THE NATIONAL UNIVERSITY
of SINGAPORE

School *of* Computing
Lower Kent Ridge Road, Singapore 119260

TRE6/03

Resolving Schematic Discrepancy in the Integration of ER Schemas

Qi HE and Tok Wang LING

June 2003

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Resolving Schematic Discrepancy in the Integration of ER Schemas

Qi He, Tok Wang Ling

Dept. of Computer Science, School of Computing
National University of Singapore
{heqi,lingtw}@comp.nus.edu.sg

Abstract. In this work, we address the schematic discrepancy problem in the integration of ER schemas. Schematic discrepancy occurs when same information is modeled as attribute values, object type (entity type or relationship set) names, or attribute names in different ER schemas of databases. We first define the concepts of local/global attributes and identifiers of entity types and relationship sets resp in an ER schema, which provide essential semantics to resolve schematic discrepancy. Then based on those concepts, we give algorithms to resolve schematic discrepancy among ER schemas to be integrated.

1 Introduction

Database integration includes schema integration and data integration. Schema integration involves merging several schemas into one integrated schema. More precisely, [Bati86] defines schema integration as "the activity of integrating the schemas of existing or proposed databases into a global, unified schema". With the current research into heterogeneous databases, this process plays an important role in integrating export schemas into a global schema. [Ling96] proposes an Entity-Relationship (ER) based federated database system ([Shet90]) where local schemas modeled in the relational, network or hierarchical models are first translated into the corresponding ER export schemas before they are integrated. In the integration of ER export schemas into a global schema, people have identified many conflicts need to be resolved:

(1) Naming conflict – homonyms and synonyms are the two sources of naming conflicts. Renaming is a frequently chosen solution in existing works.
(2) Structural conflict [Bati84, Lee95] – Same real world concept may be represented in two schemas using different modeling constructs. For example, a same concept is modeled as an entity type in one schema, but an attribute in another schema.
(3) Constraint conflict [Lee97, Redd95] – include key conflict, domain mismatch, and cardinality conflict (e.g., phone number is a single valued attribute in one schema, but multivalued in another schema).
(4) Contextual conflict [He03] – equivalent entity types or relationship sets model a same concept in different contexts of databases.

In this paper, we discuss one kind of conflicts need to be resolved in the integration of ER schemas, which hasn't been noticed much before, i.e. *schematic discrepancy*. Schematic discrepancy occurs when same information is modeled as object type (entity type or relationship set) names, attribute names, or attribute values in different ER schemas of databases. In the following, we first see an example of schematic discrepancy among ER schemas. For easy to understand, we also give the corresponding relational schemas of those ER schemas.

Ex. 1.1: Suppose sales companies keep supply information of products (i.e. product number, product name, quantity of product) of each month. Suppose a constraint says "*in a same month, the supplying quantity of a same product is unique*". They may design one of the following relational schemas to keep the data:

- *DB1*: *Sup(p#,pname,month,qty)*. In this schema, months are modeled as values of attribute *month*. The following FDs hold in the relation of *DB1*: *p#* → *pname*; *p#,month* → *qty*.
- *DB2*: *Jan_prod(p#,pname,qty)*,…, *Dec_prod(p#,pname,qty)*. In these schemas, months are modeled as relation names. That is, each relation keeps the supply information of one month. The following FD holds in each relation of *DB2*: *p#* → *pname,qty*.
- *DB3*: *Prod(p#,pname,jan_qty,…,dec_qty)*. In this schema, months are modeled as attribute names. The attributes *jan_qty,…,dec_qty* represent the supplying quantity of products in different months resp. The following FD holds in the relation of *DB3*: *p#* → *pname,jan_qty,…,dec_qty*.

In a federated database system, to integrate databases from different sources, they first translate the local schemas (relational model) to export schemas (ER model), and then integrate the ER schemas. As an example, the above relational schemas can be translated to ER schemas as Fig. 1.1, in which months are modeled as attribute values, entity type names and attribute names resp. But the ER schemas in Fig. 1.1 can not be merged directly because of the schematic discrepancy problem on months. To solve the problem, we will first transform these schemas to a unified form, and then merge them (Section 5). □

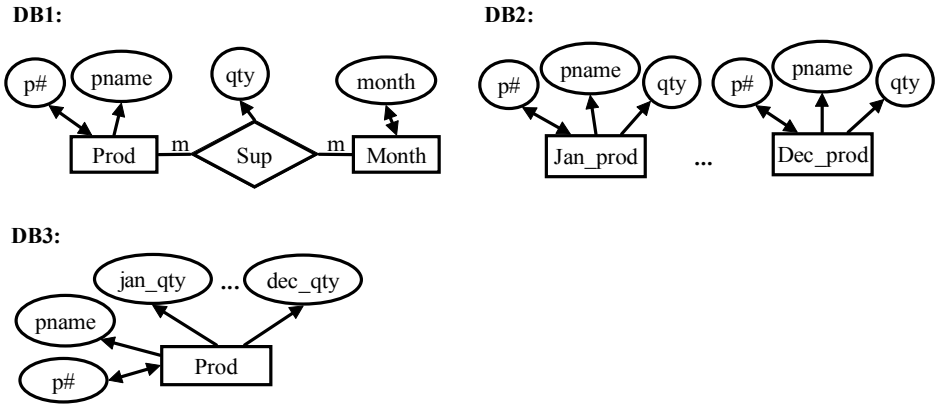

Fig. 1.1: Schematic discrepancy in entity types: months are modeled as attribute values, entity type names or attribute names in different databases

The rest of the paper is organized as follows. In Section 2, we survey some related works on schematic discrepancy. Section 3 is an introduction to the ER approach. Section 4 and 5 is the main contribution of this paper. In Section 4, we define some concepts (i.e., local/global attributes and identifiers) which provide essential semantics to resolve schematic discrepancy among ER schemas. And in Section 5, we give algorithms to resolve schematic discrepancy using the concepts defined in Section 4. At last, Section 6 concludes the whole paper.

2 Related works

[Kris91, Laks99, Laks01] discussed the schematic discrepancy problem in the context of relational model. They proposed languages to query information from multiple relational databases whose schemas are schematically discrepant. While in a federated database system,

component databases are not necessary in relational model. It is required that the federated system can support local schemas expressed in different data models. To facilitate the design, integration and maintenance of the federated system, however all component, export and federated schemas should be in the same data model. This data model is called canonical or common data model. As to [Shet90, Ling96], the relational model does not possess the necessary semantics for defining all the integration mappings that might be desired. Furthermore, relational model does not possess semantics to represent integrity constraints (e.g., functional dependencies, multi-valued dependencies, etc). Recently, people have opted to use semantic data models such as ER model [Chen76] to be the common data model in a federated database system [Ling96]. We'll learn more about ER model in Sec 3.

[He03] resolved contextual conflicts in the integration of ER schemas. Contextual conflicts occur when equivalent entity types or relationship sets model a same real world concept within different contexts of databases, which will cause problems in schema integration and data integration. During the integration of ER schemas, the resolution of contextual conflicts follows the resolution of schematic discrepancy and other conflicts among ER schemas. We can treat contextual conflict as schematic discrepancy problem on a multidatabase level. However, the resolution of schematic discrepancy is more complex than that of contextual conflicts, as the former need to deal with different kinds of discrepancies.

3 ER Approach

Generally, the integration of heterogeneous databases requires a semantically-rich canonical model to fully capture the semantics of data to be integrated. ER model [Chen76] is such a data model. The schema objects of ER model, entity type and relationship set, correspond to the real world concepts well. This facilitates the decision of schema equivalence, as well as the use of domain knowledge in integration. Furthermore, ER model supports functional dependencies, multi-valued dependencies and cardinality constraints among entity types, relationship sets and attributes. As mentioned, the conflicts of these constraints need to be resolved in schema integration.

The ER model incorporates the concepts of entity type and relationship set. An entity type or relationship set has attributes which represent its structural properties. An attribute can be single-valued, i.e., 1:1 (one-to-one) or m:1 (many-to-one); multivalued, i.e., 1:m (one-to-many) or m:m (many-to-many); or composite. A minimal set of attributes K of an entity type E which defines a one-to-one mapping from E into the Cartesian product of the associated value sets of K is called a key of E. An entity type may have more than one key and we designate one of them as the primary key or the identifier of the entity type. A minimal set of identifiers of some entity types participating in a relationship set R which uniquely identifies R is called a key of R. A relationship set may have more than one key and we designate one of them as the identifier of the relationship set.

4 Definitions

To resolve schematic discrepancy in the integration of ER schemas, we propose to first transform discrepant schemas to a unified form (in which all the interesting information are modeled as attribute values; the precise definition will be given in Definition 4), then merge the transformed schemas. We will give the transformation algorithm in Sec.5. In this section, we define some concepts providing essential semantics which will be used in the transformation process.

Definition 1 (*Local/global attribute of entity type*): Given an ER schema, let $u_1\_E,...,u_n\_E$ [1] be a set of entity types having the same set of attributes and identifier K. Suppose $u_1,...,u_n$ are values from some attribute B. Let A be an attribute of $u_1\_E,...,u_n\_E$ resp. We call A a *global attribute* of $u_1\_E,...,u_n\_E$ resp., if for any two entities of u_i_E and u_j_E (for any $i, j \in \{1,...,n\}$) resp., if they have the same value on K, the value sets of A of the two entities resp are the same. Otherwise, we call A a *local attribute* of $u_1\_E,...,u_n\_E$ resp.

Ex. 4.1: For example, in Fig. 1.1, consider the ER schema of *DB2*. *pname* is a global attribute of the entity types *Jan_prod*,…, *Dec_prod* resp., as for any two product entities of entity types *M1_prod* and *M2_prod* resp (for any *M1*, *M2* $\in$ {*Jan,...,Dec*}), if they have the same values on the identifier *p#* (i.e., they represent a same product), they must have the same values on the *pname* attribute. That is, a product's name is only dependent on the product number, independent of the months. On the other hand, *qty* is a local attribute as the supplying quantities of a same product may be different in different months. □

Definition 2 (*Local/global attribute of relationship set*): Given an ER schema, let $u_1\_R,...,u_n\_R$ be a set of relationship sets involving the same set of entity types and having the same identifier K. Suppose $u_1,...,u_n$ are values from some attribute B. Let A be an attribute of $u_1\_R,...,u_n\_R$ resp. We call A a *global attribute* of $u_1\_R,...,u_n\_R$ resp., if for any two relationships of u_i_R and u_j_R (for any $i, j \in \{1,...,n\}$) resp., if they are the same, the value sets of A of the two relationships resp are the same. Otherwise, we call A a *local attribute* of $u_1\_R,...,u_n\_R$ resp.

Ex. 4.2: Suppose sales companies keep the supply information of products (prices, quantities and suppliers) of each month. Suppose a constraint says "*the price of a product depends only on the product and supplier, while the quantity also depends on month*". They may design one of the following relational schemas to keep the data:

- *DB1*: *SupPri(p#,s#,price), SupQty(p#,s#,month,qty)*. The two relations keep the price and quantity information of products resp, in which *p#* and *s#* represent identifiers of products and suppliers resp. Note in relation *SupQty*, months are modeled as values of attribute *month*. The following FDs hold in the relations of *DB1*: *p#,s#* → *price*; *p#,s#,month* → *qty*.
- *DB2*: *Jan_sup(p#,s#,price,qty)*,…,*Dec_sup(p#,s#,price,qty)*. In these schemas, months are modeled as relation names. Each relation keeps the supply information of one month. The following FD holds in each relation of *DB2*: *p#,s#* → *price,qty*.
- *DB3*: *Sup(p#,s#,price,jan_qty,...,dec_qty)*. In this schema, months are modeled as attribute names. The attributes *jan_qty,...,dec_qty* represent the supplying quantity of products in different months resp. The following FD holds in the relation of *DB3*: *p#,s#* → *price,jan_qty,...,dec_qty*.

Again, to integrate those databases, a federated system first translate the relational schemas to ER schemas (i.e. the common data model) as Fig. 4.1, in which months are modeled as attribute values, relationship set names and attribute names resp.

In Fig. 4.1, in the ER schema of *DB2*, *price* is a global attribute of the relationship sets *Jan_sup,...,Dec_sup* resp., as given any two relationships of the relationship sets *M1_sup* and *M2_sup* resp (for any *M1*, *M2* $\in$ {*Jan,...,Dec*}), if they are the same, they must have the same value on the *price* attribute. On the other hand, *qty* is a local attribute as the supplying quantities of a same product supplied by a same supplier may be different in different months. □

[1] The naming convention is to connect u_i (values of some attribute) and E (representing the meaning of entity types) with an underscore.

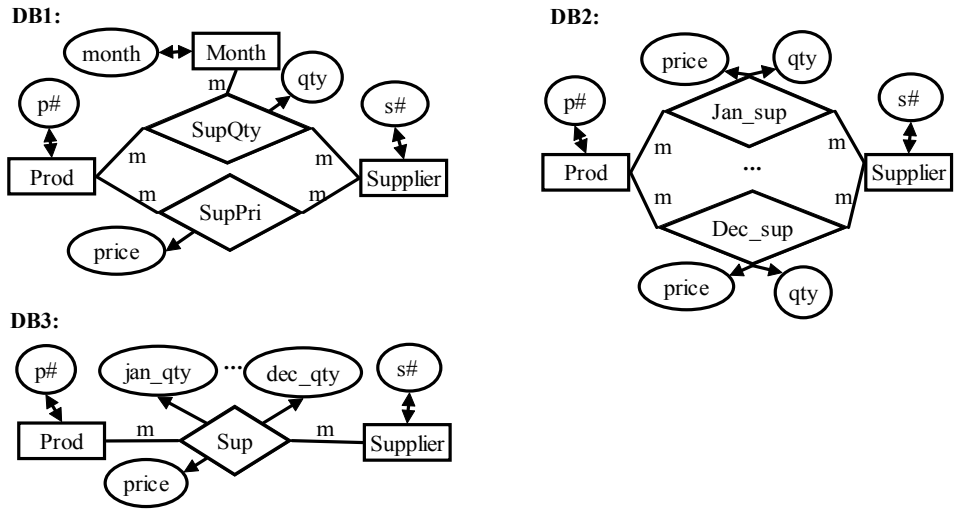

Fig. 4.1: Schematic discrepancy in relationship sets: months are modeled as attribute values, relationship set names or attribute names in different databases

Definition 3 (*Local/global identifier of relationship set*): Given an ER schema, let $u_1\_R,...,u_n\_R$ be a set of relationship sets involving the same set of entity types and having the same identifier K. Suppose $u_1,...,u_n$ are values from some attribute B. We call K a *global identifier* of $u_1\_R,...,u_n\_R$ resp if for any two relationships of u_i_R and u_j_R (for any $i, j \in \{1,...,n\}$) resp, if they have the same value on K, the two relationships are the same. Otherwise, we call K a *local identifier* of $u_1\_R,...,u_n\_R$ resp.

Ex. 4.3: In the ER schema of Fig. 4.2, *p#* is the identifier of relationship sets *Jan_sup,...,Dec_sup* resp. If for any product, the supplier supplying the product is uniquely determined in all the months, then *p#* is a global identifier of those relationship sets resp. Otherwise, *p#* is a local identifier of those relationship sets resp. □

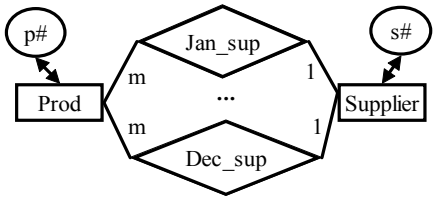

Fig. 4.2: Local/global identifier of relationship set

Note we do not define the concept of "*local/global identifier of entity type*". The reason is we assume in one database, the identifiers of entity types are always global. For example, in Fig. 1.1, in the schema of *DB2*, we assume *p#* uniquely identifies products from different entity types. This assumption is reasonable as the data are from one database instead of multiple databases.

The problem is given an entity type or relationship set, how to know whether an attribute (identifier) is local or global. Generally, as other integrity constraints, local/global attributes (identifiers) must be identified and specified based on the semantics of the real-world enterprise being modeled. By looking at an instance of a database, we might be able to tell that a certain attribute (identifier) is local. However, we can never deduce that an attribute

(identifier) is global by looking at one or more instances of the database because global attribute (identifier) is a statement about all possible legal instances of the database.

At last, we define the concept of *unified schema*. As mentioned, to resolve schematic discrepancy in the integration of ER schemas, we will transform discrepant ER schemas to a unified form. We define the "unified form" as follows.[2]

Definition 4 (*Unified schema*): Given a set of ER schemas to be integrated, we call an ER schema a *unified schema* if in this schema, all the information causing schematic discrepancy with other schemas is modeled as attribute values. Otherwise (i.e. the information is modeled as schema labels), we call the schema *un-unified.*

Ex. 3.4: For example, in Fig. 1.1, the ER schemas of *DB1, DB2* and *DB3* are schematic discrepant because the information, months, are modeled differently in the 3 schemas. And the schema of *DB1* is unified as months are modeled as attribute values in it, while the schemas of *DB2* and *DB3* are un-unified as months are modeled as entity type names or attribute names in them resp. Similarly, in Fig. 4.1, the ER schema of *DB1* is unified, while the schemas of *DB2* and *DB3* are un-unified.

5 Resolving Schematic Discrepancy in the Integration of ER Schemas

In this section, we propose a method to resolve schematic discrepancy in the integration of ER schemas using the semantics of local/global attributes and identifiers. We assume the naming conflicts among component schemas have already been resolved. Generally, in the presence of schematic discrepancy, we propose to integrate the component ER schemas in 3 steps:

Step1: Transform discrepant component schemas to unified schemas resp.;
Step2: Integrate the transformed schemas to an integrated schema;
Step3: If necessary, transform the integrated schema to an un-unified one.

In Step1, each component schema which is un-unified is transformed to a unified one. To this end, we propose algorithms (Section 5.1) to deal with different kinds of schematic discrepancy during the transformation. In Step2, the transformed unified schemas are integrated to one schema. Note in this step, the other conflicts (e.g., structural conflicts, constraint conflicts and contextual conflicts) should be resolved if any. As those conflicts have already been solved by previous works, we do not discuss this step in detail. Step3 (Section 5.2) is a converse of Step1, which transforms the unified integrated schema to an un-unified one (i.e., transform attribute values to schema labels). The purpose of Step3 is to customize the integrated schema or to improve the query efficiency.

5.1 Transforming Component Schemas to Unified Schemas

In the following, Algorithm Trans_schematic_discrepant_schema gives the general process to transform an un-unified component ER schema to a unified one, which will call 4 sub-algorithms to resolve different kinds of schematic discrepancy. Specifically, Algorithm Trans_ent_type_name (Trans_rel_set_name) transforms an ER schema whose entity type names (relationship set names) are modeled as attribute values or attribute names in other ER

[2] The concept corresponds to *flat schema* in [Laks99] or *first-order schema* in [Mill98].

schemas to a schema in which those entity type names (relationship set names) are modeled as attribute values. Algorithm Trans_ent_type_attr (Trans_rel_set_attr) transforms an ER schema whose attribute names of an entity type (relationship set) are modeled as attribute values, entity type names or relationship set names in other ER schemas to a schema in which those attribute names are modeled as attribute values. Note the order to call the four sub-algorithms is important to the correctness of a transformation.

The four sub-algorithms we will give consider all kinds of schematic discrepancy which may occur in ER schemas. This ensures our method can deal with any schematic discrepancy problem in the integration of ER schemas.

The transformation algorithm we will give is information capacity preserving [Mill93]. That is, the mapping from the instances of original schema to the instances of transformed schema is one to one. This can be proved as the 4 sub-algorithms called in the algorithm is information capacity preserving. This ensures a source database can be queried through the transformed schema correctly.

Algorithm Trans_schematic_discrepant_schema(*S*)

INPUT: (1) S, the ER schema of a component database to be integrated; (2) all the entity type names, relationship set names and attribute names in the given ER schema which cause schematic discrepancy with other component schemas; (3) all the local/global attributes and identifiers of entity types and relationship sets resp in the given ER schema; (4) the attributes whose values are modeled as entity type names, relationship set names or attribute names in the given ER schema; (5) the attributes whose domain is the union of the domains of a set of attributes in the given ER schema

OUTPUT: the transformed ER schema which is unified

/*Transform entity types */

For each set of entity types whose names are $u_1\_E, \ldots, u_n\_E$, such that $u_1, \ldots, u_n$ are values of some given attribute B, and are modeled as attribute values or attribute names in other component schemas

Call Algorithm Trans_ent_type_name($u_1\_E, \ldots, u_n\_E, B$)

For each entity type E having a set of attributes $u_1\_A, \ldots, u_n\_A$, such that the values of $u_1\_A, \ldots, u_n\_A$ are values of a given attribute A; $u_1, \ldots, u_n$ are values of a given attribute B, and are modeled as attribute values, entity type names or relationship set names in other component schemas

Call Algorithm Trans_ent_type_attr($E, u_1\_A, \ldots, u_n\_A, B, A$)

/*Transform relationship sets */

For each set of relationship sets whose names are $u_1\_R, \ldots, u_n\_R$, such that $u_1, \ldots, u_n$ are values of some given attribute B, and are modeled as attribute values or attribute names in other component schemas

Call Algorithm Trans_rel_set_name($u_1\_R, \ldots, u_n\_R, B$)

For each relationship set R having a set of attributes $u_1\_A, \ldots, u_n\_A$, such that the values of $u_1\_A, \ldots, u_n\_A$ are values of a given attribute A; $u_1, \ldots, u_n$ are values of a given attribute B, and are modeled as attribute values, entity type names or relationship set names in other component schemas

Call Algorithm Trans_rel_set_attr($R, u_1\_A, \ldots, u_n\_A, B, A$)

Algorithm Trans_ent_type_name($u_1\_E, \ldots, u_n\_E, B$)

INPUT: (1) the ER schema to be integrated (we treat it as a global variable and do the transformation on it directly); (2) $u_1\_E, \ldots, u_n\_E$, a set of entity types having the same set of attributes and identifier; (3) attribute B, such that $u_1, \ldots, u_n$ are values of B; (4) all the local/global attributes of $u_1\_E, \ldots, u_n\_E$ resp

OUTPUT: the transformed ER schema in which $u_1, \ldots, u_n$ are modeled as attribute values

If there's no entity type E_B whose identifier is B

Then Construct an entity type E_B with the only attribute (identifier) B;

Construct an entity type E involving all the attributes of $u_1\_E, \ldots, u_n\_E$ resp.;

Construct a new relationship set R associating E and E_B, in which the cardinalities of both entity types are "m";

Move all the local attributes of $u_1\_E, \ldots, u_n\_E$ resp from E to R;

Replace u_i_E ($i \in \{1, \ldots, n\}$) with E in the relationship sets that u_i_E participates in;

Remove the original entity types $u_1\_E, \ldots, u_n\_E$.

Remark: Fig. 5.1 explains Algorithm Trans_ent_type_name. In the original entity types, the names of entity types, $u_1, \ldots, u_n$, are values of some attribute B; K is the identifier, and A is a local attribute. In the transformed schema, E_B is a new entity type constructed by the algorithm if it does not exist already.

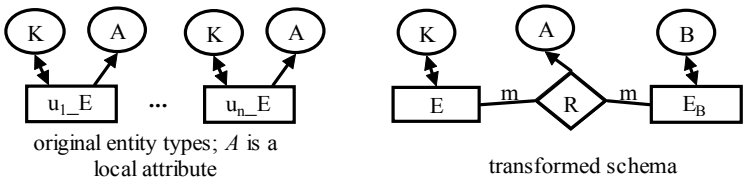

Fig. 5.1: Transform entity type names to attribute values

As an example, in Fig. 1.1, Algorithm Trans_ent_type_name(*Jan_prod,…,Dec_prod, month*) transforms the schema of *DB2* to that of *DB1*.

Algorithm Trans_ent_type_attr(*E*, u_1_*A*,…,u_n_*A*, *B*, *A*)

INPUT: (1) the ER schema to be integrated; (2) entity type *E* and a set of attributes, u_1_*A*,…,u_n_*A*, of *E*; (3) attribute *B*, such that u_1,…,u_n are values of *B*; (4) attribute *A*, such that all the values of attributes u_1_*A*,…,u_n_*A* are from the domain of *A*.

OUTPUT: the transformed ER schema in which u_1,…,u_n are modeled as attribute values.

If there's no entity type E_B whose identifier is *B*

Then Construct an entity type E_B with the only attribute (identifier) *B*;

Construct a new relationship set *R* associating *E* and E_B, in which the cardinalities of both entity types are "m";

Construct an attribute *A* of *R* whose cardinality is the same as u_1_*A*,…,u_n_*A* in *E*;

Remove the attributes u_1_*A*,…,u_n_*A* of *E*.

Remark: Fig. 5.2 explains Algorithm Trans_ent_type_attr. In the original entity type, attribute names $u_1,\ldots,u_n$ are values of some attribute *B*; *K* is the identifier. In the transformed schema, E_B is the entity type constructed by the algorithm if it does not exist already.

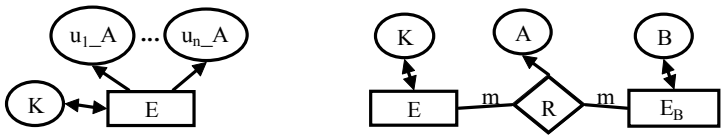

Fig. 5.2: Transform names of attributes of entity types to attribute values

As an example, in Fig. 1.1, Algorithm Trans_ent_type_attr(*Prod, jan_qty,…,dec_qty, month, qty*) transforms the schema of *DB3* to that of *DB1*.

Algorithm Trans_rel_set_name (u_1_*R*,…,u_n_*R*, *B*)

INPUT: (1) the ER schema to be integrated; (2) u_1_*R*,…,u_n_*R*, a set of entity types involving the same set of entity types and having the same identifier; (3) attribute *B*, such that u_1,…,u_n are values of *B*; (4) all the local/global attributes and identifier of u_1_*R*,…,u_n_*R* resp.

OUTPUT: the transformed ER schema in which u_1,…,u_n are modeled as attribute values.

If there's no entity type E_B whose identifier is B

Then Construct an entity type E_B with the only attribute (identifier) B;

Construct a relationship set R involving E_B and all the entity types participating in $u_1\_R,\ldots,u_n\_R$ resp., in which the cardinality of E_B is "m", and the cardinalities of the other entity types are the same as those in $u_1\_R,\ldots,u_n\_R$. The attribute set of R is consisted of all the local attributes of $u_1\_R,\ldots,u_n\_R$ resp.;

If there're any global attributes in $u_1\_R,\ldots,u_n\_R$ resp.

Then construct a new relationship set R' involving all the entity types participating in $u_1\_R,\ldots,u_n\_R$, in which the cardinalities of entity types are the same as those in $u_1\_R,\ldots,u_n\_R$;

If the identifier of $u_1\_R,\ldots,u_n\_R$ resp is local

Then adjust the cardinalities of the entity types participating in R' to "m";

Add all the global attributes of $u_1\_R,\ldots,u_n\_R$ resp to R';

Remove the original relationship sets $u_1\_R,\ldots,u_n\_R$.

Remark: Fig. 5.3 explains Algorithm Trans_rel_set_name. In the transformed schema, $u_1,\ldots,u_n$ become values of attribute B. Relationship set R is constructed to replace the original relationship sets $u_1\_R,\ldots,u_n\_R$. As $A1$ is a global attribute of $u_1\_R,\ldots,u_n\_R$ resp., which is only dependent on $E1$ and $E2$ (independent on E_B), the relationship set R' is constructed for $A1$.

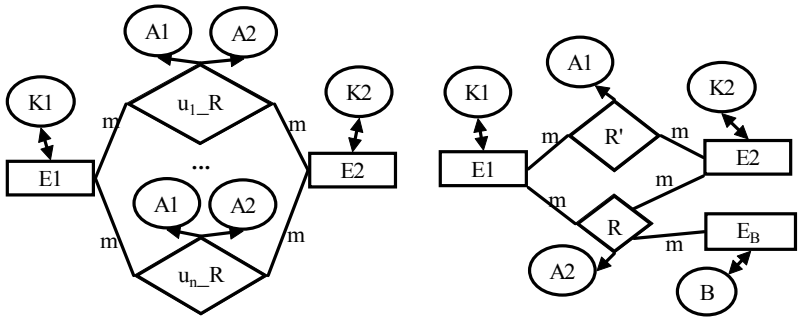

Fig. 5.3: Transform relationship set names to attribute values

As an example, in Fig. 4.1, Algorithm Trans_rel_set_name(*Jan_sup,…,Dec_sup, month*) transforms the schema of *DB2* to that of *DB1*.

Algorithm Trans_rel_set_attr (*R, u_1_A,…,u_n_A, B, A*)

INPUT: (1) the ER schema to be integrated; (2) entity type *R* and a set of attributes, u_1_A,…,u_n_A, of *R*; (3) attribute *B*, such that $u_1,…,u_n$ are values of *B*; (4) attribute *A*, such that all the values of attributes u_1_A,…,u_n_A are from the domain of *A*.

OUTPUT: the transformed ER schema in which $u_1,…,u_n$ are modeled as attribute values.

If there're any other attributes except u_1_A,…,u_n_A in *R*

Then construct a new relationship set *R'* involving all the entity types participating in *R*, in which the cardinalities of entity types are the same as those in *R*;

Move all the attributes of *R* except u_1_A,…,u_n_A to *R'*;

If there's no entity type E_B whose identifier is *B*

Then Construct an entity type E_B with the only attribute (identifier) *B*;

Connect E_B to *R* with the cardinality "m";

Replace the attributes u_1_A,…,u_n_A with a single attribute *A* in *R* whose cardinality is the same as that of u_1_A,…,u_n_A.

Remark: Fig. 5.4 explains Algorithm Trans_rel_set_attr. In the transformed schema, $u_1,…,u_n$ which are modeled as attribute names in original relationship set become values of attribute *B*.

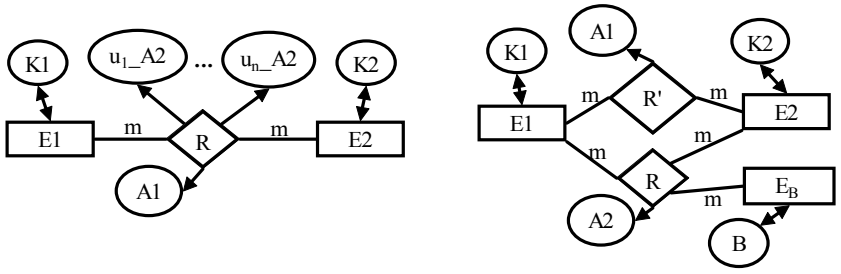

Fig. 5.4: Transform names of attributes of relationship set to attribute values

As an example, in Fig. 4.1, Algorithm Transf_rel_set_attr(*Sup, jan_qty,…,dec_qty, month, qty*) transforms the schema of *DB3* to that of *DB1*.

Ex. 1.1 (Fig. 1.1) and Ex. 4.2 (Fig. 4.1) give simple examples of schematic discrepancy as there's only one attribute (i.e. *month*) whose values are modeled differently in component schemas. In the following, we give a more complex example to show the process of integrating ER schemas in the presence of schematic discrepancy using Algorithm Trans_schematic_discrepant_schema.

Ex. 5.1: Suppose we want to integrate the two ER schemas in Fig. 5.5 (we omit the original relational schemas for convenience). In the ER schema of *DB1*, each entity type models the supplying information in one month. In each entity type, the attributes *s1_price,…,sn_price* represent prices of products supplied by different suppliers resp. We assume *s1_price,…,sn_price* are local attributes of *Jan_prod,…,Dec_prod* resp. In the ER schema of *DB2*, the attributes *jan_qty,…,dec_qty* represent supplied quantities in different months. Note supplier numbers are modeled as attribute names in *DB1*, but attribute values in *DB2*, and months are modeled as entity type names in *DB1*, but attribute names in *DB2*. That is, there're 2 attributes whose values cause schematic discrepancy in this example.

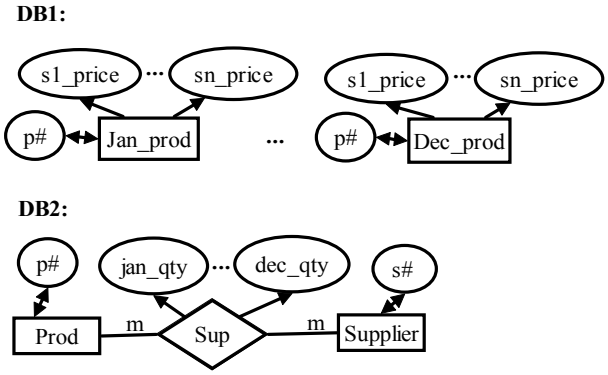

Fig. 5.5: Schematic discrepant schemas: supplier names and months modeled differently in *DB1*and *DB2*

Fig. 5.6 shows the transformed schemas of *DB1* and *DB2* using our algorithm. Note to transform the ER schema of *DB1*, the sub-algorithms Trans_ent_type_name(*Jan_prod,…, Dec_prod,month*) and Trans_rel_set_attr(*Sup, s1_price,…,sn_price, supplier, price*) are called sequentially. To transform the ER schema of *DB2*, the sub-algorithm Trans_rel_set_attr(*Sup, jan_qty,…,dec_qty, month, qty*) is called. The integrated ER schema is obtained by merging the transformed schemas directly as no other conflicts (e.g. structural conflicts) exist. □

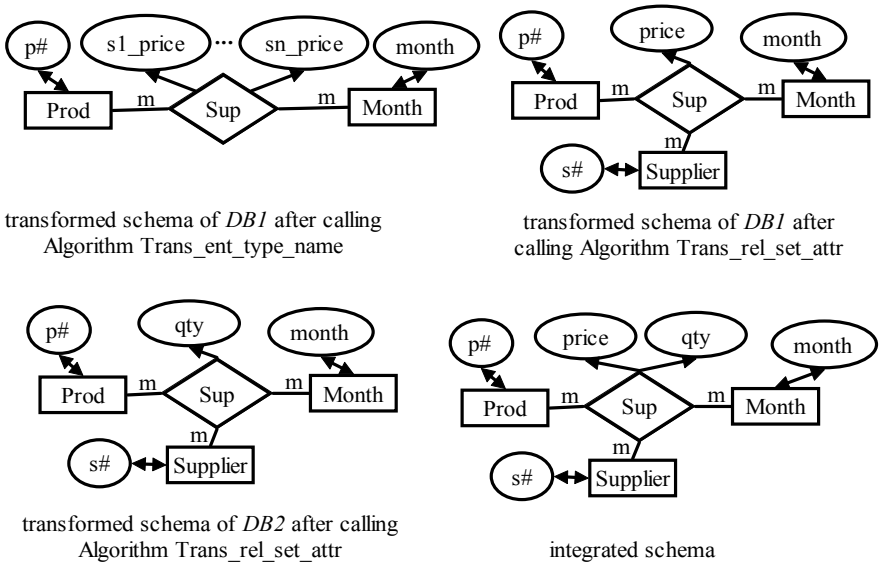

Fig. 5.6: Transform and integrate discrepant schemas

5.2 Transforming Integrated Schema to Un-unified One

This subsection is about the 3rd step of the integration process, as mentioned at the beginning of Section 5. This step is meaningful as we can customize the integrated schema according to users' requirements. For example, in a federated database system, users of a component database may want to access all the databases with a schema similar to the component database which may be un-unified. Furthermore, using an appropriate schema can improve query efficiency [Agra01, He03].

This step (Step3) is a converse of the transformation of Step1 (Algorithm Trans_schematic_discrepant_schema). That is, this step transforms a unified schema to an un-unified one. Recall in Section 5.1, we developed 4 sub-algorithms to deal with different kinds of schematic discrepancy. Similarly, to implement Step3, we need also develop 4 sub-algorithms. But this time, the original (transformed) schemas of those sub-algorithms implementing Step1 become transformed (original) schemas in the sub-algorithms implementing Step3. We omit the detailed algorithms as they are just the converse of those in Section 5.1, while give an example as follows.

Ex. 5.2: In Fig. 5.6, the integrated schema can be transformed to Fig. 5.7, such that the values of attribute *month* in the original schema become relationship set names in the transformed schema. This is a converse transformation of the Algorithm Trans_rel_set_names in Section 5.1. □

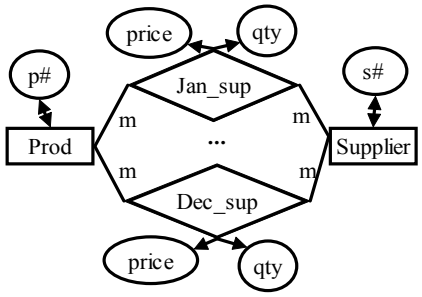

Fig. 5.7: Un-unified schema transformed from the integrated schema in Fig. 5.6

6 Conclusion

Schematic discrepancy is common in real world applications [Kris91, Laks01]. People have done some work on schematic discrepancy in relational model. But little work has been done in the context of ER model which plays an important role in a federated database system. The resolution of schematic discrepancy among ER schemas needs more semantics than that of relational schemas as ER model carries more semantics than relational model. In this paper, we focused on the resolution of schematic discrepancy in the integration of ER schemas. We defined the concepts of local/global attributes and identifiers of entity types and relationship sets resp., which provide essential semantics to transform discrepant ER schemas. Using those concepts, we proposed algorithms to resolve schematic discrepancy problem in the integration of ER schemas.

Reference

[Agra01] Rakesh Agrawal, Amit Somani, Yirong Xu, Storing and querying of e-commerce data, *VLDB journal*, 2001.

[Bati84] Batini,C.,Lenzerini,M, A methodology for data schema integration in the Entity-Relationship model. *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 6, November 1984. 650-664

[Bati86] C. Batini, M. Lenzerini, A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys*, Vol 18, No 4, 1986.

[Chen76] Chen,P.P., The entity-relationship model: toward a unified view of data, *ACM TODS* vol 1, no 1, 1976.

[He03] Qi He, Tok Wang Ling, Resolving contextual conflicts in the integration of ER schemas, *technique report, School of Computing, National University of Singapore*, 2003.

[Jark00] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, Panos Vassiliadis, Fundamentals of data warehouses, *Springer press* 2000.

[Kris91] Ravi Krishnamurthy, Witold Litwin, William Kent, Language features for interoperability of databases with schematic discrepancies, *SIGMOD conference*, 1991.

[Laks99] Lakshmanan, L. V. S., Sadri, F., Subramanian, S. N., On efficiently implementing schemaSQL on SQL database system, *VLDB conference* 1999

[Laks01] Lakshmanan, L. V. S., Sadri, F., Subramanian, S. N. SchemaSQL—an extension to SQL for multidatabase interoperability, *TODS* 2001

[Lee95] Mong Li Lee, Tok Wang Ling, Resolving structural conflicts in the integration of entity-relationship schemas, *OOER conference* 1995.

[Lee97] Mong Li Lee, Tok Wang Ling, Resolving constraint conflicts in the integration of entity-relationship schemas, *ER conference* 1997.

[Ling85] Ling, T.W., A normal norm for Entity-Relationship Diagrams, *Proc. 4th International Conference on Entity-Relationship Approach*, 1985.

[Ling96] Tok Wang Ling, Mong Li Lee, Issues in an entity-relationship based federated database system, *CODAS* 1996.

[Low01] Wai Lup Low, Mong Li Lee, Tok Wang Ling, A knowledge-based approach for duplicate elimination in data cleaning, *Information Systems 26(8)* 2001.

[Mill93] R.J.Miller, Y.E.Ioannidis, r.Ramakrishnan, The use of information capacity in schema integration and translation, *VLDB conf*, 1993.

[Mill98] R J Miller, Using schematically heterogeneous structures, *SIGMOD* 1998.

[Redd95] M.P.Reddy, B.E.Prasad, Amar Gupta, Formulating global integrity constraints during derivation of global schema, *Data & Knowledge Engineering*, 1995

[Shet90] Sheth,A.P. and Gala,S.K. Federated database systems for managing distributed, heterogeneous, and autonomous databases, *ACM Computing surveys*, 22, 3, 1990.

[Verm96] M.W.W.Vermeer & P.M.G.Apers, On the application of schema integration techniques to database interoperation, *Conf. ER'96*.