

# A FRAME MODEL APPROACH FOR EXPERT AND DATABASE SYSTEM INTEGRATION<sup>1</sup>

Joseph Fong<sup>2</sup> and Shi-Ming Huang<sup>3</sup>

---

<sup>1</sup> This research has been funded by NSC 86-2213-E-036-005 and Tatung B84015

<sup>2</sup> Associate Professor, Department of Computer Science, 83 Tat Chee Avenue, Kowloon, Hong Kong, email: csjfong@cityu.edu.hk

<sup>3</sup> Computer Science & Engineering Department, Tatung Institute of Technology, 40 Chungshan N. Rd., 3<sup>rd</sup> Sec. Taipei, Taiwan, R.O.C. email: smhuang@cse.ttit.edu.tw

## **Abstract**

Expert systems(ES) and database systems(DBS) are major components of information systems and important assets to companies. The development of these systems represent users' knowledge in the application systems. As computer technologies evolve, and as users requirements change, there is a need to upgrade these system to meet the new application requirements. To preserve the knowledge of the existing information systems, a methodology for integrating ES and DBS into an expert database system(EDS) is proposed. The integrated EDS is a knowledge based system(KBS) which derives and stores knowledge in a frame model consisting of a class header, attributes, methods and constraints. It extracts the ES rules and DBS data for an application into coupling classes at run time only. The attributes of the coupling classes are matched with synonyms in a synonym table which resolves their naming and semantic conflicts with user assistance in knowledge acquisition. The resultant EDS is a KBS ready for application development.

Keywords: expert systems, database systems, expert database systems, knowledge based systems, frame model.

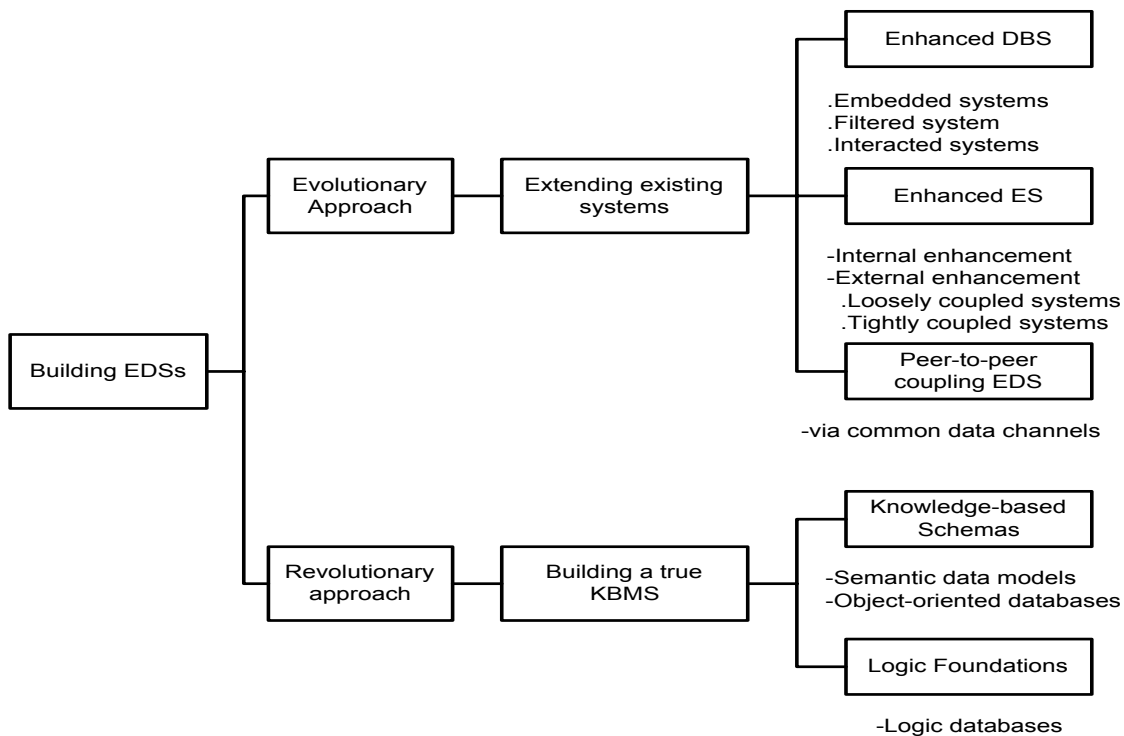
# 1 INTRODUCTION

Reusing or developing an integrated system for existing expert systems and database systems is a complex process. The system developer reuses both existing databases and expert systems to create an expert database system[1]. For example, the company links expert systems and databases, or the company has bought a new expert system and links it with their existing databases.

There are fundamental different opinions coming from the current ES and DB communities for EDS. The use of ES functions in DB products is to achieve 'deductive data', retrieve the semantics of data, and create intelligent interface, integrity constraints, etc. The use of DB functions in ES products is to represent factual knowledge in the original knowledge base. These differences mean that current EDSs have very different working environments.

Different approaches have been taken by various research projects and commercial products to achieve the requirements of an EDS. They can be classified into two different groups (see Figure 1):

- Based on existing systems: there are four different architectures in this area, i.e. enhancing existing database systems, enhancing existing expert systems, master-slaver coupling of ES-DB, and peer-to-peer coupling of ES-DB. Most current products can be categorized into one of these four architectures.
- A new knowledge-based management system: this architecture involves searching for a new model to represent knowledge. One example of this type of system is Generis [2].



**Figure 1 EDSs typology**

## The problems

Very often, the EDSs described above are heterogeneous systems. Schematic and operation heterogeneity are a crucial problem in building and using a heterogeneous system. This is due to the different systems operate independently and the data or knowledge may include structural and representational discrepancies (i.e. conflicts). Schematic heterogeneity concerns knowledge representation aspects. It can take form of:

- naming conflicts: when different systems use different names to represent the same concepts;
- domain conflicts: when different systems use different values to represent the same concepts;

- meta-data conflicts: when the same concepts are represented at the schema level in one system and at the instance level in another;
- structural conflicts: when different systems use different structure to represent the same concepts.

In most ESs, facts are realized according to the constraints imposed by the characteristics of the inference engine and by the properties of the problem at hand. Most of these systems mention nothing of the ad hoc ways of structuring any database of facts. That is why this type of problem becomes a major task in enhanced ESs. On the other hand, the relational model is not really compatible with logic, rules, frames, and semantic networks, which are typical of ES systems. Several performance problems arise from this mismatch, especially those requiring data to be exchanged by using redundant data descriptions which form the interface between the coupled systems.

An ES reasoning mechanism makes use of data through its variables instantly; therefore, it requires some data during each inference and in an atomic form (individual tuples of data values). However, a relational DBMS answers a query by returning results as sets of tuples. Accordingly, when the front-end breaks down a query into a sequence of queries on tuples, each of them incurs a heavy back-end performance overhead: we lose, therefore, the benefits of the set-oriented optimization that is characteristic of the back-end relational database.

The third criticism concerns the limited functionality and general information provided by the integrating system. Ideally the integrated system should support the full functionality of both systems plus some additional functionality arising from the integration. Unfortunately, most current systems either do not support all of the functions of both systems, or support only a very limited set of additional functions. Also the general resource information, (i.e. the data dictionary), is poor in current EDSs. Most systems do not support this resource information. This makes programming expert database systems extremely difficult.

The fourth criticism concerns the development life cycle of re-using the existing systems to create a new information system. Currently there are no formal methodologies to implement this type of system. How can the developer know the existing data is sufficient for the new system requirement? If it is not sufficient, what will be a remedial action? How can the existing system join the system analysis and design phase? How do we test this type of system during the development life cycle?

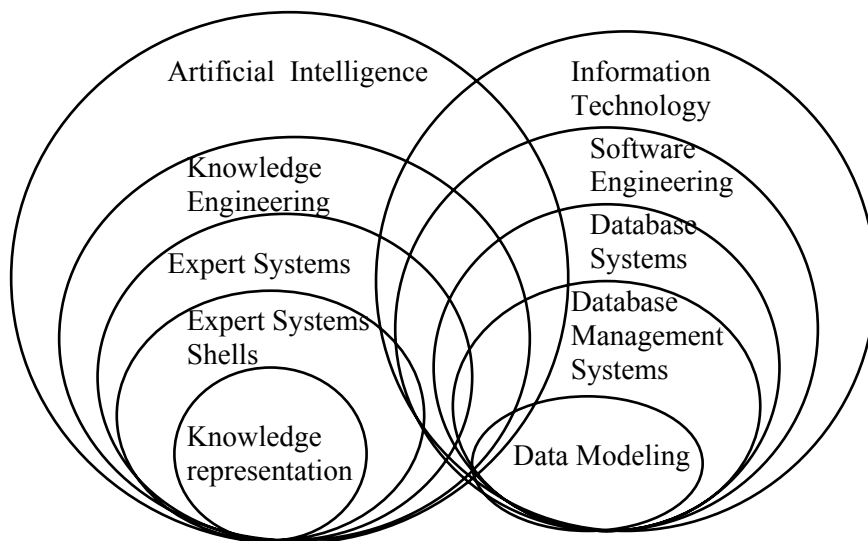
## **Our approach**

Our approach is a peer-to-peer coupling EDS(refer to Figure 1) and emphasizes in reengineering existing expert systems and database systems with a higher level synthesis model - frame model (refer to section 2). There are three scenarios that our approach can be applied into:

- 1.Reusing expert systems - The system developer reuses an existing expert system and builds new databases to create an integrated expert database system. This happens when:
  - the existing expert system has difficulty handling a growing volume of factual data;
  - a new database is required in the organization and this database can support the existing expert systems;
  - a new database system is required to work underneath an existing intelligent interface, such as a natural language interface.
- 2.Reusing databases - The system developer reuses existing databases and builds a new expert system to create an expert database system. This happens when:
  - there is a requirement to build intelligent components into existing databases, for example, integrity constraints, natural language interfaces or intelligent interfaces, deductive rules, intelligent retrieval, or query optimization.
  - a new expert system is required and the existing databases can support this system.
- 3.Reusing both database and expert systems - The system developer reuses both existing database and expert systems to create an expert database system. For example, the company links expert systems and databases, or the company has bought a new expert system and links it with their existing databases.

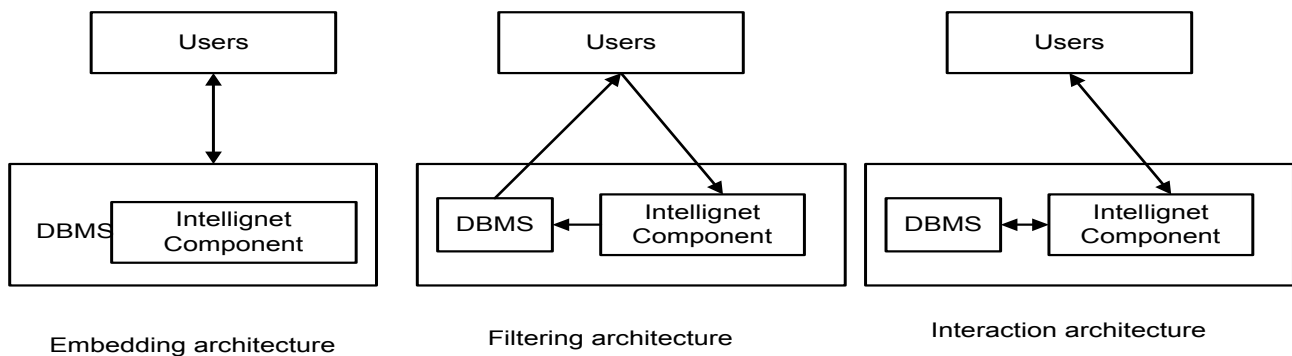
## Related Work

EDSs have arisen from the demand for higher-level interfaces and intelligent reasoning capabilities from traditional DBMS users, and the practicality involved in applying artificial intelligence (AI) research to large-scale 'real-world' problems [3]. They thus represent the confluence of concepts, tools and techniques from several diverse areas as shown in Figure 2. Each area has several interesting aspects; some related to database(DB) technology, some to artificial intelligence(AI) technology. To implement such a technology, first we must know the difference between AI and DB technology, especially knowledge-based systems and database systems technology. [4] described the basic difference between knowledge-based systems and database systems is that a knowledge-based system can perform inferences on knowledge, but a database system has a primary search function. From the analysis of the differences between database and knowledge bases, an EDS should have the characteristics of being a highly efficient management of a large shared information base and having the ability to make intelligent inferences from that information. In order to achieve the above, much work has been done to extend existing systems, to couple existing systems, and to design a new knowledge model for large knowledge-based systems.



**Figure 2 The diverse areas of Expert Database Systems**

The extended existing DBMS systems approach takes an existing database management system (DBMS) as a starting point and moves in an evolutionary fashion towards the goal of a knowledge base management system. The general idea behind linking an intelligent component to a DBMS is to improve either the efficiency, or the functionality of the DBMS, or both. Increased efficiency can result from techniques such as semantic query optimization. Increased functionality can be demonstrated by supporting such elements as AI-based programming language features, natural language interfaces, multiple user views, integrity constraints, and mechanisms for handling incomplete data within the DBMS environment[5]. There are three possible ways to build this type of EDS[6]: Embedding, Filtering and Interaction. For example, deductive routines or AI-based programming features are embedded as an integral part of the DBMS. User and application program queries can be filtered through an intelligent component before processed by the DBMS. The DBMS can dynamically interact with the intelligent component to improve the efficiency and functionality of a conventional DBMS. Figure 3 shows these examples.



**Figure 3 Extended Existing Systems**

The coupling existing system approach is based on the notion of two co-operating, but separate, 'front-end' and 'back end' subsystems. It allows an expert system and database to exist as independent systems that communicate via a common data channel. This permits the expert system and the DBMS to operate either as two entirely separate systems with their own set of users, or as two cooperating systems.

The new knowledge model approach suggests a true knowledge base management system from the marriage of existing technologies[7][8] by designing a new higher-order synthesis. The knowledge base management system is an evolutionary approach to the problem of building a system for managing knowledge. Most current new knowledge models are based on the semantic data model [9], production rules systems[10], frame-based systems[11] and knowledge representation.

Computer scientists have spent over 30 years investigating methodologies to represent knowledge. There are two popular foundations for this research. One is data modeling; the other is knowledge representation. Knowledge representation places its emphases on tackling the problem of expressiveness and structure, while the data modeling approach places its emphasis on an efficient physical data structure.

A general model for knowledge representation would be one which formed the basis of a system exhibiting human intelligence. Such a model is likely to require a wide variety of knowledge representation formalisms to represent different types of knowledge; such as current facts, past and future knowledge, meaning of words, certain and uncertain situations, negative situations, etc. This is a symptom of what [12] call a 'fundamental trade-off between expressiveness and tractability.' Projects which have been undertaken in the area of knowledge representation include: Cyc: A software integrates expert systems, natural language understanding, and machine learning within a frame-based knowledge representation language [13]. Sora: This system can manage a large number of production rules [14]. SRL: An integrated knowledge engineering environment with logic, rule base, and a frame-based object-oriented programming language [15]. KEE: A frame based knowledge base management system which allows production rules to be presented as a frame[16]. Semantic networks is an explicit taxonomic hierarchical structure for categorizing classes of real world objects. Production rules are sometimes called 'condition-action rules', 'situation-action rules', 'premise-conclusion rules', or 'if-then rules'. The above tools emphasize in connecting expert system rules with database only.

The conventional approach to EDS architecture is to enhance or couple existing database systems and expert system shells. The major problem of extending current existing systems to become an EDS is system performance. The major criticisms from a theoretical point of view are the 'fundamental mismatch' and the loss of information in the relational paradigm [17]. The 'fundamental mismatch' happens when coupling two subsystems or enhancing an expert system shell. The loss of information in the relational paradigm happens in the case of enhanced database systems. There are also some criticisms from a practical point of view, like limited functionality of integrated systems and the limited scope of the integrating information (e.g. data dictionary), etc.

## 2 FRAME MODEL

Frame-based systems is to collect all information related to one concept in one place. A frame is a

complex structure which can store and represent knowledge by using the 駢lot and Filler' formalisms [4]. An expert system frame model [18] is a good example of a knowledge based model which fulfills the requirements for constructing an integrated EDS. It is a higher-order synthesis which includes frame concepts, semantic data modeling concepts and object-oriented concepts to ensure no real distinction between "data" and "knowledge". The frame model follows an object-oriented paradigm. All conceptual entities are modeled as objects. The same attribute and behavior objects are classified into an object type, called a class. An object belongs to one, and only one, class. Both facts and rules are objects in the frame model. The frame model is implemented with a knowledge representation schema that includes object structure descriptions (i.e. classes), user-defined relationships between entities, and structure inheritance descriptions defined by taxonomies of structure that support data and behavior inheritance (i.e. abstract relationship). The frame model consists of four classes: Main, Attribute, Method and Constraint in a data dictionary.

**Table 1 The internal dynamic schema of frame model in data dictionary**

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Main Class</b></p> <pre>{ <b>Class_Name</b> /* a unique name in all system */   <b>Primary_Key</b> /* an attribute name or by default a class_name */   <b>Parents</b> /* a list of class names */   <b>Description</b> /* the description of the class */   <b>Class_Type</b> /* type of class, e.g. coupling and active */}</pre>                                                                                                                                                                                           |
| <p><b>Attribute Class</b></p> <pre>{ <b>Attribute_Name</b> /* a unique name in this class to represent the attribute identity */   <b>Class_Name</b> /* name of the main class that the attribute classes belong to */   <b>Method_Name</b> /* the value function of the attribute */   <b>Attribute_Type</b> /* the data type for the attribute */   <b>Default_Value</b> /* predefined value for the attribute */   <b>Cardinality</b> /* is the attribute single or multi-valued */   <b>IO</b> /* data flow direction */}</pre> |
| <p><b>Method class</b></p> <pre>{ <b>Method_Name</b> /* a unique name in this class */   <b>Class_Name</b> /* name of the main class that the method classes belong to */   <b>Parameters</b> /* a list of arguments for the method */   <b>Method_Type</b> /* the final result data type */   <b>Description</b> /* the description of the method */   <b>Condition</b> /* the rule conditions */   <b>Action</b> /* the rule actions or normal methods */ }</pre>                                                                 |
| <p><b>Constraint class</b></p> <pre>{ <b>Constraint_Name</b> /* a unique name for each constraint */   <b>Class_Name</b> /* name of the main class that the constraint classes belong to */   <b>Method_Name</b> /* constraint method name */   <b>Parameters</b> /* a list of arguments for the method */   <b>Ownership</b> /* the class name of the owner of the method */   <b>Event</b> /* triggered event */   <b>Sequence</b> /* method action time */   <b>Timing</b> /* the method action timer */ }</pre>                 |

The components of the frame model can be described as follows:

- Main Classes -

The main class includes active classes and coupling classes. Active classes represent rule entities and is event driven. Coupling classes represent the temporal entities, and is created at run time. The two classes use the same structure. Combining these two types of object within the inheritance hierarchy structure enables the frame model to represent and combine heterogeneous knowledge.

- Attributes class -

Most attributes have a single value for a particular object; such attributes are called single-valued. In some cases an attribute can have a set of values for the same object. A coupling class attribute has a value function to display its content. The value function will communicate with the existing system. The IO type represents the data flow direction for the communication between the integrated system and the existing system. There are two types of data flow direction:

1. Input Part Direction: The data flow direction is from the existing system to the integrated system.

2. Output Part Direction: The data flow direction is from the integrated system to the existing system.

- Methods class -

Rules extend the semantics of the data. The specification of rules is an important part of semantic data modeling, since many of the facts in the real world are derived rather than consisting of pure data [19]. It is increasingly important to integrate rules into data models in new information systems. The methods of the frame model represent the behavior, the active rules, and the deductive rules of a particular object. The method body takes a production rule structure in the frame model.

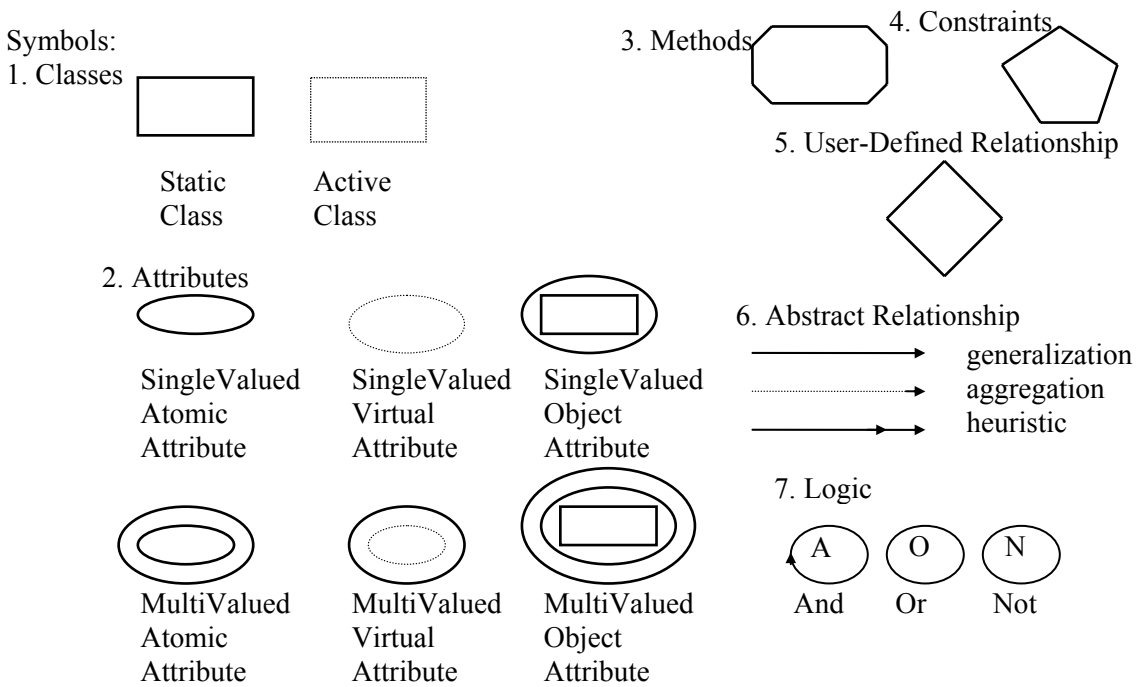
- Constraints -

There are many properties of data that cannot be captured in the form of structures. These properties essentially serve as additional restrictions on the values of the data and/or how the data may be related (structured). Constraints can be used to prevent a possibly expensive database search operation or to answer otherwise unsolvable queries [20]. The constraint technology used in current database systems requires different levels of integrity constraint. There are two types of constraints used in database technology:

1. Static constraints that limits the allowable database states so as to reflect accurately the real world situation.
2. Dynamic constraints that restrict the possible database state transitions.

To resolve the synonyms and homonyms conflicts among data models, their locations are defined in synonyms and homonyms tables, and active classes are provided to define their constraints. To ensure the interoperability among data models, an efficient way is to translate them into a frame model. The translated frame model can be integrated into a global frame model by identifying their related semantics.

The frame model conceptual schema can be represented in a diagram in Figure 4. The emphasis is on representation at a schema level rather than at an instance level. This is useful because a knowledge schema rarely changes; whereas the extension may change frequently. The schema is usually easier to display than the extension of a knowledge base, because it is compact. The frame model conceptual schema integrates the conceptual requirements of individual users into a single “community” view. The frame model logical schema describes the version of the frame model in conceptual schema that can be presented to the DBMS.



**Figure 4 Frame model conceptual schema diagram notations**

Figure 5 displays the family knowledge base schema in a frame model conceptual schema diagram. The knowledge base shows that a person is described by Name, Sex, Date\_of\_Birth, Age,

Generation, Address, Father, and Mother. A male is a person and his sex is always male. A person is male. He is a child of a person and that person is the parent of the child. This is an inverse relationship. Age and generation are derived by the date of birth of the person. The father and mother of a person are themselves persons. This is a recursive situation. An address is an associated attribute which is described by Street, City, and Postcode. Sex, and Date\_of\_Birth are shown in ovals. Virtual attributes such as Generation and Age are shown in dotted ovals. Object attributes such as Father, Mother, and Address are shown in ovals with an internal rectangular box. Each attribute is attached to its class type or relationship type by a straight line. Methods such as Male are shown in Octagons and each method is attached to its class type or relationship type by a straight line.

Generalization (isa) relationships such as relationship between Person and Male are shown by an arrow line. Disjoint generalization can be shown by Disjoint constraint among Mother class and Father class. A person can either be a natural father or mother, but cannot be both. Aggregation relationships such as relationship between the Address attribute of the Person class and the Address class are shown by a dotted arrow line. Heuristic relationships such as the relationship between Date\_of\_Birth and Age, or that between Age and Generation, are shown as a double arrow line. Constraints such as Always are shown as a Pentagon and each constraint is attached to its method by a heuristic relationship in a double arrow line. For example, the Always constraint is implemented in the Sex attribute via the Male method. Logical operators such as AND, OR, and NOT are shown in a circle with the first letter of the operators. They are implied by heuristic relationships. For example, the Son class is deduced from the Male and Child entity via an AND operation. Similarly, the Parents class is deduced from the Father and Mother entity via an OR operation.

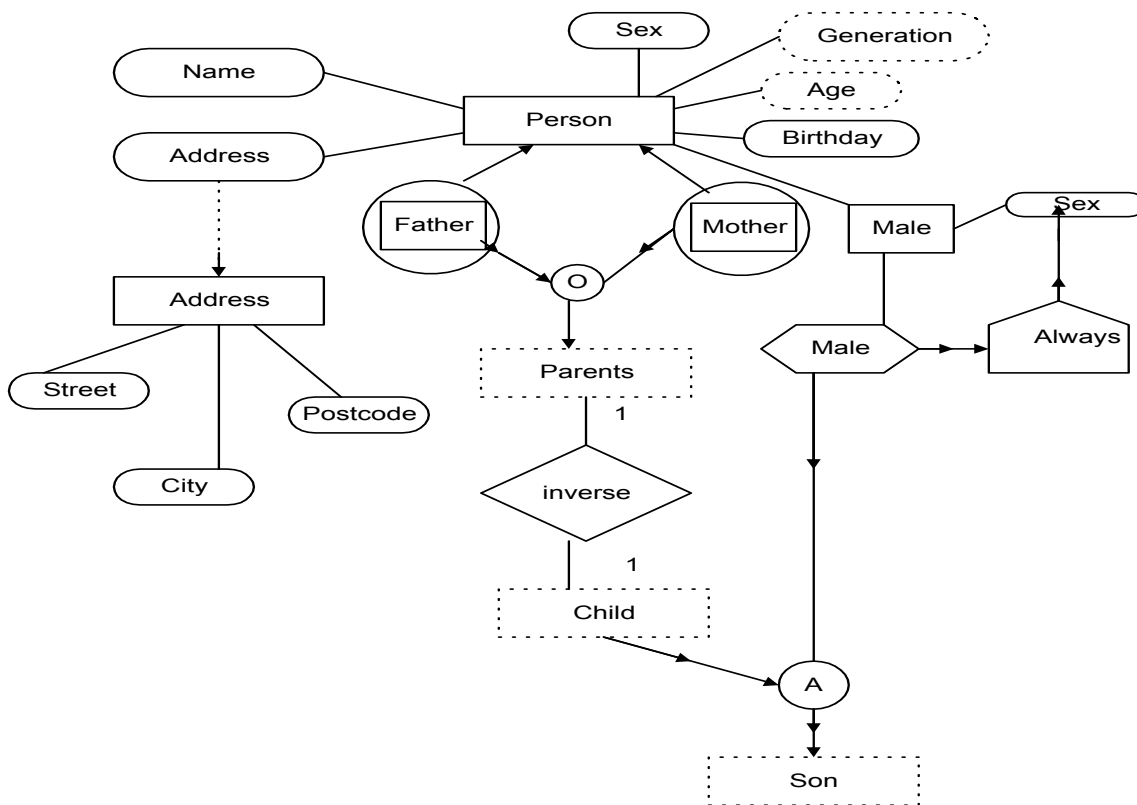
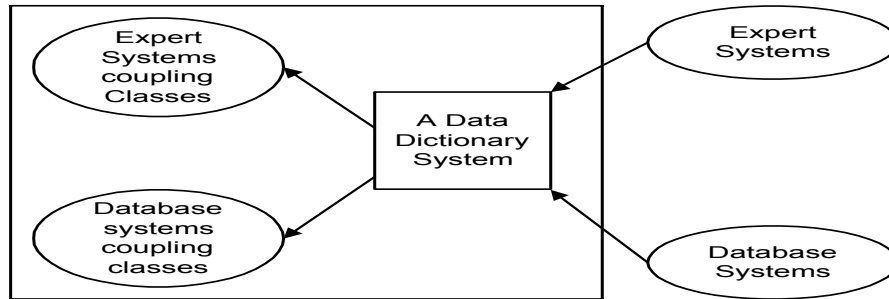


Figure 5 Frame Model conceptual schema Diagram for the Family Knowledge Base

### Reengineering in the frame model

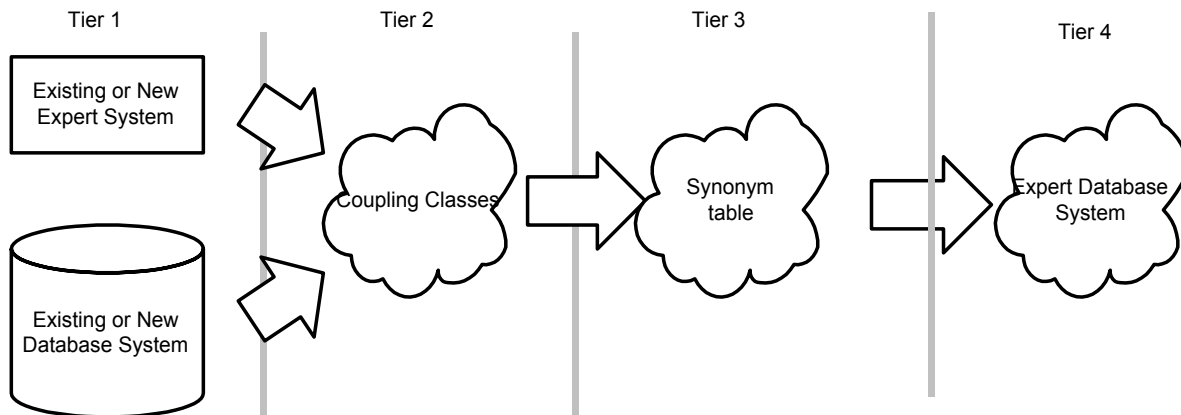
Reengineering is an important feature in the frame model. The system enables reengineering through the coupling classes. To implement the frame model, we must include as inputs, the database system and the expert system, and as output the frame model classes. To integrate these two coupling classes, the developer must update the synonym table into a data dictionary system as a repository for an updated synonym table as shown in Figure 6. The synonym table is to integrate and resolve

naming conflicts among the coupling classes.



**Figure 6 The overview of the EDS architecture**

We focus in integrating an existing ES and DB system into an EDS and choose frame model as a knowledge representation due to its dynamic data structure, that is, static data and its operations for each object. To provide a solution for the integration of DBs and ESs, a knowledge based model with a four tier framework is depicted in Figure 7. In this figure the existing systems form the lower tier. The required data from these systems is extracted using coupling EDS. The coupling EDS extract, and possibly transform the data of the lower tier into knowledge usable by the integrated system. The upper tier combines and enhances the knowledge of the existing system with additional knowledge to create an integrated expert database system.



**Figure 7 The Four Tier Integrate Expert Database System Model**

**Tier 1: Existing Systems**

The existing systems contain data to be reused in the integrated EDS. Only the data required for the operation of the integrated system is extracted. This data is brought into a consistent state through the coupling classes of tier 2.

**Tier 2: Coupling Classes**

Coupling classes describe the information in existing systems. A coupling class provides the interface between the extension layer and the existing systems. The uniformity of this interface layer insulates the upper layers from changes in the lower layers and can be used to bring information together so that data representing the same entities or attributes are consistent. An attribute in a coupling class is derived from the values of the entities stored in the underlying systems. The derivation is a simple one-to-one mapping. The coupling classes provide information from existing knowledge repositories, additional information can also be stored by the integrated system.

**Tier 3: Synonym table**

The third layer combines the components of the coupling classes with additional classes to create a synonym table. To form an integrated system, name conflict and semantic conflict problems must be solved. Since the system has a unified structure, i.e. a higher level synthesis model, the name conflict problem can be easily resolved by using the synonym index. The synonym index creates a relationship between two different attributes with the same values.

**Tier 4: Expert Database System**

After the integrated EDS has been developed, the system developer can use it as a knowledge base to develop its own application. The application system defines the components necessary to answer and give explanations for all problems that it is to solve.

### 3 METHODOLOGY FOR EXPERT AND DATABASE SYSTEMS INTEGRATION

We can apply the frame model in integrating existing database systems and expert systems. The system developer is required to build a communication channel between these two systems. The usual method to build this integrated system requires change to some parts of the existing systems. The data can be passed into the existing system by the system I/O stream. The stepwise procedure is shown in Figure 8.

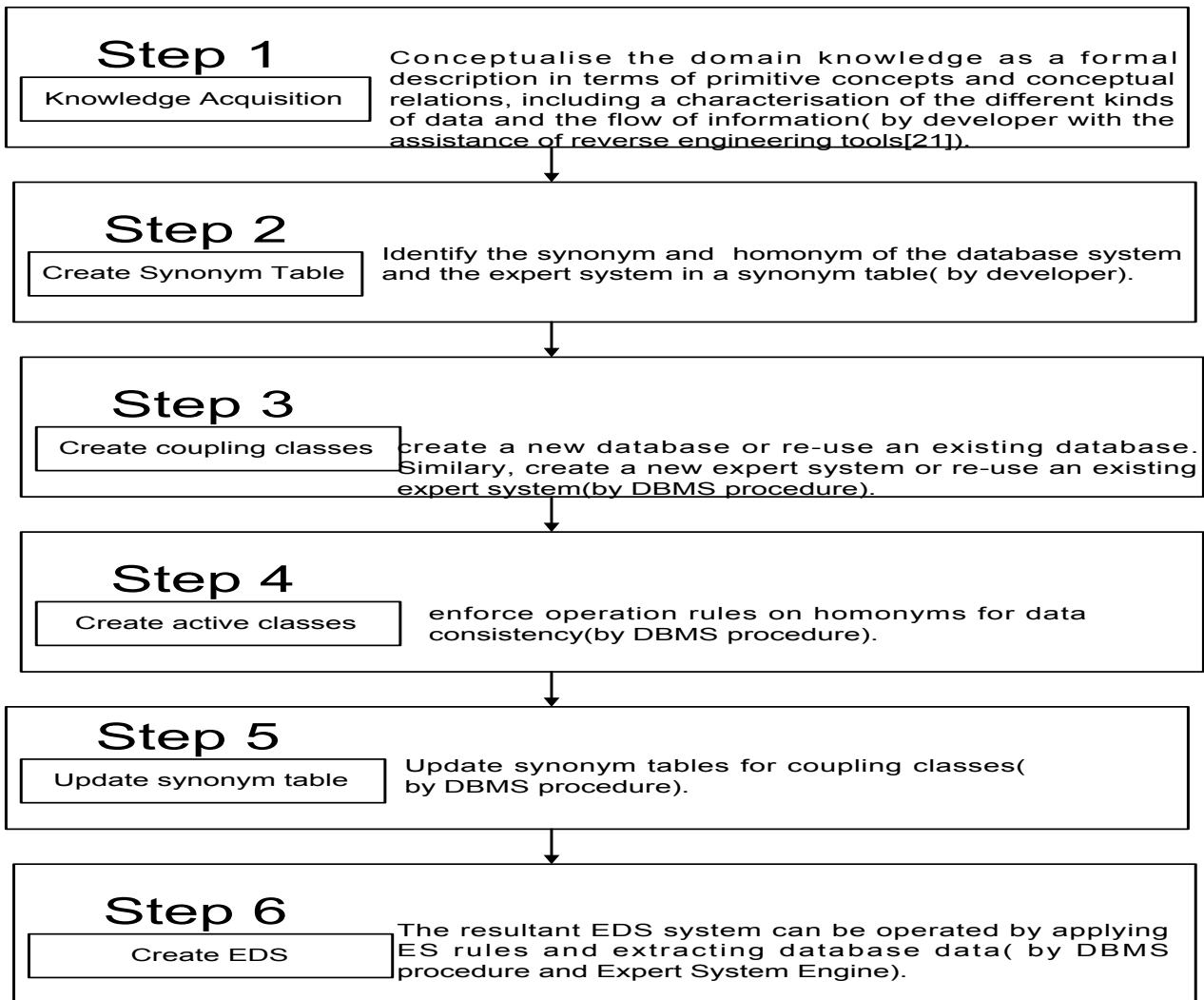


Figure 8 The steps in EDS methodology

#### Step 1 Knowledge Acquisition

In this step, the attributes of existing ES and DBS must be analyzed. The EDS developer must define the characteristic of each attribute, and provide a table structure used to represent the result of the knowledge acquisition from an ES. The schema of the table includes four attributes: name, style, type and memo. The name attribute represents the object name in the existing system. The object exists in three different kinds of styles, i.e. atom, rule, and variable, in an ES. Atom means that the object is a fact. Rule means that the object is an inference rule. Variable means that the object value will be generated or supported by other event. The type attribute represents the content type of the object. Memo is to enable the developer to write down comments for the object. This will assist the developer to understand the meaning of the object during the development cycle. Since the schema will be stored in a data dictionary as a meta database (a database for another database), users can use search command to browse memos efficiently.

The EDS developer must know which existing databases will relate to the new system. The developer must also understand the existing database schema. There are various ways in which the developer can discover the existing database schema. One way is to go through the database documents to find out its schema. Another way is to retrieve the database schema from the data dictionary system of the existing database system. A third way is to use the database conversion or migration tools to reverse the database schema into a developer understandable format. We apply data dictionary approach for practical reason as a repository for both DBS and ES.

#### Step 2 Create Synonym table

This step is to analyze the synonym relationship between these attributes of the existing two systems. We need users input to identify the synonyms and homonyms of the attributes in DBS and ES. A data dictionary is a powerful documentation tool for users record the semantics of each attribute for both ES and DBS. We map each attribute in the ES to an attribute in the DBS. If they provide the same meanings, we identify them as 'same' in synonym degree. If they use different semantic to represent the same object, we identify them as 'derivable' (i.e. value of an attribute can be derived from another attribute) in synonym degree. We can also rename homonyms to alternative names to avoid confusions in the developed EDS.

#### Step 3 Create coupling classes -

The database system and the expert system will be coupled within the frame model as two separate coupling classes. The form of generalization between the coupling classes is the same as active generalization. Different coupling classes can use the generalization relationship to combine together to form a new coupling object. This hierarchical structure can represent distributed knowledge (or distributed DBS) semantics.

#### Step 4 Create active classes

An application of active class is to resolve naming conflicts of derivable attributes among ES and DBS by enforcing the relationship between them by operations. The meaning of these attributes are based on application requirement. Very often, we can derive them from users assistance or ES rules.

#### Step 5 Update synonym table for active classes

Update the synonym table by identifying the derivable attributes among the active classes. EDS can extract information from source ES and DBS into active classes, integrate them by the synonym tables, and transform information into knowledge to meet the application requirements. The synonym table is stored in the data dictionary as an event driven active stored procedure, which can be triggered on the changed of value of an attribute in the data dictionary. The system developer will then integrate these two sub-systems into a system within the frame model.

#### Step 6 Create EDS

The source ES and the source DBS can be integrated into an EDS, which transform the input information into knowledge by developing a knowledge based system, i.e. applying ES rules and extract data from a DBS. To implement the frame model, we must include the database system and the expert system as input and the frame model classes as output. The information resource dictionary system is an information repository for the frame model.

In summary, the methodology can be implemented semi-automatically by the developers with the users assistance. It applies the stored procedures to retrieve data from the database. The developer can set up a command file to invoke both the database procedures and the expert systems engines.

### **4 A CASE STUDY – AN INTELLIGENT HUMAN RESOURCE SYSTEM**

The UK government agency employs approximately 4,000 staff, and is sub-divided into a number of Directorates. Each Directorate has a resource manager who is responsible for a number of projects. The duty of the resource manager is to fit suitably qualified people to specific jobs within each of

the projects for their Directorate. The main task of this project was to match staff with suitable placements in an EDS. A prototype system of SEDSDT (refer to section 5) has been implemented for the project. Given is Table 2 as a subset of the knowledge base held in a Human Resource Management(HRM) System ES and Table 3 as a part of the personnel database.

| Table 2 The sample rules for the HRMSystem ES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Rule Find-Employee:<br/> IF First-Priority-Group<br/>     AND Skill-Sufficient<br/>     AND Location = Preferred-Working-Area<br/> THEN Display Person-id AND Name</p> <p>Rule First-Priority-Group:<br/> IF Project-Directorate = Person-Directorate<br/>     AND Staff-Type = "Internal"<br/>     AND Age &lt; Job-Required-Age<br/>     AND Availability = "Yes"<br/>     AND Average-Grade = "High"<br/> THEN True</p> <p>Rule Skill-Sufficient:<br/> IF Job-Required-Skill-1 = Person-Skill-1<br/>     AND Job-Required-Skill-2 = Person-Skill-2<br/> THEN True</p> |

| Table 3 Personnel Database Sub-Schema |           |       |
|---------------------------------------|-----------|-------|
| Field Name                            | Type      | Width |
| ID                                    | Character | 8     |
| Name                                  | Character | 20    |
| Age                                   | Number    | 2     |
| Staff-Type                            | Character | 15    |
| Directorate                           | Character | 20    |
| Current-Status                        | Character | 15    |
| Average-Mark                          | Number    | 1     |
| Skill-1                               | Character | 20    |
| Skill-2                               | Character | 20    |

In this case study, the system will integrate the HRM System ES and the Personnel database into an EDS. The frame model forms a communication bridge between them.

## EDS Development

### Step 1 Knowledge Acquisition

Table 4 shows the result of the knowledge acquisition for the HRM System ES.

| Table 4 The Result of the Knowledge Acquisition for the HRM System ES |          |           |                            |
|-----------------------------------------------------------------------|----------|-----------|----------------------------|
| Name                                                                  | Style    | Type      | Memo                       |
| Find-Employee                                                         | Rule     | Boolean   |                            |
| First-Priority-Group                                                  | Rule     | Boolean   |                            |
| Skill-Sufficient                                                      | Rule     | Boolean   |                            |
| Display                                                               | Rule     | Boolean   |                            |
| Person-id                                                             | Variable | Character |                            |
| Name                                                                  | Variable | Character |                            |
| Location                                                              | Variable | Character |                            |
| Preferred-Working-Area                                                | Variable | Character |                            |
| Project-Directorate                                                   | Variable | Character |                            |
| Person-Directorate                                                    | Variable | Character |                            |
| Staff-Type                                                            | Variable | Character |                            |
| Age                                                                   | Variable | Number    |                            |
| Job-Required-Age                                                      | Variable | Number    |                            |
| Availability                                                          | Variable | Character | “yes” or “no”              |
| Average-Grade                                                         | Variable | Character | “high”, “middle”, or “low” |
| Job-Required-Skill-1                                                  | Variable | Character |                            |
| Person-Skill-1                                                        | Variable | Character |                            |
| Job-Required-Skill-2                                                  | Variable | Character |                            |
| Person-Skill-2                                                        | Variable | Character |                            |
| ”Internal”                                                            | Atom     | Character |                            |
| “High”                                                                | Atom     | Character |                            |
| ”Yes”                                                                 | Atom     | Character |                            |
| True                                                                  | Atom     | Boolean   |                            |

### Step 2 Create synonym table

The existing database is stored in a relational DBMS. We used its data dictionary system to retrieve the database schema. Table 5 shows the synonym of the attributes between the ES and the DBS for this case.

| Attribute              | System<br>-Name | System<br>-Name | Synonym-<br>Degree | Attribute      | System<br>-Type | System<br>-Name |
|------------------------|-----------------|-----------------|--------------------|----------------|-----------------|-----------------|
| Person-id              | ES              | HRM             | Same               | ID             | DBS             | Personnel       |
| Name                   | ES              | HRM             | Same               | Name           | DBS             | Personnel       |
| Person-<br>Directorate | ES              | HRM             | Same               | Directorate    | DBS             | Personnel       |
| Staff-Type             | ES              | HRM             | Same               | Staff-Type     | DBS             | Personnel       |
| Age                    | ES              | HRM             | Same               | Age            | DBS             | Personnel       |
| Availability           | ES              | HRM             | Derivable          | Current-Status | DBS             | Personnel       |
| Average-Grade          | ES              | HRM             | Derivable          | Average-Mark   | DBS             | Personnel       |
| Person-Skill-1         | ES              | HRM             | Same               | Skill-1        | DBS             | Personnel       |
| Person-Skill-2         | ES              | HRM             | Same               | Skill-2        | DBS             | Personnel       |

### Step 3 Create Coupling Classes

The frame model will create two coupling classes for the EDS. Figure 9 shows the HRM System ES coupling class and Figure 10 shows the Personnel DBS coupling class. The attributes of ES coupling class will come from the attributes of HRM. The 'variable' style of HRM attributes will become output part attributes of the ES coupling class. The system developer must define an input part attribute which will store the result of the HRM System ES. The system developer defines an attribute, called find-employee. The attached method of this attribute will execute the external ES. The attribute of DBS coupling class is a mirror of personnel database schema. Each attribute within the coupling class will contain a method.

#### HRM Main Class

| Class Name | Parents | Description | Class Type |
|------------|---------|-------------|------------|
| HRM        | 0       | Person      | Coupling   |

#### HRM attribute Class

| Attribute_name         | Class Name | Method Name            | Attribute type | Default | Cardinality | IO     |
|------------------------|------------|------------------------|----------------|---------|-------------|--------|
| Person-id              | HRM        | Person-id              | Text           |         | single      | output |
| Name                   | HRM        | Name                   | Text           |         | single      | output |
| Location               | HRM        | Location               | Text           |         | single      | output |
| Preferred-Working-Area | HRM        | Preferred-working-area | Text           |         | single      | output |
| Project-Directorate    | HRM        | Project-Directorate    | Text           |         | single      | output |
| Person-Directorate     | HRM        | Person-Directorate     | Text           |         | single      | output |
| Staff-Type             | HRM        | Staff-Type             | Text           |         | single      | output |
| Age                    | HRM        | Age                    | Number         |         | single      | output |
| Job-Required-Age       | HRM        | Job-Required-Age       | Number         |         | single      | output |
| Availability           | HRM        | Availability           | Text           |         | single      | output |
| Average-Grade          | HRM        | Average-Grade          | Text           |         | single      | output |
| Job-Required-Skill-1   | HRM        | Job-Required-Skill-1   | Text           |         | single      | output |
| Person-Skill-1         | HRM        | Person-Skill-1         | Text           |         | single      | output |
| Job-Required-Skill-2   | HRM        | Job-Required-Skill-2   | Text           |         | single      | output |
| Person-Skill-2         | HRM        | Person-Skill-2         | Text           |         | single      | output |
| Find-Employee          | HRM        | Find-Employee          | Text           |         | single      | input  |

Figure 9 HRM System ES coupling class

Personnel Main Class

| Class_Name | Parents | Description      | Class_Type |
|------------|---------|------------------|------------|
| Personnel  | 0       | Personnel system | Coupling   |

Personnel attribute Class

| Attribute_name | Class_Name | Method_Name    | Attribute_type | Default | Cardinality | IO    |
|----------------|------------|----------------|----------------|---------|-------------|-------|
| ID             | Personnel  | ID             | Text           |         | single      | input |
| Name           | Personnel  | Name           | Text           |         | single      | input |
| Age            | Personnel  | Age            | Number         |         | single      | input |
| Staff-Type     | Personnel  | Staff-Type     | Text           |         | single      | input |
| Directorate    | Personnel  | Directorate    | Text           |         | single      | input |
| Current-Status | Personnel  | Current-status | Text           |         | single      | input |
| Average-Mark   | Personnel  | Average-Mark   | Text           |         | single      | input |
| Skill-1        | Personnel  | Skill-1        | Text           |         | single      | input |
| Skill-2        | Personnel  | Skill-2        | Text           |         | single      | input |

Figure 10 Personnel DBS coupling class

The standard frame for the method will depend on the attribute of an input or output parameter in an I/O stored procedure as shown in the following standard algorithm of these two type methods.

```

Standard-Output-Parameter-Attribute( ); <result-data-type>;
{
Request the data from the system (i.e. Request)
Send the data to the external existing system (i.e. Write)
}

Standard-Input-Parameter-Attribute( ); <result-data-type>;
{
Receive the data from the existing external system (i.e. Receive)
Save the data to the system (i.e. Save)
}
    
```

Step 4 Create active classes

The resultant synonym table (Table 5) is to integrate the coupling classes. The solution is to create active classes for the derivable rules described in Step 1. The developer must solve the semantic conflict problem between these attributes. For example, Availability of HRM can be derived from the current-status in the Personnel database.

```

IF Current-Status = "Vacancy"
THEN Availability = "yes"
ELSE Availability ="no"
    
```

The corresponding active classes are:

Availability Main Class

| Class_Name   | Parents | Description             | Class_Type |
|--------------|---------|-------------------------|------------|
| Availability | 0       | Person 德 vacancy status | Active     |

Availability Attribute Class

| Attribute_name | Class_Name   | Method_Name   | Attribute_type | Default | Cardinality | IO |
|----------------|--------------|---------------|----------------|---------|-------------|----|
| Availability   | Availability | Availability  | Text           |         | single      |    |
| Available      | Availability | Available     | Text           |         | single      |    |
| Non_available  | Availability | Non_available | Text           |         | single      |    |

Availability Method Class

| Method_name   | Class_name   | Parameter                | Method_type | Condition                     | Action            |
|---------------|--------------|--------------------------|-------------|-------------------------------|-------------------|
| Availability  | Availability |                          | Text        | If Available or Non_available |                   |
| Available     | Availability | current_status@personnel | Text        | If current_status = "vacancy" | Available = "yes" |
| Non-available | Availability | current_status@personnel | Text        | If current_status ≠ 'vacancy' | Available = "no"  |

### Step 5 Update synonym table

After this, the developer can insert the synonym data into the data dictionary System. Table 6 shows the synonym part information of the data dictionary for the Information HRM System.

| Attribute          | System-Type | System-Name | Synonym-Degree | Attribute    | System-Type | System-Name  |
|--------------------|-------------|-------------|----------------|--------------|-------------|--------------|
| Person-id          | ES          | HRM         | Same           | ID           | DB          | Personnel    |
| Name               | ES          | HRM         | Same           | Name         | DB          | Personnel    |
| Person-Directorate | ES          | HRM         | Same           | Directorate  | DB          | Personnel    |
| Staff-Type         | ES          | HRM         | Same           | Staff-Type   | DB          | Personnel    |
| Age                | ES          | HRM         | Same           | Age          | DB          | Personnel    |
| Availability       | ES          | HRM         | Same           | Availability | Active      | Availability |
| Average-Grade      | ES          | HRM         | Same           | Average-Mark | Active      | Average_mark |
| Person-Skill-1     | ES          | HRM         | Same           | Skill-1      | DB          | Personnel    |
| Person-Skill-2     | ES          | HRM         | Same           | Skill-2      | DB          | Personnel    |

The processing flow for the coupling class method is:

```

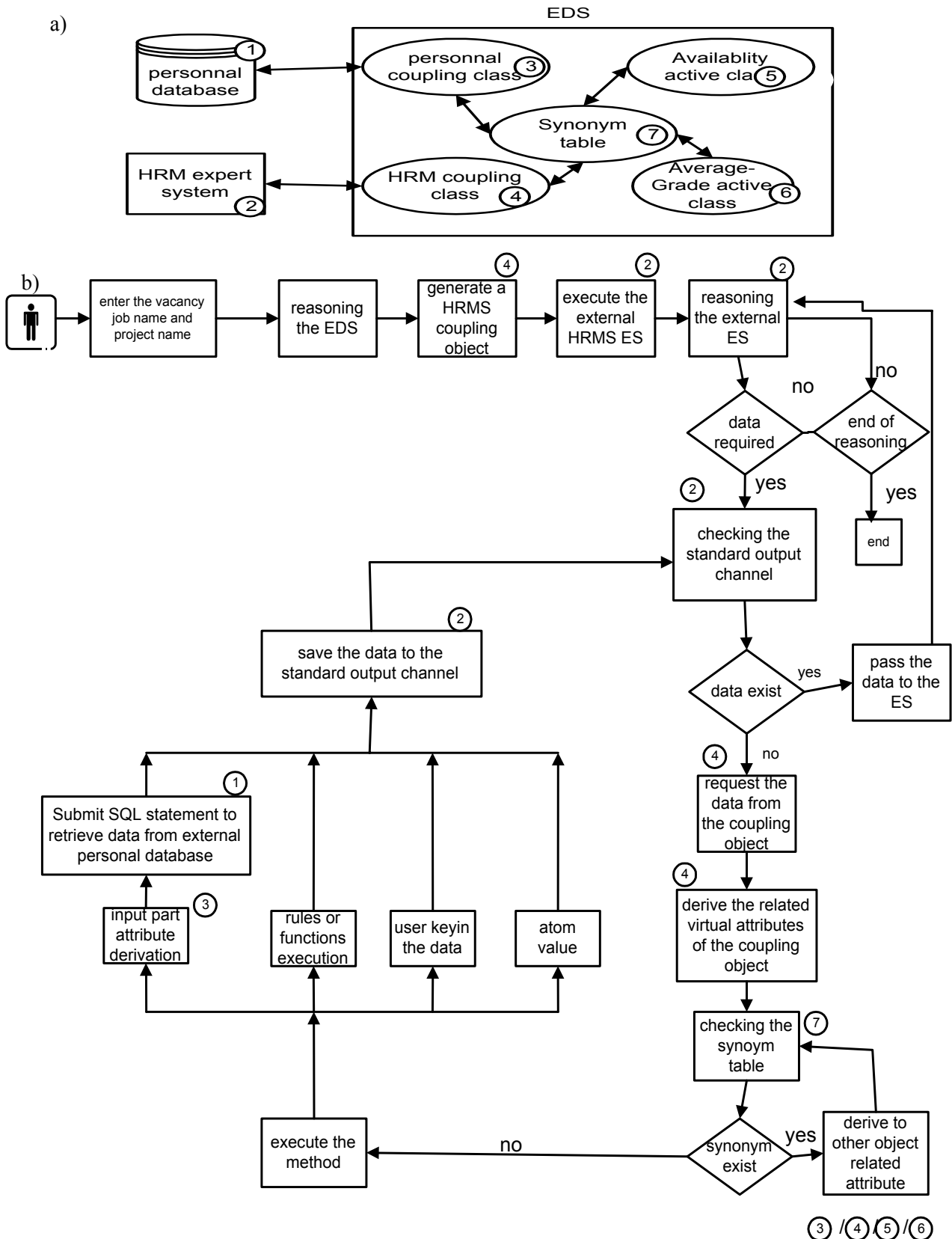
IF the attribute is an output parameter of the I/O procedure
THEN IF the attribute is identified as "same" or "derivable"
    THEN send message to the external to retrieve the data
    ELSE ask users to input the data
ELSE execute the coupling module functions

```

Figure 11 shows the data flow diagrams of the developed EDS. which acts as a knowledge based system for new application. Messages will be passed to the HRM coupling object to execute the virtual attributes of Person-Directorate and Project-Directorate. The HRM.Person-Directorate has a synonym Personnel Directorate. The system will generate a Personnel DB coupling object and pass a message to the object to derive the data of the Directorate. The Directorate is an input parameter attribute. The method will submit a SQL statement to retrieve the data from the external existing database. The data will pass back to the HRM System ES. The reasoning continues.

### Step 6 Create EDS

After integrating personnel DBS and HRM System ES, users can ask the Information HRM System EDS to give an advice for a particular vacancy job. In this case, the EDS will ask users to key-in the vacancy job information. Figure 11 shows the data flow diagram and the process flow chart for the personnel information system using the developed EDS.



**Figure 11 a) Integrated environment of the EDS b)The Process flow of the EDS**

Here, the system will generate a query to ask the end-users to enter the data (see Figure 12). There are seven different modules within the Information HRM System EDS. Module 1 to 7 represent personnel database, HRM System expert system, personnel coupling class, HRM coupling class, availability active class, and synonym table. Figure 11 identifies how these modules integrate with the process flow. Figure 12, Figure 13, and Figure 14 show a prototype of the Information HRM System EDS.

| <b>Human Resource Management Expert Database System<br/>(Information HRM EDS)</b> |                 |
|-----------------------------------------------------------------------------------|-----------------|
| Please Enter Vacancy Job Name: [Chief Programmer]                                 |                 |
| Please Enter Project-Name: [VLDB]                                                 |                 |
| Reasoning .....                                                                   |                 |
| Please Enter Project-Directorate: [ Directorate 1]                                |                 |
| Reasoning .....                                                                   |                 |
| Please Enter Job-Required-Age: [ 35 ]                                             |                 |
| Reasoning .....                                                                   |                 |
| Please Enter Job-Required-Skill-1: [Database Design ]                             |                 |
| Reasoning .....                                                                   |                 |
| Please Enter Job-Required-Skill-2: [Telecommunication ]                           |                 |
| Reasoning .....                                                                   |                 |
| Please Enter Job Location: [Newcastle]                                            |                 |
| Reasoning .....                                                                   |                 |
| Continue .....                                                                    | <b>Screen 1</b> |

**Figure 12 Sample Information HRM EDS Interactive Session**

| <b>Human Resource Management Expert Database System<br/>(Information HRM EDS)</b> |                 |
|-----------------------------------------------------------------------------------|-----------------|
| Please Enter 0001 Russell Parsons Preferred-Working-Area<br>[Newcastle]           |                 |
| Reasoning .....                                                                   |                 |
| Please Enter 01001 Peter Smith Preferred-Working-Area<br>[Edinburgh]              |                 |
| Reasoning .....                                                                   |                 |
| Please Enter 0125 Jack Huang Preferred-Working-Area:<br>[Newcastle]               |                 |
| Reasoning .....                                                                   |                 |
| Conclusion .....                                                                  | <b>Screen 2</b> |

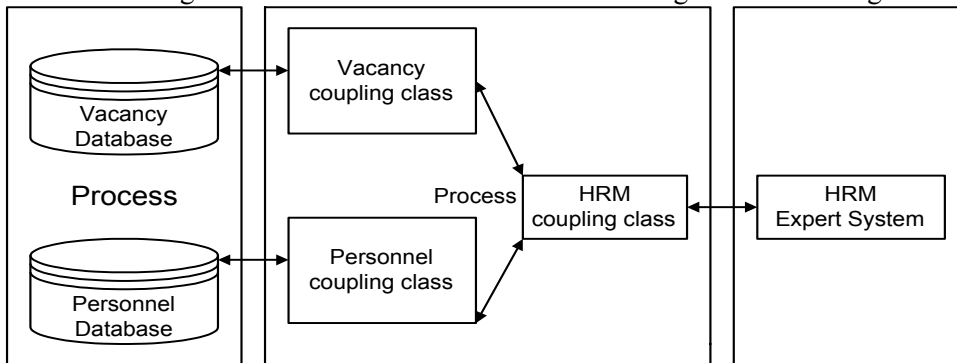
**Figure 13 Sample IHRM EDS Interactive Session**

| <b>Human Resource Management Expert Database System<br/>(Information HRM EDS)</b>     |                 |
|---------------------------------------------------------------------------------------|-----------------|
| My Advice for the Vacancy Job (Chief Programmer in Project VLSI of Directorate 1) is: |                 |
| Person ID: 0001                                                                       |                 |
| Name: Russell Parsons                                                                 |                 |
| Person ID: 0125                                                                       |                 |
| Name: Jack Huang                                                                      |                 |
| Total 2 persons are qualified for this job.                                           |                 |
| Press Any Key to Continue                                                             |                 |
|                                                                                       | <b>Screen 3</b> |

**Figure 14 Sample IHRM EDS Conclusions**

**EDS Enhancement**

To enable the system do the automatic and batch vacancy job consultancy, a vacancy database can be developed and integrated into the Information HRM System EDS. The Vacancy database stores all vacancy job information. Figure 15 shows the ideal integration system of Information HRM System EDS. The new vacancy database can be designed into the frame model or other existing database management system.



**Figure 15 HRM Expert Database System**

Step 1. Knowledge acquisition for new application system requirement analysis

Since we know the vacancy job information which the Information HRM System EDS need is job location, the directorate of the job, age of the job required, skill of the job required, we can design the new database schema in Table 7.

| Table 7 Vacancy Database Schema |           |       |
|---------------------------------|-----------|-------|
| Field Name                      | Type      | Width |
| Job-code                        | Character | 4     |
| Project-Name                    | Character | 20    |
| Location                        | Character | 20    |
| Required-Age                    | Number    | 2     |
| Directorate                     | Character | 20    |
| Required-Skill-1                | Character | 20    |
| Required-Skill-2                | Character | 20    |

Step 2 Create Synonym table

The developer also needs to analysis the synonym information for this new database and the existing system. Table 8 shows the synonym information which regards to the new environment.

| Table 8 The Synonym Table for the EDS |                 |                 |                    |                      |                 |                 |
|---------------------------------------|-----------------|-----------------|--------------------|----------------------|-----------------|-----------------|
| Attribute                             | System<br>-Type | System<br>-Name | Synonym<br>-Degree | Attribute            | System<br>-Type | System<br>-Name |
| Person-id                             | ES              | HRM             | Same               | ID                   | DB              | Personnel       |
| Name                                  | ES              | HRM             | Same               | Name                 | DB              | Personnel       |
| Person<br>-Directorate                | ES              | HRM             | Same               | Directorate          | DB              | Personnel       |
| Staff-Type                            | ES              | HRM             | Same               | Staff-Type           | DB              | Personnel       |
| Age                                   | ES              | HRM             | Same               | Age                  | DB              | Personnel       |
| Availability                          | ES              | HRM             | Same               | Current-Status       | DB              | Personnel       |
| Average-Grade                         | ES              | HRM             | Same               | Average-Mark         | DB              | Personnel       |
| Person-Skill-1                        | ES              | HRM             | Same               | Skill-1              | DB              | Personnel       |
| Person-Skill-2                        | ES              | HRM             | Same               | Skill-2              | DB              | Personnel       |
| Location                              | ES              | HRM             | Same               | Location             | DB              | Vacancy         |
| Project<br>-Directorate               | ES              | HRM             | Same               | Directorate          | DB              | Vacancy         |
| Job-Required-<br>Age                  | ES              | HRM             | Same               | Required -Age        | DB              | Vacancy         |
| Job-Required<br>-Skill-1              | ES              | HRM             | Same               | Required<br>-Skill-1 | DB              | Vacancy         |
| Job-Required<br>-Skill-2              | ES              | HRM             | Same               | Required<br>-Skill-2 | DB              | Vacancy         |

Step 3 Create coupling classes

A coupling class will be created for the Vacancy database (see Figure 16)

|                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Class Name:</b> Vacancy<br/> <b>Attributes:</b><br/> Job-code: Text<br/> Project-Name: Text<br/> Location: Text<br/> Required-Age: Number<br/> Directorate: Text<br/> Required-Skill-1: Text<br/> Required-Skill-2: Text</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Figure 16 Vacancy Database**

Step 4 and Step 5 can be skipped due to lack of derivable attributes.

Step 6 Create EDS

To integrate the vacancy active class into the existing system, the synonym information in the data dictionary must be changed as shown in the Table 9.

Table 9 Synonym Information for the Information HRM System (enhanced EDS)

HRM.Person-id = Personnel.ID  
 HRM.Name = Personnel.Name  
 HRM.Person-Directorate = Personnel.Directorate  
 HRM.Staff-Type = Personnel.Staff-Type  
 HRM.Age = Personnel.Age  
 HRM.Person-Skill-1 = Personnel.Skill-1  
 HRM.Person-Skill-2 = Personnel.Skill-2  
 HRM.Availability = Availability.Availability  
 HRM.Average-Grade = Average-Grade.Average-Grade  
 HRM.Location = Vacancy.Location  
 HRM.Project-Directorate = Vacancy.Directorate  
 HRM.Job-Required-Age = Vacancy.Required-Age  
 HRM.Job-Required-Skill-1 = Vacancy-Required-Skill-1  
 HRM.Job-Required-Skill-2 = Vacancy-Required-Skill-2

As a result, a new Information HRM System EDS can be developed. Figure 17 and Figure 18 show a prototype of the IRMS.

| <b>Human Resource Management Expert Database System<br/>(HRM EDS)</b> |             |
|-----------------------------------------------------------------------|-------------|
| Please Enter Job Code: [0001 ]                                        |             |
| What Special Conditions Would You Like To Set Up In This Transaction? |             |
| 1. Skill                                                              | 2. Location |
| 3. Family Situation                                                   | 4. Hobby    |
| 5. Sex                                                                | 6. Age      |
| 7. Current Position                                                   | 8. Others   |
| 9. No                                                                 |             |
| Please Select: [ 2 ]                                                  |             |

Figure 17 Sample HRM EDS Interactive Session

| <b>My Advice for the Job code 0001 (Chief Programmer in Project 2 of Directorate 1) is:<br/>There are 3 persons who are suitable for this job</b> |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Name: Russell Parsons                                                                                                                             | Person: 1/3 |
| Suitability Rating: 100%                                                                                                                          |             |
| Current Position: Chief Programmer in Project 1 of Directorate 1                                                                                  |             |
| Reasons: 1. 10 years experience in this type of job                                                                                               |             |
| 2. Full skill requirement                                                                                                                         |             |
| 3. Review marking good                                                                                                                            |             |
| 4. Lives close to the work place.                                                                                                                 |             |
| Warning: He is also suitable for Job 0004 and Job 0007.                                                                                           |             |

Figure 18 Sample HRM EDS Conclusion

## 5 PROTOTYPE OF SUNDERLAND EXPERT DATABASE SYSTEM DEVELOPMENT(SEDSDT)

SEDSDT is an open architecture and can couple many different modules into the frame model. These modules can be grouped into three generic classes; i.e. the knowledge representation module, the expert system coupling module, and the database coupling module as shown in Figure 19. The knowledge representation module presents the knowledge which is described by the language of SEDSDT, called the K-Language, an internal stored procedure language. The expert system coupling module represents knowledge which is encapsulated in an existing external expert system shell, such as Crystal [22] and GoldWorks [23].

The database coupling module represents the 'knowledge' which is encapsulated in an existing external database system, such as dBase and Oracle. The ability to attach procedures to objects enables behavioral models of objects and expertise in an application domain to be constructed. This allows an extremely effective and efficient form of object-oriented programming whereby objects represented by record-like data structures can automatically respond to method calls. The attached procedure follows the IF-THEN structure which enables representation of production rules and normal procedures. The coupling procedure for an object is also an attached procedure within the object.

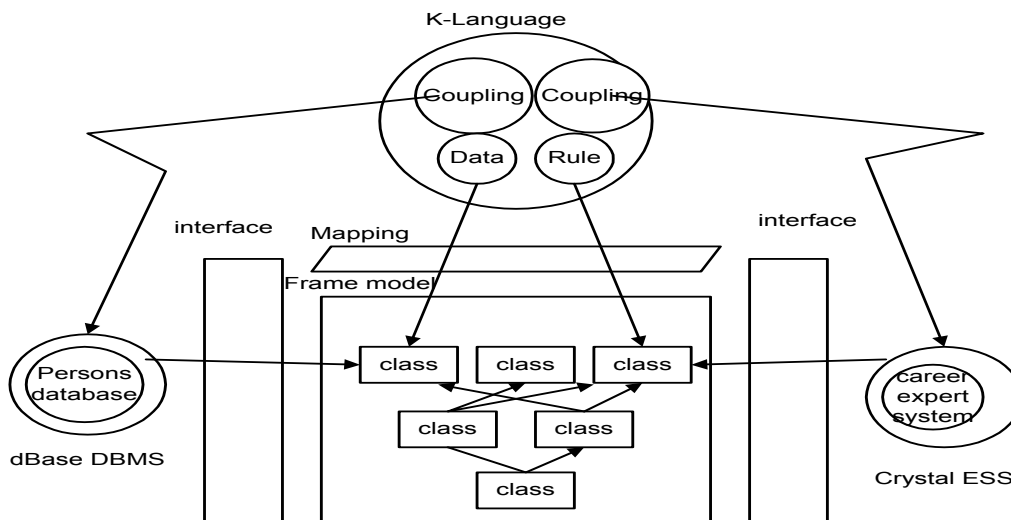


Figure 19 The Architecture of SEDSDT

## 6 CONCLUSION

This paper presents a unique solution in integrating DBS and ES into EDS by using frame model with the emphasis in reengineering. We describe the need of reengineering ES (or DBS) for the purpose of updating ES (or DBS) information by integrating it with DBS (or ES) to form an EDS. The users can reengineer existing ES and DBS by integrating them into an EDS. The technique to integrate an ES and a DBS is to form a common frame model for both of them. It takes each frame model as a class which consists of class name, static attribute, dynamic methods and constraints. These frame models form coupling classes which extract data from source DBS, or rules from source ES. To resolve the naming conflict between the source ES and the source DBS, synonym table is formed to link them together by using a set of common names for their attributes (i.e. resolve naming conflict). With the synonym tables, the source ES, source DBS, coupling class and active class can pass information via messages to each other. The resultant EDS thus becomes a knowledge-based because it consists of both ES and DBS information, and the application knowledge from the users after analysis. System developer can then use the EDS to develop new application.

## REFERENCE

1. Smith, P., Bloor, C. Huang, Shi-Ming, and Gillies, A. (1995), "The need for re-engineering when integrating expert system and database technology", *The proceeding of the 6<sup>th</sup> international Hong Kong Computer Society Database Workshop*, Database re-engineering and interoperability, pp14-23.
2. Deductive Systems Ltd., (1988), "Generis:User", *Menu, Deductive Systems Ltd*, Middlesex UB83PQ, UK
3. Robinson, A.E., (1990), "Current Ideas in Knowledge-Base Management Systems", *Information and Software Technology*, Vol 32, No 4, pp266-273.
4. Frost, R.A., (1987), "Introduction to Knowledge-Base Systems", *William Collins*, New York, ISBN 0-00-383114-0.
5. Stonebraker, M. and Rowe, L.A. (1987) "The Design of POSTGRES", *University of California at Berkeley*, Electronic Research Laboratory, Internal Memorandum UCB/ERL 85/95.
6. Jarke, M. and Vassiliou, Y., (1984) "Databases and Expert Systems: Opportunities and Architectures for Integration", *New Applications of Database System*, Academic Press, London

7. Lu, J.J., Nerode, A. And Subrabmanian, V.S., (1996), "Hybrid Knowledge Base", *IEEE Transactions on Knowledge and Data Engineering*, Vol 8, No 5, pp773-785.
8. Shan, Ming-Chien, ed. (1995), "Pegasus: A Heterogeneous Information Management System", *Modern Database System*, Won Kim(ed.) Addison Wesley publish Co., ISBN 0-201-59098-0, pp664-682.
9. Brachman, R. J., and Levesque, H.J., (1986), "What Makes a Knowledge Base Knowledgeable?" A View of Databases from the Knowledge Level", Proceedings of the 1<sup>st</sup> International Conference on Expert Database Systems, Benjamin/Cummings, pp69-78, USA.
10. McInnes, S.T., (1987), "Knowledge Representation and Access Structures for Large Knowledge Bases: The Generic Relational Model and Its Implementation", Ph.D. Thesis, Department of Computer Science, University of Strathclyde.
11. Buchanan, B.G. and Feigenbaum, E.A. (1978), "DENDRAL and Meta-DENDRAL: Their Applications Dimension", *Artificial Intelligence*, Vol. 11, pp5-24.
12. Forst, R.A., (1987), "Introduction to Knowledge-Base Systems", William Collins, New York, ISBN 0-00-383114-0.
13. Guha, L., (1990) "Building Large Knowledge-Based Systems - Representation and Inference in the Cyc Project", Addison Wesley, New York, ISBN 0-201-517752-3.
14. Laird, J.E., Newell, A. and Rosenbloom, P.S. (1987)"SOAR: An Architecture for General Intelligence", *Artificial Intelligence*, Vol 33, pp 1-64.
15. Fox M.S., Wright, J.M. and David, A. (1986) "Experiences with SRL: An Analysis of a Frame-Based Knowledge Representation", *Proceedings of the 1<sup>st</sup> international Workshop, Benjamin/Cummings,USA.*
16. Fikes, R. and Kehler, T., (1985), "The Role of Frame-Based Representation in Reasoning", *Communications of the ACM*, Vol. 28, No. 9, September 1985, pp904-920.
17. Zarr, G.P. (1992), "Expert Database Systems: State of The Art", Tutorial Documents of the First World Congress in Expert Systems, USA.
18. Huang, Shi-Ming., Smith, P., Tait, J.I. and Pollitt, S., (1993), "A Survey of Approaches to Commercial Expert Database System Development Tools", Occasional Paper 93-4, *University of Sunderland.*
19. Gray, P.M.D., Kulkarni, K.G., and Paton, N.M., (1992), "Object-Oriented Databases: A Semantic Data Model Approach", *Prentice Hall*, New Jersey, ISBN 0-13-630203-3.
20. Houtsma, M.A.W. and Apers, P.M.G, (1990), "Data and Knowledge Model: A Proposal", *Advances in Database Programming Languages*, ACM Press, New York, ISBN 0-201-50257-7.
21. Carma McClure, (1992), "The Three Rs of Software Automation: Reengineering, Repository, Reusability", *Prentice Hall*, ISBN 0-13-915240-7.
22. Intelligent Environments Ltd., (1987), "Crystal User Manual", *Intelligent Environments Ltd.*, Richomond, Surrey, TW105BR, England.
23. Casey, J.S., (1989), "GoldWorks II for the SUN-3 or SUN 386i", *Gold Hills Computers Inc.*, 26 Landsdiwbe street, Cambridge, MA 02139, USA.