Application of Knowledge-Based System in Automated Data Warehouse Design

Opim Salim Sitompul and Shahrul Azman Mohd. Noah

Faculty of Information Science & Technology
Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor
Tel : 03-89216653
E-mails:{oss19877, samn} @ftsm.ukm.my

ABSTRACT

Data warehouse has become more and more popular for an enterprise as a data repository system. Yet tools to appropriately design its conceptual model are rarely available, even though this model is known as a key for the successful of the overall design. In this paper we propose an approach and a tool to guide the decision makers in designing data warehouse conceptual model based on the Entity Relationship (ER) model of the existing operational database systems. Using this approach, the ER model is automatically transformed into the multidimensional model.

Keywords

Automated Tool, Data Warehouse Design

1.0 INTRODUCTION

Data maintained in the operational databases are continually increasing as they are accumulated from the day-to-day enterprise operation. The enterprise executives eventually realize that they need an appropriate tool to manage and access those data in order to acquire useful information. As decision makers, analyzing data trends and correlations from different aspects of the business are the most valuable business query requirements (Gardner, 1998).

Data warehousing is a technology that allows information to be easily and efficiently accessed for decision-making activities (Bellatreche et al., 1998). In a data warehouse system, different operational data sources are collected into a data repository to provide access for information access tools such as OLAP (Online Analytical Processing), data visualization, executive information systems and decision support systems (EIS/DSS), Spreadsheet, data mining, and other development languages (Gardner, 1998; Gray and Watson, 1998).

In its implementation, data warehouse system relies on multidimensional model. By this model, data to be analyzed is represented conceptually as fact schemes, which consists of quantifying values (facts) and qualifying context (dimensions) specified by a lattice of dimension levels (Hüsemann et al., 2000). At the center of a fact scheme is the measurements of the business that contain the numeric and additive fields, measured at the intersection of all of the dimension values (Lechtenbörger and Vossen. 2003). In addition, dimension hierarchies are also created to indicate all plausible aggregation among the dimension of related measures (Letz et al., 2002).

The multidimensional model as a conceptual view plays an important role in data warehouse design. The model can be considered as a mediator between system analysts and enterprise managers as they work together in formulating the data warehouse requirements. At this conceptual level, analysts and managers could bring in their ideas in terms that they understood, avoiding both technical and theoretical jargons. In addition, the conceptual design is the basic building block for subsequent stages of data warehouse design. It is considered as the most important phase for the successful of the overall design where modeling errors could be detected early and the schema could be extended easily (Hüsemann et al., 2000; Tryfona et al., 1999).

Even though the conceptual model is known as a key for the successful of the overall design of data warehouse, yet tools to help designing this model are rarely available. In this research we propose our approach in designing the conceptual model of the data warehouse using a transformation-oriented methodology whereby the ER model of an operational database is transformed into the multidimensional model. The description about the knowledge base system used in the architecture of the DWExpert to implement that approach is also provided.

2.0 RELATED RESEARCH

The majority approaches taken for data warehouse design are based on database design, which consists of requirement analysis and specification, conceptual design, logical design, and physical design (Elmasri and Navathe, 2000). As for the conceptual design, the existing methodologies are mainly based on the entity-relationship (ER) model in which the ER model is gradually transformed into multidimensional model. In performing the transformation, however, there are variety of techniques being used, such as attributes tree (Golfarelli et al., 1998), one-to-one translation into star schema (Ttryfona et al., 1999), multidimensional normal form (Hüsemann et al., 2000), entity classification (Moody and Kortink, 2000), and table data structures (Phipps and Davis, 2002).

Basically, the methodology used in developing the conceptual data warehouse design in the form of multidimensional model consists of two major tasks, namely determining facts/measures, and setting up dimensions/hierarchies. In determining facts and measures, the facts are obtained from entities that have numeric attributes and the measures are got from the numeric attributes. On the other hand, in determining dimensions and dimension hierarchies, the approaches are varies from entity attributes (Golfarelli et al., 1998), relationship sets (Tryfona et al., 1999), component entities (Moody and Kortink, 2000), to non-numeric, non-key, non-dates attributes, and many relationships (Phipps and Davis, 2002).

Furthermore, some endeavors have also been given on the development of automatic tool to implement the conceptual design methodology. In this case, we could find some research works such as Golfarelli et al. (1998) who developed the conceptual model semi-automatically and presented algorithms in building attributes tree as well as algorithms for pruning and grafting the attribute trees. In addition, work by Phipps and Davis (2002) have also proposed an automated design methodology and presented an algorithm to build the conceptual design.

Explicitly, some preliminary works on the development of CASE tool for data warehouse design could also be found on several research works. In their CASE tool, Miller and Nilakanta (1998) and Wu et al. (2001) focused on the generation of SQL query from a set of operational relational database in order to build a data warehouse. The user interface facilitates the creation of query in the form of command line by choosing a list of attributes and conditions. Franconi and Ng (2000) developed i●com, an intelligent conceptual modeling equipped with a very powerful description logics reasoning server as its background inference engine. There are two conceptual modeling scenarios for their tool, namely source integration and multidimensional aggregation. Another CASE tool to implement data warehouse design methodology has also been developed by Golfarelli et al. (2001, 2002). The WAND (Warehouse Integrated Designer) was developed to carry out data warehouse conceptual design semi-automatically from relational operational sources, defining a core workload on the conceptual scheme, acquiring data volume, and carry out logical and physical design to produce a data mart scheme. The tool also generates SQL statement for creating tables and indexes as output.

3.0 METHODOLOGY

The methodology used in this research work is called the transformation-oriented methodology (Marotta and Ruggia, 2002), which progressively transforms the ER model into multidimensional model in five stages as depicted in Figure 1.

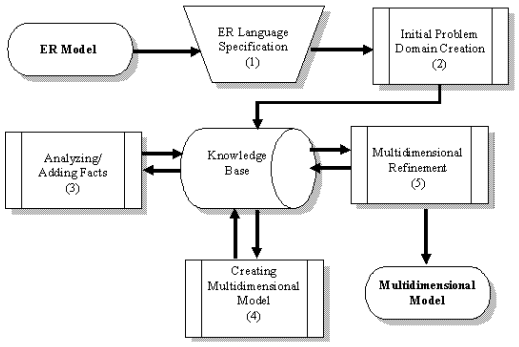

Figure 1: The transformation-oriented methodology

The ER language specification stage is a manual process to translate the source input represented in the form of ER model into a language specification model. Each entity in the ER model is configured in the language specification model as a class structure with the name of the entity as the class name and the entity properties as the class properties. The entity properties that are

specified in the class structure are attribute, identifier, subclass, aggregation, and relationship. The translation of the ER model into the specification language model is guided by a set of syntax rules. The language specification model that becomes the initial representation of the application domain (the problem domain model) is saved into a text file.

The initial problem domain creation stage is a stage responsible for the transformation of the language specification model created at the first stage into the initial problem domain model. The problem domain model is represented as a list of compound terms (Luger and Stubblefield, 1998), which are ordered in *property-entity-value* pairs. The initial problem domain will include the non-null value properties of each entity found in the specification language model. In addition, this stage is also responsible for the creation of a database in which the problem domain is stored.

The third stage is the analysis of the problem domain model in order to obtain new facts. The analysis is performed by a set of inference and translation rules using production and procedural rules (Sitompul and Noah, 2003). Those analysis will cause some new facts are added into the database. The new added facts, however, may cause redundancies and inconsistencies within the database. Thus, in this stage some diagnostic tasks will be performed in order to prevent the database from such discrepancies. After the analysis-synthesis tasks are completed, this stage also performs an important task of classifying each entity attributes into *numeric, temporal*, and *other* categories as the basis for the creation of the multidimensional model as suggested in Phipps and Davis (2002).

The fourth stage is the creation of the multidimensional model. The multidimensional constructs are created from the three categories of the attributes. Fact is created from an entity that has numeric attribute and will be called the fact entity. This fact will become a candidate fact scheme, whereby the numeric attribute will become the fact attribute (measure). The dimensions of the multidimensional model are created from the *temporal* attribute and *other* attribute categories of the entity. The *temporal* attribute will become the temporal dimension and the *other* attribute will add other dimensions into the fact scheme. In addition, the fact scheme will also obtain dimensions from the relationship property of the fact entity. In this case, each one-to-many relationship between the fact entity and another entity will create a new dimension. Recursively, if there is a one-to-many relationship between the other entity and yet another entity, a new dimension level will be added, forming a dimension hierarchy.

The last stage is a refinement of the multidimensional model obtained from the previous stages. As those previous stages are automatic processes without any user interventions, the resulted model will only portray the basic multidimensional constructs similar to how they are established in the application domain model. Therefore, the refinement is necessary to further integrate user's requirements into the model by modifying measures, temporal dimension, and dimension hierarchies.

4.0 THE ARCHITECTURE

In order to implement the aforementioned methodology, an automated tool called the DWExpert has been developed. The architecture of the tool as promoted in Noah and Williams (2003b) consists of three layers, namely the user interface, the inference engine, and the knowledge base as depicted in Figure 2.

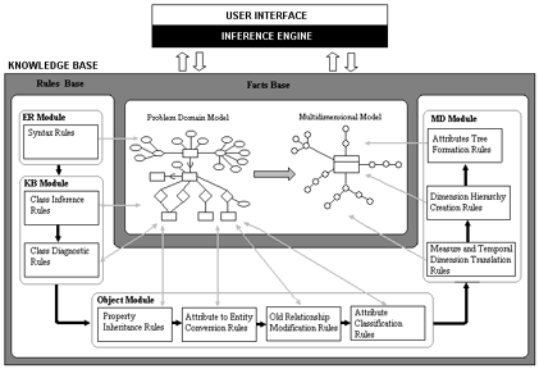

Figure 2: Architecture of the DWExpert

The user interface is made up of a main window and a pop-up window. The main window is where the interactions between the user and the tool established during the transformation process, while the pop-up window is for the interactions during the refinement process. During the transformation process, however, the interaction is only minimal. The actions perform by the user during this process are loading the application domain file and running the transformation process. After the transformation process is completed, the results of the transformation are

displayed in four folders in the main window. From these folders, the user could see the list of each entity and its description, the problem domain database, the list of each object and its description, and the candidate fact schemes of the multidimensional model. On the other hand, during the refinement process the user interactions are more intensive. During this process the user interacts with the tool by a sequence of simple question-answer dialogs, whereby the user response only with one-character option except for the construct being modified. A screenshot of the DWExpert user interface can be seen in Figure 3.

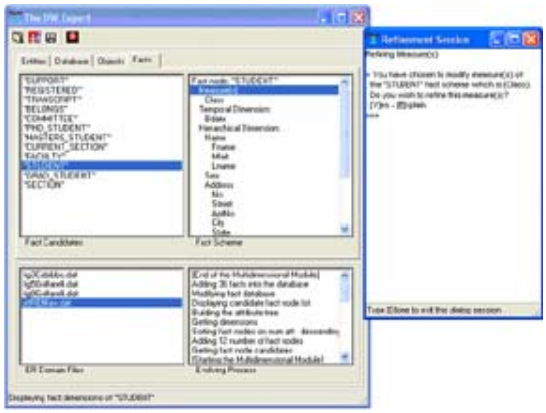

Figure 3: Screenshot of the DWExpert user interface

The inference engine is in charge of controlling the interaction between the user and the tool, such as loading the application domain into memory, directing inputs to the appropriate processors, and selecting which rules to fire during the transformation process. These tasks are divided into five modules corresponds to the five stages of the transformation-oriented methodology, i.e. the ER module, the database (DB) module, the object module, the multidimensional (MD) module, and the refinement module. In this case, the inference engine is responsible for sequencing the processing tasks and passing information from one module to another module.

The knowledge base consists of the rules base and the facts base. Rules in the rules base could be divided into three categories, i.e. syntax rules, production rules, and procedural rules. The syntax rules implemented in the ER module are rules used to translate the ER model of the application domain into the specification language model. These rules regulate the mapping between the entity of the ER model and the class construct of the specification language model. The production rules implemented in the database module functions as the inference mechanism in inheriting indirect subclass, direct/indirect superclass, and acquiring the aggregation parts. In applying the production rules, the tool uses the forward chaining technique in order to arrive at conclusions (Luger and Stubblefield, 1998; Negnevitsky, 2002). Lastly, the procedural rules implemented in the object and multidimensional modules are for the analysis and diagnosis of the problem domain using procedural representations (Barr and Feigenbaum, 1981).

The analysis rules are used in the object module to perform several tasks, such as inheriting attributes and identifiers from the superclass of an entity (if it is a subclass of another entity); converting numeric attributes of a relationship into entities; modifying relationships as the result of the numeric attribute conversions; and classifying attributes into *numeric*, *temporal*, and *other* categories. In the multidimensional module, the analysis rules are responsible for deriving measures and temporal dimension; creating dimension hierarchies, and generating attributes tree. Meanwhile, the diagnostic rules are implemented in the database, object, and multidimensional modules to ensure the integrity of the database by removing redundancies and inconsistencies as the result of augmenting new facts into the database or modifying an existing fact.

The facts base maintains the initial application domain representation (the problem domain model) in the form of compound terms within a database. During the transformation process, the database would be progressively analyzed and synthesized in order to transform the problem domain into the multidimensional model. In the facts base, the multidimensional model is represented in the form of fact schemes.

5.0 A UNIVERSITY EXAMPLE

To illustrate the whole process of the transformation-oriented methodology, we will look at one example from a university domain adapted from Elmasri and Navathe (2000). For this illustration, we only take a portion of the ER diagram for a *Student* entity as depicted in Figure 4. The transformation process will be described following the five stages of the methodology.

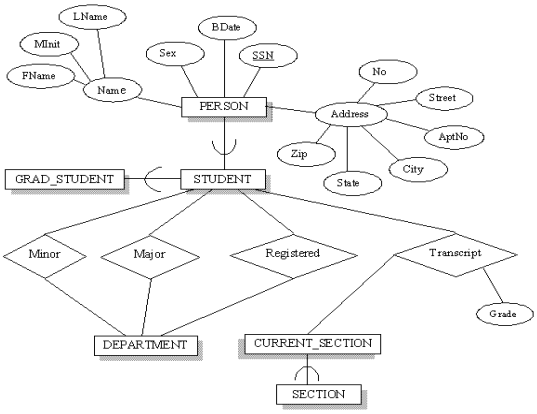

Figure 4: Portion of ER diagram for Student entity

5.1 The ER Language Specification

The objective of this stage is to translate the ER model into the language specification model, so we will look at each entity in the ER model to examine the properties that will be translated into the language specification model, i.e. attribute, identifier, subclass, aggregation, and relationship. The top-level syntax rule for translating the diagram into the specification language model is shown in figure 5.

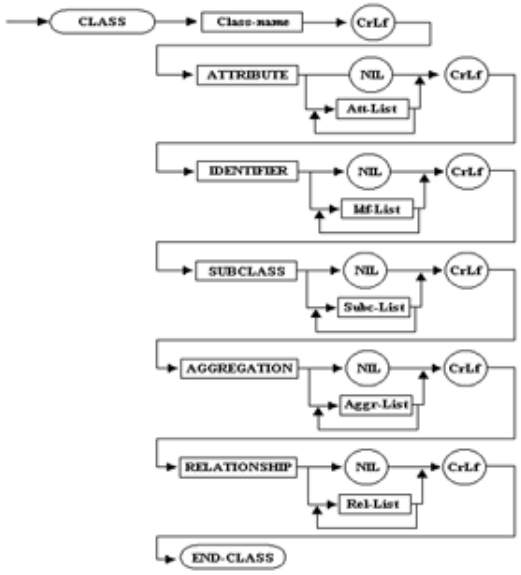

Figure 5: Top-level Syntax Rule for the Class Contruct

From the ER model in Figure 4, the *Student* entity has several properties, namely: an attribute *Class*, a subclass *Grad_Student* and some relationships *Minor*, *Major*, *Registered*, and *Transcript*. In the language specification model, only the non-null properties will be recorded, while those that are unavailable will be set to NIL. Referring to the syntax rule, the *Student* entity will be recorded as the following:

```
CLASS "STUDENT"
ATTRIBUTE (("Class": Integer))
IDENTIFIER NIL
SUBCLASS ("GRAD_STUDENT")
AGGREGATION NIL
RELATIONSHIP (("Minor" "DEPARTMENT" "NIL" "(1 1)" "(1 n)")\
                ("Major" "DEPARTMENT" "NIL" "(1 1)" "(1 n)")\
     ("Registered" "CURRENT_SECTION" "(("Count": Integer))"
                                        "(1 n)" "(1 m)")\
     ("Transcript" "SECTION" "(("Grade": Float))" "(1 n)" "(1 m)"))
End-Class
```

The class construct formulated above is a direct mapping of the ER constructs. One thing that the user should consider is the type of each attribute, which is not shown in the ER model. The relationship component of the language syntax is a little bit complex, where it is composed of five parts: the name of the relationship, the participating entity, the relationship attribute, first (min, max) relationship constraint, and second (min, max) relationship constraint.

5.2 Initial Problem Domain Creation

In this stage, the tool will automatically transform the language specification model into the initial problem domain model, which is represented in the form of *property-entity-value* pairs. The result of this transformation would be:

```
Has-Attribute “STUDENT” ((“Class”: Integer))
Has-Subclass “STUDENT” (“GRAD_STUDENT”)
Has-Relationship “STUDENT”
   ((“Minor”, “DEPARTMENT” “NIL” “(1 1)” “(1 n)”)
   (“Major”, “DEPARTMENT” “NIL” “(1 1)” “(1 n)”)
   (“Registered”, “CURRENT_SECTION” “((“Count”: Integer))”
                                    “(1 n)” “(1 m)”)
   (“Transcript”, “SECTION” “((“Grade”: Float))” “(1 n)” “(1 m)”)
```

Subsequently, the initial problem domain is stored into the database.

The next step is the inference step, inferring indirect subclass, direct/indirect superclass, and deriving the aggregation objects using the forward chaining method. From this process, the subclass fact is modified and new fact related to superclass is added to the database, namely:

```
Has-Subclass “STUDENT” ((“GRAD_STUDENT”)
                        (“MASTERS_STUDENT” “PHD_STUDENT”))
Has-Superclass “STUDENT” (“PERSON”]
```

5.3 Analyzing/Adding Facts

This stage is the core of the transformation process where each fact within the database is progressively analyzed. In this stage there are four steps that are performed. The first step is examining if the entity has a superclass and inheriting attributes and identifiers from the superclass if one exists. Next the tool will examine the existence of numeric relationship attribute among the available relationships. If one exists, it is converted into an entity. As a consequence, the exiting relationships should be modified to reflect the changes. Lastly, the tool will classify the attributes of the entity into *numeric*, *temporal*, and *other* categories. As the result of this stage, the database is modified and new facts are augmented as follows:

```
(Has-Attribute "STUDENT"
 (((Numeric-Att (Class . Integer)) (Date-Att) (Other-Att))
  (Inherited-Attribute
   ((Numeric-Att) (Date-Att (Bdate . Date))
    (Other-Att (Composite . Name) (Fname . String[15])
     (MInit . String[3]) (Lname . String[20]) (Composite . End)
     (Ssn . String[12]) (Sex . String[1]) (Composite . Address)
     (No . String[4]) (Street . String[20]) (AptNo . String[4])
     (City . String[15]) (State . String[2]) (Zip . String[5])
     (Composite . End)))))))
(Has-Identifier "STUDENT" (Inherited ("Ssn")))
(Has-Subclass "STUDENT" (("GRAD_STUDENT")
      ("PHD_STUDENT" "MASTERS_STUDENT")))
(Has-Superclass "STUDENT" ("PERSON"))
(Has-Relationship "STUDENT"
 (((Name . of) (Participating-obj . TRANSCRIPT) (Rel-Attribute . NIL)
   (First-constraint . (1 n)) (Second-constraint . (1 1)))
  ((Name . of) (Participating-obj . REGISTERED) (Rel-Attribute . NIL)
   (First-constraint . (1 n)) (Second-constraint . (1 1)))
  ((Name . Major) (Participating-obj . DEPARTMENT)
   (Rel-Attribute . NIL) (First-constraint . (1 1))
   (Second-constraint . (1 n)))
  ((Name . Minor) (Participating-obj . DEPARTMENT)
   (Rel-Attribute . NIL) (First-constraint . (1 1))
   (Second-constraint . (1 n)))))
```

5.4 Creating Multidimensional Model

This step is a process to generate the multidimensional model constructs based on the categories of attributes obtained in the previous stage. In this case, the *Student* entity is eligible to be a fact scheme because the entity has a numeric attribute, namely *Class*. This attribute is then specified as the measure of the fact scheme. The temporal dimension is created from the temporal attribute of the *Student* entity, which is *BDate*. The *other* attribute together with the one-to-many relationships of the *Student* entity are then converted into dimension hierarchies. As a result, the following facts are added into the database:

```
(Has-Measure "STUDENT" ((Class . Integer)))
(Has-Dimension "STUDENT"
 ((Temporal Dimension ((Bdate . Date)))
  (Other Dimension
   ((Composite . Name) (Fname . String[15]) (MInit . String[3])
    (Lname . String[20]) (Composite . End) (Ssn . String[12])
    (Sex . String[1]) (Composite . Address) (No . String[4])
    (Street . String[20]) (AptNo . String[4]) (City . String[15])
    (State . String[2]) (Zip . String[5]) (Composite . End)))))
(Has-Hierarchy "STUDENT"
 (("PERSON" . 0) ("DEPARTMENT" . 0) ("COLLEGE" . 1)
("DEPARTMENT" . 0) ("COLLEGE" . 1)))
```

In order to identify each dimension level in a dimension hierarchy, the tool uses an integer number to denote its position. The diagnosis task could detect any redundancy that may exist in a dimension hierarchy by examining whether each entity has the same position number in a particular hierarchy. In the above example, the dimension hierarchy created for the *Department* and *College* entities are recorded twice because there are two one-to-many relationships with the same participating entities, namely the *Major* and *Minor* relationships. To remove such redundancy, the multidimensional constructs are refined accordingly.

The next step performed in this stage is the creation of attributes tree from the *other* attribute of the fact entity and the attributes of each entity specified in dimension hierarchies. From the above example, those entities are *Person*, *Department*, and *College*. The attributes tree created from this process can be illustrated as in Figure 6.

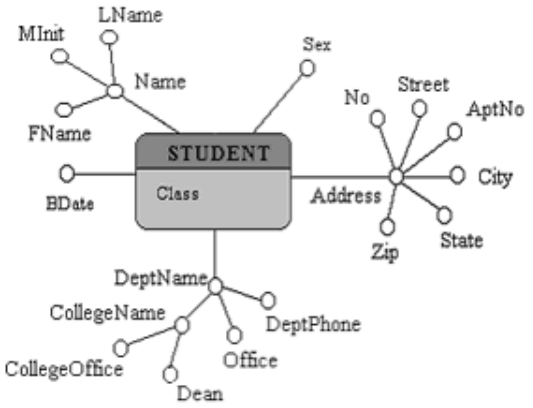

Figure 6: Attributes tree for the Student fact scheme

5.5 Refining Multidimensional Model

Refining the multidimensional model means modifying its dimensional constructs in order to suit user's requirements. For this purpose, the user could modify the measure, temporal dimension, and dimension hierarchy that is generated by the

automated tool. The refinement process is done interactively between the user and the tool through a simple question-answer dialogs.

Measure as the focus of interest in data warehouse design is the first thing that the user would consider refining. Since the tool automatically derived this measure from the numeric attribute of the fact entity, it might be inappropriate to what the user needs. Supposing the user need to count the number of undergraduate and graduate students instead of counting the student on each class, then the *Class* measure should be modified, for instance, into the appropriate measures such as *Number_of_Undergraduate* and *Number_of_Graduate*. The most important thing to consider in modifying measure is the additive nature of this construct, whereby the user should ensure that operations such as sum, count, average, maximum, and minimum could be performed.

As for the temporal dimension, the user almost always modifies this dimension because this construct is rarely available in the initial problem domain model or unsuitable with user's requirements. Continuing with the above example, since the measure is the count of the student, then the existing temporal dimension, which is *BDate*, may be modified into *year1*, *year5*, and *year10* to enable the analysis of the number of student on five-year basis.

Lastly, by referring to the attributes tree, the user could modify the dimension hierarchies by pruning, grafting, and aggregating the attributes tree. Pruning the attributes tree is intended to remove a dimension node and all its descendants, while grafting means removing only the specified dimension node but still maintaining all it's descendant. Moreover, aggregating the attributes tree is a way of combining one or more nodes or adding new nodes into an existing dimension in order to create a new aggregate dimension. For example, the user might need to analyze to number of students based on their sex, place of origin, and department in five-year interval. The refinement process conducted by the user could generate the fact scheme as depicted in Figure 7.

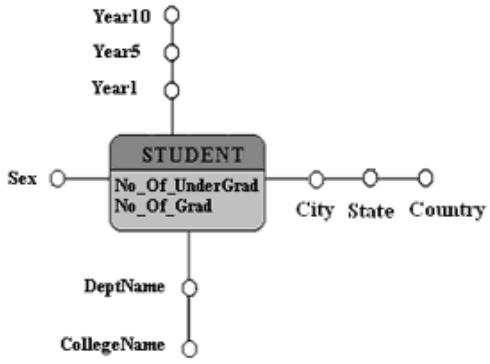

Figure 7: Refined fact scheme for the Student entity

6.0 CONCLUSIONS

In this paper we have described our approach to the conceptual data warehouse design using the transformation-oriented methodology. The approach employed has shown a preliminary step in applying the artificial intelligence (AI) technique to the field of data warehouse design as has been achieved within the context of database analysis and design (Noah and Williams, 2003a; 2003b).

In order to implement the aforementioned design approach, a tool called DWExpert has been developed. For the time being, the tool could only provide the multidimensional model in the form of plain text. It is our intention to further enhances the capability as well as the functionality of the tool in our future research.

7.0 REFERENCES

Barr, A. and Feigenbaum, E. A., (1981). *The Handbook of Artificial Intelligence*, Volume I, Los Altos, CA: William Kaufmann.

Bellatreche, L., Karlapalem, K., and Mohania, M, (1998). *Some Issues in Design of Data Warehousing Systems*. In Developing Quality Complex Database Systems: Practices, Techniques, and Technologies, Dr. Shirley A. Becker (Eds.). 125-172. Ideas Group Publishing.

Elmasri, R. and Navathe, S. B., (2000). *Fundamentals of Database Systems,* 3rd Edition, Reading, Mass.: Addison-Wesley.

Franconi, E. and Ng, G., (2000). *The i.com Tool for Intelligent Conceptual Modeling*. In

Proceedings of 7th International Workshop on Knowledge Representation meets Databases (KRDB-2000). 42-53. Berlin, Germany.

Gardner, S. R. (1998). *Building the data warehouse*. *Communication of the ACM*, 41(9): 52-60.

Golfarelli, M., Maio, D. and Rizzi, S., (1998). *Conceptual Design of Data Warehouses from E/R Schemes*. In Proceedings of 31st Hawaii International Conference on System Sciences. 334-343. Kona, Hawaii.

Golfarelli, M. and Rizzi, S., (2001). *WAND: A CASE Tool for Data Warehouse Design*. In Demo Proceedings of 17th International Conference on Data Engineering (ICDE'2001). 7-9. Heidelberg, Germany.

Golfarelli, M., Rizzi, S. and Saltarelli, E., (2002). *WAND: A CASE Tool for Workload-Based Design of a Data Mart*. In Proceedings Decimo Convegno Nazionale su Sistemi Evoluti Per Basi Di Dati. 422-426. Portoferraio, Italy.

Gray, P. and Watson, H. J., (1998). *Present and future directions in data warehousing*. *DATA BASE*. 29(3): 83-90.

Hüsemann, B., Lechtenbörger, J. and Vossen, G., (2000). *Conceptual data warehouse design*. In *Proceedings of the International Workshop on Design and Management of Data Warehouse (DMDW '2000)*. 6:1-11. Stockholm, Sweden.

Lechtenbörger, J., and Vossen, G., (2003). *Multidimensional Normal Forms for Data Warehouse Design*. *Information Systems.* 28(5): 415-434.

Letz, C., Henn, E. T., and Vossen, G., (2002). *Consistency in Data Warehouse Dimensions*. In Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02). 224-232. Edmonton, Canada.

Luger, G. F. and Stubblefield, W. A., (1998). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 3rd Edition, London: Addison-Wesley.

Marotta, A. and Ruggia, R, (2002). *Data Warehouse Design: A Schema-Transformation Approach*. In Proceedings of the XXII International Conference of the Chilean Computer Science Society (SCCC'02). 153-161. Montevideo, Uruguay.

Miller, L. and Nilakanta, S., (1998). *Data Warehouse Modeler: a CASE Tool for Data Warehouse Design*. In Proceedings of 31st Annual Hawaii International Conference on System Sciences. : 42-48. Kona, Hawaii.

Moody, D. and Kortink, M.A.R., (2000). *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design*. In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000). 5-1 – 5-12. Stockholm, Sweden.

Negnevitsky, M., (2002). *Artificial Intelligence: A Guide to Intelligent Systems*, London: Addison-Wesley.

Noah, S. A. and Williams, M., (2003a). *Enhancing the Diagnostic Performance of Intelligence Database Design Tools - An Evaluation of the Thesaurus Technique*. *Chiang Mai J. of Science*. 30(1): 7-17.

Noah, S. A. and Williams, M., (2003b). *Intelligent Object Analyser for Conceptual Database Design Model*. (To Appear in *Jurnal Teknologi*).

Phipps, C. and Davis, K. C., (2002). *Automating Data Warehouse Conceptual Schema Design and Evaluation*. In Proceedings of the 4th International Workshop on Design and Management of Data Warehouses (DMDW'2002). 23-32. Toronto, Canada.

Sitompul, O. S. and Noah, S. A. M, (2003). *Rules for the Automatic Translation of ER Model into Multidimensional Model*. In Proceedings of the Conference on Intelligent Systems & Robotics (CISAR 2003). Putrajaya, Malaysia: Advanced Technology Congress 2003 (ATC'2003). (To appear).

Tryfona, N., Busborg, F. and Christiansen, J. G. B., (1999). *starER: A Conceptual Model for Data Warehouse Design*. In Proceedings of the ACM 2nd International Workshop on Data Warehousing and OLAP. 3-8. New York, N.Y.

Wu, L., Miller, L. and Nilakanta, S., (2001). Design of Data Warehouse Using Metadata. Information & Software Technology. 43:109-119.