PERGAMON

Expert Systems with Applications 17 (1999) 45–49

Expert Systems
with Applications

Integrating Intelligent Systems using an SQL-database

R.M. Sonar*

National Institute of Bank Management, NIBM PO, Pune 411 048, India

Abstract

The intelligent systems are capable of addressing the applications in variety of domains. Each intelligent system has inherent strengths and limitations. The recent trend has been towards integrating the intelligent systems to solve a problem involving complex decision making. The integration not only improves the strengths of individual techniques but also lessens their drawbacks. Integrating these systems with the databases give them the power of the database management features and ability to access large volume of information in real-time. The article presents an approach: (1) to integrate intelligent systems to databases, and; (2) to use an SQL-database as a coupler to integrate the intelligent systems to build real-time distributed hybrid applications. This approach has been implemented in the shell called Hybrid Expert System Shell (HESS) to integrate expert systems and artificial neural networks.

Keywords: Intelligent systems; Hybrid expert system shell; Neural networks; SQL-database

1. Introduction

There are many applications coming up where more than one intelligent technique has been integrated to solve a common problem (Medsker, 1995; Suran & Sukhdev, 1995a). The hybrid approach has been found suitable to enhance the strengths of individual techniques and eliminate their weaknesses and thereby addressing more complex problems (Medsker, 1995; Suran & Sukhdev, 1995a). Financial applications like investment analysis, credit monitoring require large volume of information to be accessed and processed to make the intelligent decisions in real-time. For such applications, the existing powerful database management systems can be used instead of developing specifically tailored data management subsystems within the development tools for building hybrid systems.

The purpose of this article is to suggest an approach as how intelligent systems can be coupled together using an SQL-database to build real-time distributed intercommunicating hybrid applications. This approach has been implemented in the shell environment 'HESS' (Hybrid Expert System Shell) that integrates expert systems and neural networks.

2. Review of related literature

Suran and Sukhdev (1995a) have classified the hybrid systems as:

1. function-replacing hybrids to replace the principal function of a given technique by another intelligent technique for technical enhancement;
2. intercommunicating hybrids are independent, self-contained intelligent modules that exchange information and perform separate functions to a general solution; and
3. polymorphic hybrids to exploit the multi-functionality of a technique.

Various strategies or models to integrate the expert system and neural networks and their applications have been explored (Medsker, 1994; Suran & Sukhdev, 1995b) along with the benefits of integration. The five different hybrid strategies have been explained by Medsker (1995). These are: (1) stand-alone models; (2) transformational models; (3) loose-coupling models; (4) tight-coupling models; and (5) fully integrated models. These strategies are variations of the types of hybrid systems classified by Suran and Sukhdev (1995a). Most of the applications appeared in the application survey by Medsker (1995) have used loose-coupling strategy. In loose-coupling the intelligent systems communicate through the data files while in the tight-coupling models, the information is passed via memory resident data structures rather than the data files. The tight-coupling has benefits of reduced communication overhead and improved run-time performance compared to loose-coupling.

* Corresponding author. Tel.: + 91-20-673080; fax: + 91-20-674478.
E-mail address: sonar@nibm.ernet.in (R.M. Sonar)

0957-4174/99/$ - see front matter
PII: S0957-4174(99)00020-2

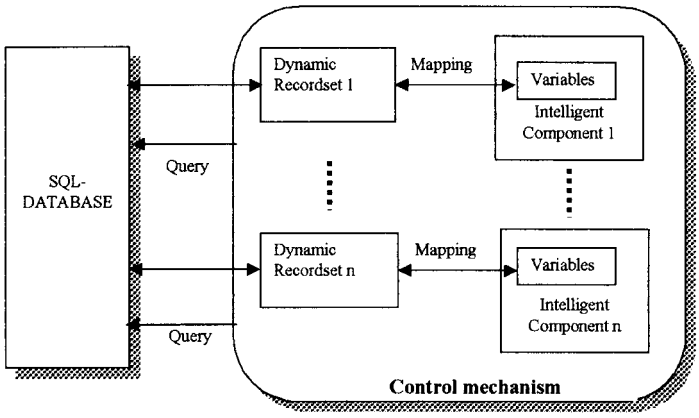

Fig. 1. A Framework to couple intelligent components.

Many researchers have suggested the approaches or ideas to integrate AI and databases (Bic & Gilbert, 1986; Amamolt & Nygard, 1995; Jarke & Vassiliou, 1984a). Vassiliou et al. (1983, 1985) and Jarke and Vassiliou (1984b) have described four ways to integrate expert systems and databases: (1) elementary data management within ES; (2) generalized data management within ES; (3) loose coupling of ES and existing DBMS; and (4) tight coupling of the ES and existing DBMS. Yang (1997) has given detailed review of the integration of expert system and databases and represented an approach to link existing database management systems and expert systems using a simple coupler. Yang (1997) classified literature as: (1) enhanced database systems that incorporate semantic-integrity constraints, adding rules into database and putting an ES component into database system; (2) enhanced expert systems to extend logic programming; (3) coupling existing ES and database; and (4) expert database systems that propose new constructs or apply new approach to represent data and knowledge in order to build new type of intelligent system.

Most of the research literature on integrating AI and databases represent the models to integrate expert systems and databases. There is a need to integrate other intelligent techniques like neural networks with databases.

3. Tools and environments for hybrid systems

Few commercial environments have been developed specifically for the creation of hybrid systems (Medsker, 1995) and some of the existing tools can be used to build hybrid applications. These development environments are suitable to develop intercommunicating hybrid systems. The machintosh based NeuX, offered by Charles River Analytics, Inc. facilitates the development of neural networks and expert systems that can be linked together via hypertext commands. NeurOn-Line a product by Gensym Corporation allows neural networks to be included with the expert system, database and other software. Most of the systems like Level 5 Object, ExSys provide access to databases and allow procedural codes to enable to incorporate execution of other intelligent modules inside the shell. The recent tools like ReSolver from Multi logic, USA combines the power of rule-based expert system with procedural language and the database access. ReSolver uses the blackboard architecture to share and exchange the data among the modules. The other hybrid tools and approaches to integrate intelligent systems have been discussed by Suran and Danny (1995). Some of the environments and approaches that are suitable to develop integrated system include use of 1 > Object oriented environments 2 > Microsoft's DDE (Dynamic Data Exchange) and OLE (Object Linking and Embedding) 3 > CORBA (Common Object Broker Architecture) 4 > Microsoft's COM (component Object Model).

4. The framework used to couple intelligent components

The Fig. 1 shows the framework used in HESS. The intelligent components are separate and communicate through the connected SQL-database. The components implemented are expert system and neural network. Each one is self-contained and independent. Each component can be used individually. The component is connected to the database through the dynamic recordset[1] (henceforth recordset) opened when an SQL-query is fired. The recordset acts as a memory buffer between a component and a database. The use of the recordset makes it possible to reflect the modifications immediately to the database as well as all associated open recordsets. The associated

[1] A CDaorecordset object in MFC (Microsoft Foundation Classes) represents a set of records selected from a data source, known as "recordset". Dynamic recordsets are the result of a query that can have updatable records. A dynamic recordset is a set of records that you can use to examine, add, change, or delete records from an underlying database table or tables. A dynamic recordset can contain fields (columns) from one or more tables in a database. The data types of columns or values can be determined at runtime.

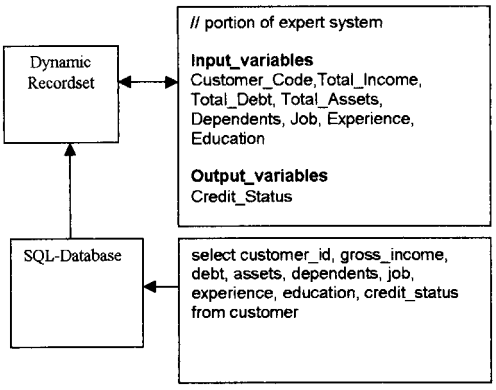

Fig. 2. Mapping of expert system variables with recordset columns.

recordset means the recordsets that have common part of database among them. However, any change in the database contents refreshes the associated recordsets automatically. The intelligent components can be integrated to share and exchange the information by linking them to the recordsets that overlap the parts of a database. For example a recordset say 'A' opened by firing the query 'select C1, C2, C3 from dummy' and the recordset say 'B' by the query 'select C2,C4 from dummy'. The recordset 'A' and the recordset 'B' will share all the values of column C2.

The controlling mechanism (1) controls the selection, editing and firing of queries, (2) links/unlinks the component to/from a particular open recordset, (3) loads the components into the memory, and, (4) executes the components. This control mechanism is an interactive and command-driven. However, the command language interface has been built to perform the operations that the control mechanism performs. The command language includes using various functions and programming constructs. The existing recordset can be refreshed sending a new query. The mapping is set between variables in intelligent component with the columns of the recordset to exchange the data. The control mechanism takes care of this exchange of the data between a component and a recordset. It maintains a pointer in each recordset. Whenever a component is initialized, the control mechanism sets the pointer to the first record in the linked recordset and increments it to the next record after each run. After the last record the pointer is set to the first record again.

HESS has been implemented using Visual C++ under Windows 95 environment using object oriented methodology. It uses Microsoft DAO (Database Access Object) API for database connectivity and to implement the recordset object. Use of DAO API makes it flexible to connect to heterogeneous databases and ISAM based software like spreadsheets, Foxpro.

4.1. Expert system component

The expert system component is a rule-based expert system integrated into the environment. The neural networks can be embedded into the expert system. Most of the features and syntax are same as that a typical expert system shell has. Some features are added to interact with the recordset. The mapping between recordset and the expert system is done, by setting the order of variables according to retrieved columns. The order is important and the names of the variables may not be same as the column names. The mapping is illustrated in Fig. 2.

During each session run, the expert system takes appropriate column values for input variables from the current available record in the recordset. After finishing the session, the output variables' values are updated to the recordset. The system can be run in batch-mode that takes entire set of records. The expert system determines the data type at run time. The data types of the columns of the recordset are also automatically recognized by the expert system at the run time. This eliminates the need to match the data types while defining the mapping between columns and variables, and, thereby makes the system independent of database scheme. However, when the values are updated into database the data types of the values should match with the column types. Whenever there is no value stored in the database for a particular data item in a record, the expert system prompts the user to enter the value interactively and updates the recordset. When the recordset is not linked, the system is capable of running in the question–answer mode. In this case it is proper to assign a question, data type of input when systems interactively asks the user to provide the value for it. The interesting feature of this system is that if the column name 'inference' occurs in the query, the system automatically stores the inference report in that column after each session. The type of column inference must be either text or memo while defining the database schema. This feature makes it possible to store the inference report for each record in the database itself.

4.1.1. Operations

1. Loading the rules.

   The user either opens the existing expert system rules' file or enters the rules using rule editor.
   The rules are loaded into the memory (after verifying for all syntax etc.) by executing load command.

2. Database access.

   If the user wants to have the database access, the input variables and output variables must be specified in the 'input_variables' and 'output_variable' section in expert system to map with the recordset columns.
   He/she can select the database and the control mechanism connects to it. After successful database connection, the names of tables existing in that database are displayed. The user either can select the table or write a query to open the recordset. If a table is selected, entire records from that table are fetched in the recordset and

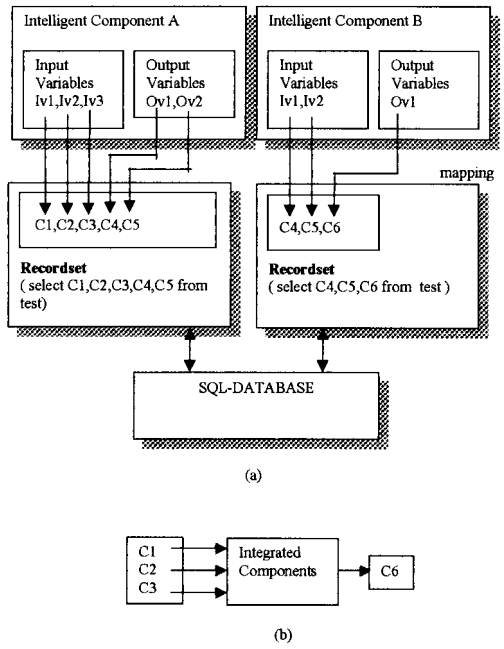

Fig. 3. (a) Schematic presentation of integration of components. (b) Input and output columns of the integrated system.

this recordset is linked to the expert system. If the expert system is already linked to a recordset, the control mechanism refreshes it with the new records.

3. Running the system.

   If the expert system is not linked to the recordset, it runs (may be backward or forward) in question–answer mode asking the values for variables as and when required during the run.
   If the expert system is linked to the recordset, the control mechanism can run the system in batch-mode or one session run at a time. The control mechanism sets the pointer to the next available record in the recordset for the next session run till the last record. After the last session run, the control mechanism sets pointer to the first record in the recordset. During the session run if any variable is not initialized, the system asks the user to enter the value.

4. Closing the recordset or the database.

The user can close the database or unlink the recordset from the expert system.

4.2. Neural network component

The second component is the implementation of multilayer feed-forward error-backpropagation neural network architecture. The neuron model is represented by a matrix element. All the algorithms are implemented using matrix as a major object. To interact with the recordset, the component is capable of directly interfacing the entire matrix or portion of matrix with the recordset. The variables used are all matrix types. In case of the expert system, the variables are assigned the values from the currently available record from the recordset. But in case of neural network, the matrix variable is shared with the entire recordset or the portion of the recordset. Depending upon the number of inputs and outputs defined in network topology the query should return the columns in that order. E.g. Inputs = 4 and output = 1 The query "select FinancialStatus, DebtStatus, JobStatus, EducationStatus, AnnCreditStatus from Customer" will take the first four columns as the input and the last column as output. (Note that in training mode one additional column should be specified to store the calculated outputs). The number of rows (vectors) of the matrix would be same as the number of records in the recordset. When training and testing is completed, the neural network topology along with the weights is saved to a project file to use it at the run time.

4.3. Command language interface

The command language is a small set of instructions and programming constructs. It is a run time environment to integrate the components after they have been trained and tested. These instructions perform the tasks that the control mechanism performs. The syntax of the command language instructions is simple and easy to understand. The instructions are executed in the interpreter mode.

4.4. Integrating the components

The components can be integrated by selecting the recordsets that have a common database portion among them. The columns selected may be output for some component and may be input for the other. When the first component is executed, it will make available the output for the second component. Fig. 3(a) illustrates how the components can be integrated. The component A takes input from columns C1–C3 of table 'test' and produces output into the columns C4 and C5. The component B takes input from columns C4 and C5 and produces the output into C6. The columns C4 and C5 are shared between component A and component B. The columns C1–C3 act as the main input columns while C6 as the final output column of integrated components as shown in Fig. 3(b). More than two components can be shared in this fashion.

4.4.1. Example

The following example illustrates how the expert system can be used to preprocess the data. Preprocessing the quantitative data and converting it into point scales reduces the search space of the neural network. This is because the number of parameters can be reduced. It also reduces the possible range of the values that a particular variable can take. For example 'Gross_Income' may take values within the broad range but when it is point scaled only few values are possible. Consider the simple example to determine the consumer credit status to grant the loan. The inputs are

Gross_Income, Debt, Net_Assets, Dependents, Job, Experience and Education. Instead of giving these inputs directly to the neural network these are preprocessed and converted into the output (only four variables instead of seven): FinancialStatus, DebtStatus, JobStatus, EducationStatus and then applied to the neural network. The part of expert system will look like:

```
If Job
and Experience > 5
then JobStatus is 3
// note that 3 means good
```

The following is the script representing the integration. Expert system 'credit.rb' is integrated to neural network 'credit.prj'. '//' and /*…*/ used for adding comments.

```
variables
// variable declaration
ExpertQry = Select Customer_ID, Gross_Income, Debt,
Net_Assets, Dependents, Job, Experience, Education,
FinancialStatus, DebtStatus, JobStatus, EducationStatus
from Customer
/* Customer_ID, Gross_Income, Debt, Net_Assets,
Dependents, Job,
Experience, Education are the inputs to the expert system
whereas FinancialStatus, DebtStatus, JobStatus and
EducationStatus store the outputs (goals). System assigns
quantitative values for goals (Good = 1, Medium = 2 and
Poor = 3)*/

ANNQry = Select FinancialStatus, DebtStatus, JobStatus,
EducationStatus, AnnCreditStatus from Customer
// Outputs from expert system are inputs to ANN. //
AnnCreditStatus is output
// column for ANN (number of inputs = 4 and number of
output = 1)

begin
OpenDatabase ("c:\expert\credit.mdb")
LoadExpert ("Credit.rb") // opens and loads the specified
//expert system into memory
LoadANN ("Credit.prj") // opens and loads ANN project
file // (trained neural network)
RunExpert (ExpertQry) // links to the recordset opened //
by ExpertQry and runs the ES
RunANN (ANNQry) // links to recordset using ANNQry
// and runs ANN
CloseDatabase ( )
end
```

The shell has been tested with smaller prototypes. The database management system used is MS-Access but system can be connected to any database having the database-driver installed.

5. Conclusions

This article represented a way to use an SQL-database to couple various intelligent systems. Use of SQL and the dynamic recordset makes it easy to retrieve, manipulate and share a required portion of a database. The SQL interface offers design flexibility and database independence. The approach suggested provides some kind of flexible and robust tight-coupling strategy using the features of database systems. The database management systems like MS-Access provides GUI based database management, form design and query processing. Instead of adding the separate data management subsystems, the existing database systems can be used to build the hybrid system tools. With some modifications to the existing intelligent system development tools, they can be integrated using the dynamic recordset object and the control mechanism discussed here.

References

Amamolt, A., & Nygard, M. (1995). Different roles and mutual dependencies of data information and knowledge—an AI perspective on their integration. *Data and knowledge, 16*, 191–222.

Bic, L., & Gilbert, J. (1986). Learning from AI: new trends in database technology. *Computer, 19*, 44–54.

Jarke, M., & Vassiliou, Y. (1984a). Coupling expert systems with database management systems. In W. R. Reitman (Ed.), *Artificial intelligence applications for business*, (pp. 65–85). Norwood, NJ: Ablex publishing.

Jarke, M., & Vassiliou, Y. (1984b). Database and expert systems: opportunities and architectures for integration. In G. Gardarin & E. Gelenbe (Eds.), *New applications of data bases*, (pp. 185). London: Academic Press.

Medsker, L. R. (1994). *Hybrid neural networks and expert systems.*. Dordrecht: Kluwer Academic Publishers.

Medsker, L. R. (1995). *Hybrid intelligent systems*. Dordrecht: Kluwer Academic Publishers.

Suran, G., & Sukhdev, K. (1995a). *Intelligent hybrid systems*. New York: Wiley.

Suran, G., & Sukhdev, K. (1995b). Intelligent hybrid systems: issues, classifications and future directions. In G. Suran & K. Sukhdev (Eds.), *Intelligent hybrid systems*, (pp. 1–2). New York: Wiley.

Suran, G., & Danny, S. (1995). Tools and environments for hybrid systems. In G. Suran & K. Sukhdev (Eds.), *Intelligent hybrid systems*, (pp. 275–308). New York: Wiley.

Vassiliou, Y., Clifford, J., Jarke, M. (1983). How does an expert system gets its data? Proceedings of the ninth international conference on VLDB, Florence, Italy (pp. 70–72).

Vassiliou, Y., Clifford, J., & Jarke, M. (1985). Database access requirements of knowledge-based systems. In W. Kim & D. S. Reiner & D. S. Batory (Eds.), *Query processing in database systems*, (pp. 156–170). Berlin: Springer.

Yang, H.-L. (1997). A simple coupler to link expert system with database systems. *Expert System With Applications, 12*, 179–188.