Semantic Conflicts Reconciliation as a Viable Solution for Semantic Heterogeneity Problems

Walaa S. Ismail
Faculty of Computers and Information, Information Systems Department, Helwan University

Mona M. Nasr
Faculty of Computers and Information, Information Systems Department, Helwan University

Torky I. Sultan
Faculty of Computers and Information, Information Systems Department, Helwan University

Ayman E. Khedr
Faculty of Computers and Information, Information Systems Department, Helwan University

***Abstractt*—Achieving semantic interoperability is a current challenge in the field of data integration in order to bridge semantic conflicts occurring when the participating sources and receivers use different or implicit data assumptions. Providing a framework that automatically detects and resolves semantic conflicts is considered as a daunting task for many reasons, it should preserve the local autonomy of the integrated sources, as well as provides a standard query language for accessing the integrated data on a global basis. Many existing traditional and ontology-based approaches have tried to achieve semantic interoperability, but they have certain drawbacks that make them inappropriate for integrating data from a large number of participating sources.**

We propose semantic conflicts reconciliation (SCR) framework, it is ontology-based system in which all data semantics explicitly described in the knowledge representation phase and automatically taken into account through the interpretation mediation service phase, so conflicts detected and resolved automatically at the query time.

***Keywords—Data Integration; Heterogeneous Sources; Interoperability; Semantic Conflicts; Context; Reconciliation Ontology*.**

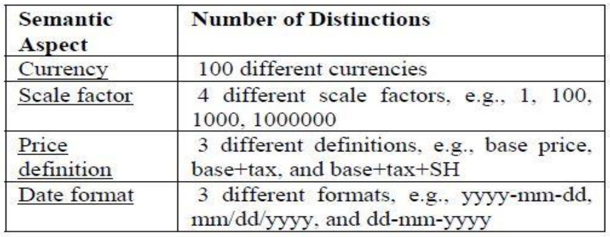
I. INTRODUCTION

Despite the fact that a typical large organization spends nearly 30% of its IT budget on integration and interoperation related efforts, many inter- and intra- organizational systems still have poor interoperability [10]. Technologies already exist to overcome the heterogeneity in hardware, software, and syntax that is used in different systems (e.g., the ODBC standard, XML based standards, web services and SOA-Service Oriented Architectures) .While these capabilities are essential to information integration, they do not address the issue of heterogeneous data semantics that exist both within and across enterprises [11].

Heterogeneity problem occurs when data sources and receivers use different contexts (assumptions); a user submit query and interprets the results in a certain context, which completely different from contexts received from sources. Implicit assumptions made in each source need to be explicitly described and used to reconcile conflicts when data from these systems are combined [3]. Ontology plays an important role on making domain assumptions unambiguous or uniquely identifies the meaning of concepts in a specific domain of interest.

Let us assume that the comparison service covers 100 countries, each having its unique currency and each consisting of 100 vendors. Thus, there are a total of 10,000 sources in this example. For simplicity, let's assume the consumer chooses his context to be the same as one of the sources. Although all vendors in the same country may use the same currency for price, they may use different price definitions and scale factors [9]. Table 1 summarizes the potential context differences in terms of just these four semantic aspects : currency, scale factor, price definition, and date format (for the purpose of finding exchange rate at a given day).

TABLE I. Semantic Differences in Data Sources [9]



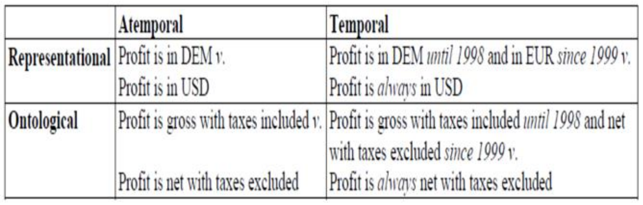
Semantic Aspect	Number of Distinctions
Currency	100 different currencies
Scale factor	4 different scale factors, e.g., 1, 100, 1000, 1000000
Price definition	3 different definitions, e.g., base price, base+tax, and base+tax+SH
Date format	3 different formats, e.g., yyyy-mm-dd, mm/dd/yyyy, and dd-mm-yyyy

Thus, there could be 3600 (i.e., 100*4*3*3) different contexts amongst these sources; e.g., one source has US dollars for currency, scale factor being 1, price as tax and shipping and handling included, with mm/dd/yyyy date format; another source has Turkish liras for currency, scale factor being 1000000, price as only tax included, with dd-mm-yyyy date format, etc. The online comparison service needs to implement the conversions so that the comparison can be performed for sources in any context.

Implementing tens of thousands of data conversions is not an easy task; but maintaining them to cope with changes in data sources and receiver requirements over time is even more challenging [2]

According to Firat [1] there are three dimensions of semantic heterogeneity: contextual, ontological and temporal. Contextual heterogeneity occurs when different systems (sender/receiver) make different assumptions about the representation of the same concept, such as the profit of a company can be represented in DEM (i.e., Deutschmarks) in one system or in USD (i.e., U.S. dollars) in another, where the currency used is the assumption. So there will be two or more not identical representations of the same thing. Ontological heterogeneity occurs when different meanings denoted by the same term (e.g., whether the profit is gross profit including taxes or net profit excluding taxes) because there is a definitional conflicts concerning the inclusion or exclusion of

TABLE II. Temporal vs. Atemporal heterogeneity [4].tax in the profit.



	Atemporal	Temporal
Representational	Profit is in DEM *v.* Profit is in USD	Profit is in DEM *until 1998* and in EUR *since 1999 v.* Profit is *always* in USD
Ontological	Profit is gross with taxes included *v.* Profit is net with taxes excluded	Profit is gross with taxes included *until 1998* and net with taxes excluded *since 1999 v.* Profit is *always* net with taxes excluded

Both the representational and the ontological assumptions can be static and do not change over time within an interested time period, in which case time is not of concern. The resulting heterogeneity is atemporal. Conversely, the assumptions can change over time, and the resulting heterogeneity is temporal [4].

There should be systematic approaches in order to reconcile semantic heterogeneity among heterogeneous sources and receivers.

II. Existing Approaches For Achieving Semantic Interoperability

We can resolve semantic conflicts by hand-coded programs but on small scale only; alternative solutions are needed as the number of systems and the complexity of each system increase.

A. *Traditional Approaches*

Brute-force Data Conversions (*BF*)

In the Brute-force Data Conversions (BF) approach all necessary conversions implemented with hand-coded programs. For example, if we have N data sources and receivers, N (N-1) such conversions need to be implemented to convert the sources context to the receiver context. These conversions become costly to implement and very difficult to maintain When N is large. This is a labor-intensive process; nearly 70% of integration costs come from the implementation of these data conversion programs. A possible variation of the (BF) approach is to group sources that share the same set of semantic assumptions into one context. The approach allows multiple sources in the same context to share the same conversion programs, so the numbers of conversion programs will be reduced. We refer to the original approach and this variation as BFS and BFC, respectively [2]. These approaches are illustrated schematically in Fig 1.

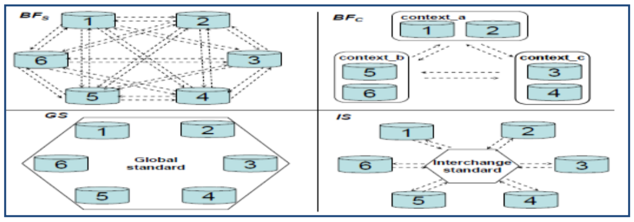


Fig.1. Traditional approaches to Semantic Interoperability [9].

Global Data Standardization (GS)

If we could develop and maintain a single data standard that defines a set of concepts and specifies the corresponding representation, all semantic differences would disappear and there would be no need for data conversion. Unfortunately, such standardization is usually infeasible in practice for several reasons. There are legitimate needs for having different definitions for concepts, storing and reporting data in different formats. Most integration and information exchange efforts involve many existing systems, agreeing to a standard often means someone has to change his/her current implementation, which creates obstacles and makes the standard development and enforcement extremely difficult [7].

Interchange Data Standardization (IS)

Data exchange systems can sometimes agree on the data to be exchanged, i.e., standardizing a set of concepts as well as their interchange formats. The underlying systems do not need to store the data according to the standard; it suffices as long as each data sender generates the data according to the standard. That is, this approach requires that each system have conversions between its local data and an interchange standard used for exchanging data with other systems. Thus, each system still maintains its own autonomy. This is different from the global data standardization, where all systems must store data according to a global standard. With N systems exchanging information, the Interchange Standardization approach requires 2N conversions. The IS approach is a significant improvement over the brute-force approach that might need to implement conversions between every pair of systems [9]. Although this approach has certain advantages, it also has several serious limitations [2]. From which, all parties should reach an agreement on the data definition and data format. Reaching such an agreement can be a costly and time-consuming process besides; any change to the interchange standard affects all systems and the existing conversion programs. Lastly, the approach can involve many unnecessary data conversions

B. *Ontology-Based Data Integration Approaches*

Most of the shortcomings in the previous traditional approaches can be overcome by using ontology-based systems .We explain the most popular ontology-based systems for data integration, which are SCROL and COIN with respect to the role and use of ontologies.

SCROL is a global schema approach that uses an ontology to explicitly categorize and represent predetermined types of semantic heterogeneity [6]. It is based on the use of a common ontology, which specifies a vocabulary to describe and interpret shared information among its users. It is similar to the federated schema approach. However, an ontology-based domain model captures much richer semantics and covers a much broader range of knowledge within a target domain. But it uses a fully specified ontology to explicitly categorize and represent predetermined types of semantic heterogeneity. SCROL assumes that the underlying information sources are structured data that may reside in the structurally organized text files or database systems. However, the unprecedented growth of Internet technologies has made vast amounts of resources instantly accessible to various users via the World Wide Web (WWW) [6].

COIN Project was initiated in 1991 with the goal of achieving semantics interoperability among heterogeneous information sources. The main elements of this architecture are wrappers, context axioms, elevation axioms, a domain model, context mediators, an optimizer and an executioner. A domain model in COIN is a collection of primitive types and semantic types (similar to type in the object-oriented paradigm), which defines the application domain corresponding to the data sources that are to be integrated COIN introduces a new definition for describing things in the world. It states that the truth of a statement can only be understood with reference to a given context. The context information can be obtained by examining the data environment of each data source [11].

The problem of semantic interoperability is not new, and people have tried to achieve semantic interoperability in the past using various approaches. Traditional approaches have sometimes been reasonably successful in limited applications, but have proven either very costly to use, hard to scale to larger applications, or both. Traditional approaches have certain drawbacks that make them inappropriate for integrating information from a large number of data sources. Existing ontology-based approaches for semantic interoperability also have not been sufficiently effective because there is no systematic methodology to follow, no concert methodology for building ontologies and all existing ontology-based not able to reconcile all types of semantic conflicts.

III. SCR ARCHITECTURE

The Semantic Conflicts Reconciliation (SCR) framework is considered as ontology based system aims to solve semantic data level conflicts among different sources and receivers in a systematic methodology. SCR is based on domain specific ontology to create user queries. The user can browse the merged ontology and selects specific terms and conditions to create global query. There is no need for the user to be aware of terms in databases in order to query them. The selected terms are mapped to the corresponding terms in each data source to decompose the global query to a set of sub naïve queries. The decomposed sub-queries are converted to well-formed sub-queries before sending it to the suitable database. Finally the SCR combine and resend the well-formed query results after reconciling the detected conflicts to the users according to the required contexts.

SCR consists of two phases, the knowledge representation phase and the interpretation mediation service phase [8].

A. Knowledge Representation

The knowledge representation phase consists of the following components:

- Ontology Extraction: Extract local ontology from each database.
- Global Ontology: Merge all local ontologies to construct a global one that contains all major concepts and the relationships between them.
- Contexts: Explicitly describing the sources and receivers assumptions about data.
- Mapping: Linking between the constructed merged ontology and the corresponding terms in each data source in order to produce the semantic catalog.

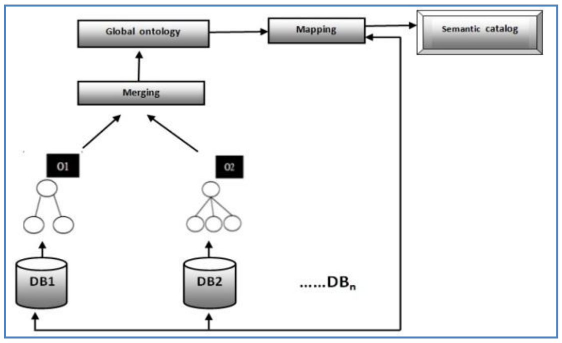


Fig.2. Knowledge representation phase[8].

Database to Ontology Extraction:

In the Ontology extraction step, we have multiple databases to extract a local ontology from each one. A local ontology contains all database information like tables, columns, relations, constraints. Moreover, it contains intentional definitions to represent higher level of abstraction than traditional data models.

The local ontology represents a relational database tables as concept and columns as slots of the concept. The local ontologies are represented in a formal standard language called OWL (Ontology Web Language).

Creating local ontology for each database saves them independent. Any changes in the schema or relations can be added easily to its local ontology. The local ontology includes only the metadata and additional semantics; however, the database instances or members still in the data sources separated from its ontology.

Global Ontology Construction:

Our framework is based on the hybrid ontology approach in which we create local ontology for each data source and a global ontology which is considered a reference for all local ontologies involved in the integration process.

The Merging process aims to create one global (merged) ontology that contains multiple local ontologies contents. It contains all the knowledge of the initial ontologies [5].in order to create a merged ontology, the corresponding objects will be matched from two or more local ontologies. Subsequently, suitable matching algorithm should choose. Matching is the core of the merging process to make one vantage point of view from multiple ontologies, where some concepts and slots will be represented as a new concept and new slots, or some slots may be merged and follow another concept. We can say that, there is a new structure that will be created in the merged ontology. This structure does not affect the information sources, because each local ontology is independent. Creating a standard formal model (merged ontology) makes query multiple databases satisfy the user requirements at the semantic level.

The SCR framework uses PROMPT tool to matching and merging local ontologies. PROMPT is a semi-automatic tool. It is protégé plug in. It guides the expert by providing suggestions. PROMPT provides suggestions about merging and copying classes. Figure 3 explains the PROMPT algorithm [4]. PROMPT takes two ontologies as input and guide the user to create a merged ontology as output. IT generates a list of suggestions based on the choose matching algorithm. Our Framework uses PROMPT lexical matching algorithm.

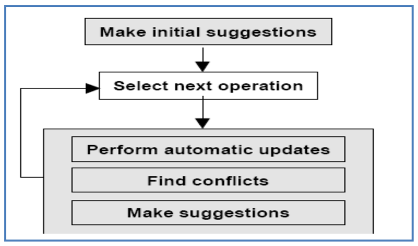


Fig.3. The Flow of PROMPT Algorithm [4].

According to Noy and Musen [4] about PROMPT evaluation, human experts follow 90% of PROMPT suggestions. During the merging process, PROMPT suggested 74% of the total knowledge_ base operations that the user invoked. PROMPT s able to perform a large number of merging operations on its own (or with simple "approval" of a human expert). Thus, it can save the expert's time and efforts.

Explicitly define contexts

The proposed framework assigns contexts descriptions about data items in each source using the following two steps.

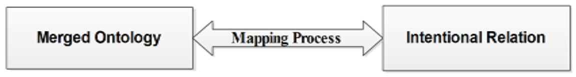
Adding annotation: Adding annotation properties (modifiers) to the global ontology slots to denote their contexts. We consider annotation properties as special properties that affect the interpretation of data values.

Value assignment: Assign values that explicitly describe the semantics of data in different aspects for each annotation properties created in the previous step.

We can associate more than one annotation property to the same. We can easily add, remove and change the assigned values in the ontology whenever the context changed in the sources over time.

Mapping global ontology to multiple databases:

We developed a semi automatic mapping tool used to map the merged ontology to multiple databases. The main purpose of the proposed mapping tool is to find and match between semantically similar terms in the global query with the corresponding terms in the data sources of the integrated system. The output of the mapping process is the semantic catalog.



The mapping process in our proposed mapping tool follows the following steps:

- The mapping process started by creating a database with two tables, to save the mapping data in the first table, and saving the metadata of the database system in the second table. This process is done once when the mapping process is started.
- The expert selects the first database in the heterogeneous integrated sources to link its schema (intentional relation) with the terms in the global ontology.
- When the user selects database from a list of all databases that existed, then all tables in the selected database will be listed. Then, press to select columns, all columns in the selected table will be listed and saved in the table created in the first step along with the correspondence terms in the global ontology. All the primary keys, foreign keys, and referenced tables for each table in the selected database are automatically retrieved and saved in the second created table as metadata, to use it in query processing.

B. *Interpretation Mediation Service*

Knowledge representation phase is not enough to solve semantic conflicts among data sources and receivers. The second phase in our proposed system is the interpretation mediation service in which the user interacts with the system through graphical user interface (GUI). With the Interpretation Mediation Service support the user no longer concerned about context differences and how contexts evolve.

The SCR framework architecture, consisting of the following four main components.

- System interface
- Query preprocessor
- SQL generator
- Converter (query engine)

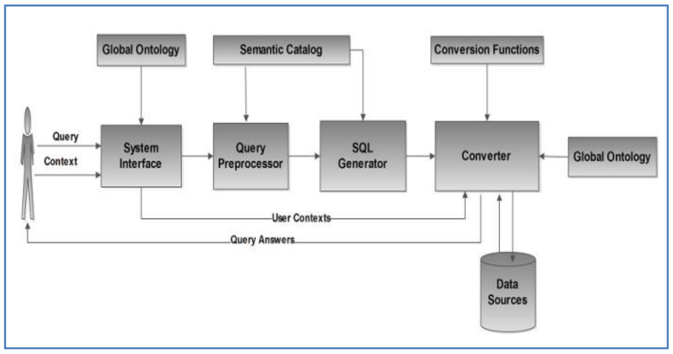


Fig.4. Architecture of the SCR Framework.

System Interface

We cannot suppose that users have intimate knowledge about data sources being queried especially when the number of these sources are big. Users should remain isolated from semantic conflicts problems and no need to learn one of the ontology query languages to formulate his query. User interacts with the system through graphical user interface (GUI). The GUI displays the global ontology terms to facilitate finding the global query terms easily and quickly. User browses the global ontology to select specific terms for his query.

Query preprocessor

Query preprocessor receives the global query terms and semantic catalog as input and produce blocks of user's data based on the selected items from the system interface. Each block represents a query but without any language format. Once the user selects terms and conditions from the system the query preprocessor does the following actions.

- Query preprocessor utilizes semantic catalog (mapping file) to retrieve the database name, table name and columns names that mapped to the selected terms and conditions in the user query.
- The query preprocessor reorganizes the retrieved data from the previous step into blocks according to the database name. Each block represents a query but it does not present in any language format.

SQL Generator

SQL generator turns the query blocks received from the query preprocessor into SQL queries and directs them to the converter. It uses the semantic catalog (metadata) to translate the previous blocks into SQL correct syntax. In order to transform the blocks to correct syntax, the generator adds select, from and where clauses. In addition, if the query needs to retrieve instances from more than one table the primary keys, foreign keys and referenced tables of the integrated databases may be added from the semantic catalog metadata file as well.

Converter (query engine)

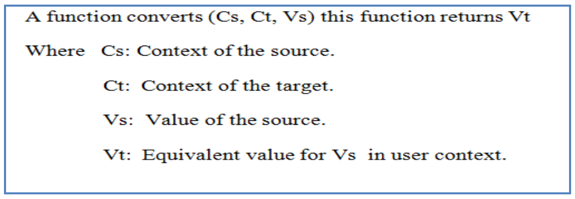
We consider converter as a query engine that takes SQL queries from the SQL generator and the user context as input. Converter connects to the merged (global) ontology to retrieve and compare annotations in order to transform the user naïve query (that ignores differences in assumptions between sources) into a well-formed query that respects differences among sources and receivers contexts. The query engine rewrites the user query into a mediated query (multiple sub-queries) with a set of instructions and functions in order to reconcile the semantic conflicts.

The Converter detects and reconciles semantic conflicts between sources and receivers according to the following two stages:

First stage: before sending the SQL queries to the suitable data source, the query engine connects to the merged ontology and compares the annotation values between the global query contexts and the data sources contexts that are involved in the global query. The query engine detects and reconciles the conflicts at the query time then directs each SQL query to the suitable database.

Second stage: the query engine compares the annotation values of each item in the sources involved in the global query with the user required context .
Whether in the first or the second stage, the converter connects to a set of conversion functions defined in order to convert between contexts and satisfy the user expectations about results.

The general form of the conversion function can be as follows:


A function converts (Cs, Ct, Vs) this function returns Vt

Where Cs: Context of the source.

Ct: Context of the target.

Vs: Value of the source.

Vt: Equivalent value for Vs in user context.

Fig.5. General form of the conversion functions[8].

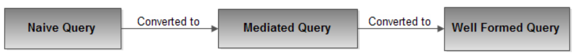
Conversion functions represent the conversions among all annotation property values or contexts described in the merged ontology. In the SCR system, there is no relation between the number of sources or receivers and the number of conversion functions. Conversion functions in SCR system are parameterized Component conversion that is defined for each modifier in the ontology. These functions can convert between all values of a modifier automatically (e.g., a conversion that reconciles between currencies of price modifier).

IV. SCR Software System Description

We developed the SCR software system for demonstrating the feasibility and the features of our approach. It helps the user to query integrated sources with minimal efforts using Query-by-Example language (QBE). As a result, the user needs only to know little information about the global ontology terms.

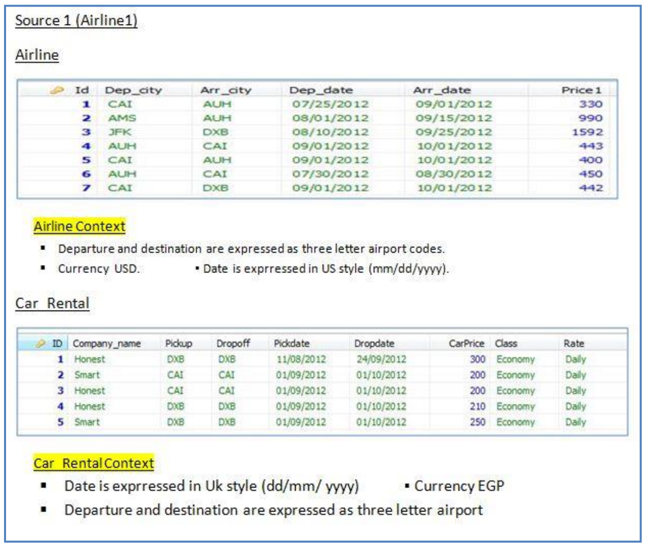
A. *Naive to well-formed query conversion*

Consider the following scenario in order to describe how the SCR system transforms the user naïve query into a well-formed query.



Illustrative example (Airline Reservation Scenario)

Consider the example where a comprehensive travel system involving multiple airlines, and car rental services. Assume we have two data sources and each source has its own contexts. The User can find the most suitable and best cheapest airline booking along with the cheapest car rental prices through the different car rental providers available. The airline reservation scenario displays prices and airline information from different airlines data sources given departure and destination locations and dates. We show snapshots from this scenario in Figure 6 and Figure 7. As shown there are many conflicts between the sources contexts in addition to the user different contexts or assumptions about data.



Source 1 (Airline1)

Airline

Id	Dep_city	Arr_city	Dep_date	Arr_date	Price 1
1	CAI	AUH	07/25/2012	09/01/2012	330
2	AMS	AUH	08/01/2012	09/15/2012	990
3	JFK	DXB	08/10/2012	09/25/2012	1592
4	AUH	CAI	09/01/2012	10/01/2012	443
5	CAI	AUH	09/01/2012	10/01/2012	400
6	AUH	CAI	07/30/2012	08/30/2012	450
7	CAI	DXB	09/01/2012	10/01/2012	442

Airline Context

- Departure and destination are expressed as three letter airport codes.
- Currency USD. • Date is exprressed in US style (mm/dd/yyyy).

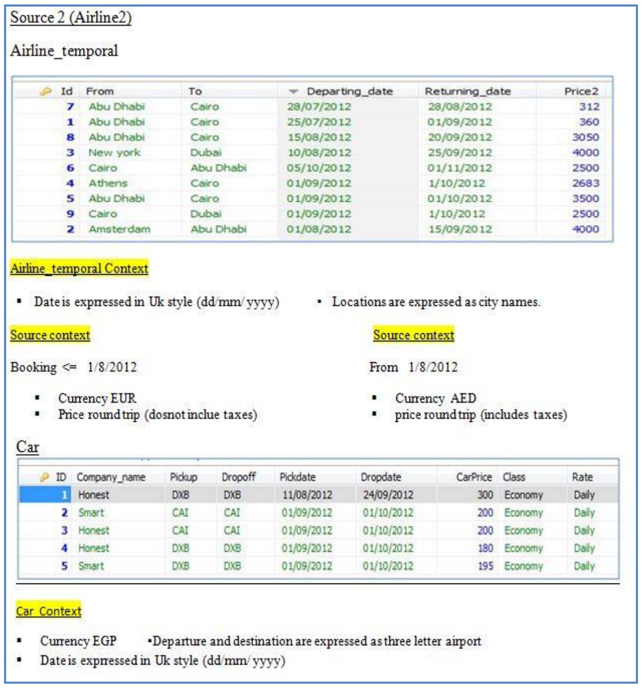
Car Rental

ID	Company_name	Pickup	Dropoff	Pickdate	Dropdate	CarPrice	Class	Rate
1	Honest	DXB	DXB	11/08/2012	24/09/2012	300	Economy	Daily
2	Smart	CAI	CAI	01/09/2012	01/10/2012	200	Economy	Daily
3	Honest	CAI	CAI	01/09/2012	01/10/2012	200	Economy	Daily
4	Honest	DXB	DXB	01/09/2012	01/10/2012	210	Economy	Daily
5	Smart	DXB	DXB	01/09/2012	01/10/2012	250	Economy	Daily

Car Rental Context

- Date is exprressed in Uk style (dd/mm/ yyyy) • Currency EGP
- Departure and destination are expressed as three letter airport

Fig.6. Data assumptions in source1 (Airline1)



Source 2 (Airline2)

Airline_temporal

Id	From	To	Departing_date	Returning_date	Price2
7	Abu Dhabi	Cairo	28/07/2012	28/08/2012	312
1	Abu Dhabi	Cairo	25/07/2012	01/09/2012	360
8	Abu Dhabi	Cairo	15/08/2012	20/09/2012	3050
3	New york	Dubai	10/08/2012	25/09/2012	4000
6	Cairo	Abu Dhabi	05/10/2012	01/11/2012	2500
4	Athens	Cairo	01/09/2012	1/10/2012	2683
5	Abu Dhabi	Cairo	01/09/2012	01/10/2012	3500
9	Cairo	Dubai	01/09/2012	1/10/2012	2500
2	Amsterdam	Abu Dhabi	01/08/2012	15/09/2012	4000

Airline_temporal Context

- Date is exprressed in Uk style (dd/mm/ yyyy) • Locations are expressed as city names.

Source context

Booking <= 1/8/2012

- Currency EUR
- Price round trip (dosnot inclue taxes)

Source context

From 1/8/2012

- Currency AED
- price round trip (includes taxes)

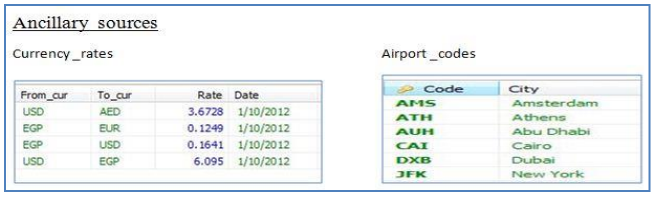
Car

ID	Company_name	Pickup	Dropoff	Pickdate	Dropdate	CarPrice	Class	Rate
1	Honest	DXB	DXB	11/08/2012	24/09/2012	300	Economy	Daily
2	Smart	CAI	CAI	01/09/2012	01/10/2012	200	Economy	Daily
3	Honest	CAI	CAI	01/09/2012	01/10/2012	200	Economy	Daily
4	Honest	DXB	DXB	01/09/2012	01/10/2012	180	Economy	Daily
5	Smart	DXB	DXB	01/09/2012	01/10/2012	195	Economy	Daily

Car Context

- Currency EGP •Departure and destination are expressed as three letter airport
- Date is exprressed in Uk style (dd/mm/ yyyy)

Fig.7. Data Assumptions in Source2 (Airline2)



Ancillary sources

Currency_rates

From_cur	To_cur	Rate	Date
USD	AED	3.6728	1/10/2012
EGP	EUR	0.1249	1/10/2012
EGP	USD	0.1641	1/10/2012
USD	EGP	6.095	1/10/2012

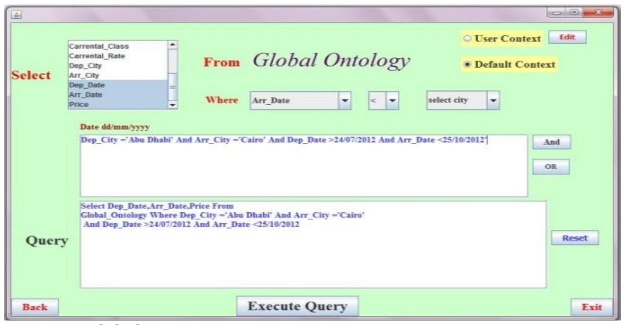
Airport_codes

Code	City
AMS	Amsterdam
ATH	Athens
AUH	Abu Dhabi
CAI	Cairo
DXB	Dubai
JFK	New York

Fig.8. Ancillary Tables

According to Firat [1] we mentioned that there are three dimensions of semantic heterogeneity: contextual, ontological and temporal. Now we describe how our proposed framework (SCR) can reconcile these conflicts.

Suppose the user submit the following global query1 to the SCR system in order to know the available airlines along with their prices.



Date dd/mm/yyyy

Dep_City ='Abu Dhabi' And Arr_City ='Cairo' And Dep_Date >24/07/2012 And Arr_Date <25/10/2012

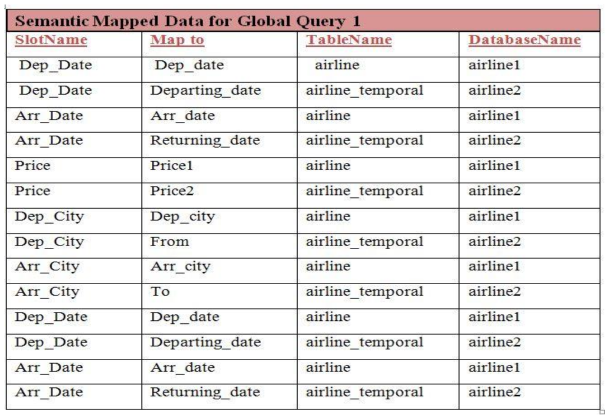
Select Dep_Date,Arr_Date,Price From
Global_Ontology Where Dep_City ='Abu Dhabi' And Arr_City ='Cairo'
And Dep_Date >24/07/2012 And Arr_Date <25/10/2012

Fig.9. Global query1

1. The GUI displays the global ontology terms to facilitate finding the global query terms easily and quickly. User browses the global ontology to selects specific terms and conditions for query1 as shown in Figure 9.

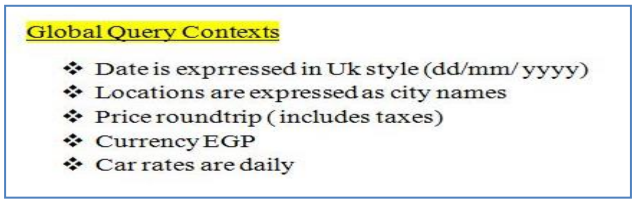
Global Query1 Before Conversion
Select Dep_Date,Arr_Date,Price From Global_Ontology Where Dep_City ='Abu Dhabi' And Arr_City ='Cairo' And Dep_Date >24/07/2012 And Arr_Date <25/10/2012

2. Query preprocessor utilizes semantic catalog (mapping file) to retrieve the semantic mapped data for global query1 that consists of database name, table name and columns names that mapped to each selected term and condition in the user global query1 and reorganizes the retrieved data into blocks according to the database name.



Semantic Mapped Data for Global Query 1			
SlotName	Map to	TableName	DatabaseName
Dep_Date	Dep_date	airline	airline1
Dep_Date	Departing_date	airline_temporal	airline2
Arr_Date	Arr_date	airline	airline1
Arr_Date	Returning_date	airline_temporal	airline2
Price	Price1	airline	airline1
Price	Price2	airline_temporal	airline2
Dep_City	Dep_city	airline	airline1
Dep_City	From	airline_temporal	airline2
Arr_City	Arr_city	airline	airline1
Arr_City	To	airline_temporal	airline2
Dep_Date	Dep_date	airline	airline1
Dep_Date	Departing_date	airline_temporal	airline2
Arr_Date	Arr_date	airline	airline1
Arr_Date	Returning_date	airline_temporal	airline2

3. SQL Generator uses the semantic catalog (metadata) to translate the previous created blocks received from the query preprocessor into SQL correct syntax.
4. The created queries are named a naïve queries, because the direct execution of them would not respect the semantic conflicts and would most likely return empty or semantically incorrect answers. If the created queries from the global query1 submitted to Airline1 and Airline2 databases without any mediation (conversion) would return empty results from source1 and semantically incorrect result set from source2.
5. Before sending the queries to the suitable data source the query engine connects to the merged ontology and compares the annotation values between the global query context (Figure 10) and the data sources contexts (Figure 6 and Figure 7) that involved in global query1.



Global Query Contexts

- ❖ Date is exprressed in Uk style (dd/mm/ yyyy)
- ❖ Locations are expressed as city names
- ❖ Price roundtrip (includes taxes)
- ❖ Currency EGP
- ❖ Car rates are daily

Fig.10. The Global Query Context

6. The SCR query engine detects the following semantic conflicts

<u>Contextual Heterogeneity</u>: As we mentioned before, Contextual heterogeneity occurs when different systems (sender/receiver) make different assumptions about the representation of the same concept) so there will be two or more not identical representations of the same thing. Such as in query 1 there are different representation of date format (UK vs. US Date Format) and locations (city name vs. airport code). The query engine detects two Contextual conflicts between the created SQL query contexts (Figure 10) and source 1 contexts:

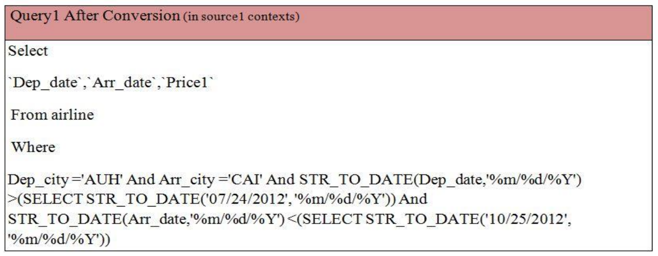
Date is expressed in US style (mm/dd/yyyy) in source 1 (Airline) and in Uk style (dd/mm/ yyyy) in the query context. This type of conflict can be solved in the mediation step using direct conversion from UK to US .

Cities represented as three letters airport code in source 1 (Airline) and as city full names in the query context. This type of conflict requires auxiliary tables in order to reconcile them as shown in Figure 8.

7. The converter connects to the conversion functions in order to reconcile the previous contextual conflicts at the query time by mediated queries then directs each query to the suitable database.

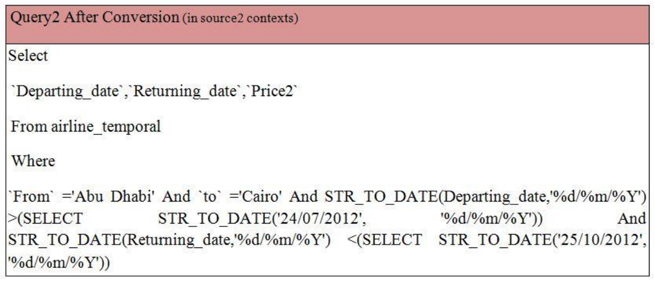
City name conflicts were dynamically reconciled, with the help of Airport_codes table that is used as ancillary table to convert from full city names to airport codes.

Date conflicts were statically reconciled by converting date values from Uk style (dd/mm/ yyyy) to US style (mm/dd/yyyy) using the direct conversion operation.



Query1 After Conversion (in source1 contexts)

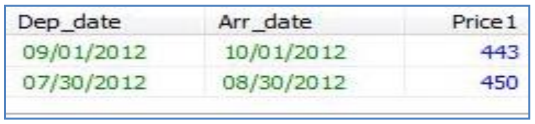
```
Select
`Dep_date`,`Arr_date`,`Price1`
From airline
Where
Dep_city ='AUH' And Arr_city ='CAI' And STR_TO_DATE(Dep_date,'%m/%d/%Y')
>(SELECT STR_TO_DATE('07/24/2012', '%m/%d/%Y')) And
STR_TO_DATE(Arr_date,'%m/%d/%Y') <(SELECT STR_TO_DATE('10/25/2012',
'%m/%d/%Y'))
```



Query2 After Conversion (in source2 contexts)

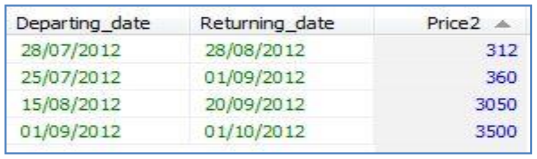
```
Select
`Departing_date`,`Returning_date`,`Price2`
From airline_temporal
Where
`From` ='Abu Dhabi' And `to` ='Cairo' And STR_TO_DATE(Departing_date,'%d/%m/%Y')
>(SELECT STR_TO_DATE('24/07/2012', '%d/%m/%Y')) And
STR_TO_DATE(Returning_date,'%d/%m/%Y') <(SELECT STR_TO_DATE('25/10/2012',
'%d/%m/%Y'))
```

8. Now If queries after the previous conversion submitted to the suitable database would return semantically incorrect results from source 1 (Figure 6) and from source 2 (Figure 7) because the retrieved results not respect the user expectations (user required context).



Dep_date	Arr_date	Price1
09/01/2012	10/01/2012	443
07/30/2012	08/30/2012	450

Fig.11. Source1 Semantically Incorrect Results of Query1

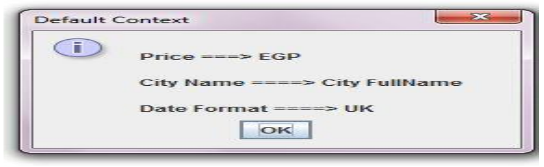


Departing_date	Returning_date	Price2
28/07/2012	28/08/2012	312
25/07/2012	01/09/2012	360
15/08/2012	20/09/2012	3050
01/09/2012	01/10/2012	3500

Fig.12. Source2 Semantically Incorrect Results of Query1

9. After directing each query to the suitable data source, then query engine compares the annotation values of each item in the sources with the user required contexts.

In global query 1 our user selects to display the query results in the default context.


Default Context

Price ===> EGP

City Name ====> City FullName

Date Format ====> UK

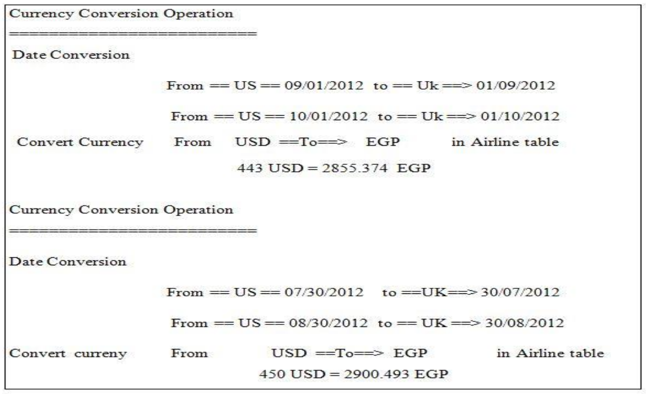
OK

Query1 Semantic Conflicts (Airline1):

The query engine detects two contextual conflicts between the user contexts (default) and source1 contexts

Contextual conflict in the price concept, currency modifier value in source1 is USD context but in EGP in the user context. Such conflict detected and reconciled through the mediated query. Date is expressed in US style (mm/dd/yyyy) in source1 (Airline) but our user expects it as UK style (dd/mm/ yyyy).

The previous contextual conflicts reconciled on the query time by the converter through the mediated queries. Figure 13 shows a Trace of reconciling source 1 detected contextual semantic conflict as follow.


```
Currency Conversion Operation
==========================
 Date Conversion
                               From == US == 09/01/2012  to == Uk ==> 01/09/2012
                                From == US == 10/01/2012  to == Uk ==> 01/10/2012
  Convert Currency      From     USD ==To==>    EGP         in Airline table
                                         443 USD = 2855.374  EGP

Currency Conversion Operation
==========================
Date Conversion
                               From == US == 07/30/2012    to ==UK==> 30/07/2012
                                From == US == 08/30/2012  to == UK ==> 30/08/2012
Convert  curreny        From          USD  ==To==>  EGP            in Airline table
                                              450 USD = 2900.493 EGP
```

Fig.13. Trace of reconciling source1 contextual semantic conflict

Query 1 Semantic Conflicts (Airline 2):

The query engine detects semantic mismatches among contexts caused by the implicit assumptions. The implicit assumptions not only differ between the two sources, but also changed in the same source from one context to another over time as shown in the airline_temporal table in Figure 7.
The user expects the price always be in EGP and with taxes included. While in source 2 (airline_temporal table) the price is in EUR and excluding taxes if departing date <= 01/08/2012 and changed the currency from EUR to AED and price includes taxes if departing date > 01/08/2012 , Such conflicts lead us to the third dimension of semantic heterogeneity which is the temporal dimension).

A. Reconciling Temporal Semantic Heterogeneity:

Temporal semantic heterogeneities are related to both contextual and ontological heterogeneities and occur in situation where the semantics between data sources, even in the same data source, change over time [11]. Both the representational and the ontological assumptions can be static and do not change over time within an interested time period, in which case time is not of concern (same previous examples) or, the assumptions can change over time, and the resulting heterogeneity is temporal. When the implicit assumptions change over time, data corresponding to different time periods are subject to different interpretations. Based on the previous definition we have two categories of temporal heterogeneity, temporal representational (contextual) heterogeneity and temporal ontological heterogeneity.

It's challenging to deal with data if it's meaning changes overtime. These challenges will be more difficult if we want to handle the changes through the integration of multiple heterogeneous sources. We describe the implicit assumptions about data elements in temporal concepts by similar way of describing non temporal concepts by adding annotations (modifiers) in the merged ontology for each term.
We identify the following approach to explicitly describe the temporal assumptions in the knowledge representation phase and reconciling the temporal semantic conflicts through the interpretation mediation service phase.
Proposed Approach for Representing Temporal Contexts
Temporal assumptions approach: Describing each data element (attribute) in a data source that has assumptions change over time with different ontological concepts at different times. So we convert temporal assumptions to set of static assumptions within an interested time period.

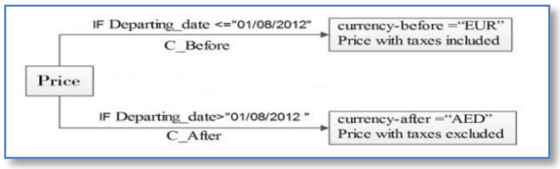

IF Departing_date <="01/08/2012"
C_Before
currency-before ="EUR"
Price with taxes included
Price
IF Departing_date>"01/08/2012 "
C_After
currency-after ="AED"
Price with taxes excluded

Fig.14. Temporal Assumptions Approach

In query 1 we explain the representation of temporal contexts in the ontology according to our proposed approach as shown in Figure 14.

Reconciling Temporal Representational Conflicts of Query1

In temporal representational assumptions the same attribute may be represented differently at different times of a data source according to our previous approach.
In order to resolve the temporal contextual conflict between our user context of query 1 and the source 2 context of airline_temporal table as shown in Figure 7. We explicitly describe the price data element in the ontology by two contexts instead of describing each attribute by one modifier context as in atemporal concepts. In query 1 the user expects the currency always be in EGP. While in source 2 (

airline_temporal table) the currency is in EUR if departing date <= "01/08/2012" and changed from EUR to AED when departing date > "01/08/2012".In Source 2 we link the price attribute with C_Before context if the value of Departing_date attribute <= "01/08/2012"and with C_After context if the value of Departing_date attribute >"01/08/2012"

In C_Before context: modifier currency has a value of "EUR"

In C_After context: modifier currency has a value of "AED"

Reconciling Temporal Ontological Conflicts of Query2

In temporal ontological assumptions the ontological concept represented by an attribute may change over time. In airline_temporal table of source 2, Price on and before "01/08/2012" includes taxes, afterwards it excludes taxes .In figure 7 we have two different assumptions (interpretation) for Price (including and excluding taxes). According to our proposed approach we create a more general concept in the ontology that includes both variations as a special cases based on the value of the modifier limit.

In T_Inclusion context: Price (+tax)

In T_Exclusion context: Price (nominal).

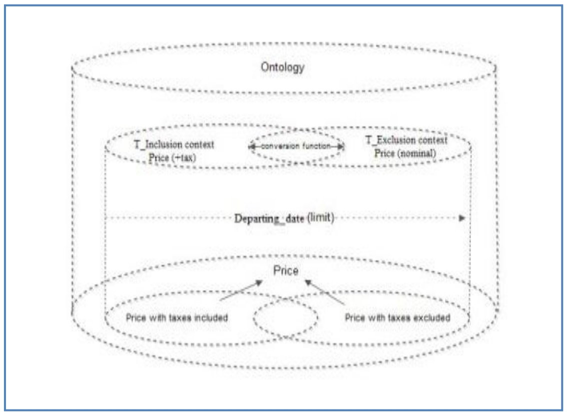
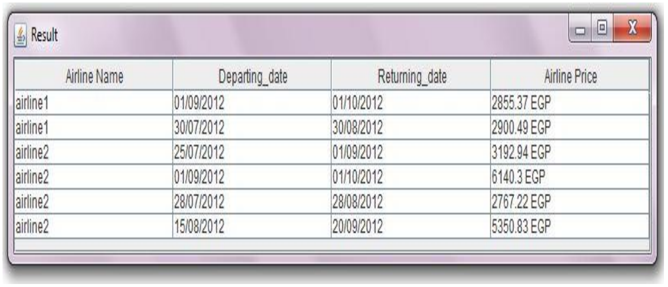


Fig.15. Temporal Ontological Representation

SCR query engine able to handle terms in the ontology that can be expressed based on explicitly define contexts of other terms or adjusting the value of one context to another after assigning the required equations with each context (reconciling equational semantic conflicts).

In source 2 there is an equational conflict between T_Inclusion context and T_Exclusion context so we use the conversion functions associated with each context in the mediated queries in order to reconciling such a conflict.

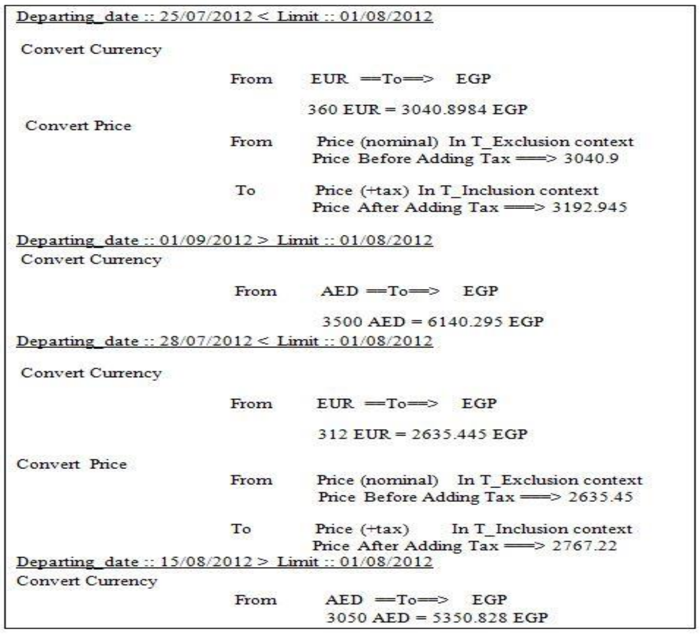
- To reconcile the previous temporal semantic conflicts the converter checks the value of modifier limit in the ontology and the corresponding currency and price modifiers values in order to compare them with the user required context.
- In C_Before context : modifier currency has a value of "EUR"
- In C_After context: modifier currency has a value of "AED"
- In T_Inclusion context : Price (+tax)
- In T_Exclusion context: Price (nominal).



Airline Name	Departing_date	Returning_date	Airline Price
airline1	01/09/2012	01/10/2012	2855.37 EGP
airline1	30/07/2012	30/08/2012	2900.49 EGP
airline2	25/07/2012	01/09/2012	3192.94 EGP
airline2	01/09/2012	01/10/2012	6140.3 EGP
airline2	28/07/2012	28/08/2012	2767.22 EGP
airline2	15/08/2012	20/09/2012	5350.83 EGP

Fig.16. Query1 Final Results

Figure 17 shows trace of reconciling the detected temporal semantic conflicts in Airline2 as follow.



```
Departing_date :: 25/07/2012 < Limit :: 01/08/2012

Convert Currency
                         From     EUR  ==To==>   EGP
                                  360 EUR = 3040.8984 EGP
Convert Price
                         From     Price (nominal) In T_Exclusion context
                                  Price Before Adding Tax ==> 3040.9

                         To       Price (+tax) In T_Inclusion context
                                  Price After Adding Tax ==> 3192.945
Departing_date :: 01/09/2012 > Limit :: 01/08/2012
Convert Currency
                         From     AED ==To==>   EGP
                                  3500 AED = 6140.295 EGP
Departing_date :: 28/07/2012 < Limit :: 01/08/2012

Convert Currency
                         From     EUR  ==To==>   EGP
                                  312 EUR = 2635.445 EGP
Convert  Price
                         From     Price (nominal)   In T_Exclusion context
                                  Price Before Adding Tax ==> 2635.45

                         To       Price (+tax)      In T_Inclusion context
                                  Price After Adding Tax ==> 2767.22
Departing_date :: 15/08/2012 > Limit :: 01/08/2012
Convert Currency
                         From     AED  ==To==>   EGP
                                  3050 AED = 5350.828 EGP
```

Fig.17. Trace of reconciling temporal semantic conflicts in Airlin2

V. CONCLUSION

We developed an ontology-based approach, in which all data semantics explicitly described in the knowledge representation phase and automatically taken into account by Interpretation Mediation Services phase, conflicts detected and resolved automatically at the query runtime. Unlike the traditional approaches there is no need for any changes for the sources involved in the integration process even if these sources with time-varying semantics, each source should only explicitly records its semantic assumptions in addition to a small number of conversion functions, which are used by the converter for automatically make required conversions. The SCR preserve local autonomy for each data source to change and maintain independently. Data sources still independent from the integration process that is mean we can retrieve up to date data and smoothly update the data in each data source without affecting the integration process. The SCR framework provides a systematic methodology for explicitly describing assumptions through the knowledge representation phase. Changes in sources contexts can be accommodated by modifying annotation values without affecting both the global ontology and the conversion functions (no hand-code needs to be maintained).The proposed SCR framework assumes that the underlying information sources are structured data, testing our proposed approach for detecting and reconciling semantic conflicts using semi-structured data such as web pages is required to identify new challenging issues. We have demonstrated the capability of the SCR framework using simple illustrative example. It is interesting to ensure interoperability and knowledge-based information sharing in real world environments of fertile fields like bioinformatics, digital libraries and GIS systems in order to test the feasibility of our approach.

REFERENCES

[1] Firat, A.(2003). Information Integration Using Contextual Knowledge and Ontology Merging. Ph.D. Thesis. Massachusetts Institute of Technology.

[2] Madnick S., Gannon T., Zhu, H., Siegel M., Moulton A., Sabbouh M.,(2009):" Framework for the Analysis of the Adaptability, Extensibility, and Scalability of Semantic Information Integration and the Context Mediation Approach", IT.

[3] Madnick, S.E., & Zhu, H. (2006) .Improving Data Quality with Effective Use of Data Semantics. *Data and Knowledge Engineering, 59*(2), 460-475.

[4] Noy, N. F. and Musen, M. A., PROMPT: Algorithm and tool for automated ontology merging and alignment, National Conference on Artificial Intelligence - AAAI , pp. 450-455, 2000.

[5] Nyulas, M. Connor, S. Tu, DataMaster_a plug in for Relational Databases into protégé, Stanford University, 10th international protégé conference, 2007.

[6] Ram ,S., Park, J., Semantic Conflict Resolution Ontology (SCROL): A Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts, IEEE Transactions on Knowledge and Data Engineering, v.16 n.2 , p.189-202, 2004.

[7] Rosenthal, A., Seligman, L. and Renner, S. From Semantic Integration to Semantics Management: Case Studies and a Way Forward, ACM SIGMOD Record, 33(4), 44-50, 2004.

[8] Sultan, T. I., Nasr, M. M. , Khedr, A. E., & Ismail , W., S. (2013) " Semantic Conflicts Reconciliation (SCR): A Framework for Detecting and Reconciling Data-Level Semantic Conflicts" , International Journal of Engineering Research and Applications (IJERA),ISSN: 2248-9622 , Volume 3, Issue: 1, pp.766-773, India.

[9] Zhu, H., & Madnick, S. E. (2004) "Context Interchange as a Scalable Solution to Interoperating Amongst Heterogeneous Dynamic Service", *3rd Workshop on E-Business*. Washington, D.C., 150-161.

[10] Zhu, H., (2005): Effective Information Integration and Reutilization: Solutions to Deficiency and Legal Uncertainty, PhD Thesis, Massachusetts Institute of Technology Cambridge, MA, USA.

[11] Zhu, H., Madnick, S., Reconciliation of temporal semantic heterogeneity in evolving information systems", ESD-WP-2009-03, Massachusetts Institute of Technology Cambridge, MA, USA, 2009