Merging Multidimensional Data Models: A Practical Approach for Schema and Data Instances

Michael Mireku Kwakye, Iluju Kiringa, Herna L. Viktor
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada.
mmire083@uottawa.ca, {kiringa, hlviktor}@eecs.uottawa.ca

***Abstract*—Meta-model merging is the process of incorporating data models into an integrated, consistent model against which accurate queries may be processed. Within the data warehousing domain, the integration of data marts is often time-consuming. In this paper, we introduce an approach for the integration of relational star schemas, which are instances of multidimensional data models. These instance schemas represented as data marts are integrated into a single consolidated data warehouse. Our methodology which is based on model management operations focuses on a formulated merge algorithm and adopts first-order Global-and-Local-As-View (GLAV) mapping models, to deliver a polynomial time, near-optimal solution of a single integrated data warehouse.**

Keywords-Schema Merging; Data Integration; Model Management; Multidimensional Merge Algorithm; Data Warehousing

I. INTRODUCTION

Schema merging and data integration are important research areas with many practical applications. Some of the application areas are federated database systems, Enterprise Information Integration (EII), bioinformatics data integration, and financial information integration. Schema merging involves the integration of instance schema of meta-data models using the mappings between the elements of the instance schemas [1]. Data integration, on the other hand, involves the consolidation of the instance data within the framework of a merged instance schema to deliver efficient query solutions [2]. Most procedures that involve these concepts have focused on traditionally identifying the independent data sources and the associated element mapping correspondences. Recent studies have emphasized the importance of inferring the semantic meaning of the data source elements during integration. Some problems that are associated with the procedural methodologies for these concepts are the identification of prime meta-models, and the formulation of algorithms for specific meta-models and their schema and data instances.

The conceptual processes of data integration and schema merging largely come from the fundamental operations of model management [3] [4]. Some of these operations are namely, *match schemas* (expressed as schema matching), *compose mappings* and *apply mappings* (both expressed as schema mapping discovery), and *merge schemas* (expressed as schema merging) [3]. In line with multidimensional data integration for data warehouses, a number of studies have been investigated. Cabibbo and Torlone [5] [6] introduce and address dimension algebra and dimension compatibility in relation to data marts integration. Riazati et al. [7] also propose a solution for integration of data marts where they infer on the aggregations in the hierarchies of the dimension. Although these studies and others attempt to address this integration problem, they fail to investigate in detail areas such as an elaborate merge algorithm, element conflict management, technical merge requirements, amongst others.

In this paper, we introduce an integration procedure for both instance schema and instance data of multidimensional data models. Our motivation is to employ the concept of model management to address the above-mentioned shortcomings of merge algorithm, conflict management and technical merge requirements for integration of data marts. Our key contribution in this paper is the formulation of a novel well-defined algorithm capable of delivering an efficient integrated data warehouse. Our presentation focuses on the proposition of star schema instances in our analyses. We deal with different procedures starting with finding of mapping correspondences to a more complex procedure of merging. Our work subsumes and extends prior work on generic models [1], to present a practical solution for merging schema instances of multidimensional data models.

The technical contributions may be summarized as follows. We adopt a hybrid form of schema matching, in which we use both instance schema structure and instance data and extension algorithms to deliver correct attribute mapping correspondences. To this end, we employ first-order Global-and-Local-As-View (GLAV) mapping models in the mapping discovery procedure. We identify and resolve specific conflicts that are exposed as a result of the integration of data marts. We further define technical qualitative merge correctness requirements which serve to validate the formulation of our merge algorithm.

This paper is organized as follows. In Section II, we discuss our integration methodology. We present the multidimensional instance schema and data merging in Section III. In Section IV, we address the implementation and evaluation analysis of the merge methodology. In Section V, we conclude, discuss the open issues, and the areas of future work.

II. INTEGRATION METHODOLOGY

Our approach for generating a single integrated data warehouse from independent, but related, multidimensional star schemas extends from the above-mentioned concept of model management.

 ISBN: 978-1-61208-247-9

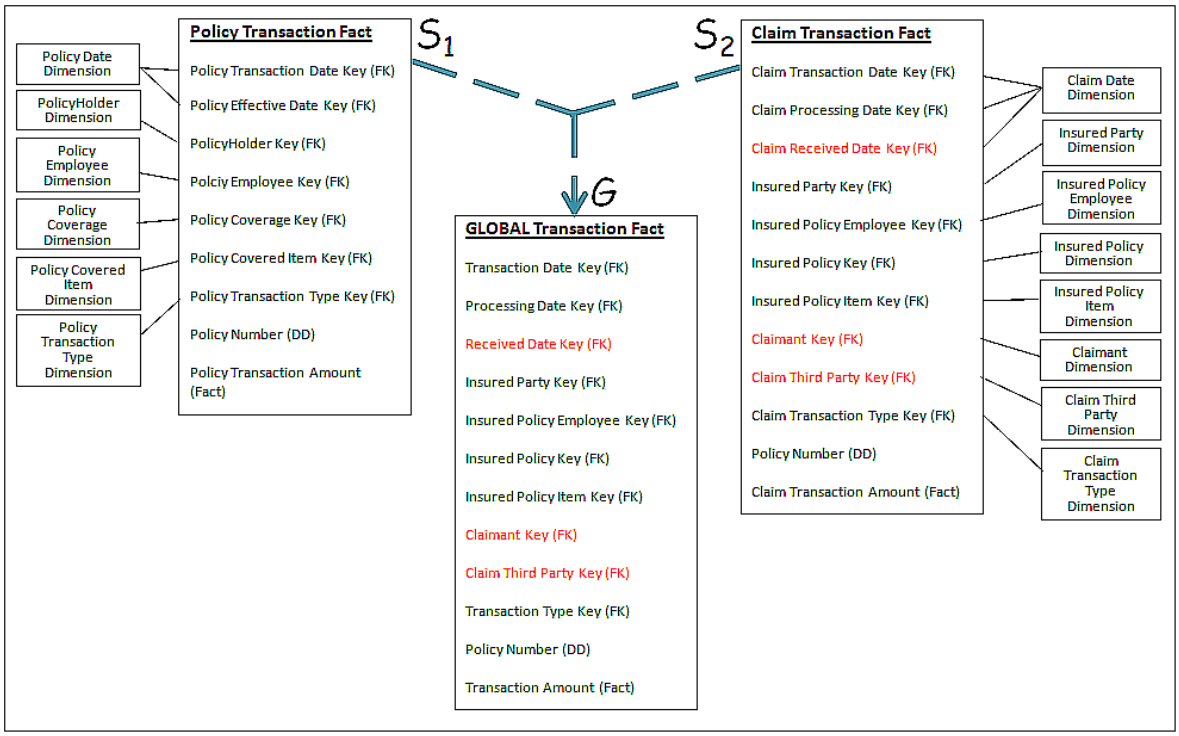

Figure 1. Logical and Conceptual Multidimensional Schema Merge

In line with this meta-data conceptual assertion, we present an overview of our integration methodology, as depicted in Figure 1. The figure shows a logical and conceptual merging of the fact and dimension tables from the *Policy* and *Claims* data marts, of an Insurance industry, to form an enterprise data warehouse. We explain further our motivation using Example 1 and Figure 1.

Example 1. *Suppose we have 2 data marts from an Insurance industry – Policy Transactions and Claims Transactions – and we have to integrate these data marts into an enterprise-wide data warehouse, as illustrated in Figure 1. The existence of corresponding attributes will enable the possibility of integrating the attributes of the fact and dimension tables of these data marts. A merge algorithm can be applied to the corresponding mappings to generate the integrated data warehouse needed in answering queries, as it will be posed to the independent data marts.* ■

A. Overview of Integration Methodology

We outline our methodology based on 3 main streamlined procedures. These are finding mapping correspondences, mapping models discovery, and the formulation of merge algorithm. Figure 2 illustrates a description of our methodology and framework architecture in a workflow chain. Here, we describe the step-wise procedures and processes, algorithm executions, and the generated outputs, as well as, query analyses. We further describe into detail the first 2 procedures (Finding Mapping Correspondences and Mapping Models Discovery & Modelling) and give also a detailed description of procedure 3 (Merge Algorithm) in Section III.

B. Finding Mapping Correspondences

In our methodology, we adopt a hybrid form of schema matching which aim to deliver efficient schema attribute correspondences. Our adoption of this hybrid approach uses the logical and conceptual features of the multidimensional schema structure in schema-based matching and the instance data and extensions in instance-based matching, to find attribute correspondences. We adopted schema-based algorithms in the form of Lexical Similarity and Semantic Names. The Lexical Similarity uses schema string names and text, equality of names, synonyms, homonyms, and similarity of common substrings. The Semantic Names, on the other hand, uses schema data types, constraints, value ranges, relationship types, amongst others to match attributes [10]. We use Example 2 to illustrate the schema-based form of finding mapping correspondences.

Example 2. *Following up on Example 1, suppose we want to merge the dimensions of DimPolicyHolder and DimInsuredParty from Policy and Claims data marts, respectively. The application of Lexical Similarity algorithm will produce mapping correspondences, such as:*

1. $PolicyHolder.PolicyHolderKey \approx InsuredParty.InsuredPartyKey$

2. $PolicyHolder.FullName \approx InsuredParty.FamilyName, InsuredParty.GivenName, InsuredParty.CityName$

3. $PolicyHolder.Address \approx InsuredParty.StreetAddress, InsuredParty.EmailAddress$

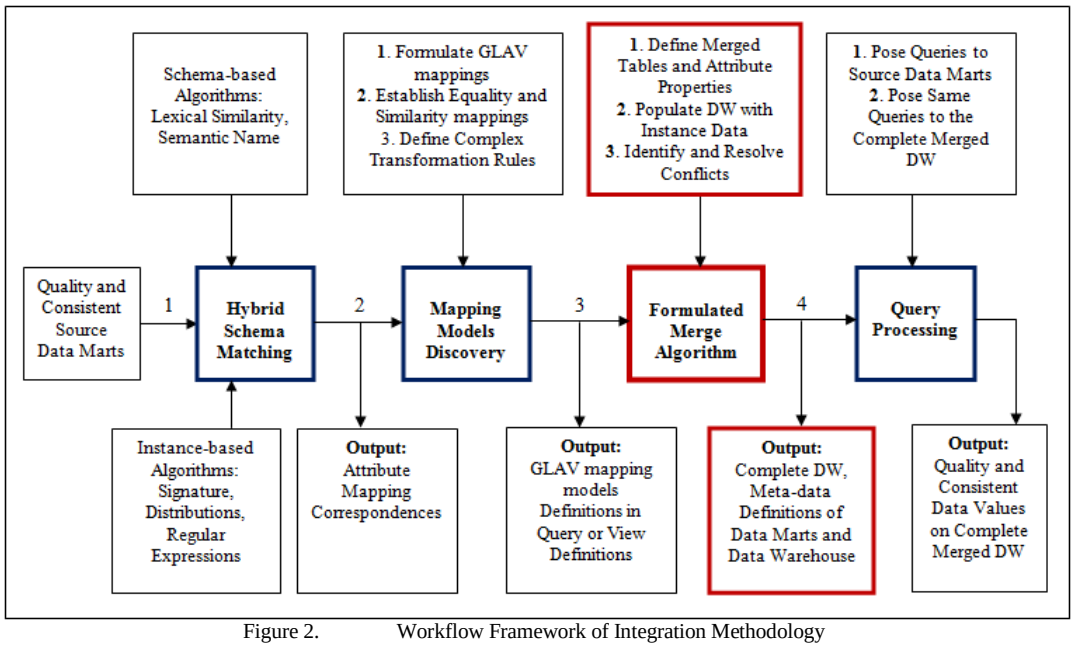

Figure 2. Workflow Framework of Integration Methodology

Moreover, the application of the Semantic Names algorithm will offer an improved schema matching. This matching eliminated ***InsuredParty.CityName*** *in the 2nd matching to deliver mapping correspondence, as in:*

2. $PolicyHolder.FullName\ [varchar(60)] \approx InsuredParty.FamilyName[varchar(30)],\ InsuredParty.GivenNames[varchar(40)]$ ■

The instance-based algorithms that were adopted are Signature, Distributions, and Regular Expressions. The Signature algorithm uses the similarity in the actual data values contained in the schemas based on data sampling. The Distributions algorithm, on the other hand, uses the common values and frequent occurrences of data values based on sampling. The Regular Expressions algorithm uses textual or string searches based on regular string expressions or pattern matching [10]. We use Example 3 to illustrate a generalized form of instance-based algorithm.

Example 3. *Following up on Examples 2, we complement the results of the initial schema-based mapping correspondences with a generalized instance-based mapping to produce a final semantically correct mapping correspondence for the 3rd matching, as in:*

3. $PolicyHolder.Address \approx InsuredParty.StreetAddress$

This final matching was attained because of the data values and extensions from the dimension attributes. Some of the instance data values contained in PolicyHolder.Address are {39 Baywood Drive, 178 Flora Ave., 79 Golden Rain St.}, where as data values contained in InsuredParty.StreetAddress and InsuredParty.EmailAddress are {40 Roslyn St., 68 Hastings Drive, 48 Whitehall Avenue} and {amartens@cybserv.com, drice@vipe2k.com, jtausig@fitexes.com}, respectively. ■

C. Mapping Models Discovery and Modelling

Definition 1. (First-Order Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping model from Source, S and Target, T schemas. Let $a \in \{S \cup T\}$ represent disjoint variable element where a denotes $\{a_1, a_2, \ldots, a_n\}$. The mapping assertion, $\mathcal{M}$ is said to be in first-order if $f: \{\forall a\ (S(a) \rightarrow T(a))\}$, where f represents the logical view from the Source to the Target. ■

We adopted first-order Global-and-Local-As-View (GLAV) mapping model formalisms in the mapping discovery procedural step. Our motivation is based on the expressiveness of the correspondences that exist between the attributes of the schemas [2]. This mapping model combines mapping formalisms from both the Local-As-View (LAV) and Global-As-View (GAV) mappings. It expresses mapping views where the extensions of the source schemas provide any subsets of tuples satisfying the corresponding view over the global mediated schema. Moreover, an equivalent number of attribute view definitions are expressed in both the LAV and GAV queries [2]. One other unique feature is the expression of multi-cardinality mappings between mapping elements. This enables the expression of complex transformation formula which is much useful in our integration methodology [12].

Definition 2. (Equality Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping for Source, S and Target, T schemas. The assertion $f: \{\forall x \forall y\ (S(x, y) \rightarrow \exists z\ T(x, z))\}$ for disjoint variable elements x, y, z is an Equality mapping such that $y = z$. ■

Definition 3. (Similarity Mapping): Let $\mathcal{M} = (S, T, f)$ represent a mapping for Source, S and Target, T schemas. For disjoint element variables x, y, z the assertion $f: \{\forall x \forall y\ (S(x, y) \rightarrow \exists z\ T(x, z))\}$ is a Similarity mapping

such that $g(y) = z$ where g denotes or encloses a complex transformation expression. ■

In this second step of mapping discovery and modelling, 2 forms of mapping relationships were adopted. These are equality and similarity mapping relationships. It should be emphasized that these defined classifications were based on expressive characterization of relationship cardinality, and the attribute semantic representation, amongst others [11]. We used these forms of mapping relationships in a GLAV mapping model, as explained in Example 4.

Example 4. *Continuing on Example 1, suppose we want to integrate the DimPolicyHolder and DimInsuredParty dimensions from Policy and Claims data marts, respectively, into DimInsuredPolicyHolder dimension. The Datalog queries for the GLAV mapping model will be expressed as:*

InsuredPolicyHolder *(InsuredPhKey, InsuredPhID, InsuredPhName, BirthDate, StateProvince, Region, City, Status):-*
PolicyHolder *(PolicyHolderKey, PolicyHolderID, PolicyHolderFullName, DateOfBirth, State, City, Status),*
InsuredParty *(InsuredPartyKey, InsuredPartyID, InsuredFamilyName, InsuredGivenName, BirthDate, Province, Region, CityName)*

In this Datalog query, the existence of corresponding attributes in both dimensions are automatically expressed in the merged dimension, as well as, local attributes of ***Status*** *and* ***Region*** *from Policy and Claims data marts, respectively, are included in the global or merged dimension.* ■

III. MULTIDIMENSIONAL INSTANCE SCHEMA AND DATA MERGING

In this section, we present the technical qualitative requirements necessary for producing an efficient single consolidated data warehouse. We further outline and describe an elaborate merge algorithm (Algorithm 1) for integrating the instance schema and data of data marts fact and dimension tables. We finally describe the resolution of identifiable conflicts associated with the integration of the data marts.

A. Merge Correctness Requirements

The single consolidated data warehouse that is generated as a result of the implementation of the merge algorithm needs to satisfy some requirements, to ensure the correctness of the data values from the queries that would be posed to it. These qualitative technical requirements describe the properties that the data warehouse schema should exhibit.

Drawing on the propositions in the requirements defined by the authors in [1] for merging generic meta-models, we performed a gap analysis on their propositions in relation to generating a data warehouse. Hence, we formulate and describe a set of correctness requirements in relation to merging of multidimensional star schemas. These technical requirements extend the requirements already proposed in [1], in order to address star schemas. We outline the set of *Merge Correctness Requirements* (MCR) that validates the formulated merge algorithm needed for the generation of a global data warehouse.

Dimensionality Preservation. For each kind of dimension table connected to any of the integrating fact tables, there is a representation of corresponding dimension also connected to the merged fact table.

Measure and Attribute Entity Preservation. All fact or measure attribute values in either of the integrating fact tables are represented in the merged fact table. Additionally, all other attribute values in each of the dimension tables are represented through an equality or similarity mapping. Finally, there is an automatic inclusion for non-corresponding attributes in the merged fact (dimension) tables based on the condition of no attribute redundancy or duplication.

Slowly Changing Dimension Preservation. Slowing Changing Dimension is the occurrence where an entity in a dimension has multiple representations based on the changes in instance data values in some key attributes. For such dimensional entity occurences, the merged dimension should offer an inclusion of all the instance representations from each integrating dimension. Hence, we enforce an automatic inclusion of attributes that contribute to the dimensional change in the merge dimension.

Attribute Property Value Preservation. The merged attribute should preserve the value properties of the integrating attributes, whether the mapping correspondence is an equality or similarity mapping. Equality mapping should be trivially satisfied by the *UNION* property for all equal attributes. For a similarity mapping, the transformation expression should have the properties to be able to satisfy the attribute property value of each integrating dimension attribute.

Definition 4. (Surrogate Key): Let $\mathcal{D}_i$ represent a dimension table for a multidimensional model, $\mathcal{B}$ such that $\mathcal{D}_i \in \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n\}$ for $i \leq n$. Let $\mathcal{E}$ represent each entity of a dimension, $\mathcal{D}_i$ such that $\mathcal{E} \in \mathcal{D}_i$. The identifier, $\mathcal{K}$ is said to be a Surrogate Key for $\mathcal{E}$ such that $\mathcal{K}_m \equiv \mathcal{E}_m$ ■

Tuple Containment Preservation. The single consolidated data warehouse should offer the containment of all unique tuples as they are valuable in returning correct answers to queries posed. This ensures the preservation of all *Surrogate Keys* needed in identifying each dimensional entity.

B. Merge Algorithm

The merge algorithm (Algorithm 1) is formulated and designed to generate the single consolidated data warehouse from different related data marts, modelled as star schemas instances. The algorithm primarily performs 2 levels of integration. Firstly, the integration of the instance schema structure which comprises the attribute relationships and properties for the fact and dimension tables. These procedures are described in Steps (1) to (9).

Algorithm 1: Multidimensional Instance Schema and Data Merging

Input:
(a) A set of *star schema* data marts, *A* and *B*
(b) A set of *first-order* GLAV mapping model; $Mapping_{AB}$, consisting of $factMapping_{AB}$ and $dimMapping_{AB}$
(c) An optional designation of a data mart, *A* or *B*, as the $preferredDataMart$;

Output:
(a) A single consolidated star schema instance data warehouse free of *duplicate* and *redundant* schema and instance data.
(b) A metadata consisting of data definition of the integrating data marts and the single consolidated data warehouse.

Procedure:

Initialization
(1) ***Let*** $mergeDataWarehouse \leftarrow NULL$

Generate Merged Table
(2) ***For each*** $correspondingMappingType \in factMapping_{AB}$ ***do***
  (a) ***If*** $corrrespondingMappingType = NULL$ ***then***
    i. ***Return*** $mergeDataWarehouse \leftarrow NULL$
  (b) ***Else***
    i. ***Let*** $mergeDataWarehouse \leftarrow mergeFactTable \in \{factTableA, factTableB\}$
(3) ***Repeat*** Step (2) for each $mergeDimTable$ using $dimMapping_{AB}$, add $\{nonCorrespondingDimTable\}$
(4) ***Return*** $mergeDataWarehouse \supset \{mergeFactTable, \{mergeDimTable, nonCorrespondingDimTable\}\}$

Merged Table Attribute Relationships
(5) ***For each*** $correspondingMappingType \in factMapping_{AB}$ ***do***
  (a) ***Let*** $mergeFactTable \leftarrow NULL$
  (b) ***If*** $correspondingMappingType =$ "*Equality*" ***then***
    i. ***Let*** $mergeFactAttribute \leftarrow definedAttribute \in \{factMapping_{AB} \in preferredDataMart\}$
  (c) ***Else If*** $correspondingMappingType =$ "*Similarity*" ***then***
    i. ***Let*** $mergeFactAttribute \leftarrow definedAttribute \in factMapping_{AB}$
(6) ***For each*** $nonCorrespondingAttribute \in \{factTableA, factTableB\}$ ***do***
  (a) ***If*** $nonCorrespondingAttribute \notin \{mergeAttribute\}$ ***then***
    i. ***Let*** $mergeFactAttribute \leftarrow nonCorrespondingAttriubte$
  (b) ***Return*** $mergeFactTable \supset \{mergeFactAttribute, nonCorrespondingAttribute\}$
(7) ***For each*** $correspondingMappingType \in dimMapping_{AB}$ ***do***
  (a) ***Repeat*** Step (3) for each $correspondingAttribute \in \{dimTableA, dimTableB\}$
  (b) ***Repeat*** Step (4) for each $nonCorrespondingAttribute \in \{dimTableA, dimTableB\}$
  (c) ***Return*** $mergeDimTable \supset \{mergeDimAttribute, nonCorrespondingAttribute\}$

Merged Table Attribute Properties
(8) ***For each*** $mergedFactAttribute \in mergeFactTable$ ***do***
  (a) ***Let*** $mergeAttributeTypeValue \leftarrow definedAttributeType \in factMapping_{AB}$
(9) ***Repeat*** Step (6) for each $mergeDimTable$ using $dimMapping_{AB}$

Dimension Tables Data Population
(10) ***For each*** $mergeDimTable$ ***do***
  (a) ***If*** ($keyIdentifierConflict$ OR $multipleEntityRepresentation$) = $TRUE$ ***then***
    i. ***Let*** $entityKeyIdentifier \leftarrow surrogateKey \in preferredDataMart$
  (b) ***Else***
    i. ***Let*** $entityKeyIdentifier \leftarrow (newSurrogateKey \equiv primaryKey) \in nonPreferredDataMart$

Fact Table Data Population
(11) ***For each*** $mergeFactTable$ ***do***
  (a) Load fact records using $entityKeyIdentifier \in \{surrogateKey, newSurrogateKey\}$
(12) ***Let*** $mergeDataWarehouse \supset \{mergeFactTable, \{mergeDimTable, nonCorrespondingDimTable\}\}$
(13) ***Return*** $mergeDataWarehouse$

Steps (1) to (4) initialize and generate the integrated fact and dimension tables. Steps (5) to (7) describe the generation of attributes for the integrated tables. Finally, Steps (8) and (9) describe the derivation of attribute property values of the merged fact and dimension tables.

Secondly, the algorithm performs integration of the instance data contained in the star schema data marts. This involves the population of these instance data from the data marts fact and dimension tables into the merged tables in the data warehouse. Steps (10) to (13) describe these procedures of data population.

We further summarize the merge algorithm in fulfilment of the technical *Merge Correctness Requirements* (MCRs) outlined in Section III.A.

a) Step (2) satisfies Dimensionality Preservation: Each fact and dimension table is iterated to form the Merged Fact Table.

b) Steps (3), (4), (5) satisfy Measure and Attribute Entity Preservation: All the attributes contained in the Fact or Dimension Tables are represented in the Merged Table (Fact or Dimension) through equality or similarity mapping.

c) Steps (6) and (7) satisfy Attribute Property Value Preservation: Value properties of attributes are represented for each of the Fact or Dimension Tables.

d) Step (8) satisfies Slowly Changing Dimension Preservation and Tuple Containment Preservation: Entity

representations from the different data marts are included in the merged dimensions.

e) Steps (9), (10) satisfy Tuple Containment Preservation: Tuple data values from each of the data marts are populated in the merged data warehouse.

C. Conflicts Identification and Resolution

The integration of meta-data models are generally coupled with different forms of conflicts in either the instance schema structures or instance data. These conflicts are resolved through different propositions from the algorithm and based on the semantic representation of the meta-data models and their instance schemas. In our integration approach, we identify and propose resolution measures for these conflicts that are encountered during merging.

Identifier Conflicts. These conflicts arise as a result of the same identifier for different real-world entities in the merged dimension. These categories of conflicts are practically exposed as a result of the possibility of different entities from the integrating data marts having the same surrogate key identifier in their individual dimensions. A resolution measure for these conflicts is explained in Example 5.

Example 5. *Suppose we aim to merge the employee dimensions into a single merged dimension, using DimPolicyEmployee and DimInsuredPolicyEmployee from Policy and Claims data marts, respectively. In such an integration procedure, it happens that Employee P from DimPolicyEmployee and Employee Q from DimInsuredPolicyEmployee have the same identifiers of a surrogate key. There is the need to resolve such a conflict, in the algorithm, by preserving the surrogate key identifier in the preferred data mart and re-assign a new surrogate key identifier for the non-preferred data mart(s).* ■

Entity Representation Conflicts. These conflicts arise as a result of the multiple representations of the same real-world entity in the merged dimension by the different identifiers. This occurrence is traced to different representations of surrogate key identifiers from different dimensions for the same real-world entity in the merged dimension. Following on Example 1, a proposed resolution measure, outlined in the merged algorithm, will be to perform a de-duplication of the conflicting entities, by preserving the entity from the preferred data mart as the sole representation of the real-world entity in the merged dimension.

Attribute Property Type Conflicts. These forms of conflicts occur as a result of the existence of different attribute property values from the integrating attributes into a merged attribute. Using Example 5, a merged attribute for *HireStatus* and *EmployeeStatus* from *DimPolicyEmployee*, *DimInsuredPolicyEmployee*, respectively, will hold a data type value of, say *varchar(1)*, being the *UNION* of integrating attribute data types for *char(1)* and *bit* data types from *HireStatus* and *EmployeeStatus*, respectively. We resolve these conflicts by using the attribute data types as defined in the mapping model.

IV. IMPLEMENTATION AND EVALUATION

In this section, we discuss the implementation and evaluation work based on the integration methodology and formulated merge algorithm. We present our implementation framework and the procedures we followed, and we discuss and analyze the evaluation results.

A. Implementation

We implemented our methodology using 2 different data warehouses, for the Insurance business and Transportation services. The Insurance data consisted of 2 initial data marts. These were Policy and Claims data marts. The Policy and the Claims data marts contained 7 and 10 dimension table schemas, respectively. The Policy fact table schema contained instance data of 3070 tuples of data, whilst the Claims fact contained 1144 tuples of data. The Transport data set contained 3 data marts. These were Frequent Flyer, Hotel Stays, and Car Rental data marts. Their fact tables contained 7257, 2449, 2449 tuples of data for Frequent Flyer, Hotel Stays, and Car Rental, respectively. The data marts resided in a Microsoft SQL Server DBMS. Each entity representation in the dimensions was identified by a unique surrogate key and with a clustered indexing as created on the primary key.

The schema matching and mapping models discovery procedural steps were implemented using IBM Infosphere Data Architect [9] [10]. This tool incorporated the schemas of the data mart source repositories, together with their contained instance data. The schema matching step was implemented using the set of algorithms incorporated in the application software. The algorithms were configured by sequentially manipulating the order of execution, configuration of rejection threshold, sampling size and sampling rate. The manipulations of these configurations for finding mapping correspondences were based on an iterative procedure of inspection. With regards to the mapping models discovery and modelling step, the adoption of GLAV mappings enabled the inclusion of all attributes for each mapping formulation of fact and dimension table attributes. Moreover, complex transformation expressions were derived for multi-cardinality mappings. An output file in a Comma Separated Values (.csv) format was later generated, which contained the mapping definitions based on the tables, their attributes, and the attribute property values from each of the data marts. The merge algorithm was implemented using C# .Net programming.

B. Evaluation

Our evaluation analyses were primarily based on the single consolidated data warehouse from the formulated merge algorithm, in Section III.B, as against the independent data marts. We compared both the outputs of the query processing on the data marts and the generated data warehouse. We first ran a formulated query on one or more data marts, and afterwards ran the same query on the generated data warehouse. With this ordering, we are able to effectively compare the results from the data marts and the single consolidated data warehouse.

 ISBN: 978-1-61208-247-9

Evaluation Criteria and Analysis. We evaluate the outcome of the experiments conducted based on a set of criteria based on the guidelines proposed by Pedersen et al. [8]. We performed a gap analysis on their study and then adapted the correctness of data values, dimensionality hierarchy, and rate of query processing as criteria.

The metrics that we used in evaluating these criteria for query processing were recall, precision, and accuracy. These were proposed by Junker et al. [13] to evaluate the performance of database query processing and information retrieval. Recall is computed by the number of tuples retrieved from a data mart divided by the number of tuples that should have been retrieved from the generated data warehouse from each original data mart. Precision is computed by the number of tuples retrieved from a data mart divided by the number of tuples that were retrieved from the single consolidated data warehouse, per the data mart. Accuracy is determined by the degree of validity or exactness of the data values generated from a query posed to the data warehouse in comparison to the data values retrieved from a data mart.

All formulated queries that were posed to the data warehouse were based on fact and dimension attributes from all the data marts. For recall, an evaluation of 100% was trivially attained and verified. The verification was based on the assertion that the merge algorithm fulfilled the MCRs of measure and attribute entity preservation and tuple containment preservation.

Precision evaluation was very important, as it measured the proportion of relevant and non-relevant tuples that were retrieved based on a formulated query. This gives us insight into the composition of our merged data warehouse, in terms of the level of integration of related data from multiple sources. Deducing from the precision values, a higher rate was attained for all formulated queries that were posed against the data warehouse. For cases of dimensions that were only related to some specific data marts, a formulated query against the fact and these dimension tables yielded a very high precision rate. This was as a result of the retrieval of few non-relevant tuples. An example query was, "*What insurance claimant employment type receives the most claims processed for the current Calendar Season*"? Conversely, for queries on dimensions that related to all data marts, an average precision rate was observed where a considerable number of non-relevant tuples were retrieved in reference to a particular data mart. An example query was, "*What type of Policy Coverage is most popular? What are the trends since the 2nd Calendar Quarter."*

Figures 3 and 4 show the precision evaluation for Insurance and Transportation data warehouses, respectively. In Figure 3, an average rate of 86% was achieved for the queries posed to dimensions only related to the Claims data mart. The precision rate increases significantly with an increase in the tuples in these dimensions, as more relevant tuples are generated. This is evident in queries 1 to 7. In terms of corresponding dimensions for all data marts, processed queries generated an average rate of 51% and 49% for Claims and Policy data marts, respectively, as highlighted in queries 8 to 12. In Figure 4, an average precision rate of 72%, 74%, and 83% were attained for Hotel Stays, Car Rental, and Frequent Flyer data marts, respectively, for the set of formulated queries posed. In summary, we were able to provide the user with details regarding the proportion of the data in the merged data warehouse that originate from a specific source. This holds important practical value, for data warehouse practitioners, who want to be able to have statistics regarding the composition of the merged data.

In terms of accuracy, we achieved a 100% return rate of valid and exact data values from the data warehouse in comparison to each individual data mart. This was affirmed based on the merge algorithm fulfilling MCRs of Tuple Containment Preservation and Measure and Attribute Entity Preservation. Additionally, the adoption of GLAV mapping model enabled the processing of exact and sound queries on the data warehouse.

We also analyzed the rate of query processing to ensure that queries posed to the data warehouse are of optimal rate. With an integration of instance data from the data marts, a considerable volume of expected data cannot be overemphasized in the data warehouse. We recorded the query response time for an average of 20 query executions for each of the data sets. These queries were processed on a single 3.20 GHz processor with a 4 GB of RAM.

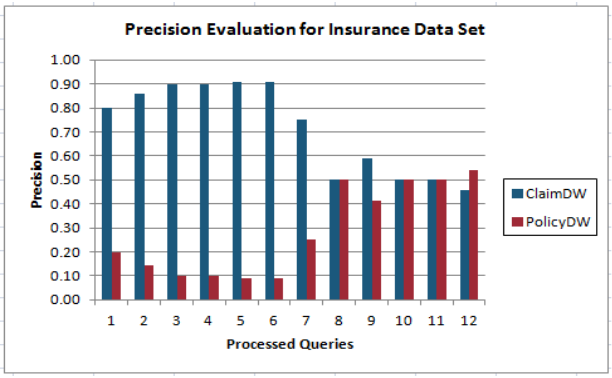

Figure 3. Precision for Insurance Data Set

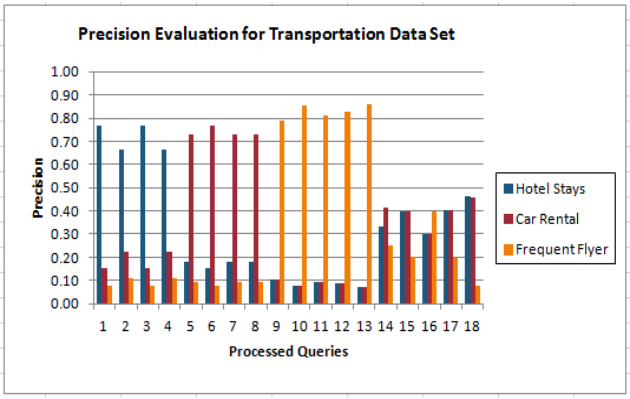

Figure 4. Precision for Transportation Data Set

TABLE I. SUMMARY OF AVERAGE QUERY RESPONSE TIME AND VARIANCES

Data Set	Average Query Response Time and Variances		
	Data Mart / Data Warehouse	*Avg. Query Response (ms)*	*Variance From Integrated Data Warehouse (ms)*
Transportation	Car Rental	26.70	63.95
Transportation	Hotel Stays	27.10	63.55
Transportation	Frequent Flyer	70.95	19.70
Transportation	DataWarehouse	90.65	0.00
Insurance	Policy	29.65	19.60
Insurance	Claim	13.75	35.50
Insurance	DataWarehouse	49.25	0.00

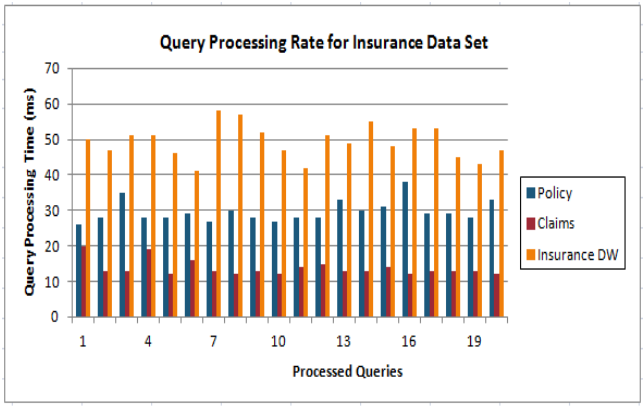

Figure 5. Query Processing Rate for Insurance Data Set

We further computed the variance of the average query rate per data mart as it quantitatively differs from its consolidated data warehouse. Our evaluation showed that queries generally ran at almost the same rate or slightly higher than when posed against the data mart sources.

The query execution durations for the data marts and data warehouses for the Insurance data set are shown in Figure 5. It can be deduced from these data values that the query rate for the data warehouses were appreciable taking note of the compared values generated from the data marts. In some cases, such as queries 7 and 8, the rates were a bit higher due to higher level of aggregation and increased number dimension attributes involved in data values retrieved. We present a summary of the variances in the average query response time for the data marts in comparison to the respective data warehouse. Table 1 shows the query response (in milliseconds) for the Insurance and Transportation data sets.

V. CONCLUSION

This paper presents a methodology for the merging of multidimensional data models using star schemas instances. We formulated a merge algorithm for integrating disparate data marts into a single consolidated star schema data warehouse. We further identified and outlined the resolution of likely conflicts that may be encountered when merging data marts. Moreover, we outlined the satisfaction of some technical merge correctness requirements for integrating data marts into a data warehouse.

Analyses of our evaluation showed that the rates of recall, precision and accuracy of the data values retrieved from the generated data warehouse are high and noticeable. Our approach, thus, provides data warehouse researchers and practitioners with procedures, criteria, and exact measures as to how successful an integration process is achieved.

A number of future research directions remain. The potential enrichment of the mapping language by modelling the functional dependencies between the attributes of the fact and dimension tables is an interesting future direction. Additionally, incorporation of data mart level integrity constraints into the data warehouse needs to be investigated further. We also envisage the extension of the methodology to handle snowflake and fact constellation multidimensional schema models.

ACKNOWLEDGEMENT

This work was supported by grants from the Natural Sciences and Engineering Research Council (NSERC) Strategic Network on Business Intelligence (BI).

REFERENCES

[1] R. A. POTTINGER and P. A. BERNSTEIN, “Merging Models Based on Given Correspondences,” VLDB 2003: 826-873 & Microsoft Research 2000: MSR-TR-2000-53.

[2] M. LENZERINI: Data Integration, “A Theoretical Perspective,” PODS 2002:233-246.

[3] P. A. BERNSTEIN and S. MELNIK, “Model Management 2.0: Manipulating Richer Mappings,” SIGMOD 2007:1-12.

[4] S. MELNIK, “Generic Model Management: Concepts and Algorithms,” Springer LNCS 2967. (2004).

[5] L. CABIBBO and R. TORLONE, “Integrating Heterogeneous Multidimensional Databases,” SSDBM 2005:205-214.

[6] L. CABIBBO and R. TORLONE, “Dimension Compatibility for Data Mart Integration,” SEBD 2004:6-17.

[7] D. RIAZATI, J. A. THOM and X. ZHANG, “Inferring Aggregation Hierarchies for Integration of Data Marts,” DEXA 2010:96-110.

[8] T. B. PEDERSEN, C. S. JENSEN and C. E. DYRESON, “A Foundation for Capturing and Querying Complex Multidimensional Data,” Elsevier Sci. 26(5):383-423 (2001).

[9] R. FAGIN, L. M. HAAS, M. A. HERNÁNDEZ, R. J. MILLER, L. POPA and Y. VELEGRAKIS, “Clio: Schema Mapping Creation and Data Exchange,” Conceptual Modelling: Foundations and Applications 2009:198-236.

[10] IBM: IBM Infosphere Data Architect 7.5.3.0 – Finding Relationships.http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp?topic=/com.ibm.datatools.metadata.mapping.ui.doc/topics/iiymdadconfiguring.html (Accessed–Dec. 8, 2012).

[11] B. TEN CATE and P. G. KOLAITIS, “Structural Characterizations of Schema-Mapping Languages,” ICDT 2009:63-72.

[12] D. KENSCHE, C. QUIX, X. LI, Y. LI and M. JARKE, “Generic Schema Mappings for Composition and Query Answering,” Data Knowl. Eng. (DKE). 68(7):599-621 (2009).

[13] M. JUNKER, A. DENGEL and R. HOCH, “On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy,” ICDAR 1999:713-716.

 ISBN: 978-1-61208-247-9