Malaysian Journal of Science **28** (2): 113– 125 (2009)

Communication Architecture for Biodiversity Information Retrieval (CABIR)

Sarinder K. K. S.[1*], Lim L. H. S.[1], Dimyati K.[2] and Merican A. F.[1]

[1] Institute of Biological Sciences, Faculty of Science, [2] Department of Electronic and Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia
* sarinder@um.edu.my (corresponding author)
Received 11th September 2008, accepted in revised form 14th July 2009.

ABSTRACT This paper focused on biodiversity database integration. Firstly, the problems are studied and explained. They are divided into four broad categories, which are technical, political, syntactical and semantical. Following this, the approaches to build a database integration system were reviewed after which, the mediator based approach was selected. A new database integration system, which is called CABIR (Communication Architecture for Biodiversity Information Retrieval) was built and discussed thoroughly in this paper. A performance test was also conducted to measure the efficiency of the CABIR system. The development of CABIR system is expected to contribute in the field of biodiversity as this system could link distributed biodiversity databases regardless of the heterogeneity and the platform they were built upon.

ABSTRAK Artikel ini memberi tumpuan kepada integrasi pangkalan data biodiversiti. Pertama sekali, masalah-masalah yang dihadapi dikenalpasti and dijelaskan. Masalah-masalah ini dibahagikan kepada empat kategori iaitu teknikal, politikal, sintetikal and semantikal. Seterusnya, langkah-langkah untuk membangunkan satu integrasi pangkalan data dikaji setelah "mediator" telah dipilih. Satu sistem integrasi pangkalan data yang baru, iaitu CABIR (Communication Architecture for Biodiversity Information Retrieval) telah dibangunakan and dibincang dengan teliti dalam artikel ini. Satu ujian prestasi telah dijalankan untuk menentukan kecekapan sistem CABIR. Pembangunan sistem CABIR ini dijangka akan menyumbang kepada bidang biodiversiti kerana sistem ini boleh menghubungkan pangkalan data biodiversiti yang bertaburan tanpa mengira kepelbagaian and platform dimana pangkalan-pangkalan data tersebut dibina.

(**Keywords:** Biodiversity, database integration, heterogeneous)

INTRODUCTION

The revolutions caused by advanced computing power, advanced informatics and the Internet have changed the way biodiversity data are stored, located and disseminated. As the internet provides a unified transport infrastructure for information sharing, today almost every scientist uses the internet to share data, sometimes just with a close colleague, other times through large-scale databases. The volume, complexity and heterogeneity of biodiversity data make data sharing a daunting task. According to an article by Philippi [1], the problems of data integration in life sciences faces a variety of problems and can be grouped into four categories: technical, political, syntactical and semantical.

In a syntactical point of view, flatfiles are still the de-facto standard for exchange of data whereas technical problems arise if a database is not available as a flatfile, but via dynamically generated HTML pages [1]. However, this paper specifically addresses relational databases systems rather than flatfiles and HTML pages, therefore syntactical and technical problems are beyond the scope of this research.

Other problems are of political and semantical. Political reasons such as copyright and ownership issues affect the way data is stored, disseminated and shared among the contemporaries. While data warehouses are easier to manage and manipulate due to their homogeneity, they are not very well accepted by the scientific community. This is due to the political reason that data goes out of the researcher's storage to a central warehouse managed by an administrator, who might be unknown to him. In addition, data integrity becomes a problem as the warehouse administrator might not be an expert in the scientific field. In addition, the data is not longer protected by the owner (researcher). Due to these reasons, generally, warehouse integration is no longer the best accepted approach. On the other hand, gigantic amount of biodiversity data stored in dispersed warehouses creates isolated platforms for data retrieval using search engines. These distributed databases are structured in heterogeneous formats and are not integrated, hence making sharing a difficult task for scientists. In addition, the same data is always described by different terms amongst data-bases (e.g. [2] and [3]). This is considered a semantic problem. Semantic conflicts manifest themselves on database schema level which is caused by different names of equivalent database fields and on entry level where different names are used for the same things [4]. Therefore standardization is needed to overcome semantic heterogeneity. According to Xia [5], standardization in this context implies two issues which are standard terminology and nomenclature and standard format for data submission and storage, exchange and query. Much standardization have been developed or proposed in biology, such as standardization of Gene Nomenclature [6], the Access to Biological Collections Data (ABCD) Schema [7], Taxon Concept Schema (TCS) [8] and Dublin Core [9]. However, the Darwin Core (DwC) standard defines a simpler set of fields common to all taxonomic groups. The result is a simple XML schema with a small number of elements covering basic and essential information [10].

Common approaches for heterogeneous database integration for information retrieval are mediator based [11, 12] and data warehousing [13]. We rule out data warehousing due to the political reason described earlier. Mediator based integration approach uses a module called "mediator" which accepts a query from the user, determine the data sources needed to answer the query. Queries are translated to the source-specific query language via modules called "wrappers". The results from the sources are translated back into the common query language by the wrappers. Finally the mediator obtains results from the wrappers, integrates them and returns the final answer to user [14]. Using the mediator architecture, source databases transparently deliver up to date results to a client [11]. The mediator based approach can overcome warehousing problem described above. Using this approach, data sources do not need to be integrated into a common warehouse but kept in distributed warehouses. Mediators are used to integrate these sources using the architecture described above. However, integrating data-bases is not a straightforward task. The mediator based approach cannot solve all problems related to database integration. This approach do not account for issues such as semantic heterogeneity, level of sharing, DBMS heterogeneity, platform independence, interface presentation and efficiency. In order to address these issues, additional effort to database integration is needed. Therefore this paper describes a new approach to database integration that address issues described above.

PRIOR WORK

Prior to building a new database integration system in this research, a preliminary study on an existing system, Distributed Generic Information Retrieval (DiGIR) was conducted. This study was carried out in two phases described below.

First Phase Implementation

The DiGIR protocol specification is an XML-Schema document that defines the structure of messages sent to, and returned by a DiGIR provider. It provides a single point of access (portal) to distributed information resources (MIMODS). It enables search and retrieval of structured data and makes location and technical characteristics of native resource transparent to users. DiGIR (Distributed Generic Information Retrieval) protocol was used as the communication protocol for queries between distributed data-bases. As for the data sharing/exchange format, the Darwin Core (DwC) schema was used.

Implementation of DiGIR using an Algae database

The Distributed Generic Information Retrieval (DiGIR) Provider has been installed and tested using an Algae data-base in the bioinformatics server physically located at University of Malaya. The Algae database is part of the Malaysian Microbial Online Database (MIMODS) [21]. The procedure of installing a DIGIR provider can be briefly outlined in a few steps.

1. Installation of PHP on the web server
2. Installation of DiGIR provider distribution
3. Editing of localconfig.php to reflect local installation choices
4. Editing of providerMeta.xml to present appropriate metadata about the installation.

5. Creating the database configuration file and saving it in the configuration folder
6. Updating resources.xml to point to the new configuration file.

An Adodb database connectivity was used to establish a connection with the MS Access test database. The installation showed that DiGIR Provider could run under the following specifications.

The installation in Figure 1 was based on the Bioinformatics server system configuration in University of Malaya (http://www.bioinformatik.um.edu.my). The MIMODS databases were stored in this server and a DiGIR provider is being developed to serve these databases. A scan was done to check the Algae database contents. The scan was invoked using the "scan" operation parameter. The results in will be hence extracted by the DiGIR portal, which should be installed in a remote machine.

A simple search operation was also run to check whether it extracts the correct data from the database. This simple search looks for the string "otus%" in attribute #1 which maps to "darwin:ScientificName?". The query is displayed in the following URL: http://127.0.0.1/DiGIR/DiGIR.php?operation=search&resource=test&filter=@attr+1=1+%22otus%25%22. An XML format results were generated.

Installation of a simple portal on a remote machine
There was a problem installing the DiGIR portal to search these databases using the interface. The installation of DiGIR's presentation layer was deemed to be unsuccessful in the context of this project and due to time limitation; we decided to develop our own portal.

A simple portal was designed to search the MIMODS provider remotely. The portal was installed on a remote machine, which is running on a Linux platform. The portal interface was designed using Ruby-programming language. By entering an algae family name, the meta search engine can extract relevant data from MIMODS local algae database.

The results of the search are displayed in a tabular form. The search option is currently limited to microbial family name in the algae database in MIMODS.

SECOND PHASE IMPLEMENTATION

In the second phase, DiGIR was implemented on five existing databases; Fungi, Algae, Protozoa, Virus and Bacteria which were developed using Microsoft Access. These data-bases comprise the Malaysian Indigenous Microbial Online Database System (MIMODS).

DiGIR Software Setup and Configuration
Both layers, the portal and presentation layer had their own Web applications, therefore two archives were downloaded from the DiGIR website [22]. For the portal application, the DiGIR_portal_engine.war was placed in the directory %DIGIR_HOME%/tcinstance/engine/Webapps. This file was unpacked using the jar xvf DiGIR_portal_engine.war command. This action created a WEB-INF directory containing a directory of Java classes and a directory of libraries.

Navigation showed packages org.calacademy.digir.common, org.calacademy.digir.engine and org.calacademy.digir.util were installed. Finally, the .war file was removed from the Webapps directory as it may interfere with the running Web application.

As for the presentation application, the presentation layer archive, DIGIR_portal_pres.war was downloaded and placed in the directory %DiGIR_HOME%tcinstance/pres/Webapps.

This file was unpacked using the command jar xvf DiGIR_Portal_pres.war. Hence, WEB-INF directory was created containing a directory of java classes and a directory of libraries. Java packages org.calacademy.digir.common, org.calacademy.digir.presentation and org.calacademy.digir.util were also installed. Like the portal setup, the .war file was also removed in this layer.

The engine configuration involved editing the portal.xml file located in the %DiGIR_HOME/tcinstance/engine/Webapps/WEB-INF/classes/org/calacademy/digir/engine directory. The configuration of the portal.xml is displayed in Table 1.

For the presentation layer, the presentation.xml file was con-figured. The configuration of the presentation.xml file is displayed in the Table 2.

In summary, the implementation of DiGIR comprised of the engine and presentation layer. The two most important files are essentially the portal.xml and presentation.xml file.

Provider Setup

The DiGIR provider hides the details of the underlying data-base. Therefore the provider file had to reside in the hosting machine of the database. The DiGIR.php file is the primary mechanism for invoking the DiGIR provider.

The Algae database is used here to provide an example for the provider setup. Algae database is one of the databases in the Malaysian Indigenous Microbial Online Database Systems (MIMODS). This Algae database is represented using an XML file, which acts as the wrapper between the portal and the back end database. This XML wrapper is part of the provider, besides the database itself. XML file for the Algae database (digir_algae.xml) is presented in Table 3 Before this file was configured, the Algae database had to be altered as the DiGIR system required five fields to be present in the database for it to be eligible as a DiGIR provider. The five fields are Date Last Modified, Institution Code, Collection Code, Catalogue Number and Scientific Name. The provider setup completes the implementation of DiGIR.

Running the DiGIR Portal

In order to see the results, the engine and presentation layers of DiGIR must be started. The steps performed are described in (a) and (b).

Engine Startup

The commands (1) and (2) were executed at the command prompt to run the engine;

```
cd %DIGIR_HOME%/tcinstance/engine/bin   (1)
startup_engine                          (2)
```

Next, the URL (3) was typed in the Web browser to test whether the engine is running.

```
http://YOUR.IP.OR.DOMAIN:8080/portal/Port
alServlet?action=getProviders           (3)
```

Presentation Startup

Following the start of the DiGIR engine, the presentation was started with the commands (4) and (5).

```
cd %DIGIR_HOME%/tcinstance/pres/bin     (4)
startup_pres                            (5)
```

Next, the URL (6) was typed in the Web browser to test whether the presentation is running.

```
http://YOUR.IP.OR.DOMAIN:10080/pres/Pres
entationServlet?action=home             (6)
```

Table 1. Portal.xml file configuration: The coding in bold is important for inserting a provider into the portal.

```
<?xml version="1.0" encoding="UTF-8"?>
<configxmlns=http://www.calacademy.org/portal
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.calacademy.org/portal portal.xsd">
<version>0.91</version>
<logfileName>/DiGIR/logs/portal_services.log</logfileName>
<providerFilter>org.calacademy.digir.engine.Darwin2ProviderFilterer</providerFilter>
<handlerPoolThreadMax>10</handlerPoolThreadMax>
<registry>
<uddiInquiryURL>http://uddi.microsoft.com/inquire</uddiInquiryURL>
<uddiServiceKey>UUID:4DFAB7E8-6387-431D-BC20-6291E99A51A8</uddiServiceKey>
</registry>
<providerCacheTimeout>0</providerCacheTimeout>
<provider>
<name>MIMODS Provider</name>
<accessPoint>http://202.185.72.215/digir/DiGIR.php</accessPoint>
</provider
```

Table 2. Presentation.xml file configuration. This file signifies the user interface. The text in bold were inserted to imply a University Malaya test portal

```
<display>
<color>#663366</color>
<title>DiGIR University Malaya BioWeb</title>
<image>digir.gif</image>
<link>http://digir.sourceforge.net</link>
<intro><p>This is a generic DiGIR for University Malaya test portal.</p></intro>
<about><p>This is a generic DiGIR for University Malaya test portal.</p></about>
<footer>This is the footer for the generic DiGIR test portal. More information about DiGIR is available
on the <a href="http://digir.sourceforge.net">DiGIR SourceForge Site</a></footer>
</display>
```

Table 3. Part of the Digir_algae.xml file (This file contains important information regarding the database)

```
......................
<concept searchable="1" returnable="1" name="darwin:InstitutionCode" type="text" table="algae" zid="10"
field="institutioncode" />
<concept searchable="1" returnable="1" name="darwin:CollectionCode" type="text" table="algae" zid="11"
field="collectioncode" />
<concept searchable="1" returnable="1" name="darwin:CatalogNumber" type="text" table="algae" zid="12"
field="catalognumber" />
<concept searchable="1" returnable="1" name="darwin:ScientificName" type="text" table="algae" zid="11"
field="scientificname" />
<concept searchable="1" returnable="1" name="darwin:Family" type="text" table="algae" zid="1"
field="family" />
<concept searchable="1" returnable="1" name="darwin:Genus" type="text" table="algae" zid="1"
field="genus" />
<concept searchable="1" returnable="1" name="darwin:Species" type="text" table="algae" zid="2"
field="species" />
</concepts>
.....................
...................
...............
```

DiGIR Implementation Results

Following the startup processes described above, the DiGIR application was run to examine its results. However, the results showed unexpected outcome. The unsatisfactory results are presented in the following sub-sections.

Main Page

The DiGIR mainpage was presented on URL (7).
http://localhost:10080/PresentationServlet?action =home (7).
The build query button takes the user to the query form.

Query Form

Here the providers and search conditions were displayed. The highlighted providers are the MIMODS databases (Malaysian Indigenous Algae, Malaysian Indigenous Bacteria and Malaysian Indigenous Fungi). The conditions of query must be selected in order to see the results.

Query Results

Although the configuration to insert MIMODS providers was successful and the providers were displayed in the query form, the queries on the MIMODS providers were unsuccessful. No results were produced from the query. Despite many attempts, the solution could not be found.

Other Features

The DiGIR system provides feedback to users in case of any mistake done. For example if a user does not select any provider, an error text message will appear informing the user that there was no provider selected.

The results did not show any data retrieved from the MIMODS database. This means there was no connection established with those databases. The problem could not be resolved as the system appeared to be correct. Despite the many attempts to solve the problem using different workstations, the problem still persisted. Therefore, the test on MIMODS databases was not successful.

The failure of DiGIR implementation has lead to the development of a new system in this research. The system developed in this study which is named CABIR, used DiGIR as a model but it is much simpler compared to DiGIR. The architecture is presented in the following section.

THE ARCHITECTURE

CABIR is designed to furnish the needs of a scientist in terms of looking for various specimens in various repositories using one stop search engine or web portal. Therefore, the underlying design required the use of integrating technologies, query processing and retrieval capabilities which produces a system with the following features:

- Links remote databases on networked environment
- Supports heterogeneous data format
- Supports heterogeneous database management systems (DBMS)
- Links databases hosted in Windows and UNIX based platforms
- Provides data security for database owners by allowing them to keep and maintain their own data and to choose information to be shared and linked
- Simple user interface
- Efficient in terms of information retrieval speed

The architecture in Figure 1 shows the essential components which are Resource Locator, XML wrapper, ODBC, Filtering and Data Cleaning and the Presentation Layer which consists of the Search Engine and the Query Results.

Based on the user's selection, the resource locator will look up for the resources which are web biodiversity data sources connected using CABIR. At this point users can select more than one database and CABIR will execute the processing simultaneously to these repositories. Once the

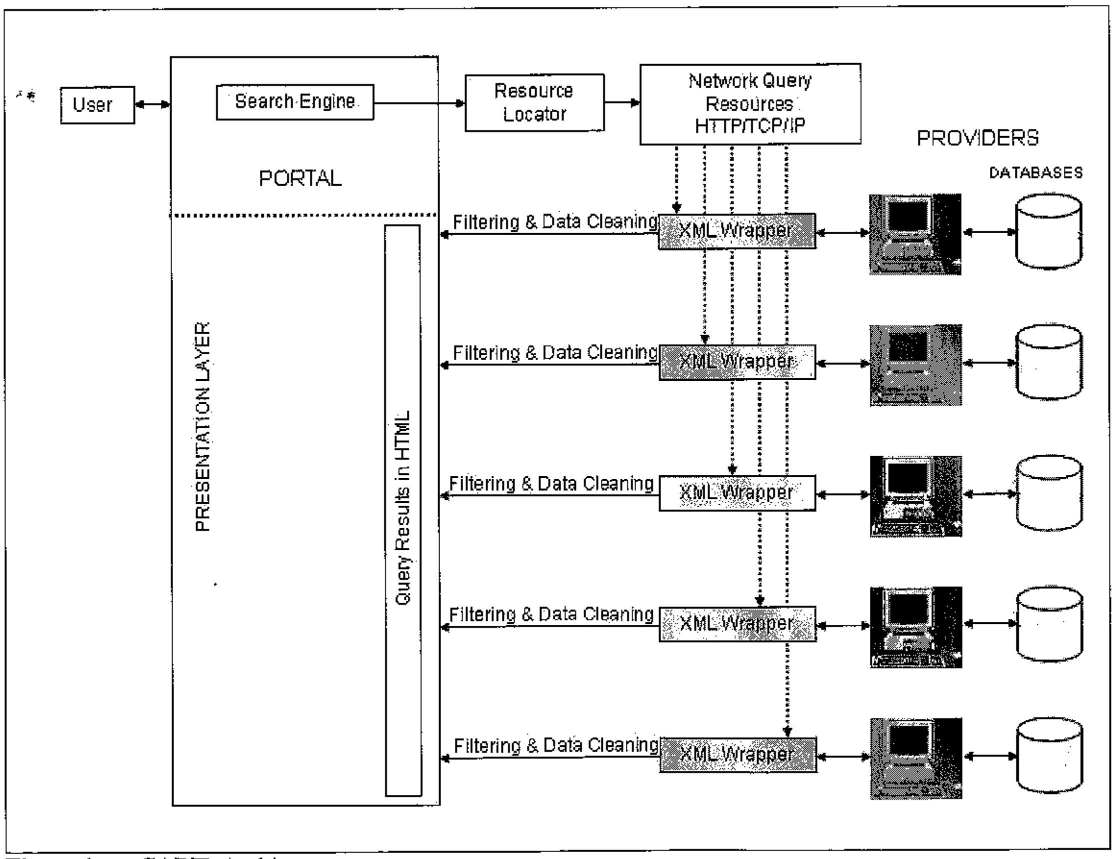

User
Search Engine
PORTAL
Resource Locator
Network Query Resources HTTP/TCP/IP
PROVIDERS
DATABASES
PRESENTATION LAYER
Query Results in HTML
Filtering & Data Cleaning
XML Wrapper
Filtering & Data Cleaning
XML Wrapper
Filtering & Data Cleaning
XML Wrapper
Filtering & Data Cleaning
XML Wrapper
Filtering & Data Cleaning
XML Wrapper

Figure 1. CABIR Architecture

resource locator has identified the web data sources, query will be sent via the TCP/IP protocol using the uniform resource locator (URL) to retrieve all the necessary data. XML wrappers contains provider information document, xml schema and xml documents containing query results. The provider in-formation document consists of the database connectivity details, namespace and query statements. This document will be installed at the client side where the database resides. The xml schema will map the query results into a well formed data structure which applies the Darwin Core V2 global standard [15]. Once the results are produced in the XML document, the data goes through a clean-up phase to meet the user's requirements. At this point, surplus data and empty fields are filtered out. Thus, data is sent to the presentation layer, which is then converted into HTML for viewing.

Semantic Heterogeneity

Retrieving data from several heterogeneous databases is not straightforward, because each of them has its own schema and mapping the schema to the CABIR schema is a crucial task. The schema in CABIR was built using XML according to the data standard presented by Darwin Core v2, which is acknowledged globally. The Darwin Core attempts to provide a set guideline for addressing this commonality regardless of the underlying mechanism for storing the record content. The Darwin Core profile pro-vides a list of suggested access points and recommendations for their usage for searching natural history specimen and observation databases. It provides suggestions for queries such that they are protocol independent. It also provides guidance as to the content, structure and format of records retrieved from an information server supporting the Darwin Core [15]. The goal of CABIR is to provide users with a system which supports a mechanism to access heterogeneous biological databases regardless of the schema they adopt. Using the CABIR system, users do not need to understand the details of schemas or structures of target databases, nor do they need a special knowledge about operation of XML or a system. They can issue complex demands by same interface as existing web services.

Level of Sharing

As providers of CABIR, databases owned by scientists are secured. A database owner can choose the data fields to be shared with users in the public domain. This can be achieved by configuring the provider file. In this way, they are able to protect their data with different levels of sharing. Basically, CABIR brings forth interested users to share data while protecting confidential information.

DBMS Heterogeneity

CABIR is implemented using web scripting languages and structured query languages. The databases are connected via Open DataBase connectivity (ODBC) which is a standard for most database connections. In the research done, CABIR was tested on FileMaker, MySQL, DB2, Oracle and Microsoft Access. CABIR uses the ODBC database connectivity; therefore it can connect to FileMaker databases [16]. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be ODBC-compliant that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them.

Interface Presentation

The presentation layer in CABIR architecture communicates with the user. Therefore, it was designed vigilantly to meet the user's expectations of a friendly interface. It has two main components which are the search engine and query results. Users key in the query via the search engine using simple terms. The search engine thus uses a smart query agent to manipulate the query to make the search more extensive and robust. This generates a larger set of results which will then be filtered based on users view. The presentation layer basically sends query and receives results.

System Reliability

XML and ASP are two main languages used to build CABIR. It is a very efficient and a powerful combination. XML is a simple and a powerful tool for Web developers. It was created to handle complex Web documents. It also allows programmer to define all his own tags with rules such as data description and data relationships. XML is used in order to remove the cumbersome problems that were faced with HTML. Information can be accessed easier using XML. ASP and XML are powerful tools for creating dynamic Web pages (see http://www.xml-training-guide.com/asp-xml.html). ASP uses XML as a tool in its application for a simple reason that data in HTML is allowed to transmit between dissimilar platforms, whereas XML allows one to express complex structure. Moreover it allows the programmer to create his own tags with all sorts of rules. A new document instance can be created using MSXML. There are a number of ways to access XML data from an ASP page. Document Object Model (DOM) plays an important role in retrieving the XML data from ASP. In DOM a document is viewed as a tree of nodes. Every node of the tree can be accessed randomly. The main advantage is that it provides all functions in an object based way. An XML parser based on DOM from Microsoft is MSXML. This component is used for accessing the XML documents. These tools make CABIR an efficient application for database integration.

PROTOTYPE IMPLEMENTATION

The first step towards building a new database integration system was to identify the integration approach which excellently suites the requirements of the database integration system. This was followed by selection of relational biodiversity databases which are the materials to test the system. In the final step, the database integration system was built. It involved integrating the databases, writing the xml wrapper and developing the query based portal.

Selection of Integration Approach

Mediator based approach was used in this paper to implement CABIR. This is because the aim of CABIR to integrate remote and distributed databases matches the attributes of mediator based integration approach. Mediator based approach is also one of the characteristics of many of the current database integration systems.

Relational Biodiversity Databases for CABIR

The selection of biodiversity databases was based the DBMS employed to build them. It was necessary to test that CABIR could link databases on diverse DBMS mainly mySQL, Microsoft Access, FileMaker and DB2. Therefore, biodiversity databases built using these systems were selected. It was also necessary to test that the host of databases can be UNIX platforms or Windows platforms; therefore biodiversity databases residing on both the plat-forms were selected. The prototype was built to integrate genuine biodiversity databases built at Institute of Bio-logical Sciences, University Malaya.

Database Integration (Providers)

The process of building a database integration system initially necessitates integration of databases with the Web. This process required a Web server, application program and connectivity. In this research, Apache and Internet Information Service (IIS) were employed as Web servers for the databases to be Web accessible. ASP and PHP were chosen as scripting languages to integrate database with the Web. ASP was used for Windows-based providers whereas PHP for UNIX-based providers. As for the database connectivity, ODBC, DataDirect32-BIT SequeLink 5.4 and Oledb were the driver managers used (see Figure 2). Two methods were used for database connection in CABIR. They are DSN (Data Source Name) connection and DSN-less connection, depending on type of DBMS used. Structured Query Language (SQL) was used to retrieve and manipulate the data from the relational databases. It was also applied to relate the tables using the JOIN command.

Each database in the proposed system required a provider which connects the database to the Web service. The provider also contained the SQL strings to perform the desired query to the database according to the request from the wrappers. The data extracted from the database was returned as XML schema and XML document. Figure 3 shows the architectural view of the database integration process for one database. The application sends a query to the Web server, through the Internet. The query is then forwarded to the specific provider. The provider returns results in XML format which is then converted into read-able HTML format. The Web server contains a connection with the back-end database. The document that contains the ASP/PHP script and database connection is called a Provider. In this research, heterogeneous databases are used and they are queried simultaneously.

XML Wrapper

For each database integrated through the proposed system the provider connects to the database and manipulates the data using the SQL command. The results from the database were converted into an XML format and saved as XML documents. The wrapper basically contains the XML documents which are results of the search. Part of XML wrapper is presented in Figure 4.

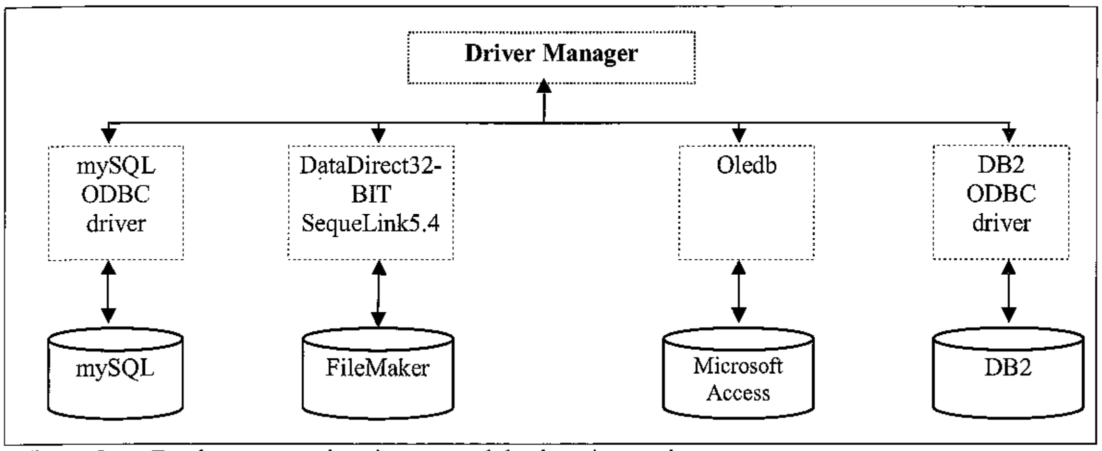


Figure 2. Database connections in proposed database integration

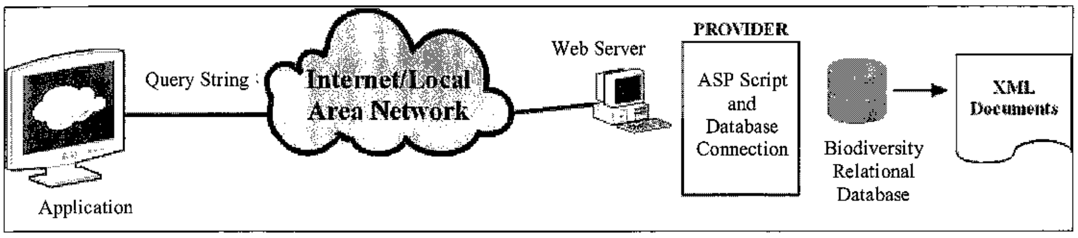

Application
Query String
Internet/Local Area Network
Web Server
PROVIDER
ASP Script and Database Connection
Biodiversity Relational Database
XML Documents

Figure 3. Architectural view of database integration proposed in this research (In this diagram, only one database is used)

```
<DateLastModified><% = strDateL %></DateLastModified>
<CollectionCode><% = strCC %></CollectionCode>
<InstitutionCode><% = strIC %></InstitutionCode>
<CatalogNumber><% = strCN %></CatalogNumber>
<ScientificName><% = strSN %></ScientificName>
<Family><% = strFamily %></Family>
<Genus><% = strGenus %></Genus>
<Morphology><% = strMorphology %></Morphology>
<Colour><% = strColour %></Colour>
<Advantages><% = strAdvantages %></Advantages>
<Characteristics>..................
```

Figure 4. Part of XML wrapper.

Development of Query Based Portal
The subsequent stage of building the database integration system was the development of application server to interact with the users. The application server is called query-based portal. There are two levels forming the query-based portal. They are the user interface and application engine (Figure 5).

User Interface
The interface of the query-based portal was developed using Hypertext Markup Language (HTML), VBScript, JavaScript and Active server pages. It contained a form with a search textbox and options to select the data providers (databases). The database integration system is a

component of the Living Web portal [17] which also contains link to all the databases under the Integrated Biological Database Initiative (IBDI) [18]. The user interface for the database integration system was developed using simple codes to meet the requirements of the system.

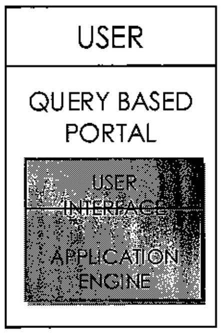


Figure 5. Architectural view of query-based portal

Application Engine

In CABIR, the application engine is responsible for communicating with the providers, using the HTTP GET protocol (using the library in article [19]). The engine was implemented using ASP scripting language. The main duty of the application engine is to act as a resource locator for sending the query to the exact destination. The engine takes the query string supplied by user and constructs a uniform resource locator (URL) for the corresponding database. The engine is also responsible for handling the response produced by the provider in the XML wrapper. This is done by converting the results in the wrapper into HTML using the XSLT (Extensible Stylesheet Language Transformations). XSLT is a trans-formation language for converting XML instances. In simpler words, the application engine here performs the intermediary duties between the user interface and the providers. An instance of the HTTP GET method is pre-sented in Figure 6.

```
Xml.Open "GET", "UniformResourceLoca-
tor?item="&query&"&resource="
&variable&"&", False
```

Figure 6. HTTP GET method

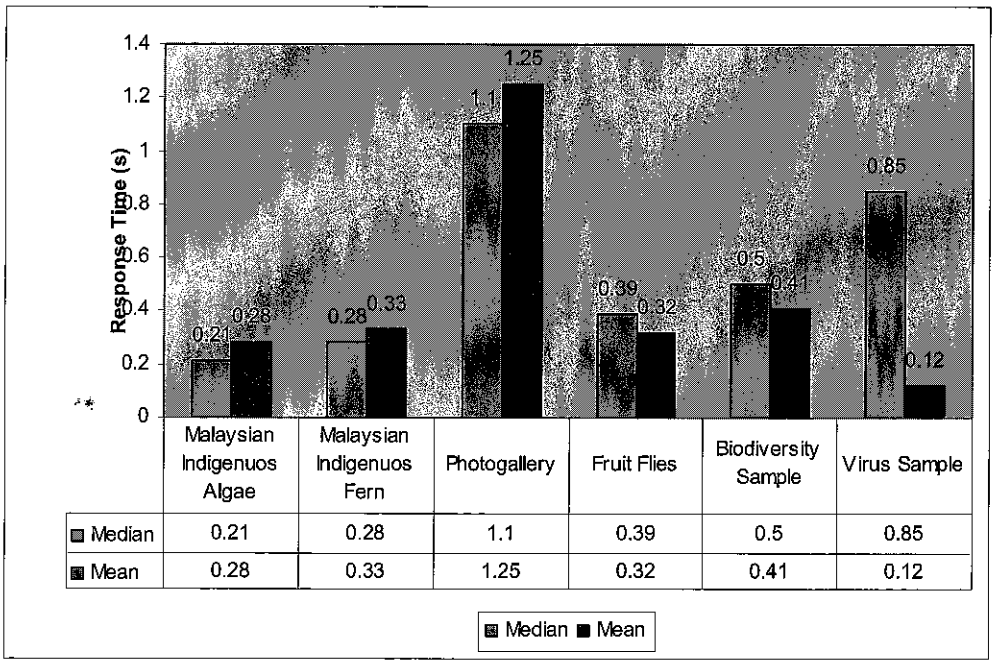
Due to the powerful programming tool and techniques, the underlying details of CABIR were concealed from users. Users do not need to know where the databases reside, the structure of the system, Database Management System used to build databases and the programming behind the system. The information retrieval is similar to a system accessing a local database. This is to ensure its simplicity. CABIR's user interface is made simple and straightforward. A user does not need much effort in understanding the functionality of the system. Thus it will suit a wide spectrum of users. Every page is designed to be as simple and as compact as possible. These pages load in a fast response time to ensure that users do not need wait long to view the pages. CABIR system is also designed to support search in multiple heterogeneous data-bases simultaneously. Heterogeneous here means the databases are developed using a variety of Database Management Systems and in heterogeneous data format CABIR also works for providers in UNIX based systems. It has been tested with a mySQL database in Solaris 9 platform and it has shown success. Its efficiency and speed of retrieval is further justified in the performance test conducted in the next section.

PERFORMANCE TEST

A performance test was done using six biodiversity databases presented in Table 4. The performance test method in this research followed a standard model used by Roderic [20]. For each database, the species name was searched 100 times and response time was recorded from the time the query was made until the time the results were returned. The results of the performance bench-marks are shown in Figure 7.

Table 4. Relational Biodiversity Databases Tested

NAME	DBMS	OPERATING SYSTEMS	NO OF RECORDS	LOCATION
Malaysian Indigenous Algae	Microsoft Access	Windows XP	7	Bioinformatics Research Lab, UM
Malaysian Indigenous Fern	FileMaker	Windows XP	125	Zoology Museum, UM
Photogallery	MySQL	Windows XP	48	Bioinformatics Research Lab, UM
South East Asia Fruit Flies	MySQL	Solaris 9	10	Zoology Museum, UM
Biodiversity sample database	mySQL	Linux	10	MIMOS, Technology Park Malaysia
Virus sample database	DB2	Windows XP	21	Bioinformatics Research Lab, UM



	Malaysian Indigenuos Algae	Malaysian Indigenuos Fern	Photogallery	Fruit Flies	Biodiversity Sample	Virus Sample
▨ Median	0.21	0.28	1.1	0.39	0.5	0.85
■ Mean	0.28	0.33	1.25	0.32	0.41	0.12

Figure 7. CABIR performance test results

The Malaysian Indigenous Algae (MIA) and Malaysian Indigenous Fern databases showed the best median response times (0.21 and 0.28 respectively). MIA even showed 0 failures. Photogallery, running on mySQL has the slowest median response time (1.1), with the highest number of failures (3). This is because Photogallery is an image database and due to the size of images, it takes longer to retrieve them. However, it is difficult to generalize about these results as the performance of a data source actually depends on number of factors, such as the server hardware and software, the database design and the net-work performance. For the six databases queried, the operating systems used include both UNIX and Windows based and the Web servers were Apache and IIS. Nevertheless, it is indeed very promising that five out of six databases have median response time of less than a second. This means that the response time for search using the CABIR system is less than a second.

DISCUSSION AND CONCLUSION

CABIR is a simple system designed to serve the purpose of storing, disseminating and sharing biodiversity information among the scientific community. It also provides interaction among researchers while maintaining the privacy of their databases by controlling the permission for sharing. They are also able to host and maintain their own databases rather than exporting data into a warehouse. Search results are typically returned in few seconds in CABIR. Therefore, it is a high performance database integration system while maintaining its simplicity and having the generic characteristics of a database integration system. This system can be further explored and improved to handle the up to date demands of the scientific community. CABIR is expected to work with data sets outside the biological domain due to its flexibility. Finally, CABIR is deemed to attract various researchers interested in database integration field.

ACKNOWLEDGEMENTS

We would like to thank Professors Haji Mohamed and Halijah Ibrahim from University Malaya and Dr Chua Tock Hing from Monash University, Sunway for providing us with data used in this study. We also would like to thank Professor Mashkuri, Dr Lai and Mr Luke from MIMOS, Malaysia for allowing us to use their facilities to test the system.

REFERENCES

1. Philippi, S. (2004). Light-weight integration of molecular biological databases. *Bioinformatics J.* **20** (1): 51-57. Oxford University Press.
2. Davidson, S., Overton, C. and Buneman, P. (1995). Challenges in integrating biological data sources. *Comput. Bio. J.* **20**: 557-572.
3. Schulze-Kremer, S. (2002). Ontologies for molecular biology and bioinformatics. ***Silico Biol* 2:** 179–193.
4. Jacob, K. (2004). Integration of life sciences databases. *DDT: Biosilico.* **2** (2): 61-69
5. Xia, Y., Stinner, R. E. and Chu, P. C. (2002). Database integration with the web for biologist to share data and information. *EJB Electronic J. of Biotech.* ISSN: 0717-3458, Vol. 5 No. 2, 154-161
6. White, J. A., Maltais, L. J., and Nerbert, D. W. (2002). *An Increasingly Urgent Need for Standardized Gene Nomenclature.* Nature Genetics, Nomenclature (online). Available: http://www.nature.com/ng/web_specials/nomen/nomen_article.
7. Limi, A., Runyan, A. and Anderson, V. (2005). *Darwin Core 2* (online). Taxonomic Databases Working Group (TDWG). Available: http://darwincore.calacademy.org/Documentation/PurposeGoals
8. International Union of Biological Sciences (IUBS) (2005). *Taxonomic Databases Working Group: Time Limited Ballot* (online). Natural History Museum, London, United Kingdom. Available from: http://www.tdwg.org/TDWGStandardsBallot2005.htm
9. IABIN (2004). *Biodiversity Informatics - Information Standards and Metadata Formats* (online). Available: http://www.iabin.net/english/bioinformatics/protocols/standards_&_formats.shtml
10. Canhos, V.P., Souza, S., Giovanni, R. and Canhos, D. A. L. (2004). Global biodiversity informatics: Setting the scene for a "New World" of ecological modeling. *Biodiversity Informatics J.* **1**: 1 - 13.
11. Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer* **25** (3): 38 - 49.

12. Domenig, R and Dittirich, K. (1999). An overview and classification of mediated query systems. *ACM SIGMOD Record* **28**: 63 - 72.
13. Widom, J. (1995). Research problems in data warehousing. In: Proceedings of the *4th International Conference on Information and Knowledge Management (CIKM)*. pp. 25 - 30. Baltimore, Maryland, ACM Press.
14. Bichutskiy, V. (2004). Heterogenous biomedical database integration system using hybrid strategy. *UCI Undergraduate Research J.* **7**: 11 - 18.
15. Vieglais, D. (2003). *The Darwin Core Species Analyst* (online) Available:http://speciesanalyst.net/docs/dwc/index.html
16. FileMaker, Inc. (2005). *FileMaker 8 ODBC and JDBC Developer's Guide* (online). California. Available: http://www.filemaker.com/downloads/documentation/fm8_odbc_jdbc_developer.pdf
17. Sarinder, K. K. S, Majid, M. A, Lim, L. H. S, Ibrahim, H. and Merican, A. F. (2006). *Living Web* (online). University Malaya, Kuala Lumpur. Available: http://umbioweb.um.edu.my.
18. Sarinder, K. K. S, Majid, M. A, Lim, L. H. S, Ibrahim, H. and Merican, A. F. (2005). Integrated biological database initiative (IBDI). In: Proceedings of the *International Conference on Biogeography and Biodiversity Wallace in Sarawak – 150 Years Later*. Kuching, Malaysia. Institute of Biodiversity and Environment Conservation, University Malaysia Sarawak, Sarawak, Malaysia.
19. West, L. (2001). *Net HTTP Client* (online). Available: http://lwest.free.fr/doc/php/lib/net_http_client-en.html.
20. Page, R. D. M. (2005). A taxonomic search engine: Federating taxonomic databases using web services. *BMC Bioinformatics* **6** (48): 1-8.
21. Merican, A. F, Othman, R. Y., Sarinder, K., Ismail, N., Kok, C. Y., Khoo, P. C., Yong, C. F., Moak, S. F. L. (2002). Development of Malaysian Indigenous Microbial Online Database System. *Asia Pacific J. Molecular Biology and Biotechnology* (in press).
22. SourceForge (2005). *Distributed Generic Information Retrieval (DiGIR)* (online). Available from: http://digir.sourceforge.net/