Constructing an Object-relational Data Warehouse Using Semi-automated Methods

Julie Yu-Chih Liu
Information Management Department, Yuan Ze University

Shih-Ya Lai
Information Management Department, Yuan Ze University

Abstract

As databases become increasingly complicated, building data warehouse with object concepts becomes the essential trend in the future. Designing such a data warehouse usually requires much human involvement. This work proposes a conceptual model for designing an object-relational data warehouse, and provides a semi-automated methodology for deriving the model from the standard documentation of the object-relational database. The construction of the model using the semi-automated methodology is demonstrated in simple examples. Furthermore, this work contributes a mapping mechanism from the conceptual schema to the logical design, and displays the mapping result in both a logical diagram and physical code. The physical code generated based on the logical diagram is refined by using the domain properties in object-relational database to improve the storage cost as well as query performance.

Keywords: Object-relational data warehouse, Conceptual models, Multidimensional data model, Enhanced entity-relational model.

以半自動化方式建構物件關連式資料倉儲

劉俞志
元智大學資訊管理學系

賴詩雅
元智大學資訊管理學系

摘要

隨著資料庫的日漸複雜，以物件為概念的資料倉儲將成為未來的建構趨勢。而物件資料倉儲的設計通常耗費大量人力。本文針對物件關連式資料倉儲的設計提出其概念模型，並提供半自動化的方式由物件關連式資料庫的標準文件推導出此概念模型。簡單的實例將說明此半自動化的建構方式。本研究的貢獻還包括建立概念綱要到邏輯設計的映對機制，映對結果可直接以邏輯圖及程式碼呈現，此外，亦提出如何根據物件關連式資料庫的定義域特質來精煉此程式碼，以改善儲存成本與存取效能。

關鍵字：物件關連式資料倉儲、概念模型、多維度資料模型、增強式實體關聯模型。

1. INTRODUCTION

Data warehousing is a popular approach for storing large volumes of historical data (Kimball 2002; Inmon 1996; Gyssens and Lakshmanan 1996; Li and Wang 1996). It has been widely adopted to support many applications such as data mining tools, decision support systems, and executive information systems (Poe et al. 1997; Akoka et al. 2001; Gray and Waston 1998). As a data warehouse (DW) extracts its data from multiple data sources, designing a DW needs to map the schemas of these data sources to the DW's schema to support the decision making in enterprises. However, the schemas of these data sources are normally complicated. Designing a DW, therefore, requires much involvement of experienced designers.

Many data models have been proposed to facilitate constructing DW during different design phases (conceptual design, logical design and schema creation). Among them, the dimensional fact (DF) model proposed by Galifarelli et al. (1998) is one of the primary design for conceptual phase. Constructing a DW using the DF model has several advantages. First, the DF schema of a DW can be automatically constructed from the Entity Relationship (ER) schemas of the data sources through the mapping processes in literature (Golfarelli et al. 1998), and this greatly expedites DW's conceptual design. Since ER schema has been regarded as a standard documentation for the systems based on the relational data model, this technique is useful in practice. Second, the data hierarchies are well presented in the DF schema so that a typical DW query can be diagrammatically presented in the DF schema to assist query writing. The data hierarchies also help integrating multiple DF schemas by testing their overlapping and inclusion. This is particularly helpful when the DW's data sources are multiple databases or data marts. Third, the DF model clearly distinguishes measures and dimensions, and consequently improves the efficiency of DF's logical design. Fourth, the DF schema can be semi-automatically transformed into DW's logical design by using a mapping mechanism (Golfarelli et al. 1999). Overall, these properties of DF model enable cost-saving and less error-prone in the construction of DW.

The DF model, however, addresses only simple data types like number and string, and cannot handle the complex semantics of object. Most of modern information systems are object-relational, which must handle objects and their relationships to represent complex data such as images, graphics, spatial data and multimedia. While constructing a DW from such data sources, designers no longer benefit from the advantages of the DF model described above. Thus, it is practical to extend the DF model to conceptually represent the data warehouse which involves the concept of objects and their relationships. Such a data warehouse is called an object-relational data warehouse (ORDW) in literature (Gopalkrishnan et al. 1999).

In this work, we first proposes the Extended DF (EDF) as the conceptual model of ORDW,

which not only employs all the modeling concepts of the DF model but also includes the characteristics of objects, namely super-class/subclass and inheritance. In order to construct an ORDW from object-relational data source in which we save more manpower, we then provide a mapping algorithm to transform the Enhanced ER (EER) schemas of the object-relational data sources into the EDF schema of the ORDW. EER schema is the standard conceptual design used for object-relational databases. Next, the EDF schema of the ORDW will be mapped into the logical design of the ORDW as well as the schema in the form of DDL. The resulted schema will be further refined to reduce the cost on storage.

After surveying the literature on data models at different design level in Section 2, in Section 3 we explain the EDF model, and utilize the graphical expression to denote ORDW queries. In Section 4, we propose a methodology for deriving EDF schema from the EER schema, and depict the conversion between them. In Section 5, we develop a mapping algorithm from an EDF schema to the logical design of ORDW, and also provide an approach to refine the mapping result. This approach is applied to a real project from Golfarelli et al. (1998) to illustrate its efficiency on saving storage. Finally, we draw a conclusion in Section 6.

2. RELATED WORK

The design of a DW can be divided into three levels, including the conceptual design, logical design and physical design (Goncalves 2006). Many data models (Sapia et al. 1998; Tryfona et al. 1999; Bækgaard 1999; Agrawal et al.1997; Cabibbo and Torlone 1998; Hacid and Sattler 1997; Kimball 1998; Franconi and Kamble 2003) have been developed for each of them. Some of the proposals also further discuss the query operator or algebra for OLAP as in literature (Vassiliadis 1998; Franconi and Sattler 1999; Pedersen et al. 2001; Thomas and Datta 2001). Some focus on the conceptual level, which provides concept close to the way many users perceive data and hides the details of how data is stored or implemented. Tryfona et al. (1999) combined the properties of ER model and star schema to be a mixed model to describe the DW conceptually. The attributes of facts in the model were categorized into three types, including additive (being able to be summarized), non-additive and having been summarized. This model also includes the concept of objects to express the complex relation between entities. Another ER-like model is proposed by Sapia et al. It is extended to capture multidimensional semantics by adding the entity sets and relationship sets. The relationships in the model are treated as facts (Sapia et al.1998). Bækgaard (1999) used events and entities respectively to present the measured data and dimensions. The star schema in the model consists of several star paths, each of which collects all entities related to a single event ad can be presented in literal. These events are specified with time domains, which could be a time point, a time interval or a set of time points.

Most models that incorporate OO semantics with data warehouse modeling are based on UML. On the basis of aggregation and generalization hierarchies, Akoka et al. (2001) derived multidimensional model from the UML model for different design phases. Nguyen et al. (2000) modeled conceptual multidimensional semantics in term of classes using UML. Abelló et al. (2001) defined different kinds of nodes and arcs to depict the structure of the DW in addition to those used in UML. The data are classified and grouped with respect to different levels of aggregation. The schema in the model can also be described in some different detail levels. Rahayu et al. (2001) extended the star schema to deal with the object feature in the hierarchical dimension of a DW. Gopalikrishnan et al. (1999) designed the object-relational view for DW, which also signifies an OO view to the underlying RDB. Hacid and Sattler (1997) have proposed a formal framework extending description logic for hierarchically structured dimensions. Two kinds of modeling methods (Trujillo et al. 2000; Trujillo et al. 2001; Lujan-Mora et al. 2002; Abello et al. 2001) have been developed to transform a data cube into OO models with class hierarchies. Although the works described above have applied object concepts to handle complex data in data warehouses, a model to deal with the object-relational data warehouses is still lacked.

Some other studies on DW provide a translation mechanism between various models (Fong 1995; Krippendorf and Song 1997; Moody et al. 2000; Akoka et al. 2001). Joseph (1995) furnished the techniques to transform the EER model into the object-oriented model. Krippendorf and Song (1997) presented the process for mapping a star schema into an ER diagram. Reversely, Moody (2000) developed techniques for obtaining a DW schema from an ER schema. With the application of fuzzy technology, Feng and Dillon (2003) supplied a three-layered DW model for capturing and illustrating numerical, categorical and quantifier summaries. Franconi and Kamble (2003) provided a general formalism to encompass several proposals for the data models on the data warehouse, in order to compare formally different data models. Various logical models on DW for OLAP were also discussed and compared in literature (Vassiliadis et al. 1999; Tsois et al. 2001; Abello et al. 2001). These works assist constructing DW in parts of levels. And, a complete three-level procedure of constructing an ORDW is expected.

3. THE ENHANCED DIMENSIONAL FACT (EDF) MODEL

This section describes the basic data-structuring concepts and relational constraints of the EDF model, and demonstrates their use in the design of conceptual schemas for DW applications. The development of this model is based on the ED model of Golgarelli et al to deal with the concepts of objects, including inheritances and compositions. The diagrammatic

notation associated with the EDF model, known as EDF, is also presented.

The EDF schema comprises a set of fact schemas. The basic components of a fact schema are facts, dimensions, hierarchies and inheritances. A fact represents data of interest for analysis, and a dimension presents the transaction attribute chosen to represent a fact. A hierarchy expresses the attributes having the same property but different granularity to aggregate the fact. The inheritance represents the inheritance relationship between dimensions.

3.1 EDF Schema Structure

The structure of EDF schema l is defined formally in the graph model below. The query on an EDF can also be represented in a diagram.

Definition 1. A fact schema is an acyclic-directed graph $G=(V, E)$, where the set of vertices V is a triple $V=(f, D, N)$, where f represents a fact; D denotes the set of all dimension attributes, and N denotes the set of all non-dimension attributes. If G is regarded as a quasi-tree, then the vertex f is its root. The vertex $d \in D$ is a dimension if and only if there exists $(f, d) \in E$. The sub-trees rooted by dimensions are known as hierarchies. Let $p(y)$ denote the parent of vertex y. The in-directed edge $(x, y) \in E$ represents the "many-to-one" relationship of x and y as $x, y \in D$ and $x=p(y)$, or "one-to-one" relationship as $x \in D$, $y \in N$. The directed edge $<x, y> \in E$ represents the inheritance relationship between x and y.

A fact represents a many-to-many relationship among the dimensions, and each combination of values of some dimension attributes defines a fact instance on one of the fact attributes, which is similar to the data cube in DW. A fact f is described by its name and the collection of fact attributes (measures). A fact schema may have no fact attribute, since each fact instance expresses the occurrence of an event, and which coincides logically with the factless fact table described in literature (Kimball 2002). A hierarchy comprises the dimension attributes linked by many-to-one relationships, and the domain of each dimension attribute is assumed to be a set of discrete values. A hierarchy may also include non-dimension attributes. A dimension d defines the finest aggregation granularity among the dimension attributes of the hierarchy rooted by d.

The vertices in a fact schema diagram are four categories, and are described below.

- A fact is represented in an EDF diagram as a rectangular box enclosing the fact name on the top, and the collection of measure names.
- A dimension attribute $d \in D$ is represented by a circle.
- A non-dimension attribute $v \in N$ is denoted by a dot, and therefore is not displayed explicitly.
- A supper-class dimension attribute is represented by a small square with a circle inside, and is linked to its subclass dimension attribute by an edge directed from the super-class to subclass.

Fig.1 displays a small fact schema for Transcript, describing the case of the courses being taken in a school. The name of the fact is Transcript, and the fact attributes are grade and number of students. The fact schema has three dimensions, semester, courses and students. The dimension attribute person is the super-class of dimension student, and the directed edge represents their inheritance relationship. The non-dimension attributes express the additional information for dimension attributes, and cannot be adopted to perform the aggregation. For instance, office describes the office locations of the departments, and cannot be used to aggregate the numbers of students. An example of a fact instance is the collection of the grades of a student for courses.

3.2 Operations of EDF Model

In DW, OLAP operations such as slicing and dicing are performed to shrink the data cube by lowering the range of values in each dimension. Every data cube represents an aggregation result of the transaction data. In the EDF model, a fact instance corresponds to a data cube. The

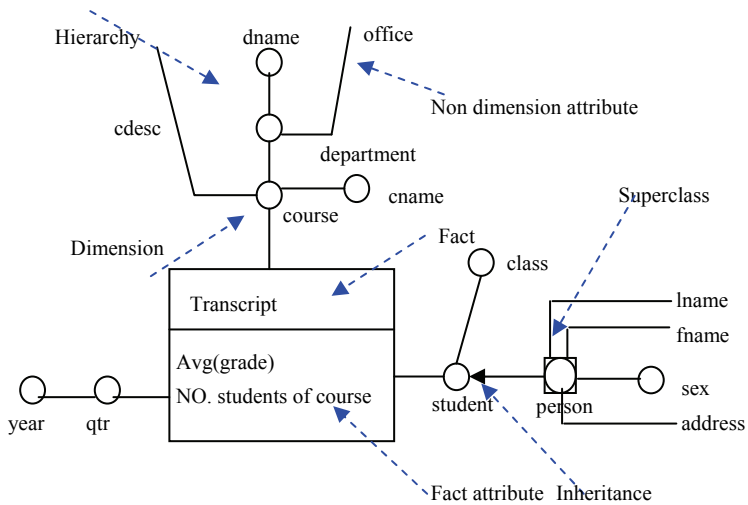

Fig 1：A simple fact schema with three dimensions

traversal of a hierarchy (except the non-dimension attributes) represents the roll-up or drill-down operations in OLAP. The operation of EDF model describes the presentation of querying an ORDW. Every query employs operators to aggregate fact instances into clusters for the fact attributes. The aggregation operators include sum, count, max, min, avg and some others in statistic arithmetic.

EDF model enables users to express intuitively queries by query diagram as the way in DF model since it is an enhancement of DF model. According to the definitions in literature (Golfarelli et al. 1998), a fact attribute is called additive if the sum operator can be applied

to aggregate a fact instance along the hierarchy on the fact attribute; non-additive if it is not additive along any dimensions, and semi-additive if it is additive along some dimensions but not others. An example of an additive attribute is quantity sold in the fact schema SALE in literature (Golfarelli et al. 1998). The student qty in Fig. 1 is semi-additive attribute, because it is non-additive along dimension student, but is additive along the course and semester dimensions. The grade attribute is non-additive, because adding up two grades does not make sense. However, aggregation is meaningful on semi-additive or non-additive attributes using operators besides sum, such as avg, max and min. For instance, grade might be averaged along the semester, course or student dimensions.

A query model on a fact schema is a diagram in which fact instances are aggregated by the marked dimension attributes. The mark may include a parameter indicating the filter (the Where clause in SQL) or simply aggregation (group by in SQL). Each hierarchy has at most one marked dimension attribute. A dimension may have no marked dimension attributes, indicating that none of its attributes are involved in the query. The query of the number of male students for each course per year can be represented by the query diagram in Fig. 2, where the dimension attribute sex is marked with the parameter male. Non-dimension attributes need not be shown on the query diagram.

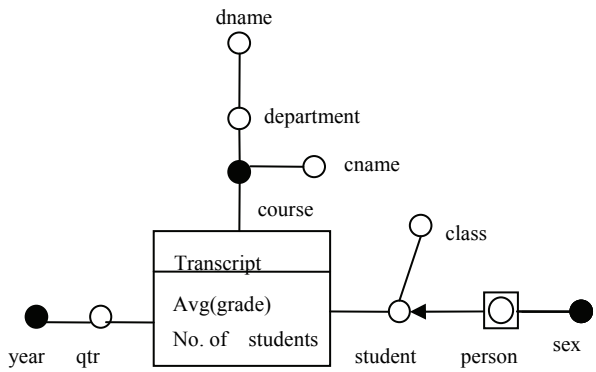

Fig 2：Query Model on a Fact Schema

4. CONCEPTUAL DESIGN FROM EER SCHEMA

The section provides the methodology to build an EDF model starting from the documentation describing the operational object-relational databases. Similar to other conceptual models, it is time-consuming to construct an EDF schema as the schema must map the data sources that are complicated. Because of the fact that the EER schema is a popular

conceptual design in object-relational database, obtaining the EDF schema from the existing EER schema in an automatic way reduces the human involvement on the conceptual design of ORDW, and consequently is time-saving and less error-prone. The transformation from EER to EDF comprises the following steps:

1. Degrading EER and defining facts.
2. Constructing the attribute tree.
3. Pruning and grafting the attribute tree.
4. Transforming the attribute tree to the EDF schema.

These steps are illustrated with reference to the University example, a simplified EER schema depicted in Fig. 3, in which dots denote the identified attributes. The notation of EER model is described by Batini et al. (1992). Every instance of relationship transcript represents a learning

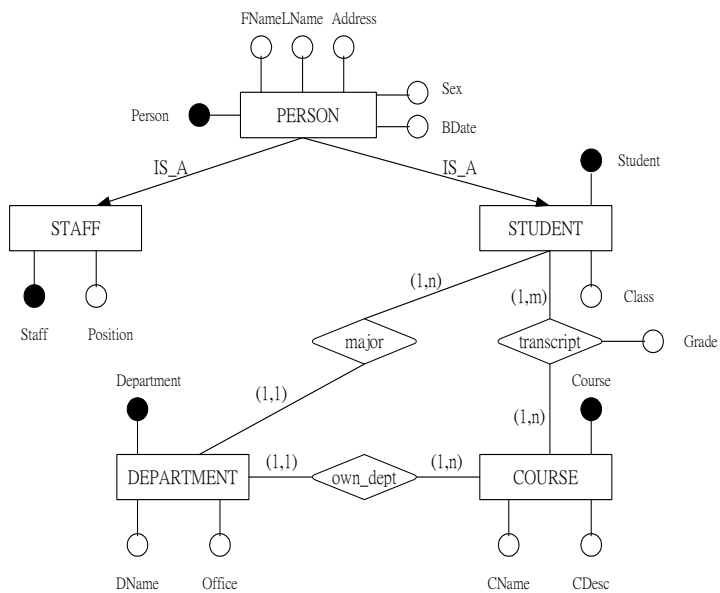

Fig 3：A simplified EER schema

status referring to a single student for a course. A student may take a course more than once, especially when he has failed the course.

4.1 Degrading EER and Defining Facts

Since a fact represents the transaction data increased or updated timely and interest for analysis, it should correspond to either an entity type or a relationship type in an EER schema. Therefore, entity types and relationship types are the candidates for defining facts. Some

tools for conceptual design only allow to-one relationships, and replace all many-to-many the relationships with association entities and to-one relationships (Elmasri and Navathe 2004). Every entity type in such an EER, may be in one of four categories, association, subclass, super-class and general. However, neither entity types in the latter two categories nor relationship types are considered when defining facts.

Accordingly, to simplify the transformation of an EER to an EDF schema, the given EER is first converted into the form with only "to-one" relationships, based on the processes of association entity type (Codd 1979). Each relationship of two categories is transformed into association entities, n-ary relationships and binary many-to-many relationship. Each n-ary relationship is converted into four to-one binary relationships, and each many-to-many relationship is converted into two. Every converted association entity type still contains each of the attributes that exist on the original relationship type. The identifier of the converted entity type representing the participating entity types by composing their identifiers. The relationship transcript in the example in Fig. 3, is many-to-many, and its converted entity type with identifier (Student, Course) is depicted in Fig. 4.

A fact schema based on an EER can be constructed from more than one fact. Every selected fact becomes the root of an individual fact schema. In Fig. 3, the subclass entities, staff and student, and general entities course and department are almost static, and only the relationship transcript represents frequently updated data, thus transcript is defined as the fact after being converted into an association entity type, namely the root of the fact schema. The entity type and relationship type are respectively called "entity" and "relationship" hereafter for short.

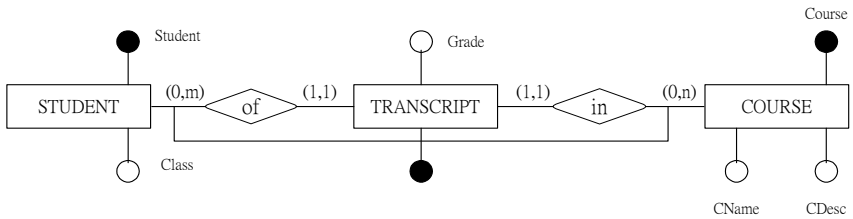

Fig 4：Degrading transcript into an association entity

4.2 Building the Attribute Tree

After selecting facts, the attribute trees are constructed to define their dimensions. An attribute tree *T* has the following characteristics.

- Each vertex of *T* corresponds to an attribute or an identifier of an entity in *S*.
- The root of *T* corresponds to the identifier of the entity selected as fact *F*.
- For *a* vertex *v* corresponding to the identifier of an entity e_1, every child *c* of *v* must

correspond to either the attributes of e_1, or the identifier of the entity which is linked to entity e_1 by a "to-one" relationship in S. The child c in the former case must be a leaf.

The attribute tree for fact F may be constructed automatically by calling the recursive procedure Builder, shown in Table 1, with input parameters fact F and the vertex v_F corresponding to the identifier of F. Namely, The output of procedure Builder is the attribute tree rooted by vertex v_F. Namely, the attribute tree can be built by invoking

Builde(F, v_F);

In procedure Builder, we let v_E denote the vertex corresponding to the identifier of entity E, and v_a the vertex corresponding to an attribute a.

Table 1: Algorithm of using EER model to building the attribute tree

```
1 :  Builder (E, vE) :
2 :      For each attribute a of E and a is not an identifier,
3 :          create vertex va for attribute a, and attach vertex va as the child of vertex v.
4 :          If a is a composite attribute, then
5 :              for each constituent c of attribute a,
6 :                  create vertex vc for c, and attach vc as the child of vertex va.
7 :      For each entity Ei which is connected to E by a is_a relationship
8 :          if Ei is a super-class of E, then
9 :              create vertex vEi for identifier of Ei , and add directed edge from vEi to vertex vE
10 :         else  create vertex vEi for identifie of Ei , and add directed edge from vE to vertex vEi
11 :         call procedure Builder with parameters Ei and vEi .
12 :     For each entity Ei which is connected to E by a x-to-one relationship R,
13 :         create vertex vEi for identifier of Ei , and attach vEi as the child of vertex vE,
14 :         create vertex vb for each attribute b of R, and attach vb as the child of vertex vEi , and
15 :         if b is a composite attribute then for each constituent c of attribute b,
16 :             create vertex vc for c, and attach vc as the child of vertex vb .
17 :         call procedure Builder with parameters Ei and vEi .
```

Procedure builder comprises three steps, as shown in Table 1. In the first step, the first for loop in lines 2–6, each of the attributes (other than the identifier of the current entity) is added to the attribute tree as a vertex and is linked to the input (current) vertex. If the attribute is composite, then every constituent of the attribute is added as a child vertex of the vertex corresponding to the attribute. The second step, in lines 7–11, is that the vertices corresponding for the entities, either super-class or sub-class of entity E, are built and connected by directed edges to present the inheritance relationships. In lines 12–17, this procedure finally deals with every entity E_i linked to the current entity E by the "to-one" relationship R, and all attributes of the "to-one" relationship R. The ends of part two and part three (line 11 and line 17) will call procedure builder recursively to further process every entity.

Briefly, procedure builder forms a vertex for each processed attribute and connects the vertex onto the attribute tree. Given the EER scheme of the Transcript in Fig. 3 as an example, the attribute tree generated by the procedure builder is shown in Fig. 5.

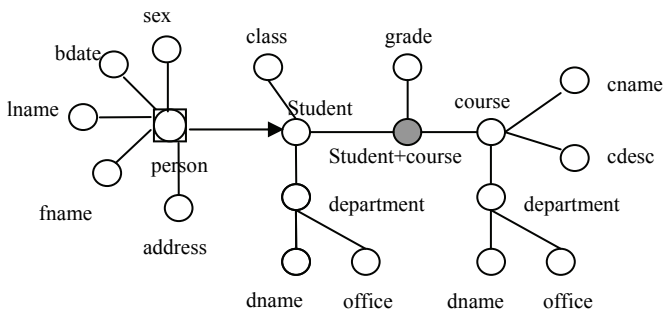

Fig 5：Attribute tree, rooted by transcript, for the EER in Fig. 3

4.3 Pruning and Grafting of the Attribute Tree

The "to-one" relationship could be either one-to-one or many-to-one, and the identifiers of entities connected by the both types of to-one relationships are inserted into the attribute tree. However, the drill-down or roll-up operation, along with the one-to-one relationship, do not change the aggregation result except to change the header name in a DW query. Accordingly, vertices that are not useful in aggregation must be eliminated from the attribute tree. Additionally, the builder may create identical sub-trees, which might be redundant. Therefore, the attribute tree must be modified according to the requirement of the designer. The modification can be performed by grafting and pruning processes described below.

- grafting(u, v): disconnecting all sub-trees rooted by the descendants of vertex u, and connecting each sub-tree to vertex v.
- pruning(s): disconnecting a sub-tree or a vertex s from the parent of s.

As the sub-tree is considered as redundant, it can be pruned directly by invoking pruning function. To eliminate the one-to-one relationship between vertices u and v, where u is the child of v, first call function grafting(u, v), and then call pruning(u) to delete vertex u from the tree. The descendants of vertex u, which are dimensions different from u, remain in the tree. Vertices that are not adopted for aggregation can also be eliminated by grafting and pruning operations. Fig.6 shows the attribute tree modified by pruning the sub-tree department of the vertex student.

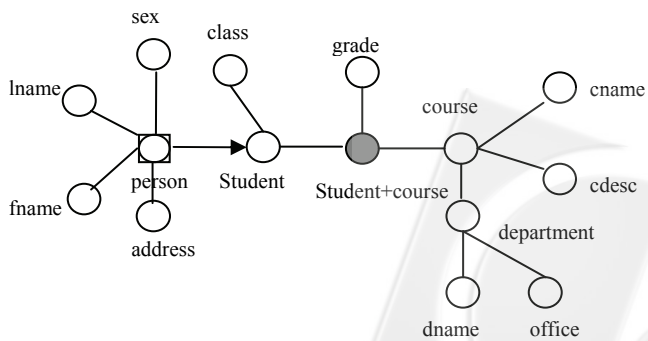

Fig 6：The result of grafting the attribute tree in Fig. 5

4.4 Transforming the Attribute Tree to the EDF Schema

After the attribute tree is modified, the EDF schema can be obtained by transforming the tree with four processes: marking non-dimension attributes, defining fact attributes, defining dimensions and defining hierarchies. These processes require only simple recognition by users or designers. First, the leaf vertices corresponding to the attributes that are adopted to describe the fact, but not to perform aggregation, are examined. Such vertices are marked as non-dimensional attributes using a dot to distinguish them from the others vertices, depicted by a circle. All children of the root that are not marked are then defined as dimensions or fact attributes.

Fact attributes are units of the fact instance to be aggregated in different view. Some of these units are typically displayed in the EER schema as the attributes of the entity types. Thus, the fact attributes can be selected among the leaves that are the children of the root, and added in the box representing the fact in EDF schema. Additionally, a fact may have no attribute if it only records an event (Kimball 2002). In this case, a dummy fact attribute "number" or "amount" needs to be added to the EDF schema.

Each parent–child pair in the tree is a to-one relationship, indicating that the fact instances can be aggregated along with the children of the root (fact). Going thorough a path from a child of the root to a leaf (which is a dimension attribute) represents analyzing fact instances in increasingly coarse units. Restated, the children of the root in the attribute tree could be equivalent to the dimensions of the EDF schema, which determine the method of aggregating fact instances. Accordingly, the dimensions must be chosen among the children of the root in the tree.

A crucial dimension is time. The EER schema normally describes the static relationship among entities for an object-relational database. That is, the database contains only the current data. In such cases, the EER may have no attribute related to time on the entity types or relationship types that are selected as facts. By contrast, data warehouses are mainly employed to store the historical data. Therefore, time should be automatically incorporated into the tree as a dimension if it does not appear as a child of the root. The dimensions chosen in Fig. 6 are course, time and student.

All sub-trees rooted by dimensions are defined as hierarchies. Based on the property of constructing attribute tree, a to-one relationship must be held between each attribute and its child for each hierarchy of an EDF schema. Additionally, the designer can add a new dimension attribute onto a hierarchy under the to-one relationship to aggregate the fact instance in a different granularity. In Fig. 6, the time is variegated with year, quarter and month. The EDF schema shown in Fig. 1 could be one example of transforming the attribute tree of Fig. 5 to an EDF schema.

5. LOGICAL DESIGN FROM EDF SCHEMAS

The EDF schema can express the entities and relationship concept of ORDW, and can thus be a reference for designing the table (or class) schemas of ORDW logically. As the EDF schema is complex, the logical design with respect to schema would be also difficult. This leads to the need for a mechanism that can derive logical design from EDF schema. This section describes a mapping algorithm that can be used to translate EDF schema into the logical design. The process of translation is demonstrated by the example of the transcript. Finally, three checking points are provided to refine the table schema derived from the logical design. The refinement makes the table schema more coincident with the properties and application of ORDW.

5.1 Mapping Algorithm of EDF Schema into Logical Design

The collections of vertices classified according to Definition 1, as fact, dimension attribute and non-dimension, are denoted by F, D and N, respectively. The algorithm maps every node in D or F into a class (table), and every node in N into an attribute of a class. Accordingly, the edge of the EDF schema denotes the relationship either between two classes, or between a class and an attribute. In object-relational data, the relationship between two classes could be either a class composition hierarchy (CCH) (Kimball 2002) or an inheritance (ISA) relationship. Hence, this algorithm establishes the classes mapped from adjacent vertices with CCH or ISA relationships. Additionally, a mapped attribute is included in the transformed class of which vertex connected to the vertex of the converted attribute. The ORDW schema can therefore be constructed automatically by performing the mapping algorithm with the following statements:

C_f = newClass(f);
Translate(C_f, f);

where the fact f is selected arbitrarily from the EDF schema if more than one fact exists, and function Translate is a recursive call with two parameters, the fact f and the class C_f mapped from f, as shown in Table 2.

Table 2 : Translate function: converting an EDF schema into an ORDW schema in logical design

```
Translate(C_v, v) // v is the processing vertex, and C_v is the class mapped from v
{
  for each vertex u connect to v do {
    if u is a leaf node or u is a non-dimension attribute then
        add_attribute(C_v, u);
    else {
 C_u =newClass(u);
```

```
switch (u) {
    case ( u is super class of v) :
        Add_ ISA_relation (Cu, Cv); // establish Cv inheriting Cu
    case ( u is sub-class of v) :
        Add_ISA_relation(Cv, Cu); // establish Cu inheriting Cv
    default ( u is a fact or a non-leaf node) :
        Add_CCH_relation(Cv, Cu); // establish Cu as attribute of Cv
    } } // end switch and end if
    Translate(Cu, u);
  } //end for loop
}
```

The translate function recursively traces all nodes of the EDF model, beginning from the root through a depth-first search. It also works for an EDF schema containing no object property at all. Recall that the time dimension needs to be automatically added into the EDF model with the dimension attributes determined by the designer. Therefore, the translate function should skip the mapping of the dimension attributes of time. Instead, a class for time dimension must establish containing attributes of the nodes on the time hierarchy, or attributes determined by the designer. Additionally, each dimension of the fact in EDF schema can be directly defined as a method belonging to the class of the fact. Given the input as the example in Fig. 1, the mapping algorithm can produce a logical schema shown in Fig. 7, and the output can be displayed directly as a physical design shown in Table 3.

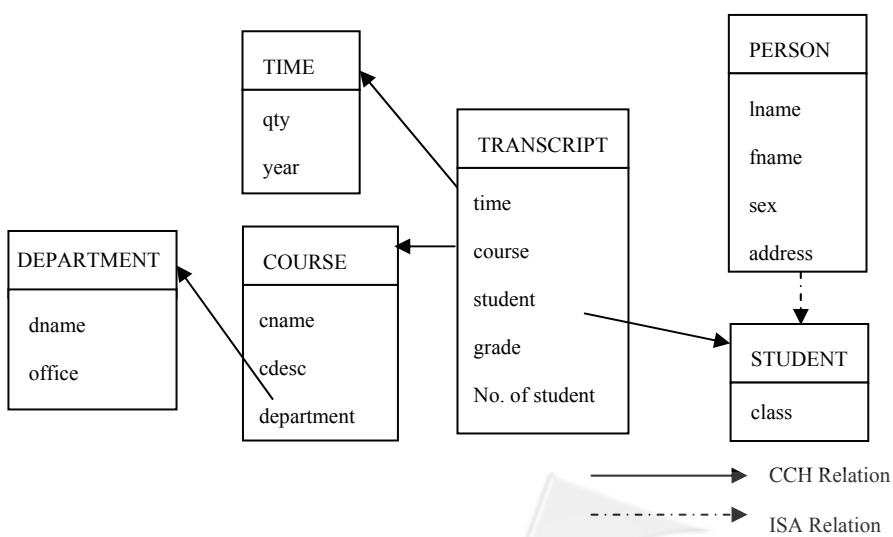

Fig 7：Translated ORWD schema of Transcript in logical design

Table 3: Physical design in Unix-SQL syntax

```
crea te class DEPARTMENT
        (dname string;
       office string);

create class COURSE
        (cname string;
       cdesc string;
       department DEPARTMENT);

create class PERSON
        (lname string;
       fname string;
       sex string;
       address string);

create class STUDENT under PERSON
        (class string);

create class TRANSCRIPT
        (time TIME;
       course COURSE;
       student STUDENT;
       grade integer;
       No. of student integer);
```

5.2 The Refinement of the Logical Design

The variety of data types provided in object-relational systems favors the design on attribute domains, such as SET and CLASS. This section introduces four techniques to refine the translated schema by exploiting the beneficial characteristics of the object-relational model. These techniques are building the set domain attribute, building new class for composite attributes, shortening the length of the CCH relationship and defining methods. They are described with examples below.

（1）Building the Attribute of Set Domain

The attribute value cannot be a set in a relational database or data warehouse. Additionally, the multiple attributes with Boolean values are generally adopted to denote one attribute of the set domain, in which the true value represents the existence of the element. In the object-relational data model, the attribute value may be made non-atomic by using the attribute domain SET, MULTISET or LIST. An example of the resident billing star schema is taken from Golfarelli et al. (1998), in which the dimension table SERVICE is established by the following statement.

```
create class SERVICE {
        call_waiting_flag boolean;
        caller_id_flag boolean;
        voice_mail_flag boolean;
        cellular_flag boolean;
        internet_flag boolean;
        isdn_flag boolean };
```

These attributes can be replaced by a single attribute, as in the statement below,

```
create class SERVICE
        { service_option set varchar(20)
        default {'call_waiting', 'caller_id', 'voice_mail', 'cellular', 'internet', 'isdn'}
};
```

where the attribute is given the default value consisting of all considered values of the set. The query of the ORDW can thus be simplified and made more efficient by using IN to test the existence of elements, instead of examining multiple attributes.

（2）Building a New Class for Composite Attributes

A composite attribute in a relational database is defined in terms of separate attributes. When the domain of a composite attribute occurs frequently for many attributes in a class, the separated attributes may resemble each other, and can therefore be replaced by a CCH to simplify the class in object-relational data.

An example of a software vendor's sales is taken from Golfarelli et al. (1998), where the dimension table CUSTOMER is as follows.

```
class CUSTOMER {
        name string;
        bill_to_city string;
        bill_to_state_other string;
        bill_to_country string;
        ship_to_city string;
        ship_to_state_other string;
        ship_to_country string;
        … };
```

The attributes bill_to_city, bill_to_state_other and bill_to_country resemble the attributes ship_to_city, ship_to_state_other and ship_to_country. A new class comprising attributes city, state_other and country is established with CCH relationships into class CUSTOMER, as given by the statement below,

```
create class PLACE {
        city string;
        state_other string;
        country string; };

create class CUSTOMER {
        name string;
        bill_to PLACE
        ship_to PLACE;
        … };
```

（3）Decreasing the Length of the CCH Relationship

Since the pruning and grafting processes in building the EDF schema are artificial, the path of the translated CCH relationship might be very long because of designer preference. If the predicate of the data query involves the node getting far from the fact in the EDF schema, then the traversal cost of the query rises, downgrading the performance. To avoid this situation, the

designer can graft the sub-tree to shorten the length of CCH based on the trend of data queries. The motivation of this modification is similar to that of de-normalization in databases.

（4）Adding Methods into Classes

In the object-relational data model, the predicate of a data query might include a complicated formula, which raises the difficulty for users in querying. A good ORDW design should define a complex query to be a method within the class. Accordingly, users can directly invoke the method by adding the class name as is the first argument in the formula for the query to simplify and ease the query statement.

6. CONCLUSION

The contribution of this work is providing a semi-automated approach to construct an object-relational data warehouse from object-relational databases efficiently. First, we present an EDF model for conceptual design of ORDW. The EDF model well represents facts, dimensions, super-class/subclass and inheritance, and it also preserves the elegant properties of DF model to enable users to express intuitively queries by the EDF query diagram. Second, we develop a semi-automated methodology for obtaining the EDF schema directly from the EER schema. The semi-automated methodology reduces the manpower on conceptual design of ORDW. Third, we develop an algorithm to translate the EDF schema into a logical design and also its physical codes. Additionally, exploiting the advantage of the domain properties, we provide the methods to refine the translated result to be more efficient on space and data query.

The automated transformation from conceptual design to logical design is significant in software engineering, since it can shorten the life-cycle of ORDW development by reducing the time and the amount of coding errors during the designing period. All proposed methods could be implemented as components or functions within a software design tool to accelerate the construction of ORDW. Our future work will be devoted to developing the methodology for integrating DW and ORDW in the conceptual level.

ACKNOWLEDGMENT

The authors would like to thank NSC for financial support of the research. The useful comments of Dr. James J. Jiang and Dr. Shaw S. Wang are also gratefully acknowledged.

REFERENCES

1. Kimball, R. *The data warehouse ETL toolkit: The Complete Guide to Dimensional Modeling*, John Wiley Sons Inc, 2002.
2. Inmon, W.H. *Building the Data Warehouse*, John Wiley and Sons, New York, 1996, pp:100-110.
3. Gyssens, M. and Lakshmanan, L. "A Foundation for Multidimensional Database," *In Proc. 22nd VLDB Conference*, Mumbai Bombay, India, 1996.
4. Li, C. and Wang, X.S. "A Data Model for Supporting On-Line Analytical Processing," *in Proceedings of the Fifth International Conference on Information and Knowledge Management*, 1996, pp:81-88.
5. Golfarelli, M., Maio, D. and Rizzi, S. "The Dimensional Fact Model: A Conceptual Model for Data Warehouses," *International Journal of Cooperative Information Systems* (7:23) 1998, pp:215-247.
6. Golfarelli, M., Maio, D. and Rizzi, S. "Conceptual Design of Data Warehouses from E/R Schemes," *in Proceedings of 31st Hawaii International Conference on System Sciences* (2) 1998, pp:334-343.
7. Golfarelli, M. and Rizzi, S. "Designing the Data Warehouse: Key Steps and Crucial Issues," *Journal of Computer Science and Information Management* (2:3) 1999, pp:1-14.
8. Sapia, C., Blaschka, M., Hoing, G. and Dinter, B. "Extending the E/R Model for the Multidimensional Paradigm," *in Proceeding of the Int'l Workshop on Data Warehouse and Data Mining*, Singapore 1998, pp:105-116.
9. Tryfona, N., Busborg, F. and Christiansen, J.G.B. "StarER: A Conceptual Model for Data Warehouse Design," *In ACM Second International Workshop on Data Warehousing and OLAP* 1999, pp:3-8.
10. Bækgaard, L. "Event-Entity-Relationship Modeling in Data Warehouse Environments," *in Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP* 1999, pp:9-14.
11. Poe, V., Klauer, P. and Brobst. S. *Building A Data Warehouse for Decision Support*, Prentice Hall PTR, NJ, 1997.
12. Batini, C., Ceri, S. and Navathe, S.B. *Conceptual Database Design: An Entity-Relationship Approach*, AddisonWesley, Spanish, 1992.
13. Fong, J. "Mapping Extended Entity Relationship Model to Object Modeling Technique," *ACM SIGMOD Record* (24:3) 1995, pp:18-22.
14. Gonçalves, S.P.F.C.B.Q. "Easy Does It," *in Proceedings of Q2006 European Conference on Quality in Survey Statistics*, Cardiff, UK, 2006.
15. Nguyen, T., Tjoa, A. and Wagner, R. "An Object Oriented Multidimensional Data Model

for OLAP," *in Proc. of 1st Int. Conf. on Web-Age Information Management*, 2000, pp:69-82.

16. Trujillo, J., Palomar, M. and Gomez, J. "Applying Object-Oriented Conceptual Modeling Techniques to the Design of Multidimensional Databases and OLAP Applications," *In. Proc. of 1st Int. Conf. on Web-Age Information Management* , 2000, pp:83-94.
17. Trujillo, J., Palomar, M., Gomez, J. and Song, I.Y. "Designing Data Warehouses with OO Conceptual Models," *IEEE Computer Society, special issue on Data Warehouses* (34:12) 2001, pp:66-75.
18. Krippendorf, M. and Song, I.Y. "The Translation of Star Schema into Entity Relationship Diagrams," *in Proceeding of the Eighth Int'l Workshop on Database and Expert Systems Applications*, Toulouse, France, 1997, pp:390-395.
19. Moody, L.D. and Kortink, M.A.R. "From Enterprise Models to Dimensional Models: A Methodology for Datawarehouses and Data Mart Design," *Proc. of the Int'l Workshop on Design and Management of Datawarehouses* , Stockholm, Sweden, 2000.
20. Akoka, J., Comyn-Wattiau, I. and Prat, N. "Dimension Hierarchies Design from UML Generalizations and Aggregations," *In International Conference on Conceptual Modeling* (ER2001), Springer Berlin, Heidelberg, 2001, pp:15.
21. Lujan-Mora, S., Trujillo, J. and Song, I.Y. "Extending the UML for Multidimensional Modeling," *Fifth Int. Conf. on the Unified Modeling. Language and its applications (UML '02) Dresden, Germany, Lecture Note in Computer Science*, 2002, pp:290-304.
22. Lujan-Mora, S., Trujillo, J. and Song, I.Y. "Multidimensional Modeling with UML Package Diagrams," *in Proc. of the 21st Int. Conf. on. Conceptual Modeling ER '02*, 2002.
23. Abelló, A., Samos, J. and Saltor, F. "Understanding Facts in a Multidimensional Object-Oriented Model," *4th Int. Workshop on Data Warehousing and OLAP* (DOLAP '01), 2001.
24. Rahayu, W., Dillon, T.S., Mohammed, S. and Taniar, D. "Object-Relational Star Schemas," *13th IASTED Int. Conf. on Paraellel. Disibuted Computing and Systems*, LA, USA, 2001.
25. Agrawal, R., Gupta, A. and Sarawagi, S. "Modeling Multi-dimensional Databases," *In Proc. of the 13th Int'l Conference on Data Engineering*, Birmingham, UK, April 1997.
26. Cabibbo, L. and Torlone, R. "A Logical Approach to Multidimensional Databases," *In Sixth Int. Conference on Extending Database Technology* (EDBT' 98), 1998, pp:183–197.
27. Hacid, M. and Sattler, U. "An Object-centered Multi-dimensional Data Model with Hierarchically Structured Dimensions," *in Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop*, 1997.
28. Vassiliadis, P. "Modeling Multidimensional Databases, Cubes and Cube Operations," *in Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, 1998.

29. Franconi, E. and Sattler, U. "A Data Warehouse Conceptual. Data Model for Multidimensional Aggregation," *In Proc. of the. Workshop on Design and Management of Data Warehouses*, 1999.
30. Pedersen, T.B., Jensen, C.S. and Dyreson, C.E. "A Foundation for Capturing and Querying Complex Multidimensional Data," *Information Systems* (26:5) 2001, pp:383–423.
31. Kimball, R. *The Data Warehouse ETL toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*, New York: Wiley Computer Pub, 1998, pp:125-130.
32. Franconi, E. and Kamble, A. "The GMD Data Model for Multidimensional Information: A Brief Introduction," *in Proceedings Conference on Data Warehousing and Knowledge Discovery*, 2003, pp:55-65.
33. Thomas, H. and Datta, A. "A Conceptual Model and Algebra for On-line Analytical in Processing of Decision Support Databases," *Information Systems Res.* (12:1) 2001, pp:83-102.
34. Elmasri, R. and Navathe, S.B. *Fundamentals of Database System*, Addison Wesley, 2004.
35. Vassiliadis, P. and Sellis, T. "A Survey of Logical Models for OLAP Databases," *SIGMOD* (28:4) 1999, pp:64-69.
36. Tsois, A., Karayannidis, N. and Sellis, T. "MAC: Conceptual Data Modeling for OLAP," *in Proc. of the International Workshop on DMDW*, 2001.
37. Abello, A., Samos, J. and Saltor, F. "A Framework for the Classification and Description of Multidimensional Data Models," *in Proc. of DEXA '01*, Munich, Germany, September, 2001, pp:668-677.
38. Feng, L. and Dillon, T.S. "Using Fuzzy Linguistic Representations to Provide Explanatory Semantics for Data Warehouses," *IEEE Trans. on KDE* (15:1) 2003, pp:86-102.
39. Codd, E.F. "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. Database System* (4:4) 1979, pp:397- 434.
40. Gray, P. and Watson, H. J. *Decision Support in the Data Warehouse*, Prentice-Hall, New Jersey, 1998.
41. Gopalkrishnan, V., Li, Q. and Karlapalem, K. "Star/snow-flake Schema Driven Object Relational Data Warehouse - Design and Query Processing Strategies," *in Proceedings of the First International Conference on Data Warehousing and Knowledge*, 1999.