Pergamon

Expert Systems With Applications, Vol. 12, No. 2, pp. 179–188, 1997

0957-4174/97 $17.00+0.00

PII: S0957-4174(96)00093-0

A Simple Coupler to Link Expert Systems with Database Systems[1]

HENG-LI YANG

Department of Management Information Systems, National Cheng-Chi University, 64, Sec. 2, Chihnan Road, Mucha District, 116, Taipei, Taiwan

Abstract—*Over the years, there has been a confluence of concepts, tools and techniques from two diverse areas: artificial intelligence (AI) and database (DB) systems. There are several ways to integrate expert systems (ES) and database systems. This paper surveys the related literature and classifies the integrated systems into four classes: enhanced DB, enhanced ES, coupling of existing ES and DB, and expert database system. A new loose coupling approach (Simple Coupler) based on predefined SQLs is then proposed.*

Its system architecture and system operations are described. The Simple Coupler is compared with the DIFEAD (Dictionary Interface for Expert Systems and Databases) approach and the commercial ES shell approach on the criteria of the independence of the DB, ES, the complexities and future expansibility. This approach has great practical values because of its simplicity. Finally, this paper reports on two prototypes coupling existing systems: accounting DB & financial ES and medical history database & medical diagnosis system, respectively, in the PC window environment. © 1997 Elsevier Science Ltd

1. INTRODUCTION

A DATABASE (DB) SYSTEM is a computerized system whose overall purpose is to maintain a large collection of data representing facts and to make that information available on demand (Date, 1995). An expert system (ES) is an application of artificial intelligence (AI) techniques. It is a computerized advisory program that attempts to imitate or substitute the reasoning processes and knowledge of experts in solving specific types of problems (Turban, 1993). It contains a knowledge base. There are some distinctions between data and knowledge (Wiederhold, 1984, 1986). For example, data include much detail (specific instances), are voluminous and will change over time. They appear in reports that are used mainly for operational purposes and can be objectively verified for correctness. On the other hand, knowledge is typically subjective, relates to generalization, will not change as frequently and is significantly smaller than data (owing to summarizing the voluminous material).

These two kinds of systems have become prevalent for decades. However, traditionally, there has been little interaction between researchers in the areas of AI and databases. Recent trends have shown that techniques or ideas developed in each of these areas may be adapted for use in the other (see, for example, Al-Zobaidie & Grimson, 1987, 1988; Amamodt & Nygård, 1995; Bic & Gilbert, 1986; Jarke & Vassiliou, 1984a, b; Kennedy & Yen, 1990; Kerschberg, 1990; Missikoff & Wiederhold, 1986; Rundensteiner, 1990; Stonebraker & Hearst, 1989; Vassiliou et al., 1983, 1985; Zhao, 1994). Bic (1984) has indicated that the interdisciplinary cooperation of AI and DB researchers represents a promising effort to provide the computational power necessary to support a highly intelligent system.

This paper presents an approach (*Simple Coupler*) based on predefined SQL codes to couple existing database systems with existing expert systems. Because of its simplicity, this approach would have great practical values for a number of businesses. In the following, the paper is divided into six sections. Section 2 briefly reviews related research. Section 3 presents the overall architecture of the Simple Coupler approach. Sections 4 and 5 compare this approach with the DIFEAD approach and ES shell approach, respectively. Section 6 describes the implemented prototypes and Section 7 concludes the paper.

2. REVIEW OF RELATED RESEARCH

Vassiliou et al. (1983, 1985) and Jarke and Vassiliou (1984a,b) have described four ways that expert systems can access databases: (1) elementary data management

[1] The author wishes to thank anonymous reviewers for their helpful comments.

within the ES; (2) generalized data management within the ES; (3) loose coupling of the ES with an existing DBMS; and (4) tight coupling of the ES with an existing DBMS. One should note that they used "loose coupling" to mean *static* coupling, i.e. data extractions occur statically before the actual operation of the ES; and "tight coupling" to mean *dynamic* coupling, i.e. during the same ES session, many different portions of the external database may be required at different times. In contrast, other researchers (e.g. Kerschberg, 1989; Parker, 1989; Sheth, 1989; Stonebraker & Hearst, 1989) have used these two terms differently. For example, Sheth (1989) defines an integration to be "*loose coupling*" if the following hold:

(1) there is a well-defined interface between the two systems;
(2) if a system being integrated is **preexisting**, it is not changed or the changes are very minor.

This is the definition of "loose coupling" that this paper adopts. Under this definition, a loose coupling could be either static or dynamic.

Jarke and Vassiliou (1984a,b) have also mentioned some ways to enhance databases by expert systems methods: e.g. intelligent database interfaces and intelligent database operation (e.g. query optimization and transaction management). Based on these discussions, researchers (Al-Zobaidie & Grimson, 1987; Jarke & Vassiliou, 1984b) classified these integrated systems into three classes: (1) intelligent DB; (2) enhanced ES; and (3) inter-system communication.

Rundensteiner (1990) has proposed an *AI–DB connection hierarchy* to summarize these types of interdisciplinary research. This hierarchy is based on three features, which are techniques (concepts), applications (features) and hardware. There are five different levels (stages of cooperation) in this hierarchy depending on these three features. The basis of this hierarchy, level zero, represents the pure fields (pure DB or pure AI applications). Level one indicates the exchange of hardware: the use of special-purpose AI machines for DB applications or the development of AI techniques on a special-purpose database machine. Level two signifies a practical cooperation of these two fields, the borrowing of techniques, methodologies and ideas from the one area to fulfil the respective tasks of the other. Compared with the research (Al-Zobaidie & Grimson, 1987; Jarke & Vassiliou, 1984b) in the preceding paragraph, his level three actually corresponds to the intelligent DB and enhanced ES, and level four concerns inter-system communication.

However, he has also recognized a new approach of research: to design a new generation of intelligent systems, what he called "knowledge based management systems", providing highly efficient management of large, shared knowledge bases. This new level is called level three-and-half.

Based on the above literature, this paper classifies the integrated systems into four classes. The classification here is more complete than the one mentioned by Jarke and Vassiliou (1984a,b); and succinctly covers Rundensteiner's (1990) ideas. The following is a brief listing of the related research.

(1) *Enhanced Database Systems.* These include a deductive database approach (Kellogg, 1986; Gallaire et al., 1984; Han, 1994; Seljée, 1995), incorporating more semantic integrity constraints (Morgenstern et al., 1986; Mylopoulos & Brodie, 1985; Yang, 1992; Grefen & Apers, 1993), adding rules into DB (or mining rules from DB) (Amamodt & Nygård, 1995; Agrawal et al., 1993; Dayal et al., 1995; Houtsma & Swami, 1995; Stonebraker & Kemnitz, 1991; Stonebraker, 1992; Zhao, 1994) and putting an ES component into a DB system (Smith, 1986).
(2) *Enhanced Expert Systems.* These mainly involve extending logic programming (Brodie & Jarke, 1986; Parker et al., 1986; Tsu, 1988).
(3) *Coupling of Existing ES and DB.* There are three possible architectures of coupling the existing ES and DB (Al-Zobaidie & Grimson, 1987; Jarke & Vassiliou, 1984a,b; Rundensteiner, 1990): (1) distributed processing and control; (2) concentration of processing and control (one system assumes a dominant role); and (3) distributed processing controlled by an independent subsystem. Among these three possible ways of coupling DB with ES, the way with distributed control would create problems of inconsistency and redundancy; and the way with one dominant control would restrain the flexibility of the system (Rundensteiner, 1990). The method of an independent interface subsystem is to guarantee high flexibility without problems of inconsistency and redundancy. It also satisfies the above definition of loose coupling. DIFEAD (Dictionary Interface for Expert Systems and Databases) is one example adopting this approach (Al-Zobaidie & Grimson, 1987, 1988).
(4) *Expert Database Systems.* This class includes those proposing new constructs or applying a new approach to model and represent data and knowledge in order to build a new type of intelligent system. For example, Missikoff and Wiederhold (1986) have proposed to manage knowledge consisting of both facts and rules in a uniform, database-like frame (the knowledge organization, "kluster", is used in parallel with "relation" used in a relational database). Risch et al. (1988) have proposed a functional approach to integrating databases and expert systems. Higa et al. (1992) (see also Sheng & Wei, 1992) have applied the object-oriented approach to incorporate data and knowledge (including general semantic knowledge and task-specific semantic knowledge). Those researches (Anwar et al., 1993; Buchmann et al.,

1995; Dillon & Tan, 1993; Frost et al., 1994) on rule support for object-oriented databases may also be classified into this class since they are based on the new object construct. Kerschberg (1990) has mentioned the necessity of having a unified and formal knowledge/data model for creating knowledge schemes that express the semantics of both knowledge and data. He also suggests the knowledge encyclopedia that represents both system and application knowledge in an object-oriented view, in which object behavior is specified by explicit constraints.

The approach adopted by this paper is loose coupling. The main advantage of this approach is that one could make use of already existing systems instead of having to reconstruct them. As stated by Sheth (1989), loose coupling is a political and economic necessity. It is a short term solution before the goal of achieving expert database systems—it may meet some of the immediate needs but fall short in terms of the expressiveness, uniformity in representation and/or efficiency (Sheth, 1989; Stonebraker & Hearst, 1989).

3. SIMPLE COUPLER APPROACH

The system architecture proposed here is shown in Fig. 1. It is an integrated computer system to couple an existing *DB system* with *ES* through a *coupling module* in a *multi-tasking environment*. A multi-tasking platform, e.g. UNIX at workstations or MS-Windows 3.1 at personal computers (PCs), is necessary because there are actually three programs (coupling module, DB application and ES) running in a memory. The communication between the coupling module and both the ES and the DB is through some channels, e.g. UNIX pipes, or MS-Windows 3.1 DDE [Dynamic Data Exchange (DDE) or Object Linking & Embedded (OLE)].

These two existing systems are independent. The ES consists of its own *user interface, inference engine, knowledge base, working memory (workplace)* and possible others (e.g. *explanation subsystem*). The DB

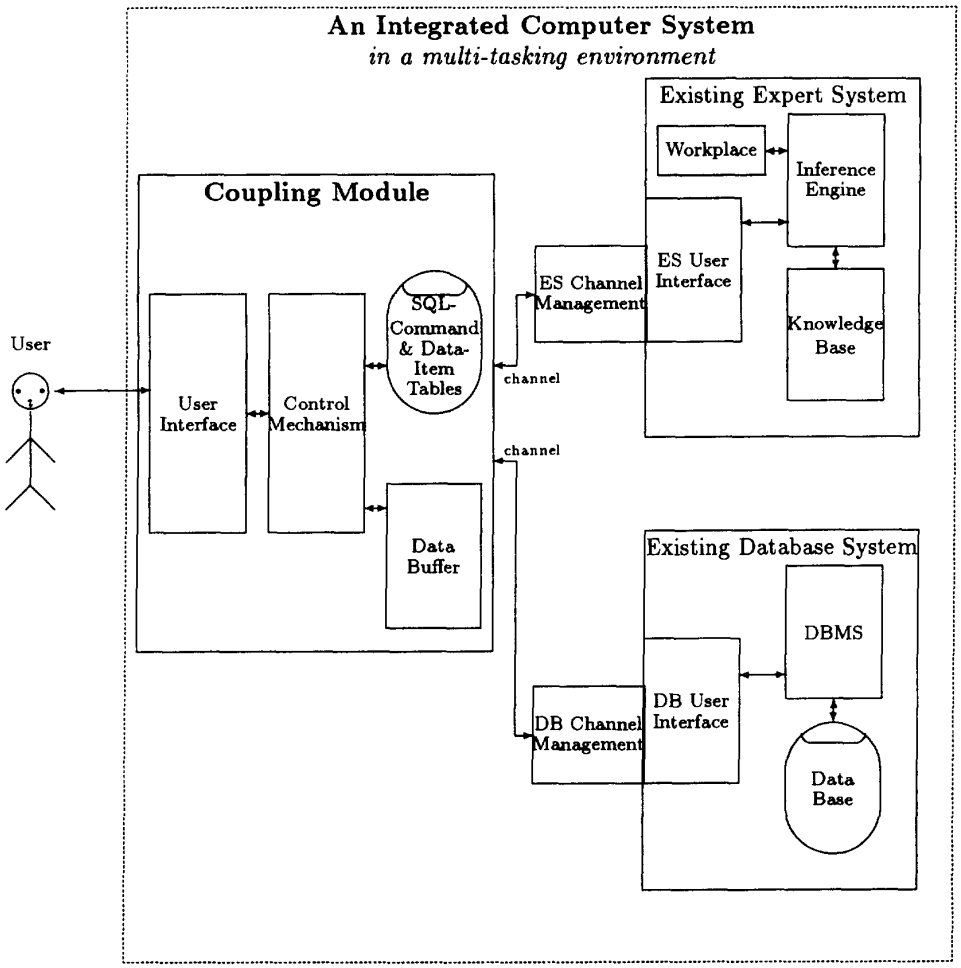

FIGURE 1. System architecture.

system consists of its own *user interface, data base management sub-system* and *data base* in a secondary storage. The only minor modifications are two add-on channel management modules. The functionalities of these two add-on modules will be explained in Sections 3.5 and 3.6 after the coupling module is described.

The coupling module consists of four components: *user interface, control mechanism, SQL command & data-item tables* and *data buffer*. They are explained as follows:

3.1. Coupling User Interface

There are four choices in the coupling user interface: (1) execute a DB application; (2) consult an ES; (3) coupler system maintenance; (4) quit. From the point of view of an end-user, this integration is transparent. The end-user interacts with this user interface to use either the DB or ES. The control will be transferred to either of these systems appropriately. Only system administrators have the privilege of conducting coupler system maintenance. The maintenance includes setting user privileges and updating SQL command & data-item tables.

3.2. SQL Command & Data-Item Tables

These tables are located in a secondary storage. The contents of tables are pre-loaded by system administrators before constructing the coupling system, but can be modified in later usages. The SQL command table consists of two types of code information.

3.2.1. For Query and Update

It stores canned SQL codes for each data item requested in the ES. For example, suppose that an ES for financial diagnosis uses two variables, *X-Cash* and *X-Inventory*, to denote the current balance of cash and inventory, respectively, during its inference. Also assume that its coupling accounting DB contains the relation *detailed account* with corresponding column names "cash" and "inventory". The appropriate SQL codes would be pre-stored (Table 1) as parts of the SQL command table.

It is assumed that the names of the data variables in the ES are unique, i.e. each name in the whole ES represents a single thing[2]. When the ES asks for the value of a specific data item, the control mechanism of the coupling module would check this table, obtain the corresponding SQL codes and send them to the DB application to fetch the value.

If some data (e.g. *cash* in the above example) are missing in the DB application, the DB application would return a blank or "null" value (depending on the DBMS). In this case, the user interface of the ES would prompt the user. If the user provides a value (say, 50,000) for the missing data, ES would notify the coupling module. The coupling module could use the same table to obtain the corresponding SQL codes and rewrite into the appropriate update codes (e.g. the following), and pass them to the DB.

```
update detailed_account set amount
  =50,000 where name="cash"
```

Note that the data variables used in the ES might not be the same as the ones in the DB. They might even be some aggregate (e.g. minimal or average) values or even the results of some complicated formula of the data items in the DB.

3.2.2. For Insertion

The ES (e.g. for medical diagnosis) might derive some useful conclusions and can be stored for future reference. In this case, the SQL command table (Table 2) contains a second type of code information that are parameterized codes rather than actual codes.

The control mechanism of the coupling module would also need another table, *data-item table*, mapping the data variables used in the ES into columns of relations in the DB. For example, suppose that an ES for medical diagnosis used by a doctor (John, variable name: *X-Doctor* in the ES) reached a conclusion that today (7/7/95, variable: *X-Date*) a patient (Mary, variable: *X-Patient*) might have "cholera" (variable: *X-Disease*), but does not know any other information (those *other-1*, *other-2* in the relation *diagnosis* of the DB). The control mechanism would first check the data-item table (Table 3) to find the corresponding column names, *doctor*, *date*,

TABLE 1
Part of the SQL Command Table—Example 1

X-Cash	select amount from detailed_account where name="cash"
X-Inventory	select amount from detailed_account where name="inventory"

TABLE 2
Part of the SQL Command Table—Example 2

Insert X-Diagnosis	insert into diagnosis (patient, doctor, disease, date, other_1, other_2)

TABLE 3
Part of the Data-Item Table—An Example

X-Doctor	diagnosis, doctor
X-Date	diagnosis, date
X-Patient	diagnosis, patient
X-Disease	diagnosis, disease

[2] Some ES might be constructed from a language like Prolog. In those cases, the scope of a variable in each rule is limited to that rule. A possible solution might then be prefixing a rule number to the variable name.

patient, disease of the relation *diagnosis* in the DB.

Then, it would check the SQL command table to get the appropriate parameterized codes and rewrite them into the following, and pass this on to the DB:

insert into diagnosis (patient, doctor, disease, date)
values ("Mary", "John", "cholera", 7/7/95)

The control mechanism should notify the user of adding the missing values for other information (those *other-1*, *other-2* in the relation *diagnosis* of the DB). The data-item table only stores the mapping of those data that will be inserted into the DB as new records. It would not be needed in the cases of query and update. However, if the original SQL command table contains some SQL codes for fetching multi-column values for some variable (e.g. all liquid assets which account IDs are prefixed with "11"), the data-item table is also needed for updating missing data.

3.3. Data Buffer

The data buffer is a working memory. It stores those temporary data between the ES and DB, which include query results from the DB, missing data obtained from the ES or inference conclusions from the ES, etc.

3.4. Control Mechanism

The control mechanism is responsible for all communication between the ES and DB and for performing the maintenance of the SQL command & data-item tables. It would pick up a request, fetch or format SQL commands, transfer control to either ES or DB, and receive responses. The functionalities of the control mechanism might be better understood after the system operation flows of this integrated computer system are described.

3.5. ES Channel Management Module

The existing ES needs a minor modification to add this ES channel management module. This module establishes the channel and deals with the communication protocols between the ES and the coupling module. When an ES needs some data, it first requests this channel module to provide the value(s). Then, the channel module transmits the data request to the coupling module. The data query results from the DB may be blank (or "null"), single-value, or multi-values (including multi-columns or multi-rows, or both). If the coupling module cannot get the data from the DB (the reply is blank or "null"), the channel module should pass the data request to the user interface of the ES. If the reply includes multi-column or multi-row values, the channel needs to pre-process it (partition them into several single-values, and return them to the appropriate variables). The data missing from the DB and obtained from the user through the user interface of the ES or the conclusions by the ES will also be passed through this channel module to the coupling module in order to be stored in the DB.

3.6. DB Channel Management Module

The communication between the coupling module and the DB needs some cooperation from the DB. The DB channel management module receives commands (queries, updates, or insertions) from the channel and passes the query results back to the coupling module. In fact, most of large database servers (e.g. Sybase) have already such functionalities, and need no modifications. However, for those PC-based DB applications, some modifications are needed to establish this channel management module.

3.7. System Operation Flows

To give the whole picture, the operation flows of this integrated computer system are described as follows.

The entry point to this integrated computer system is the user interface menu of the coupling module. There are four scenarios:

(1) **A user chooses to use the DB**
   (a) The control mechanism calls the DB application program and transfers the control right to it. The coupling module is now idle.
   (b) The DB invokes its own user interface menu.
   (c) Within his (her) privileges, the user performs any operation on the DB.
   (d) When the user quits from the DB, the coupling module resumes the control right.

(2) **A user chooses to use the ES**
   (a) The control mechanism calls the ES application program and transfers the control right to it. The coupling module is now idle.
   (b) The ES begins its inferences based on the facts it knows.
   (c) Whenever the ES needs to instantiate some data variable, if its value is not in the working memory, the ES would ask its channel management module for data.
   (d) The ES channel management sends a data request to the coupling module.
   (e) The control mechanism of the coupling module checks the SQL command table to find the appropriate SQL codes for such a data item. If the control mechanism could not find such data entry, a negative reply would be sent back to the ES; the user interface of the ES would prompt the user for data; and the operation flows are branched to (f). If the appropriate SQL codes are found, the system continues the following substeps:
      (i) The control mechanism sends the appro-

priate SQL codes to the DB.
(ii) The DB performs the SQL query.
(iii) The DB sends the query results back to the coupling module.
(iv) The control mechanism sends the value(s) of data back to the ES.
(v) The ES channel management module formats the value(s) of data for the inference engine.
(vi) If any data value is missing, the user interface of the ES prompts the user for data.

(f) The inference engine of the ES performs inferences and reaches a conclusion.
(g) If there are value(s) of any data missing from the DB which can be obtained from the user, or conclusions by the inference engine which can be stored for future reference, the ES channel management module would send these types of information to the coupling module and continue the following. Otherwise, the control just goes back to the coupling module.
(h) The control mechanism of the coupling module checks the SQL command & data-item tables, rewrites the appropriate SQL codes, and sends the update or insertion requests to the DB.
(i) The DB performs the update or insertion.
(j) After the DB finishes the update/insertion, the control goes back to the coupling module.

(3) **A system administrator chooses to maintain the coupler system**
The control mechanism of the coupling module allows the update of the SQL command & data-item tables through its user interface. However, in the current system architecture, there is still no facility to help the system administrator to fill in the tables. The system administrator is responsible for both the syntax and semantics of the SQL codes.

(4) **A user chooses to quit**
Either the end-user or the system administrator just quits from this integrated system.

Note that the protocols between the ES and the coupling module might not be the same as the above. For example, we might change the step of (2.e.iv). The control mechanism of the coupling module might not send the value(s) of data back to the ES when the DB finishes the query. Rather, the ES could actively try to fetch the data from the coupling module. However, in this case, timing sequences of the above steps of (2.e.i) to (2.e.iv) become important. If the ES fetches the data too fast, it might get data that are actually results of the last query or the default values pre-set by the system. There would be a flag in the coupling module, which would be set to some certain value when the query results are sent back from the DB. The channel management module of the ES could monitor some flag(s) of the control mechanism to determine the availabilities of the requested data.

4. COMPARISON WITH THE DIFEAD APPROACH

This section compares the Simple Coupler approach with the DIFEAD approach (Al-Zobaidie & Grimson, 1987, 1988).

- **The independence of the DB**
In both approaches, the DB application can exist independently. All accesses to the DB are passed by the coupling module to the database management facility.

- **The independence of the ES**
In the DIFEAD approach, the ES can be accessed through its kernel, the MLC (Metalevel Component). There are three main modules in the MLC: UIM (User Interface Module), MQM (Metadata Query Module) and DUM (Data Update Module). The UIM interacts with the user. The MQM finds out whether some data item can be found in the application or not. If no answer can be found in the application DB, the MQM informs the UIM to request the missing information directly from the user. In such a case, all data inputs of the ES are through MLC. This would decrease the independence of the ES.
However, in the Simple Coupler approach, there is still a user interface component in the ES. It only needs a minor modification to add a channel management module.

- **The complexities of the system**
In the DIFEAD system, the MLC uses two types of information—dictionary data and control data—to format a DB query to the application DB through its DBMS. It looks more flexible, but more complex in terms of efficiency. In the Simple Coupler approach, the SQL command & data-item tables are pre-defined by the system administrator. It would be less complex and more efficient when executing a data query[3]. This is why we call it *Simple* Coupler. The disadvantage of this approach is that it is not suitable for a loosely coupled system in which there are *ad hoc* queries to the database during the ES consultation. However, its simplicity deserves practical attention[4]. It would help a

[3]The time complexity of searching the SQL command table and data-item table and rewriting the appropriate SQL codes for a single requested data item could be linear to the size of the table or less.
[4]For those ES applications for which queries for the DB can be planned in advance, the designer may then decide either static or dynamic coupling. The Simple Coupler approach can be applied in either case. However, in the case of dynamic coupling, the concurrency control issue might first be solved because the database may be updated when the ES would be using out-of-date information that was queried from the DB and stored in a cache.

number of businesses to quickly gain benefit from integrating DB and ES.

- **The future expansibility of the system**
  In the DIFEAD system, the MLC itself is an independent database application. For example, its prototype runs as an application under Troll DBMS, and also interfaces to a Troll DB application (a clinical DB). Though the DIFEAD's authors expected that the architecture of DIFEAD could include an interface to a different DBMS, such an expansibility would be limited by the functionalities of the DBMS on which the MLC is based.

  In the Simple Coupler, the coupling module is just a program that can be constructed by any language. It could be expanded to include some knowledge so that it could even become an expert system (e.g. having a more intelligent user interface). In addition, the SQL is carefully chosen as the interfacing language between the coupling module and the DB because it has ANSI standards (SQL1—ANSI 1986, SQL2—ANSI 1992, and the planned SQL3). Most relational DBMSs have some versions of SQL which are compatible to ANSI standards to a certain extent. No modification or minor modification of the SQL commands table may be necessary while we change from one DBMS to another[5]. Such an expansibility is very useful in a heterogeneous or multi-database environment (Litwin, 1994).

5. COMPARISON WITH THE COMMERCIAL ES SHELL APPROACH

In commercial markets, there are many ES shells (e.g. EXSYS) which have links to databases (e.g. dBase, Orcale, etc.). However, those links are specially established by vendors. The following compares the Simple Coupler approach with the ES shell approach.

- **The independence of the DB**
  In the ES shell approach, the ES must usually know the *physical* data and file structures of the external databases, and uses special commands to link with them. In the Simple Coupler approach, the coupling module only needs to know the *logical* table (which might even be a virtual "view") structure of the database, and uses SQL tables to link them. Therefore, the independence of the DB application would be better. Besides, through the coupling module, the user can transparently access either the ES or the DB in the Simple Coupler approach. This is not true in the ES shell approach.

- **The independence of the ES**
  If a user wants to adopt the ES shell approach, he (she) must carefully choose the ES shell since not *every* ES shell has links to *any* database. Therefore, the independence of the ES is severely impaired. Moreover, there is no solution to the existing ES application. On the other hand, in the Simple Coupler approach, the ES only needs a minor modification to add a channel management module. Then, through the coupling module, it can access any SQL-compatible (or ODBC-based) databases. The framework is more open.

- **The complexities of the system**
  This is only point on which the ES shell approach may be better since the database links have been specially built-in by vendors and no add-on coupling module is needed. However, as stated in Section 4, the Simple Coupler is also less complex.

- **The future expansibility of the system**
  In the ES shell approach, the expansibility is limited to future support from the chosen vendor. In the Simple Coupler approach, as stated in Section 4, the coupling module can be enhanced and the system can also switch to other databases.

6. IMPLEMENTED PROTOTYPES

There are two prototypes in this research project. The first couples an accounting database with a financial diagnosis ES. The second couples a medical history database with a medical diagnosis ES. In the first prototype, the data flows are one-way, from DB to ES, because the financial ES would need data from the accounting DB, but its derived conclusions need not be stored in the accounting DB. In the second prototype, the data flows are two-way, between DB and ES, since the diagnosis conclusions need to be stored back in the medical history database for future reference; there might be some missing data while the patient is first registered. The first prototype was finished within 1–2 months by a graduate student. Similarly, the second prototype was also finished in a short period of time. They demonstrated the simplicity of the approach. This approach is very suitable for the ES application for which queries for the DB can be planned in advance.

Both prototypes are implemented in a MS-Windows 3.1 (Chinese version) environment on a PC. The coupling modules are written in Visual Basic Professional version 3.0. The main reason of choosing Visual Basic is its ease of use, so we can build the

[5]We can also switch from SQL to ODBC API functions. The ODBC (Open Database Connectivity) is an interface for heterogenous databases. This is proposed by Microsoft Co. and supported by a number of database vendors, e.g. DB2, Oracle, etc. While we are accessing a heterogenous database in a network, through ODBC's Dynamic Linking Library and database drivers, the ODBC API functions will be transformed into corresponding database commands.

prototype in a relatively short time[6]. Because suitable existing DB and ES could not be found in the commercial market[7], the "existing" DB (implemented by Fox Pro for Windows version 2.5)[8] and ES (implemented by Knowledge Pro Windows version 2.35)[9] are taken from two groups of undergraduate student projects. Note that though Visual Basic could efficiently access those database files (with extensions "dbf") through a special database engine JET, this research did not construct the prototypes in this way. The communication channels between the coupling module and both DB & ES are all through Windows' DDE. This decision was made based on two reasons: (1) to maintain the independence of the DB; and (2) to prevent the feasibility of our prototypes from depending on the special software—Visual Basic.

A short introduction for Visual Basic is given in this paragraph. The operation of a Visual Basic program is based on "objects" and "event procedures". There are two types of objects: form (the body of a window) and control (the components of a window, including text box, label, button, etc.). An object has many properties, including name, size, content (text), color, etc. the values of which are pre-defined by the Visual Basic and can be changed by the programmers. Either "user events" or "system events" may occur in any object. Programmers must write some codes (as an event procedure) to define the actions when some event occurs.

The following paragraphs will briefly describe the first prototype. The construction of the second prototype is similar.

In the first prototype, there is only one form (called, say, *form 1*) which includes four text boxes, four command buttons and three event procedures. Owing to the features of Visual Basic and Knowledge Pro, the alternative protocol (on p. 184) is taken here. That is, the channel management module of the ES monitors the flags ("*Text3*" and "*Text4*" in the following) of the control mechanism to determine the availability of the requested data. The four text boxes are invisible to the user.

- *Text1:* is a data buffer to receive the query results from the accounting DB.
- *Text2:* is another data buffer to temporarily store the database commands from the ES.
- *Text3:* is a flag for checking the SQL command table. If the requested data-item can be found from the SQL command table, then this flag is set to 1; otherwise it is set to 0 (the financial ES would then know the need for prompting the user).
- *Text4:* is a flag for indicating whether the query results are already placed in *Text1*. The financial ES would continue monitoring this flag.

The four command buttons are the four choices to use the financial ES, the accounting DB, maintain the coupler system, and quit the system. The four event procedures respond to the occurrences of user events—clicking the four command buttons, respectively. When the value of *Text2* changes, another event procedure, *Text2_Change*, would check the SQL command table and send the SQL codes to the DB through DDE. In this link, the coupling module is the "client" and the accounting DB is the "server" of the DDE. A test version of the Visual Basic codes of *Text2* may look as follows:

```
Sub Text_Change ()
For i = 1 to 999
If table(i) = Text2.Text
Then sqlcmd = Sql(i)
   Text3.Text = 1
   Text1.LinkTopic = "Account | C:\Account;sqlcmd"
   Text1.LinkItem = "data"
   Text1.LinkMode = 2
   Text1.LinkRequest
   Text4.Text = 1
   exit for
End If
Next
End Sub
```

Explanation:

Assume that two arrays, *table(i)* (storing variable names) and *Sql(i)* (storing the corresponding SQL codes), are used to implement the two columns of the SQL command table (Table 1). When the value of *Text2* changes, *Text2_Change* would check the SQL command table. If found, it stores the SQL codes (e.g. selects the amount from the detailed_account where name="cash" into a working variable "sqlcmd", and sets the flag *Text3* to 1. Then, it establishes the DDE channel. In the *Link Topic*, "*Account*" is the name of the accounting DB application program and "*C:\Account*" is its path on a PC. The "*data*" is the DDE link item indicating that the return through DDE would only include query results, without column names. The *Link Mode=2* establishes a cold linkage, which means that after the coupling module receives the SQL query results, even when the requested data changes, the accounting DB would not actively notify the coupling module of the changes. (If this notification is desirable in other applications, *Link Mode = 1.*) The coupling module then receives the SQL query results by the command "*Text1.LinkRequest*", and sets the flag *Text4* to 1.

The DB channel management module should establish the DDE service, define the DDE actions, valid topics and callback functions. For example, the following command is used to define a DDE server, named

[6]From the efficiency point of view, we should choose C or C++. But efficiency is not the focus of our prototype stages.

[7]To be qualified as a suitable existing product, its source codes should be available for minor modification.

[8]Fox Pro, Visual Basic, and Windows are products of Microsoft Co.

[9]Knowledge Pro is a product of Knowledge Garden Inc., USA.

"Account":

=DDESetService("Account","define")

The following command is used to allow the "request" DDE actions:

=DDESetService("Account","request",.T.)

The following is used to set a valid topic called "SQL" and assign a callback function "cbSQLTopic" to process:

=DDESetTopic("Account","SQL","cbSQLTopic")

There is a demo program called "FOXDATA.PRG" in the Fox Pro distribution package. Programmers can easily modify it for their purposes.

The ES channel management module should establish the DDE channel, send the request data-item variable to the coupling module and receive the SQL query results from it. In this case, the financial ES is the DDE "client", and the coupling module is the DDE "server". The codes in Knowledge Pro may include the following to establish a DDE channel called *ChData*, where *coupler* is the name of coupling module, *form1* is the form name of the Visual Basic codes. The *VBTopic* is a DDE topic that is defined when the DDE channel is opened using dde_open. Whenever a DDE event occurs, this topic is called to pass information about the event.

ChData is dde_open(VBTopic,coupler,form1)

The following is used to write the data-item name into *Text2* of the coupling module. The *first(?Var)* contains the name (e.g. "XCash") of the parameterized data-item *(Var)*[10]

dde_write(?ChData,text2,first(?Var))

The following is used to request SQL query results from *Text1* of the coupling module.

dde_request(?ChData,text1)

As stated, the alternative protocol (on p. 184) has been adopted in this prototype. The ES channel management module checks "*Text3*" to determine whether the name of the requested data-item can be found in the SQL command table. It also continues monitoring "*Text4*" of the coupling module to determine whether the SQL query has been completed and results have been available. The time interval between two successive tests and the total number of tests depends on the performance of the hardware and application software.

In the second prototype, because Fox Pro does not support the full SQL standard—does not recognize the SQL UPDATE keyword—the traditional dbase command "REPLACE" is used instead.

[10]The "?" mark before a variable in Knowledge Pro returns its value.

7. CONCLUSIONS AND FUTURE RESEARCH

This paper has surveyed related literature to integrate ES and DB. A new loose coupling approach (Simple Coupler) based on SQL has been proposed. Compared with the DIFEAD approach, the Simple Coupler would not decrease the independence of the ES, should be less complex and more efficient when executing a data query, and have high future expansibility in a heterogeneous or multi-database environment. Compared with the commercial ES shell approach, its advantages are obvious in terms of the independence of the DB and the ES, and future system expansibility.

Two prototypes (in Visual Basic) coupling an existing accounting DB & financial ES and an existing medical history database & medical diagnosis system, respectively, through MS-Windows DDE channels have been implemented. These prototypes have shown that it is easy to implement this coupling. It would be of benefit to a number of businesses that had problems in applying DB and ES in an integrated environment. In the future, this research will continue to explore the possibilities of implementing this system architecture in a network of multi-databases and in a multi-ES environment. There are a number of related research topics, e.g. concurrency control, communication protocols, in these truly heterogeneous network environments. In addition, the research on providing some facilities (e.g. verification and validation for correctness of syntax and semantics) to construct and maintain the SQL command table is interesting. Another direction of the related research would be to further enhance the coupling module as an intelligent front end to handle the query optimization or semantic integrity constraints of the DB. Furthermore, when querying the DB, if the data are not here, the coupling module might be able to apply its own rules or the ES rules to the available data to see if the requested information could be derived based on the data stored in the DB.

Acknowledgement—The author wishes to thank anonymous reviewers for their helpful comments.

REFERENCES

Agrawal, R., Imielinski, T., & Swami (1993). Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, DC (pp. 207–226).

Al-Zobaidie, A. & Grimson, J. B. (1987). Expert systems and database systems: How can they serve each other?. *Expert Systems*, **4**, 30–37.

Al-Zobaidie, A. & Grimson, J. B. (1988). Use of metadata to drive the interaction between database and expert systems. *Information and Software Technology*, **30**, 484–496.

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

Anwar, E., Maugis, L., & Chakravarthy, S. (1993). A new perspective on rule support for object-oriented databases. *Proceedings of the*

ACM SIGMOD International Conference on Management of Data, Washington, DC (pp. 99–108).

Bic, L. (1984). The fifth generation grail: A survey of related research. *Proceedings of ACM Annual Conference: The Fifth Generation Challenge* (pp. 293–296).

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

Brodie, M. L., & Jarke, M. (1986). On integrating logic programming and databases. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the First International Workshop)* (pp. 191–207). Reading, MA: Benjamin/Cummings.

Buchmann, A. P., Zimmermann, J., Balakeley, J. A., & Wells, D. L. (1995). Building an integrated active OODBMS: Requirements, architecture, and design decisions. *Proceedings of the Eleventh International Conference on Data Engineering*, Taipei, Taiwan (pp. 117–128).

Date, C. J. (1995). *An introduction to database systems* (6th ed.). Reading, MA: Addision–Wesley.

Dayal, U., Hanson, E., & Widom, J. (1995). Active database systems. In W. Kim (Ed.), *Modern database systems* (pp. 434–456). Reading, MA: Addison–Wesley.

Dillon, T., & Tan, P. L. (1993). *Object-oriented conceptual modeling*. New York: Prentice–Hall.

Frost, R. D., Gillenson, M. & Kilpatrick, M. G. (1994). *Journal of Object-Oriented Programming*, **5**, 31–36.

Gallaire, H., Minker, J. & Nicolas, J. (1984). Logic and databases: A deductive approach. *Computing Surveys*, **16**, 153–174.

Grefen, P. & Apers, P. (1993). Integrity control in relational database systems—An overview. *Data & Knowledge Engineering*, **10**, 187–223.

Han, J. (1994). Constraint-based query evaluation in deductive databases. *IEEE Transactions on Knowledge and Data Engineering*, **6**, 96–107.

Higa, K., Morrison, M., Morrison, J. & Sheng, O. R. L. (1992). An object-oriented methodology for knowledge base/database coupling. *Communications of the ACM*, **35**, 99–113.

Houtsma, M. & Swami, A. (1995). Set-oriented data mining in relational databases. *Data & Knowledge Engineering*, **17**, 245–262.

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). Databases and expert systems: Opportunities and architectures for integration. In G. Gardarin & E. Gelenbe (Eds), *New applications of data bases* (pp. 185–201). London: Academic Press.

Kellogg, C. (1986). From data management to knowledge management. *Computer*, **19**, 75–84.

Kennedy, A. J., & Yen, D. C. (1990). Enhancing a DBMS through the use of an expert system. *Journal of Information System Management*, 55–60.

Kerschberg, L. (1989). The role of loose coupling in expert database system architectures. *Proceedings of the Fifth International Conference on Data Engineering*, Los Angeles, CA (pp. 255–256).

Kerschberg, L. (1990). Expert database systems: Knowledge/data management environments for intelligent information systems. *Information Systems*, **15**, 151–160.

Litwin, W. (1994). From database systems to multidatabase systems: Why and how. In A. R. Hurson, M. W. Bright, & S. Pakzad (Eds), *Multidatabase systems: An advanced solution for global information sharing* (pp. 13–40). Washington, DC: IEEE Computer Society Press.

Missikoff, M., & Wiederhold, G. (1986). Towards a unified approach for expert and database systems. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the First International Workshop)* (pp. 383–399). Reading, MA: Benjamin/Cummings.

Morgenstern, M., Borgida, A., Lassez, C., Maier, D., & Wiederhold, G. (1986). Constraint-based systems: Knowledge about data. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the Second International Conference)* (pp. 23–43). Reading, MA: Benjamin/Cummings.

Mylopoulos, J., & Brodie, M. L. (1985). AI and databases: Semantic vs conceptual theories of information. In G. Ariar, & Clifford, J. (Eds), *New directions for database systems*. Norwood, NJ: Ablex Publishing.

Parker, D. S. (1989). Integrating AI and DBMS through stream processing. *Proceedings of the Fifth International Conference on Data Engineering*, Los Angeles, CA (pp. 259–260).

Parker, D. S., Carey, M., Golshani, F., Jarke, M., Sciore, E., & Walker, A. (1986). Logic programming and databases. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the First International Workshop)* (pp. 35–48). Reading, MA: Benjamin/Cummings.

Risch, T., Reboh, R., Hart, P. & Duda, R. (1988). A functional approach to integrating database and expert systems. *Communications of the ACM*, **31**, 1424–1437.

Rundensteiner, E. A. (1990). The role of AI in databases versus the role of database theory in AI. In R. A. Meersman, Z. Shi, & C-H. Kung (Eds), *Artificial intelligence in databases and information systems (DS-3)* (pp. 233–252). Amsterdam: North–Holland.

Seljée, R. (1995). A new method for integrity constraint checking in deductive databases. *Data & Knowledge Engineering*, **15**, 63–102.

Sheng, O. R. L., & Wei, C.-P. (1992). Object-oriented modeling and design of coupled knowledge-base/database systems. *Proceedings of the Eighth International Conference on Data Engineering*, Tempe, AZ (pp. 98–105).

Sheth, A. P. (1989). Does loose AI–DBMS coupling stand a chance? *Proceedings of the Fifth International Conference on Data Engineering*, Los Angeles, CA (pp. 252–254).

Smith, J. M. (1986). Expert database systems: A database perspective. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the First International Workshop)* (pp. 3–15). Reading, MA: Benjamin/Cummings.

Stonebraker, M. (1992). The integration of rule systems and database systems. *IEEE Transactions on Knowledge and Data Engineering*, **4**, 415–423.

Stonebraker, M., & Hearst, M. (1989). Future trends in expert data base systems. In L. Kerschberg (Ed.), *Expert Database Systems (Proceedings from the Second International Conference)* (pp. 3–19). Reading, MA: Benjamin/Cummings.

Stonebraker, M. & Kemnitz, G. (1991). The POSTGRES next generation database management system. *Communications of the ACM*, **34**, 78–92.

Tsu, S. (1988). LDL—A technology for the realization of tightly coupled expert database systems. *IEEE Expert*, **3**, 41–51.

Turban, E. (1993). *Decision support and expert systems: Management support systems* (3rd ed.). New York: Macmillan.

Vassiliou, Y., Clifford, J., & Jarke, M. (1983). How does an expert systems get its data? *Proc. 9th 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.

Wiederhold, G. (1984). Knowledge and database management. *IEEE Software*, **1**, 63–73.

Wiederhold, G. (1986). Knowledge versus data. In M. L. Brodie, & J. Mylopoulos (Eds), *On knowledge management systems—Integrating artificial intelligence and database technologies* (pp. 77–86). Berlin: Springer.

Yang, H.-L. (1992). Incorporating semantic integrity constraints in a database schema. Ph.D. dissertation, University of British Columbia, Canada.

Zhao, S. (1994). Rule management in expert database systems. *Management Science*, **40**, 685–707.