JUNCTION-BASED LOCATION QUERY SYSTEM FOR RESTAURANTS

KANOKWAN RODDAM

TITIPORN NUNTAPRAMOTE

NAPAK VORATITIPONG

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE INTERNATIONAL PROGRAMS, FACULTY OF SCIENCE KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

Junction-Based Location Query System for Restaurants

Kanokwan Roddam

Titiporn Nuntapramote

Napak Voratitipong

A Special Project Submitted in Partial Fulfillment of the Requirements for

The Degree of Bachelor of Science

International Programs, Faculty of Science

King Mongkut's Institute of Technology Ladkrabang

2007

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

Special Project Title	Junction-Based Location Query System for Restaurants
Student	Kanokwan Roddam 47050358
	Titiporn Nuntapramote 47050360
	Napak Voratitipong 47050363
Degree	Bachelor's Degree of Science
Department	Mathematics and Computer Science, Faculty of Science
Program	Computer Science (International)
Advisor	Suntana Oudomying

ABSTRACT

We propose an application which ranks results of user's interest in term of proximity from his point of origin. Our local search application is based on the idea of semantic query and WordNet where an input (query) from a user implies the area surrounding it. In our implementation the system reckons the surrounding junctions to the input area by using adjacency matrix. Our application focuses on restaurants in Bangkok area where in practice we could create any layer of interest on this location base system. We, thus, also propose an XML schema to make it easy to enhance this web application.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Acknowledgement

The author would like to take this opportunity to express sincere thanks to teachers and persons who have given useful advice and full support in this research.

The author wishes to express our deep gratitude to Mr.Suntana Oudomying our advisor, for this valuable guidance, attention, and encouragement throughout this research.

We are greatly appreciated all the professors who have given invaluable knowledge while we are studying in the department of computer, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang.

We are also would like to give the special thanks to all of our friends at department of science who had been encouraged us.

Lastly, the authors would like to express our deepest appreciation to our dearest fathers, mothers, brothers and sisters for love, care, and encouragement. They are the most important in our life forever.

Kanokwan Roddam

Titiporn Nuntapramote

Napak Voratitipong

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

Table of Contents

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

List of Tables

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

List of Figures

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Chapter 1

Introduction

1.1 Rationale

Many web sites in Thailand nowadays offer services on information about locations. For example real estate websites allow users to query for a house in certain district or region. The result would base on an exact match of the value in certain attribute in a standalone database. Such results often do not meet the user expectation. E.g. to find a house for sale in Soi Ladprao 107 area, the user would prefer to have a result ranked relative to the distance from the area which should cover houses for sale in area next to Bangkapi, which Soi Ladpro 104 is belonged to, area as well instead of only the houses for sale in the area. In general, an exact match query would not adequate to give results where users expect in a location query.

Our system would serve as an infrastructure for any layer of interested class of places. For example, in our case, we would like to create a layer of restaurants which, if combines with our underlying information, would be able to list restaurants from a certain location. In order to obtain the underlying data, our methodology emphasizes on participation which means user would contribute the underlying information only from area where he / she thoroughly knows.

1.2 Objectives

Our system searches the information related to user specified location. In this case we use restaurants to be the data. The system will retrieve not only the restaurants in the location that user wants, but also restaurants around the vicinity locations. The system is developed to be a web application for user's convenience to access.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

1.3 Scope of Study

Objective in 1.2 has shown the ability of our system. However, it still has some limitation. First, in the webpage of our system, users are not allowed to type an area by themselves to avoid misspelling problem. Second, the restaurants are searched from the system come from our database; it does not cover all of the restaurants in the area. It depends on how soon the data up to date.

Before ending this part, one may argue that to find restaurants in an area is simple. Such site could switch to GPS where it maps its products onto the GPS coordination and let the user understand from what the GPS displays. Our argument is that not only such site need to switch its infrastructure, our system would complement the GPS in a situation where the user does not know where on GPS is the destined area of his. Therefore, we leave out integrating GPS from our scope. Rather, we will focus on integrating map to our system.

1.4 Research Methodology

To create our system, we have studied and learned some related articles. All related technology's background provides in chapter 2. Chapter 3, it shows how our system works from the beginning and how it shows the result. In chapter 4, we evaluate the system to show how well it works and show how to use it. The last part, chapter 5, we discuss about system's conclusion and recommendation. In the appendix, we provide brief explanation about how to install and configure all technologies we used.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Chapter 2

Background

2.1 Ontology as a specification mechanism

In both computer science and information science, ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. It is used to reason about the objects within that domain. Ontology is used in artificial intelligence, the semantic web, software engineering, biomedical informatics and information architecture as a form of knowledge representation about the world or some part of it.

The ontology contains a set of contexts which form the ontology itself. Within one ontology, every context should have a unique name. It is annotated with a term within the meta context for this ontology. As attributes, the ontology has a name (mandatory and unique), a contributor, an owner, a status and documentation (an arbitrary string in which the contributor or the owner can specify relevant information).

Ontology objects can also be expressed in XML format, using DTD. This representation in the popular XML data description language enables us to use the whole variety of tools available for XML, at this point; the ontology manager only provides support for storing ontology expressed in XML.

2.1.1 Extensible Markup Language (XML)

XML is a great technology that changes the way of collecting and using data. It can also merge the usage of database and document together. XML can already use many of the standards applied to HTML. However, using XML with most efficiency it is necessary to deal with an appropriate query language too.

XML (Extensible Markup Language) provides a foundation for creating documents and document systems. XML operates on two main levels: first, it provides syntax for document markup; and second, it

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

provides syntax for declaring the structures of documents. XML is clearly targeted at the Web, though it certainly has applications beyond it.

A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents. For example, the following bit of a document includes three elements: two elements with content, and one empty tag.

```
<FIGURE DESCRIPTION="Harvey"><IMAGE/><CAPTION>This is data!!</CAPTION></FIGURE>
```

The first start tag opens the FIGURE element, which has the attribute DESCRIPTION set to "Harvey", and contains an empty IMAGE element and the CAPTION element with its content. End tags close both the CAPTION and the FIGURE elements, producing a nested structure. These nested structures are fairly good at representing typical document and data structures and a very easy for computer programs to store and manipulate.

2.1.2 Document Type Definition (DTD)

In addition to providing syntax for document markup, XML provides syntax for specifying document structure; DTD and XML Schema. Document Type Definition (DTD) is primarily used for the expression of a schema via a set of declarations that conform to a particular markup syntax and that describe a class, or type, of SGML or XML documents, in terms of constraints on the structure of those documents. A DTD can be declared inline in your XML document, or as an external reference.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Internal DTD

```
<!--1-->  <?xml version="1.0" ?>
<!--2-->  <!DOCTYPE BookCatalogue [
<!--3-->          <!ELEMENT BookCatalogue (Book) *>
<!--4-->          <!ELEMENT Book(Title, Author,
<!--5-->  Date, ISBN, Publisher)>
<!--6-->          <!ELEMENT Title (#PCDATA)>
<!--7-->          <!ELEMENT Author (#PCDATA)>
<!--8-->          <!ELEMENT Date (#PCDATA)>
<!--9-->          <!ELEMENT ISBN (#PCDATA)>
<!--10-->         <!ELEMENT Publisher (#PCDATA)>
<!--11-->  ]>
<!--12-->  <BookCatalogue>
<!--13-->          <Book>
<!--14-->                    <Title>Introduction to
<!--15-->  XML</Title>
<!--16-->                    <Author>Alex
<!--17-->  Ferguson</Author>
<!--18-->                    <Date>1 July 2547 </Date>
<!--19-->                    <ISBN> 974-78910-4-
<!--20-->  2</ISBN>
<!--21-->
<!--22-->          <Publisher>Manchester</Publisher>
<!--23-->          </Book>
<!--24-->  </BookCatalogue>
```

Fig 2.1: **Internal DTD:** An XML document with a DTD.

The DTD is interpreted like this:

!ELEMENT Book (in line 4)

Defines the element "Book" as having five elements: "Title, Author, Date, ISBN, and Publisher"

!ELEMENT Title (in line 6)

Defines the "Title" element to be of the type "CDATA"

!ELEMENT Author (in line 7)

Defines the "Author" element to be of the type "CDATA"

External DTD

```
<!--1-->  <?xml version="1.0">
<!--2-->  <!DOCTYPE BookCatalogue SYSTEM
<!--3-->  "bookcatalogue.dtd">
<!--4-->  <BookCatalogue>
<!--5-->          <Book>
<!--6-->                    <Title>Introduction to
<!--7-->  XML</Title>
<!--8-->                    <Author>Alex
<!--9-->  Ferguson</Author>
<!--10-->                   <Date>1 July 2547 </Date>
<!--11-->                   <ISBN> 974-78910-4-2</ISBN>
<!--12-->
<!--13-->         <Publisher>Manchester</Publisher>
<!--14-->         </Book>
<!--15-->  </BookCatalogue>
```

Fig 2.2: **External DTD:** Same XML document with an external DTD.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
<!--1-->  <?xml version="1.0"?>
<!--2-->  <!ELEMENT BookCatalogue (Book) *>
<!--3-->          <!ELEMENT Book(Title, Author,
<!--4-->  Date, ISBN, Publisher)>
<!--5-->          <!ELEMENT Title (#PCDATA)>
<!--6-->          <!ELEMENT Author (#PCDATA)>
<!--7-->          <!ELEMENT Date (#PCDATA)>
<!--8-->          <!ELEMENT ISBN (#PCDATA)>
<!--9-->        <!ELEMENT Publisher (#PCDATA)>
```

Fig 2.3: A copy of the file "bookcatalogue.dtd" containing the DTD.

XML provides an application independent way of sharing data. With a DTD, independent groups of people can agree to use a standard DTD for interchanging data. Your application can use a standard DTD to verify that the data you receive from the outside world is valid. You can also use a DTD to verify your own data. XML documents themselves are also considerably more open than their binary counterparts. Anyone can parse a well-formed XML document, and validate it if a DTD is provided.

2.1.3 XML Schema

XML Schema published as a W3C Recommendation in May 2001, is used to define the legal elements of an XML document, just like a DTD. XML Schemas has support for data types and namespaces. We think that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. But the significantly more complicated than DTDs, it's not widely used. XML Schema does not implement most of the DTD ability to provide data elements to a document. While technically a comparative deficiency, it also does not have the problems that this ability can create as well, which makes its strength.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

```
<!--1-->  <?xml version="1.0"?>
<!--2-->  <xsd:schema xmlns="http://www.w3.org/2000/10/XMLSchema"
<!--3-->  targetNamespace="http://www.publishing.org"
<!--4-->  xmlns="http://www.publishing.org" elementFormDefault="qualified">
<!--5-->  <xsd:element name="BookCatalogue">
<!--6-->          <xsd:complexType>
<!--7-->          <xsd:sequence>
<!--8-->                    <xsd:element ref="Book" minOccurs="0"
<!--9-->  maxOccurs="unbounded" />
<!--10-->         </xsd:sequence>
<!--11-->         </xsd:complexType>
<!--12--> </xsd:element>
<!--13--> <xsd:element name="Book">
<!--14-->         <xsd:complexType>
<!--15-->         <xsd:sequence>
<!--16-->                   <xsd:element ref="Title" minOccurs="1" max
<!--17--> Occurs="1"/>
<!--18-->                   <xsd:element ref="Author" minOccurs="1" max
<!--19--> Occurs="1"/>
<!--20-->                   <xsd:element ref="Date" minOccurs="1" max
<!--21--> Occurs="1"/>
<!--22-->                   <xsd:element ref="ISBN" minOccurs="1" max
<!--23--> Occurs="1"/>
<!--24-->                   <xsd:element ref="Publisher" minOccurs="1" max
<!--25--> Occurs="1"/>
<!--26-->         </xsd:sequence>
<!--27-->         </xsd:complexType>
<!--28--> </xsd:element>
<!--.....-->          . . . . . .
<!--.....--> </xsd:schema>
<!--.....-->
```

Fig 2.4: **XML Schema:** The same structure but different type schema.

2.1.4 XQuery, XSLT alternative tools

Furthermore, at present XQuery is the latest query language for XML that is proposed by W3C, the organization that has the authority to settling many Internet standards. XQuery is being developed continuously and is expected to become the complete standard of query language for XML soon. However XQuery still lacks several features to manage data collecting in document forms. This is an analysis and comparison between 2 famous XML proposals related to query fields, XQuery and XSLT. XSLT is the standard for transforming XML document that can work with document very well.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.1.4.1 XQuery

XQuery is the language for querying XML data. The best way to explain XQuery is to say that XQuery is to XML what SQL is to database tables. XQuery is defined by the W3C and uses XPath expressions to extract XML data. XPath is used to navigate through elements and attributes in an XML document.Example: doc("books.xml")/ bookstore/ book .

The doc() function is used to open the "books.xml" file , /bookstore selects the bookstore element and /book selects all the book elements under the bookstore element.

Instruction of XQuery is FOR-LET-WHERE-RETURN

- For - variables that range over the results of XPath expressions. If more than 1 variable, Cartesian product of values taken by the variables.
- Let - allows complicated expressions to be assigned to variable names (for simplicity)
- Where - test on tuples given by the 'For' clause.
- Return - construction of results (in XML)

For example

```
for $x in doc("books.xml")/bookstore/book
 where $x/price>30
 return $x/title
```

- The **for** clause selects all book elements under the bookstore element into a variable called $x.
- The **where** clause selects only book elements with a price element with a value greater than 30.
- The **return** clause specifies what should be returned. Here it returns the title elements.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.1.4.2 Extensible Stylesheet Language Transformations (XSLT)

The XML Stylesheet Language (XSL) was originally designed for generating HTML from XML. XSLT is a general-purpose transformation language can translate XML to XML, and XML to HTML. XSLT transformations are expressed using rules called templates. Templates combine selection using XPath with construction of results.

For example

```
<xsl:template match="/">
    <html><body>
     <h1><xsl:value-of select="message"/></h1>
          </body></html>
</xsl:template>
```

- The <xsl:template match="/"> chooses the root
- The <html><body> <h1> is written to the output file
- The contents of message is written to the output file
- The </h1> </body></html> is written to the output file

The XML text document is read in and stored as a tree of nodes. The <xsl:template match="/"> template is used to select the entire tree. The rules within the template are applied to the matching nodes, thus changing the structure of the XML tree, if there are other templates, they must be called explicitly from the main template. Unmatched parts of the XML tree are not changed. After the template is applied, the tree is written out again as a text document.

The Difference between XQuery and XSLT, XSLT is written in a XSL file which is later connected to a XML file but XQuery gets the XML file dynamically. It allows many XML files to be included in a single query. The joint operations can be performed in XQuery whereas they cannot be performed in XSLT.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

2.2 WordNet

WordNet is a large lexical database of English. WordNet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets (synsets), each representing one underlying lexical concept. Different relations link the synonym sets. WordNet was developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller (Principal Investigator). WordNet is considered to be the most important resource available to researchers in computational linguistics, text analysis, and many related areas. Its design is inspired by current psycholinguistic and computational theories of human lexical memory.

Synsets are interlinked by means of conceptual-semantic and lexical relations: Hypernymy (is-a relation), Meronymy (has-part relation), Holonymy (part-of relation).This WordNet module (Word Net::QueryData) is a Perl interface to the WordNet database. It contains tens of thousands of words and numerous semantic relationships for each.

Unlike other dictionaries, WordNet does not include information about etymology, pronunciation and the forms of irregular verbs and contains only limited information about usage. No explicit distinction between proper and common nouns. It was too difficult to include this information.

Wordnet has been extended to a FrameNet. The FrameNet database is a lexical resource with unique characteristics that differentiate it from other resources such as commercially available dictionaries and thesauri as from the best-known on-line lexical resource (WordNet). WordNet and all ontology provide some sort of hierarchical relations between their nodes; likewise, FrameNet includes a network of relations between frames.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

With the idea of the WordNet, we could search and rank for relevant restaurants from a location of interest. If there is a query for area A restaurants, restaurants in area B is queried as well and the result is ranked lower than the result of restaurants in area A because the degree of relevancy is weaker that those of area A. We, again, use Perl to create multiple queries from the user query.

We apply adjacency matrix technique to find transitivity of area. This technique we use is a non-direction graph. The figure below is an example of a simple graph of non-direction adjacency matrix, a matrix with rows and columns labeled by graph vertices with a 1 (have adjacent vertices) or 0 (does not have any adjacent vertex). For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

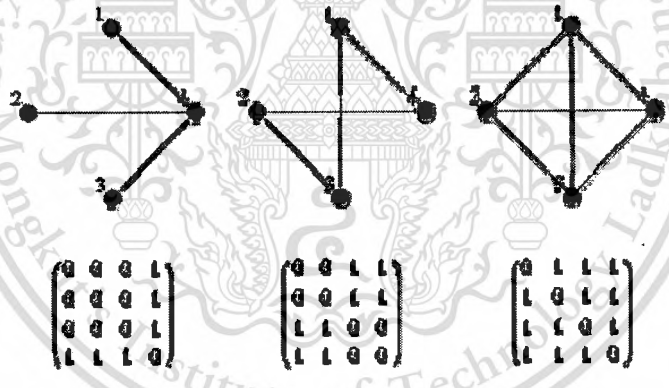


Fig 2.5: An example of adjacency matrix.

2.3 Practical Extraction and Report Language (PERL)

Perl is a language that is available free over the Web, and it's used for a variety of things, from writing CGI scripts to assisting administrators in maintaining their systems. Perl was created, and is still maintained, by Larry Wall. It's slower than C, but faster than a normal interpreted language. Because it's compiled when executed and then interpreted.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Perl is popular because of two reasons: First, most of what is being done on the Web happens with text, and is best done with a language that's designed for text processing. More importantly, Perl was appreciably better than the alternatives at the time when people needed something to use. C is complex and can produce security problems (especially with untrusted data).

There is another test programming languages for example; Python, and TCL. Python is a better than Perl, but some may not consider Python as good a scripting language as Perl. TCL (Tool Command Language) has a lot of good utilities like Expect, great integration with GUIs, TCL is simple, and is also ported to a lot of environments. TCL is also slow and has some problems as a language.

We use Perl to construct from previous page because Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

2.4 Web Server: Apache

By running Perl in the browser, we need an Apache to be our web server. Apache is an open source web server that runs on most commonly used platforms. A web server is the remote program that gives you an Internet page when you click a link in your browser. Apache has a modular design that provides a variety of services such as server-side scripting and it is the most widely available HTTP server on the Internet. It supports the PERL languages.

2.5 Database: My SQL

The My SQL database has become the world's most popular open source database because of its consistent fast performance, high reliability, ease of use and its freeware. It also becomes the database of

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

choice for a new generation of applications built on the LAMP stack (Linux, Apache, MySQL, PHP / Perl / Python).

The Table Joins are the key to building efficient databases and harnessing the essential power of MySQL. Without table joins, a complex relational database will be as slow as a standard flat file database. PerlScripts will teach you how to utilize table joins and greatly increase the speed of your database queries. PerlScripts has taught lead programmers of other programming firms how to utilize the true power of MySQL. For example, join Table

area Table

areaID	areaName
01	Thong Lo
02	Bangna
03	Ekkamai

alias Table

areaID	aliasName
01	Thong Lo
01	Sukhumvit 55
02	Bangna
03	Ekkamai
03	Sukhumvit 63

Example of an explicit inner join:

```
SELECT *
FROM   area, alias
WHERE  area.areaID = alias.areaID
```

Join Table result:

areaID	areaName	areaID	aliasName
01	Thong Lo	01	Thong Lo
01	Thong Lo	01	Sukhumvit 55
02	Bangna	02	Bangna
03	Ekkamai	03	Ekkamai
03	Ekkamai	03	Sukhumvit 63

As an example, the following query takes all the records from the area table and finds the matching record(s) in the alias table, based on the join predicate. The join predicate compares the values in the areaID column in both tables.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Chapter 3

Junction-Based Location Query System for Restaurants

3.1 System Overview

In the Junction-based location query system for restaurants, after a user enters a searching area, the system retrieves junctions in and around that area. Then the relation of junctions is constructed by applying the adjacency matrix technique. After that the system queries the restaurant information in each junction and ranks the result. The output will be represented in the xml file. Finally, we use HTML with java script to get and demonstrate the output data. This system adopts the idea of pipe and filter architecture.

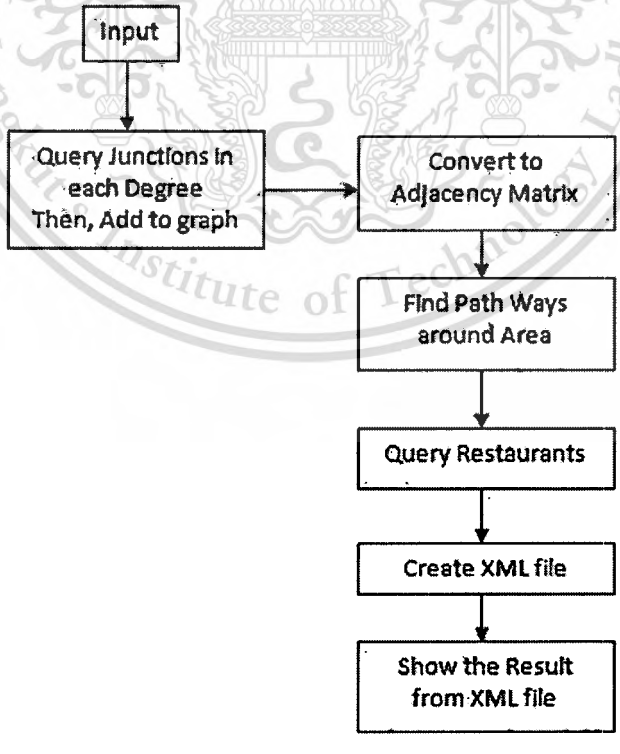


Figure 3.1: System Architecture

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

From the Figure 3.1, there are 3 variables of input from a user; area, restaurant type, and price per person. The area is used for retrieving restaurants. The restaurant type and the price per person are for only filtering restaurants.

The system uses the area to get junctions from database. It will query junctions in the area and 3 more degrees of junctions around the area. In our solution, instead of query from the table directly, we create views to help us. If we query directly from table 3 times (4 degrees: area plus 3 junction level), each time, we will get duplicate data itself and duplicate data of next degree, for example, in Figure 3.2 at 1st time, Junction_1 has 2 J2s and Junction_2 has J1 and J2 duplicate Junction_1. Not only the duplicate data, this way needs to store Junction_2 data of previous time in array, then use a loop getting a data from array and query in the next time.

Junction_1	Junction_2
J1	J2
J2	J1
J2	J3
J3	J2
J3	J4
...	...

Assume: J1 and J2 is in area A

1st time: Query junctions in area A
Jucntion_1: J1 J2 J2
Jucntion_2: J2 J1 J3

2nd time: Query junctions in Jucntion_2 of 1st time
Jucntion_1: J1 J2 J2 J3 J3
Jucntion_2: J2 J1 J3 J2 J4

Figure 3.2: Query junctions directly from the table

By using views, we can deal with duplicate data, and does not need to store data in array as well. In our case, we need 3 views represent 3 times of query. The Figure 3.3 shows the data will be the outcome of the views. Moreover, views can be used in other steps of the program.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

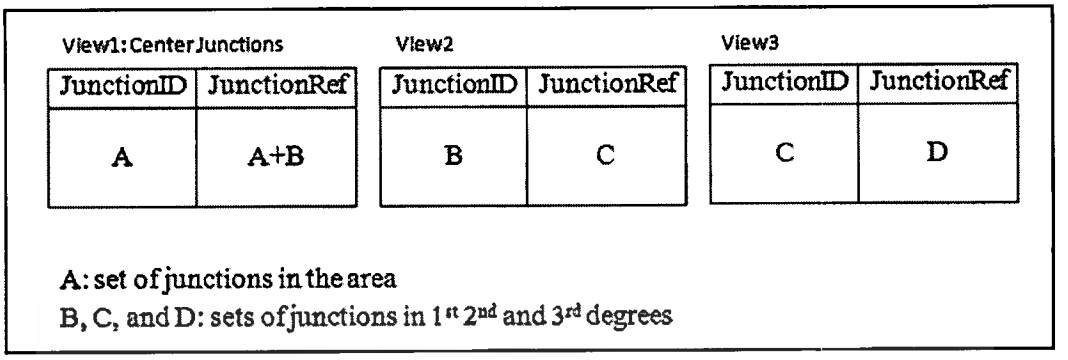


Figure 3.3: View Outcome

At the end of each time the system query junctions, the system add each pair of junctions into a graph. After finish querying and adding to graph, the system convert a graph to be an adjacency matrix graph, just for it can use the adjacent function.

Since we want the system rank restaurants in direction, finding route direction is very important. Again, we use views that have been created. The B in Figure 3.2 can be used to be the start point of each route. Then, we will find 2 more degrees.

After that, the system will query restaurants. The output of the program is in the XML file. We will explain more about XML file and how we show the result this in section 3.5.

For all processes in the Junction-based location query system for restaurants, it can be divided in 4 main parts as follows:

1. Analyze and Design database schemas

2. Creating Relationship Among Junctions

3. Querying and ranking

4. Demonstration of the result

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3.2 Analyze and Design Database Schemas

3.2.1 DTD Schema

In our map structure, we have 3 main DTD schemas; area, junction and street. The area schema is the highest level information in our system. On the other hand the street schema is the lowest level in the system. It refers to restaurants, buildings or other information in each street. The junction, where at least 2 streets intersect, is the middle level between area and street. Each junction is annotated with (sub) area and next to junctions. In the junction schema, a junction doesn't mean only an intersection, it means to every kind of junction including roundabout, traffic circle and a route way that refer to junction itself.

With this information, we can have a logical image of map of area of our interest – meaning we could draw this imaginary image to become a draft map. We also have a logical of area block which is surrounded by at least 3 junctions. We use Figure 3.4 to present the information of our all schema.

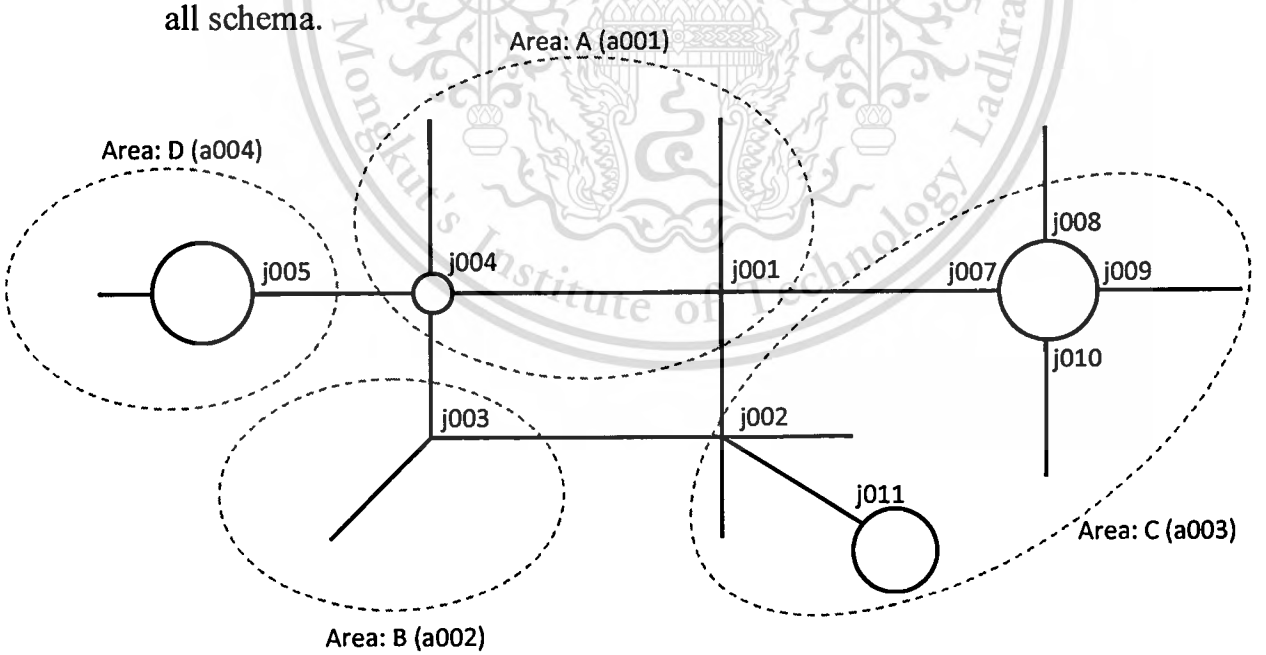


Figure 3.4: Physical map example.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Our area schema contains an ID of an area (*areaID*), information of area itself such as name (*areaName*), and IDs of junctions in each area. The *junctionRef* element refers to the junctions in the junction schema. The area schema looks like the Table 3.1.

Table 3.1: Area DTD schema

```
<!DOCTYPE areas [
  <!ELEMENT area (areaName, areaDescr,
areatojunction)>
  <!ATTLIST area areaID ID #REQUIRED>
  <!ELEMENT areaName (#PCDATA)>
  <!ELEMENT areaDescr (#PCDATA)>
  <!ELEMENT areatojunction (junctionID)*>
  <!ELEMENT junctionID (#PCDATA)>
]>
```

```
<area areaID = "a001">
      <areaName>A</areaName>
      <areaDescr></areaDescr>
      <areatojunction>
            <junctionRef>j001</junctionRef>
            <junctionRef>j004</junctionRef>
      </areatojunction>
</area>
```

The junction DTD schema is constructed for every case of junctions such as intersection, junction, roundabout, and etc. The schema looks like a code in Table 3.2.

Table 3.2: Junction DTD schema

```
<!DOCTYPE junctions [
  <!ELEMENT junction (junctionName, junctionDescr, junctiontojunction, junctiontostreet)>
  <!ATTLIST junction junctionID ID #REQUIRED>
  <!ELEMENT junctionName (#PCDATA)>
  <!ELEMENT junctionDescr (#PCDATA)>
  <!ELEMENT junctiontojunction (junctionRef)+>
  <!ELEMENT junctionRef (#PCDATA)>
  <!ELEMENT junctiontostreet (streetID)+>
  <!ELEMENT streetID (#PCDATA)>
]>
```

The junction schema contains junction IDs (*junctionID*), junction names (*junctionName*), junction descriptions (*junctionDescr*), *junctiontojunction* and *junctiontostreet*. The *junctiontojunction* has sub element named *junctionRef* refers to junctions around itself. The *junctiontostreet* has sub element named *streetID* refers to streets that connect directly to this junction in the street schema.

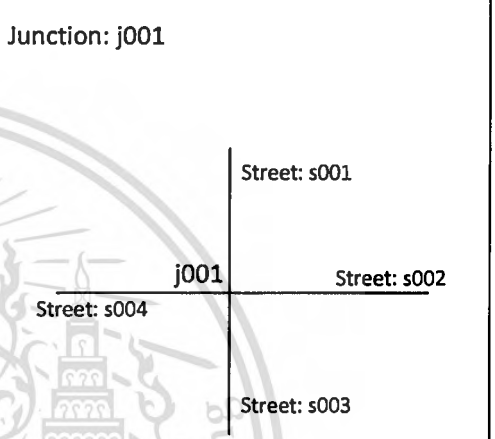
There are a lot of types of junctions. We are to show that our proposed schema can accommodate all kind of junctions in practice.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

For the first case of junction, it is a simple junction; an intersection. In this type of junction have four streets that connect to this junction. From the Figure 3.4, the *j001* junction is an intersection connects to three junctions and four streets. The information is shown in Table 3.3.

Table 3.3: 1st junction case

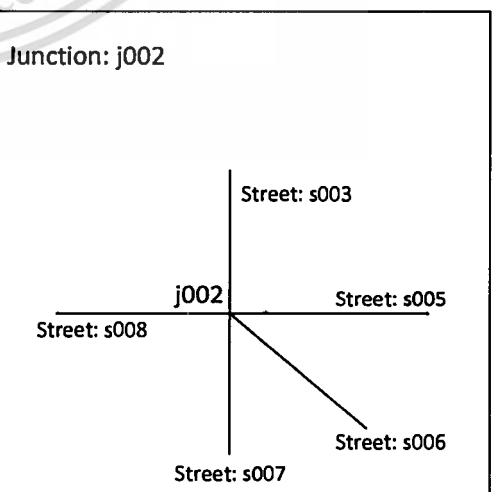
```
<!-- Case 1 -->
<junction junctionID = "j001">
    <junctionName>J1</junctionName>
    <junctiontojunction>
        <junctionRef>j002</junctionRef>
        <junctionRef>j004</junctionRef>
        <junctionRef>j007</junctionRef>
    </junctiontojunction>
    <junctiontostreet>
        <streetID>s001</streetID>
        <streetID>s002</streetID>
        <streetID>s003</streetID>
        <streetID>s004</streetID>
    </junctiontostreet>
</junction>
```



The second case of junction, a junction is where at five streets intersect. From the Figure 3.4, the *j002* is in this case. It connects to three junctions and five streets. The information is shown in Table 3.4.

Table 3.4: 2nd junction case

```
<!-- Case 2 -->
<junction junctionID = "j002">
    <junctionName>J2</junctionName>
    <nextToJunction>
        <junctionRef>j001</junctionRef>
        <junctionRef>j003</junctionRef>
        <junctionRef>j011</junctionRef>
    </nextToJunction>
    <nextToStreet>
        <streetID>s003</streetID>
        <streetID>s005</streetID>
        <streetID>s006</streetID>
        <streetID>s007</streetID>
        <streetID>s008</streetID>
    </nextToStreet>
</junction>
```

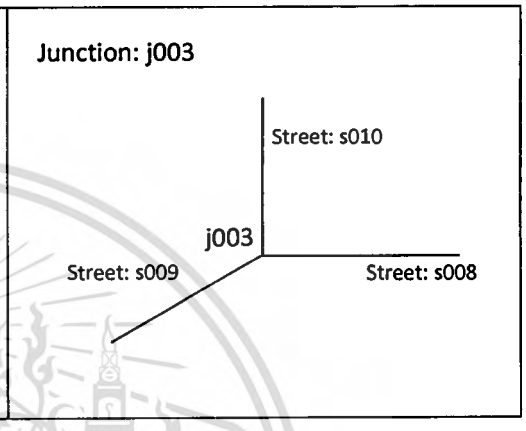


This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

The third case of junction, a junction is where at three streets intersect. From the Figure 3.4, the *j003* is in this case. The information of this junction type is shown in Table 3.5.

Table 3.5: 3rd junction case

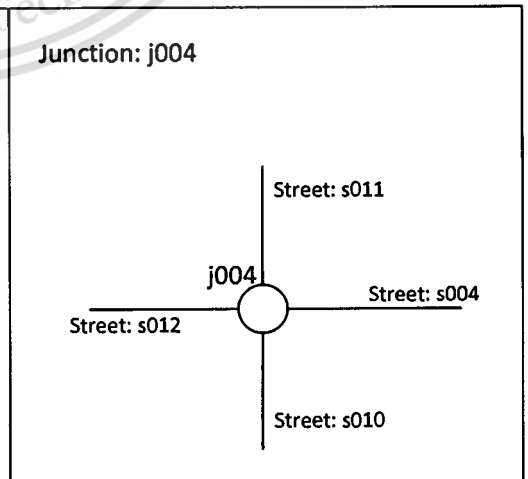
```
<!-- Case 3 -->
<junction junctionID = "j003">
    <junctionName>J3</junctionName>
    <junctionDescr>J3</junctionDescr>
    <nextToJunction>
        <junctionRef>j002</junctionRef>
        <junctionRef>j004</junctionRef>
    </nextToJunction>
    <nextToStreet>
        <streetID>s008</streetID>
        <streetID>s009</streetID>
        <streetID>s010</streetID>
    </nextToStreet>
</junction>
```



For the forth case of junction, it is a roundabout. We see a roundabout as a simple junction in case of the roundabout is just for cars pass and no buildings or places we want to reference around. From the Figure 3.4, the *j004* junction is categorized in the type of junction connects to three junctions and four streets. The information of this junction type is shown in Table 3.6.

Table 3.6: 4th junction case

```
<!-- Case 4 -->
<junction junctionID = "j004">
    <junctionName>J4</junctionName>
    <junctionDescr>J4</junctionDescr>
    <nextToJunction>
        <junctionRef>j001</junctionRef>
        <junctionRef>j003</junctionRef>
        <junctionRef>j005</junctionRef>
    </nextToJunction>
    <nextToStreet>
        <streetID>s004</streetID>
        <streetID>s010</streetID>
        <streetID>s011</streetID>
        <streetID>s012</streetID>
    </nextToStreet>
</junction>
```

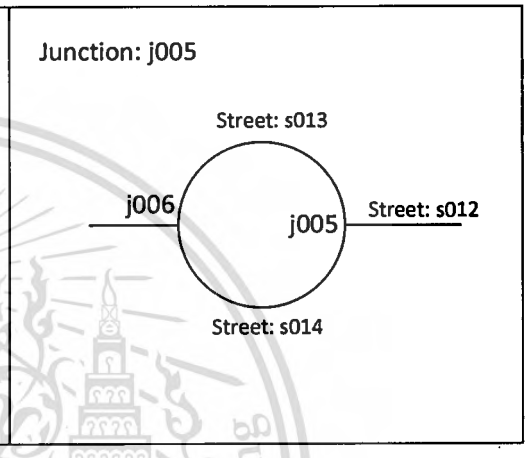


This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

The fifth case of a junction is the junction has two streets refer to the same next junction. From the Figure 3.4, the *j005* junction is categorized in the type of junction that has two junctions and three streets are referred to. The information is shown in Table 3.7.

Table 3.7: 5th junction case

```
<!-- Case 5 -->
<junction junctionID = "j005">
     <junctionName>J5</junctionName>
      <junctionDescr>J5</junctionDescr>
      <nextToJunction>
            <junctionRef>j004</junctionRef>
            <junctionRef>j006</junctionRef>
     </nextToJunction>
     <nextToStreet>
            <streetID>s012</streetID>
            <streetID>s013</streetID>
            <streetID>s014</streetID>
     </nextToStreet>
 </junction>
```

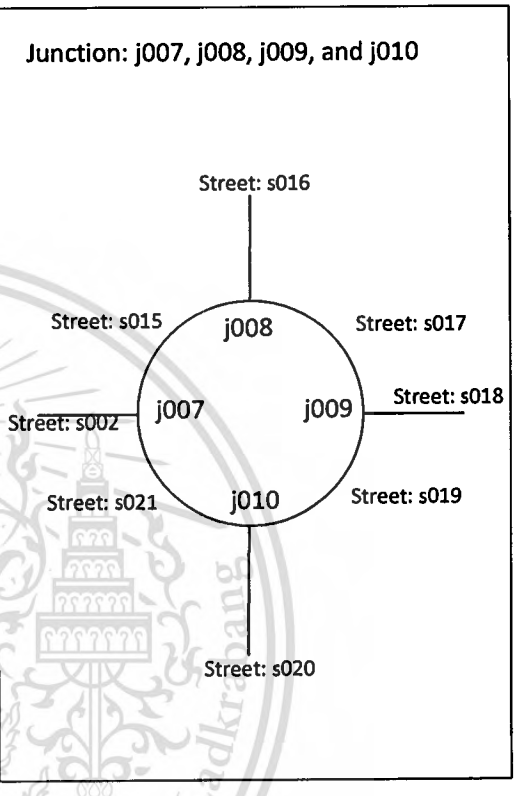


For the sixth case of junction, it is a traffic circle or a big roundabout. We see this case as a junction is where at three streets intersect for each routeway that goes to the circle. The traffic circle or roundabout in this case has to be big enough or has places want to refer to. From the Figure 3.4, the *j007*, *j008*, *j009*, and *j010* junctions are examples in the type of junction. The information of this junction type is shown in Table 3.8.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Table 3.8: 6th junction case

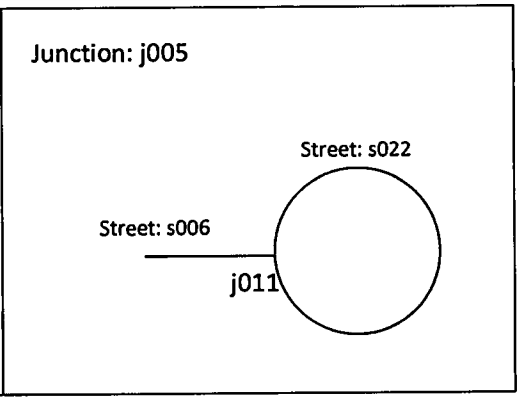
```
<!-- Case 6 -->
<junction jID = "j007">
     <junctionName>J7</junctionName>
      <junctionDescr>J7</junctionDescr>
     <nextToJunction>
            <junctionRef>j001</junctionRef>
            <junctionRef>j008</junctionRef>
            <junctionRef>j010</junctionRef>
     </nextToJunction>
     <nextToStreet>
            <streetID>s002</streetID>
            <streetID>s015</streetID>
            <streetID>s021</streetID>
     </nextToStreet>
 </junction>
<junction jID = "j008">
     <junctionName>J8</junctionName>
     <junctionDescr>J8</junctionDescr>
      <nextToJunction>
            <junctionRef>j007</junctionRef>
            <junctionRef>j009</junctionRef>
     </nextToJunction>
     <nextToStreet>
            <streetID>s015</streetID>
            <streetID>s016</streetID>
            <streetID>s017</streetID>
     </nextToStreet>
</junction>
...
```



For the last case of junction, the junction can refer to itself. From the Figure 3.4, the *j011* junction is in this type of junction that has two junctions and two streets are referred to. The information of this junction type is shown in Table 3.9.

Table 3.9: 7th junction case

```
<!-- Case 7 -->
<junction jID = "j011">
     <junctionName>J11</junctionName>
      <junctionDescr>J11</junctionDescr>
     <nextToJunction>
            <junctionRef>j002</junctionRef>
            <junctionRef>j011</junctionRef>
     </nextToJunction>
     <nextToStreet>
            <streetID>s006</streetID>
            <streetID>s022</streetID>
     </nextToStreet>
 </junction>
```



This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

From all case above, all types of junctions can be modified and constructed from the junction DTD schema. The junction can be a junction, can refer to itself, can be a traffic circle, and can be a roundabout.

Our last schema is the street schema (Table 3.10). It has street IDs (streetID), street names (streetName), street descriptions (streetDescr), and other information (in this case is restaurants). In the restaurant contains sub elements named restaurantID refers to restaurants in the restaurant schema.

Our system treats every individual place as a building as an entity in Buildings with a criteria in mind that if the class of locations in building is interesting enough, we partition it to a dedicated DTD. Therefore, Restaurants is an example of a layer of a class of our interest. In this example, we put the restaurant into street schema for better understanding.

Table 3.10: Street DTD schema

```
<!DOCTYPE streets[
  <!ELEMENT street (streetName,
streetDescr, restaurant)>
  <!ATTLIST street streetID ID #REQUIRED>
  <!ELEMENT streetName (#PCDATA)>
  <!ELEMENT streetDescr (#PCDATA)>
  <!<!ELEMENT streettorestaurant
(restaurantID)*>
  <!ELEMENT restaurantID (#PCDATA)>
]>
```

```
<street streetID="s001">
      <streetName>Street1</streetName>
      <streetDescr>Street1</streetDescr>
      <streettorestaurant>
            <restaurantID>r001</restaurantID>
            <restaurantID>r002</restaurantID>
      </streettorestaurant>
</street>
```

3.2.2 Mapping DTD to database

3.2.2.1 Object-relational mapping overview

The way to map DTD to database we will use technique called "object-relational mapping". This models the XML document as a tree of objects that are specific to the data in the document, then maps these objects to the database. An object-relational mapping is performed in two steps. First, an XML schema (a DTD in this case) is mapped to an object schema, and then the object schema is mapped to the database schema.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

The two mappings can optionally be combined for a direct DTD-to-database mapping.

The mapping starts by recognizing that element types are data types. Element types that have PCDATA-only content are called simple element types. These hold a single data value and are equivalent to scalar data types in an object-oriented programming language. Attribute types are also simple types. Element types that have element or mixed content, or that have attributes, are called complex data types. These hold a structured value and are equivalent to classes in an object-oriented programming language. Note that an element type that has empty content and attributes is still "complex". The reason for this is that the attributes also provide structure and are roughly equivalent to child PCDATA-only elements.

The object-relational mapping first maps simple types to scalar data types. It then maps complex types to classes, with each element type in the content model of the complex type mapped to a property of that class. The data type of each property is the data type to which the referenced element type is mapped. References to complex element types are mapped to pointers/references to an object of the class to which the complex element type is mapped. The last part of the mapping maps attributes to properties, with the data type of the property determined from the data type of the attribute. Note that attributes are equivalent to references to element types in a content model. This is because, like references in a content model, they are local to a given element type. The only conceptual difference is that the attribute type is defined locally, rather than at a global (DTD-wide) level, as is the case with element types.

For example; Table 3.11, in the following the simple element types B, D, E, and the attribute F are all mapped to Strings and the complex element types A and C are mapped to classes A and C. The content models and attributes of A and C are mapped to properties of classes A and C. The references to B, D, and E in the content models of A and C are mapped to String properties (because the types are mapped to Strings) and the attribute F is also mapped to a String property. The reference to C

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

in the content model of A is mapped to a property with the type pointer/reference to an object of class C because element type C is mapped to class C.

Table 3.11: Mapping DTD to object

```
          DTD                                  Classes
========================                  =============

<!ELEMENT A (B, C)>                       class A {
<!ELEMENT B (#PCDATA)>                       String b;
<!ATTLIST A                          ==>     C      c;
       F CDATA #REQUIRED>                    String f;
                                          }
<!ELEMENT C (D, E)>                       class C {
<!ELEMENT D (#PCDATA)>               ==>     String d;
<!ELEMENT E (#PCDATA)>                       String e;
                                          }
```

In the second part of the object-relational mapping; Table 3.12, classes are mapped to tables (known as class tables), scalar properties are mapped to columns, and pointer/reference properties are mapped to primary key/foreign key relationships. For example:

Table 3.12: Mapping object to table

```
     Classes                                Tables
=============                          ==================
class A {                              Table A:
   String b;                              Column b
   C      c;          ==>                 Column c_fk
   String f;                              Column f
}

class C {                              Table C:
   String d;          ==>                 Column d
   String e;                              Column e
}                                         Column c_pk
```

Note that the tables are joined by a primary key (C.c_pk) and a foreign key (A.c_fk). Because the relationship between the parent and child elements is one-to-one, the primary key can be in either table. If the relationship is one-to-many, the primary key must be on the "one" side of the relationship, regardless of whether this is the parent or child.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

A primary key column can be created as part of the mapping, as is the case for the column c_pk, or an existing column or columns can be used as the primary key. If a primary key column is created as part of the mapping, its value must be generated by the database or the data transfer software. While this is generally considered better database design than using data columns as primary keys, it has a disadvantage when used with XML, and this is that the generated key is meaningless outside the source database. Thus, when data with a generated key is transferred to an XML document, it will either contain a meaningless primary key (if the primary key value is transferred) or no primary key at all (if the key is not transferred). In the latter case, it may be impossible to re-identify the source of the data, which is a problem if the data is modified and returned to the database as an XML document.

If the + or * or ? Operator is applied to a reference, the reference is again mapped to a single property, which this time is an array of unknown size. Since the number of values can be arbitrarily large, the property must be mapped to a property table, which will contain one row for each value. The property table is linked to the class table by a primary key, foreign key relationship, where the primary key is in the class table. For example,

Table 3.13: Mapping DTD that has the + or * or ? operator to database

DTD		Classes		Tables

<!ELEMENT A (B+, C)>
<!ELEMENT B (#PCDATA)>
<!ELEMENT C (#PCDATA)>
	==>	class A {
		String[] b;
		String c;
		}
			==>	Table A
				Column a_pk
				Column c
				
				Table B
				Column a_fk
				Column b

3.2.2.2 Mapping our DTDs to database

In our mapping, we modify from object-relational mapping to suit our database schema. For example in Figure 3.5, we use attribute of area (*areaID*) as primary key in an area schema. The sub element of the

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

areatojunction, we create new table name *areatojunction*, *junctionID* element can be more than one value.

```
<!-- Area DTD schema -->
<!ELEMENT area (areaName, areaDescr,
areatojunction)>
<!ATTLIST area areaID ID #REQUIRED>
<!ELEMENT areaName (#PCDATA)>
<!ELEMENT areaDescr (#PCDATA)>
<!ELEMENT areatojunction (junctionID)*>
<!ELEMENT junctionID (#PCDATA)>
```

==>

```
Table area{
   Column  areaID;
   Column  areaName;
   Column  areaDescr;
}
Table areatojunction{
   Column  areaID;
   Column  junctionID;
}
```

Table: area

areaID	areaName	areaDescr
Int(2)	Varchar(45)	Varchar(45)

Table: areatojunction

areaID	junctionID
Int	Char(4)

Figure 3.5: Mapping area DTD schema to database

If we use object-relational mapping, the table *area* will have a foreign key refer to the primary key field in the table *areatojunction*. Instead of that idea, we have a primary key of the table *area* that the table *areatojunction* can refer to. In table *areatojunction*, we use both columns to be its primary key.

For the junction schema; Figure 3.6, we have three tables; *junction*, *junctiontojunction* and *junctiontostreet*. The table *junction*, with the same idea of the area schema, *junctiontojunction* and *junctiontostreet* tables have fields that refer to a primary key of table *junction* and both columns of each table are primary keys.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

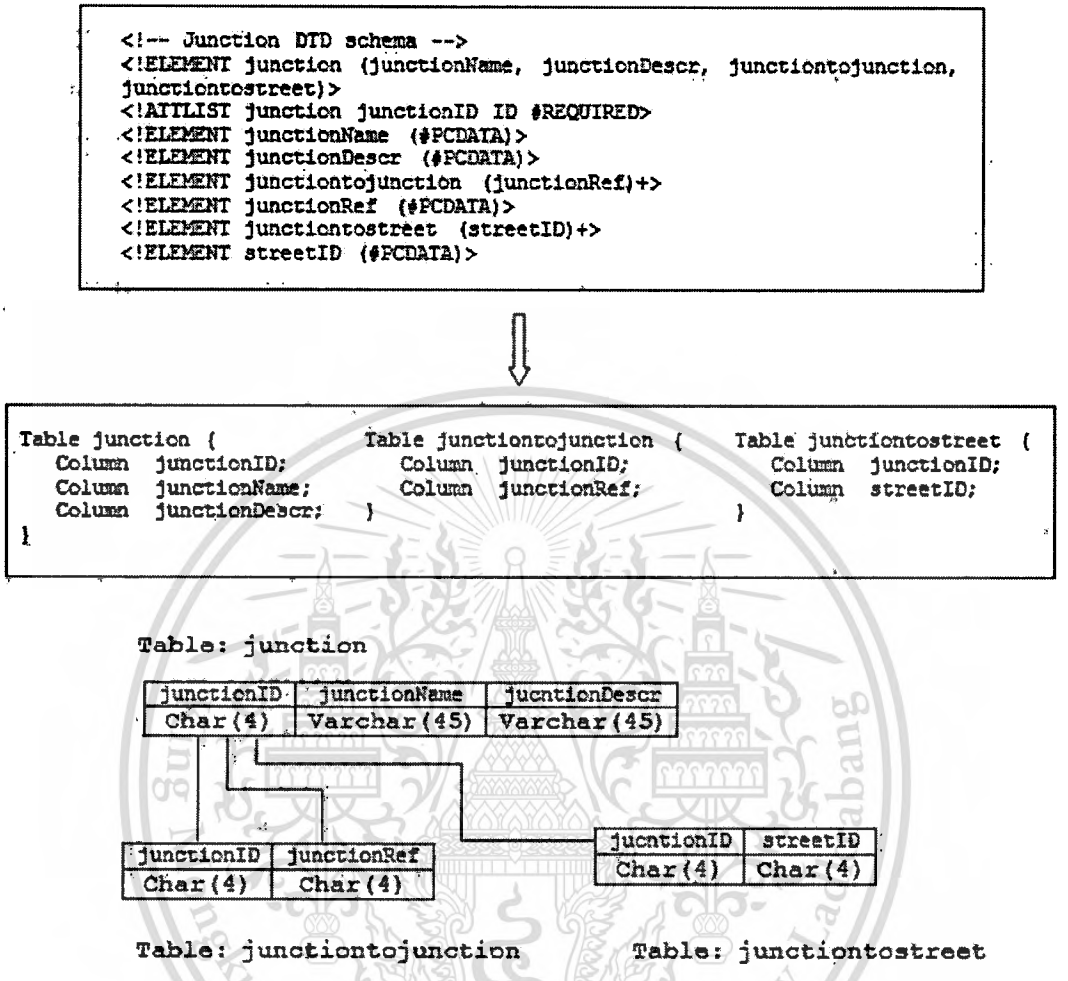


Figure 3.6: Mapping junction DTD schema to database

Mapping street DTD schema to the database (Figure 3.7) is similar to two schemas above. We have main table named *street* and another tables refer to the table *street*. For the example of this schema, *restaurant* is referred.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

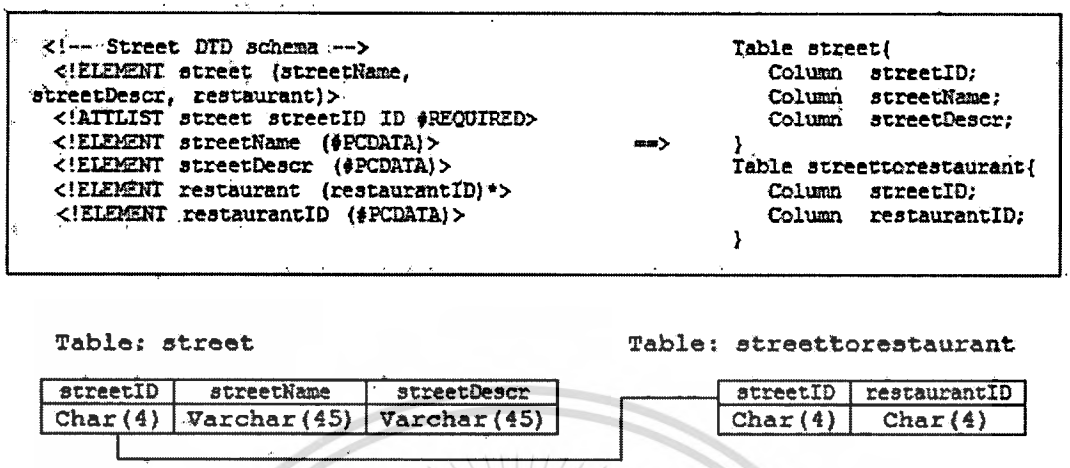


Figure 3.7: Mapping street DTD schema to database

We store our data in a DBMS. The reason is that at this moment we prefer DBMS to manage system memory resource. The way most people connect databases to the Web is with some external program such as a CGI script, PHP, or ASP. But the data itself is not defined as a database entry until it is actually in the database. If you use XML connects to your databases to the Web, once the data is entered into XML it is in a format compatible with the database.

3.3 Creating Relationship among Junctions

We fill our adjacency matrix after we obtain the graph of our interested area. For example, Figure 3.8.

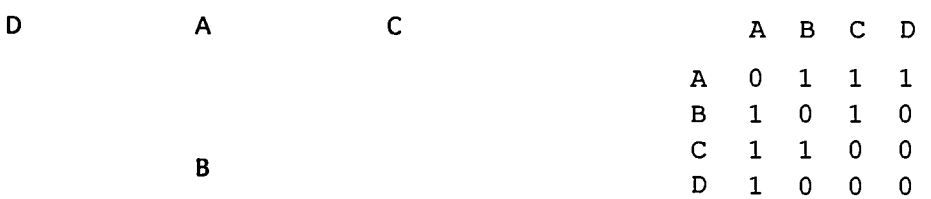


Figure 3.8: Adjacency Matrix Example

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

To find a relationship between each junction, we query pairs of junctions that adjacent to each other from database in table *jucntiontojunction*. Then, Perl will create an adjacency matrix graph. In this way, the program can retrieve relationships between junctions. For example, according to Figure 3.4, table *jucntiontojunction* stores junctions in pairs such as (*j001*, *j002*), (*j001*, *j004*), (*j002*, *j001*), (*j002*, *j011*) and etc, the adjacency matrix will be liked Figure 3.9.

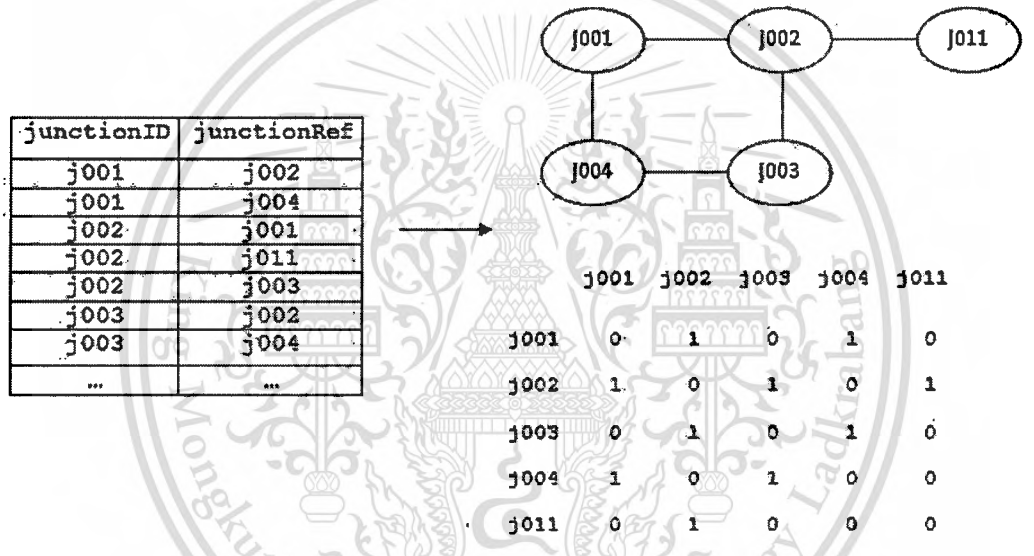


Figure 3.9: Constructing WordNet of Figure 3.4 by adjacency Matrix

3.4 Querying and Ranking Result

We now could search and rank for relevant restaurants from a location of interest. If there is a query for area A restaurants, restaurants around area A are queried as well. Its result is ranked lower than the result of restaurants in area A because of its distance is obviously further than those in A area. We, again, use Perl to create multiple queries from the user query to obtain the restaurants.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

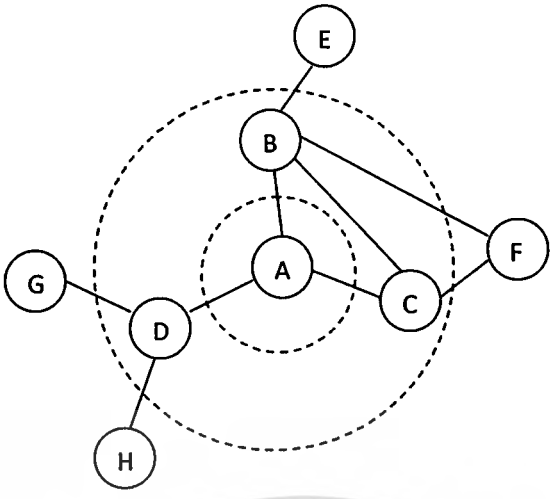


Figure 3.10: Searching and ranking degree.

We query the area in degrees; the first degree covers area A plus outside-A junctions that are next to junctions along area A border . Then we query the lower degree. From the Figure 3.10, the area A is the first degree, the B, C and D are in the second degree, and the E, F, G and H are in the third degree.

The rank result will be represented in a degree and a group of direction. If we can construct a graph as Figure 3.10, A is a center area. D, H and G is one group of direction (H and G is ranked lower than D). From the route A to F, the graph has two routes; A – B – F and A – C – F. The problem is which group of direction F belongs to (B or C). In our system, we leave F result with in both routes to tell that both ways can go to F. Therefore, from the Figure 3.10, the system will rank area A in the highest priority. Then follows with 3 routes; B – E – F (E and F are the same priority), C – F, and D – G – H (G and F are the same priority).

In the system, we will rank restaurants in a center area with 3 expanded junctions around it. In other word, from Figure 3.10, the A node is a center area and B – G nodes are junctions.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3.5 Demonstration of the Result

The result of the system is shown in an XML format. We used XML DOM and Java Script to get data in the XML file and show the result in a HTML page. A DTD Schema of XML file is shown in table 3.14

Table 3.14: Output DTD Schema

```
<!ELEMENT RESTAURANTS (ROUTE)*>
<!ELEMENT ROUTE (ROUTENAME, RESTAURANT*)>
<!ELEMENT ROUTENAME (#PCDATA)>
<!ELEMENT RESTAURANT (ID, NAME, URL, MAP, LOGO, PRICE, TYPE)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT MAP (#PCDATA)>
<!ELEMENT LOGO (#PCDATA)>
<!ELEMENT PRICE (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
```

The ROUTE element represents each group of direction, the first ROUTE element always represent the center area. For the first ROUTENAME element contains area name, otherwise the ROUTENAME element contains the first junction name that next to the center area. The RESTAURANT element has all information of each restaurant. Table 3.15 is an example of an XML file.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Table 3.15: Output XML file

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE RESTAURANTS SYSTEM "restaurants.dtd">
<RESTAURANTS>
       <ROUTE>
              <ROUTENAME>Thong Lo(Sukhumvit 55)</ROUTENAME>
              <RESTAURANT>
                     <ID>R051</ID>
                     <NAME>After You Dessert Cafe</NAME>
                     <URL></URL>
                     <MAP>T51.gif</MAP>
                     <LOGO>T036.gif</LOGO>
                     <PRICE>Less Than 100</PRICE>
                     <TYPE>International Food</TYPE>
              </RESTAURANT>
              <RESTAURANT>
                     <ID>R011</ID>
                     <NAME>Banrie coffee</NAME>
                     <URL>http://www.banriecoffee.com/index.htm</URL>
                     <MAP>E11.gif</MAP>
                     <LOGO>E011.gif</LOGO>
                     <PRICE>101 - 300</PRICE>
                     <TYPE>Thai Food</TYPE>
              </RESTAURANT>
       <ROUTE>
<RESTAURANTS>
```

3.6 Filtering the Result

From more efficient searching of our program, we allow users to specify a restaurant type and a price per person. These features are queried in restaurants table in database. This way, the users will receive more relevant information.

In our program, users can retrieve restaurants they want, by using a restaurant type and price per person features to filter. However users can only choose one choice for each feature. In the other word, they can choose only one type of the restaurant and/or one type of price per person, for example, choose a restaurant type feature to be Thai restaurant and/or choose a price per person feature to be 101 – 300 baht, they cannot choose Thai and Japanese restaurant types at once.

Nevertheless, if users do not want to choose any of these features, they can set the restaurant type to be unclassified and the price per person to be any. Then, the system will retrieve all kinds of restaurant to the users.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Chapter 4

Evaluation

4.1 System Operation

4.1.1 First User Interface Page of the System

When users go to the first page of the Junction-based location query system for restaurants web site, they will see as Figure 4.1.

A. Area list
B. Price per person list
C. Restaurant type list
D. Search button

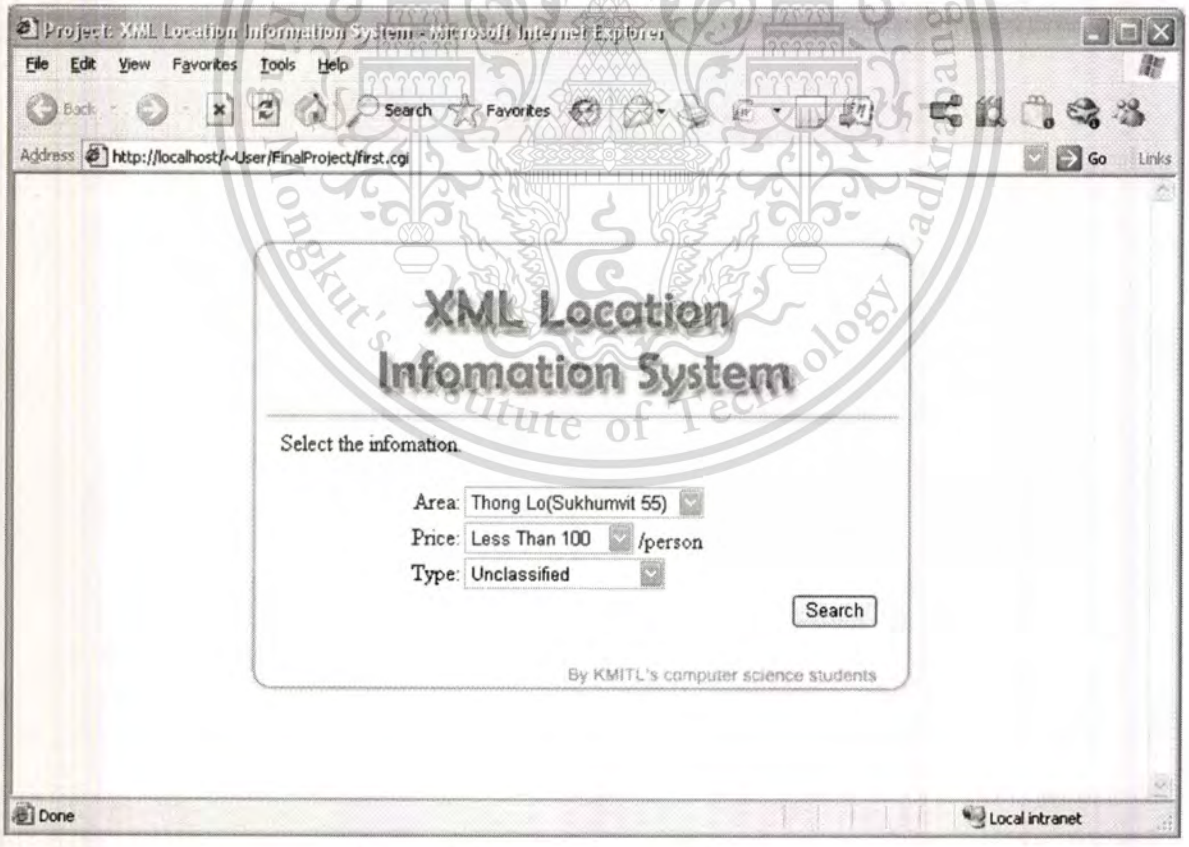


Figure 4.1: First Page of the System

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

4.1.2 Display the Result

To display the result from the Junction-based location query system for restaurants consists of 2 parts that connect to each other; data (result from the system) and demonstration.

4.1.2.1 Data

As we discussed in chapter 3, the result of the system is in the XML format named *data.xml* in Figure 4.2. The first route in the XML file always is the center area.

Figure 4.2: data.xml

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

4.1.2.2 Demonstrate the result

After running the program, the system will create the *data.xml* file to contain all of the restaurant result. Our program shows the restaurant result in HTML file by using JavaScript to get the data from the *data.xml* file. The page looks like the Figure 4.3.

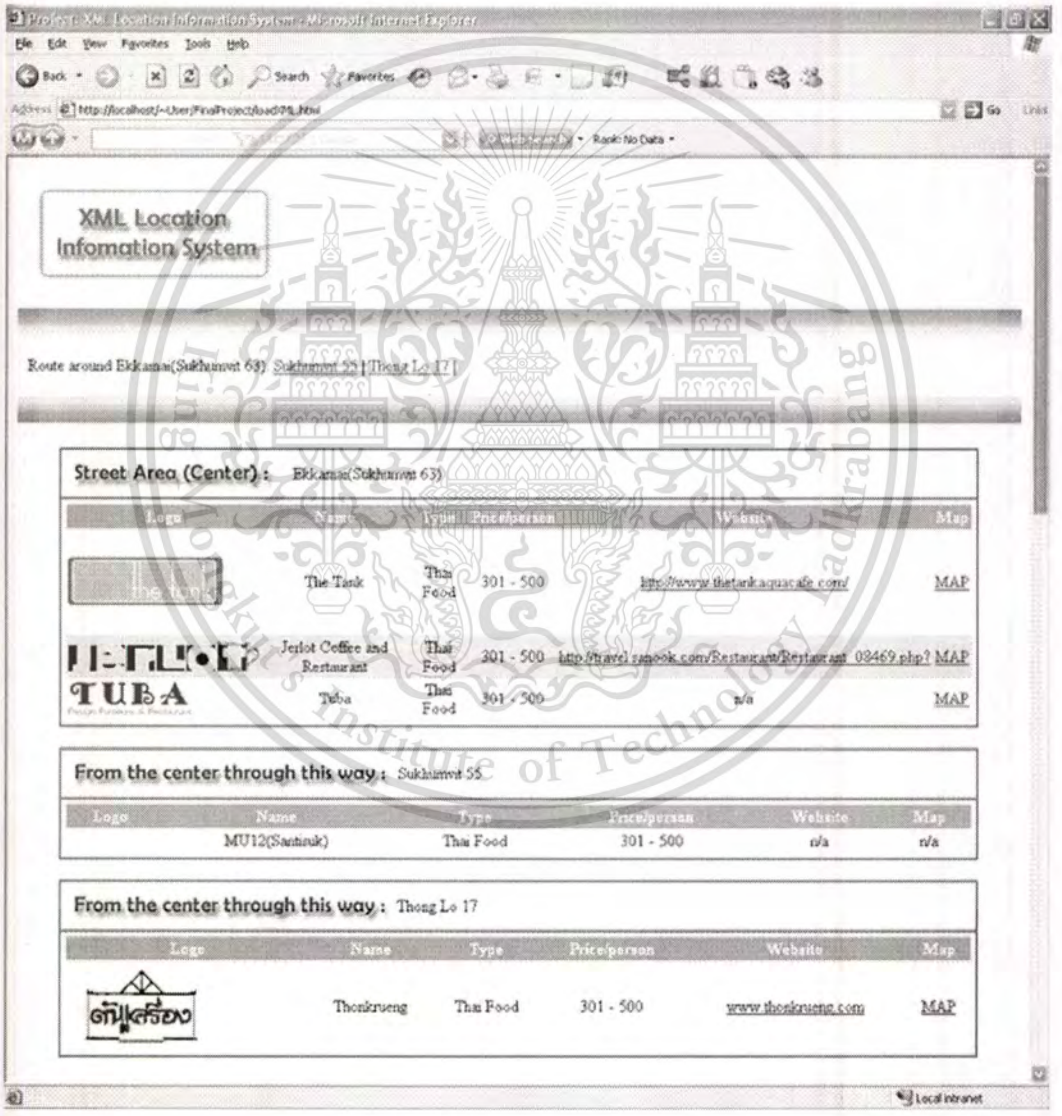


Figure 4.3: Output Page

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

From the Figure 4.3, you will see that the center area, *Ekkamai*, is the highest ranking. *Sukhumvit 55* and *Thong Lo 17*, the route ways going out from the center, are the same priority, even though; *Sukhumvit 55* is shown higher than *Thong Lo 77*. Therefore, we create links to each route above the results makes the users can go to all routes evenly. However, restaurants in each route are ranked from the nearest from the center.

4.2 Evaluation

In our program, after we specify the area of our interest, the result shows restaurants in and around the area. We compare the outcome with Google search engine by use the same keyword in our program or concatenates with "restaurant", for example "Thong Lo restaurant", "Sukhumvit 63" and etc.

We compare to Google search engine because it is the most popular search engine right now. For this evaluation, we do count only the non-duplicate restaurants. Since Google shows only ten web results per
Google's page, we manually open ten first result pages and count for non-duplicate name only compare to our program.

Table 4.1: Compare to Google (05/01/2008)

Area	Our Program (restaurants)	Keyword	Google's 1st page (restaurants/web pages)
Thong Lo	26 (center) 18 (around)	Thong Lo	11/2
		Thong Lo Restaurant	8/3
		Sukhumvit 55	1/1
		Sukhumvit 55 Restaurant	9/6
		Sukhumvit 55 Thong Lo Restaurant	10/2
Ekkamai	15 (center) 30 (around)	Ekkamai	3/3
		Ekkamai Restaurant	13/7
		Sukhumvit 63	1/1
		Sukhumvit 63 Restaurant	7/6
		Sukhumvit 63 Ekkamai Restaurant	15/4

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

From the Table 4.1, you can see that the results from all keywords are equal or less than the results from a single query in our system.

Comparing to Google shows that our system is simpler way to use. Users have to try many keywords in Google to get the relevant information. The users have to pick the right one on their own that means they have to type in and check spelling themselves. Since the name of areas, streets and roads in Bangkok are in non-English language, some sites spell different from others.

When searching our system, it shows straight answer in next page. The users do not have to click again to find out information. Google is a search engine for webpage its first page result shows ten webpage as in Figure 4.4. The users have to decide on their own which result is interesting for them. Then, click to find whether the information matches their expectation.

Our system display much more meaningful information. From our evaluation, some pages liked from Google are irrelevant information such as hotels or condominiums information even the search key used has a word "restaurant". Of all result shown in table 4.1, only one page from Google with a search key "Sukhumvit 63 Ekkamai Restaurant" is dedicated to restaurant- a somewhat a direct hit in our perspective. It returns 15 restaurants like our system. (By the way, it was not the same list as our system.) Furthermore, our system result is sorted in relative to distance from our location of origin and provides useful information for users such as type of restaurant and price per head.

Many results from Google are duplicated information. Some pages from sites contain identical information. Our system shows duplicate restaurants around the center area like Google does. However, it does that with a purpose. Usually users already have their mind set on which direction from the origin they would prefer to choose. This way the system helps filtering restaurants for direction users choose. It should be noted that Google beats our system in term of response time.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Figure 4.4: Google's Example First Page

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

The objectives of this project to develop Junction-based location query system for restaurants for search the restaurant by using the user keyword to rank the places nearby. User can also choose price per person or type of food to improve the result. The system will show the list of the restaurant that the user query such as rank in between '300-500 baht' per person or 'Thai Food', 'Japanese Food' or 'Bakery'. The evaluation shows that our system is much easier to use as well as understanding the precise result.

We design the system's infrastructure for an area semantic searching on web platform by using MySQL, Perl and Apache Server. The result of our system is represented in XML format, which means it is very easy to use HTML page illustrates our result.

5.2 Recommendation

There are many areas where we can improve our system. For example:

1. If the database is huge, it will need a well management. Partial database can be one of solutions to reduce search space.

2. An area could have many aliases. Though our system bases on unofficial name, we have not allowed user to refer to the area by any other names.

3. The system could combine other classes of layers altogether to create a more complete map. As we mentioned, restaurant is just an example of a class of our interest. It also should be noted here that

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

different class such as real-estate might require different search algorithm due to different coverage area.

4. The system should be able to cooperate with Google Earth. With visual map feature, user can obtain driving direction.

5. Finding a way to import the real map to our structure, so that we could obtain a consistent and complete data in our system. Currently we allow user to participate in creating our map.

6. Web application could allow users to quickly change to location of origin. Once users see the result page, they should be able to re-submit the query from the page to change their point of origin.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

References

[1] Quigley, Ellie. 1997. "*Perl by example*" 2nd ed., Upper Saddle River, NJ: Prentice Hall PTR.

[2] Chase, Nicholas. 2001. "*XML and Java from scratch*" Indianapolis, Ind., Que.

[3] Pitts-Moultis, Natanya. 2001. "*XML black book*" 2nd ed., Scottsdale, AZ: Coriolis Group Books.

[4] MySQL AB *Why MySQL? 1995-2008* [Online]. Available URL: http://www.mysql.com/why-mysql/

[5] PerlScriptsJavaScripts.com *Using MySQL with Perl 1999-2007* [Online]. Available URL: http://www.perlscriptsjavascripts.com/tutorials/mysql/index.html

[6] CGI101.com *CGI Programming 101* [Online]. Available URL: http://www.cgi101.com/learn/

[7] Erack Network *PERL Tutorial* [Online]. Available URL: http://www.tizag.com/perlT/

[8] Ronald Bourret *Mapping DTDs to Databases* [Online]. Available URL: http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html?page=1

[9] WolframMathWorld Adjacency Matrix [Online]. Available URL: http://mathworld.wolfram.com/AdjacencyMatrix.html

[10] Perlmeme.org *HOWTO* [Online]. Available URL: http://www.perlmeme.org/howtos/index.html

[11] W3 Schools *XML Tutorial* [Online]. Available URL: http://www.w3schools.com/xml/default.asp

[12] W3 Schools *DTD Tutorial* [Online]. Available URL: http://www.w3schools.com/dtd/default.asp

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

[13] W3 Schools *XML DOM Tutorial* [Online]. Available URL: http://www.w3schools.com/dom/default.asp

[14] W3 Schools *JavaScript Tutorial* [Online]. Available URL: http://www.w3schools.com/js/default.asp

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Appendices

A: Installing & Configuration

B: Brief Perl Tutorial

C: Related Literatures

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

For using our system, it needs a perl's server and database therefore three programs have to be installed:

- Apache HTTP Server 2.0
- ActivePerl 5.8.8
- MySQL Server 5.0 & Query Browser

A.1: Installing Apache on Windows XP

Firstly, go to http://httpd.apache.org/download.cgi and download the program (download the binary .msi file).

Next, double click file 

, it will launch the installer start screen that show the name and version of the program. Click "Next" for installing. (Fig A.1)

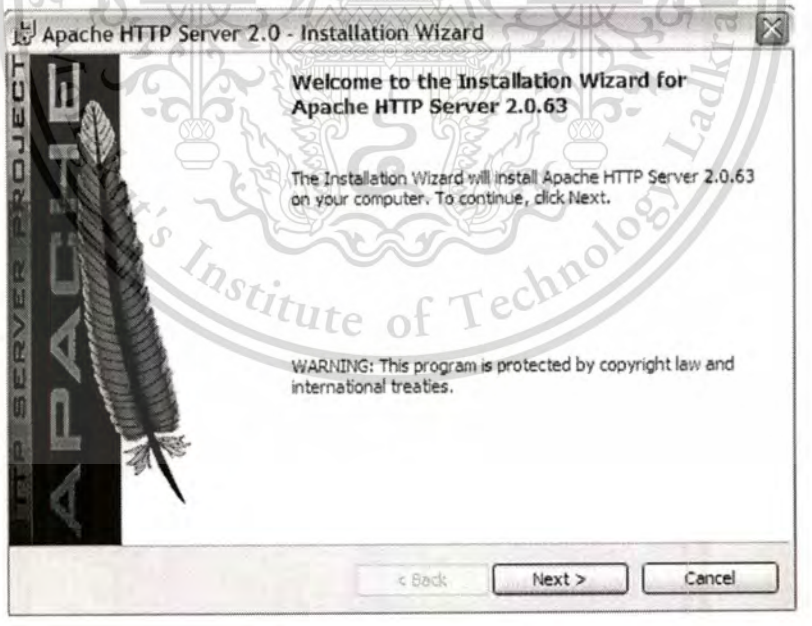


Fig A.1: Apache Installer Start Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

1. Read "Terms and Conditions", then choose "I accept…" for installing. (Fig A.2).

Fig A.2: License Agreement Screen

2. Use "localhost" for both the Network Domain and Server Name. Put your email address for the Administrator's Email Address. (Fig A.3)

Fig A.3: Server Information screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3. Select “Typical” and click “Next”. (Fig A.4)

Fig A.4: Setup Type Screen

4. Choose the Folder that you want to install. (Fig A.5)

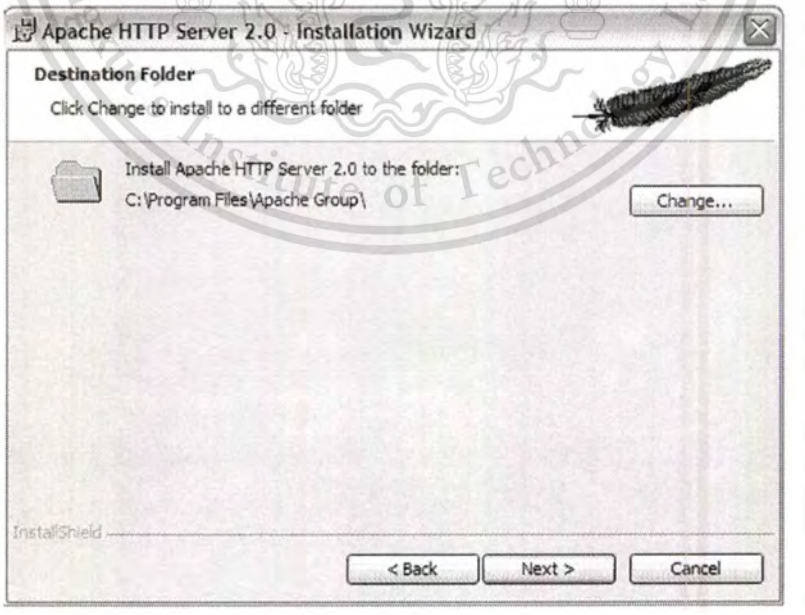


Fig A.5: Destination Folder Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

5. Click “Install” to start the installation. (Fig A.6&7)

Fig A.6: Installation Screen

Fig A.7: Installing Status Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

6. Click “Finish” to exit. (Fig A.8)

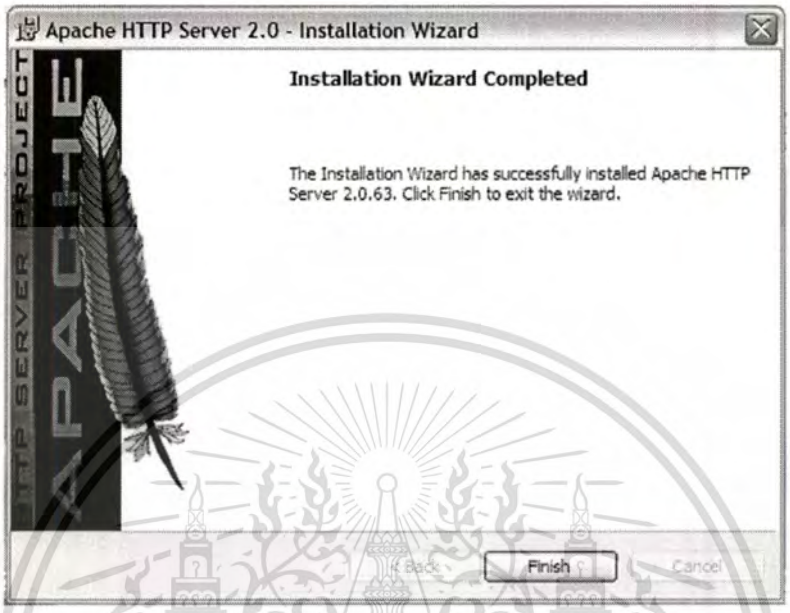


Fig A.8: Installation Complete Screen

7. Apache is already run on your machine. You can see from the icon . After that go to http://localhost to view your start page. (Fig A.9)

Fig A.9: Start Page in Web browser

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

To start or stop the Apache server has 2 ways.

1. Go to the Start menu and select All Programs → Apache HTTP Server → Control Apache Server. (Fig A.10)

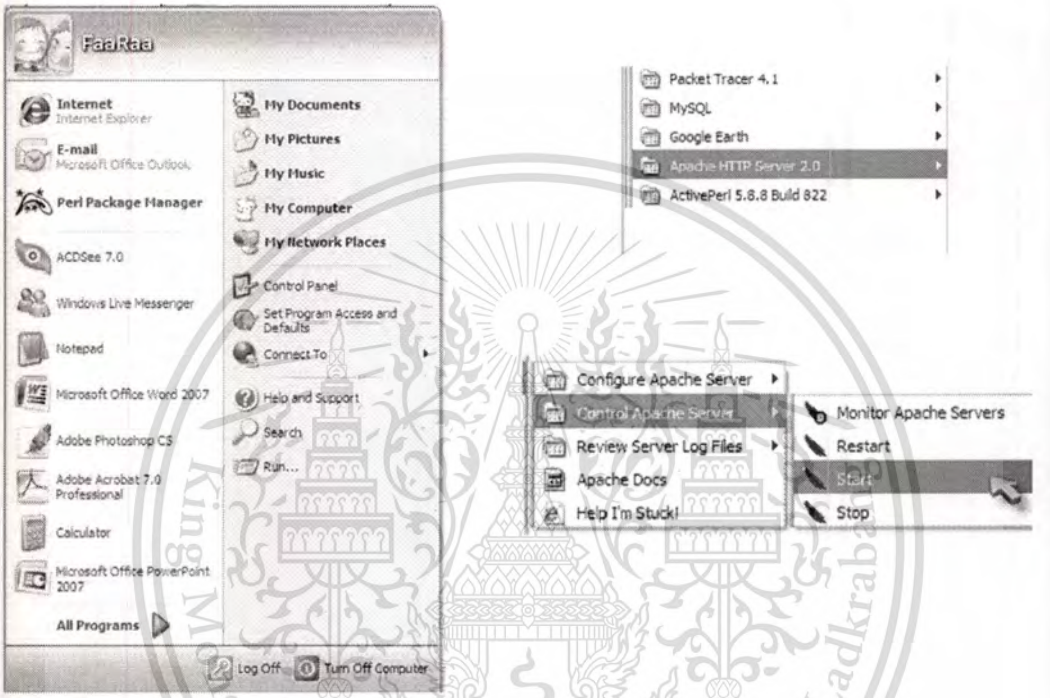


Fig A.10: The1st way to control the Apache

2. Go to icon  (Notification Area). (Fig A.11)

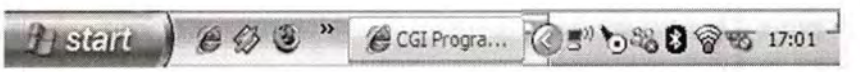


Fig A.11: The 2nd way to control the Apache

 Green color: Apache server is running.

 Red color: Apache server is stopping.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

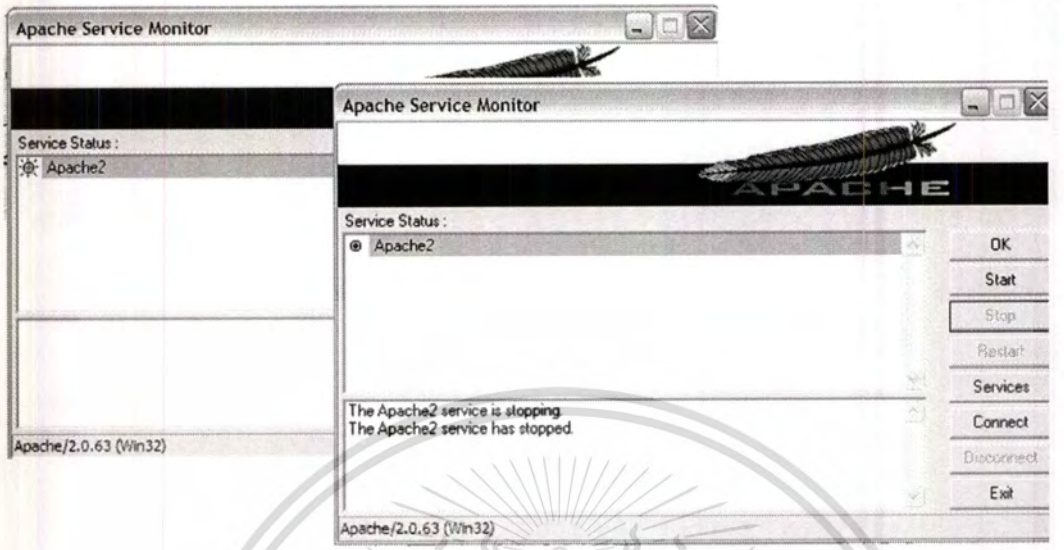


Fig A.12: Apache's Status Display

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

A.2: Installing ActivePerl on Windows XP

Go to http://www.activestate.com/Products/ActivePerl to download the latest version of Perl (Download the MSI file).

Next, double click file 

, it will launch the installer start screen that show the name and version of the program. Click "Next" for installing. (Fig A.13).

Fig A.13: ActivePerl Installer Start Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

1. Read “Terms and Conditions”, then choose “I accept…” for installing. (Fig A.14).

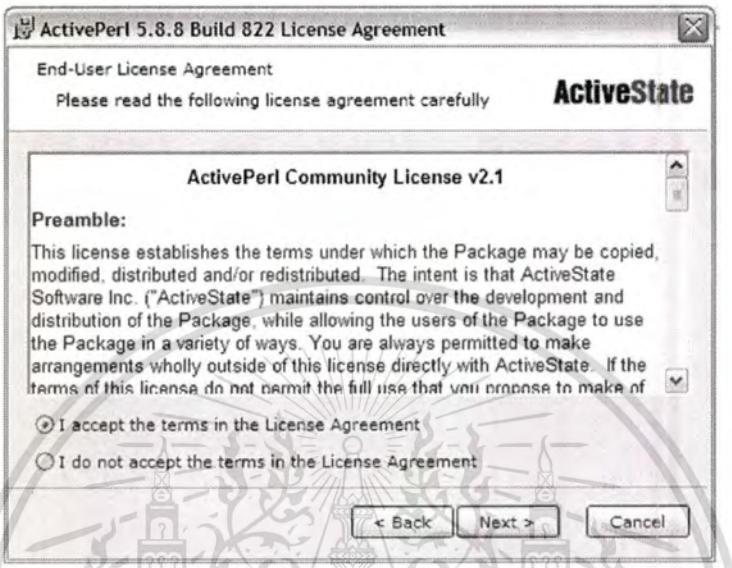


Fig A.14: License Agreement Screen

2. Choose the Folder that you want to install. (Fig A.15)

Fig A.15: Custom Setup Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3. Both “Add Perl to PATH…” and “Create Perl file…” should be checked. Then, click “Next”. (Fig A.16)

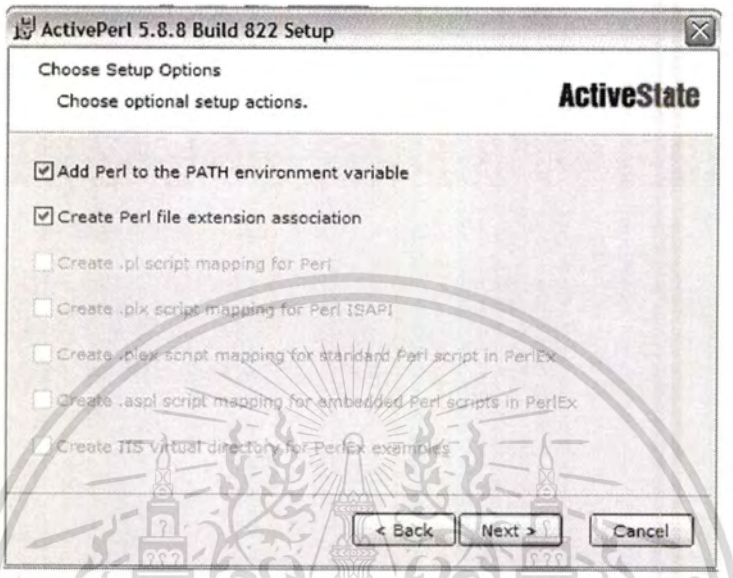


Fig A.16: Setup Options Screen

4. Click “Install” to start the installation. (Fig A.17&18)

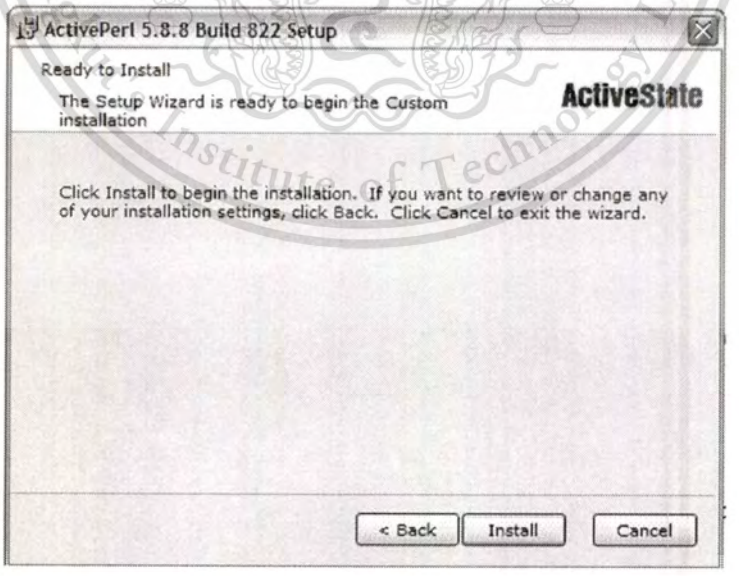


Fig A.17: Installation Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

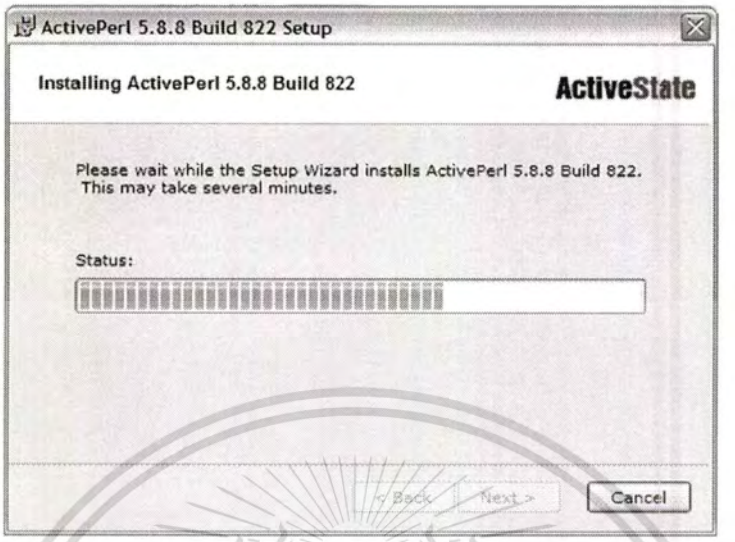


Fig A.18: Installing Status Screen

5. Click "Finish" to exit. (Fig A.19)

Fig A.19: Installation Complete Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Go to "C:\Perl\html" directory and select file "index.html" to open ActivePerl User Guide. (Fig A.20)

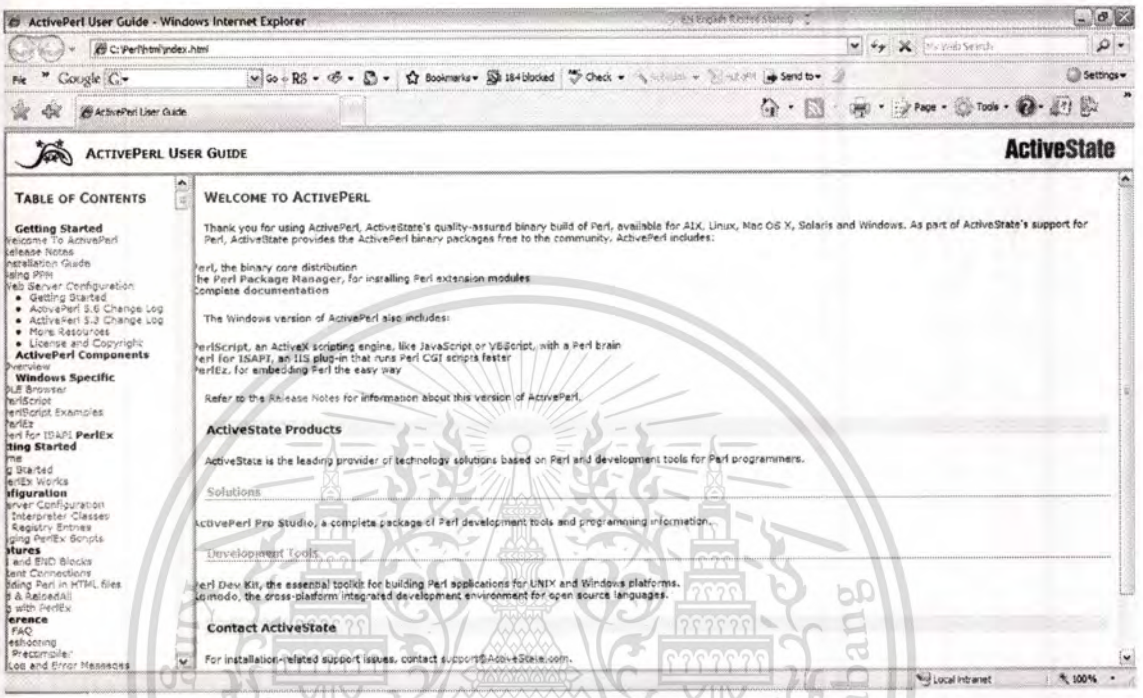


Fig A.20: Perl User Guide

Now Perl is installed. You must to modify the Apache server Configuration, to tell where your pages are and enable CGI programs.

Configuring Apache

1. Go to "My Documents" and create a new folder called "My Website" to store your web pages and CGI programs.

2. To open the config file by go to Start menu → All Programs → Apache HTTP Server → Configure Apache Server →Edit the Apache http.conf Configuration file.

3. Scroll down to find UserDir section. It should be:

```
UserDir "My Documents/My Website"
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

4. You have to uncomment this section by removing "#" signs at the beginning of each line.

```
#<Directory "C:/Documents and Settings/*/My Documents/My Website">
#  AllowOverride FileInfo AuthConfig Limit
#  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#  <Limit GET POST OPTIONS PROPFIND>
#    Order allow,deny
#    Allow from all
#  </Limit>
#  <LimitExcept GET POST OPTIONS PROPFIND>
#    Order deny,allow
#    Deny from all
#  </LimitExcept>
#</Directory>
```

Change the "Options" line like this:

```
Options MultiViews Indexes SymLinksIfOwnerMatch Includes ExecCGI
```

"Options" allow what options are available in the directory. The important ones are Indexes, that enables server-side includes and ExecCGI, which enables CGI programs in this directory.

5. Look for the DirectoryIndex line and add "index.cgi" at the end of the line.

```
DirectoryIndex index.html index.html.var index.cgi
```

6. Find the AddHandler section and uncomment the CGI line:

```
AddHandler cgi-script .cgi
```

That makes any file with .cgi to be processed as a CGI program.

If you also have files with .pl extension, input the .pl on the same line like this:

```
AddHandler cgi-script .cgi .pl
```

Next, enter to the new line below and add this line:

```
AddHandler server-parsed .html
```

To makes .html files to be searched for server-side includes tags.

After finish the configuration file, save and restart Apache.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Viewing Your Site

http://localhost is the homepage for your site.

To view the pages in your "My Website" folder, the URL is http://localhost/~yourUsername/ . For example, your username is "user", so the URL will be http://localhost/~user/ . (If you don't know your username, go to Start menu; your username is at the top of the Start box.) (Fig A.21)

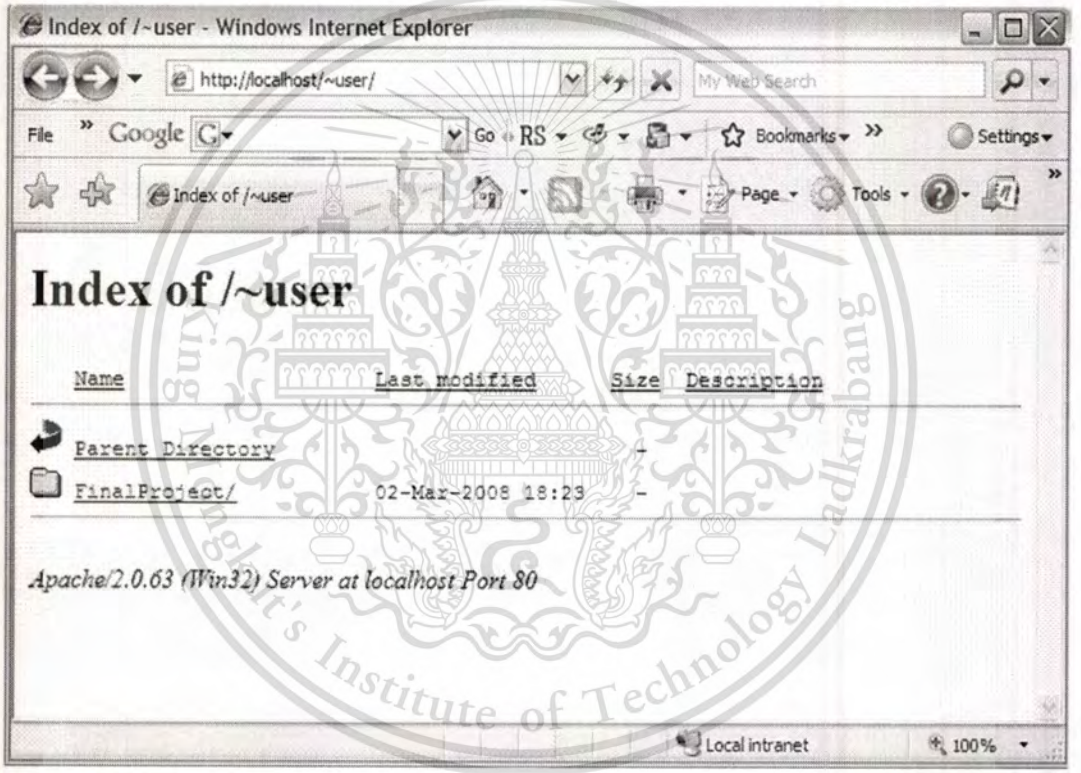


Fig A.21: Example of Index page of the web server

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

A.3: Installing MySQL Server on Windows

1. Download the MySQL installation packages from http://dev.mysql.com/downloads/

2. Extract it; you will get the setup.exe file.

Then, double click 

to start the installation process.

It will launch the installer start screen that shows the name and the version of the program. (Fig A.22). Then, click "Next".

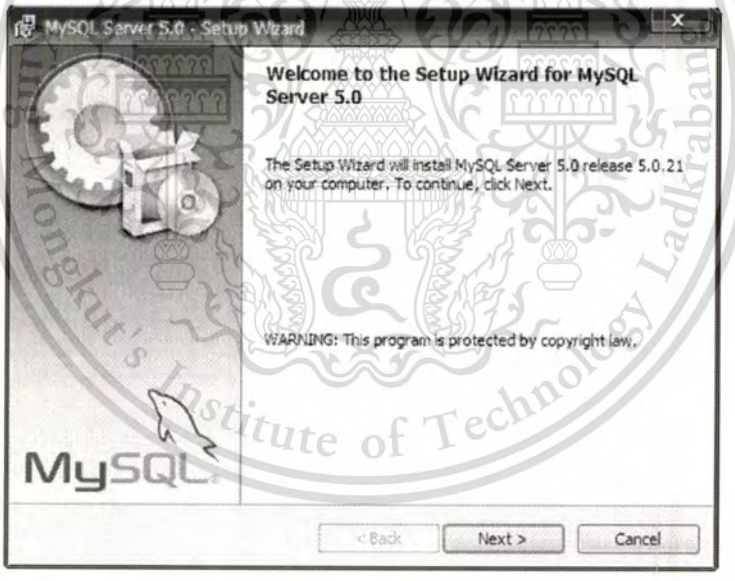


Fig A.22: MySQL Installer Start Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

3. It has three installation types: Typical, Complete and Custom. In our case, "Typical" type is enough. (Fig A.23)

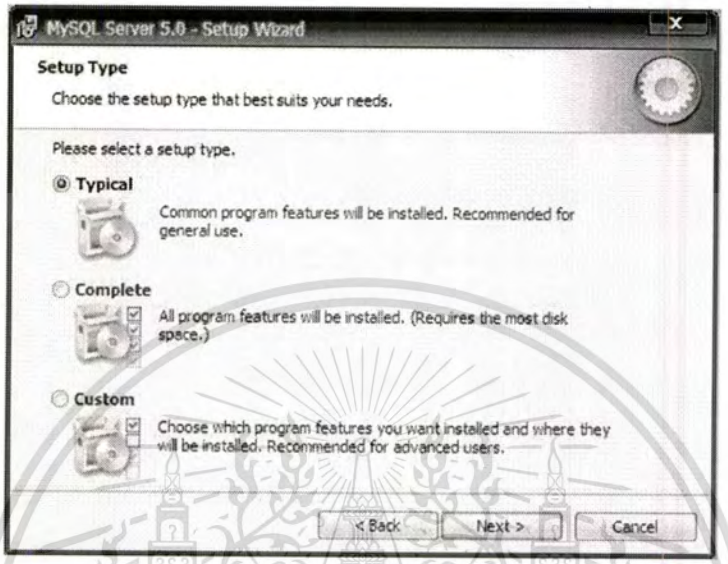


Fig A.23: Installation Types Screen

4. Click "Next" to start installing. (Fig A.24)

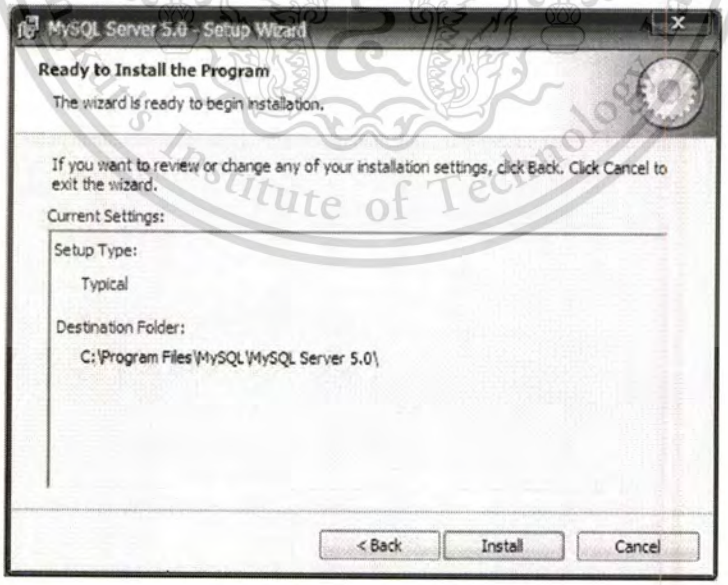


Fig A.24: Confirmation Page

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

5. Now, this page is for create an account on the MySQL.com website. If you would like to do, choose the appropriate option and follow the instructions. Otherwise, choose the Skip Sign-Up option and click "Next". (Fig A.25)

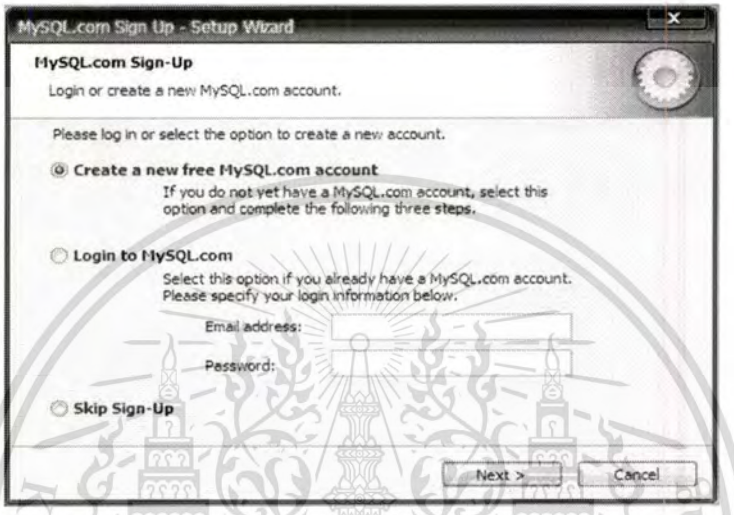


Fig A.25: Sign-Up Screen

6. Now, the MySQL Server was installed. It must be configured, choose the "Configure the MySQL Server now" and click "Finish". (Fig A.26) It will launch MySQL Configuration Wizard (Fig A.27) Click "Next".

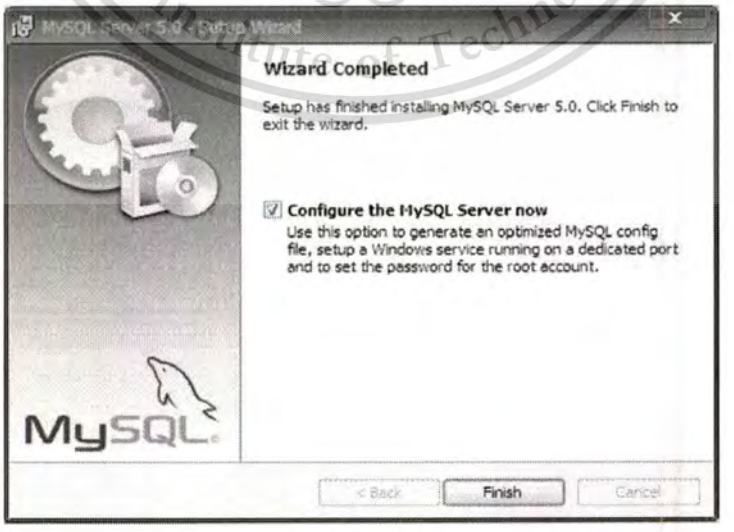


Fig A.26: Installing complete Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

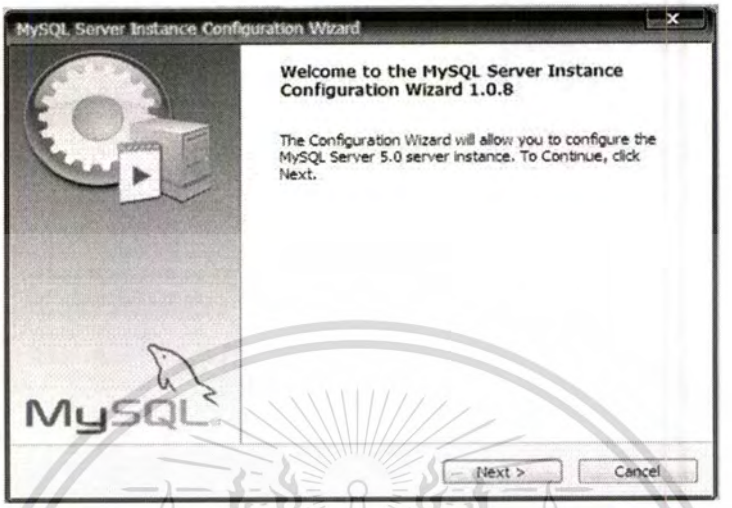


Fig A.27: Configuration Wizard Screen

7. Choose "Detailed Configuration" and click "Next". (Fig A.28)

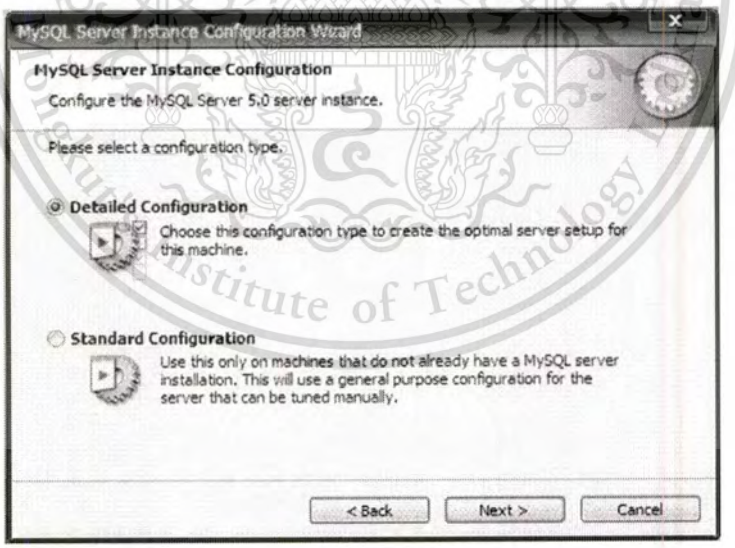


Fig A.28: Configuration Type Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

8. Select the Machine. In our case, we choose “Server Machine” and click “Next”. (Fig A.29)

Fig A.29: Server Type Screen

9. Choose “Multifunctional Database” option and click “Next” (Fig A.30)

Fig A.30: Database Usage Type Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

10. Choose the location of your database, the default is fine. (Fig A.31)

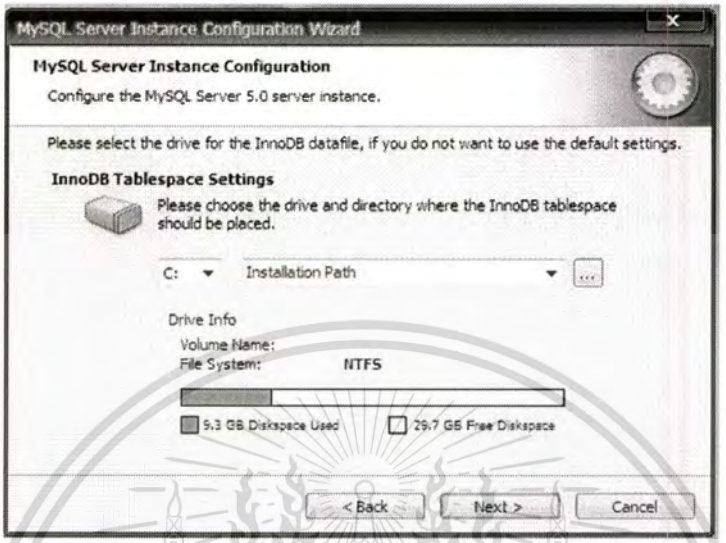


Fig A.31: Set Destination Folder & Drive Info

11. This step allows you to help optimize the database for the number of concurrent connections that you expect to have to a Portfolio catalog. If you are not sure how many concurrent users you will have, choose the "Online Transaction Processing (OLTP)" and click "Next". (Fig A.32)

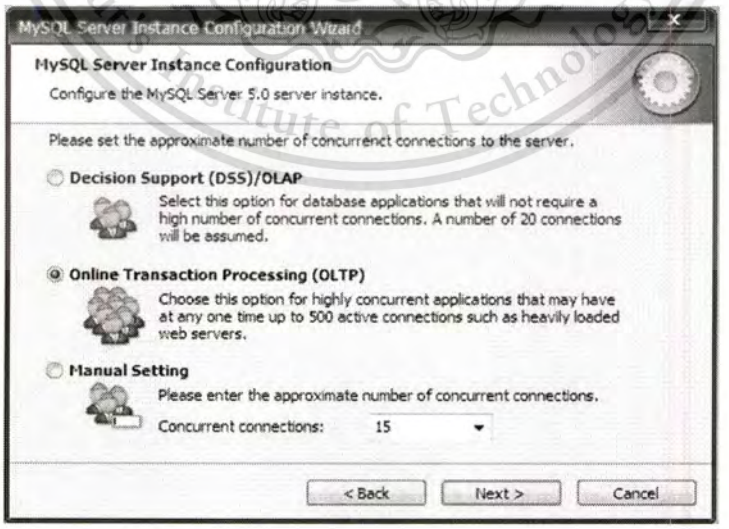


Fig A.32: Set the Connection Type

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

12. Choose “Enable TCP/IP Networking” and click “Next”. (Fig A.33)

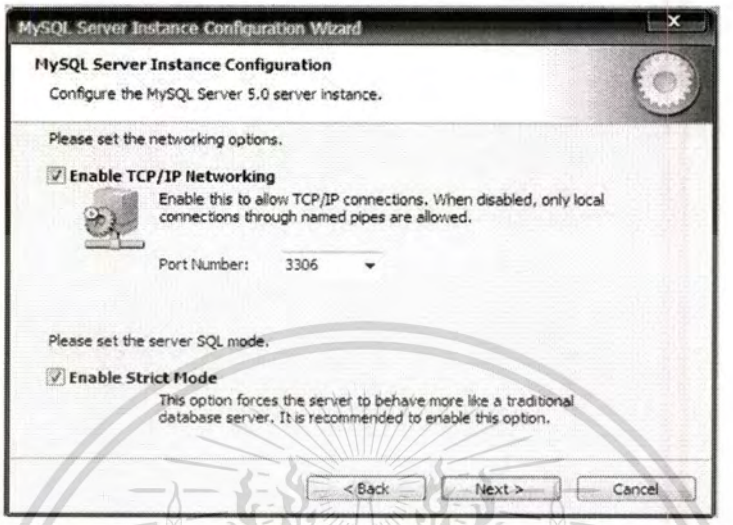


Fig A.33: Networking options Screen

13. Choose “Best Support for Multilingualism”, click “Next” (Fig A.34)

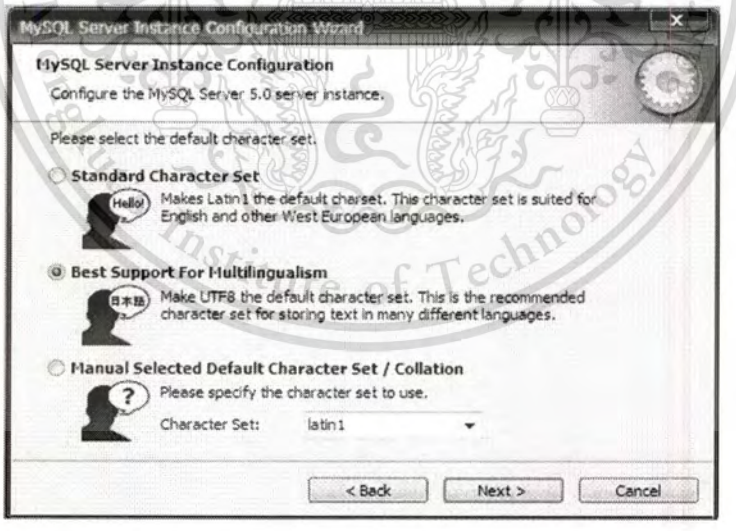


Fig A.34: Set the default character

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

14. We set the Windows options by select both "Install as Windows Service" and "Include Bin Directory in Windows PATH (Fig A.35)

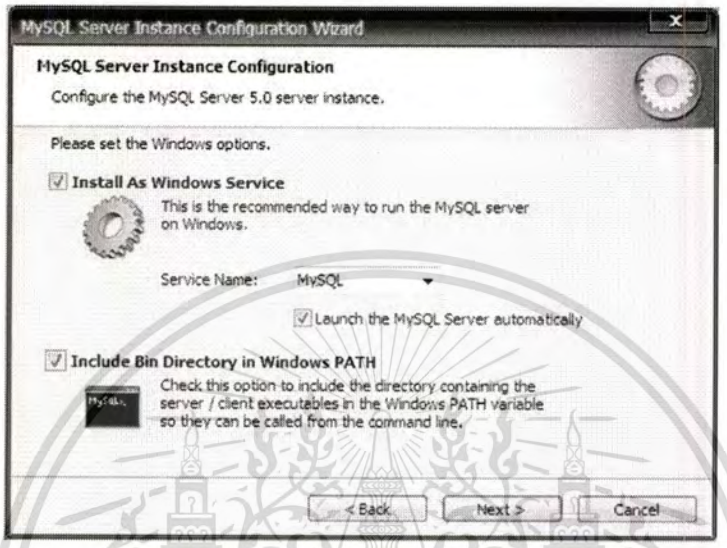


Fig A.35: Set the windows options Screen

15. Check the "Modify Security Settings" option and enter and confirm the new root user password. Then, click "Next". (Fig A.36)

Fig A.36: Security options Screen

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

16. Click “Execute” to start the configuration of MySQL. (Fig A.37)

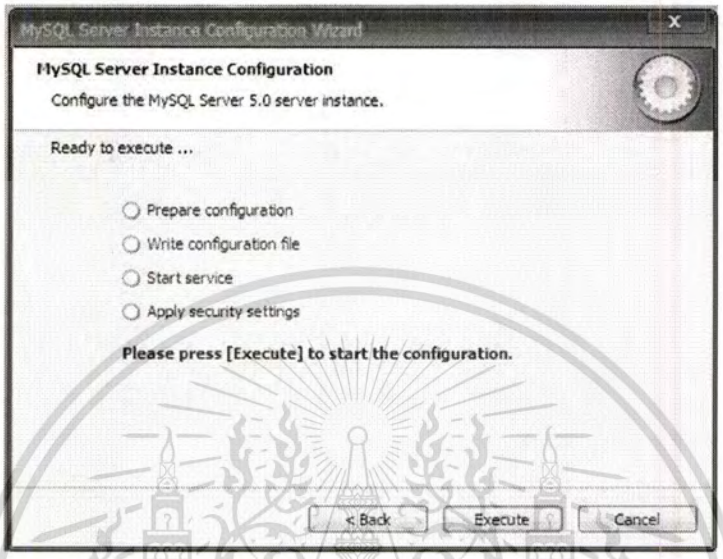


Fig A.37: Confirmation of the configuration

17. Click Finish to close the Wizard

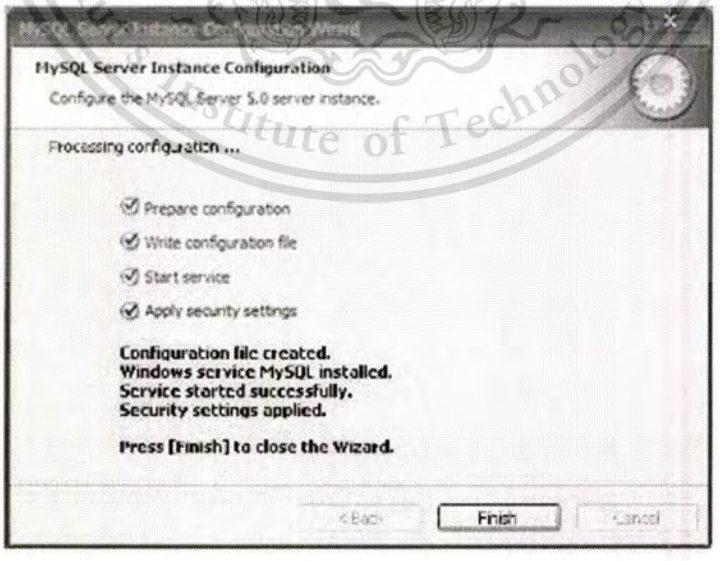


Fig A.38: Complete Configure

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

B: Perl Tutorial

This is a brief Perl tutorial for more understanding our Perl program for example, how Perl connects to database or which modules needs to be use. In the tutorial, we divide into 4 sections, first we discuss about database, then file handing, Perl/CGI, and adjacency matrix. For more about Perl, you can go to ActivePerl User Guide.

B.1 Connecting Perl to Database

PERL is capable of running SQL and MySQL queries including: inserts, selects, updates, deletes, etc through a module termed *DBI*. Therefore the server needs two modules to be install; DBI and DBD::mysql. DBI stands for database interface. Any functions associated with DBI should work with the entire available SQL platform including: SQL Server, Oracle, DB2, and MySQL.

Before continuing, be sure the following modules are installed:

- DBI
- DBD::mysql

Once they are installed, we can build the introduction to our script by telling PERL to *use* these modules as follows:

```
#!/perl/bin/perl

# PERL MODULES WE WILL BE USING
use DBI;
use DBD::mysql
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Again, these modules allow for us to call upon functions specific to working with any database platform. These modules must be in "use" to ensure proper functionality of our scripts.

We will be calling on our database, table, and host machine from time to time. We recommend setting up some variables for your database and table name, so that you can call upon them as you wish throughout this brief tutorial. You may also set up some variables for your user name and password as we will also need to connect to your MySQL.

```
#!/perl/bin/perl

# PERL MODULES WE WILL BE USING
use DBI;
use DBD::mysql

$database = "project";
$hostname = "localhost";
$port = "3306";
$username = "root";
$password = 'kmitl';
```

In order to connect to our database platform we first need to know our web server's *data source name*. This information should be readily accessible in your server's documentation.

```
$dsn = "dbi:SQL_Platform:database_name:host_name:port";
```

Since we plan on executing our scripts from our server through our browser, we can alternatively substitute our host's name with the term *localhost*.

```
$dsn = "dbi:SQL_Platform:database_name:localhost:port";
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Previously, we had set up a *config* script with some information about our web host and SQL platform including a user name and password. We can now plug all those variables into the connection string and connect to our database.

We can establish a connection with a script like the following.

```
#!/perl/bin/perl
use DBI;
use DBD::mysql;

$database = "project";
$hostname = "localhost";
$port = "3306";
$username = "root";
$password = 'kmitl';

$dsn = "DBI:mysql:database=$database;host=$hostname;port=$port";
```

On a side note, we have also created what is known as a *database handle*. Our variable, $dsn, is now the handle which we will have to use each time we wish to execute a query. We should probably go ahead and shorten up that handle since we will be using it in every query script.

```
#!/perl/bin/perl
use DBI;
use DBD::mysql;

$database = "project";
$hostname = "localhost";
$port = "3306";
$username = "root";
$password = 'kmitl';

$dsn = "DBI:mysql:database=$database;host=$hostname;port=$port";

$dbh = DBI->connect($dsn, $username, $password) or die("Could not
connect!");
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

To querying it must be prepared and then executed. A few lines of code are required for this, first the *prepare()* function and then the *execute()* function.

```
#!/usr/bin/perl

# PERL MODULES WE WILL BE USING
use DBI;
use DBD::mysql

# HTTP HEADER
print "Content-type: text/html \n\n";

# CONFIG VARIABLES
$platform = "mysql";
$database = "store";
$host = "localhost";
$port = "3306";
$tablename = "inventory";
$user = "username";
$pw = "password";

# DATA SOURCE NAME
$dsn = "dbi:mysql:$database:localhost:3306";

# PERL DBI CONNECT
$connect = DBI->connect($dsn, $user, $pw);

# PREPARE THE QUERY
$query = "SELECT * FROM area;
$sth = $dbh->prepare($query);

# EXECUTE THE QUERY
$sth->execute;

# LOOP THROUGH RESULTS
while(($col1,$col2) = $sth->fetchrow_array){
   print “Column1 = $col1, Column2 = $col2”;
}
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

B.2: File Handling

Now we shift gears as we introduce file handling. In PERL files are given a name, a handle, basically another way of saying alias. All input and output with files is achieved through file handling. Filehandles are also a means by one program may communicate with another program.

A filehandle is nothing more than a nickname for the files you intend to use in your PERL scripts and programs. A handle is a temporary name assigned to a file. A great filehandle is an abbreviated version of the filename. The example below illustrates how you will use a file handle in your PERL code.

```
#!/usr/bin/perl

use Fcntl;

$FileRoute = "home/html/myhtml.html"
sysopen(HANDLE, $FileRoute, O_RDWR|O_CREAT|O_TRUNC,
0755);
printf HANDLE "Junction-based location query system for restaurants";
close (HANDLE);
```

Files are opened using the *sysopen* function. It requires the declaration of a new module for PERL; *Fcntl* module. The *sysopen* function may be passed up to 4 arguments, the first is always the file handle, then our file name also known as a URL or fileroute, flags, and finally any permissions to be granted to this file.

B.3: Perl /CGI

For using HTML with Perl, rather than use *print* function to type all of HTML code, we use *CGI* module. Let's see how to use a module in your CGI program. First you have to actually include the module via the use command.

```
use CGI qw(:standard);
```

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Note we're not doing use *CGI.pm* but rather use *CGI*. The *.pm* is implied in the use statement. The *qw(:standard)* part of this line indicates that we're importing the "standard" set of functions from *CGI.pm*. The example below shows how we use Perl/CGI.

```
use CGI qw(:standard);
print header;
print start_html("Hello World");
print end_html;
```

After running the script, the HTML code will be as below:

```
Content-Type: text/html; charset=ISO-8859-1
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"
xml:lang="en-US">
<head>
<title>Hello World</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
</head>
<body>

</body>
</html>
```

B.4: Adjacency Graph

Perl has module name Graph to help us deal with any kind of graph data structures. To create adjacency matrix of our system, we use

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

undirected adjacency matrix graph, the program needs to include 2 sub-modules of graph via the use command:

```
use Graph::Undirected;
use Graph::AdjacencyMatrix;
```

This is an example code:

```
use Graph::Undirected;
use Graph::AdjacencyMatrix;

my $g = Graph::Undirected->new;
$g->add_edge($node1, $node2);

$am = Graph::AdjacencyMatrix->new($g);
```

The *add_edge* function adds the edge to the graph. Implicitly first adds the vertices if the graph does not have them. You can use *Graph::AdjacencyMatrix* to compute the adjacency matrix and optionally also the distance matrix of a graph, and after that query the adjacencyness between vertices by using the *is_adjacent()* method

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

C: Related Literatures

C.1: Semantic Information Retrieval in the COMPASS Location System

Author: Frank Kargl, Günter Dannhäuser, Stefan Schlott and Jürgen Nagler-Ihlein

Publication: Lecture Notes in Computer Science Volume 4239/2006 pp. 129-143

Abstract

In our previous work, we have described the COMPASS location system that uses multiple information sources to determine the current position of a node. The raw output of this process is a location in geo-coordinates, which is not suitable for many applications. In this paper we present an extension to COMPASS, the so called Translator, that can provide facts about the location like city name, address, room number, etc. to the application. These facts are represented in the Semantic Web RDF/XML language and stored on distributed Geo RDF Servers. The main focus of this paper is a location-based service discovery mechanism which allows a node to find all services that can provide facts about its current location. This discovery service is built upon a structured Peer-to-Peer system implementing a distributed hash table.

C.2: A Collaborative Framework for Location-Based Services

Author: Shijun Yu, Marie-Aude Aufaure, Nadine Cullot and Stefano Spaccapietra

Publication: In Proc. of the 15th Conference On Advanced Information Systems Engineering, Forum Proceedings of CAiSEÂ'03, Klagenfurt/Velden, Austria, June 16-20 2003

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

Abstract

This short paper is aimed to discuss the challenges for location-based services and proposes our framework, which makes it possible to obtain information from heterogeneous sources, and further set up the collaboration between Data Repositories and derived Top Hits Repository to improve the request-response efficiency. In our framework, the Data Handler, Profile Manager, Data Repository and TOP Hits Repository are key components. Through analyzing user profiles and location, Data Handler can locate suitable data sources and keep frequent queries and their answers in TOP Hits Repository for later requests.

C.3: Location-Based Spatial Modeling Using Ontology

Author: Shijun Yu, Marie-Aude Aufaure, Nadine Cullot and Stefano Spaccapietra

Publication: In Proceedings of the 6th AGILE on Geographic Information Science - "The Science behind the Infrastructure", Lyon, France, April 24-26 2003

Abstract

Location-Based Spatial Modelling Using Ontology Providing location-based service is an increasingly demanding challenge in our free and convenient e-commerce era. To model and fully use the existing information is a necessary prerequisite to better services. We have described issues involved in supporting an ontology-based information searching approach. We presented our architecture of mobile information integration and the approach of navigating between the branches and nodes in the ontology tree. Building the mappings between ontologies and between the heterogeneous data sources and making precise inference reasoning regardless of the data sources appear as the major issues.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.

C.4: Location- and Time-Based Information Delivery in Tourism

Author: Annika Hinze and Agnés Voisard

Publication: Advances in Spatial and Temporal Databases, 8th International Symposium, LNCS 2750, Santorini Island, Greece, 2003

Abstract

Today's mobile devices allow end users to get information related to a particular domain based on their current location, such as the fastest route to the nearest drugstore. However, in such Location-Based Services (LBS), richer and more targeted information is desirable. In many applications, end users would like to be notified about relevant events or places to visit in the near future according to their profile. They also do not wish to get the same information many times unless they explicitly ask for it. In this paper, we describe our system, TIP (Tourism Information Provider), which delivers various types of information to mobile devices based on location, time, profile of end users, and their "history", i.e., their accumulated knowledge. The system hinges on a hierarchical semantic geospatial model as well as on an Event Notification System (ENS).

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use.