# A Generic Model for Querying Multiple Databases in a Distributed Environment Using JDBC and an Uniform Interface

Barkha Bhagwant Keni, M.Madiajagan, B.Vijayakumar

***Abstract*** **- This paper discusses about an algorithm for accessing and querying multiple databases in a distributed environment. It makes use of JDBC and an UNIFORM INTERFACE to validate a SQL Query initiated from a Client Site, generates sub-queries based on Global Schema information, executes sub-queries in parallel using SQL Servers at appropriate sites, assembles the results of the query at the Client Site and outputs the results. This model reduces the effort required to search multiple databases stored at various sites in a network of heterogeneous systems, using a generic approach.**

**The proposed model can be very helpful in practical applications such as *banking systems* and *airline enquiry systems* where there is a need to execute several queries over multiple databases located in different servers.**

*Index Terms*: **API, Database Search, Global Schema, JDBC, MYSQL, PSQL.**

## I. INTRODUCTION

A database application requires an access to multiple databases and each database has several views and tables [3, 5].

Whenever one wants to search for information in multiple databases, he/she have to do so using the *front end* of the particular database server. The user has to spend considerable amount of time and effort in *formulation* and *execution* of several queries over multiple databases located in one or more servers. The proposed work simplifies the above two tasks using JDBC (Java Database Connectivity) and an Uniform Interface.

Manuscript received on June 24, 2007.
Barkha Bhagwant Keni is with the Computer Science Dept., BITS, Pilani-Dubai Campus, P.O.Box 688, Sharjah, United Arab Emirates; e-mail: barkha.keni@gmail.com .
M. Madiajagan is with the Computer Science Dept., BITS, Pilani-Dubai Campus, P.O. Box 500022, Dubai, United Arab Emirates; e-mail: jagan@bitsdubai.com .
B.Vijayakumar is with the Computer Science Dept., BITS, Pilani-Dubai Campus, P.O. Box 500022, Dubai, United Arab Emirates; e-mail: bv_uma@yahoo.com .

## II. OBJECTIVES

The present work is intended to meet the following objectives:

1. Design of an efficient and interactive user interface that would accept valid SQL query (over multiple databases stored in one or more sites) as input from end user.
2. Provide an environment for effective query execution in a distributed environment.
3. Allow the end user to view/print the output of the query.

## III. ACRONYMS

API : Application Program Interface
DBMS : Database Management System
GS : Generic Search
JDBC : Java Database Connectivity
MYSQL &
PSQL : Open Source DBMS for SQL queries
OS : Operating System
SQL : Structured Query Language

## IV. RELATED WORK

The *universal relation* as an user interface has been dealt in greater detail in [6]. Here, the data of the entire database are modeled using a single relation. The *schema* of the universal relation encompasses all the attributes in any of the relation schemes of the database. The advantage of this scheme is that an user need not remember the details of all attributes and their groupings in each relation.

The proposed work considers multiple databases spread over one or more sites in a heterogeneous network. Hence, it is necessary to have efficient Schema Management and Search Techniques to retrieve information in the above mentioned environment and a general approach to database search has been presented in this paper.

## V. PROBLEM DESCRIPTION

A generic model for database search application is shown in **Fig 1** and **Section 6**. The user enters the query and passes it to the GS (Generic Search) module. This module consults global schema which contains the following information:

*Site Identifier* (IP Address), details of all *relations* and their respective *databases* under which they are present, names of *DBMS software* supported and the *operating system* environment

The Global Schema is present at all the client sites where the queries are initiated.

The GS module maps the *relation(s) name(s)* in the query with the *Database Name* and *Site Identifier* and generates sub-query for each site. The sub-queries are dispatched to the appropriate sites. It then sets up the query for execution by the appropriate SQL server at each site and assembles the results at the client [5]. The GS module is implemented using JAVA code supported with JAVA API and JDBC. For experimental setup, three sites have been considered, although this model is applicable to any number of sites.

The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access. JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data.

The user will enter the query in the space provided in the interface. The program will search for the occurrence of the clause "**from**". Extract the **table name** which follows "**from**" in the query. The back end server will open the text file and locate the **Database Name** and **Database Server**. The **JDBC** driver will connect to the appropriate **database server** and **database** under the server as shown in Fig 2. The query undergoes execution in the database server(s) [1] and assembles the results, in a text file at the Client Site. The output generated in this text file can be viewed by the end user.

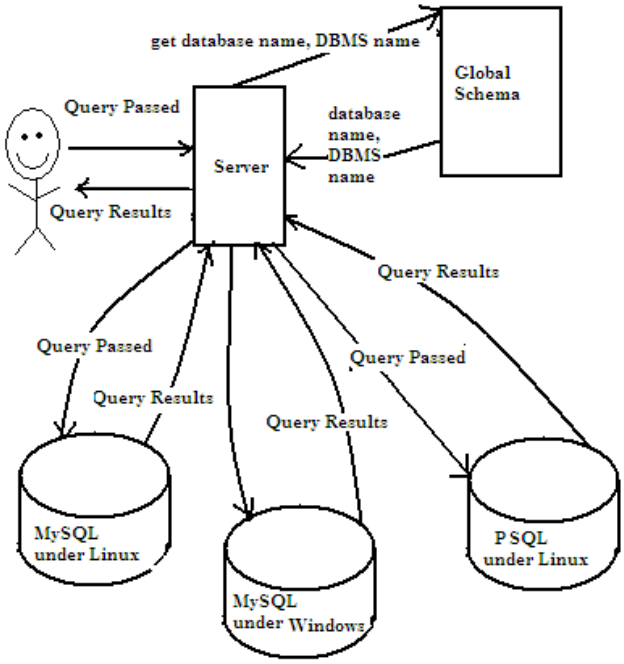


Figure 1: SCHEMATIC for DATABASE SEARCH in a GENERIC MODEL

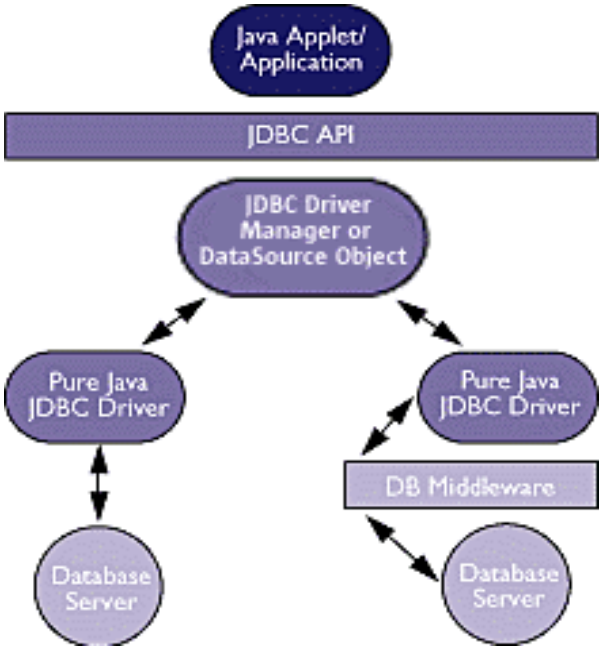


Figure 2: Database Access using JDBC Driver

## VI. GENERIC ALGORITHM

The generic algorithm works as follows**:**

Algorithm: GS (Generic Search)

Input: A text file containing SQL Query initiated from a client site, Global Schema.

Output: The results of the query (in tabular form) at the client site.

Procedure:

1. The USER enters the SQL query using the UNIFORM INTERFACE.
2. The query is stored in a text file and is checked for correct Syntax and the **relation names** & **attribute names** are validated from the **Global Schema**.
3. Open *Global Schema* containing the *database names & relations* under them, Site *Identifier* information, OS name and DBMS name [POSTGRES/MYSQL].
4. Associate the names corresponding to *attributes, relations & database* present in the query with the *Global Schema* information.
5. Generate Sub-Query for each site.
6. Dispatch the Sub-Query to the appropriate Site and Connect to the appropriate Database Server at each site, using JDBC.
7. The SQL SERVERS execute the sub-queries in parallel at each site.
8. Assemble the *query results* in an output file at the client site.
9. The output file at the client site can be viewed as well as printed progressively.

The above algorithm has been implemented using JAVA code supported with JAVA API and JDBC.

## VII. EXPERIMENTAL SETUP

POSTGRES and MySQL are the DBMS softwares that have been considered here.

MySQL [4] and PSQL (POSTGRES SQL) [9] are available on both Linux and Windows Platforms. They are widely used for developing database applications over the web. LINUX and WINDOWS OS have been chosen since both of them are widely used for running many database applications.

The Relational Model for Databases has been considered in this paper. The Queries assumed to be in standard SQL format and are submitted from query files.

Examples used to illustrate this paper are drawn from the following three databases. Each database has three relations. Each Database is stored at a separate site.

1. MySQL Database under LINUX : **library1**
   Site 1:- IP Address**:** 172.16.12.1

The schema of all the tables in the above database is as follows:

lib_journal (ISBN, Publisher, title, no_of_issues, subprice)
lib_cse (ISBN, Publisher, Author, title)
lib_catalog (ISBN, Author, Title, Year, subprice, accno)

(The primary key field is underlined in each relation.)

2. MySQL Database under WINDOWS : **library2**
   Site 2:- IP Address**:** 172.16.12.2

The schema of all the tables in the above database is as follows:

lib_journal (ISBN, Publisher, title, no_of_issues, subprice)
lib_cse (ISBN, Publisher, Author, title)
lib_catalog (ISBN, Author, Title, Year, subprice, accno)

*library1 & library2* are horizontally partitioned databases on the primary key: ISBN.

3. PSQL Database under LINUX: **student**
   Site 3:- IP Address**:** 172.16.12.3

The schema of all the tables in the above database is as follows:

Stud_master (idno, name, address, reg_date)
Stud_tran1 (idno, cgpa)
Stud_tran2 (idno,Courseid,t1,t2,q1,q2,ce,total,accno)

## VIII. DATABASE CREATION

The following commands were used to create tables under the **library1** & **library2** databases:

```
CREATE TABLE lib_journal (
ISBN int(10) NOT NULL Primary Key,
Publisher char(20),
Title char(25),
Noofissues varchar(10),
Subprice real);
CREATE TABLE lib_cse(
ISBN int(10) NOT NULL,
Publisher char(10),
Author char(10),
Title char(25),
CONSTRAINT ISBN_ID FOREIGN
KEY(ISBN)REFERENCES lib_journal (ISBN));
CREATE TABLE lib_catalog (
ISBN int(10) NOT NULL,
Discipline char(10),
```

```
Author char(20),
Title char(25),
Year int,
Subprice real,
accno char(7),
CONSTRAINT ISBN_ID2 FOREIGN KEY (ISBN)
REFERENCES lib_journal (ISBN));
```

The following commands were used to create relations under the **student** database:

```
CREATE TABLE stud_master(
id int primary key,
name varchar(20),
address varchar(20),
reg_date varchar(30));
CREATE TABLE stud_tran2 (
idno varchar,
courseid varchar,
t1 real,
t2 real,
q1 real,
q2 real,
ce real,
total real,
accno char(7),
Foreign key (idno) references stud_master);
CREATE TABLE stud_tran1 (
idno varchar,
cgpa real,
Foreign key (idno) references stud_master);
```

## IX. INTERFACE DESIGN

The UNIFORM INTERFACE as shown in Fig 3, allows the user to enter the query. The interface has been implemented using JAVA API [2] and connection to database is achieved by locating the JDBC driver [7, 8]. Once the user has entered the query and has clicked the "Execute Query" button, the back end will find out the database server and connect to it. The SQL query is then executed and the user can view the query as well as its output results.

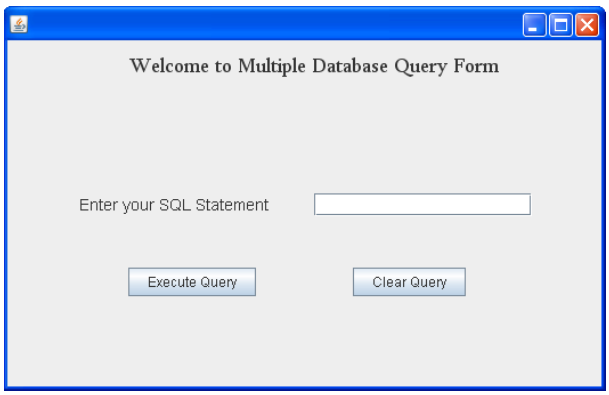


Figure 3: Layout of the UNIFORM INTERFACE

## X. TEST SCENARIO

The JAVA program for the present work has been compiled and run using the following sequence of two steps:

```
$ javac Mainclass.java // to create the class file

$ java Mainclass // for execution
```

a) The following query has been entered in the input text file (in1.qry) using the uniform interface:

```
select stud_master.name, stud_tran2.t1, stud_tran2.t2,
stud_tran2.q1, stud_tran2.q2 from stud_master,
stud_tran2 where stud_master.idno=stud_tran2.idno;
```

A complete trace of the java program's execution is shown as follows:

*Access details*:
Site(s) : 172.16.12.3
DBMS(s) & OS(s): PSQL under LINUX
Database(s) : student

Table 1: Results of Sample Query a).

| Name | T1 | T2 | Q1 | Q2 |
|---|---|---|---|---|
| Aquarius | 45 | 65 | 7 | 6 |
| Venus | 67 | 43 | 9 | 9 |
| Ruby | 60 | 66 | 9 | 9 |
| …….. | …….. | ……. | ……. | …….. |

b) The following query has been entered in the input text file (in2.qry) using the uniform interface:

```
select stud_tran2.idno, stud_tran2.accno,
lib_catalog.author, lib_catalog.title from stud_tran2,
lib_catalog where stud_tran2.accno =
lib_catalog.accno;
```

A complete trace of the java program's execution is shown as follows:

*Access details*:
Site(s) : 172.16.12.1, 172.16.12.2, 172.16.12.3
DBMS(s) & OS(s): MYSQL under LINUX,
MYSQL under WINDOWS,
PSQL under LINUX.
Database(s) : student, library1, library2

| IDNO | ACCNO | AUTHOR | TITLE |
|---|---|---|---|
| 2002u7ps035 | B32562 | J. Ullman | Database Systems |
| 2002u7ps023 | C40023 | Fred Halsall | Multimedia communications |
| …….. | …….. | …….. | ……. |
| …….. | …….. | ……. | ……. |

Table 2: Results of Sample Query b

## XI. COMPLEXITY

**Table 3** summarizes the time complexity of SQL Operations at each site, as referred in Section 6.

Let **n** be the number of tuples in a relation / fragment.

| Operation | Complexity |
|---|---|
| SELECT | **O(n)** |
| PROJECT | **O(n)** |
| JOIN (sort-merge) | **O(n * log n)** |

Table 3: Complexity of SQL Operations at each site

## XII. CONCLUSION

An algorithm for searching multiple databases in a heterogeneous environment has been discussed and implemented using JDBC and an Uniform Interface. It has been successfully tested with several queries which have been initiated from multiple client sites. The present work can be further extended to provide interoperability with additional DBMS software such as ORACLE & SQL SERVER 2000 and also incorporate a graphical user interface at client sites for viewing/updating S*chema Information* for the Global Relations, in accordance with access privileges assigned to the user.

The proposed model can be very helpful in practical applications such as banking *systems* and *airline enquiry systems* where there is a need to execute several queries over multiple databases located in different servers.

## ACKNOWLEDGEMENT

The authors are thankful to the referees for their valuable comments in revising this paper.

## REFERENCES

[1] Ellsworth,Abigail, “Databases”, EBSCO Electronic Database, Radcliff,Carolyn Jerence & User Services quarterly, spring 2002, Vol.41, Issue3, pp. 276-278.
[2] George Reese, “Database Programming with JDBC and Java”, OReily Publications, 1st Edition June 1997, pp 40-57.
[3] Johnson,James C, EBSCO Electronic Database, “Black Enterprise”, Apr 2005, Vol.35, Iss. 2, pp. 52-54.
[4] MYSQL, “The MYSQL Open Source Database”, [Online]. Available: www.mysql.com .
[5] Xi (Michael) Zhang, Tony C. Pan, Umit V. Catalyurek, Tahsin M. Kurc, Joel H. Saltz, "Serving queries to multi-resolution datasets on disk-based storage clusters", Proceedings of the 4th IEEE International Symposium on Cluster Computing and the Grid, 2004 (CCGrid 2004), 2004, pp. 490-498.
[6] Ullman J.D., “Principles of Database Systems”, 2nd edition, W H Freeman & Co., NY, 2001.
[7] JAVA by API, [Online]. Available: www.java2s.com .
[8] Sun Developer Network (SDN), “The Source for JAVA Developers”, [Online]. Available: www.java.sun.com .
[9] POSTGRESQL, “The POSTGRES Open Source Database”, [Online]. Available: www.postgresql.org .