International Journal of Advanced Science and Technology
Vol.120 (2018), pp.29-48
http://dx.doi.org/10.14257/ijast.2018.120.03

A Mapping Approach for Fully Virtual Data Integration System Processes

Ali Zidane El Qutaany[1*], Ali Hamid El Bastawissy[2] and Osman Hegazi[1]

[1,3]Faculty of Computers and Information, Cairo University, Egypt
[2]Faculty of Computer Science, MSA University, Egypt
[1]a.zidane@fci-cu.edu.eg, [2]aelbastawissy@msa.eun.eg, [3]o.hegazy@fci-cu.edu.eg*

Abstract

Nowadays, organizations cannot satisfy their information needs from one data source. Moreover, multiple data sources across the organization fuels the need for data integration. Data integration system's users pose queries in terms of an integrated schema and expect accurate, unambiguous, and complete answers. So the data integration system is not limited to getting the answers for the queries from the sources, but also it is extended to detect and resolve the data quality problems appeared due to the integration process. The most crucial component in any data integration system is the mappings constructed between the data sources and the integrated schema. In this paper a new mapping approach is proposed to map not only the elements of the integrated schema as done by the existing approaches, but also it maps other elements required in detecting and resolving the duplicates. It provides means to facilitate future extensibility and changes to both the sources and the integrated schema. The proposed approach provides a linkage between the fundamental components required to provide accurate and unambiguous answers to the users' queries from the integration system.

Keywords: *Virtual data integration, Inconsistency detection, Inconsistency resolution*

1. Introduction

Data integration refers to the problem of combining data residing at autonomous, homogenous/heterogeneous sources, and providing users with a unified global schema [1]. Data integration system I is formalized in terms of a triple (GS, S, M) [2], where; GS is the integrated schema to represent the participating data sources or the data integration requirements based on predetermined business objectives, it is also called mediated schema between the users and the data sources, S is the "data Sources" participating in the integration process, and M is to map GS to S. There are two radically different integration methods: virtualization and materialization. Virtualization leaves the data where it is, as it is, and dynamically retrieves, merges and transforms it on request. Materialization does the integration up front, creating a new data set for requests to run against. Authors of this research are interested in virtualization. Two main concepts constitute the architecture of a virtual data integration system: wrappers and mediators. Wrapper wraps and models the source using a source schema while mediator maintains a global schema and mappings between the global and source schemas [3]. Users are posing their queries to the integration system in terms of the global schema and expecting to receive accurate, complete and unambiguous answers. To ensure users' expectations; integration system should perform three main processes; Data Integration (DI) process including getting the raw answers from the sources, Inconsistency Detection (ID) process, and Inconsistency Resolution (IR) process. The three main processes can be detailed as follows.

Received (June 10, 2018), Review Result (August 21, 2018), Accepted (August 30, 2018)
[*] Corresponding Author

Data Integration (DI) Process: In this process, the GS is constructed, the S are marked, and M is built. Users pose queries in terms of the GS, and the data integration system converts these queries using M into a set of subqueries over S. Each data source answers the subquery with the help of its wrapper(s). Data sources were created in heterogonous environments; thus data quality problems [4] appear in the collected answers from the sources. These problems occur because the sources often contain redundant data in different representations. Even if, the sources are clean, accurate and the data representations are unified across all the participating sources; some data quality problems appear due to the integration process. One of these problems is mutual inconsistencies which need efforts to be detected and resolved as functions of the successive processes to the integration process. The collected answers for each user's query should be sent as an input to the inconsistency detection process.

Inconsistency Detection Process: This process is called "Duplicate Records Detection" or "Entity Matching", and due to the duplicates; inconsistencies appear, so this process also called "Inconsistencies Detection". In this process; duplicates are detected [5-10] in preparation to remove the ambiguities in the generated answers and to fuse inconsistencies before passing the answers to the user. Detected duplicates are marked in the answer set, and passed as an input to the successive process to resolve the inconsistencies.

Inconsistency Resolution Process: In this process; detected inconsistencies are resolved [11-18] before passing the generated answers to the users. In the literature; there are 3 different strategies [19] to deal with the inconsistencies, some researchers ignore the conflicts resolving process at all, this strategy called "conflict ignorance", others are avoiding [20, 21] dealing with conflicts by defining a pre-determined decision to be taken in case of conflicts called "conflict avoidance", and the rest [22-25] are trying to resolve the inconsistencies once detected called "conflict resolution".

Obviously, one of the main tasks in the design of a data integration system is to establish the mapping M between S and GS, such mapping should be suitably taken into consideration in formalizing a data integration system to serve all of its processes not only the DI process. Basically, there are two mapping approaches [1] to define M: Global-as-View (GAV) and Local-as-View (LAV). However, both approaches have their limitations. To overcome these limitations, another mapping approach is introduced to combine the best of GAV and LAV called Both-as-View BAV. Other derivatives of these approaches, such as Global-Local-as-View GLAV, and Both-Global-Local-as-View provide alternatives for more flexible and scalable data integration but still has a set of limitations. GAV, LAV, and BAV have a common limitation, which is; while defining M, they are not considering the data integration successive processes, they only used for the integration and query answering process, and also they are facing a lot of issues when no shared identifier is used for the integrated real world object from different sources. In this paper, a mapping approach is proposed not only to define the mappings between GS and S, but also to prepare parameters assisting in performing the after integration processes; i.e., the inconsistency detection and resolution processes, and provide means to facilitate future changes, extensibility, flexibility, and scalability of the integration system, and to work with the non-federated and heterogeneous data sources as well as the federated and homogenous ones. The rest of the paper is organized as follows, GAV, LAV, and BAV will be detailed in Section 2 showing their principles, advantages, and limitations, while Section 3 introduces the proposed approach, Section 4 compares the proposed approach with the existing ones, and finally Section 5 concludes the work and states the future work.

2. Related Work

One of the most important aspects in the design of a data integration system is the specification of the correspondence between GS and S. It is exactly this correspondence that will determine how the users' queries posed to the integration system are answered. Three basic approaches for specifying such mapping in a data integration system have been proposed in the literature: LAV, GAV, and BAV. Some derivations are also examined to avoid drawbacks noticed in both GAV and LAV, *e.g.*, BGLAV and GLAV. In this section; the basic approaches are investigated, showing their principles, pros and cons. Then the common limitations faced in the approaches are listed, and a demonstration example is shown to be used throughout the full paper.

2.1. Mapping Approaches

2.1.1. Global as View (GAV) Approach: Mappings in data integration systems based on **GAV** as shown in Figure 1 (a) associates each global relation symbols with views over local relation symbols. In GAV based mapping integration systems, the same GS relation may have more than one mapping assertions over S in case of the unavailability of global relation elements in all data sources. Query processing and simple query reformulation is the most important advantage of GAV. GAV is effective whenever the data integration system is based on a set of stable (do not change too much) sources, but it does not support scalability for the data integration system as changes in GS and/or local schemes derive the designer to revise and alter the mappings. GS in the systems based on GAV approach; can only contain available elements in S at the design time. Finally, it does not prepare parameters for the successive data integration processes as it only considers the data integration and query answering process. IBIS [26], Multiplex, Fusionplex and Autoplex [21] are GAV data integration systems examples.

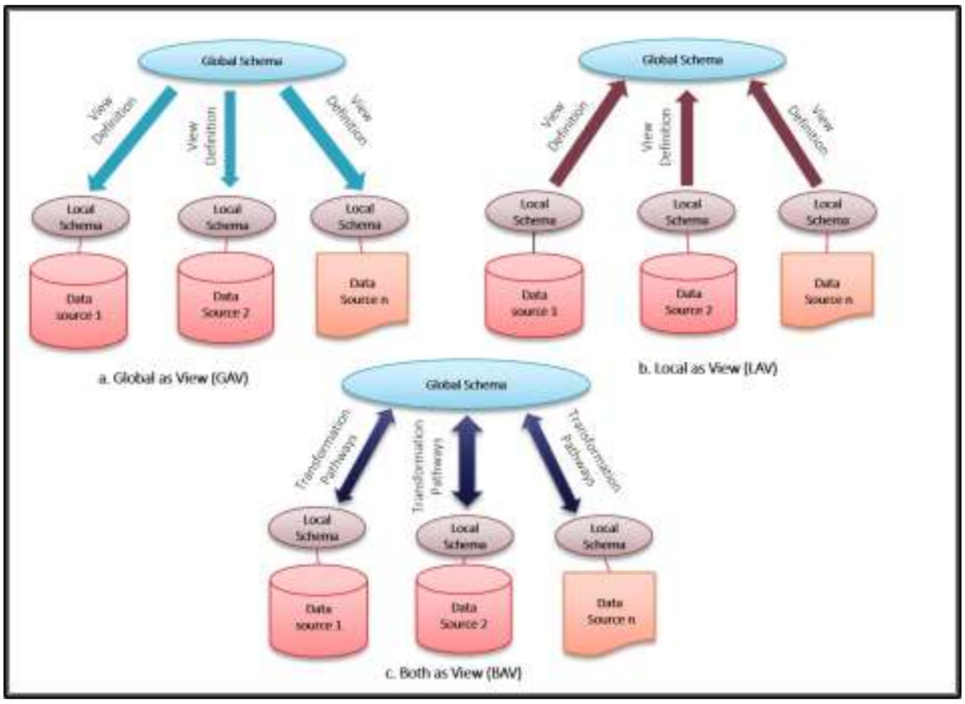


Figure 1. Mapping Approaches GAV, LAV and BAV

2.1.2. Local as View (LAV) Approach: The mapping in data integration systems based on LAV as shown in Figure 1 (b) associates local relation symbols with a view over global relation symbols. LAV approach favors the extensibility of the system where adding a new source simply means enriching the mapping with new assertions, without other changes, so it is effective whenever the data integration system is based on a global schema that is stable and well-established in the organization. But query reformulation has exponential time complexity with respect to query and source schema definitions. GS in the systems based on LAV approach; can only contain available elements in S at the design time. Like GAV; LAV does not prepare parameters for the successive data integration processes as it only considers the data integration and query answering process. Information Manifold [27], System described in [22] are LAV data integration systems examples.

2.1.3. Both-as-View (BAV) Approach: BAV as shown in Figure 1 (c) is an alternative point of view that is neither GAV nor LAV as it uses source to-target mappings based on a predefined conceptual target schema, which is specified ontologically and independently of any of the sources. In BAV for each pair (vS, vG) incrementally modify vS / vG using primitive schema transformations to match vG /vS. BAV [28] is easier to maintain than both GAV and LAV, and query reformulation reduces to rule unfolding [1]. GS can only contain available elements in the sources at the design time. And like both LAV and GAV; BAV does not prepare parameters for the successive data integration processes as it only considers the data integration and query answering process. Clio [29] is a BAV data integration systems example.

2.2. Common Imitations for all Mapping Approaches

These mapping approaches are used to define the mappings between a global schema GS which was designed to integrate data existing in 8 heterogeneous data sources S built under different platforms, these sources use different identification method for the same real world object, *i.e.*, no common identifier for the integrated object from all sources, but some sources may agree on one identifier and others may agree on another identifier. The 8 sources contain data for around 5,000,000 real world objects. In S; the same object may have records in different sources, but each source does not have duplicates for the same object, the GS contains around 80 relations. Attributes within each GS relation are not mapped to all data sources and none of the sources has all attributes of one GS relation. The issues faced during the implementation:

1. None of the mapping approaches, allows the possibility of adding elements to GS for future extensibility of the business objectives, if they are not existing in sources at the design time,
2. As number of the participating information sources increases [30], as the mappings construction, the query answering, and adding new information source or modifying existing one becomes more complicated processes.
3. More than one mapping assertion built for each global schema relation, as not all data sources provide the same attributes and the same number of attributes for the global schema relation.
4. None of the mapping approaches considering the data integration successive processes. As they do not consider mapping the parameters which may help in the detection and resolution processes, e.g. source qualifications.
5. When two of the participating information sources share an identifier for the real world object; then some duplicates are prevented by the mapping assertions definition. In this case, the accurate and most recent information is **not** always presented in the chosen source of providing information in the mappings.

6. Changes in the data sources and/or the global schema require extensive efforts to keep the mappings consistent.

These limitations in the existing mapping approaches become challenges for the proposed approach.

2.3. Demonstration Example

To explain and approve the limitations and drawbacks of the existing approaches, let's start by hypothetically demonstrate the following schemes, later the same hypothesis will be also used to highlight the advantages of the proposed mapping approach.

Example 1. Suppose we have 5 data sources representing oil and gas wells data with their semantics and a global schema which is designed and uses notations and naming independently from the sources.

GS: Well (WellAPI, WellName, Latitude, Longitude, FieldName, County, CompIntervalID) – GS is designed to integrate USA wells.

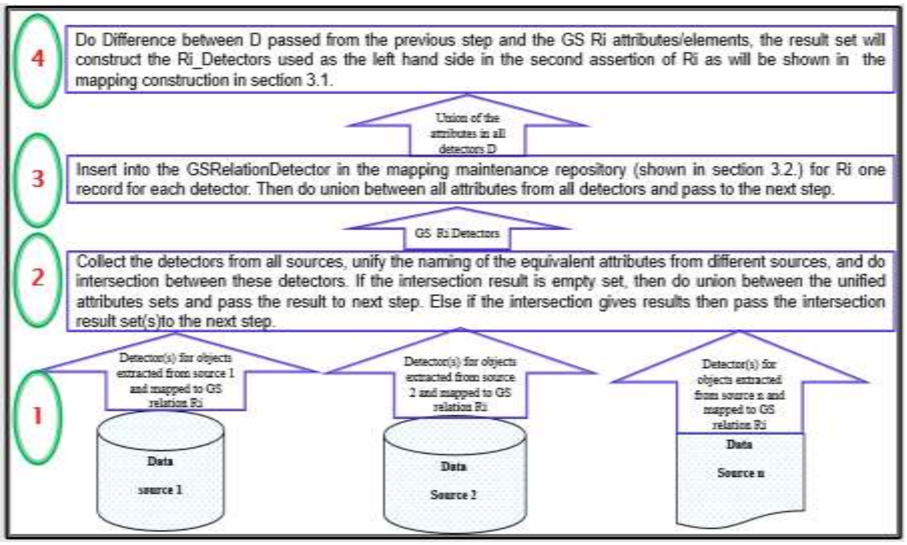
Data sources S – heterogeneous data sources; as the real world object (Well Object) represented in the 5 sources does not has the same identification key across all sources, sources are partially agree on the well object identifier.

1. S1: WellDetails (WellAPI, WellName, Latitude, Longitude, MeasuredDepthFt, HorizontalWell, Country) – Contains data about wells from different countries.
2. S2: Well (APINo, WellName1, Lat, Long) – Contains data about wells from GOM (Gulf of Mexico).
3. S3: USAWellData (WellName, WellSuffix, Latitude, Longitude, FieldName, County, WellMD, HWFlag) – Contains data about wells from USA.
4. S4: GulfArabiaOilWells (Name, TopLatitude, TopLongitude, PrimaryField, Country, MD, HWFlag) – contains data about wells from Gulf of Arabia countries. This is irrelevant source to the integration objective.
5. S5: NorthDakotaWells (Name, APICompSTR, SurfaceLat, SurfaceLong, FieldLocation, Field) – contains data about wells from only North Dakota state (USA).

Detectors (this term will be explained and used while exploring the proposed approach, these detectors can be automatically detected or defined by domain experts. Here in this paper, they defined by domain expert): Detectors for the well object in S1 are {WellAPI} and {Latitude, Longitude, MeasuredDepthFt, HorizontalWell}, S2 uses {APINo}, S3 uses {Latitude, Longitude, WellMD, HWFlag}, and finally S5 uses {Substring (APICompSTR, 0, CharIndex (' ')-1)} as WellAPINumber. WellAPI from S1 is equivalent to both APINo from S2 and Substring (APICompSTR, 0, CharIndex (' ')-1) from S5. WellMD from S3 and MeasuredDepthFt from S1 are equivalent and HorizontalWell from S1 and HWFlag from S3 are equivalent.

3. The Proposed Mapping Approach

Not all of the participating sources in the data integration process are federated as they do not use the same identifier for the real world object. A new term called *detector* is invented to be used in this case. *Detector* is an identifier for the real world object in its origin and it may **not** be shared between all the sources mapped to the GS relation Ri. Real world objects indicated in some sources may agree on a set of detectors while others may agree on another set. Detectors may be one or many for the real world object in its data source. A detector may be single or composite. As Ri will be mapped to data coming from different sources, so the union of these detectors constructs the detectors of Ri although the attributes of these detectors may not be appearing in Ri, by default if the sources are sharing the same identifier then the detector of Ri will be the shared identifier between all sources. One detector or many may be existing per the GS relation. None of the detectors can be considered as an identifier for the GS relation as it will contain nulls for the records extracted from the source(s) which do not agree on these detectors and then violates the entity integrity constraints. Each detector identifies only the objects which extracted from the sources agreed on such detector(s). All the detectors will be used in the duplicate detection process in a hierarchy based, by starting with the first detector and ending with the last detector. Duplicate record detection is out of scope in this paper, but mapping of the detectors for each GS relation is considered. Figure 2 shows how detectors are collected from the sources and processed to construct the GS relation Ri detectors. In example 1 there is no unified identifier for the **well** object in all the data sources, so each data source is required to provide its detectors for the well object as shown in the example. The union of these detectors will construct the detectors of the GS relation. As in the example; S1, S2, and S5 agreed on the detector {WellAPI} and S1, and S3 agreed on the detector {Latitude, Longitude, WellMD, HWFlag}. The GS relation **Well** have two detectors {WellAPI} and {Latitude, Longitude, WellMD, HWFlag}, these detectors will be used during the duplicate record detection process. Inconsistency resolution process is required before passing the results to the user and after the duplicate record detection process.


4 Do Difference between D passed from the previous step and the GS Ri attributes/elements, the result set will construct the Ri_Detectors used as the left hand side in the second assertion of Ri as will be shown in the mapping construction in section 3.1.

3 Insert into the GSRelationDetector in the mapping maintenance repository (shown in section 3.2.) for Ri one record for each detector. Then do union between all attributes from all detectors and pass to the next step.

2 Collect the detectors from all sources, unify the naming of the equivalent attributes from different sources, and do intersection between these detectors. If the intersection result is empty set, then do union between the unified attributes sets and pass the result to next step. Else if the intersection gives results then pass the intersection result set(s)to the next step.

Union of the attributes in all detectors D
GS Ri Detectors
1
Detector(s) for objects extracted from source 1 and mapped to GS relation Ri
Detector(s) for objects extracted from source 2 and mapped to GS relation Ri
Detector(s) for objects extracted from source n and mapped to GS relation Ri
Data source 1
Data Source 2
Data Source n

Figure 2. Construction Process for Detectors of GS Relation Ri

Source preference [22] is one of the fusion policies known in the inconsistency resolution, which fuse the conflicting data based on the preferred source, but to apply such policy you should have the source name in the result set passed to the inconsistency resolution process. In order to accomplish this; source name will be considered in the mapping construction process with the detector sets even if they are not considered in the GS design. The source qualifications, *e.g.*, Timestamp, Cost, Availability… used for the inconsistency resolution process may also be extracted and mapped during the mapping construction process. Here a mapping approach is proposed which is unlike all of the existing mapping approaches, it does not assume the homogeneity between all of the participating data sources, as it works for federated and non-federated data sources. The proposed approach provides means to facilitate the process of defragmenting the results from the data sources, add a new data source(s), remove an existing data source(s), and modifying data source(s). The detectors and source name element defined in this mapping approach may not be part of the elements required in the GS relations for business objectives, but they will be mapped only for performing the data integration successive processes; entity matching and resolution.

3.1. Principles of the Proposed Approach

1. GS designed independently of the sources, and can contain relations and elements which may not be presented in the available sources but added for future scalability and extensibility of the integration system objectives.

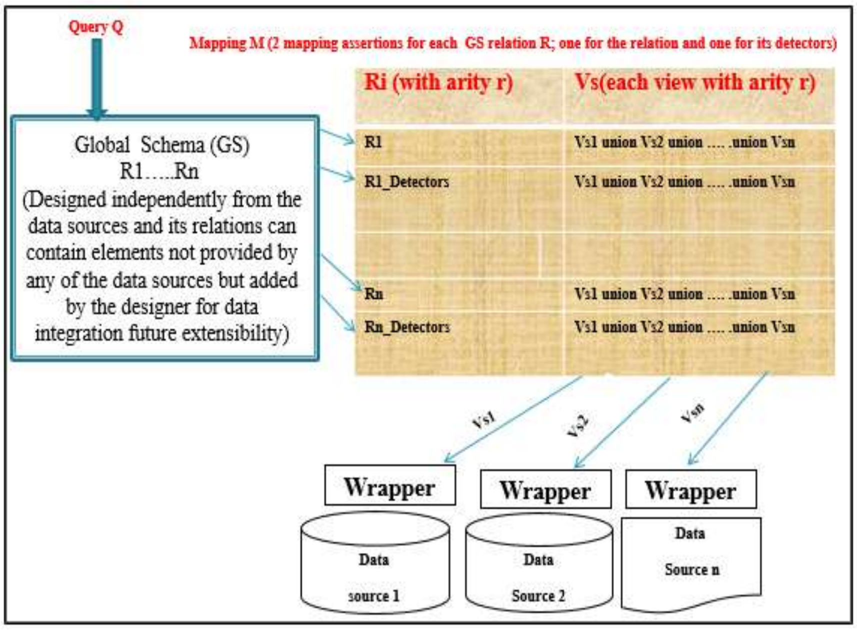

Query Q
Mapping M (2 mapping assertions for each GS relation R; one for the relation and one for its detectors)
Ri (with arity r)
Vs(each view with arity r)
Global Schema (GS)
R1.....Rn
(Designed independently from the data sources and its relations can contain elements not provided by any of the data sources but added by the designer for data integration future extensibility)
R1
Vs1 union Vs2 union .... .union Vsn
R1_Detectors
Vs1 union Vs2 union .... .union Vsn
Rn
Vs1 union Vs2 union .... .union Vsn
Rn_Detectors
Vs1 union Vs2 union .... .union Vsn
Vs1
Vs2
Vsn
Wrapper
Wrapper
Wrapper
Data source 1
Data Source 2
Data Source n

Figure 3. The Mapping Assertions Construction Process Using the Proposed Approach

2. The mappings between GS constructs and S constructs are built as shown in figure 3, where each GS relation Ri has two assertions; one assertion to map Ri elements in the form Ri ➔ views Vs over all data sources linked by union, such that a single view per each source appears in the union of local views to map such source to Ri. View V over source s has the same arity as Ri, such that each attribute appeared on Ri and does not have correspondence with attribute from s is replaced with **Null** and aliased with the corresponding attribute from Ri to facilitate modifying of both the data sources and the GS relations. The second assertion will be constructed to map the Ri detectors and the source name element with the sources participating in Ri mapping assertion, even if they are not present in the GS for business objectives.
3. First mapping assertion for the GS relation will be used for the traditional query answering, and the second mapping assertion is used for the successive data integration processes.
4. Appearance of a specific data source in the mapping assertions follows a specific ordering, where the ordering of the view vS over sourcei is predetermined and stored in MappingHelper table in a standalone repository, shown in section 3.2. This repository will aid in adding or removing data source (s).
5. Users pose their queries in terms of the GS relations.

6. A query Q on the global relations must be translated to a set of subqueries over the data sources.

As an example; the mapping assertions for the GS relation **WELL** in example 1 with the sources will look like:

Assertion-1: Well (WellAPI, WellName, Latitude, Longitude, FieldName, County, CompIntervalID) ➔ Select WellAPI, WellName, Latitude, Longitude, Null as FieldName, Null as County, Null as CompIntervalID from S1.WellDetails Where Country = 'USA' **Union** Select APINo, WellName1, Lat, Long, Null as FieldName, Null as County, Null as CompIntervalID from S2.Well **Union** Select Substring (APICompSTR, 0, CHARINDEX (APICompSTR, ' ')-1) as WellAPI, Name, SurfaceLat, SurfaceLong, FieldLocation+ '-'+ Field as FieldName, Null as County, Substring (APICompSTR, CHARINDEX (APICompSTR, ' ')+1, length (APICompSTR)-1) as CompIntervalID from S5.NorthDakotaWells **Union** Select Null as WellAPI, WellSuffix +' –'+ WellName as WellName , Latitude, Longitude, FieldName, County, Null as CompIntervalID from S3.USAWellData

Assertion-2: Well_Detectors (WellMD, HorizontalWellFlag, SourceName)➔ Select MeasuredDepthFt, HorizontalWell, 'S1' as SourceName From S1.WellDetails Where Country = 'USA' **Union** Select Null as WellMD, Null as HorizontalWellFlag, 'S2' as SourceName from S2.Well **Union** Select Null as WellMD, Null as HorizontalWellFlag, 'S5' as SourceName from S5.NorthDakotaWells **Union** Select WellMD, HWFlag, 'S3' as SourceName from S3.USAWellData.

Well_Detectors only contains three attributes and it is supposed to contain 6 attributes; 5 for detectors (WellAPI Latitude, Longitude, WellMD, and HorizontalWellFlag) and another attribute for SourceName. But as Well relation contains 3 attributes from these 6, so the difference process between Well_Detectors and Well gives {WellMD, HorizontalWellFlag, SourceName} which is used in the Well_Detectors. In the first assertion (Well Assertion), all the local attributes may be aliased with the GS corresponding attributes names even if they provided from the sources, to facilitate the process of query answering afterwards. In example 1 if the designer needs to add two elements WellType and WellStatus to the GS relation Well at the design time although they do not have correspondence with any of the data sources, this is possible in the proposed approach; it becomes as easy as ; just adding them to the GS relation, modifying the mapping assertion Well, and enriching each view over the sources with two elements Null as WellType, Null as WellStatus.

3.2. Mapping Maintenance Helper Repository

This repository contains 2 tables; one called GSRelationDetector, and it has the detectors of each GS relation, it takes the form GSRelationDetector (GSRelationName, Detector), and it is used to help in the query answering to prepare the answer for the duplicate record detection and resolution processes. The second called MappingHelper and it takes the form MappingHelper (GSRelationName, SourceName, SourceIndex), where the SourceIndex is the order of this data source's view within the mapping assertion for the corresponding GS relation. MappingHelper table helps in adding, removing, modifying data sources and/or GS relations. In example 1, the MappingHelper table takes the form shown in table 1. These ordering was used in section 3.1 to build the mappings. And Table 2 shows the GSRelationDetector for example 1.

Table 1. MappingHelper for Example 1

GSRelationName	SourceName	SourceIndex
Well	S1	1
Well	S2	2
Well	S3	4
Well	S5	3

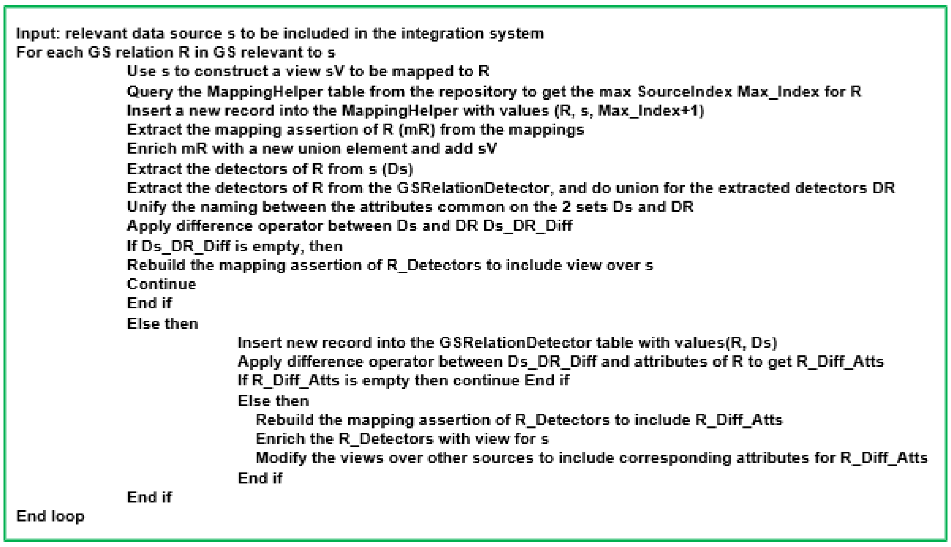
Table 2. GSRelationDetector for Example 1

GSRelationName	Detector
Well	WellAPINumber
Well	Latitude, Longitude, WellMD, HorizontalWellFlag

3.3. Data Sources Management in the Proposed Approach

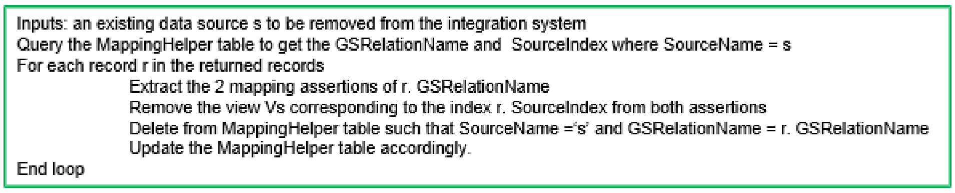
In this section, the operations applied in the data sources are shown, such operations are:

3.3.1. Addition and Removal of a Data Source: One of the features in the proposed mapping approach is the way of adding and removing a data source(s) to and from the integration system. Figure 4 shows an algorithm for the addition process, and Figure 5 shows an algorithm to be used to remove a data source. The same 2 algorithms can be used when adding a relation to a data source or removing a relation from a data source.



```
Input: relevant data source s to be included in the integration system
For each GS relation R in GS relevant to s
        Use s to construct a view sV to be mapped to R
        Query the MappingHelper table from the repository to get the max SourceIndex Max_Index for R
        Insert a new record into the MappingHelper with values (R, s, Max_Index+1)
        Extract the mapping assertion of R (mR) from the mappings
        Enrich mR with a new union element and add sV
        Extract the detectors of R from s (Ds)
        Extract the detectors of R from the GSRelationDetector, and do union for the extracted detectors DR
        Unify the naming between the attributes common on the 2 sets Ds and DR
        Apply difference operator between Ds and DR Ds_DR_Diff
        If Ds_DR_Diff is empty, then
        Rebuild the mapping assertion of R_Detectors to include view over s
        Continue
        End if
        Else then
                Insert new record into the GSRelationDetector table with values(R, Ds)
                Apply difference operator between Ds_DR_Diff and attributes of R to get R_Diff_Atts
                If R_Diff_Atts is empty then continue End if
                Else then
                   Rebuild the mapping assertion of R_Detectors to include R_Diff_Atts
                   Enrich the R_Detectors with view for s
                   Modify the views over other sources to include corresponding attributes for R_Diff_Atts
                End if
        End if
End loop
```

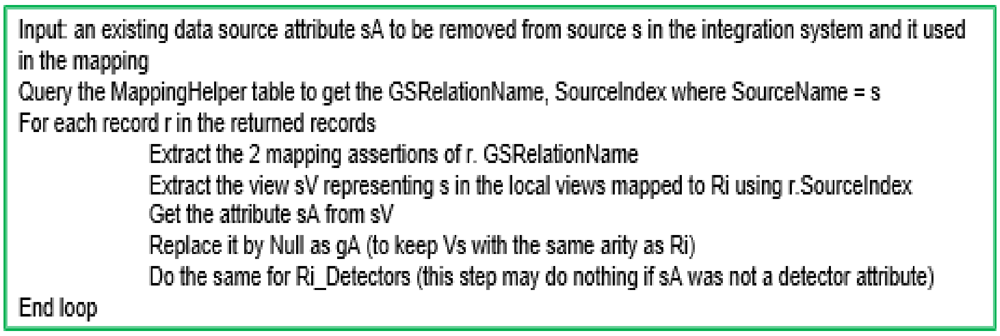
Figure 4. Algorithm for Addition of a New Data Source to the Integration System Using the Proposed Approach



```
Inputs: an existing data source s to be removed from the integration system
Query the MappingHelper table to get the GSRelationName and SourceIndex where SourceName = s
For each record r in the returned records
        Extract the 2 mapping assertions of r. GSRelationName
        Remove the view Vs corresponding to the index r. SourceIndex from both assertions
        Delete from MappingHelper table such that SourceName ='s' and GSRelationName = r. GSRelationName
        Update the MappingHelper table accordingly.
End loop
```

Figure 5. Algorithm for Removal of an Existing Data Source Using the Proposed Approach

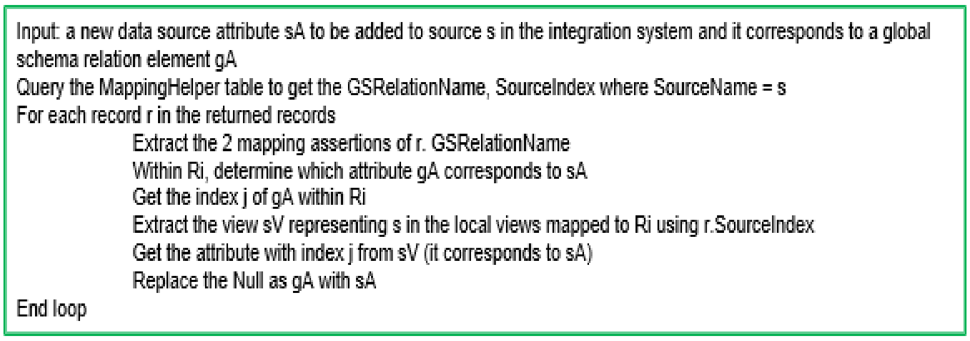
3.3.2. Addition/Removal of an Element to a Data Source: In the proposed mapping approach; the views built over the data sources have the same arity as the GS relation in the mapping assertion.



```
Input: an existing data source attribute sA to be removed from source s in the integration system and it used
in the mapping
Query the MappingHelper table to get the GSRelationName, SourceIndex where SourceName = s
For each record r in the returned records
        Extract the 2 mapping assertions of r. GSRelationName
        Extract the view sV representing s in the local views mapped to Ri using r.SourceIndex
        Get the attribute sA from sV
        Replace it by Null as gA (to keep Vs with the same arity as Ri)
        Do the same for Ri_Detectors (this step may do nothing if sA was not a detector attribute)
End loop
```

Figure 6. Algorithm for Addition of a New Attribute to a Data Source

Thus adding and removing attributes to and from a data source become an easy process. Figure 6 presents an algorithm to remove an old attribute from a data source and Figure 7 presents an algorithm to show how a new attribute can be added to a data source. In Figure 6, if sA is a detector and does not exist in Ri and Ri_Detectors, it will be added to the Ri_Detectors as the last element, and added to the view representing s in the detectors assertion and finally add a new field to the other sources' views to represent this attribute, this new field will take the form *Null as* gA, where gA is the GS relation element corresponding to sA.



```
Input: a new data source attribute sA to be added to source s in the integration system and it corresponds to a global
schema relation element gA
Query the MappingHelper table to get the GSRelationName, SourceIndex where SourceName = s
For each record r in the returned records
        Extract the 2 mapping assertions of r. GSRelationName
        Within Ri, determine which attribute gA corresponds to sA
        Get the index j of gA within Ri
        Extract the view sV representing s in the local views mapped to Ri using r.SourceIndex
        Get the attribute with index j from sV (it corresponds to sA)
        Replace the Null as gA with sA
End loop
```

Figure 7. Algorithm for Removal of an Attribute from a Data Source

3.4. GS Management in the Proposed Approach

In this section the operations done over the GS are detailed, such operations are:

3.4.1. Addition and Removal of a GS Relation: To remove a GS relation Ri from an integration system; search in the mapping assertions for Ri and Ri_Detectors and remove them. If a new GS relation Ri needed to be added to the GS:

1. Construct the views sV over the relevant sources, perform a union over all the constructed views, fill in the MappingHelper table with the order of the sources appearing in the union, and build the mapping assertion.
2. Collect the detectors as shown in Figure 2, fill in the GSRelationDetector with Ri detectors and construct another mapping assertion for Ri_Detectors.

3.4.2. Addition and Removal of an Element in a GS Relation: Figure 8 presents an algorithm to add a new attribute gA to a GS relation Ri, while Figure 9 presents an algorithm to remove an attribute gA from a GS relation Ri.

```
Input: a new GS relation attribute gA to be added to a GS relation Ri in the integration system
Extract the 2 mapping assertions of Ri
Add gA as the last element in Ri
Query the MappingHelper to get the sources (relevantS) which have correspondence with Ri
Get the sources which contain elements corresponding to gA (gARelevenat)
Perform a difference between gARelevenat and relevantS to get the new sources
Build view sV over each of the new sources and enrich both Ri and Ri_Detectors assertions with these
view(s), then update the MappingHelper table. This step may do nothing if the difference returns empty set.
Add a new element in views over relevantS. If it corresponds to an attribute in s then it will be mapped
normally, otherwise add NULL as gA to keep the views with the same arity as Ri
```

Figure 8. Algorithm for Addition of an Attribute gA to a GS relation Ri

```
Input: an existing GS relation attribute/element gA to be removed from GS relation Ri in the integration system
Extract the mapping assertion of Ri
Get the index j of gA within Ri
Remove the attribute of index j from all local views in this assertion
Check if gA is a detector attribute then add it to Ri_Detectors and modify the assertion.
```

Figure 9. Algorithm for Removal of an Attribute gA from a GS Relation Ri

3.5. Query Answering in the Proposed Approach

Figure 10 shows the query answering in the proposed approach. A query Q is answered as follows:

1) The query Q is parsed against the GS relations.

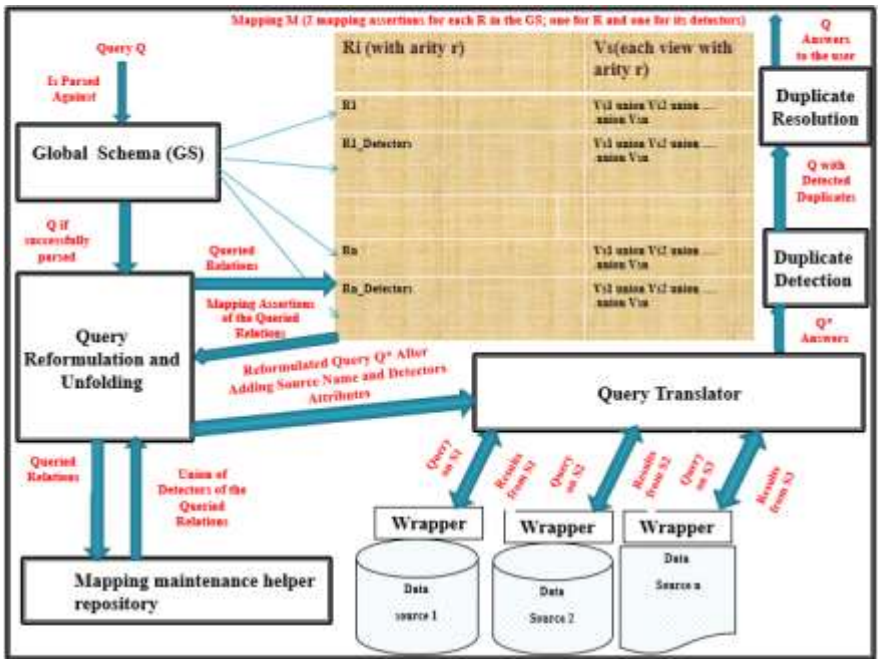

Mapping M (2 mapping assertions for each R in the GS; one for R and one for its detectors)
Query Q
Is Parsed Against
Global Schema (GS)
Q if successfully parsed
Queried Relations
Ri (with arity r)
Vs(each view with arity r)
R1
Vs1 union Vs2 union ... union Vsn
R1_Detectors
Vs1 union Vs2 union ... union Vsn
Rn
Vs1 union Vs2 union ... union Vsn
Rn_Detectors
Vs1 union Vs2 union ... union Vsn
Mapping Assertions of the Queried Relations
Query Reformulation and Unfolding
Reformulated Query Q* After Adding Source Name and Detectors Attributes
Query Translator
Q Answers to the user
Duplicate Resolution
Q with Detected Duplicates
Duplicate Detection
Q* Answers
Queried Relations
Union of Detectors of the Queried Relations
Mapping maintenance helper repository
Query on S1
Results from S1
Query on S2
Results from S2
Query on S3
Results from S3
Wrapper
Wrapper
Wrapper
Data source 1
Data Source 2
Data Source n

Figure 10. Query Answering in the Proposed Mapping Approach

2) The queried GS relations are extracted using the query reformulation and unfolding module and asks the mapping helper repository for the detectors of the queried relations. The query Q is reformulated to add the detectors (if they are not in Q), the source name, and the filtering attributes, ask for the 2 mapping assertions for each queried relation which serve the query elements, merge the elements/attributes of the

2 assertions of each GS relation and at the end replace each GS relation with its corresponding merged assertion to construct Q*.

3) The reformulated query Q* is passed to the query translator to prepare a subquery for each data source. The subqueries prepared for the sources are adjusted to include the filtering attributes of Q, such that any of the filtering attributes corresponds to Null value in the view is removed from the filtering clause of the subquery, and if the filtering attribute is one and corresponds to null in any of the source views or have "and" condition with any of the other attributes, then this means the subquery will not return any answers from the source, so it will not be sent to the source from the beginning. This serves as a huge optimization since a whole data source will not be visited in this case.

4) The answers are collected from the sources.

5) The answers of Q* are sent to the duplicate detection process, to detect the duplicates using the detectors, send the answers with detected duplicates to duplicate resolution to resolve the conflicts, and finally project over the original query attributes to be sent to the user as the final query answer.

As an example, using example 1 and the mapping assertions defined in 3.1. If a user poses a query Q *(Select WellAPI, WellName, Latitude from Well where FieldName = 'CHARLES KRAMER 1608')*, this Q will be answered as follows:

1. The query is parsed against GS.

2. The query reformulation and unfolding module extracts the queried relation(s) from Q, in this case it will be the relation **Well**. It then asks the mapping maintenance helper repository for the detectors of Well and the source name attribute, it will be WellAPI, Latitude, Longitude, WellMD, and HorizontalWellFlag, reformulates Q to be (Select WellAPI, WellName, Latitude, Longitude, WellMD, HorizontalWellFlag, FieldName SourceName from Well where FieldName = 'CHARLES KRAMER 1608') after union the query projection part, the query selection part, the detectors, and the SourceName attribute. The query reformulation and unfolding module asks for the mapping assertions of the relation **Well**, this will result in the 2 mapping assertions in Section 3.1.

3. The query reformulation and unfolding module merges the 2 mapping assertions to be one assertion to serve Q elements, the merged assertion looks like: Well (WellAPI, WellName, Latitude, Longitude, WellMD, HorizontalWellFlag, FieldName, SourceName) ➔ Select WellAPI, WellName, Latitude, Longitude, MeasuredDepthFt, HorizontalWell, Null as FieldName , 'S1' as SourceName From S1.WellDetails Where Country = 'USA' **Union** Select APINo, WellName1, Lat, Long, Null as WellMD, Null as HorizontalWellFlag, Null as FieldName , 'S2' as SourceName From S2.Well **Union** Select Substring (APICompSTR, 0, CHARINDEX (APICompSTR ,' ')-1) as WellAPINumber, Name, SurfaceLat, SurfaceLong, , Null as WellMD, Null as HorizontalWellFlag, FieldLocation+ '-'+ Field as FieldName, 'S5' as SourceName From S5.NorthDakotaWells **Union** Select Null as WellAPI, WellSuffix +' –'+ WellName as WellName , Latitude, Longitude, WellMD, HWFlag, FieldName, 'S3' as SourceName From S3.USAWellData. Finally replace **Well** by the new merged assertion to construct Q* and pass Q* to the query translator.

4. The query translator translates the Q* into a set of sub-queries for the data sources, so query on S1 will be Select WellAPI, WellName, Latitude, Longitude, MeasuredDepthFt, HorizontalWell, Null as FieldName, 'S1' as SourceName from S1.WellDetails Where Country = 'USA' and FieldName = 'CHARLES KRAMER 1608'. Query on S2 will be Select APINo, WellName1, Lat, Long, Null as WellMD,

Null as HorizontalWellFlag, Null as FieldName, ‘S2’ as SourceName from S2.Well where FieldName = ‘CHARLES KRAMER 1608’. Query on S5 will be Select Substring (APICompSTR, 0, CHARINDEX (APICompSTR, ‘ ’)-1) as WellAPI, Name, SurfaceLat, SurfaceLong, , Null as WellMD, Null as HorizontalWellFlag, FieldLocation+ ‘-’+ Field as FieldName, ‘S5’ as SourceName from S5.NorthDakotaWells. where FieldName = ‘CHARLES KRAMER 1608’ Query on S3 will be Select Null as WellAPI, WellSuffix +’ –‘+ WellName as WellName , Latitude, Longitude, WellMD, HWFlag, FieldName, ‘S3’ as SourceName from S3.USAWellData where FieldName = ‘CHARLES KRAMER 1608’

5. Subqueries over S1and S2 will **not** be sent to the sources as they will not retrieve answers, while sub-queries over S3 and S5 will be sent.
6. The query translator will collect the answers from the sources, do simple union between the collected answers, as all sub-queries are with the same arity and the columns headers of the query result will be using the GS corresponding headers. The answers sent to the duplicate detection and resolution modules.

4. Comparison between the Existing Approaches and the Proposed Approach

In this section, a comparison is performed between the proposed mapping approach and the existing ones, through showing how all the operations are done using the different approaches.

4.1. Mapping Assertions Construction

When using the mapping approaches to define the mapping assertions between the GS relation and the data sources shown in example 1; the mapping assertions using the existing mapping approaches will be done as follow:

Mapping assertions in GAV: GAV produces 4 assertions for example.1 as below

1) Well (WellName, Latitude, Longitude) ➔ Select WellName, Latitude, Longitude from S1.WellDetails Where Country = ‘USA’ **Union** Select WellName1, Lat, Long from S2.Well **Union** Select, Name, SurfaceLat, SurfaceLong from S5. NorthDakotaWells
2) Well (WellAPI) ➔ Select WellAPI from S1.WellDetails Where Country = ‘USA’ Union Select Substring (APICompSTR, 0, CHARINDEX (APICompSTR ,‘ ’)-1) As WellAPI from S5.NorthDakotaWells Where WellAPI not in (Select WellAPI from S1.WellDetails) Union Select APINo from S2.Well Where WellAPI not in (Select WellAPI from S1.WellDetails union Select WellAPI from S5.NorthDakotaWells)
3) Well (FieldName, County) ➔ Select FieldLocation +‘-’+Field As FieldName, ‘North Dakota’ as County from S5.NorthDakotaWells union Select FieldName, County from S3.USAWellData
4) Well (CompIntervalID) ➔ Select Substring (APICompSTR, CHARINDEX (APICompSTR ,‘ ’)+1, Length (APICompSTR) As CompIntervalID from S5.NorthDakotaWells

Drawbacks of GAV assertions: drawbacks noticed when using GAV in example 1

1. When shared identifier found as mapping assertion number 2, incomplete answer may be generated due to the mapping assertion construction as when the real world object extracted from S1, it will not be extracted from the other sources S2 and S5. This

means the most recent data, the complete data, and/or the accurate data may not be extracted for the resolution process.

2. GAV cannot add elements which are not previously existing in the sources to the GS relation.
3. More than one mapping assertions represent the same GS relation.
4. The source name is not considered in the mappings as it does not appear in the GS relation Well, so the successive processes will not have enough parameters to be done effectively. As the source preferences will not be performed, and the duplicates will be checked between all records including the ones coming from the same data source, even if there are no duplicates in data coming from the same data source which is time consuming.
5. In case of no shared identifier defined commonly between all sources for the same real world object, shared identifier may be available partially between some sources, but these identifiers will not be mapped if they are not requested in the GS relations, so they will not be leveraged in the successive processes.

Mapping Assertions in LAV: LAV produces 4 assertions for example.1 as below

1) S1.WellDetails (WellAPI, WellName, Latitude, Longitude, Country)➔Select WellAPI, WellName, Latitude, Longitude, ‘USA’ as Country from GS.Well
2) S2.Well (APINo, WellName1, Lat, Long) ➔ Select WellAPI, WellName, Latitude, Longitude from GS.Well where Latitude between 26.01614 and 30.23556 and longitude between - 97.14265 and -86.594202.
3) S3.USAWellData (WellName, WellSuffix, Latitude, Longitude, FieldName, County)➔ Select Substring (WellName, CHARINDEX (WellName,‘-’) + 1, Length (WellName)) as WellName, Substring (WellName, 0, CHARINDEX (WellName ,‘-’)-1) as WellSuffix, Latitude, Longitude, FieldName, County from GS. Well
4) S5.NorthDakotaWells (Name, APICompSTR, SurfaceLat, SurfaceLong, FieldLocation, Field)➔ Select WellName, WellAPI + ‘ ‘+ CompIntervalID as APICompSTR, Latitude, Longitude, Substring (FieldName, 0 , CHARINDEX (FieldName ,‘-’)-1) as FieldLocation, Substring (FieldName, CHARINDEX (FieldName ,‘-’)+ 1, Length (FieldName)) as Field, from GS. Well where County = ‘North Dakota’

Drawbacks of LAV assertions: drawbacks noticed when using LAV in example 1

1. Source views may be mapped to complete set of objects of the GS view; *e.g.*, mapping assertion number 1 associates all records of GS relation Well to S1.WellDetails. And this causes complexities in the query translations and answering.
2. Requires extra information about the source semantics, e.g. mapping assertion number 2 requires information about how we can determine the GOM wells, the min/max of latitude and longitude for GOM wells which stored in S2.
3. Drawbacks 2, 3, 4, and 5 noticed in GAV are also noticed here in LAV.

Mapping assertions in BAV: BAV produces 4 assertions for example.1 as below

1) Select WellAPI, WellName, Latitude, Longitude from GS.Well ➔ Select WellAPI, WellName, Latitude, Longitude from S1.WellDetails Where Country = ‘USA’

2) Select WellAPI, WellName, Latitude, Longitude from GS.Well where Latitude between 26.01614 and 30.235566 and longitude between - 97.14265 and -86.59420 ➔Select APINo, WellName1, Lat, Long from S2.Well.

3) Select WellName, Substring (WellName, 0, CHARINDEX (WellName ,'-')-1) as WellSuffix, Substring (WellName, CHARINDEX (WellName ,'-') + 1, Length (WellName)) as WellName, Latitude, Longitude, FieldName, County from GS. Well➔Select (WellSuffix +'-'+ WellName) As WellName, WellSuffix, WellName, Latitude, Longitude, FieldName, County from S3.USAWellData

4) Select WellAPI, WellName, Latitude, Longitude, Substring (FieldName, 0 , CHARINDEX (FieldName ,'-')-1) As FieldLocation, Substring (FieldName, CHARINDEX (FieldName ,'-')+ 1, Length (FieldName)) as Field, FieldName, CompIntervalID, WellAPINumber + ' '+ CompIntervalID as APICompSTR from GS. Well Where County = 'North Dakota' ➔ Select Substring (APICompSTR, 0, CHARINDEX (APICompSTR ,' ')-1) as WellAPI, Name, SurfaceLat, SurfaceLong, FieldLocation , field, (FieldLocation+ '-'+ Field) as FieldName, Substring (APICompSTR , CharIndex (APICompSTR ,' ')+1, Length (APICompSTR)) as CompIntervalID, APICompSTR from S5.NorthDakotaWells

Drawbacks of BAV assertions: drawbacks noticed when using BAV in example 1 are

1. Drawbacks 2, 3, 4, and 5 noticed in GAV are noticed also here in BAV.
2. Needs extra efforts and time to keep matching between the local and global views.

4.2. Data Sources Management using the Existing and Proposed Approaches

4.2.1. Addition of a Data Source: A new source S6 with *WellData (API, Name, Field, County, Comp, Country)* will be added to the integration system in example 1

Using GAV: In GAV, adding a new data source leads to revisiting all the mapping assertions to see which one should be altered and may lead to addition of a new assertion. The addition of the relation WellData will cause:

- Changes to the mapping assertion 2, 3, and 4 under the GAV mapping assertions shown in section 4.1. to include union with new views Select API from S6.WellData where Country = 'USA' and API not in (Select WellAPI from S1.WellDetails union Select WellAPI from S5.NorthDakotaWells **Union** Select APINo from S2.Well), Select Field, County from S6.WellData where Country ='USA', and Select Comp from S6.WellData where Country ='USA' respectively.

- Adding of a new mapping assertion to map the Name element to the GS relation Well. The new mapping assertion will be number 5, and will take this form Well (WellName) ➔ Select Name from S5.NorthDakotaWells Where Country = 'USA'

Using LAV: In LAV, adding a new data source S6 will only cause adding a new mapping assertion 5, for the LAV mapping assertion Select API, Name, Field, County, Comp, Country from S6.WellData Where Country= 'USA'➔Select WellAPI, WellName, FieldName, County , CompIntervalID, "USA" as Country from GS.Well

Using BAV: Adding S6 to the integration system in example 3.1 using BAV approach, will be done by building a view vG over the GS relation Well and a view vS over the WellData relation from S6, and mapping vG to vS.

Using the Proposed Approach

In the proposed approach, adding a new data source S6 will be performed by adding a new union in the Well assertion with a view over the added source and the

Well_Detectors assertion will be modified to include a new detector view for the added source. The two views are Select API, Name, Null as Latitude, Null as Longitude, Null as FieldName, County, Comp from S6.WellData Where Country = 'USA'. And the detectors view will be Select Null as WellMD, Null as HorizontalWellFlag, 'S6' as SourceName from S6.USAWell Where Country = 'USA' Afterwards the MappingHelper table will be updated to have this record ('Well', 'S6', 5)

4.2.2. Removal of a Data Source

If S5 in example 1 will be removed from the integration system built:

Using GAV: The following steps will be required to remove the S5 source:

- Reconstruct the mapping assertion number 1 under the GAV mapping assertions shown in section 4.1. To remove the view representing S5.
- Remove the mapping assertion number 4 as it contains CompIntervalID which comes only from S5,
- Revisit the GS relation **Well** to remove the element CompIntervalID as it is only existing in S5.

Using LAV: If S5 is removed from the integration system in example 1 LAV will remove the mapping assertion number 4, and revisit the GS relation Well to remove the CompIntervalID from there.

Using BAV: Removal of S5 in example 1 from the integration system built using BAV will cause to remove the mapping assertion number 4 underneath the BAV assertions, and revise the GS to remove the CompIntervalID from there.

Using the Proposed Approach: Removing S5 from the integration system build using the proposed approach in example 1 will be performed as follows:

1. Remove the view number 3 from both the **Well_Detectors** and **Well** mapping assertions.
2. Remove from MappingHelper the records related to S5 and GS relation Well, finally update the MappingHelper data to keep the consistency of the ordering of the sources in the mappings caused by the removal of S5.

4.2.3. Removal and Addition of an Attribute in a Data Source

Using GAV: To add API attribute to S3 in GAV, the mapping assertions which will be affected are:

- The mapping assertion number 2 will be removed as it will not be needed.
- The mapping assertion number 1 will be modified to include WellAPI attribute from all sources.

To remove APICompSTR attribute from S5 in GAV, both the mappings and the GS will be affected, where:

- The mapping assertion number 2 will be revised to remove the view over S5.
- The mapping assertion number 4 will be removed.
- The GS will be revised to remove the CompIntervalID attribute from there.

Using BAV and LAV: To add API attribute to S3 in LAV and BAV, only the mapping assertion number 3 will be modified to include the API attribute. And to remove APICompSTR attribute from S5 in LAV and BAV, the mappings and the GS will be affected, where:

- The mapping assertion number 4 modified to not include two attributes WellAPI and CompIntervalID
- The GS will be revised to remove the CompIntervalID attribute from there.

Using the Proposed Approach: To add API attribute to S3 in the proposed approach, only the local view corresponding to S3 will be extracted and modified such that Null as WellAPI will be replaced by API. And to remove APICompSTR attribute from S5 in the proposed approach, only the local view corresponding to S5 will be extracted and modified to replace Substring (APICompSTR, 0, CHARINDEX (APICompSTR, ' ')-1) as WellAPI by Null as WellAPI and Substring (APICompSTR, CHARINDEX (APICompSTR,' ')+1, Length(APICompSTR)) as CompIntervalID by Null as CompIntervalID.

4.3. GS Management using the Existing and Proposed Approaches

If the attribute WellType intended to be added to the GS relation **Well** for future usage, and at the same time the attribute WellAPI will be removed from GS relation Well, the existing mapping approaches will refuse the addition process and can handle the removal process as follows:

Using GAV: The removal of the attribute in GAV will cause to modify all the mapping assertions with correspondence to this attribute. E.g. mapping assertion number 2 will removed.

Using LAV and BAV: The removal of the attribute in LAV and BAV will cause to modify most of the mapping assertions with correspondence to this attribute. Explicitly mapping assertions number 1, 2, and 4 will be revised.

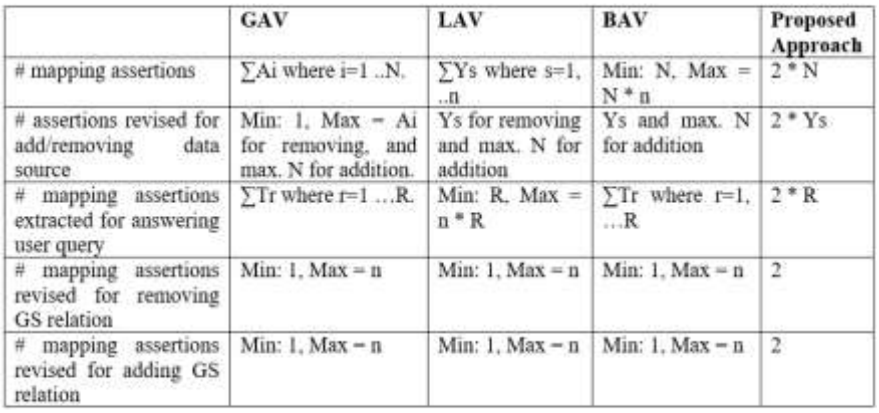
Using the Proposed Approach: For the addition, the proposed approach will be considering it, and will modify the mapping assertion of GS relation **Well** to include such attribute as the last attribute in the relation and map it with the sources as usual. If the attribute existing in any source will be mapped normally, else on the other case, the view over such source will have extra attribute `NULL as` WellType. In example 1 all local views will have `NULL as WellType`. Moreover, the removal of an attribute will be simpler as no need to revise the GS relation Well, what will be done is only parsing the local views and replace the attribute mapping with a `NULL as WellAPI`.
Finally, these are other features provided by the proposed approach:

a) Prepares the environment for the successive process, duplicate detection and resolution.
b) Handles the situation where no shared identifier for the real world object between the data sources in the integration system.
c) Ensures correctness and efficiency of collecting the answers from the different data sources as the views already linked with a traditional union operator and all the views have the same arity as the GS relation.
d) Ensures the completeness of the query answers, as it allows all alternatives for the same real world object from all data sources, and does not prevent any source to participate in the mapping construction.

4.4. Complexities in the Proposed Approach Compared to other Approaches

Table 3 shows the mapping assertions complexities for the proposed approach compared to other approaches.

Table 3. Comparison between the Proposed and Existing Approaches



	GAV	LAV	BAV	Proposed Approach
# mapping assertions	∑Ai where i=1 ..N.	∑Ys where s=1, ..n	Min: N, Max = N * n	2 * N
# assertions revised for add/removing data source	Min: 1, Max = Ai for removing, and max. N for addition.	Ys for removing and max. N for addition	Ys and max. N for addition	2 * Ys
# mapping assertions extracted for answering user query	∑Tr where r=1 …R.	Min: R, Max = n * R	∑Tr where r=1, …R	2 * R
# mapping assertions revised for removing GS relation	Min: 1, Max = n	Min: 1, Max = n	Min: 1, Max = n	2
# mapping assertions revised for adding GS relation	Min: 1, Max = n	Min: 1, Max = n	Min: 1, Max = n	2

The notations used in the comparison and calculations of the complexities are; **N**: # GS relations, **n**: # relevant information sources, **Ys**: # GS relations a data source s has correspondence with, **R**: # GS relations appear in the user's query, and **Tr**: # Mappings for a given GS relation R, Ai is number of sources used to map GS relation Ri.

5. Conclusion and Future Works

In this paper a new mapping approach is introduced to avoid most of the noticed limitations in the existing approaches; as it is not only mapping the GS elements with the local schemes but also mapping the elements required for detecting and resolving the conflicts happened due to the integration process. The proposed approach facilitates the extensibility of the GS, and the sources. The proposed approach provides improvement in adding, removing and updating the global schema GS and the sources S. The proposed approach links the 3 main processes required to answer the user's queries to help in providing complete, and unambiguous answers to those queries. As a future work; formalizing a duplicate detection algorithm to leverage this mapping approach and the detectors defined to detect the duplicates, and use the sources of the data to resolve the duplicate through source preferences.

References

[1] M. Lenzerini, "Data integration: A theoretical perspective", Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems), Madison, Wisconsin, USA, **(2002)** June 3-5.

[2] B. Golshan, A. Y. Halevy, M. Mihaila and W. C. Tan, "Data Integration: After the Teenage Years", Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'17), Chicago, Illinois, USA, **(2017)** May 14-19.

[3] L. Xu and D. W. Embley, "Combining the Best of Global-as-View and Local-as-View for Data Integration", Proceedings of the 3rd ISTA, Salt Lake city, Utah, USA, **(2004)** June 15-17.

[4] E. Rahm and H. H. Do, "Data Cleaning: Problems and Current Approaches", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 23, no. 4, **(2001)**, pp. 103-113.

[5] X. Chu, I. F. Ilyas and P. Koutris, "Distributed Data Deduplication", Proceedings of the VLDB Endowment, vol. 9, no. 11, **(2016)**, pp. 864-875.

[6] A. Elmagaramid, P. G. Ipeirotis and V. S. Verykios, "Duplicate Record Detection: A Survey", IEEE Transactions on Knowledge and Data engineering, vol. 19, no. 1, **(2007)**, pp. 1-16.

[7] M. Nentwig, M. Hartung, A. Ngomo and E. Rahm, "A Survey of Current Link Discovery Frameworks", Semantic Web Journal, vol. 8, no. 3, **(2016)**, pp. 419-436.

[8] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute and V. Raghavendra, "Deep Learning for Entity Matching: A Design Space Exploration", Proceedings of the International Conference on Management of Data SIGMOD'18, TX, USA, **(2018)** June 10-15.

[9] Y. Yang, Y. Sun, J. Tang, B. Ma and J. Li, "Entity Matching across Heterogeneous Sources", Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15), Sydney, NSW, Australia, **(2015)** August 10-13.

[10] A. Gruenheid, X. L. Dong and D. Srivastava, "Incremental Record Linkage", VLDB Endowment, vol. 7, no. 9, **(2014)**, pp. 697-708.

[11] E. K. Rezig, E. C. Dragut, M. Ouzzani and A. K. Elmagarmid, "Query-time record linkage and fusion over Web databases", Proceedings of IEEE 31st International Conference on Data Engineering, Seoul, South Korea, **(2015)** April 13-17.

[12] E. K. Rezig, E. C. Dragut, M. Ouzzani, A. K. Elmagarmid and W. G. Aref, "ORLF: A flexible framework for online record linkage and fusion", Proceedings of IEEE 32nd International Conference on Data Engineering, Helsinki, Finland, **(2016)** May 16-20.

[13] I. F. Ilyas and X. Chu, "Trends in Cleaning Relational Data: Consistency and Deduplication", Foundations and Trends in Databases Journal, vol. 5, no. 4, **(2015)**, pp. 281-393.

[14] A. Bronselaer, D. V. Britsom and G. D. Tre, "Pointwise multi-values fusion", Proceedings of the 18th International Conference on Information Fusion, Washington, USA, **(2015)** July 7-9.

[15] D. Dubois, W. Liu, J. Ma and H. Prade, "The Basic Principles of Uncertain Information Fusion. An organized review of merging rules in different representation frameworks", Proceedings of Information Fusion Heidelberg, Germany, **(2016)** July 5-8.

[16] A. Bronselaer, D. V. Britsom and G. D. Tre, "Propagation of Data Fusion", IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 5, **(2015)**, pp. 1330-1342.

[17] X. Chen, E. Schallehn and G. Saake, "Cloud-Scale Entity Resolution: Current State and Open Challenges", Open Journal of Big Data (OJBD), vol. 4, no. 1, **(2018)**, pp. 30-51.

[18] A. Gal, "Tutorial: Uncertain Entity Resolution", VLDB Endowment, vol. 7, no. 13, **(2014)**, pp. 1711-1712.

[19] J. Bleiholder and F. Neumann, "Conflict Handling Strategies in an Integrated Information System", Workshop on Information Integration on the Web (IIWeb), Edinburgh, UK, **(2006)** May 22-26.

[20] A. Bilke, J. Bleiholder, C. Bohm, K. Draba, F. Naumann and M. Weis, "Automatic data fusion with HumMer", Proceedings of the 31st VLDB, Trondheim, Norway, **(2005)** August 30-September 2.

[21] A Motro, J Berlin and P. Anokhin, "Multiplex, Fusionplex and Autoplex: three generations of information integration", ACM SIGMOD Record, vol. 33, no. 4, **(2004)**, pp. 51-57.

[22] G. D. Giacomo, D. Lembo, M. Lenzerini and R. Rosati, "Tackling Inconsistencies in Data Integration through Source Preferences", Proceedings of the International Workshop on Information Quality in Information Systems, Paris, France, **(2004)** June 18.

[23] Y. Katsis, A. Deutsch, Y. Papakonstantinou and V. Vassalos, "Inconsistency resolution in online databases", Proceedings of IEEE 26th International Conference on Data Engineering (ICDE), Long Beach, California, USA, **(2010)** March 1-6.

[24] P. N. Mendes, H. Muhleisen and C. Bizer, "Sieve: Linked Data Quality Assessment and Fusion", 2nd International Workshop on Linked Web Data Management (LWDM 2012) at the 15th International Conference on Extending Database Technology, Berlin, Germany, **(2012)** March 26-30.

[25] W. Fan, F. Geerts, N. Tang and W. Yu, "Inferring Data Currency and Consistency for Conflict Resolution", Proceedings of the 2013 IEEE International Conference on Data Engineering, Brisbane, Australia, **(2013)** April 8-12.

[26] A. Cali, D. Calvanese, G. De Giacomo and M. Lenzerini, "Data integration under integrity constraints", In Proceedings of the 14th International Conference on Advanced Information Systems Engineering, Ontario, Canada, **(2002)** May 27-31.

[27] T. Kirk, A. Y. Levy, Y. Sagiv and D. Srivastava, "The Information Manifold", Proceedings of the AAAI Spring Symp. On Information Gathering from Heterogeneous, Distributed Enviroments, Cambridge, Massachusetts, United States, **(1995)** November 10-12.

[28] P. J. McBrien and A. Poulovassilis, "Data integration by bi-directional schema transformation rules", Proceedings 19th International Conference on Data Engineering, Bangalore, India, **(2003)** March 5-8.

[29] R. Fagin, L. M. Haas, M. Hernandez, R. J. Miller, L. Popa and Y. Velegrakis, "Clio: Schema Mapping Creation and Data Exchange". Conceptual Modeling: Foundations and Applications, Springer-Verlag, Berlin, Heidelberg, **(2009)**.

[30] E. Rahm. "The Case for Holistic Data Integration", Proceedings of East European Conference on Advances in Databases and Information Systems, Prague, Czech Republic, **(2016)** August 28-31.