used to generate the Global Search Form that is displayed in the user’s natural language.

4.1.2 Country Information

The file country_info1.txt contains the Web Service URL of all the participating nations along with the specific method name of the service, which needs to be invoked in order to access the Local Query Processing component. The query issued by the user is sent in XML format to the local site by accessing this Web Service of the LQP.

4.1.3 Tomcat Authentication and User Profile Information

To set up tomcat user authentication, we did the following

1. Created a conf/users/tomcat-users.xml that has entries as shown below

   <tomcat-users>
   <role rolename="roleA_super"/>
   <role rolename="Belize"/>
   <user username="superbe" password="superbe" roles="Belize, roleA_super"/>
   </tomcat-users>

2. Inserted the following in the webapps/QP/WEB-INF/web.xml file

   Similar web.xml should be included for each of the applications deployed under tomcat.

   <security-constraint>
   <web-resource-collection>
       <web-resource-name>Query Form</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
  <role-name>Belize</role-name>
</auth-constraint>
</security-constraint>

<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/error.jsp</form-error-page>
  </form-login-config>
</login-config>

<security-role>
  <role-name>Belize</role-name>
</security-role>

The authorized roles and users are added in tomcat-users.xml file. The forms, which require login user authentication, are included as security constraints in the web.xml file for the application. The various user roles and their access privileges used by our system are shown in Table 4-1 below. For example, the role Super has access to all the database attributes, whereas role Police has access to only arrest information in the database.

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege to access following information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super</td>
<td>ALL</td>
</tr>
<tr>
<td>Police</td>
<td>Arrest information</td>
</tr>
<tr>
<td>Immigration</td>
<td>Immigration information</td>
</tr>
<tr>
<td>User1</td>
<td>Arrival-related immigration information</td>
</tr>
<tr>
<td>User2</td>
<td>Departure-related immigration information</td>
</tr>
<tr>
<td>Tourism</td>
<td>Tourism information</td>
</tr>
</tbody>
</table>
The participating countries can collaborate and decide on a global roles list and corresponding access privileges for the roles on the local database entities and attributes of those countries.

Figure 4-2. Global search form at the Belize site

4.1.4 Global Search Form

The Global Search Form (Figure 4-2) is generated automatically based on the integrated global schema, which is a union of the shared entities and attributes from all the participating countries. The Global Search Form is a Java Server Page (JSP) and is displayed in the natural language of the logged-in user. The user profile information, which includes the nationality of the user, is used to decide the display language for this form. Users can issue queries to the local databases of the participating nations using this form. Our system displays the global form in English for Belize users and in Spanish for
users in the Dominican Republic. The Global Search Form includes the following form fields.

- The list of all the participating countries to which users can issue queries and options to select them
- The list of all the attributes, which are included in the global schema and the data entities to which they belong. User can select any of these attributes for displaying in the query result
- User has the option to issue queries containing aggregate function ‘count’ that displays the sum total of all the records in the query result. The other aggregate functions that the query can include are max, min, avg, and sum for integer type attributes, and only max and min functions for the attribute type, date
- He/she can also specify the condition clauses (search criteria) for the query

The generated query is in a disjunctive normal form and can contain up to three ORed expressions entered into the three columns of Figure 4-2. Each of the OR expressions are a conjunction of the condition parameters entered in each column in the query form. The sequence of steps executed by the user when he invokes the Global Search Form is

1. The user has to login to the system, and then select the countries he/she wishes to query
2. Next he/she has to select the type of the query, e.g., simple query that displays the records or queries containing aggregate functions. He/she also has to select the attributes to be displayed in the query result
3. The user can enter the values for the condition clause of the query
4. If the user selects one or more aggregate type of queries, he/she is not allowed to select the other attributes on the form that are not part of the aggregate function, else it results in a SQL Exception at the database level

When the user submits the Global Search Form, the following actions take place.

- All the roles associated with the logged-in user and the countries whose databases the user wants to query are identified
- The type of the query is also identified for example, simple query that displays the actual records or query with an aggregate function that displays the result of the functions like sum, max, min, avg, and count on certain attributes in the database
An XML query document is created for each country that is queried. It is a sub-query that includes only those attributes selected by the user and are present in the country's local database. Thus, the query issued by the user is converted to an XML document. For example, belizequery_xmlfile1.xml (Figure 5-7) represents the sub-query sent to the Belize site. It contains the user's role information, the country name to which the query is issued, the query result attributes selected by the user and the condition clause part of the query. The attribute names in the XML query are the internal attribute names from the global schema, which are mapped to the local attribute names before the query is actually processed at the local site.

The sub-queries are sent to the local sites by invoking a Web Service method of those countries whose databases are being queried. Thus the Global Query Processor acts as a Web Service client of the Local Query Processor, which is deployed as a Web Service at the local sites. The Web Service URL and the method name for each participating country are stored in the country_info1.txt file (Section 4.1.2).

After a sub-query is processed at a local site, the query result is returned to the Global Search Form again in the form of an XML string. The XML query result is stored at the query issuing site as a pre-defined XML document called IndividualResult.xml (Figure 5-8). It consists of the name of the country that has sent the query result and the actual query result, which includes the attributes that were selected by the user and their values. At the local site, the local result attributes are mapped to the internal attribute names before sending the result to the query-issuing site. At the issuer’s site, the XML result document is traversed using the DOM parser and the result is extracted from it. The attributes of the returned result are again mapped to the local attribute names before being displayed to the user. This mapping uses the data mapping files.

If the query issued by the user contains an aggregate function, then the query results from the two countries have to be combined before displaying the results to the user. For example, if user issues the query "Retrieve the last date when a person of US nationality entered the country". This query will be processed independently at the local sites of the two countries, and the “max” aggregate function will be applied on the “date of entry” attribute of the two local databases. The two sites will send their local query results to the query-issuing site where the “max” function is again applied on the two returned results to get the global maximum value, which is then displayed to the user.

4.2 Local Query Processor (LQP)

The Local Query Processing component consists of a Web Service interface for the LQP and a wrapper, which interacts with the Event Server, the ETR Server, the Translation System, and the local DBMS of the country. The LQP component receives
the XML query document sent by the GQP of the query-issuing site. It translates the received query into an SQL query for processing by the local DBMS.

4.2.1 Web Service Interface for LQP

The LQP component is deployed as a Web Service by each of the participating sites. The Web Service method accepts the XML query document sent by the Global Query Processing component as a string argument and returns the query results again as an XML format string to the query-issuing site.

4.2.2 The Wrapper Associated with the LQP

The wrapper performs the following functions.

- It maps the internal attribute names present in the sub-query issued by the GQP to the local database attribute names of the site to which this LQP component belongs. The mapping uses the mapCountryname.txt file

- The extractor module of the wrapper processes the XML query document using the Document Object Model (DOM) parser to extract information such as the role information of the person who has issued the global query, the attributes being queried along with the attributes included in the condition clause of the sub-query. It also extracts the table names (relation names) to which all the attributes that are present in the sub-query belong

- Next the access controller module of the wrapper checks if the user who issues the query has the access right to all the attributes in the query. It connects to the local Event Server of the site and posts the qaccess event. The qaccess event triggers a rule to check the access right of the user on the query attributes

- The translator module deals with the table lookup translation and language translation

  The translation of internal attribute names to local attribute names and vice versa, and the translation of attribute values use the table lookup method. The mapping tables are implemented as hash tables

  There are certain attribute values, which cannot be resolved using the lookup method, and for those we need to invoke the CMU’s language translation system. For example, the officer at the port-of-entry station can enter comments about the travelers entering the country, in his/her own language. These comments get stored in the local database of that country. If the user who issues the query is interested in searching those records, which contain
specific words in the ‘comments’ field, then the issuer will enter the search criteria in his/her own natural language. But when the query is actually executed on the local database by the LQP component, that search criteria needs to be translated into the language of the local site before the query is processed. For example, the immigration officer may want to see a list of all the people who appeared nervous when they entered the country. So he/she will enter the word 'nervous' in his/her native language into the comment field. Once this query reaches the local site, it needs to be translated into the language used by the local site before the query is actually processed. Also after getting the query results, the values for the ‘comments’ attribute need to be translated back to the natural language of the query issuer before being displayed to the user. For these translations, the language translation system developed at the Carnegie Mellon University is invoked.

- The **query processor** module is responsible for generating a query in SQL format using the attributes extracted from the XML query string. The query processor then connects to the local database and issues the SQL query to the local database system. In the process of generating the SQL query, the translator is used to convert the internal query attribute names to local attribute names of the local database, before the query is sent to the local DBMS, and to convert the local attribute names of the returned query result to the corresponding internal (or neutral) attribute names. The machine translator may also be invoked to translate some query values. The query result is then sent to the wrapper, which converts the result into an XML format and passes it back to the Web Service method of the LQP. The Web Service will send the result to the Web Service client (i.e., the GQP of the issuing site).

### 4.3 Watch-List Scenario

This module will be deployed at the border stations of the participating nations. The Watch-list scenario depicts three goals.

- Allows the supervisor to mark some of the agents posted at the border stations as *under suspicion*, if they are suspected of collaborating with some people in a watch-list by admitting them into the country
- The system checks if the traveler entering the country is in the watch-list, and displays an alert message to the agent on duty, if he/she is not *under suspicion*
- The Watch-list scenario is also one of the examples for the Event-Trigger-Rule system, and is used to send event notifications to the subscribers of the event

The main components of this module are the authorized agents’ database, the watch-list database, the local database, which stores the traveler’s arrival/departure information, and the Event-Trigger-Rule system. The event depicted in the Watch-list scenario is the PEntry event. This event gets posted when the agent at the border station
fills the arrival information in the Arrival form, for a traveler who wants to enter the
country. The event triggers a rule called the WatchListCheck Rule. This rule checks if the
traveler is in the watch-list by consulting the watch-list database.

Figure 4-3. The Watch-List scenario flow chart
This module consists of the following form interfaces: a form to mark some of the border agents as ‘under suspicion’ and a form to fill the arrival information for a traveler.

The sequence of steps executed during the Watch-list scenario is as follows (Figure 4-3).

1. The supervisor will log in and create and edit an authorized agents’ list. Agents who are under suspicion of being corrupted agents are marked

2. Agent at one of the border stations logs into the system and fills the arrival form for a traveler who enters the country

3. The system checks if the traveler is in the watch-list by consulting the watch-list database

4a. If the traveler is in the watch-list, then ETR sends event notification to all the subscribers of this event. Event notification is sent via email and/or cell phone

4b. If not, then the system will insert traveler’s record into the database

5. Check if the agent is under suspicion by consulting the database of authorized agents

6a. If yes, no alert message will be posted. The agent will allow the traveler to enter the country and the traveler’s arrival information will be inserted into the database

6b. If not, the alert message, which says that the traveler is in the watch-list, will be displayed to the agent

7a. The agent who gets the alert message can either allow the traveler to enter the country and the traveler’s data is inserted into the database

7b. Or, the agent rejects the traveler from entering the country
CHAPTER 5
IMPLEMENTATION DETAILS

In this chapter, we describe the implementation details of the main components in our system. Section 5.1 describes the Export Schema Tool, Section 5.2 describes the Schema Integration Tool, Section 5.3 describes the Distributed Query Processor, and Section 5.4 gives the implementation details for the Watch-list scenario.

5.1 Export Schema Tool

The files, which are used in the implementation of the Export Schema functionality, are described below.

The language.jsp page: This JSP provides an interface, where user can select the language for displaying all the pages in the Export Schema module. The languages currently supported by our system are Spanish and English since these are the languages used by the participating countries of the Dominican Republic and Belize respectively, in the prototype system. Once the user selects the language of his choice on this page, the corresponding language.txt file will be used for displaying the user interfaces in the selected language. For example, if user selects Spanish as the language, the system will use the Spanish.txt file to display the forms in Spanish. In short, our system can easily support a new language interface by including the language.txt file for the new language. For example, to provide the support for French language, we need to add a new file to the system named “French.txt” with the required translations.

The select_category.jsp page: This interface allows the user to select the application category for which he/she wants to export the schema.
**The relations.jsp page:** Based on the category name selected by the user on the select_category page, this interface will display the data entities that are already available for this category. The data entities added, for a particular category, are appended to the relations.txt file under the “category” folder in the DQP directory of the local site. If there are no data entities that exist for the selected category, then a new relations.txt file will be created, and then the data entity name will be included in that file. As a result, this form interface allows user to add new data entities.

**The export_schema.jsp page:** This JSP (Figure 5-2) displays all the attributes that have been previously added for exporting under the selected category by reading the exported_attributes.xml file of the category. This XML file is read by traversing it using the DOM parser. If there are no attributes previously added for exporting, then a new exported_attributes.xml file is created in the same directory where the relations.txt file is present. The exported_attributes.xml file will be updated every time a new attribute is added or deleted. A sample exported_attributes.xml file is shown in Figure 5-1.

```
<COUNTRY>Belize</COUNTRY>
<RELATION>
  <RELNAME>MAIN</RELNAME>
  <ATTRIBUTE>
    <NAME>passportnum</NAME>
    <RELATTR>MAIN</RELATTR>
    <TYPE>string</TYPE>
    <DISPLAY>textbox</DISPLAY>
  </ATTRIBUTE>
</RELATION>
```

Figure 5-1. Format of the exported_attributes.xml file

In Figure 5-1, the country name in the <COUNTRY> tag indicates that this file is exported from a Belize site. MAIN is the data entity name to which the attributes belong. The attributes belonging to different data entities are stored separately in the XML file.
The `<ATTRIBUTE>` tag holds all the information related to the attribute that is to be exported. As shown in the figure, the attribute name is passportnum. It belongs to the data entity (relation) MAIN, the data type of the attribute is string, and the display type is textbox.

**Figure 5-2. Export schema page**

**The add_attribute.jsp page:** This interface is used by the user to add new attributes to the export schema. The newly added attribute is written to the file exported_attributes.xml. After the user has added all the attributes that he/she wants to export, he/she submits the export_schema.jsp page, and then the ExpSchema.java file invokes ‘sendSchema’ method of the ExportSchema Web Service that will store the exported local schema at the host site. The exported schema is saved as an XML file under the selected category folder at the host site. Here the file ExpSchema.java acts as a
client for the ExportSchema Web Service. Whenever any site exports its local schema for this category, it will be stored as a new XML file under the category folder at the host site.

Figure 5-3. Add attribute to export schema page

5.2 Integrate the Local Schemas to Generate the Global Schema

The files that are used to implement the schema mapping and integration of the exported schemas are described below.

The global_categories.jsp page: An authorized user can select the category for which he/she wants to generate the global schema after logging-in to the global_categories.jsp page. The user can also add new categories to the list of existing categories so that the local sites can also export their schemas for these newly added categories. All the categories and the category descriptions are stored in the categories.txt
file. When the user selects a category, he/she gets to see the schemas exported by all the local sites under that category.

The **published_schemas.jsp page**: This page displays all the schemas currently exported for the selected category. The exported schemas of each site (country) are displayed as columns of attributes. The metadata like the data entity of the attribute, its data type, display type, and the default values if any associated with the attribute are shown when the user clicks on a particular attribute. All this metadata information is extracted from the schema files stored under the OAS folder.

![Published Schemas](image.png)

**Figure 5-4. Published schemas page**

The **add_global_attributes.jsp page**: This JSP page is invoked when the user selects the related pair of attributes from each local site and clicks on the 'Add to Global Schema' button on the published_schemas.jsp page. The user will be shown the attributes
that he/she wants to map in order to include the attribute in the global schema. For example, if the user has selected ‘passportnum’ from Belize’s exported schema and ‘docviajenumero’ from the Dominican Republic’s exported schema, then he/she is shown the country name and the corresponding attribute name from the country’s exported schema. If an attribute that is to be added to the global schema is not present in one of the countries’ exported schemas, then user is asked to enter the attribute name in that countries' local language. The user is also asked to enter an internal name for the mapped attribute. If there are any default values for an attribute in one of the countries' exported schemas, then the user has to enter the names for those values in the other country's local language.

Figure 5-5. Mapping page for exported attributes
When the user adds an attribute to the global schema, the global schema files are created at the host site and they get updated every time a new attribute is added. The generated global schema files are form_info.txt, parserARRIVAL-DEPARTURE.met, parserARRIVAL-DEPARTUREsp.met and mapCountryname.txt. A separate mapCountryname.txt file is created for each of the participating countries.

The dummy.jsp page: Once the user has added all the attributes from the exported schemas to the global schema for the selected category, he can save the schema. This JSP contains the code for generating and storing the global schema files at the host site and sending them to the local sites, where the files are stored for use by the Global Search Form. The files are sent to the local site by invoking the GlobalSchema Web Service at the local site. The WriteGlobalSchema.java file present at the host site acts as a Web Service client of the GlobalSchema Web service. The writeSchema method of this Web Service will write the global schema files at the local sites.

5.3 Distributed Query Processor

5.3.1 Global Query Processing Component (GQPC)

The CreateGlobalForm.jsp/ CreateGlobalFormsp.jsp page: This JSP will generate the Global Search Form using the global schema files, mapCountryname.txt and form_info.txt (Section 4.1.1). The page is displayed to the user in English or Spanish language based on the user profile information stored in the tomcat-users.xml file. Figure 4-2 shows a sample Global Search Form at the Belize site. This form is used to issue the queries in English language.

The query.jsp page: This file is responsible for query generation based on the attributes selected by the user on the Global Search Form and the display of query results. The query is generated in XML format (Figure 5-6) and it is created using the internal
names of the attributes and attribute values. A separate sub-query is generated for each of the local sites that are queried by the user. The sub-query includes only those attributes that are present in the local database of the country to which the sub-query is being sent. The user’s ROLE information and the country name being queried are also included in the XML query file. This JSP acts as an AxisClient for the Web Service present at the Local Query Processing component of the local site (LQP). The Web Service information is stored in the country_info1.txt (Section 4.1.2) file.

The JSP parses the Individual-Result.xml (Figure 5-7) file of each country to which a sub-query was issued. The internal attribute names in the sub-query result are mapped to the local attribute names on the Global Search Form before displaying the result to the user. The results retrieved from the individual sites for the sub-queries involving aggregate functions such as count, sum, max, min, and average will be combined on this form.

```xml
<?xml version='1.0'?>
<!DOCTYPE QUERY [<!ELEMENT QUERY (ROLE+, ATTRIBUTE*, OR*)>
<!ELEMENT ROLE (#PCDATA)><!ELEMENT ATTRIBUTE (#PCDATA)>
<!ELEMENT OR (AND*)>
<!ELEMENT AND (ATTRNAME, OPERATOR, ATTRVAL)>
<!ELEMENT ATTRNAME (#PCDATA)> <!ELEMENT OPERATOR (#PCDATA)>
<!ELEMENT ATTRVAL (#PCDATA)> ]>
<QUERY>
   <ROLE>roleA_super</ROLE>
   <ROLE>Belize</ROLE>
   <ATTRIBUTE>avg(visitno)</ATTRIBUTE>
   <OR><AND>
      <ATTRNAME>purpose-of-trip</ATTRNAME>
      <OPERATOR>=</OPERATOR>
      <ATTRVAL>BUSINESS</ATTRVAL>
   </AND></OR>
</QUERY>
```

Figure 5-6. Sample XML query
<xml version='1.0' encoding='ISO-8859-1'>
<INDIVIDUAL_DETAILS>
  <COUNTRY>BELIZE</COUNTRY>
  <INDIVIDUAL_DETAIL>
    <ATTRIBUTE>
      <NAME>avg(visitno)</NAME>
      <VALUE>3.2500</VALUE>
    </ATTRIBUTE>
  </INDIVIDUAL_DETAIL>
</INDIVIDUAL_DETAILS>

Figure 5-7. Sample XML result

Figure 5-8. Query results displayed at the Belize site

5.3.2 Local Query Processing Component (LQPC)

The LQPC is deployed as a Web Service at each of the participating countries. This Web Service will be invoked by the query.jsp file, which acts as the Web Service client.
The classes that comprise the LQP components at the Belize site are described below. Similar files are present at the local site of each participating country.

**The belizeserver.java file:** The belizeinterface method in this file is invoked by the Web Service client of the LQPC. The method in turn invokes the belizespecxml file by passing the XML query string to it.

**The belizespecxml.java file:** The checkAccess method in this file checks the access control of the user to each attribute in the XML query string. The buildSQL method will generate the SQL query from the XML query string, and the createXML method will execute the SQL query on the local database. The createXML method also converts the SQL query results to XML format before sending them back to the GQP (Axisclient).

The translation system explained in Section 4.2.2 is invoked to translate the values, if any, in the comments field before issuing the query to the database. The value in the comments field will be translated only if it is not in the local language of the user, which is decided by the role of the logged-in user. The translator is also invoked on the values retrieved by the comments attribute in the query result subject to the constraint that the result is not in the local language of the user.

**The belizemapattr.java file:** This file is used to generate the mappings for the attribute values in the database. The map tables are implemented as hash tables. These mappings will be used while building the SQL query from the XML query string and also while converting the result into the XML format.

### 5.4 Watch-List Scenario

This scenario depicts the events that occur when a person arrives in a country and the agent at the border station fills a form to record the person’s arrival information. It is
also an example, which describes the Event registration, subscription, and notification capability of the ETR Server. The files used in the implementation of this scenario are explained below.

The **AuthorizePpl.jsp page**: This JSP (Figure 5-9) is used to create a list of authorized agents and to mark some of these agents as ‘under suspicion’. The user of this system is a supervisor, and the list of the agents not ‘under suspicion’ is stored in the “AuthorizedPpl.txt” file.

![Figure 5-9. Create authorized agents’ list page](image)

**The ArrivalDR.jsp page**: An agent at the border station will fill this form (Figure 5-10) when a person enters the country. When this form is submitted, the ArrivalServlet is invoked and the port-of-entry (PEntry) event gets posted. The event will trigger the WatchListCheckRule to check if the person (traveler) entering the country is
in the watch-list. This checking is done by comparing the traveler’s last name, first name, and nationality with the values stored in the watch-list. If the traveler is in the watch-list and the agent is not ‘under suspicion’, then he/she is shown the alert message, which says “Traveler is in the watch list”. If the agent decides to allow the traveler to enter the country, then the traveler’s record is inserted into the database. If the agent is marked as ‘under suspicion’, he/she will not be shown the alert message by the system and the traveler will be directly allowed to enter the country, and his/her record will be inserted into the database.

Figure 5-10. Arrival form at the port of entry in a Dominican Republic border station

5.5 Translation System

The translation by table lookup was explained in Section 4.2.2. This type of translation is used to convert the internal query attribute names into local database
attribute names so that it can be executed on the local database. We also use the language translation system for natural language translations.

The translator.java file: This file is used to invoke the machine translation system developed at the Carnegie Mellon University (CMU) and acts as the Web Service client of the service interface provided by CMU. The translate method of the Web Service is invoked for getting the translated result. This method takes two parameters, the actual string to be translated and the language into which the string should be translated. For example, translate (“nervous”, “sp”) means that the word “nervous” should be translated into the Spanish language. The Web Service sends the translated result back to the client.

5.6 Conversational Interface System

The conversational interface developed at the University of Colorado uses our Distributed Query Processor for issuing queries to the databases of the participating countries and retrieving the query results. The conversational interface translates the natural language query into an XML query string similar to the one generated by the Global Search Form. This interface then invokes the GlobalQuery Web Service deployed at the host site. The Web Service will invoke the Global Query Processing component of all the sites, which are being queried by the conversational interface. Once the query reaches the GQPC, it is processed in the same way as the query generated by the Global Search Form. The XML query results are sent back by the GlobalQuery Web Service to the Web Service client (i.e., the conversational interface system).
CHAPTER 6
TECHNOLOGIES USED

We have installed our system on the Windows NT platform and it is implemented using JDK 1.4.2_04. The Web server used is Tomcat 4.1.18 and Apache Axis 1.0 toolkit is used to define, deploy, and invoke Web Services. The database management system used as the local database management system is MySQL. In this chapter, we describe the technologies used to implement the following tools and functionalities of the prototype system: tool for defining export schemas, tool for integrating exported schemas, distributed query processing system for accessing data from distributed, heterogeneous databases, and event-trigger-rule processing system for implementing a Watch-list scenario. Section 6.1 describes the ETR technology and its various components. Section 6.2 describes Java related technologies, and Section 6.3 discusses the XML related technologies. Section 6.4 and Section 6.5 explain the Web Services infrastructure and SOAP, respectively. We explain how Apache Axis toolkit is used for deploying the Web Services in Section 6.6. Our prototype system uses the MySQL database management system, which is described in Section 6.7.

6.1 The Events, Triggers, and Rules (ETR) Technology

The ETR technology is a part of the Knowledge Web Server [2]. The ETR’s event-trigger-rule service is used in the implementation of the Watch-list scenario (Section 4.3). The Knowledge Web Server extends the capability of the current Web servers. Each Knowledge Web Server has a replica of an Event Manager, an ETR Server, and a Knowledge Profile Manager, which are the additional components installed on each Web
server. Replicas of the Event Manager exchange events and transfer data associated with the events (i.e., event data) between them.

6.1.1 Events, Triggers, and Rules

Any item of interest can be modeled as an event. For instance, entering a traveler’s information into the arrival form at a border station by an immigration agent can be considered as an event. A rule consists of a condition clause, an action clause, and optionally, an alternative action clause. When an event is posted, if the condition clause associated with the rule evaluates to True, the action clause is executed. Otherwise, some alternative action is performed. Triggers are used to associate events with rules. A trigger specifies that, upon the occurrence of any one of a number of events, an optional expression of occurred events (i.e., an event history) should be evaluated. If the event history is evaluated to True or if the optional expression is not given, a single rule or a structure of rules should be processed. The trigger specification maps event attributes to rule parameters so that run-time event data can be passed to a rule(s) for its (their) evaluation.

6.1.2 Event Manager

Legitimate clients can subscribe to published events. They can also specify event filters, which contain some data conditions associated with events. If the data conditions match with the data associated with the occurrence of an event, subscribers want to be notified. The Event Manager is responsible for sending and receiving events and for performing event filtering before sending out the event data, to those subscribers whose filtering conditions are satisfied. When the Event Manager receives an event from a remote web server, it passes the event and event data to the local ETR Server to initiate the processing of triggers and rules.
6.1.3 The ETR Server

The ETR Server receives events and event data from the local Event Manager, and performs trigger and rule processing. On receiving an event, the ETR Server identifies the trigger related to the event, processes the event history, and executes the rule(s).

6.1.4 Knowledge Profile Manager (KPM)

Each user of the transnational information system has a knowledge profile that is maintained by the Knowledge Profile Manager [6]. A knowledge profile includes the events that the user has subscribed to, the event filters associated with the subscribed events, and the triggers and rules that have been defined on the subscribed events. A Meta-data Manager within the KPM provides persistence for storing the user knowledge profiles.

6.1.5 Persistent Object Manager (POM)

POM [7] consists of two main components.

- Object-Relational mapping engine
- XML-Relational mapping engine

The Object-Relational mapping engine provides a persistent storage facility and a high level interface in the form of APIs for programs to store, retrieve, update, and delete objects without having to know the internal data structures of the objects. The XML-Relational mapping engine provides the persistence capability and a filtering mechanism to the Event Server. POM is implemented on top of an Object-Relational database system called Cloudscape.
6.2 The Java Technologies

6.2.1 Java Servlet Technology

Servlets [8] are the Java platform technology of choice for extending and enhancing Web servers. Building a Web page on the fly is useful for a number of reasons.

- The Web page is based on data submitted by the user
- The data changes frequently
- The Web page uses information from corporate databases or other such sources

Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), servlets are server and platform-independent.

Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. They can also access a library of HTTP-specific calls and receive all the benefits of the Java language, including portability, performance, reusability, and crash protection.

6.2.2 Java Server Pages (JSP) Technology

JSP technology [9] enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. It separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.

JSP is an extension of the servlet technology created to support authoring of HTML and XML (Section 6.3.1) pages. It uses XML-like tags that encapsulate the logic that generates the content for the page. The application logic can reside in server-based
resources (such as JavaBeans component architecture) that the page accesses with these
tags. Any and all formatting (HTML or XML) tags are passed directly back to the
response page. By separating the page logic from its design and display, and supporting a
reusable component-based design, JSP technology makes it faster and easier than ever to
build Web-based applications.

6.2.3 Tomcat

Tomcat 4 [10] implements the Servlet 6.3 and Java Server Pages 1.2 specifications
from Java Software, and includes many additional features that make it a useful platform
for developing and deploying Web applications and Web Services.

6.3 The XML-Related Technologies

6.3.1 Extensible Markup Language (XML)

XML [11] is a simple, very flexible text format derived from SGML (ISO 8879).
Originally designed to meet the challenges of large-scale electronic publishing, XML is
also playing an increasingly important role in the exchange of a wide variety of data on
the Web and elsewhere.

- XML stands for EXtensible Markup Language
- XML was designed to describe data
- XML tags are not predefined. One must define his/her own tags
- XML uses Document Type Definition (DTD) or XML Schema to describe the data
- XML can be used to exchange data between incompatible systems
- XML can also be used to store data in files or in databases

6.3.2 Xerces: XML Parsers in Java

Xerces [12] provides world-class XML parsing and generation. It offers fully
validating parsers for Java, implementing the W3C XML and DOM (Level 1 and 2)
standards, as well as the de facto SAX (version 2) standard. The parsers are highly
modular and configurable. Initial support for XML Schema is also provided by Xerces.
6.4 Web Services

The World Wide Web is increasingly used for application-to-application communication. The programmatic interfaces made available are referred to as Web Services [13]. Web Services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

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 (specifically, WSDL). 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. Thus, Web Services are Web-based enterprise applications that use open, XML-based standards, and transport protocols to exchange data with calling clients.

6.5 Simple Object Access Protocol (SOAP)

SOAP [14] provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC. SOAP can potentially be used in combination with a variety of other protocols. SOAP consists of three parts.

- The SOAP envelope construct defines an overall framework for expressing what is in a message; who should deal with it, and whether it is optional or mandatory
• The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types

• The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses

6.6 Apache Axis

Axis [15] is essentially a SOAP engine: a framework for constructing SOAP processors such as clients, servers, gateways. The current version of Axis is written in Java and a C++ implementation of the client side of Axis is also being developed. Axis is not just a SOAP engine, it also includes

• A simple stand-alone server
• A server which plugs into servlet engines such as Tomcat
• Extensive support for the Web Service Description Language (WSDL)
• Emitter tooling that generates Java classes from WSDL
• Some sample programs, and
• A tool for monitoring TCP/IP packets

Axis is the third generation of Apache SOAP (which began at IBM as "SOAP4J").

In late 2000, the committers of Apache SOAP v2 began discussing how to make the engine much more flexible, configurable, and able to handle both SOAP and the upcoming XML Protocol specification from the W3C.

6.7 The MySQL Database

MySQL [16] has become the most popular open source database and the fastest growing database in the industry. This is based on its dedication to providing a less complicated solution suitable for widespread application deployment.

MySQL offers several key advantages.

• **Reliability and performance**: MySQL AB provides early versions of all its database server software to the community to allow for several months of "battle testing" by the open source community before it deems them ready for production use.
• **Ease of use and deployment:** MySQL's architecture makes it extremely fast and easy to customize. Its unique multi-storage engine architecture gives corporate customers the flexibility they need with a database management system unmatched in speed, compactness, stability, and ease of deployment.

• **Freedom from platform lock-in:** By providing ready access to source code, MySQL's approach ensures freedom, thereby preventing lock-in to a single company or platform.

• **Cross-platform support:** MySQL is available on more than twenty different platforms including major Linux distributions, Mac OS X, UNIX, and Microsoft Windows.
CHAPTER 7
PERFORMANCE EVALUATION

In this chapter we analyze the performance of our system. Section 7.1 gives an evaluation of the queries processed by the Distributed Query Processor, Section 7.2 presents the system performance during schema exportation and integration, and Section 7.3 describes the performance of the rule-processing and event notification in the Watch-List scenario.

7.1 Query Performance

7.1.1 Simple Queries

Here we give the time required to fetch the result of each of the sample queries from the local databases of the participating sites using our DQP system. The relations in the local database of Belize contain sixty records and those in the local database of the Dominican Republic contain seventy records.

1. Fetch the passport numbers and names of all the females who arrived in the country after Jan 1, 2003.
   - Number of sites to be queried: (2)
     This query fetches results from both Belize and Dominican Republic
   - Number of where clause conditions: (2)
     gender = ‘female’ and entry-date = ‘2003-01-01’
   - Execution time: 3 seconds

2. Fetch the passport numbers, names, nationality and purpose of trips of all the people whose nationality was USA or Belize and who had come for business.
   - Number of sites to be queried: (2)
     This query fetches results from both Belize and the Dominican Republic and is an example of a query in disjunctive normal form
   - Number of where clause conditions: (3)
     nationality = ‘USA’ and purpose of trip = ‘business’ or nationality = ‘Belize’ and purpose of trip = ‘business’
3. Get the passport number, name, date of entry, port of embarkation city and port of disembarkation city of all the people who entered the country after Jan 1, 2003 and whose port of embarkation city is Boston and port of disembarkation city is Belize City.

- Number of sites to be queried: (1)
  This query fetches results only from Belize database but it can be issued from Belize or Dominican Republic site
- Number of where clause conditions: (3)
  date of entry = ‘2003-01-01’, port of embarkation city = ‘Boston’ and port of disembarkation city = ‘Belize City’
- Execution time: 2 seconds

7.1.2 Aggregate and Join Queries

This section describes the times taken to process queries with aggregate functions and join queries.

1. Give the most recent date when a person of US nationality arrived in the country on business.

   - Number of sites to be queried: (2)
     This aggregate query fetches max (date of entry) from the local databases of Belize and Dominican Republic and then combines the results before displaying them to the user
   - Number of where clause conditions: (2)
     nationality = ‘USA’ and purpose of trip = ‘business’
   - Execution time: 1 second

2. Give the passport numbers and names of all the people who were staying in a guesthouse.

   - Number of sites to be queried: (1)
     This join query is issued only to the Belize database and it does a join of two database tables ‘MAIN’ and ‘TOUR’
   - Number of where clause conditions: (2)
     lodging = ‘guesthouse’ and tour.passportnum = ‘main.passportnum’
   - Execution time: 2 seconds
7.1.3 Queries Invoking the Translator

1. Give the passport numbers and names of all the males who appeared nervous.
   - Number of sites to be queried: (2)
     This query is issued to the Belize and Dominican Republic databases from the Belize site. Here CMU’s translator module is invoked to translate ‘nervous’ to Spanish language before processing the query at the Dominican Republic site and again to translate the results retrieved from the Dominican database into English before being displayed to the user.
   - Number of where clause conditions: (2)
     comments = ‘nervous’ and gender = “male”
   - Execution time: 4 seconds

7.2 Performance of Export Schema and Schema Integration Tools

7.2.1 Export Schema Tool

The system takes approximately one second to export a schema from a local site. The time taken involves the time to invoke the Web Service at the host site, where the schema will be exported and stored.

7.2.2. Schema Integration Tool

The tool takes approximately two seconds to integrate the exported schemas in order to generate the global schema. This includes the time to invoke the Web Service at each of the local sites and to save the global schema files at the local sites.

7.3 Performance of Rule Processing and Event Notification

The Watch-List scenario takes about 2 to 3 seconds to post the arrival event, trigger the WatchListCheck rule to find if the traveler entering the country exists in the watch-list, to check if the agent on duty is ‘under suspicion’, and then to show the alert message to the agent if traveler is in watch-list and agent is not ‘under suspicion’. Although, this scenario also involves sending the email and cell phone notifications to the subscribers of this event, the specified time does not include the time for event notification because it will depend on the number of event subscribers.
CHAPTER 8
CONCLUSIONS AND FUTURE WORK

This chapter summarizes the contents of this thesis, its contributions and discusses the scope of future work. Through the various chapters in this thesis, we have established and confirmed the need for a transnational information system. The developed prototype system is the product of integrating a number of component systems: the Export Schema Tool, the Schema Integration Tool, the Distributed Query Processor system, the Language Translation system, the Conversational Interface system, and the Event-Trigger-Rule Server. With these system components in place, we have provided the means for integrating and sharing information, querying the heterogeneous databases using a form-based interface or a conversational interface, applying translations where necessary, and enforcing rules and regulations. The heterogeneous system components are made interoperable by using the Web Service technology.

The main contribution of this thesis is the design and implementation of the tool for exporting the schemas from any of the participating countries, the tool for integrating these exported schemas to generate the global schema, the module to create the authorized agents list, the enhanced Distributed Query Processor which can accommodate schema changes made to the local databases, and handle queries that contain aggregate functions and join operations over database tables. The export schema component allows the countries to export their local database entities and attributes, and also to modify the exported schemas whenever there is a change to the local database schema. The modified exported schema can again be integrated with the schemas
exported by other countries to generate the global schema. The Distributed Query Processor is able to incorporate the new global schema without making any code changes. Our prototype system is built for immigration and remote border control applications. However, it is designed and implemented in a general way that it can be used for other application domains such as agriculture inspection and protection, disease control, and can be used by a larger number of participating countries.

There are some interesting features, which can be added, and some issues that can be further investigated to extend the functionality of the transnational digital government system. The Distributed Query Processor presently handles the join operation over relations (data entities) belonging to a single site. It can be enhanced to handle the join operation over data entities stored in multiple sites. Secondly, event notification is presently done by uni-casting; i.e., the subscriber information for an event is stored at the event publisher’s knowledge web server. When an event occurs, the notification is sent to each subscriber one-at-a-time, which can be very time consuming. A more efficient approach is to distribute the subscribers’ information to the knowledge web servers of different countries or agencies to which the subscribers belong. When an event occurs, the Event Server at the event-posting site can use multi-casting to send notifications to the Event Servers of all the event subscriber sites. These Event Servers can then send notifications to their local subscribers in parallel.

We need to identify other transnational digital government problems that can be solved using the information technologies developed by the various research groups. The system should be expanded to cover multiple countries and languages, and means have to be established for field-testing and evaluation of the final system.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Manjiri Patil was born on August 11th, 1978, in Pune, Maharashtra, India. She received a Bachelor of Engineering degree in computer engineering (securing first class with honors), from the Maharashtra Institute of Technology, Pune, India, in May 2000. After graduation she worked with Satyam Computer Services Limited (Pune, India), as a Software Engineer.

In August 2002, she joined the University of Florida (Gainesville, Florida), to pursue Master of Science degree in computer and information science and engineering. She worked as a Teaching Assistant and a Research Assistant, during her studies at the University of Florida. Her research interests include databases and the Web Services technology.