# *Implementation of a Prototype Client–Server Model for a Generic Database Application over a Network*

*Aksheta Mehta*
Dept. of Computer Science
BITS PILANI, Dubai Campus
Dubai, UAE
aksheta.mehta@gmail.com

*B. Vijayakumar*
Dept. of Computer Science
BITS PILANI, Dubai Campus
Dubai, UAE
vijay@bits-dubai.ac.ae

***Abstract*—The client–server model is a computing model which acts as a distributed application that bifurcates the workload between servers and clients. A server is the provider of a service or resource while a client is the service requester. This paper gives an account of the design and implementation of a prototype client–server model for any network-based database application. It provides a generalized model that can be fine-tuned to fit the client's specifications.**

***Keywords*—client–server architecture, data–centric architecture, database application over a network, database management**

## I. Introduction

A server is a process providing a specific service or resource for a client or a collection of clients. A client, on the other hand, is the process which requests a service from a server. It is the client that initiates communication sessions with the server, which awaits incoming requests from the client [2]. The server then complies with the client's request[2][6]. This interaction between client and server can be described using sequence diagrams in Unified Modeling Language.

Studies on new trends in distributed systems [3], along with the requirements for client-server performance modeling are underway [5].

The Client–Server model is successfully used as the architecture for many applications. In fact, it is the most widely used type of architecture. Hence its performance and scalability is a subject for study [1] and a proper understanding of this architecture is essential [7], [4], [2], [6].

This prototype is capable of preprocessing input files to conform to the appropriate format, creating, populating, managing as well as querying databases and then displaying the desired output, just by accepting a few simple and basic inputs from the user. In short, it works on a minimal input from a person to give the required output, hence making it more automated than human reliant and in effect reducing the human workload as well as the chances of any error based on complications and/or erroneous input (i.e. human error).

## II. Objectives

The present work is intended to meet the following objectives:

- Specify a data–centric model for a generic application.
- Provide a mechanism to implement functions relating to client, server and user..

## III. Definitions

The definitions used in the context of this paper are as follows:

- *Server:* is the process that provides the resources and services to execute the jobs specified by the client and the user.
- *Client:* is the process that sends a request to the server to create and/or modify a database along with the specifications of the structure and contents of the database.
- *User:* is the process that submits specific queries, based on its requirements, to the server and receives the results of those queries.

## IV. Models of client-server architecture

In a Client – Server architecture, an application is modeled as a set of clients and a set of services that are provided by the server. The clients and servers can be regarded as two separate logical processes [4][2][6].

Client – server architecture can be of two types[4], namely:

### *A. Two–Tier Client–Server Architecture*

Two – tier client – server architecture can further be divided into two types:

- Thin–Client Model
- Fat–Client Model

### B. *Three–Tier Client–Server Architecture*

For our prototype we have considered the Thin–Client Two–Tier Client–Server Architecture.

## V. Deployment scenario for the proposed system

Since databases are a crucial part of the prototype, Database – centric architecture is the most applicable architectural pattern.

*Input:*

- Database with one or more relations as specified by the client.
- Queries as submitted by the user using the client module.

*Output:* Output results for the input queries as submitted by the user.

The Deployment involves two phases, namely:

- File–based Preprocessing Algorithm
- Data–Centric Client–Server Implementation

### A. *File-based preprocessing Algorithm*

Processing any kind of data is easier and faster when it is set to a format or a pattern.

To successfully populate a database from a file, the format of the file should be such that each row of the table is a new line and each value in a row is separated by a delimiter.

1) *Step 1:* Create a batch file with the required commands.
2) *Step 2:* Pick out and separate the non-repetitive data values (values that appear just once in the file and do not need to be repeated in the database).
   a) Copy and write lines with a specific keyword into another file, using sed command (line-wise command)

      sed -n '/keyword/w tofilename' fromfilename

   b) Separate the field names from the data, using cut command (column-wise command)

      cut -c columnno.- fromfilename > tofilename

   c) Paste all the different data files together with each field separated by a delimiter, using paste command (column-wise command)

      paste -d\delimiter fromfilenames separated by whitespace > tofilename

3) *Step 3:* Pick out and separate the repetitive data values (values that appear in the file once but can be mapped to a field more than once and hence need to be repeated in the database).
   a) Copy and write lines with a specific keyword into another file, using sed command (line-wise command)

      sed -n '/keyword/w tofilename' fromfilename

   b) Separate the field names from the data, using cut command (column-wise command)

      cut -c columnno.- fromfilename > tofilename

   c) Repeat the value the desired number of times, using awk command (field-wise command)

      awk -F " " 'NR==line1, NR==linelast {print NR, $1, $2, $3...$totalno.offields}.....{repeat print command as many number of times as the value/values needs/need to be repeated}' fromfilename > tofilename

### B. *Data-Centric Client-Server Implementation*

The block schematic for the proposed prototype is shown in Figure 1.

1) The program in the server asks the client to define the parameters of the database to be created.
2) The client specifies the number and names of the fields to be created.
3) The server then creates the table and then asks for the input file.
4) The client gives the location of the input file to the server.
5) The server performs file–based preprocessing on the input file before populating the database created with its contents.
6) The server then asks the client if there are any modifications to be made in the database created. If 'yes' then the process goes back to the beginning of the program in the server to repeat the above process. Else the process moves on.
7) Next the server asks the user to input the queries.
8) The user inputs the queries.
9) The server implements those queries in SQL and displays the desired results to the user.
10) The user receives the output.
11) The server now asks the user if there are any more queries. If 'yes' then the process from point *7)* through *11)* gets repeated. Else the process ends.

## VI. Test scenario

The proposed model has been implemented using JDBC provided in the jdk 7u2 package. The entire program was compiled and run using the ODBC driver on the Ubuntu 10.10 Operating System and tested successfully for the following scenarios:

a. Creating and Populating a table: under the client module, the client specified the details of the table to be created as well as the path of the preprocessed file containing the records for the table. Using the client's input the server created and populated the required table.

b. Modification of an existing table: under the client module, the server gave the client a set of options which enabled the client to delete an existing table, add a new column to an existing table, add a new row to an existing table, change the values of an existing row in a table as well as delete a row from an existing table.

c. Passing of queries: under the user module the user specified the table to be queried along with the conditions for the required output. The server created a query corresponding to the conditions specified by the user and displayed the appropriate output.

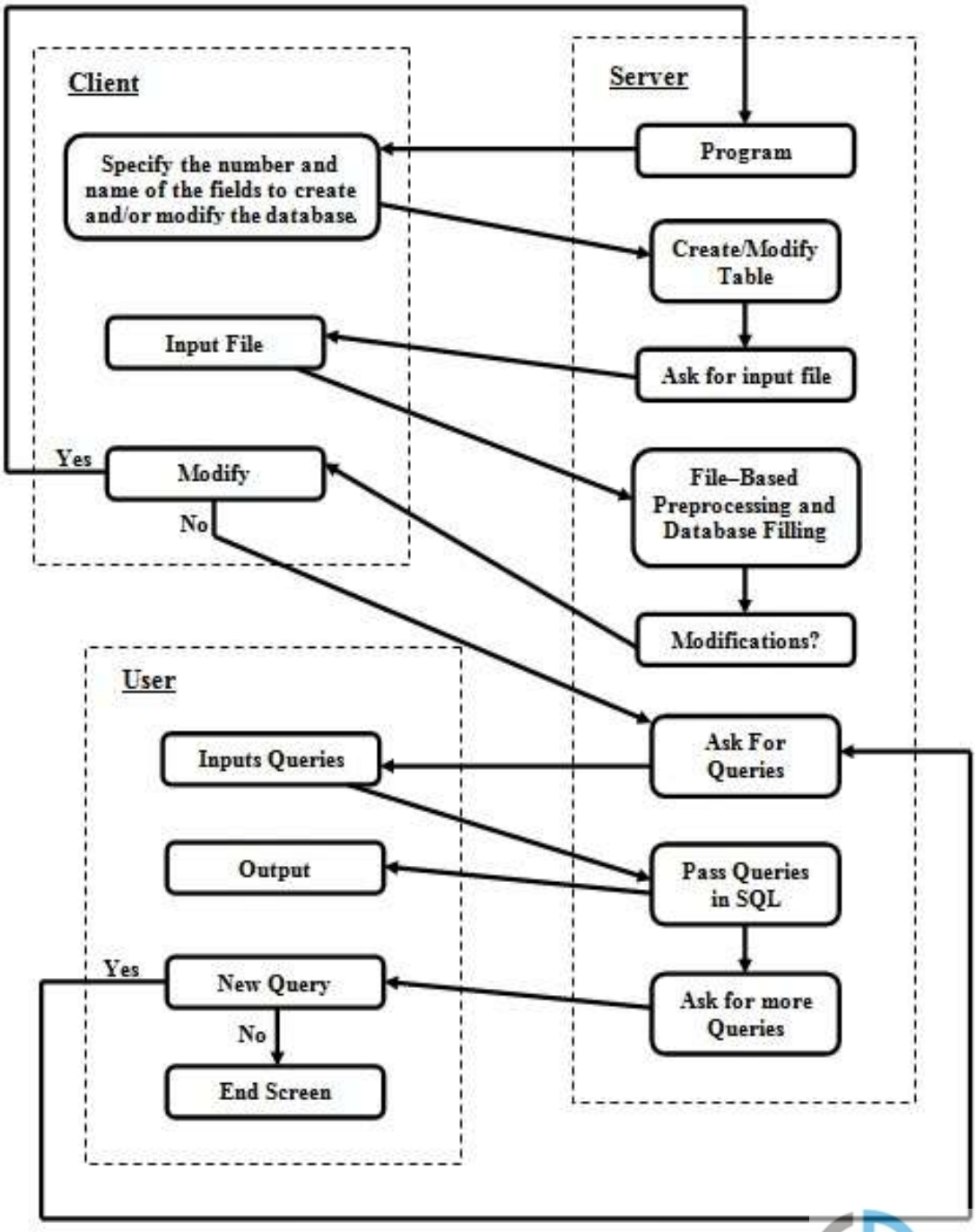

Figure 1: Data–Centric Client–Server Implementation.

# Conclusion

This paper dealt with an overview of thin–client two–tier client–server architecture. The deployment scenario has been presented for a generic database application. The prototype gives any client on the network the flexibility to define its own database structure. The user component is responsible for submitting the input queries and receiving the results of the queries from the server. The server component executes the jobs submitted by the clients and the users. The presented work can be rapidly and effortlessly deployed under any application domain.

## *Acknowledgment*

I would like to take this opportunity to extend my heartfelt gratitude to Prof. R.K.Mittal, Director, BITS- Pilani – Dubai. I would also like to sincerely thank my supervisor, Dr. B Vijayakumar, for his untiring support and uninhibited guidance.

I am also indebted to Dr. M. Madhiajagan, Dr. Santosh Kumar and Dr. G.Vijaya, BPD faculty, for their valuable suggestions and advice.

## *References*

[1] Alexios Delis and Nick Russoloulos, "Performance and Scalability of Client–Server Database Architectures", in the proceedings of the 18th VLDB Conference, Vancouver, British Columbia, Canada, 1992.

[2] Client–server model, http://www.csc.liv.ac.uk/~konev/COMP212/06-client-server.pdf

[3] D. Gaïti, Intelligent Distributed Systems: New Trends , Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, Sept 1993, pp,106-111.

[4] Ian Somerville. "Architectural Design" and "Distributed Systems Architecture", in Software Engineering, Eighth edition, Pearson Education Limited, Essex, England, 2007, pp, 247-252 and 266-289.

[5] Joseph J. Martinka, "Requirements for Client/Server performance modeling: An interpretation using Discrete Event Simulation", in proceedings of the 1st International Conference on Engineering of Complex Computer Systems, IEEE Computer Society Washington, DC, USA, November 6-10,1995, pp, 108-111.

[6] Lawrence Chung, Computer Science Program, The University of Texas, Dallas, Client–Server Architecutre, http://www.utdallas.edu/~chung/SA/2client.pdf

[7] Michael T. Goodrich and Roberto Tamassia, "Internet Algorithmics", in Algorithm Design: Foundations, Analysis, and Internet Examples, Paul Crockett and Bill Zobrist, Ed. Singapore: John Wiley & Sons (Asia) Pte. Ltd., 2002, pp, 415-545.

[8] Microsoft, Chapter 7: Client/Server Architecture, http://technet.microsoft.com/en-us/library/cc917543.aspx