Instance-Based Integration of Multidimensional Data Models

Michael Mireku Kwakye[1, 2], Iluju Kiringa[1], and Herna L. Viktor[1]
[1] School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
mmire083@uottawa.ca, {kiringa, hlviktor}@eecs.uottawa.ca

[2] Faculty of Informatics
Ghana Technology University College
Accra, Greater Accra, Ghana
mmireku@gtuc.edu.gh

***Abstract*— Meta-model merging is the process of incorporating data models into an integrated, consistent model, against which accurate queries may be processed. The efficiency of such a process is very much reliant on effective semantic representation of chosen data models, as well as the mapping relationships between the schema and data instance elements of the data models. Within the data warehousing domain, the integration of data marts is often time-consuming. Intuitively forming an all-inclusive data warehouse presents tedious tasks of identifying related fact and dimension table attributes. Moreover, the ability to process queries across these disparate, but related, data marts poses an important challenge. 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 enterprise-wide data warehouse.**

Keywords- Schema Merging, Data Integration, Model Management, Mapping Modelling Constraints, Multidimensional Merge Algorithm, Data Warehousing.

I. INTRODUCTION

The concepts of schema merging and data integration present intricate fields in databases as both have academic and industrial implications in the area of data processing. Schema merging involves integrating disparate models of related data using methods of element matching, mapping discovery, and the consolidation of data sets [2]. Schema merging adopts a concept from model management that primarily involves the integration of models and their instance schemas, and together with associated constraints. Data integration entails the consolidation of the instance data sets within the framework of a merged schema to deliver efficient query solutions [3]. The end results of schema and data integration have seen important impacts in various scientific and industrial domains. A number of the application areas are federated database systems, Enterprise Information Integration, and bioinformatics data integration. These applications continue to impact and attract attention in the need for efficient data processing and analytics.

Traditionally, most of the procedures that involve data integration have always been focused on identifying the integrating data sources, and the associated mapping correspondences of elements in the integrating data sources. Recent studies have focused instead on emphasizing the inference of semantic meaning of the elements of the data sources in integration. There usually arise various forms of problems associated with the procedural methodologies for these concepts. These challenges are the identification of prime meta-models, the expression of semantic representation of the meta-models, and the formulation of algorithms for specific meta-models and their instances. In general, these challenges make the overall procedures of data and schema integration very difficult. The conceptual processes of data integration and schema merging are derived from the fundamental operations of model management [4] [5]. Model Management operations of *match*, *compose mappings*, and *merge* offer the intuition to address the problems of data integration and schema merging within the context of multidimensional data models.

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 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, which is supposed to be the end-result of the overall integration process. This algorithm is capable of delivering an efficiently integrated data warehouse. Our presentation focuses on the proposition of star schema instances in our analyses. Our work, which subsumes prior work on generic models [2], draws on a number of their significant propositions made, and uses it as a background work in formalizing our intuition in a much more practical solution for merging schema and data instances of multidimensional data models. Additionally, this paper presents an extended and elaborate version to an earlier submission [1], as the assertions, methodology and evaluated results are described in further detail.

The technical contributions are summarized as follows;

- We adopt the application of a hybrid form of schema matching, in which we use both schema and data instance algorithms to deliver correct attribute mapping correspondences.
- We adopt first-order Global-and-Local-As-View mapping models in the mapping discovery procedure, which expresses the transformation of complex expressions between attributes of the instance schemas.
- We address the handling of functional dependency integrity constraints in the mapping discovery and modelling procedure.
- We identify and specify resolution measures for frequently observed conflicts that are exposed, as a result of integration of heterogeneous data marts.
- We define technical qualitative merge correctness requirements, which serve to validate the formulation of the merge algorithm.
- Most importantly, we formulate a merge algorithm that specifically deals with the integration of schema and instance data of the data marts.

The rest of the paper is organized as follows. In Section II, we review the fundamental background studies regarding data integration and schema merging. In Section III, we discuss our integration methodology, where we address the overview of the integration approach. In Section IV, we describe the adopted hybrid schema matching; and further on in Section V, we describe the mapping models discovery and modelling. In Section VI, we present the formulated merge algorithm; and address the proposition of the technical merge correctness requirements, semantics of query processing, and conflicts resolution measures in Section VII. In Section VIII, we address the implementation and evaluation of the integration methodology. We discuss the related work and comparison of other approaches in Section IX; and in Section X, we conclude, discuss open issues and the areas of future work.

II. BACKGROUND

The need of business users to access, in a timely and precise fashion, information originating from varied and heterogeneous sources of data repositories has lead to the investigation of engineered methods of efficient data integration methods and retrieval. The processes that comprise the generation of the final output of data integration largely stem from the fundamental operations of model management [4]. Models serve as data representation and as a result, different models denote different applications or domains and are modelled for different purposes.

Model management, in the field of databases, is a high-level, abstract programming language designed to efficiently manipulate schemas and mappings. It serves as the generic approach to solving problems of heterogeneity and data programmability, where concise and clear-cut mappings are manipulated to deliver desired output that supports robust operations related to certain metadata-oriented problems [4] [6] [7]. A number of these operations are; *match schemas*, *compose mappings*, *difference schemas*, *merge schemas*, *apply function*, *translate schemas* into different data models, and *generate data transformations* from mappings. The main abstractions that are needed in expressing model management operations are instance schemas and mappings. Practically, the choice of a language to express these instance schemas and mappings is vital. A model is described as a formal description of a complex application artefact, such as; database relational model, Unified Modelling Language (UML) model, or an ontology [4] [6]. A schema is an expression of a model that defines a set of possible instances, whilst and a meta-model is the language needed to express the schemas. These schemas could be relational schema, Extensible Markup Language (XML), Web Ontology Language (OWL), or Multidimensional Schema, amongst others.

Model management operations, in the form of schema matching, schema mappings, and schema merging have generally been attempted by Bernstein et al. [6], Melnik [7], and Gubanov et al. [8] to offer flexibility and efficiency in meta-data processing. To efficiently integrate different data sources, the model management operation of *match*, expressed as schema matching, serves as basis to other major operations [4]. It takes two schemas instances as input and produces a mapping between elements of the two schema instances that correspond semantically to each other [9]. Various surveys and studies have been conducted in the literature [9] [10] [11] in this direction of schema matching, of which incremental and new results have been shown to effectively deliver better solutions in arriving at precise mapping correspondences. Prior studies classify this procedure into 3 main categories. These are namely; schema-level matching, instance-level matching, and hybrid and/or composite matching. Out of these studies and surveys conducted, several concrete results have been developed to produce very high precisions. Enumerations of algorithms are Similarity Flooding (SF) [12], COMA [13], Cupid [14], SEMINT [15], iMAP [16], and the Clio Project [17] [18]. It will be noted that schema matching operations are enhanced from fields, such as; Knowledge Representation [19], Machine Learning [15] [20], and Natural Language Processing [21], where techniques are used to deliver near-automatic and semantically correct solutions.

Other forms of model management operations are *compose mappings* and *apply functions*, expressed as schema mapping discovery. These operations are normally a follow-up on the end results of a schema matching operation. Schema mapping is the fundamental operation that produces a semantic relationship between the associated elements from source and target schemas based on an earlier schema matching [17] [22] [23]. Recent studies conducted in generating schema mappings have shown that the strength of mapping relationship correspondences that exist between schema elements largely determines the degree of efficiency of the overall data integration procedure. Further works have shown that mapping correspondences modelled and expressed in terms of First-Order Logic (FOL) assertions exhibit unique characteristics, where various manipulations on mapping elements can be expressed distinctively. The authors in [24] define that an extensional mapping can be expressed as Local-As-View (LAV), Global-As-View

(GAV), Source-To-Target Tuple Generating Dependencies (s-t tgds), Second-Order Tuple Generating Dependencies (SO tgds), or other similar formalisms. More intuitively, a hybrid approach of the LAV and GAV mappings, termed as Global-and-Local-As-View (GLAV) mappings, which has been formalized to merit on the strengths of both mappings present a better mapping model for integration.

The final form of model management operation in our line of study is the *merge* operation, expressed as schema merging. Schema merging operation takes 2 meta-models and a set of mapping models, as inputs, and produces a merged meta-model, as an output, capable of representing all the elements and semantics of the input meta-models. In the generic sense, studies have been conducted and various results are addressed in the literature [2] [5] [25]. In the area of data warehousing, work done in the literature are presented in [25] [26] [27]. Additionally, the authors in [28] attempted to derive results on schema merging in relation to relational data sources, whiles merging based on semantic mappings have also been studied by the authors in [29]. A typical architecture of a merge system, as denoted in [27], is described in terms of 2 types of modules: wrappers and mediators. In terms of algorithms for merging, a generic approach was attempted in [30], whilst a proposition of an algorithm for relational sources that succeeds on a Mediated Schema Normal Form (MSNF) and Conjunctive Queries and Mappings is investigated in [28]. As part of our study, we draw on the significant propositions of generic merge in [2] and use them as background work in formalizing our algorithm in a much more practical solution for multidimensional data models.

The study of data integration and schema merging in relation to multidimensional data models has received minimal research in the literature. Cabibbo and Torlone [31] [32] [33] in their series of studies on dimension compatibility and data integration have attempted to deliver methodologies for fact and dimension tables and/or attributes integration. Riazati et al. [34] have also formalized a proposition for integration based on inferred aggregations in the hierarchies of the dimension tables, in each of the data marts. We expatiate on these approaches and perform a comparison of their work in line with our methodology in Section IX.

III. 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. The adopted star schema presents a modelling construct, where one large central (fact) table is referentially connected by a set of attendant (dimension) tables of varied attribute information. The fact table contains bulk data, without redundancy, whilst each dimension table contains multiple representations of attribute data instances.

We present an overview of our integration methodology, as depicted in Figure 1. The figure shows the logical and conceptual merging of the fact and dimension tables from the *Policy* and *Claims* data marts, to form an enterprise data warehouse for an Insurance industry. 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 integrating data marts.* ■

A. Problem Definition

In addressing our problem, we make reference to the scenario in Example 1, where we have 2 or more data marts, supposedly, in star schema models. It can be inferred that though the instance schema and data values representations in these separate data marts are different, the overlapping sets of real-world entity representations in the dimensions of the data marts seem to present a similarity. Hence, a proposition of integration for the real-world entities in each of the data marts into a single consolidated data warehouse is not improbable.

In another instance, the need to contract a merger or acquisition of companies of related business processes results in the generation of a consolidated data warehouse. This challenge in the generation of a data warehouse also falls in the paradigm of this study where each of the companies with disparate data marts or data warehouses are integrated into a single enterprise repository.

B. Overview of Integration Methodology

We outline our methodology based on 3 main streamlined procedures, namely; hybrid schema matching, mapping models discovery, and the formulation of merge algorithm. Figure 2 illustrates a description of our methodology and framework architecture in a workflow order. Here, we describe the step-wise procedures, algorithm executions, and the generated outputs. We describe in detail Hybrid Schema Matching (Procedure 1) in Section IV and Mapping Models Discovery (Procedure 2) in Section V. We also give a detailed description of the Formulated Merge Algorithm (Procedure 3) in Section VI. We address the methodology workflow in a manner where the results or output from a preceding step, e.g., Procedure 1, becomes the input for the succeeding procedure, e.g., Procedure 2. This approach ensures consistency in data processing and in the generation of the final integrated output of a data warehouse.

IV. HYBRID SCHEMA MATCHING

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 properties of the multidimensional schema structure in schema-based matching, and the instance data and extensions in instance-based matching, to find attribute correspondences.

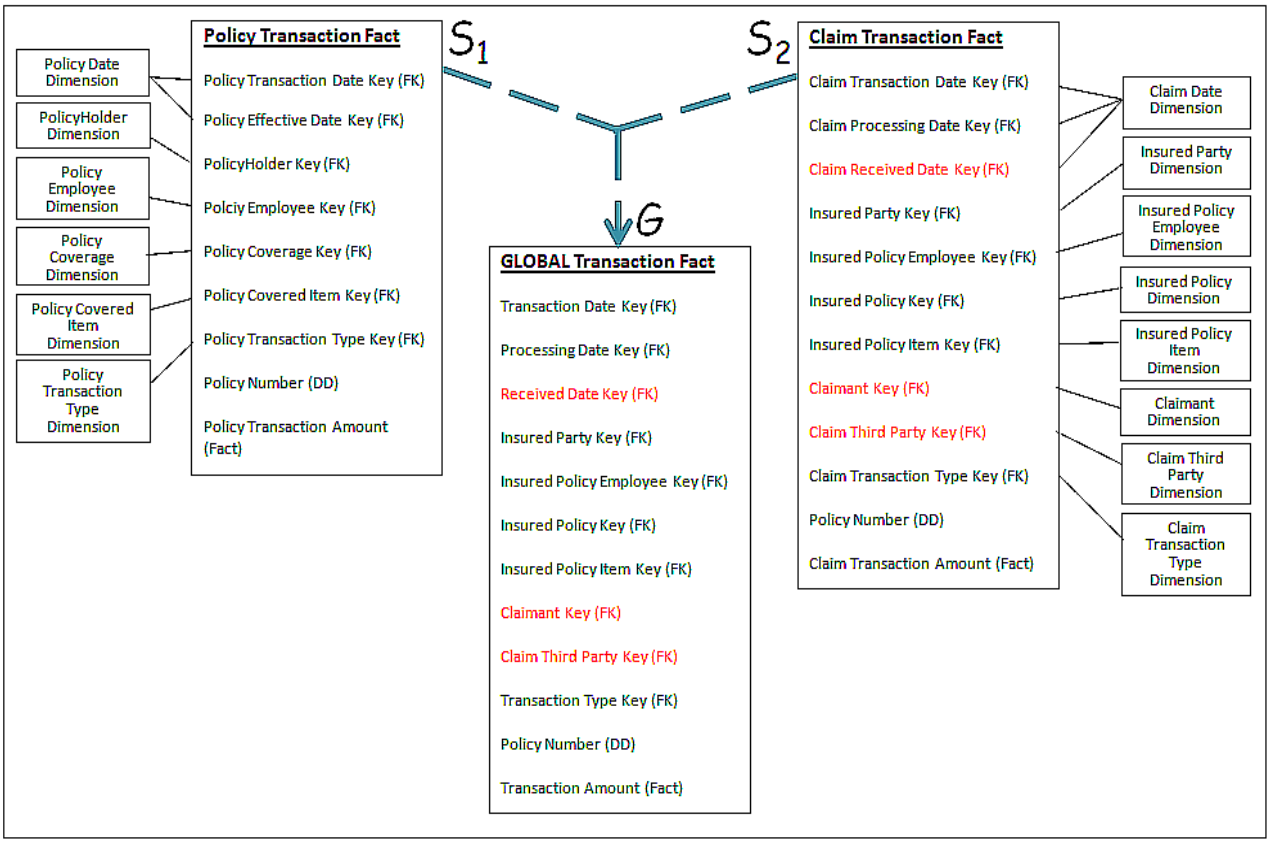

Figure 1. Logical and Conceptual Multidimensional Schema Merge

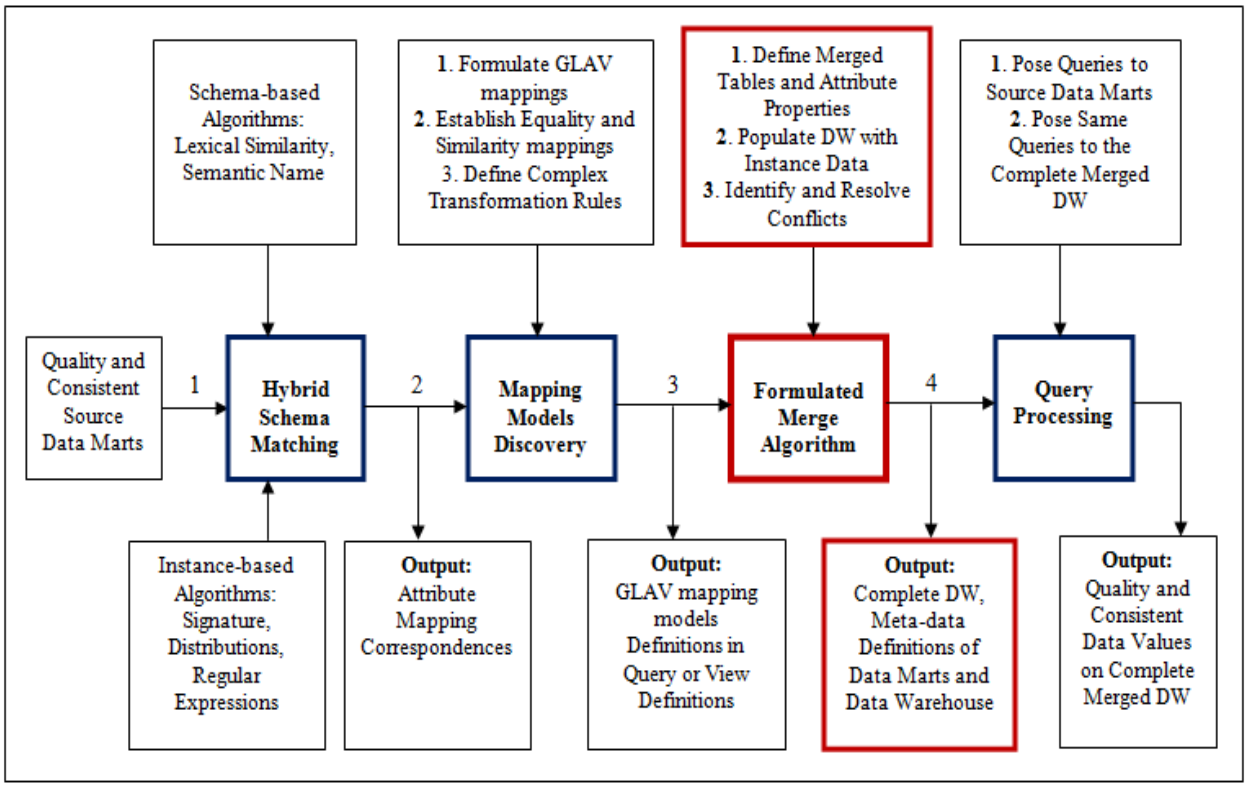

Figure 2. Workflow Framework of Integration Methodology

A. Schema-based Matching

We adopted schema-based algorithmic techniques in the form of Lexical Similarity and Semantic Names.

The Lexical Similarity is an algorithm technique based on the linguistic form of schema matching, in which string *names* and *text* are used to semantically find similar schema elements. This algorithmic technique defines a measure of the degree to which the word sets of 2 given strings are similar, and discovers maximum weight subsequence of the strings that are common to each other. The algorithm determines similarity based on schema string names and text, equality of names, equality of synonyms, homonyms, abbreviations, and similarity of common substrings, amongst others [35].

The Semantic Names, on the other hand, is an algorithmic technique based on the semantic deduction of the schemas and their characteristics. The algorithmic technique is reliant on the schema structure and the properties of the elements, and enforces on varied forms of constraints. It uses criteria such as, type similarity and metadata in relation to table name, attribute names, schema data types, value ranges, precision, uniqueness, optionality, relationship types, cardinalities, key properties, referential constraints, amongst others, to match attributes [35].

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. $DimInsuredParty.InsuredPartyKey \approx DimPolicyHolder.PolicyHolderKey$

2. $DimInsuredParty.FullName \approx DimPolicyHolder.FamilyName, DimPolicyHolder.GivenName, DimPolicyHolder.CityName, DimPolicyHolder.DistrictName$

3. $DimInsuredParty.StreetAddress, DimInsuredParty.EmailAddress \approx DimPolicyHolder.Address$

Moreover, the application of the Semantic Names algorithm will offer an improved schema matching using the data types, relationships types and constraints, and value ranges. This algorithmic matching enforces on the already generated correspondence in the Matching (1), where ***Int*** *data types and* ***Primary Key*** *constraints for both attributes of* ***DimInsuredParty.InsuredPartyKey*** *and* ***DimPolicyHolder.PolicyHolderKey*** *are used for element relationship mapping. For Matching (2), the* ***DimPolicyHolder.CityName [varchar(18)]*** *and* ***DimPolicyHolder.DistrictName [varchar(15)]*** *attributes were eliminated to deliver mapping correspondence, as in:*

2. $DimInsuredParty.FullName\ [varchar(60)] \approx DimPolicyHolder.FamilyName\ [varchar(25)], DimPolicyHolder.GivenNames\ [varchar(40)]$

The above mapping correspondence is generated as a result of the semantic representations of data type and precision, such as ***varchar(60)*** *for* ***DimInsuredParty.FullName*** *to correspondingly infer on* ***varchar(25)***, ***varchar(40)*** *for both* ***DimPolicyHolder.FamilyName*** *and* ***DimPolicyHolder.GivenName****, respectively.* ■

B. Instance-based Matching

The instance-based algorithms that were adopted are Signature, Distributions, and Regular Expressions. These algorithmic techniques are based on the instance data contained in the schemas and infer on the characteristics, meaning and similarity in the data values, as well as the relationship to other data set contained in the schema. The Signature algorithm uses the similarity in the actual data values contained in the schemas and their signature based on data sampling. The technique uses sampled data to find relationships where a weighting value is assigned to certain classes of words in the data [35]. This sampling of data is based on the valid values of sampling size and also the rate of the sampling. The determination of match signature is done by clustering according to their distance measure, either by Euclidean distance [36] or Manhattan distance [37].

The Distributions algorithm discovers mapping correspondences based on the common values in the instance data contained in the schemas. The algorithm also uses data sampling to aid the discovery function to find relationships between attribute data values, where the frequent occurrence of most data values for a particular attribute in relation to another attribute determines the candidacy of matching correspondence. Prior attempts of methodologies within the domain of machine learning that aid in the discovery of correspondences are A-priori and Laplacian [38].

The Regular Expressions algorithm uses textual or string searches based on regular string expressions or pattern matching. A simple regular expression will be an exact character match of attribute data values or of the common substrings contained in the instance data. This algorithm also uses data sampling to aid the discovery function of finding relationships between attribute data values [39].

We use Example 3 to illustrate a generalized form of instance-based algorithm.

Example 3. *Following up on Example 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 Matching (3), as in:*

3. $DimInsuredParty.StreetAddress \approx DimPolicyHolder.Address$

This final matching was attained because of the data values and extensions from the dimension attributes. A representation of the instance data values contained in ***DimPolicyHolder.Address*** *are {39 Baywood Drive, 178 Flora Ave., 79 Golden Rain St.}, where as data values contained in* ***DimInsuredParty.StreetAddress*** *and* ***DimInsuredParty.EmailAddress*** *are {40 Roslyn St., 68 Hastings Drive, 48 Whitehall Avenue} and*

{amartens@cybserv.com, drice@vipe2k.com, jtausig@fitexes.com}, respectively. ■

In general, the output generated from this step of Procedure 1, is a set of mapping correspondences between the elements of the instance schema structure and instance data values of the heterogeneous data sources.

V. MAPPING MODELS DISCOVERY

In the second procedure of our integration methodology, we discuss the adoption of first-order GLAV mapping models. We also discuss the merits of the mapping model, whilst highlighting the suitability for our integration approach. Moreover, we discuss the handling of possible functional dependency integrity constraints as they occur in the mapping models. This step utilizes the output of Procedure 1, as inputs, to aid in the discovery and establishment of mapping models.

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 $\{a_1, a_2, \dots, a_n\}$. 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 adopt a first-order GLAV mapping model formalism [40]. This mapping formalism is based on first-order logic assertions, where elements are finitely mapped using the functional relations existing between them. Our motivation is founded on the expressiveness of the correspondences that exist between the attributes of the schemas [3]. The GLAV mapping model combines mapping formalisms from both the Local-As-View (LAV) and Global-As-View (GAV) mappings [40]. This mapping model expresses mapping views where the extensions of the source schemas provide 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 [3]. One other unique feature of the GLAV mapping modelling is the expression of multi-cardinality mappings between mapping elements. This enables the expression of complex transformation formulas, which is much useful in our integration methodology [24].

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 $y = \text{z}$. ■

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 procedural step, 2 forms of mapping relationships were adopted, namely; *Equality* and *Similarity* mapping relationships. An equality mapping represents a one-to-one mapping, whilst a similarity mapping also represents a one-to-many or many-to-many mapping. The defined classifications were based on expressive characterization of relationship cardinality, and the attribute semantic representation, amongst others [39]. We used these forms of mapping relationships in a GLAV mapping model, as explained in Example 4.

Example 4. *Using the scenario described in 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:*

DimInsuredPolicyHolder *(InsuredPolicyHolderKey, InsuredPolicyHolderID, InsuredPolicyHolderName, BirthDate, ProvinceState, Region, City, Status):-*
DimPolicyHolder *(PolicyHolderKey, PolicyHolderID, PolicyHolderFamilyName, PolicyHolderGivenName, DateOfBirth, ProvinceState, CityName, Status),*
DimInsuredParty *(InsuredPartyKey, InsuredPartyID, InsuredPartyFullName, BirthDate, Province, Region, City)*

In this Datalog query, the existence of corresponding attributes in both dimensions automatically expresses an equality representation in the merged dimension. Additionally, a similarity relationship is established where, for example, ***DimPolicyHolder.InsuredFamilyName*** *and* ***DimPolicyHolder.InsuredGivenName*** *attributes are mapped onto the merge attribute of* ***DimInsuredPolicyHolder.InsuredPolicyHolderName****. Moreover, local attributes of* ***DimPolicyHolder.Status*** *and* ***DimInsuredParty.Region*** *from Policy and Claims data marts, respectively, are also included in the merged dimension schema instance.* ■

A. Propositions for GLAV Mapping Models

We further summarize a number of the characteristic features that merit the choice of the GLAV mapping model. This mapping model represents a suitable form of manipulation of the mapping relationships that exists between the instance schema attributes, as well as the instance data values, contained in the star schema data marts. Moreover, the GLAV mapping features offer the relationship needed for the generic application of the merge algorithm for disparate and heterogeneous schema and data instances.

Automatic Mapping Generation. It is a mapping formalism that facilitates the (semi-)automatic generation of schema mappings from heterogeneous instance schemas. This is evident in cases where mapping correspondences are incomplete or incorrect. This characteristic feature also offers the ability to incrementally modify mappings as correspondences change.

Mapping Reusability. The mapping model facilitates the composition of sequential mappings that enables the re-use of mappings when the instance schemas are different or change. This functionality offers the capability to

reformulate queries against one schema into queries on another schema during data integration.

Data Translation & Exchange. The semantics of such a mapping and its data exchange capabilities offers a data translation from one schema to another. Moreover, the mapping offers the transformation from one representation to the other during data exchange based on specifications.

Runtime Functionality. The mapping formalism expresses the capabilities for runtime executables; for example, to generate view definitions, query answering, and generation of XSLT transformations, amongst others.

Data Manipulation. The semantics of the GLAV mapping model makes it easily applicable and manipulated by mapping tools; for example, the IBM InfoSphere Data Architect [41], Microsoft BizTalk Mapper [42], amongst others.

Query Code Generation. The mapping formalism offers a platform where query codes are generated based on the mapping relationships. This is evident where efficient queries or transformations in various languages (e.g., native SQL) can implement the formulated mappings.

B. Functional Dependency Mapping Integrity Constraints

In our methodology, the adopted first-order GLAV mapping modelling can be enhanced to deliver efficient relationships between the attributes of the schema instances, by the application of mapping integrity constraints. One form of mapping integrity constraint is *Functional Dependencies* of the dimension instance schema attributes.

Definition 4. (Functional Dependency): Suppose $\mathcal{D} = (\mathcal{A}_1, \dots, \mathcal{A}_n)$, for $n \geq 2$ attributes represent a dimension instance schema. The assertion of functional dependency, $\mathcal{FD}$ constraint stipulates that a set of attributes $\{\mathcal{A}_1, \dots, \mathcal{A}_u\} \in \mathcal{D}$ uniquely determines another set of attributes $\{\mathcal{A}_{u+1}, \dots, \mathcal{A}_n\} \in \mathcal{D}$, based on a key constraint, say $\mathcal{A}_1$, such that $\mathcal{FD}$: $\{\forall\mathcal{A}_1\forall\mathcal{A}_u\forall\mathcal{A}_{u+1}(\mathcal{D}(\mathcal{A}_1, \mathcal{A}_u) \wedge \mathcal{D}(\mathcal{A}_1, \mathcal{A}_{u+1}) \rightarrow (\mathcal{A}_u = \mathcal{A}_{u+1}))\}$. ■

From the definition above, it can be inferred that the set of attributes $\{\mathcal{A}_1, \dots, \mathcal{A}_u\} \rightarrow \{\mathcal{A}_{u+1}, \dots, \mathcal{A}_n\}$ uniquely, where the data instance tuple values in attribute set, $\{\mathcal{A}_{u+1}, \dots, \mathcal{A}_n\}$ are dependent on, or can be derived from the tuple values in attribute set, $\{\mathcal{A}_1, \dots, \mathcal{A}_u\}$.

We use Example 5 below to illustrate the occurrence of functional dependency integrity constraint, as part of the mapping discovery and modelling.

Example 5. *Suppose an integrity constraint exists on the instance schema dimension* ***DimPolicyCoveredItem*** *in the Policy data mart where the attribute* ***DimPolicyCoveredItem.PolicyCoveredItemID*** *functionally determines the set of attributes* ***DimPolicyCoveredItem.PolicyCoveredItemType*** *and* ***DimPolicyCoveredItem.CoveringPeriod****. Suppose a principal mapping correspondence is established between the Natural Key attributes of* ***DimPolicyCoveredItem.PolicyCoveredItemID*** *and* ***DimInsuredPolicyItem.InsuredPolicyItemID.*** *Moreover, suppose mapping correspondences are established between attributes* ***DimPolicyCoveredItem.PolicyCoveredItemType****,* ***DimPolicyCoveredItem.CoveringPeriod*** *and* ***DimInsuredPolicyItem.ItemForm,*** ***DimInsuredPolicyItem.PolicyPeriod,*** *respectively. The modelling of first-order GLAV mappings between the dimensions* ***DimPolicyCoveredItem*** *and the* ***DimInsuredPolicyItem*** *will result in an automatic instance functional dependency constraint in* ***DimInsuredPolicyItem*** *dimension. This dependency is expressed in the set of attributes* ***DimInsuredPolicyItem.ItemForm*** *and* ***DimInsuredPolicyItem.PolicyPeriod,*** *to functionally depend on the* ***DimInsuredPolicyItem.InsuredPolicyItemID*** *attribute. This dependency association is modelled in the merged table and its attributes for each of the integrating table schema instances.* ■

It will be affirmed that the dependency association between attributes complements the derivation of the merge schema and data instances. Moreover, the dependency constraint enables the population of data instance tuple values, especially in the Steps (10) and (11) in the merge algorithm (Algorithm 1) in Section VI.A. This can be addressed in the scenario where, if the tuple values for the set of $\mathcal{A}_u$ attributes are known, say a_u, then the tuple values for the set of $\mathcal{A}_{u+1}$ attributes, say $a_{u+1,}$corresponding to and depending on a_u can be determined by looking them up in tuple values of the a_u.

The output generated from Procedure 2 step, is a set of mapping models outlining the types of *Equality* and *Similarity* mapping relationships. The output expresses the merge schema definitions, schema constraints, and complex transformations for the one-to-many and/or many-to-many relationships of the heterogeneous data source elements.

VI. FORMULATED MERGE ALGORITHM

We present and describe an elaborate merge algorithm (Algorithm 1) for integrating the instance schema and data of data marts fact and dimension tables. We further provide a summary of the algorithm, and conclude the section by presenting a computational complexity analysis of the formulated algorithm.

A. Merge Algorithm

The merge algorithm (Algorithm 1) is formulated to generate the single consolidated data warehouse from different related data marts, modelled as star schemas instances.

B. Merge Algorithm Summary

The merge 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). Steps (1) to (4) initialize and generate the integrated schema tables. Steps (5) to (7) describe the generation of attributes for the integrated tables.

Algorithm 1: Multidimensional Instance Schema and Data Integration

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*** $mergeDataWarehouse$

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 values 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.

C. Merge Algorithm Computational Complexity

The merge algorithm presented in previous section, Sub-section A, is projected to run with a low worst-case complexity of a polynomial time.

The Initialization step in Step (1) requires a complexity of $O(n)$, whiles the Step (2) takes $O(n^2 \log m)$ to derive a merged fact table and dimension tables, for n number of tables and m number of corresponding attributes.

The iterative processes of Step (5) and Step (6) involves a computation running time of $O(k + n^2 \log m)$ to generate the table attributes and their relationships, for n number of tables, m number of corresponding attributes, and k number of non-corresponding attributes.

Finally, the steps from Step (8) to Step (12) require running time of $O(k + m)$ for the iterations performed. An overall worst-case complexity of $O(n) + O(k + m) + O(k + n^2 \log m)$ is attained in running the merge algorithm to generate the single consolidated data warehouse.

We also give a detailed Proof of Correctness of the merge algorithm in Appendix XI.

VII. PROPOSITIONS OF MULTIDIMENSIONAL INSTANCE SCHEMA AND DATA INTEGRATION

In this section, we propose technical qualitative requirements necessary for producing an efficient single consolidated data warehouse. We also describe the semantics of query processing on integrated instances of multidimensional data models. We finally propose and 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 proposed requirements, to ensure the correctness of the data values from the queries that would be posed to it.

Drawing on the propositions in the requirements defined by the authors in [1] for merging generic meta-models, we performed a gap analysis and extend 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. We outline the set of *Merge Correctness Requirements* (MCR) that validates the formulated merge algorithm needed for the generation of a single consolidated 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, attributes in each of the dimension tables are represented through an equality or similarity mapping. Finally, an automatic inclusion for non-corresponding attributes in the merged tables, based on the condition of non-attribute redundancy or duplication, is satisfied.

Slowly Changing Dimension Preservation (SCD). SCD is the occurrence where an entity in a dimension exhibits multiple instance representations, based on the varied changes in instance data values for the key dimensional attributes, over a time period [43]. For such dimensional entity occurrences, the merged dimension should offer an inclusion of all the instance data representations from each integrating dimension. Hence, an automatic inclusion of attributes that contribute to the dimensional change in the merge dimension is satisfied.

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 attributes. For a similarity mapping, the transformation expression should have the properties to be able to satisfy the attribute property value of each integrating attribute.

Definition 5. (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. A Surrogate Key is the dimensional attribute that uniquely identifies each instance data value tuple of an entity representation. The single consolidated data warehouse should offer the containment of all unique tuples from the data marts for correctness in query answering. This ensures the preservation of all *Surrogate Keys* needed in identifying each dimensional entity.

B. Merge Algorithm Technical Requirements Summary

The integration methodology adopted by the authors largely satisfies the technical requirements, as a proposition for merging disparate data marts. We summarize the merge algorithm (Algorithm 1) in fulfilment of the technical *Merge Correctness Requirements* (MCRs) outlined in Section VI.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. Semantics of Query Processing on Integrated Instances of Multidimensional Data Models

The type of queries that are processed on multidimensional data models are based on Online-Analytical Processing (OLAP). There are a few problems that are inherent with OLAP query processing, and these are addressed as follows. On one hand, is the problem of incomplete data that arise from missing data values, and imprecise data values of varying extent. In our approach, the possibility of having missing data values, in relation to non-corresponding merge attribute, from the star schemas is highly probable. Moreover, the varying granularities caused by the different degrees of precision in the data values from the combined instance data of different star schemas, exposes a non-uniform representation of the data values needed for analytical reporting.

Definition 6. (Dimension Hierarchy): A hierarchy, $\mathcal{H}$ comprising a dimension, $\mathcal{D}$, is a 2-tuple $(\mathcal{L}_n, \nearrow)$ where $\mathcal{L}_n$ is a collection of levels and each $\mathcal{L}_i \in \{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_n\}$, $i \leq n$, and $\nearrow$ is a parent-child relation of two levels, say $\mathcal{L}_i$ and $\mathcal{L}_j$, such that a data instance element in $\mathcal{L}_i$ rolls up to a data instance element in $\mathcal{L}_j$, denoted by $(\mathcal{L}_i \nearrow \mathcal{L}_j)$. This roll-up relationship forms a partial order over the levels. ■

Definition 7. (Strict Hierarchy): For a dimension schema instance $\mathcal{D}$, any hierarchy $\mathcal{H} \in \mathcal{D}$, is said to be strict if for every pair of levels $\mathcal{L}_i, \mathcal{L}_\mathrm{j}$ with the partial ordering $(\mathcal{L}_i \nearrow * \mathcal{L}_j)$, which are through different paths, say $\{\mathcal{L}_\mathrm{i}, \mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_\mathrm{u}, \mathcal{L}_j\}$ and $\{\mathcal{L}_\mathrm{i}, \mathcal{L}_\mathrm{a}, \mathcal{L}_\mathrm{b}, \dots, \mathcal{L}_\mathrm{v}, \mathcal{L}_j\}$, and for every instance data element e in $\mathcal{L}_i$, there exist a roll-up function composition that holds for the condition: $\int_{\mathcal{L}_i}^{\mathcal{L}_1} \circ \int_{\mathcal{L}_1}^{\mathcal{L}_2} \circ \dots \circ \int_{\mathcal{L}_u}^{\mathcal{L}_j}(e) = \int_{\mathcal{L}_i}^{\mathcal{L}_a} \circ \int_{\mathcal{L}_a}^{\mathcal{L}_b} \circ \dots \circ \int_{\mathcal{L}_v}^{\mathcal{L}_j}(e)$ ■

On the other hand, the problem of imperfections inherent in the hierarchy levels of dimensional tables also places an overhead impact on query processing for multidimensional data models. Hierarchies enable drill-down and roll-up in the aggregate data, and as a result, multiple hierarchies in a particular dimensional entity support different aggregation paths within the dimension. Different forms of strict and non-strict hierarchies are exhibited in the dimensional entities of multidimensional data models. Strict hierarchies exhibit a phenomenon where a dimension data instance item or child level element has only one parent level element enforcing a constraint restriction on the data values that are rolled-up during aggregation. Hence, strictness in hierarchies ensures a consistency in the instance data values that are used in roll-up functions. Non-strict hierarchies exhibit a phenomenon where a dimension data instance item or child level element has several elements at the parent levels, thus allowing flexibility in the data aggregation.

Pedersen et al. [44] proposed requirements that a multidimensional data model should satisfy in order to fully support OLAP queries. These are outlined as; explicit hierarchies in dimensions, multiple hierarchies, non-strict hierarchies, handling different levels of granularity, and handling imprecision amongst others. These requirements give insights into how OLAP tools manage the raw data values and how data values are expressed during analytics.

As part of our study, query processing is handled in relation to the proposition in [44]. The adopted star schema model offers a platform for basic SQL star-join optimization during the processing of data values for analytical representation. The ability of structured cube modelling for each of the dimension elements by OLAP representations offers the medium for the individual hierarchies in the dimensional entities to be captured explicitly. The hierarchies and their data manipulations are captured using either, grouping relations and functions, dimension merging functions, roll-up functions, level lattices, hierarchy schemas and instances, or an explicit tree-structured hierarchy as part of the cube.

Different forms of aggregations are computed in the approach of query processing on the generated data warehouses. These aggregations are made possible because of the defined hierarchies established in the dimensional entities. The aggregations are represented in functions such as addition computations, average calculations, and constant functions through an OLAP operation of summarizability.

Definition 8. (Summarizability): A hierarchy $\mathcal{H}$ is summarizable if for all levels $\mathcal{L}_i \in \{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_n\}$, for $1 \leq i \leq n$, of this hierarchy, the single-level aggregated measure m with $\mathcal{L}_i$ granularity, can be computed by summing up data instance tuple values of a single-level specified for measure m for any $\mathcal{L}_k \in \{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_n\}$, for $1 \leq k \leq n$, granularity appearing along a path from the bottom level to $\mathcal{L}_i$. ■

Summarizability is a conceptual property of multidimensional data models where individual aggregate results can be combined directly to produce new aggregate results. In a summarizable hierarchy, the aggregated values for a measure at a level granularity can be obtained by aggregating the elements of any level of hierarchy, which directly or indirectly rolls up to the desired level. This characteristic feature guarantees the correctness of aggregated values in the resultant data warehouse.

D. Conflicts Identification and Resolution

The integration of meta-data models is generally coupled with different forms of conflicts in either the instance schema or instance data. These conflicts are resolved through different propositions from the formulated 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 likely to occur conflicts, which are frequently 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 6. *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 an instance data value, Employee P from* ***DimPolicyEmployee*** *and an instance data value, 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-assigning 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. A proposed resolution measure, outlined in the merged algorithm, will be to perform a de-duplication of the conflicting entities. This is achieved 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. In reference to Example 6, in integrating dimensions *DimPolicyEmployee and DimInsuredPolicyEmployee*, a merged attribute for *DimPolicyEmployee.HireStatus* and *DimInsuredPolicyEmployee.EmployeeStatus* attributes 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 *DimPolicyEmployee.HireStatus* and *DimInsuredPolicyEmployee.EmployeeStatus*, respectively. We resolve these conflicts by using the attribute data types as defined in the mapping model.

VIII. 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, and we discuss and analyze the evaluation results.

A. Implementation

We describe our implementation framework of various techniques and processes needed in producing the output of a single consolidated data warehouse. This sub-section focuses on the experimental setup, the datasets used in the experiments, as well as the practical procedures we performed based, on the proposed integration methodology addressed in Sections III, IV, V and VI.

Experimental Setup and Data Sets. We implemented our methodology using 2 different data warehouses, from Insurance and Transportation data sets. The Insurance data consisted of 2 data marts. These were Policy and Claims data marts. Their schema structure and instance data are described. The Policy and the Claims data marts contained 7 and 10 Dimension Table schemas, respectively. These dimensions were referentially connected to a single Fact Table schema. Each fact table schema had a Degenerate Dimension (DD) attribute of a Policy Number and a fact or measure attribute of Policy Transaction Amount. The Policy fact table schema contained instance data of 3,070 tuples of data, whilst the Claims fact contained 1,144 tuples of data. Both data sets had 6 corresponding entity representation in the dimension tables, whilst the Claims data mart had 3 other non-corresponding dimensions.

The Transport data set, on the other hand, contained 3 data marts. These were Frequent Flyer, Hotel Stays, and Car Rental data marts. All the data marts had 3 conformed dimensions; namely, Customer, Date, and Sales Channel. These dimensions were complemented with a number of non-corresponding and unique dimensions in each of the data marts. Their Fact Tables contained 7257, 2449, 2449 tuples of data for Frequent Flyer, Hotel Stays, and Car Rental, respectively. Each of the Fact Tables also contained 7, 6, and 5 facts or measures for Frequent Flyer, Hotel Stays, and Car Rental, respectively. All the source data marts had their permanent repository stored in Microsoft SQL Server DBMS [45]. Each entity representation in the dimensions was identified by unique surrogate key and based on clustered indexing.

Hybrid Schema Matching. The schema matching and mapping models discovery procedural steps were implemented using IBM Infosphere Data Architect [22] [23] [41]. 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 algorithmic techniques incorporated in the application software. The schema-based algorithmic techniques that were adopted are Lexical Similarity and Semantic Names, where as the instance-based algorithmic techniques were Signature, Distributions and Regular Expressions. 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.

Figure 3 illustrates the derivation of semantically correct matching candidates to establish mapping correspondences between the attributes of *DimPolicyTransactionType.PolicyTransactionTypeKey*, *DimPolicyTransactionType.PolicyTransactionId*, and *DimPolicyTransactionType.TransactionCodeName* of *DimPolicyTransactionType* dimension schema to the *DimClaimTransactionType.ClaimTransactionCode* attribute of *DimClaimTransactionType* dimension schema. In Figure 3, the blue-coloured mapping correspondences represent the chosen semantically correct matching candidate, where *DimPolicyTransactionType.PolicyTransactionId* attribute corresponds to the *DimClainTransactionType.ClaimTransactionCode* attribute. On the other hand, the red-coloured mappings represent the semantically incorrect matching candidates of *DimPolicyTransactionType.PolicyTransactionTypeKey* and *DimPolicyTransactionType.TransactionCodeName*, which are ignored as part of user validation by inspection. Moreover, the yellow-coloured mappings represent the correspondences that were generated for each of the dimensions, as a result of the application of the schema matching algorithms.

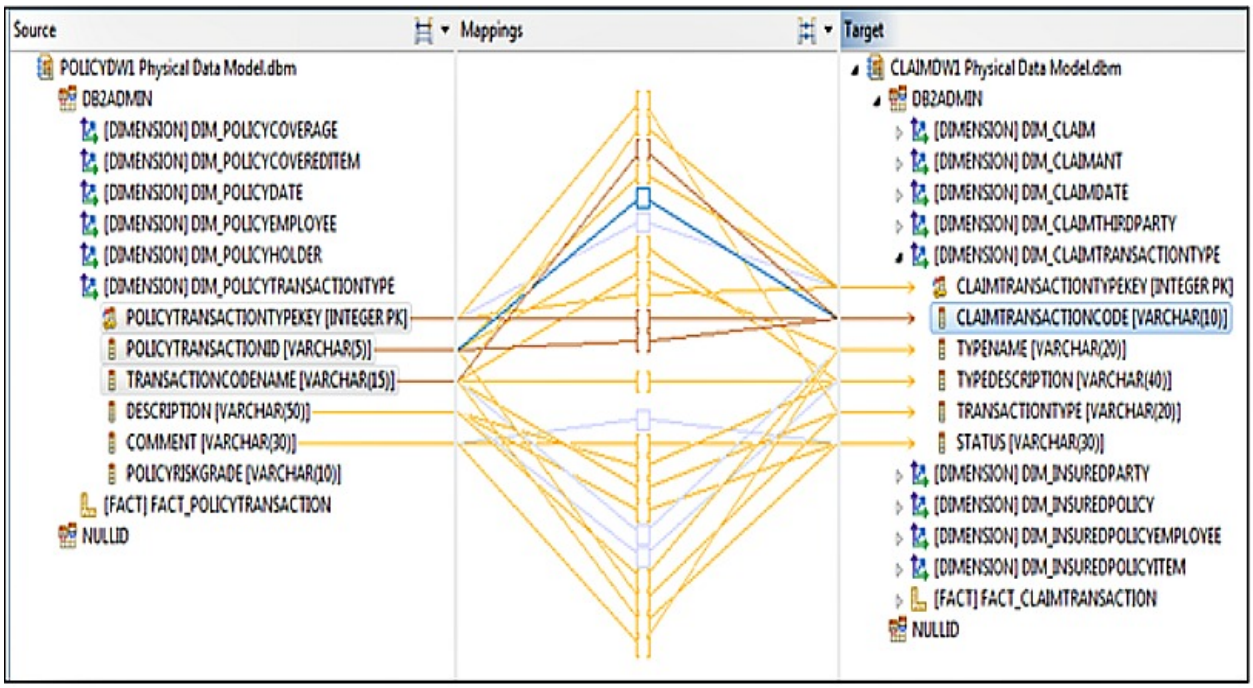

Figure 3. Hybrid Schema Matching

TABLE I. SUMMARY OF PARAMETIZED CONFIGURATIONS FOR SCHEMA MATCHING ALGORITHMS

Matching Algorithm/ Configuration Option	Rejection Threshold	Thesaurus Option	Sampling Size (Rows)	Sampling Rate (%)
1. Lexical Similarity	0.6	Not Applicable	Not Applicable	Not Applicable
2. Semantic Name	0.5	Is Applicable; But not configured	Not Applicable	Not Applicable
3. Signature	0.8	Not Applicable	150	30
4. Distributions	0.8	Not Applicable	100	20
5. Regular Expressions	0.9	Not Applicable	100	30

When generating mapping correspondences for the fact and dimension table attributes, various configuration manipulations of algorithms are performed on the discovery function. The parameters used in configuring the algorithms were Rejection Threshold, Thesaurus Option, Sampling Size, and Sample Rate. The Rejection Threshold parameter was configured with different adjustments for both the schema- and instance-based algorithms. The Thesaurus Option parameter was applicable to the Semantic Name algorithm. The Sampling Size and Sampling Rate parameters were applicable to the instance-based algorithms. We summarize the parameterized configuration of the algorithms adopted in TABLE I.

Mapping Models Discovery. In the mapping models discovery 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 illustration of multi-cardinality mapping relationship is displayed in Figure 4. In Figure 4, there is a mapping discovery and modelling between the attributes of *DimPolicyHolder* and *DimInsuredParty* dimensions. These mappings are indicated by the grey lines connecting attributes from *DimPolicyHolder* to *DimInsuredParty* dimensions. More specifically, a selected mapping relationship of the *DimInsuredParty.FullName* attribute is modelled onto 2 other attributes; namely, *DimInsuredParty.FamilyName* and *DimInsuredParty.GivenName*.

We therefore, defined a complex transformation expression, as in Equation (1), in the mapping relationship already established between these dimension attributes.

$$DimInsuredParty.FullName = DimPolicyHolder.FamilyName + DimPolicyHolder.GivenName \quad (1)$$

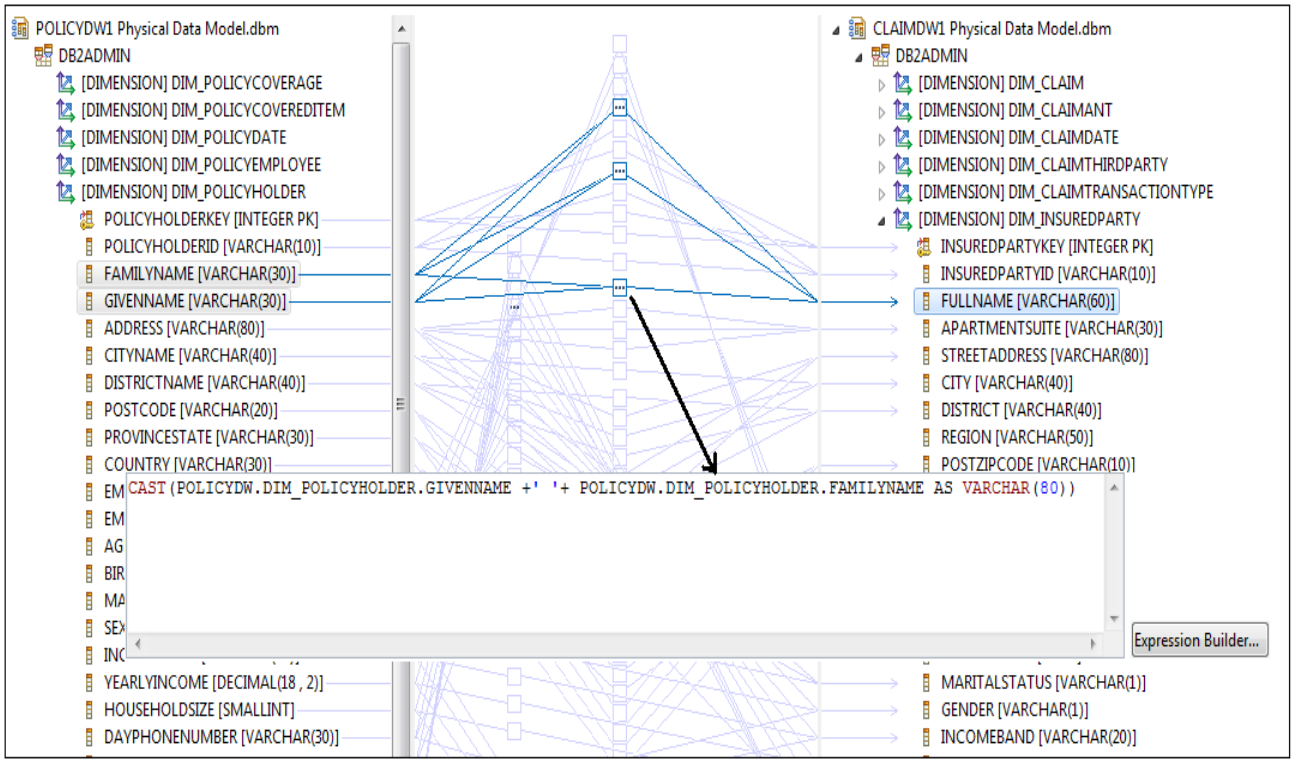

Figure 4. Mapping Models Discovery

Other forms of mapping properties that were defined in the modelling are expressive characterization of relationship cardinality, attribute semantic representation, and attribute data type representation, amongst others. In terms of the relationship cardinality, an equality or similarity mapping cardinality type was defined. To express the attribute semantic representation, a definition of the supposed merged attribute name was specified, where possible. Regarding attribute data type representation, a supposed merge data type was defined and this served as a union data type for the merging attributes. A procedural output in a *Comma Separated Values* (CSV) file 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

Formulated Merge Algorithm. The formulated merge algorithm was implemented with the availability of the mapping models and the source data marts as inputs. The implementation was programmed using Microsoft Visual C# .Net Integrated Development Environment (IDE) with 8029 lines of code from the Entity classes, Business Logic classes, Utility classes, and program control code. Stored procedures were implemented in the Microsoft SQL Server permanent repository, and these served as transaction processing medium between the data repository and the entity and business logic classes in the programming IDE.

Query Processing and Analyses. The analyses of the repository data, of both the integrating source data marts and the generated single consolidated data warehouse, were performed using IBM Cognos Business Intelligence [46] application software. The software enabled the possibility of processing queries on the instance data, in the form of report generation.

B. Evaluation

Our evaluation analyses were primarily based query processing on the single consolidated data warehouse in relation to the integrating data marts. We compared the outputs of the query processing from both the data marts and the generated data warehouse. We first ran a formulated query the data marts, and afterwards ran the same query on the generated data warehouse. Based on these processes, we are able to effectively compare the results from the data marts and the single consolidated data warehouse.

Evaluation Criteria and Analyses. We evaluate the outcome of the experiments performed based on a set of criteria from the guidelines proposed by Pedersen et al. [44]. We performed a gap analysis on their study and adapted 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* 0. 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.

For recall, an evaluation of 100% was trivially attained and verified. The verification was based on the assertion that the formulated 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 presents an 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 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 or corresponded 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 5 and 6 show the precision evaluation for Insurance and Transportation data warehouses, respectively. In Figure 5, an average rate of 86% was achieved for the queries posed to dimensions 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.

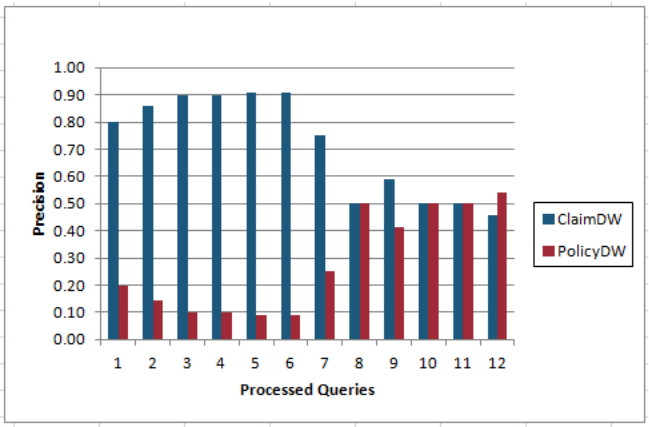

Figure 5. Precision for Insurance Data Set

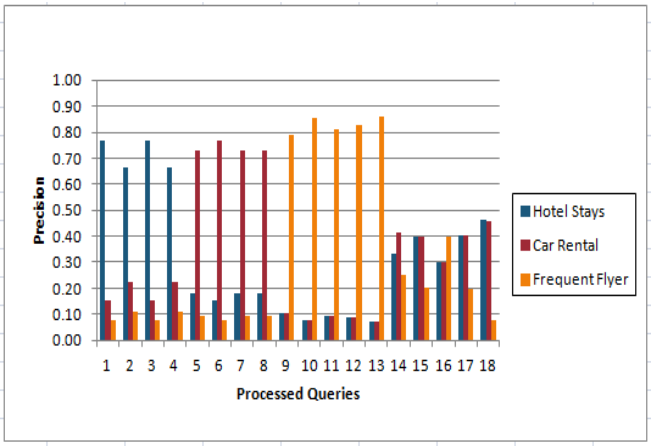

Figure 6. Precision for Transportation Data Set

In Figure 6, 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. Queries 1 and 3 for Hotel Stays, 6 for Car Rental, and 10, 12, 13 for Frequent Flyer data marts performed creditably well as a result of the higher containment of tuples to the attributes being retrieved for the formulated queries posed. Moreover, in terms of queries posed to corresponding dimensions, an average precision rate of 38%, 40%, and 23% was attained for Hotel Stays, Car Rental, and Frequent Flyer data marts, respectively. This is depicted in queries 14 to 18. It would be realized that this average rate for the Transportation data set is quite lower than that attained in respect to the Insurance data set. This is based on the claim that an increase in the number of data marts for integration is inversely proportional to the precision rate of queries for the respective data marts. This assertion is due to the distributive proportionality of tuples per each dimension of the corresponding data marts. Additionally, the attributes involved in the formulated query for these dimensions also enforces on this assertion.

In summary, the average precision rates analyzed are able to provide the user with details regarding the proportion of the data in the merged data warehouse that originate from a specific data 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.

Query Processing Rate. 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.

Our evaluation of the processed queries showed that the 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 and Transportation data sets are shown in Figure 7 and 8, respectively.

In Figures 7 and 8, it can be generally deduced from display that the data values that the query rate for the data warehouses were appreciable taking note of the compared values generated from the data marts. In certain cases, such as queries 7 and 8, in Figure 7, the rates were a bit higher due to higher level of aggregation and increased number dimension attributes involved in data values retrieved. Queries 6 and 11 recorded lower query rates because of the low quantity of attributes, as well as tuple data values, in the formulation of the answer to the query. Additionally, in Figure 8, similar observation was realized on queries 6, 14, and 16 where the query processing rate is a bit higher in comparison to the others. We also observed a lower rate of query rates for queries 4, 10, 13, and 19, which inferred a very good composition of merged tables and attributes and their contained data instance tuple values.

We further computed the variance of the average query rate per data mart as it differs quantitatively from the consolidated data warehouse. A deduction observation was ascertained, where a lower quantity of tuples of instance data values to be retrieved during query processing lead to an increase in the variance, and vice versa. This is due to the fact that an increase in the number of data marts, and resultant increase in data instance tuples, increases the rate of data retrieval, per data mart analysis in relation to the single consolidated data warehouse.

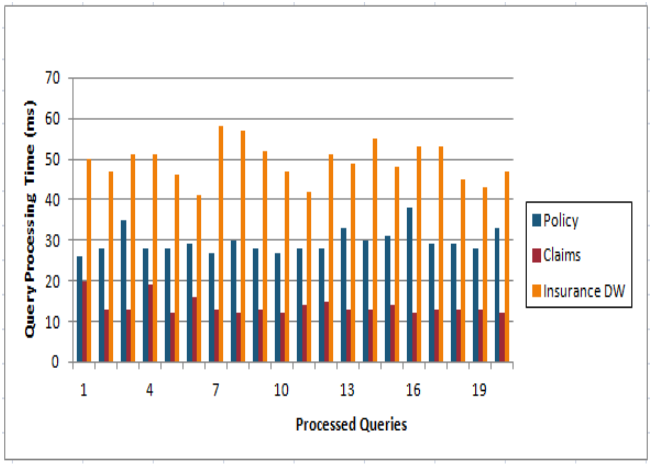

Figure 7. Query Processing Rate for Insurance Data Set

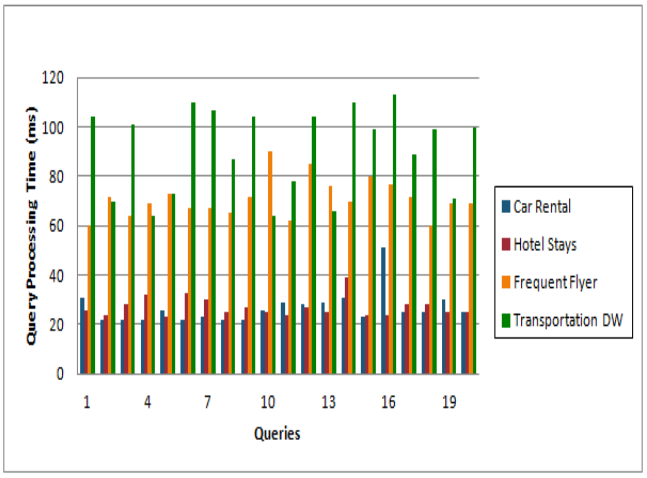

Figure 8. Query Processing Rate for Transportation Data Set

TABLE II. 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

An observation of the Claims data mart, in Figure 7 and TABLE II. reveals that the variance of 35.50 was higher because of the lower query rate of the integrating data mart. Moreover, in Figure 8 and TABLE II. the Hotel Stays and Car Rental data marts rather had a higher variance of 63.55 and 63.95, respectively, as their query rates were lower because of the lower quantity of data instance tuples.

We present a summary of the variances in the average query response time (in milliseconds) for the data marts in comparison to their respective data warehouses in TABLE II.

IX. COMPARISON TO OTHER APPROACHES

There have been minimal studies in this area of multidimensional data integration, in particular to the generation of a single consolidated data warehouse. These approaches present significant contributions with regards to element mappings and algorithms. In comparison, our approach addresses the integration problem from an important concept of model management. We discuss a number of these approaches and comparatively explain how our methodology performs better.

A. Dimension Compatibility and Heterogeneous Multidimensional Integration

Cabibbo and Torlone in their series of studies [31] [32] [33], address the problem of data integration in relation to multidimensional databases (data marts). In their work [31] [33], they introduce fundamental assertions of dimension algebra and dimension compatibility. Their work highlights different forms of heterogeneities that are existent in dimension tables. Their attempt to address these heterogeneities lead them to introduce a novel theoretical concept of dimension algebra. This concept enables the selection of relevant portions of a dimension for integration. The dimension algebra is basically based on 3 main operators; namely, selection, projection, aggregation.

The authors in [31] [33] also introduce the concept of dimension compatibility. Dimension compatibility outlines the retrieval of common dimension information based on the characterization of general properties. These general properties were outlined as; level equivalence, dimension equivalence, dimension comparability, and dimension intersection. The compatibility property of dimensions was then used as a platform to perform drill-across queries over the autonomous data marts, and aid in hierarchical aggregation of instance data. As part of their study, the authors [31] [32] [33] use the fundamental intuitions to propose 2 different approaches to the problem of integration of multidimensional databases; namely, loosely coupled integration and tightly coupled integration. They introduced concepts and algorithms, and stipulated a number of desirable properties for dimension matching; namely, coherence, soundness, and consistency.

B. Inferred Aggregation in Hierarchies

Riazati et al. [34] propose a solution for integration of data marts where they infer aggregations in the hierarchies of the dimension tables existent in the multidimensional databases. In their work, they attempted formulating a computation on minimal directed graph from the instance data. These inferred hierarchies are then used to perform roll-up relationships between levels and to ensure the summarizability of data. They further use the assertion of dimension compatibility introduced in [31] [32] [33] to develop algorithms, which in turn are used for the integration of data marts.

C. Methodology Comparisons & Evaluation

The existing approaches to multidimensional instance schema data integration addressed in [31] [32] [33] [34] explain important concepts that need to be discussed when incorporating several data marts into a single consolidated data warehouse. On the contrary, these techniques and methodologies are inadequately enough in the handling of more complex characteristics of the fact or dimension tables and their attributes. We address the shortcomings of these approaches, and highlight the enhanced ways of handling such issues through our methodology approach using the concept of model management.

Firstly, the approaches by the authors in [31] [32] [33] fail to address the issue of mapping models, although propositions of the general properties regarding the characterization of dimension compatibility seems to handle this concept. Our approach, however, adopts a first-order mapping modelling formalism, which better expresses the attribute correspondences. As a result, issues of data exchange and transformation for dissimilar and multicardinality attributes are expressed efficiently.

Secondly, the previous approaches do not lay out a precise schema merge algorithm. Descriptions of algorithms for deriving the common information between dimensions and for merging were put forward in [32] and other literatures so far. But these algorithms are inconclusive enough to solve the complex representations of schema and data instances. Our approach offers a complete formulated algorithm for integrating multidimensional data models based on star schema models.

Thirdly, conflict management relating to identification and resolution are not completely addressed by the authors in their approach. In the literature [33], the properties that underlie and establish the dimension compatibility criteria seem to partially solve the likely to occur conflicts that could be encountered in the dimensions. But these properties in their entirety fail to totally resolve such prominent conflicts during integration. Our methodology outlines a definite set of likely to occur conflicts and their resolution measures in relation to the instance schema and instance data values.

Fourthly, technical qualitative requirements, which serve to highlight the properties that a generic integrated schema should possess were addressed by the authors in [2][28]. A careful study of the specific approaches for multidimensional data integration attempted by the authors in [31] [32] [33] [34] seem not to have specified requirements for integration. A number of requirements were generally attempted by the authors in [32]. They proposed of coherence, soundness and consistency as measures for compatible dimension matching; but these are inconclusive in the larger scale of integrating schema and data instances. Our methodology approach proposes a complete set of requirements for multidimensional integration to handle the varied properties and constraints of multidimensional data models.

We present a comparative analysis and evaluation of the proposed methodology in line with other approaches in TABLE III. This tabular analysis summarizes the discussions regarding methodology approaches presented in the literature, and outlines the merits of our proposed methodology over the other approaches.

X. CONCLUSION

This paper presents a methodology for the merging of multidimensional data models using star schemas instances. We addressed extensively the methodologies and algorithms adopted in finding mapping correspondences between the elements attributes of the fact and dimension tables for the data marts.

TABLE III. QUALITATIVE ANALYSIS OF PROPOSED METHODOLOGY AND OTHER APPROACHES

Methodology Approach / Analysis Criteria	**(1) Proposed Integration Methodology**	**(2) *Cabibbo and Torlone*** [31] [32] [33] **- Dimension Compatibility and Heterogeneous Multidimensional Integration**	**(3) *Riazati et al.*** [34] **– Inferred Aggregation in Dimension Hierarchies**
Mapping Models Discovery and Modelling	Adopts a first-order GLAV mapping model, which offers effective data translation and data exchange functions	Introduces dimension compatibility for attribute mappings, but does not present complete mapping modelling and the handling of attribute relationships types	Methodology extends on the previous notions of dimension compatibility in (2); does not lay out precise mapping modelling
Formulated Merge Algorithm	Presents a complete merge algorithm that handles varied characteristics of both schema and data instances from heterogeneous data sources	Presents sets of algorithms that involves drill-across queries between dimensions instance schema attributes; but methods are inconclusive for varied properties of instances of schema attributes and data	Proposes algorithms for inferring partial order of attributes, and for identifying hierarchy levels and roll-ups. These algorithms are based on only schema instances
Conflict Identification and Resolution	Identifies likely to occur conflicts and proposes complete resolution measures in the element attributes and their properties	Conflict management is not clearly addressed by the authors. Attempts of using dimension algebra and dimension compatibility is not sufficient to handle frequently observed conflicts	Methodology does not precisely outline conflict identification and resolution measures for the schema instances of tables and their attributes
Technical Qualitative Requirements	Proposition of requirements to handle the integration of varied characteristics of schema and data instances; to generate an merged data warehouse, and for effective query processing	Proposition of *Coherence*, *Soundness*, and *Consistency*; as measures for compatible dimension matching, but the requirements are inconclusive to handle varied properties of schema and data instances	Methodology does not propose qualitative requirements; but adopts and extends the properties outlined in methodology (2) to infer attribute matchings and aggregations in the hierarchies

Here, we adopted a hybrid schema matching methodology for finding mapping correspondences. We also outlined the adoption of first-order GLAV mapping models and their attribute relationship characterization of equality and similarity mappings. Moreover, we addressed the handling of mapping modelling constraints in the form of functional dependencies in the dimensions. We formulated a merge algorithm for integrating disparate data marts into a single consolidated star schema data warehouse.

Furthermore, we addressed the semantics of query processing on the single consolidated data warehouse taking cognizance of the aggregations in hierarchy and summarizability of data instance values for the hierarchies. We identified and outlined the resolution of frequently observed conflicts that are encountered when merging data marts. To this end, we outlined the satisfaction of technical merge correctness requirements for integrating data marts into a data warehouse.

Finally, we compared our methodology of integrating schema and data instances as against other approaches. We outlined the merits and suitability of our approach for delivering an enterprise-wide single consolidated data warehouse from a number of disparate data marts.

The 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. We specifically analyzed the precision of queries in different situations of query processing for corresponding or non-corresponding dimensions from the integrating data marts. We also analyzed the rate of query processing on the single consolidated data warehouse as compared to the individual data marts. We observed that with an increase in the number of data marts, and more specifically, an increase in the data instance tuples the variance of query processing for the concerned data marts decreases considerably. 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 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 data models.

REFERENCES

[1] M. Mireku Kwakye, I. Kiringa, and H. L. Viktor, "Merging Multidimensional Data Models: A Practical Approach for Schema and Data Instances," In Proceedings of the 5th International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA), 2013, pp. 100-107.

[2] R. A. Pottinger and P. A. Bernstein, "Merging Models Based on Given Correspondences," In Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), 2003, pp. 826-873.

[3] M. Lenzerini, "Data Integration: A Theoretical Perspective," In Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 2002, pp. 233-246.

[4] P. A. Bernstein and S. Melnik., "Model Management 2.0: Manipulating Richer Mappings," In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), 2007, pp. 1-12.

[5] S. Melnik, "Generic Model Management: Concepts and Algorithms," Springer Lecture Notes in Computer Science (LNCS), 2004, pp. 2967.

[6] P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger, "A Vision of Management of Complex Models," In Proceedings Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), 2000, vol. 29, no. 4, pp. 55-63.

[7] S. Melnik, "Model Management: First Steps and Beyond," In Proceedings of the 11th Symposium of the GI Department, Database Systems in Business, Technology and Web, (BTW), 2005, pp. 455-464.

[8] M. N. Gubanov, P. A. Bernstein, and A. Moshchuk, "Model Management Engine for Data Integration with Reverse-Engineering Support," In Proceedings of the 24th International Conference on Data Engineering (ICDE), 2008, pp. 1319-1321.

[9] E. Rahm and P. A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," Very Large Data Bases (VLDB) Journal, 2001, vol. 10, no. 4, pp. 334-350.

[10] P. Shvaiko and J. Euzenat, "A Survey of Schema-based Matching Approaches," Journal of Data Semantics IV, vol. 3730, pp. 146-171, 2005, doi:10.1007/11603412_5.

[11] P. Shvaiko, "A Classification of Schema-based Matching Approaches," In Proceedings of the Meaning Coordination and Negotiation Workshop at the 3rd International Semantic Web Conference (ISWC), 2004.

[12] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching," In Proceedings of the 18th International Conference on Data Engineering (ICDE), 2002, pp. 117-128.

[13] H. H. Do and E. Rahm, "COMA: A System for Flexible Combination of Schema Matching Approaches," In Proceedings of 28th International Conference on Very Large Data Bases (VLDB), 2002, pp. 610-621.

[14] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," In Proceedings of 27th International Conference on Very Large Data Bases (VLDB), 2001, pp. 49-58.

[15] W-S. Li and C. Clifton, "SEMINT: A Tool For Identifying Attribute Correspondences In Heterogeneous Databases Using Neural Networks," Elsevier Science. Data and Knowledge Engineering (DKE), 2000, vol. 33, no. 1, pp. 49-84.

[16] R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos, "iMAP: Discovering Complex Mappings between Database Schemas," In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2004, pp. 383-394.

[17] M. A. Hernandez, R. J. Miller, and L. M. Haas, "Clio: A Semi-Automatic Tool For Schema Mapping," In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), 2001, pp. 607.

[18] R. J. Miller, M. A. Hernandez, L. M. Haas, L-L. Yan, C. T. H. Ho, R. Fagin, and L. Popa, "The Clio Project: Managing Heterogeneity," ACM SIGMOD Record, 2001, vol. 30, no. 1, pp. 78-83.

[19] A. Y. Halevy and J. Madhavan, "Corpus-Based Knowledge Representation," In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 1567-1572.

[20] J. Berlin and A. Motro, "Database Schema Matching Using Machine Learning with Feature Selection," In Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE), 2002, pp. 452-466.

[21] A. Islam, D. Z. Inkpen, and I. Kiringa, "Applications of Corpus-based Semantic Similarity and Word Segmentation to Database Schema Matching," Very Large Data Bases (VLDB) Journal, 2008, vol. 17, no. 5, pp. 1293-1320.

[22] M. A. Hernandez, L. Popa, C. T. H. Ho, and F. Naumann, "Clio: A Schema Mapping Tool for Information Integration," In Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN), 2005, pp. 11.

[23] R. Fagin, L. M. Haas, M. A. Hernandez, R. J. Miller, L. Popa, and Y. Velegrakis, "Clio: Schema Mapping Creation and Data Exchange," Conceptual Modelling: Foundations and Applications, 2009, pp. 198-236.

[24] D. Kensche, C. Quix, X. Li, Y. Li, and M. Jarke, "Generic Schema Mappings for Composition and Query Answering," Elsevier Science. Data and Knowledge Engineering (DKE), 2009, vol. 68, no. 7, pp. 599-621.

[25] R. A. Pottinger, "Database Schema Integration," Encyclopedia of GIS, 2008, pp. 226-231.

[26] P. A. Bernstein and E. Rahm, "Data Warehouse Scenarios for Model Management," In Proceedings of the 19th International Conference on Conceptual Modeling (ER), 2000, pp. 1-15.

[27] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, "Data Integration in Data Warehousing," International Journal of Cooperative Information Systems (IJCIS), 2001, vol. 10, no. 3, pp. 237-271.

[28] R. A. Pottinger and P. A. Bernstein, "Schema Merging and Mapping Creation for Relational Sources," In Proceedings of the 11th International Conference on Extending Database Technology (EDBT), 2008, pp. 73-84.

[29] N. Rizopoulos and P. McBrien, "Schema Merging Based on Semantic Mappings," In Proceedings of the 26th British National Conference on Databases (BNCOD), 2009, pp. 193-198.

[30] C. Quix, D. Kensche, and X. Li, "Generic Schema Merging," In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE), 2007, pp. 127-141.

[31] L. Cabibbo and R. Torlone, "On the Integration of Autonomous Data Marts," In Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM), 2004, pp. 223-231.

[32] L. Cabibbo and R. Torlone, "Integrating Heterogeneous Multidimensional Databases," In Proceedings of the 17th International Conference on Scientific and Statistical Database Management (SSDBM), 2005, pp. 205-214.

[33] L. Cabibbo and R. Torlone, "Dimension Compatibility for Data Mart Integration," In Proceedings of the 12th Italian Symposium on Advanced Database Systems (SEBD), 2004, pp. 6-17.

[34] D. Riazati, J. A. Thom, and X. Zhang, "Inferring Aggregation Hierarchies for Integration of Data Marts," In Proceedings of the 21st International Conference on Database and Expert Systems Applications (DEXA), 2010, pp. 96-110.

[35] IBM, *IBM Infosphere Data Architect 7.5.3.0: Finding Relationships.* [Online]. Available: http://publib.boulder.ibm.com/infocenter/idm/v2r1/index.jsp?topic=/com.ibm.datatools.metadata.mapping.ui.doc/topics/iiymdadconfiguring.html. Retrieved: 2014.05.31.

[36] E. Deza and M. M. Deza, "Euclidean Distance," Encyclopedia of Distances, Springer, 2009, pp. 94.

[37] S. Craw, "Manhattan Distance," Encyclopedia of Machine Learning, Springer, 2010, pp. 639.

[38] M. Dash and H. Liu, "Feature Selection for Classification," Intelligent Data Analysis, 1997, vol. 1, no. 3, pp. 131–156.

[39] B. Ten Cate and P. G. Kolaitis, "Structural Characterizations of Schema-Mapping Languages," In Proceedings of the 12th International Conference on Extending Database Technology (ICDT), 2009, pp. 63-72.

[40] M. Friedman, A. Levy, and T. Millstein, "Navigational Plans for Data Integration", In Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence (16. AAAI/ 11. IAAI), 1999, pp. 67-73.

[41] IBM, *IBM Infosphere Data Architect 7.5.3.0.* [Online]. Available: http://www-01.ibm.com/software/data/optim/data-architect. Retrieved: 2014.05.31.

[42] Microsoft, *Microsoft BizTalk Mapper*. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee253382(v=bts.10).aspx. Retrieved: 2014.05.31.

[43] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker, "The Data Warehouse Lifecycle Toolkit," John Wiley and Sons, 2nd Edition, 2008, ISBN-10: 0470149779.

[44] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, "A Foundation for Capturing and Querying Complex Multidimensional Data," Elsevier Science. Information Systems (IS), 2001, vol. 26, no. 5, pp. 383-423.

[45] Microsoft, *Microsoft SQL Server Database Management System*. [Online]. Available: http://www.microsoft.com/en-us/sqlserver/default.aspx. Retrieved: 2014.05.31.

[46] IBM, *IBM Cognos Business Intelligence 10.2.0.* [Online]. Available: http://www-03.ibm.com/software/products/en/business-intelligence. Retrieved: 2014.05.31.

M. Junker, A. Dengel, and R. Hoch, "On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy," In Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR), 1999, pp. 713-716.

XI. APPENDIX

MERGE ALGORITHM PROOF OF CORRECTNESS

A. Preliminaries

In this section, we provide an outlined proof of correctness of the formulated merge algorithm, which establishes query processing on the single consolidated data warehouse.

Definition 9. (Certain Query): A *Query*, $\mathcal{Q}$ is said to be *Certain* for all Instances, $\mathcal{I}$ and Properties, $\mathcal{P}$ of a Multidimensional Database, $\mathcal{MD}$ iff $\mathcal{Q} \vDash \mathcal{I}$, such that $\mathcal{I} \subseteq \mathcal{MD}$ and $\mathcal{Q}$ satisfies $\mathcal{P} \in \mathcal{MD}$ ■

Definition 10. (Certain Answer): A *Tuple*, $\mathcal{T}$ forming an *Answer* to a certain query, $\mathcal{Q}$ is said to be *Certain* iff $\mathcal{T} \vDash \mathcal{Q}$ for all Instances, $\mathcal{I}$ of Multidimensional Database, $\mathcal{MD}$ and $\mathcal{T}$ fulfils $\mathcal{I} \in \mathcal{MD}$ ■

Let $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_p\}$ represent an expected set of p tuple variables of certain answers ranging over a set of queries, $\mathcal{Q}$. Let $\mathcal{Q} = \{\mathcal{Q}_1, \dots, \mathcal{Q}_z\}$ represent a set of z possible and *certain queries* likely to be posed to the single consolidated data warehouse. For the *tuple* $\mathcal{V}$ proving a *query* $\mathcal{Q}$ will mean the *tuple* $\mathcal{V}$ computes *certain answers* to the *query* $\mathcal{Q}$ posed on the single consolidated data warehouse.

Theorem 1. (Merge Algorithm): Let $\mathcal{S}$ and $\mathcal{I}$, respectively, represent the schema and data instances of a *Multidimensional Star Schema Model*, $\mathcal{MD}$. Suppose $\mathcal{MD}$ is instantiated in a *Fact*, $\mathcal{F}$ and m number of *Dimensions* $\mathcal{D}_i, \{1 \leq i \leq m\}$ such that $\mathcal{F} = \{\mathcal{S}_\mathcal{F}, \mathcal{I}_\mathcal{F}\}$, $\mathcal{D}_i = \{\mathcal{S}_{\mathcal{D}_i}, \mathcal{I}_{\mathcal{D}_i}\}$. Then, a merge algorithm which accepts n Star Schema Instances, $\mathcal{MD}_\mathrm{j}$, for $\{2 \leq \mathrm{j} \leq n\}$, and *Mapping Correspondences*, $\mathcal{MAP}_{\mathcal{FD}_i}$ as inputs, generates a *Single Consolidated Data Warehouse*, $\mathcal{DW}$ in a worst-case polynomial time complexity, such that $\{\mathcal{S}, \mathcal{I}\} \in \mathcal{DW} \vDash \{\mathcal{Q}, \mathcal{T}\} \in \mathcal{MD}_\mathrm{j}$ ■

B. Proof of Soundness

PROOF. (SKETCH) **Soundness.** We want to show that, if a *tuple* $\mathcal{V}$ can be proven or computed as a certain answer to a posed *certain query* $\mathcal{Q}$ on the single consolidated data warehouse, $\mathcal{DW}$ then *tuple* $\mathcal{V}$ will answer the *certain query* $\mathcal{Q}$.

(⇒)

By use of inductive definition, we assume for an arbitrary *tuple* $\mathcal{V}$ and *certain query* $\mathcal{Q}$, such that the *tuple* $\mathcal{V}$ is computed in n number of steps for query $\mathcal{Q}$. Consequent to this assumption, the *tuple* $\mathcal{V}$ will represent *certain answers* to the *query* $\mathcal{Q}$. This will hold for all data instances of the single consolidated data warehouse generated from this algorithm.

For Steps (2) to (7), it can be inferred that the mapping correspondences between the integrating instance schema table attributes are iterated in finite steps. The single consolidated data warehouse will then be a representation of all instance schema table attributes.

Since instance data values are associated to each attribute of the schema instances. Hence, *certain answers* for *tuple, say* $\mathcal{V}$, is generated for any *query*, say $\mathcal{Q}$, posed to it.

For Step (5), the intuition that only 2 forms of mapping is adopted implies all forms of mapping ambiguities for possible intractability or a worst-case of an undecidability are not expected. In that regard, exact *certain answers* are expected from a posed query for *equality* mapping types. For *similarity* mapping, similar

certain answers for *tuples* are generated. Non-corresponding attributes also help in generating tuples for local instance schema attributes per data mart. As a result, by inductive proposition the correctness in tuple data values is trivially preserved.

For Step (8), the *tuples* that are generated from schema attributes will have properties of being the *UNION* of all integrating attributes. The unified property thus asserts on all the semantics from each of the integrating attributes. Hence, if a *tuple*, say $\mathcal{V}$, is generated for a *query*, say $\mathcal{Q}$, a truth validity can be ascertained such that the *tuple* will represent a *certain answer*. This makes the inference and inductive claims from the earlier premise satisfy and preserve the *soundness* criteria for correctness. ■

C. Proof of Completeness

The proof of completeness is trivially the converse to the proof of soundness and affirms the validation of the intuition proposed for soundness.

PROOF. (SKETCH) **Completeness.** We want to show that, if a tuple $\mathcal{V}$ is a *certain answer* to a *certain query* $\mathcal{Q}$ posed on the single consolidated data warehouse, $\mathcal{DW}$ then the *tuple* $\mathcal{V}$ can be proven to exist. In other words, for any *query* $\mathcal{Q}$ posed we are sure not to miss any *certain answer* from the tuples that can be generated.

($\Leftarrow$)

We begin the proof by the use of *contraposition* hypothesis to show that: If a *tuple*, say $\mathcal{V}$, cannot be computed or does not exist for a *query*, say $\mathcal{Q}$, then the *tuple* $\mathcal{V}$ cannot represent a *certain answer* to the *query* $\mathcal{Q}$.

Let us assume the *tuple* $\mathcal{V}$ cannot be computed or generated for the *query* $\mathcal{Q}$ in the strong sense.

If the *tuple* $\mathcal{V}$ cannot be computed, then we can construct an infinite general set, $\mathcal{V}^*$ of aggregated *tuples*, which will still not form computed *tuples* to answer the *query* $\mathcal{Q}$.

Based on this construction, we can inductively generate a categorization of all forms aggregation of tuples. We enumerate them as $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots \mathcal{E}_q\}$. We then will inductively define a series of different sets of tuples $\mathcal{V}_n = \{\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_n\}$.

We then let the first of the series of *tuple* sets, $\mathcal{V}_0$ represent the arbitrary *tuple* $\mathcal{V}$. As part of the inductive construction, if the union of one series set of a *tuple*, say $\mathcal{V}_k$, and a subsequent aggregation categorization, say $\mathcal{E}_{k+1}$ is a computed tuple to answer *query* $\mathcal{Q}$, then $\mathcal{V}_{k+1} = \mathcal{V}_k$, meaning we have both tuple sets having the same answering semantics.

On the contrary, if the union of a tuple set, say $\mathcal{V}_k$ and a subsequent aggregation categorization, say $\mathcal{E}_{k+1}$ **does not** form a computed tuple needed to answer *query* $\mathcal{Q}$, then $\mathcal{V}_{k+1} = \mathcal{V}_k \cup \{\mathcal{E}_{k+1}\}$, where the new tuple, $\mathcal{V}_{k+1}$ is definitely giving us a different answer from the initial one, $\mathcal{V}_k$.

We then have the general set $\mathcal{V}^*$ representing a union of all the aggregated tuples, $\mathcal{V}_n$ likely to give an answer to the query. It can be deduced that the general set $\mathcal{V}^*$ holds our supposed *tuple* $\mathcal{V}$.

The general set $\mathcal{V}^*$ does not provide enough computed tuples to form a *certain answer* to the posed *certain query* $\mathcal{Q}$. Because if it does answers the query then additional attribute tuples, as well as other complex formula to the aggregations should make it a valid *certain answer* the query.

The general set $\mathcal{V}^*$ is a closure set with attribute tuples and hierarchy aggregations in relation to our supposed *tuple* $\mathcal{V}$ to forming *certain answers* to the query $\mathcal{Q}$. Hence this closure set $\mathcal{V}^*$ exhibits a satisfiability property for a canonical evaluation of being always true, and never false.

With such a satisfiability property, we can say that there is always a truth-like claim on $\mathcal{V}^*$, where all its generated *tuples* are true and anything outside it false. This will make our computed *tuple* $\mathcal{V}$, always true and make the posted *query* $\mathcal{Q}$, false.

This assertion of the *tuple* $\mathcal{V}$ being true and the posed *query* $\mathcal{Q}$ being false does not offer a claim for the computed *tuple* $\mathcal{V}$ validating as a *certain answer* to the posted *query* $\mathcal{Q}$.

Hence, our preceding proposition of *contraposition* is satisfied and valid. ■