Logical Database Design with Ontologically Clear Entity Relationship Models

Dhammika Pieris* and Jayantha Rajapakse+
Email: *dhamika.pieris@monash.edu,
+jayantha.rajapakse@monash.edu
School of Information Technology, Level 4, Building 2
Monash University, Sunway Campus
Jalan Lagoon Selatan, 46150 Bandar Sunway
SELANGOR DARUL EHSAN, MALAYSIA

***Abstract*— It is argued that ontologically clear entity relationship models can model the real world domains more accurately than ontologically unclear models. However, transformation of such models into the relational model at the logical level has not yet been studied sufficiently with a view to formulate new transformation rules. This paper presents a set of new transformation rules to convert ontologically clear conceptual models to relational models. Finally we did a comparison of two relational models that were developed from the ontologically clear and unclear models using a quality criterion synthesized from the extant literature. The preliminary results of this ongoing research study shows that the quality of relational model developed from ontologically clear conceptual model is superior to its ontologically unclear counterpart.**

I. INTRODUCTION

Conceptual modeling is an activity undertaken during the early stages of information systems development work[3], where a graphical diagram representing the real word phenomena of an application domain is produced. Researchers have enhanced the expressive power of a well-known conceptual modeling methodology, the entity relationship (ER) model [4], using an ontology [7] that can describe the structure and the behavior of the real world. These models are called ontologically clear entity relationship diagrams (OC-ERD) [8] .

In order to take the advantage of OC-ERDs, such models should be properly transformed into a relational database schema (RDS) [9, 10] without losing the semantics of the OC-ERD. The current set of rules developed to transform generic ERDs is not fully applicable for OC-ERDs. As such, this paper presents some preliminary results of an ongoing research study undertaken to develop new transformation rules and a method of evaluating the quality of the relational model derived from such rules.

The rest of the paper is organized as follows. Accordingly, the section II demonstrates an ontologically unclear ERD (OUC-ERD) of a particular real world scenario and its transformation together with issues of transformation. Section III presents the ontologically clear version of the OUC-ERD, the OC-ERD, and issues of transforming it using the existing algorithm. The section also presents a new algorithm proposed and the transformation using it. Then in section IV, we propose a quality criteria for assessing the quality of the two types of RDSs resulted from both approaches and present the quality comparison. Finally we discuss the preliminary results and the future work of this ongoing research study in the section V.

II. LOGICAL DATABASE DESIGN ISSUES WITH OUC-ERDS

We now present an OUC-ERD and issues of transforming it to the relational model using the existing ER to relational transformation algorithm.

Fig 1 is an OUC-ERD representing a company in the real world. The diagram contains a binary 1:1(one-to-one) and optional relationship type “Manages” between two entity types “Employee” and “Department”. The cardinality ratio (0, 1) between the Employee entity type and the relationship type has two meanings i.e. an employee may not manage a department or an employee who manages a department can manage only one department. The “StartDate” attribute represents the date an employee starts managing a particular department.

Optional relationships like the one above are believed to be difficult to understand by typical domain users. Ontology proscribes the use of optional relationship types and advises using mandatory relationship types with subtyping [8]. Accordingly, the diagram depicted in Fig 1 is an ontologically unclear ER diagram (OUC-ERD).

The existing transformation algorithm proposed by Elmasri and Navathe [9] is given below.

For each regular entity type E in the ER schema, create a relation L that includes all the attributes of E. Choose one of the key attributes of E as the primary key of L.

For each binary 1:1 relationship type in the ER schema, identify the relations S and T that correspond to the entity types that participating in R. Choose one of the relations – S, say- and include as a foreign key in S the primary key of T. Include all the simple attributes of R as attributes of S.

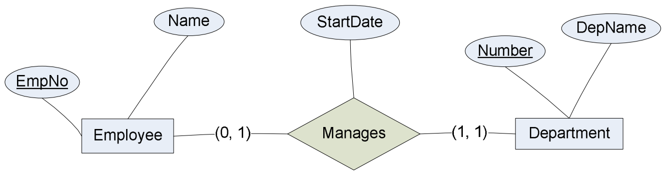

Fig 1.Ontologically unclear ERD (OUC-ERD)

Fig 2 depicts the RDS obtained by transforming the OUC-ERD using the above algorithm.

Employee | EmpNo | Name

Department | DepNo | DepName | EmpNo | StartDate

Fig 2. The RDS obtained by transforming the OUC-ERD in Fig 1. using the existing ERD to relational model transformation algorithm

Since the RDS in Fig 2 has been obtained by exactly following the transformation rules, we call it *intra-algorithm transformation*.

However, there are some ambiguities prevailed in the RDS. The reasons for having the primary key (PK), EmpNo of Employee as a foreign key (FK) in the Department relation are, firstly to represent the "Manages" relationship of the OUC-ERD, and secondly to refer to the Employee who is supposed to manage the Department. However, in general, it might be difficult for a person other than the designer of the RDS to understand these reasons.

Similarly, the meaning of the inclusion of the "StartDate" attribute in the Department relation could not be understood. The word "Manages" in the ERD has been omitted during the transformation and hence it is not being included in the RDS. One might have to refer back to the ERD and sometimes even to the domain scenario to find the reason. The solution used in the current practice to solve these ambiguities is to rename the attributes using suitable prefixes. Accordingly, EmpNo and StartDate in Department are changed to Mgr_EmpNo and Mgr_StartDate respectively. The prefix "Mgr" represents the word "Manager". The modified RDS is given in Fig 3.

However, we are of the view that these adjustment are made outside the rules. Any of the transformation rules does not address re-naming attributes, the method of re-naming and how to decide the prefixes for re-naming. Since these adjustments are made outside the legitimate transformation algorithm, we call it *extra-algorithm adjustments*. This extra algorithm process uses the designer's domain expertise, intuition, judgement and the personal opinion.

Employee | EmpNo | Name

Department | DepNo | DepName | Mgr_EmpNo | Mgr_StartDate

Fig 3. Extra algorithm adjustments made to the transformed RDS in Fig 1. The attributes "EmpNo" and "StartDate" have been re-named to Mgr_EmpNo" and "Mgr_ StartDate". The abbreviated prefix "Mgr" represents the word "Manager"

The extra algorithm adjustment creates several issues as follows.

i Re-naming attributes outside the algorithm,

ii. How a particular word can be decided to use to make a prefix for re-naming, for example "Manager" in this case,

iii. How the prefix, for example "Mgr" is made.

Even after the extra-algorithm adjustments, some semantics are still seemed to be ambiguous. For example, the OUC-ERD indicates that an employee can manage only zero or one department, and it has to be reflected in the RDS in Fig 3. However, it is not clear whether a manager employee, denoted by the re-named attribute "Mgr_EmpNo" can manage only one Department or more Departments, because the same Mgr_EmpNo can be repeated in different tuples. This shows that some semantic information has been lost during the transformation and has not been re-established even with the extra-algorithm adjustment process.

Assume a designer not familiar with the domain is given such an OUC-ERD to design a RDS. Presumably, the designer might perform the intra-algorithm transformation and come up with a RDS depicted in Fig 2. However, he/she might sometimes mess up with the entire design when it comes to perform the extra- algorithm adjustments due to lack of domain knowledge.

III. The OC-ERD and its transformation

In the OC-ERD depicted in Fig 4, the optional relationship has been resolved to a mandatory relationship between the Department and a newly created subtype "Manager" of the Employee; the Employee has become a supertype.

Cardinality ratios have now become very clear, and which indicates each "Manager" definitely "Manages" a "Department" and "Manages" one and only one "Department" at any given time.

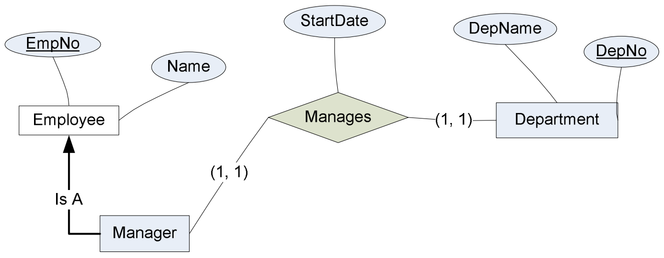

Fig 4. Ontologically clear ERD (OC-ERD)

The transformation of the OC-ERD using existing rules provides three separate optional RDSs as given in Fig 5. Various types of ambiguities are present in each of the RDS, for example, in Fig 5(a), information redundancies, inability to identify the subtype/supertype relationship and the exact entity that participate in the FK relationship, and inability to identify the purpose of the FK relationship, etc. Fig 5 (b) and (c) also contains some ambiguities.

Employee | EmpNo | Name

Manager | EmpNo

Department | DepNo | DepName | EmpNo | StartDate

(a)

Employee | EmpNo | Name | mgr

Department | DepNo | DepName | EmpNo | StartDate

(b)

Employee | EmpNo | Name

Manager | EmpNo | DepNo | StartDate

Department | DepNo | DepName

(c)

Fig 5 Transformation of the OC-ERD in Fig 4 using existing rules, (a) – optional RDS – 1, (b) –optional RDS –2, mgr is a Boolean type attribute that indicates the subtype to which the tuple belongs, (c) –optional RDS - 3

The above simple example shows that the existing transformation rules do not work properly for both OUC-ERDs and OC-ERDs. Thus, it is not possible to obtain an unambiguous, straightforward and trust worthy RDS which preserves all the information represented in either of the ERDs. Hence, it is essential to develop new or modified set of rules to accommodate OC-ERDs as well as OUC-ERDs.

It is observed that OC-ERDs provide relatively greater support to remedy the ambiguities in the transformation process. For example, the issue of resolving the prefix "Mgr" to re-name attributes. In the case of the OUC-ERD, it is completely based on the designer's expertise and opinion. However, in the case of OC-ERD, the ERD itself indicates the word "Manager" as a subtype participating in the concerned relationship "Manages", so that the word can be abbreviated to form the prefix or the complete word can be used as the prefix if desired. This situation motivated us to introduce a logical procedure to the re-naming process which links the OUC-ERD to the final RDS.

The proposed new algorithm that covers OC-ERDs consists of following aspects.

1. A naming convention for the attributes, relationships and entity types in the OC-ERD.
2. A new set of transformation rules to transform OC-ERDs to the relational model.
3. Rules for re-naming names of attributes in the relational database schema.

The proposed algorithm that has seven steps which are either new or adapted from [9] is presented below.

Step 1: Mapping regular entity types

1. For each regular entity type E in the ERD, create a relation L that includes all the attributes of E.
2. Choose one of the key attributes of E as the primary key of L, i.e. PK(L).
3. Place the PK as the first attribute of the new relation L proceeded by the remaining attributes of the entity E.

Step 2: Mapping of super type/sub type segments

Convert each super type/sub type segment with m number of subtypes { S_1, S_2, …, S_m} and a super type C into relation schemas using one of the following options:

Option 1:

1. Create a relation L for C with attributes Attrs(L) = {k, a_1, a_2, .., a_n} and the primary key, PK(L) = k.
2. Create a separate relation L_i for each sub class S_i, $1 \leq i \leq m$, with the attributes Attrs(L_i) = {k}∪ {attributes of S_i } and PK(L_i) = k.

Option 2:

1. Create a single relation L with attributes Attrs(L) = {k, a_1, a_2, …, a_n} ∪ {t_1}{attributes of S_1} ∪... ∪ {t_m}{attributes of S_m} and PK(L) = k. Each t_i, $1 \leq i \leq m$, is a Boolean type attribute indicating whether a tuple belongs to subclass S_i.

2. In L, the attributes of a particular subtype S_i should be included following its corresponding Boolean type attribute t_i.

Step 3: Naming of the transformed relations

Case 1: Super type name C is already included in the subtype name S as its last word as follows

S = XC where X is a new word used to form a part of the subtype name.

In this case the underscore must be placed in between X and C to form X_C, so that L_s = X_C.

Case 2: When the super type name is not included in the subtype name, the full subtype name should be joined to the supertype name by an underscore to form S_C, where S is the subtype name and C is the super type name.

Step 4: Naming a Boolean type attribute used to transform a subtype

The name of the Boolean type attribute, say t, should be made by joining the name of the subtype, say S, by a dash together with the text string "Flg" that represents the word Flag", so that t = S-Flg.

Step 5: Mapping a binary 1:1 relationship type R in between a regular entity type and a subtype of a supertype

Assume that a sub type S of a super type C has a binary 1: 1 relationship type with a regular entity type E.

Case 1: The subtype is transformed to a separate relation.
In order to transform the relationship type R identify the separate relations L_s and L_e that correspond to the entity type S (the sub type) and E (regular) that participate in R.

Option 1

1. Include as a foreign key in L_e the primary key of L_s, PK(L_s) and re-name it as S_PK(L_s).
2. Add a suffix (U) to be appeared as S_PK(L)(U). The letter "U" indicates that the foreign key is unique.
3. Include any simple attribute, say A, of the relationship type R in L_e following the foreign key included and re-name it as S_A.
4. If the subtype S in the ER schema does not include its own attributes the separate relation, L_s that represents the subtype S should be ignored for this option.

Option 2

1. Include as a foreign key in L_s the primary key of L_e, PK(L_e) together with a suffix U to be appeared as PK(L_e) U. The letter "U" indicates that the foreign key is unique.
2. Include all the attributes of the relationship type R in L following the foreign key.

Case 2: The subtype is transformed using a Boolean type attribute to a single relation together with its super type.

For each relationship type R identify the single relation L that correspond to the S/C subtype/supertype segment and the relation L_e of the regular entity type E where S and E participate in R.

1. Include as foreign key in L_e the primary key of L, PK(L) and rename it as S_PK(L).
2. Add a suffix (U) to be appeared as S_PK(L)(U). The letter "U" indicates that the foreign key is unique.
3. Include all the simple attributes of R as attributes of L_e following the foreign key included.
4. Include any simple attribute, say A, of the relationship type R in L_e following the foreign key included and re-name it as S_A.
5. If the subtype S in the ER schema does not include its own attributes the Boolean type attribute included in the relation L should be removed for this option

Step 6: Mapping a binary 1:1 relationship type R between two different subtypes of different supertypes

Assume that A and B to be two different subtypes of two different super types of a ER schema and assume that R is a binary 1:1 relationship type that exists in between A and B.

1. Transform one of the subtypes, say A, to a separate relation L_A.
2. Assuming that L_B to be the separate relation corresponding to the subtype B, include in L_A as a foreign key the primary key of L_B, PK(L_B), and rename it as B_PK(L_B). This foreign key should be included following the last existing attribute of L_A.
3. Include all the simple attributes of R in L_A following the foreign key.

4. If the subtype B contains its own attributes, it should be transformed together with its supertype using either of the following options appropriate.
   a) Two separate relations for each of the subtype B and its supertype (Step 2 Option 1 above).
   b) A single relation for both the subtype B and its supertype (Step 2 Option 2 above).
5. Else if the subtype B does not contain its own attributes its super type should only be transformed.

Step 7: Mapping a binary 1:N relationship type between two different subtypes of different supertypes

Assume that A and B are two different subtypes of two different supertypes of the ER schema where R is a binary 1:N relationship type existed in between the two subtypes and assume that A is at the 1 side and B is at the N side of the relationship, so that an instance of A associate with many instances of B.

1. Transform the subtype B which is at the N side to a separate relation L_B.
2. Assuming that L_A to be the relation corresponding to the subtype A, Include as a foreign key the primary key of L_A and rename it as A_PK(L_A). This foreign key should be included following the last existing attribute in L_B.
3. Include in L_B all the simple attributes of the relationship type R following the included foreign key.
4. If the subtype A contains its own attributes, A and its supertype should be transformed using either of the following optional methods
   a) Two separate relations for each of the subtype A and its supertype (Step 2 Option 1 above).
   b) A single relation for both the subtype A and its supertype (Step 2 Option 2 above).
5. Else if the subtype A does not contain its own attributes its super type should only be transformed to a separate relation.

Fig 6 (a) and (b) depict the results of the transformation of the OC-ERD given in Fig 4 using the new algorithm presented above. Accordingly, the optional RDS (b) is a result of the Case 1 option 1.1; optional RDS (a) is a result of the Case 1 option 1.2. The result of Case 2 is same as the RDS (b). Thus only two optional RDSs have been produced though there are three separate optional methods to transform the OC-ERD.

Employee | EmpNo | Name

Manager_Employee | EmpNo | DepNo (U) | StartDate

Department | DepNo | DepName

(a)

Employee | EmpNo | Name

Department | DepNo | DepName | Manager_EmpNo (U) | Manager_StartDate

(b)

Fig 6. Transformation of the OC-ERD in Fig 4 using new algorithm, (a) Optional OC-RDS - 1 (b) Optional OC-RDS - 2

Even in this transformation some attributes needed to be re-named, for example, EmpNo is re-named to Manager_EmpNo. The difference between this re-naming process and the re-naming process with regard to the OUC-ERD is that in this case it is handled by the transformation process itself without leaving for the designer intuition. There are no extra-algorithm adjustments similar to the one observed with the OUC-ERD transformation. This systematic relationship has been prior established as a general rule and given in the new algorithm in the Step 5, Case 1, option 1.1 sub step ii and iii. The re-named FK attribute has been given a suffix U indicating that the foreign key is unique preventing any unexpected repetition of the FK values in different tuples as was happened in Fig 3. Accordingly, all the ambiguities experienced in the transformation of the OUC-ERD have been resolved by a specific rule.

IV. QUALITY OF THE DATABASE SCHEMA DESIGNED USING NEW ALGORITHM

Assessing the quality of RDSs resulting from OC-ERDs via the new transformation algorithm is important. More importantly, the relative quality of RDSs produced through ontologically unclear model and the ontologically clear models need to be assessed. The success of the new algorithm can then only be claimed.

Several previous studies [1, 2, 5] have proposed quality dimensions and quality measuring schemes for data and database schemas.

Table 1 shows a set of quality dimensions synthesised form the extant literature. We believe that a preliminary quality assessment can be done using these dimensions.

TABLE I
QUALITY DIMENSIONS THAT A RDS SHOULD SATISFY

No	Quality Dimension (QD)	Description
1	Accuracy	A RDS should accurately represent information modeled in the ERD.
2	Complete	The RDS should preserve all the information represented in its predecessor ERD without losing anything(adapted from[1, 2])
3	Straightforward	The entire RDS including any tiny part of it must be a one logically and directly derived from its predecessor ERD according to a specific procedure
4	Trustworthy	The RDS should reflect only the information represented in its predecessor ERD and nothing outside it (adapted from [2])
5	Clear	The meaning intended by each item in the schema must be clear
6	Minimality	The RDS must be free from all the redundancies. Each aspect of the ERD must appear only ones in the RDS (adapted from [1, 5, 6])
7	Normalization	The RDS must satisfy the well-known normal forms as much as possible (adapted from[2])
8	Expressiveness	The RDS should be easily understood through its constructs(adapted from [1])
9	Reversible	The respective ERD could be able to re-produced only from the information presented in the RDS following a specific logic.

As a preliminary quality test we compared the RDS in Fig 3 together with the two optional RDSs in Fig 6 using the seventh quality dimension namely, normalization in the above Table I. We observed that all the relations in RDSs of Fig 6 were in the third normal form. In the meantime, all the relations in Figures 3 are in the first and the second normal forms. Thus, it can be concluded that the relational database schemas produced from the ontologically clear ERDs following the new algorithm shows higher quality in terms of the normalization level compared to its ontologically unclear counterparts.

V. DISCUSSION AND FUTURE WORK

Using a simple real life example we have shown that ontologically clear ER diagrams can produce high quality relational models compared to its counterpart i.e. ontologically unclear. This has been achieved by way of modifying the existing transformation rules. However, to obtain the optimal results this transformation algorithm has to be improved further. As a preliminary study only a part of a quality criterion was used for assessing the quality of the relational models produced.

Future work of this ongoing research study includes the following.

- To develop new rules to cover all the ontological constructs
- To modify the existing rules to overcome current deficiencies with regard to the ontologically unclear diagrams
- To extend the quality assessment to cover the rest of the quality criterion
- To improve the quality criterion in the meantime

It is expected that this study will have an impact on the popularity and the use of the ontologically clear conceptual ER models in the industry.

REFERENCES

[1] C. Batini, S. Ceri, and S. B. Navathe, *Conceptual database design: an Entity-relationship approach*: Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA ©1992, 1992.

[2] R. Y. Wang, M. P. Reddy, and H. B. Kon, "Toward quality data: An attribute-based approach," *Decision Support Systems,* vol. 13, pp. 349-372, 1995.

[3] R. Weber, "Research review paper conceptual modeling and ontology: posibilities and pitfalls," *Journal of Database Management,* vol. 14, pp. 1-20, 2003.

[4] P. P.-S. Chen, "The entity-relationship model: toward a unified view of data," *ACM Trans. Database Syst.,* vol. 1, pp. 9-36, 1976.

[5] C. Fahrner and G. Vossen, "A survey of database design transformations based on the Entity-Relationship model," *Data & Knowledge Engineering,* vol. 15, pp. 213-250, 1995.

[6] Y. Wand and R. Weber, "On the ontological expressiveness of information systems analysis and design grammars," *Information Systems Journal,* vol. 3, pp. 217-237, 1993.

[7] M. Bunge, *Treatise on basic phylosophy: Vol3: ontology I: The furniture of the world.*: D. Reidel Publishing Co., Inc., New York, NY., 1977.

[8] Y. Wand, V. C. Storey, and R. Weber, "An ontological analysis of the relationship constuct in conceptual modeling," *ACM Transactions on Database Systems,* vol. 24, pp. 494-528, December 1999.

[9] R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems," in *Fundamentals of Database Systems*, 5th Edition ed New York: Addison Wesley, 2007, pp. 223-239

[10] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM,* vol. 13, pp. 377-387, 1970.