ADAPTIVE SOFTWARE INTEGRATION MODULE USING NEURAL NETWORKS

Vasile CORNIȚĂ[1], Rodica STRUNGARU[2], Sever PAȘCA[3]

Articolul prezintă un modul software avansat de integrare între două sisteme proprietare utilizând tabele temporare şi rețele neuronale în scopul optimizării interogărilor SQL de integrare. Modulul software de integrare propus primește interogări SQL, le analizează şi încearcă să optimizeze timpul de execuție al acestora. Un prim aspect important de luat în considerare la realizarea unei soluții software de integrare între două sau mai multe sisteme îl reprezintă modalitatea de transmisie a datelor şi entităților(obiectele de business) necesare integrării efective, ținând cont de aspecte specifice sistemelor de intregrat cum ar fi: organizarea structurilor de date, modalități de extragere a datelor, interogări dinamice. La nivel de companie, atunci când deja există soluții software proprietare deja achiziționate pentru anumite domenii specifice, dar care nu funcționează unitar; integrarea între sistemele deja existente este o soluție viabilă atunci când costurile achiziționării sau dezvoltării unui sistem proprietar cu toate funcționalitățile incluse sunt mult mai mari, comparativ cu dezvoltarea unui modul de integrare specific.

This paper presents an advanced software integration module between two proprietary systems using temporary tables and neural networks for SQL integration requests optimization purpose. The proposed software integration module receives SQL queries, analyzes them and tries to optimize execution time if necessary. One important aspect to consider when realizing the integration component between two or more systems is the data structure passing technique, taking into account specific system implementation issues like: data structures organization, storage, retrieval and dynamic requests. Nowadays there are many dedicated applications for specific business to take into account, but when there is no such software application with all required functionalities; integration between existing proprietary software applications is to be considered, especially when the cost of development or purchasing of a new system with all required functionalities is significantly higher than developing a necessary software integration module.

Keywords: enterprise application integration, SQL query optimization, database management system kernel, neural networks

[1] PhD. Student, The Department of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: cornita_vasile@yahoo.com

[2] Prof., The Department of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: rodica.strungaru@elmed.pub.ro

[3] Prof., The Department of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: pasca@elmed.pub.ro

1. Introduction

Nowadays, specific businesses has acquired production software applications, mostly when necessary and targeting a set of particular requests. As a business expands, it appears the need for connecting together geographically separated departments and their associated business processes.

As corporate dependence on technology has grown more complex and far reaching, the need for a method of integration disparate applications into a unified set of business process has emerged as a priority. After creating islands of automation through generations of technology, users and business managers are demanding that seamless bridges be built to join them. In effect they are demanding that ways be found to bind these applications into a single, unified enterprise application. The development of Enterprise Application Integration(EAI), which allow many of the stovepipe applications that exist today to share both processes and data, allow for an answer to this demand. [1]

From practice, when considering the integration between two software applications, let's denote these applications App_A and App_B, good results are obtained when the integration is carried out by the software company that produced either, App_A or App_B, because only one application data details are to be learned from scratch.

In order to integrate two software applications, two important aspects are to be taken into account:

- The data model: The exchanged data are in fact business documents and not simple character strings. It is highly probable that these documents (the two application documents) will contain lots of identical data, but it will not necessary be in the same format. A conversion job from one model to another is therefore needed.
- The communication system: In this context it is very important the communication protocol used to exchange data. Here will come to the role of middleware. A large part of software application integration is about the different technologies and techniques implementing this exchange. [2]

Typically, in internet contexts, the protocol used is HTTP (Hypertext Transfer Protocol).

Also data availability and data security policies have to be taken into account when integrating software applications.

The main purpose of the proposed software integration module(SIM) is to provide with real time data from a production application that formulates a considerable number of SQL queries per second to a relational database, to the proprietary business management reporting tool application.

The integration module can be configured to start manually or automatically, and, in case of a system failure the module is configured to start automatically.

The module comprises functions for giving real time data to reporting part of the management application as well as commands formulated by the management application for the production system to execute.

The production system logic is controlled by either the business management application (external control via the integration module) or by its own logic module if activated.

The integration module uses temporary tables as buffers in order to exchange data between production application and business management tools. Reading and writing from/into these tables is done with access rights for both of the applications. The software integration module is a part of a database system kernel server which works in a conjunction with a client application that connects to the mentioned server and formulates requests.

2. Database Management System Kernel (DBMSK)

The database management system kernel represents the general server which incorporates the software application connection-integration configurable module.

The server uses a typical architecture. The remote client sends a request to the Database Server. For a particular request, the server decides on and takes the appropriate action.

This can be represented graphically as below:

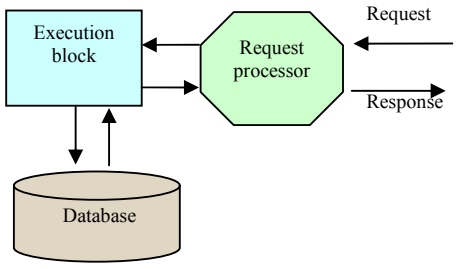


Fig. 1. DBMSK general architecture

The main functions of each block are explained below:

- Request processor takes as input a request expressed in structured query language and interprets it. After all request characteristics have been determined, it is up to the execution block to act appropriately.

- Execution block communicates with both the request processor and the Database layers in order to execute the client request. It must be mentioned that this layer executes when the request characteristics have been determined by the request processor layer and only then.

Both, requests and responses can take message or file transfer forms, depending on the specific context.

A simplified graphical representation of the Client application architecture is depicted below:

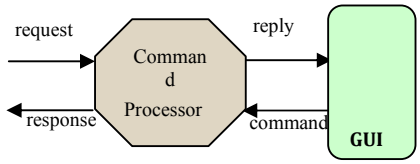


Fig. 2. Client general architecture

As it can be easily seen, all that happens is driven by user commands, making use of a well-designed graphical user interface.

The command processor is a logical entity that transforms client requests into an appropriate format for the server to understand and process.

Request can take the following two forms:

- structured query language requests
- standard requests

All these requests are transported between client and the server encapsulated in a general request type, this ensuring both, flexibility and extensibility for the request transport level. The communication mechanism between client and server uses both messages and files. The data interchange operation is realized via file transfer making use of standard XML language.

For implementing the Server and Client software applications general programming books [4] [11], C++ programming books [3] [5] [8] [12], database books systems [6] [7] [8] and socket programming books [9] [10] have been used.

3. Database software integration module

The demand of the enterprise is to share data and processes without having to make sweeping changes to the applications or data structures. Only by creating a method of accomplishing this integration can Enterprise Application Integration be both functional and cost effective. [1]

In general, software application integration must take into account all application specific aspects, communication protocol and various ways of storing

data structure (proprietary methods for manipulating data or general relational databases like Oracle, Microsoft SQL Server).

We present in Fig.3 the general architecture of a production system. Generally, the integration must be accomplished in order to give a reporting system or management software application with real data from the production system.

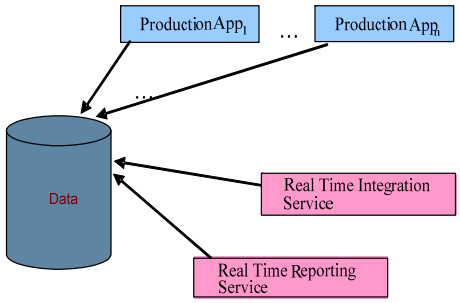


Fig. 3. Production System – General Architecture

Some important aspects considering figure above worth to be mentioned:

- The DBMSK kernel(server) and associated client application described in previous section performs the Real Time Integration Service task via contained software integration module (SIM).
- The Real Time Reporting Service represents the management application tool that monitors, triggers commands and receives feedback from production applications Production App_1 … Production App_n.

In considered integration, production system uses a relational database to which several production applications formulate continuously SQL requests. These requests are general request made by production application like: insert, select, update, delete. It is well known the fact that every relational database management system kernel achieves lock database operations on some specific table when executing specific SQL requests.

In this context, appears the problem of generating data for management applications via a real time software integration module component of the proposed database management system kernel.

A very important aspect is that management data is needed on a constant base for decision purposes. The client application sets the DBMSK kernel configuration such as business data representations, integration workflow, representing mainly a configuration interface for the DBMSK kernel.

Proposed software integration module (SIM) is capable to connect to various proprietary database management systems (DBMS) like Oracle, MSSQL and MySQL, using database specific ODBC drivers (Open Database Connectivity - a standard database access method developed by SQL Access group in 1992 which create the possibility to access any data from any application regardless of which DBMS is handling the data). The main advantage of using ODBC is that after connection realization the main focus is on the actual business logic of the software application to be developed, not on specific DBMS commands syntax. As database connection method, proposed SIM is using specific DBMS connection string. When specific SQL Queries are formulated to SIM, is its responsibility to execute mentioned queries in a reasonable amount of time.

The module's integration part main responsibility is to get the desired data from the production system taking into account specific data format and production business logic and provide the management software application with necessary data in required format and according to management application's logic. To execute this function the kernel server formulates SQL queries to production relational database. The problem to solve is that due SQL requests generated by production software application: App_1 …App_n, the production relational database is overloaded and kernel server SQL requests can not be executed in a specific, fixed period of time set by kernel server at query setup phase. To solve the problem, dynamic values for that timeout period of time, depending on production relational database overload are to be determined for specific periods of time.

The integration SQL has to get required data from tables pertaining to proprietary software production application:

Select [Field$_1$, Field$_2$, Field$_3$, Field$_4$ ... Fieldn]
From
T$_1$
Inner join T$_2$ on [specific fields]
Left join T$_3$ on [specific fields]
...
Inner join T$_m$ on [specific fields]
Where [Integration condition]

where:

- T_1, T_2, … T_m represent specific tables associated with production application.
- [$Field_1$, $Field_2$, $Field_3$, $Field_4$ … $Field_n$] represent required integration fields from production application.
- [specific fields] represent fields on which join between tables is carried out.

- [Integration condition] represents specific SQL integration condition, used to select integration data.

The integration SQL is set via the DBMSK kernel client, and is executed in the SIM module of the DBMSK kernel. Its main task is to provide feedback to the proprietary management application on production application system run. Those fields: ***$Field_1$, $Field_2$, $Field_3$, $Field_4$ … Fieldn***, returned by the integration SQL does not change over time. These fields represent data/business objects that are provided to the management application in order to track and manage the proprietary production system.

The data generated using integration SQL is used to populate the tables of management application. Production application usually writes specific data to its associated tables.

As production applications generated a considerable amount of data as a daily basis, which is reflected in the data composition of the associated tables (the number of records in production application's tables is growing faster), integration SQL execution time does not satisfy integration time requirement. Proper index structure was considered when executing integration SQL.

A solution would be to change significantly database structure associated to production application, but this is not applicable in the case of proprietary software system. Also, the cost to design and build from scratch the whole softwar

Thereby, a timestamp field will be added to large tables pertaining to production application. Let us denote this field T_S. This is a field used only by software integration module. After the integration for a particular set of data is achieved, the corresponding T_S field will be updated with the integration moment of time (this is a mark that tells us that a particular row of data has been integrated, feedback has been given to management application on that particular row of data). We also need an index associated to this field in the database.

The modified integration SQL takes into account the T_S field. The extra condition is as follows:

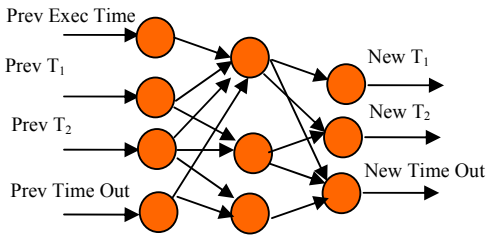
- $\mathbf{T_S > T_1}$
- $\mathbf{T_S > T_1}$ **and** $\mathbf{T_S <= T_2}$

where T_1 and T_2 are specific moments of time(time stamps) for which the integration will be carried out.

Depending on the database load, we must adjust integration SQL query time and T_1, respectively T_2 when necessary in order to successfully execute integration SQL in a reasonable amount of time (the target SQL execution time is at least 12 seconds).

The proposed strategy concerning the whole integration process is to execute the integration SQL in a maximum amount of time of 12 seconds via using previous SQL execution times, timestamp limits(T_1 and T_2) and timeouts in order to determine new timeout and timestamp limits for current integration SQL

execution. In order to achieve this, a typical back-propagation neural network [13][14], capable of solving the problem in order to constantly adjust query timeout at setup phase and T_1, respectively T_2 timestamps, based on previous SQL execution time on the production database, is used.



The neural network will have as input T_1 and T_2 corresponding to the integration moment of time, previous SQL query execution times, previous SQL query timeouts (the time interval for which the kernel server will wait for a response from production relational database for a specific query). The output layer will be used to adjust timeout interval for the integration module, T_1 moment of time and T_2 moment of time when necessary. New T_1 and T_2 values are used to modify **Where [Integration condition]**, in order to reduce the integration SQL execution time.

6. Results and discussion

Results obtained without the SQL parameters adjustment described above are presented in the following table:

Table 1

Integration static SQL execution time from production system

No. Days	4	11	18	21	31	44	51	60
Seconds	2	5	13	18	21	32	44	>56

As it can be seen from this table, after a production system run for 60 days, the integration SQL(*without the SQL adjustments with corresponding time stamps* T_1,T_2) does not execute in proper time, with negative impact on management system whose decision rely on reporting services for which the integration is realized. The *unadjusted* SQL execution time increases due to the fact that the number of rows of associated tables (those necessary to give feedback to management application) increases (due to large number of inserts performed by proprietary production applications). Technically, with that dynamic SQL, a limitation in the number of rows in the associated tables on which integration SQL is performed, is achieved, which translated in a small execution time of the integration SQL.

A significant impact concerns the production system's tasks execution time, because the execution of an SQL instruction implies specific tables locks and the production system is slowed.

With the proposed adjustment, SQL execution time is between 2-10 seconds, depending on the database load, allowing in this way a reasonable amount of time for tracking the production system (which translates in feedback provided to management application) and taking required decisions if necessary.

7. Conclusions

Presented application pertains to software application integration class. Proposed method for SQL optimization is useful when one (or more) proprietary system table comprises a large amount or records and the actual integration is realized using temporary tables acting as buffers. Using SQL parameters modification via proposed neural network the integration SQL successfully executed in a period of time 3-10 seconds. This execution time compared with static integration SQL execution time is much lower.

The neural network, the server kernel and client as well as the described connection-integration module were implemented in C++. As compared to other proprietary integration systems (IBM, Oracle, SAP) proposed solution, besides implementation cost has the advantage of having less middleware levels which translates in smaller execution times, but has not many configuration and connection options like proprietary systems mentioned above.

The Database Kernel, which contains the software applications integration module, was designed with futuristic thoughts. Therefore, it is suitable for any kind of application that involves custom data storage, retrieval and processing. In addition, the file transfer component of the server can be used in remote backup applications.

R E F E R E N C E S

[1] *D.S. Linthicum*, Enterprise Application Integration, Addison-Wesley Professional; first edition, 1999

[2] *D.Serain,* Middleware and Enterprise Application Integration: The Architecture of e-Business Solutions, Springer; 2nd ed. Edition, 2002

[3] *S. Salleh, Y. Albert Zomaya, A. Sakhinah Bakar*, Computing for Numerical Methods Using Visual C++, 1st edition, 2007

[4] *H. Thomas Cormen, E. Charles Leiserson, R. Ronald Rivest*, Introduction to algorithms, 2nd edition, The MIT Press, 2001

[5] *B. Stroustrup*, The C++ Programming Language, 3rd Edition, 2000

[6].*A. Richard Bassler, J. Jimmie Logan,* The Technology of Data Base Management Systems College Readings, 3rd ed edition, 1976

[7] *P. Rob, C. Coronel*, Database Systems: Design, Implementation, and Management, Eighth Edition, 2007

[8]. *L. Robison, K. David White*, Database Programming with Visual C++ in 21 Days , Pap/Cdr edition, 1998
[9]. *D. Roberts*, Developing for the Internet with WinSock, Bk&CD-Rom edition, Coriolis Group Books, 1995
[10]. *J. Richter, C. Nasarre*, Windows via C/C++ (Pro - Developer), 1st edition, 2007
[11]. *A. Jones, J. Ohlund*, Network Programming for Microsoft Windows, Microsoft Press, 2002
[12]. *C. Douglas Schmidt, D. Stephen Huston*, C++ network programming, 1st edition, Addison-Wesley Professional, 2001
[13]. *T. Munakata*, Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More (Texts in Computer Science), 2008
[14]. *J.Hawkins, S. Blakeslee*, On Intelligence, Numenta Inc, 2005