A Taxonomy of Changes in Schema Evolution

Nayyer Masood
Department of Computer Science
Mohammad Ali Jinnah University
Islamabad, Pakistan
nayyer@jinnah.edu.pk

Naira Andleeb
Department of Computer Science
Mohammad Ali Jinnah University
Islamabad, Pakistan
naira@jinnah.edu.pk

***Abstract*— Schema evolution is the ability of a database system to respond to changes in the real world by allowing the schema to evolve. Two main categories of research work on schema evolution exist in literature; the changes-based and the tgd-based (target generating dependencies that represent mappings between elements). The tgd-based work is more recent and is mainly based on the use of composition and inversion operators to manage schema evolution. This body of work has established different combinations of tgds that may be formed as a result of a particular schema change, like LAV to GAV. The change in type of mapping (tgd) is then handled by use of inversion and composition operators. However, these operators have not been tested for different types of schema changes. This requires the merging of two aforementioned categories of work on schema evolution which is the focus of this paper. This work provides a basis for studying the applicability of inversion and composition operators on different schema changes falling in each tgd-based category thus helps to deal schema evolution in a better way.**

Keywords—composition, inversion, schema evolution, schema integration, target generating dependencies.

I. INTRODUCTION

The applications involving multiple, distributed and heterogeneous databases have to manage diversified form of data even within the same domain. Managing data in such an environment means finding the corresponding data elements from different resources and transforming them into a single form to provide a comprehensive view. Schema mapping provides foundation for transformation of such data. It specifies how data of one schema, called source schema, corresponds to data of another schema, the target schema. Schema mapping has been used in different contexts, like in the context of data exchange to transform the instances of source schema to target schema or in data integration where queries are processed on set of heterogeneous data sources and others. Figure 1.1 below presents the scenario of schema mapping in the context of data exchange.

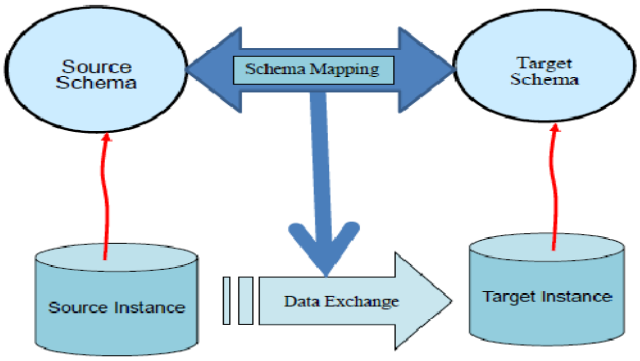

Figure 1. Schema mapping in data exchange.

In order to give a clear understanding of the schema mapping we have presented a scenario in the following.

Source Schemas	Target Schemas
S1: U-List (Name, WebId) S2: Institute (Name, City) S2: Inst-Amt (Inst, ShortId, Amt) S3: HEC-List (Id, Name, WebId) S3: HEC-Amount (C-Id, Amt)	T1: UI-List (Id, Name, Webid) T2: Amount (Amt, C-Id)

Figure 2. Example Source and Target Schemas

In first column of figure 2, three source schemas S1, S2 and S3 are given; each one containing different tables. The target schema consists of UI-list and Amount tables. Users pose queries to the target schema and data is fetched from source schemas to answer those queries. This requires some form of mapping from source schemas to target schema. There are many methods in which this type of transformation is established. Figure 3 below presents the correspondence between our example source schemas and the target schema.

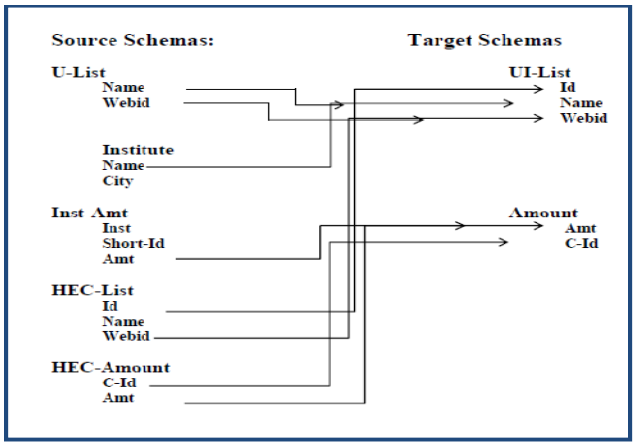

Figure 3. Schema Matching

Next step is to create set of mappings that relate the elements of source schema to the target schema. This schema mapping provides foundation for transformation of source data to the target data. It specifies how data of source schema can be transformed to the data of the target schema. A schema mapping is defined in the form of a triplet consisting of source schema, target schema and set of dependencies as follow [12]

$$M= (S,T, \Sigma)$$

Where S is source schema, T is target schema and $\sum$ explains how source schema relations are related to target schema relations in form of dependencies called source-to-target tuple generating dependencies (s-t-tgds). The s-t-tgds specify that which tuples should appear in target based on the tuples that appear in source or how instances of source schemas are converted to instances of target schema. Following is format to define s-t-tgd:

$$\phi(x) \rightarrow \exists y \, \Psi (x, y)$$

The $\phi(x)$ represents source schema relation(s), Ψ (x, y) represents target schema relation(s) and $\exists$ symbol is used for existential quantifier for an unknown variable used in target schema if it is not defined in the source schema. There are different types of tgds such as LAV (local as view), GAV (global as view) and GLAV (global and local as view) [12].

In given example, desired schema mapping can be described using following s-t-tgds:

m1: U-list (n, w) $\rightarrow \exists$ I UI-List (I, n, w)
m2: Institute (n, c) ^ Inst-Amt (i, s, a) $\rightarrow \exists$ I, W UI-List (I, n, W) ^Amount (a , I)
m3: HEC-List (i, n, s) $\rightarrow$ UI-List (i , n, s)
m4: HEC-Amount (c, a) $\rightarrow$ Amount (a, c)

Figure. 4. Source-to-Target tgds Examples

It is not very uncommon that the database schemas once defined and deployed have to be changed to cater the changes in requirements. Schema evolution is the ability of a database system to respond to changes in the real world by allowing the schema to evolve [3]. Many researchers have worked on schema evolution; some have focused on change-based approaches and some have discussed it in the form of tuple-generated-dependencies. These two approaches are discussed below:

In change-based approach, all possible changes on schemas are identified and categorized first, like adding/removing the relations or attributes and so on. Then different approaches are suggested to handle the these changes [7,8,18,19].

In tgd-based approach, when source or target schema is evolved to a new schema then this schema evolution is specified by an arbitrary schema mapping. New adapted schema mapping is obtained from original schema mapping and arbitrary schema mapping through the use of schema mapping operators composition and inversion [12,19,20].

The aim of our work is to combine these two approaches to provide a comprehensive view of schema evolution. Initially we shall summarize and categorize all types of changes in a systematic way discussed in literature, thereafter, we shall discuss different types of tgds combinations and provide a hierarchy of all types of tgds combinations possible. We will explain in detail that which changes are supported by different types of tgds combinations.

II. LITERATURE SURVEY

We have divided literature survey into two sections; one based on changes-based and other on tgd-based.

A. Review of change-based approaches to Schema Evolution

Sjøberg [18] has focused on finding all possible changes which are applicable in relational data model and provides a list of these changes, like adding new relations, fields, deleting relations and fields. Author has also discussed the aftereffects of these changes on the schema. These changes have been implemented on a real system (Health Management system) and consequences of these changes have also been discussed. Author has not given the strategy to deal with the effects of these changes.

Velegrakis et al [19] have handled schema evolution problem by first identifying the list of changes and classifying them into three categories. An algorithm has been proposed which reorganizes the particular schema mapping that gets effected in case of a source or target schema change. It the modifies the effected mapping to adjust the schema change.

Curino et al [7, 8] present an approach that aims to automate the process of schema evolution by using a tool named PRISM. A language comprising chema modification operators (SMO) has been proposed that hanfles a predefined list of schema changes. The extended version of the PRISM (named PRISM++) has the

capability of handling changes in Integrity Constraints as well. PRISM++ is first system that offers end-to-end support for query and update adaptation through both structural schema changes and integrity constraints evolution. Although PRISM tool has been designed to fully automate the schema evolution but still it depends on human DBA to solve complex issues.

B. *Review of tgd-based Approaches to handle Schema Evolution*

Popa L and Yu C [20] have considered mapping composition approach to handle the schema evolution. Their work is based on mapping composition and mapping pruning techniques. In this approach schema evolution is represented as arbitrary schema mapping. Then composition algorithm is applied between original mapping and evolved mapping. Finally a new mapping is obtained between original mapping and evolved mapping in which semantics of original schema is also incorporated.

Arenas et al [3] present a survey of the work done on composition and inversion operators by different researchers. The aspects that authors have focused in their survey are semantics of the changes, language used to express changes and algorithm to deal the changes.

Fagin et al [10, 13] have defined composition problem in schema evolution as if two source-to-target schema mappings are given then how to generate a mapping between source schema of first and target schema of second schema mapping. They give different proposition that which types of s-t tgds are handled by first-order tgds and which ones are handled by second-order tgds. In [13] authors have discussed quasi inverse of schema mapping which is relaxation of inverse of schema mapping. Authors define quasi-inverse as follows:

Schema mapping $M' = (T, S, M')$ is a quasi-inverse of M if $M \circ M' = Id$ holds modulo the equivalence relation $\sim_M$.

Arenas et al [2] discuss that how to recover data back if data is exchanged from source to target schemas. They extend the work of Fagin [10, 13] and explain that there are many s-t tgds whose inverse and quasi-inverse doesn't exist. In this paper, authors explain how to recover M from M' .i.e. maximum recovery which is mapping that brings back maximum amount of sound information.

Fagin et al [12] handle schema evolution problems with the use of two operators composition and inversion. Authors discusse how to incorporate the changes from old schema mapping to new schema mapping using the two operators. They have used different types of tgds like LAV, GAV and GLAV and gave few examples to handle changes on these tgds. For the target evolution they use composition operator to map source schema with new target schema. Then they show that first-order tgds are not sufficient for composition of some tgds so they use second-order tgds. For the source evolution, first, the inverse of given schema mapping is taken. Then it is checked that exact-chase inverse exists or not, and on the basis of this result composition is performed between old schema mapping and inverse of schema.

III. PROPOSED SYSTEM FOR TESTING SCHEMA EVOLUTION

Both approaches of handling schema evolution have their respective advantages; in change-based schema specific changes have been discussed in depth, whereas tgd-based approach is more recent and have presented a formal solution of schema evolution. We believe that these two dimensions need to be merged in order to have a clearer and better understanding of the issues and to deal schema evolution in a better way. In this research, we have proposed a system to test the ability of any system to handle the schema changes (schema evolution) using composition and inversion operators. For this, we have proposed a comprehensive list of tgds-based changes, we collected different types of schema changes from literature and then we merged these schema changes in each of the tgd-based change. Moreover, we have discussed this list both on the source and target sides by giving an example for each change in each tgds-based change. This establishes a combined taxonomy of schema changes, that can later be used to test the handling of these changes using composition and inversion operations. The proposed work can precisely be represented in the following steps:

1. Prepare comprehensive lists of changes based on tgds and schema changes
2. Merge the schema changes in each of the tgds-based changes
3. Test the applicability of composition and inversion operators on these changes

The figure below presents our proposed system. First two steps have been discussed in this paper, whereas last step is our proposed future work

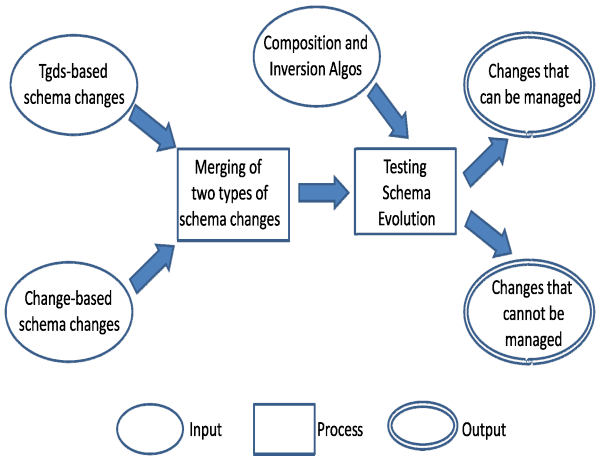

Figure 5. Proposed System's Architecture

A. Lists of schema changes

In this step we have compiled two lists of schema changes from the literature. One list is based on tgds and other on different types of schema changes.

B. Change-based Schema Evolution

There is a volume of research where schema evolution has been discussed on the basis of particular schema changes [7,8, 9,18,19]. In this section, we have presented a comprehensive list of such schema changes. We do not find a comprehensive discussion on the treatment of all possible schema changes. Another aspect that is missing in this research is that the factor of schema evolution on source and target sides has not been discussed separately as in [12]. This does make a difference because same change when introduced on source or target side, like add attribute, needs to be treated differently. We have placed changes into two major categories; Relation based and Attribute based.

C. Changes in Relations

Following are different changes that can be made on the relations included in the source or target sides

AR: Add Relation; it means a new relation is added on the source or target side. The addition of only those relations is of concern for the schema evolution that somehow affect source to target mapping

DR: Delete Relation; a relation involved in schema mapping is deleted

JR: Join Relation; two relations are joined with each other forming one relation

SR: Split Relation; one relation is split into two relations

RR: Rename Relation; a relation is renamed

D. Changes in Attributes

Different types of changes can be made in the attributes on the source or target sides. The names of these changes, like of relations, are self-explanatory and do not need elaboration

AA Add Attribute
DA Delete Attribute
RA Rename Attribute
CA Copy Attribute
MA Move Attributes

As discussed earlier, all these changes can be made on the source or the target side and in each case may need different treatment. In our final list of schema changes we have considered these changes on both sides.

E. Tgds-based Schema Evolution

Many researchers [5,10,12,15,20] have discussed schema evolution on the basis of mapping between source and target schemas. The schema mapping between S and T is represented by a triple $M = \{S, T, \sum\}$ where $\sum$ is set of tgds between S and T. The second list that we have prepared is based on such work. If S is a source schema and T is target schema then the source-to-target tuple generating dependencies (source-to-target tgd) represent the relationship between elements of S and T. A tgd is typically represented as a formula in first order logic represented in [12] as:

$$\phi(x) \rightarrow \exists y \, \Psi (x, y)$$

It is obvious from the above discussion that tgds play the central role in schema mapping. A source-to-target tgd is also called GLAV (Global and Local as View). In GLAV we can define more than one relation on source or target side. Existential variable can also be defined in GLAV. GLAV contains two constraints LAV((Local-As-View) and GAV (Global-as-View). LAV is an s-t-tgd in which source-side contains a single relation. Second type of constraint GAV is an s-t tgd where target side contains a single relation with no existential variable. Considering the target schema evolution, if M is an LAV then a change on the target side resulting M' may be of either of LAV, GAV or GLAV type. Considering all possible combinations, we have following 9 cases for M → M' scenario:

GAV→ GLAV	**LAV→ GLAV**	**GLAV →GLAV**
GAV→ LAV	**LAV →LAV**	**GLAV → LAV**
GAV→ GAV	**LAV → GAV**	**GLAV →GAV**

Same 9 cases arise for the M → M", i.e. when a change is made on the source side

F. Merging of Schema Changes Lists

In this section, we are discussing the second step of our propose work in which we will merge the two lists presented in the previous section and also we present example of each change in each tgd-based change. This will build in-depth understanding of different scenarios of schema evolution that can occur in real life systems.

G. Combined List of Target Side Schema Changes

Now we will discuss different types of (selected) changes on the target side in each tgd-based schema evolution scenario. We will consider first category of tgd combinations GAV – GLAV transformation

This category concerns the schema evolution where a change in target schema T of a GAV mapping M causes a new schema mapping M′ which is of GLAV type, this transformation can be represented as:

M (GAV) → M′ (GLAV)

If we have given following GAV mapping M:

Student (id, name, phone) ^ Address (id, city) →
Std-Detail(id, name, city)

Above mapping is GAV because it does not contain existential variable and there is only one relation on target side. Suppose we add a new relation on target side of M. A new mapping M′ is formed from old target to new target as follow:

Std-Detail (id, name, city) →□P Std-Detail(id, name, city) ^ Std-Ph (id, P)

As we can see that above mapping is GLAV because it contains more than one relation on target side and also an existential variable, So this is an example of M (GAV) to M′ (GLAV).

In the following, we discuss different types of changes in GAV-GLAV transformation:

AR (Add Relation)

Suppose we define following GAV mapping M:

Student (st-deg, st-name, st-credit) → St-Rec (st-deg, st-name)

Student (st-deg, st-name, st-credit) → St-Credit (st-deg, st-credit)

Target schema T of mapping M is changed to new target schema T' by adding a new relation Grade.

Schema mapping transforming T to T' denoted by M' is as follows:

St-Rec(st-deg, st-name) ^ St-Credit(st-deg, st-credit) → □G St-Rec(st-deg, st-name) ^ St-Credit(st-deg, st-credit) ^ Grade (st-deg, G)

DR (Delete Relation)

To delete a relation from Target schema we will use *M (GAV)* as follow:

Std_Course (st-id, st-sem, st-year, st-degree) → St-Sem (st-id, st-sem)

Std_Course (st-id,st-sem, st-year, st-degree) → St-Year(st-id, st-year)

Std_Course (st-id, st-sem, st-year, st-degree) → St-Degree (st-id, st-degree)

From above *M* suppose we remove the relation *St-Degree* from target schema**.** Schema mapping transforming *T* to *T′* denoted by *M′* as *GLAV* is as follow:

St-Sem (st-id, st-sem) ^ St-Year (st-id, st-year) ^St-Degree (st-id, st-degree) → St-Sem (st-id, st-sem) ^ St-Year (st-id, st-year)

AA (Add Attribute)

To add a new attribute in target schema we will use *M (GAV)* as follow:

Lecturer (st-id, lct-name, lct-dpt)→Lect-Biodata (lct-name, lct-dpt)

Assume that Target schema *T* of mapping *M* is changed to new target schema *T′* by adding a new attribute *Salary***.** Schema mapping transforming *T* to *T′* denoted by *M′* ***as*** *GLAV* is as follows

Lect-Biodata (lct-name, lct-dpt) → ∃SALARY Lect-Biodata(lct-name, lct-dpt, SALARY)

DA (Delete Attribute)

Suppose we want to remove an attribute *dpt-building* from target relation *Dept-Add* in following schema mapping where M is defined as GAV:

Dept (dpt-id,dpt-name, dpt-building, dpt-phone) → Dept-Add(dpt-id, dpt-building, dpt-phone)

Target schema *T* of mapping *M* is changed to new target schema *T′* by removing an attribute *dpt-building* from *Dept-Add***.** Schema mapping transforming *T* to *T′* denoted by *M′* as *GLAV* as follows

Dept-Add (dpt-id, dpt-building, dpt-phone) →Dept-Add (dpt-id, dpt-phone)

Similarly we apply these changes on other tgd combinations *GAV→ LAV, GAV→ GAV, LAV→ GAV, LAV→ LAV, LAV→ GAV, GLAV→ GLAV, GLAV→ GAV, GLAV→ LAV* . Following is a summarized list of changes applied on all tgd combinations:

TABLE I: TARGET SIDE CHANGES

Combined List of Target Side Schema Changes

Tgd Combinatins	AR	DR	AA	DA
GAV-GLAV	Yes	Yes	Yes	Yes
GAV-LAV	Yes	N/A	Yes	Yes
GAV-GAV	N/A	N/A	N/A	Yes
LAV-GLAV	Yes	Yes	Yes	Yes
LAV-LAV	Yes	Yes	Yes	Yes
LAV-GAV	N/A	Yes	N/A	Yes
GLAV-GLAV	Yes	Yes	Yes	Yes
GLAV-LAV	Yes	Yes	Yes	Yes
GLAV-GAV	N/A	Yes	N/A	Yes

H. Combined List of Source Side Schema Changes

Now we will discuss different types of (selected) changes on the source side in each tgd-based schema evolution scenario. We will consider first category of tgd combinations *GAV – GLAV transformation*

This category concerns the schema evolution where a change in source schema S of a GAV mapping M causes a new schema mapping M″ which is of GLAV type, this transformation can be represented as:

M (GAV) → M″ (GLAV)

In the following, we discuss different types of changes in GAV-GLAV transformation:

AR (Add Relation)

Suppose we define following GAV mapping M

Faculty (fac-id, fac-name, fac-phone) → Fac-Data (fac-id, fac-name)

Source Schema S of mapping M is changed to new Source schema S″ by adding a new relation Fac-Add Schema mapping transforming S to S″ denoted by M″ as GLAV is as follows

Faculty (fac-id, fac-name, fac-phone) →□FCITY Faculty (fac-id, fac-name, , fac-phone) ^ Fac-Add(fac-id, fac-phone, FCITY)

DR (Delete Relation)

To delete a relation from Source schema first we will define M as GAV as follow

Lect-Sem(lct-id, st-sem) ^ Lect-Crse (lct-id, crse-id) → Lect-Teaches(lct-id, crs-id,st-sem)

From above M suppose we remove relation Lect-Crse from source schema transforming S to S″ denoted by M″ as GLAV is as follow:

Lect-Sem (lct-id, st-sem) ^ Lect-Crse (lct-id, crse-id) → Lect-Sem(lct-id, st-sem)

AA (Add Attribute)

To add a new attribute in source schema we will use following schema mapping M as GAV

Exam-Section (st-id, st-degree, st-result) →

St-Transcript (st-id, st-result)

Suppose that we add a new attribute in Source schema S. Schema mapping transforming S to S″ is as follows where M″ is GLAV

Exam-Section(st-id, st-degree, st-result) →

□ST-STATUS Exam-Section(st-id,st-degree,st-result, ST-STATUS)

DA (Delete Attribute)

To delete an attribute from source schema we will use following schema mapping M as GAV

Att-Dpt(st-id, st-crse, st-ph) → St-Lve (st-id, st-leave)

Suppose that we delete attribute st-ph from Source schema S. Schema mapping transforming S to S″ where M″ as GLAV is as follows

Att-Dpt(st-id, st-crse, st-ph) → Att-Dpt(st-id, st-crse)

Similarly we apply these changes on other tgd combinations GAV→ LAV, GAV→ GAV, LAV→ GAV, LAV→ LAV, LAV→ GAV, GLAV→ GLAV, GLAV→ GAV, GLAV→ LAV .

Following is a summarized list of changes applied on all tgd combinations:

TABLE 2: SOURCE SIDE CHANGES

Combined List of Source Side Schema Changes

Tgd Combinations	AR	DR	AA	DA
GAV-GLAV	Yes	Yes	Yes	Yes
GAV-LAV	Yes	N/A	Yes	Yes
GAV-GAV	N/A	Yes	N/A	Yes
LAV-GLAV	Yes	Yes	Yes	Yes
LAV-LAV	Yes	N/A	Yes	Yes
LAV-GAV	N/A	N/A	N/A	Yes
GLAV-GLAV	Yes	Yes	Yes	Yes
GLAV-LAV	Yes	N/A	Yes	Yes
GLAV-GAV	N/A	Yes	N/A	Yes

IV. CONCLUSION AND FUTURE WORK

Schema evolution has been studied using change-based and tgd-based approaches in literature. We elaborated work of researchers on both of these approaches and then merged both approaches. For this, first we have explained all types of changes possible on schemas and then applied these changes on different types of tgds (LAV, GAV, GLAV). If a change is not applicable on a particular type of tgd then we gave reasons for this. In future, we plan to apply the composition and inversion operators on the schema changes listed in different tgd-based categories.

V. REFERENCES

[1] P. Alvaro, W. Marcza, N. Conway, J. Hellerstein, and D. Maier, "Datalog in Time and Space," Verlag Berlin Heidelberg:Springer, 2012, pp.262-281.

[2] M. Arenas, and J. Perez, "The Recovery of a Schema Mapping: Bringing Exchanged Data Back," Transactions on Database Systems, Vol. 34, No. 4, Article 22, December 2009. pp 13–22.

[3] M. Arenas, J. Perez, J. Reutter, and C. Riveros, "Composition and Inversion of Schema Mappings," SIGMOD Record, September 2009, pp. 16-28.

[4] M. Arenas, J. Perez, J. Reutter, and C. Riveros, "Query Language based Inverses of Schema Mappings: Semantics, Computation, and Closure Properties," The VLDB Journal 2012, pp. 823-842

[5] C. Arocena Patricia, F. Ariel, and R.J. Miller, "Composing Local-As-View Mappings: Closure and Applications," Proceedings of the 13th International Conference on Database Theory. 2010, pp. 209-218.

[6] P. Bernstein, J.T. Green, S. Melnik, and A. Nash, "Implementing Mapping Composition," VLDB 2006, pp. 333–353.

[7] C. Curino, H. Moon, and C. Zaniolo, "Automating Database Schema Evolution in Information System Upgrades," VLDB 2008, pp. 441-444.

[8] C. Curino, H. Moon, A. Deutsch, and C. Zaniolo, "Update Rewriting and Integrity Constraint Maintenance in a Schema Evolution Support System: PRISM++," Proceedings of the VLDB Endowment, Vol. 4, No. 2 Copyright 2010, pp.126-128

[9] C. Curino, H. Moon, A. Deutsch, C. Zaniolo, "Automating the Database Schema Evolution Process," The VLDB Journal 2013, pp. 63–98

[10] R. Fagin, P. Kolaitis, L. Popa, and W. Tan, "Composing Schema Mappings: Second-Order Dependencies to the Rescue Transactions on Database Systems," ACM TODS, Vol. 30, No. 4, December 2005, p.994–1055.

[11] R. Fagin, P. Kolaitis, L. Popa, and W. Tan, "Reverse Data Exchange: Coping with Nulls," ACM TODS, Vol. 36, No. 2, Article 12, May 2012, pp 23–32.

[12] R. Fagin, P. Kolaitis, L. Popa, and W. Tan, "Schema Mapping Evolution through Composition and Inversion," in Schema Matching and Mapping, Z. Bellahsene, A. Bonifati, and E. Rahm, Eds. Springer, 2012, pp. 191-222.

[13] R. Fagin, P. Kolaitis, L. Popa, and W. Tan, "Quasi-inverses of Schema Mappings," PODS 2006, June, 2006, Beijing, China, pp. 123–132.

[14] M. Hartung, J. Terwilliger, and E. Rahm, "Recent Advances in Schema and Ontology Evolution," in Schema Matching and Mapping, Z. Bellahsene, A. Bonifati, and E. Rahm, Eds. Springer, 2012, pp. 149-190.

[15]. J.M. Hyun, "Supporting Schema Evolution in Information Systems and Historical Databases" Ph.D. Dissertation, UoC LA, 2008.

[16] P. Kolaitis, "Schema Mappings Data Exchange & Metadata Management," PODS 2005, Baltimore, Maryland. pp. 61-65.

[17] M. Golfarelli , J. Lechtenbörger, S. Rizzi, and G. Vossen, "Schema Versioning and Cross-Version Querying in Data Warehouses," Trans DKE, 2006, pp. 435–459.

[18] D. Sjøberg, "Quantifying Schema Evolution," Information and Software Technology, Vol. 35, January 1993, pp. 35-44

[19] Y. Velegrakis, R. Miller, and L. Popa, "Mapping Adaptation under Evolving Schemas," VLDB, Berlin, Germany, 2003, pp 584–595.

[20] C. Yu, and L. Popa, "Semantic Adaptation of Schema Mappings when Schemas Evolve," VLDB, Trondheim, Norway, 2005, pp 1006–1016.