Towards a Context Ontology to Enhance Data Integration Processes

Damires Souza
Federal University of Pernambuco
PO Box 7851
50.732-970 Recife, PE, Brazil
+55 81 3453 9213
dysf@cin.ufpe.br

Rosalie Belian
Federal University of Pernambuco
PO Box 7851
50.732-970 Recife, PE, Brazil
+55 81 2126 8430
rbb@cin.ufpe.br

Ana Carolina Salgado
Federal University of Pernambuco
PO Box 7851
50.732-970 Recife, PE, Brazil
+55 81 2126 8430
acs@cin.ufpe.br

Patricia Tedesco
Federal University of Pernambuco
PO Box 7851
50.732-970 Recife, PE, Brazil
+55 81 2126 8430
pcart@cin.ufpe.br

ABSTRACT
Data Integration has become one of the most relevant research fields in data management for the last years. The problem of integrating data from diverse, heterogeneous data sources is ubiquitous and has been tackled by some different approaches. A recent one concerns Peer Data Management Systems (PDMS) which are characterized by their dynamicity. To help matters, semantic information like *context* may be employed to ease some processes in DI: schema reconciling and query processing. However, dealing with contextual information entails a high development cost because several tasks (e.g. context acquisition and processing) must be performed. In order to provide means for that, first we have to define how to better represent contextual information. In that respect, ontologies are an interesting approach since they enable reasoning, reusability and knowledge sharing. In this paper, we propose CODI - a Context Ontology to formally represent *context* in Data Integration processes. We also present a case study illustrating how CODI can be used to enhance query processing in a PDMS environment, so that users will get more complete and relevant results.

1. INTRODUCTION
The problem of Data Integration (DI) is a pervasive challenge faced by applications that need to query across multiple autonomous and heterogeneous data sources [7]. The environment in which a data integration system operates is very dynamic and the system deals with much less information and control than in a traditional database setting. Consequently, it is more difficult to plan some tasks. As an illustration, it is hard for the system to decide on a good query execution plan, since it may not have enough information previously, and, at the same time, a plan that looks good initially, may be arbitrarily bad if the sources do not respond exactly as expected.

In order to better address tasks as query processing and schema reconciling, we need more semantics and control that may be only acquired on the fly. To face these issues, we propose the use of *Context*, i.e. the circumstantial elements that make a situation unique and comprehensible [4], as a way to provide more precise semantics, control information and reasoning as well.

We are able to understand *context* when identifying how humans use it in practice. Humans seem to be able to build complex contexts instinctively [10]: first context is recognized and understood; then the relevant set of properties (e.g. location, interests) required to deal with that context is automatically assembled. Thus, we define *Context* as a set of elements surrounding a domain entity of interest which are considered relevant in a specific situation during some time interval. The *domain entity* of interest may be a person, a procedure, a file, a set of data or even an inter-schema mapping. Furthermore, we use the term *contextual element* (CE) referring to pieces of data, information or knowledge that can be used to define the *Context*, in accordance with the definition provided by Vieira *et al*. [18].

In DI, *context* has been mainly used to represent different understanding of data and schema elements [9]. We argue that context may be used in a broader way to improve data integration processes. In this sense, our goals when using context are twofold: (i) to ease schema reconciling, trying to identify in which context the elements occur and determining the semantic affinity between them and; (ii) to enhance query processing capabilities, providing users with more meaningful and complete answers according to the context acquired at query submission and execution time. More specifically, in this work we focus on using *context* to improve query processing capabilities in a Peer Data Management System (PDMS) environment.

Nevertheless, an important issue in using context is how to represent its elements [3, 13, 19]. A challenge to be faced is the fact that there is not a standard model for representing it yet. Context ontologies have been considered an interesting approach because they enable sharing and reusability and may be used by different reasoning mechanisms [13, 19]. Hence, in this paper, we present CODI - an ontology to represent CEs in the data integration realm. Through this model, it is possible to compose inference rules that enable the discovery of high-level (complex, implicit) context from low-level (basic, explicit) context. To clarify matters, we present a case study illustrating how the proposed ontology can be used to enhance a query execution process in PDMS.

This paper is organized as follows: Section 2 discusses context in the light of Data Integration; Section 3 introduces a motivating scenario; Section 4 describes our proposal, presenting the identified domain entities and their respective contextual elements and Section 5 shows the proposed CODI in practice. Related work is discussed in Section 6. Finally, Section 7 draws our conclusions and points out some future work.

2. CONTEXT IN DATA INTEGRATION

The use of context in Data Integration systems is quite different from other context-sensitive applications. Integrating heterogeneous data sources requires solving schematic and semantic conflicts which may arise at schema or instance-level [6, 9, 13, 14, 15]. Some of the metadata that describe the data sources may be used as contextual information (e.g. the data scale). Other contextual elements are perceived or inferred dynamically during the execution of a given process (e.g. in query processing, the availability of data sources is perceived at run time). Thus, in DI, context may be used to ease two main issues: schema reconciling and query processing.

A schema reconciling operation receives a set of distinct data source schemas, with varying structures and semantics, and produces a set of mappings among semantically related schema elements [11]. A process for schema reconciling usually executes the following tasks [11]: i) a preprocessing routine that translates schemas into a common format and makes schema element names processable; and ii) a schema matching and mapping routine that produces inter-schema mappings. Element names can have different meanings depending on the semantic context to which they are related. Hence, CEs may provide a more accurate semantic interpretation, allowing restrictions or characterizations of an element name according to a specific semantic context. For instance, two entities having equal names may refer to different real world objects and then need to be considered semantically dissimilar (e.g. entity1 referring to *block* in a *city* context and entity2 referring to *block* in a *building* context, considering a *geographic knowledge domain*). In this case, there is a semantic dissimilarity relation between these terms that needs to be considered when interpreting meaning. This semantic relation may be identified by considering the entities to which the *block* entities are connected in their schemas or even by the kind of application that are using them. Likewise, contextual knowledge may be used in reconciling structural differences in order to make the necessary transformations to turn data source elements capable of being integrated (e.g. to integrate data from two data sources with different scales).

The other main issue in DI is query processing. In this paper, we focus on query processing in a PDMS. PDMS represent a natural step beyond data integration systems, replacing their single logical schema with an interlinked collection of semantic mappings between peers' individual schemas [8, 16]. A PDMS, as a P2P system itself, keeps the properties of all P2P systems, e.g., every peer may join and leave the network at any time; moreover, all peers are autonomously created and managed. PDMS are intended to be used for query answering and information sharing, but, to this end, their dynamicity must be dealt with accordingly. In this light, context is used as a way to deal with such dynamicity. Therefore, a context-based query execution process in a PDMS is usually accomplished by the following steps:

(1) Query Submission: whenever the user poses a query, the current values of user, PDMS and query's context are acquired. For example, user preferences, query language, user interface and submission peer's identification are acquired.

(2) Query Analysis: the query is analyzed in order to identify essential features and objectives. For instance, its required entities, attributes and operators are query context elements that are discovered in this step.

(3) Relevant Peers' Establishment: in our work, relevant peers are the ones which are neighbors (i.e. connected through a semantic path of mappings) of the submission peer and, at the same time, are able to provide answers for a given query. Thus, in this step, the submission peer neighbors context (data model, peer's availability, whether or not it can apply the required operators) are analyzed to help determine which ones are really relevant.

(3) Query Reformulation: semantic inter-schema mappings between peers are also considered contextual information since they are rather dynamic and their application may produce different query rewritings. According to such mappings, the submitted query is rewritten into another to be executed in a relevant peer.

(4) Query Execution and Answer Integration: each relevant peer executes the query and returns its result to the initial submission peer. Then, this peer analyzes the query's context in order to integrate the produced answers. Also, queries' context elements may be stored in a knowledge base for later recovery. As a result, historical context data will be maintained according to user interaction's trajectories or to query processing steps to help to predict users' needs or establish trends in query processing.

(5) Result Presentation: the query result can be presented in various forms according to the user's preference, query interface and intended usage. The final result is presented by the submission peer where the original query was formulated.

In summary, context usage in query processing may entail three important benefits: (i) it enables the analysis of the user's query through its interpretation and identification of related entities and necessary operators on the fly; (ii) it helps to identify relevant peers that may contribute with answers to a given query, thus improving query processing results and; (iii) since the effects of collecting and integrating content from various sources need to be handled, context may enrich the post-processing of the retrieved answers to adjust the final result representation according to the user preferences or intended level of detail.

3. A MOTIVATING SCENARIO

Water is one of the most important resources on Earth. In most Brazilian regions, water is abundant in rivers and lakes, although in other areas it is not sufficient to provide benefits to habitants. In this sense, our motivating example is concerned with the *Brazilian Hydrographic System* which has been developed in a PDMS environment. For the sake of simplicity, we only consider two peers **A** and **B** which store geospatial data sources, depicted in Figure 1. Peer A is at scale of 1:1000'000, while peer B is more detailed and is at scale of 1:250'000. In addition, Peer A contains three classes – *Lake*, *StreamofWater* and *Town* which inherit some characteristics from *Geographical_entity*. The three classes have a geometry attribute. Peer B contains *Lake*, *River* and *City* which are subclasses of *Basic_geo_entity*. These classes have a *shape* attribute.

In this scenario, some conflicts arise due to the heterogeneity of the peers. The semantic conflicts related to schema level are: (1) different entity names – *Geographical_entity* vs. *Basic_geo_entity*, *StreamofWater* vs. *River* and *Town* vs. *City*; (2) different attribute names – *geometry* vs. *shape*; and also different data types – *integer* vs. *string* (GID) and *point* vs. *polygon* (lake, and town and city). These conflicts are resolved in schema reconciling time, when inter-schema mappings are identified. Other relevant conflicts (found in query processing) are the instance level ones. Here we have different *scales* 1:1000'000 (Peer A) vs. 1:250'000 (Peer B) and the *multi-representation* problem, since *lake* is represented by a *point* in Peer A and by a *polygon* in Peer B. Also, both peers are considered to be vector, but, in fact, real geospatial data sets may be vector or raster, which raises complexity and may entail format conversions.

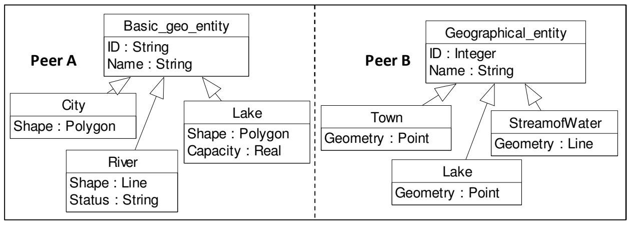


Figure 1. Schemas for Peers A and B

For instance, consider that a user poses the query "*Select Capacity, Area from Lake;*" as s/he is looking for all the lake capacities and their corresponding areas. To answer this query, all contextual elements around its formulation are considered. Firstly a mapping such as *A.Lake* ≡ *B.Lake* (suppose it is already generated) is observed. This means that both entities are semantically equivalent. However, in Peer A, *Lake* is represented as a *Polygon* and, in Peer B, as a *Point*. As a result, Peer B is not able to answer the query since it is not possible to calculate the *area* over a *Point* representation. In this example (considering only the two peers), the answer will be retrieved by Peer A, according to the user application scale and his/her preferences.

4. CODI – A CONTEXT ONTOLOGY FOR DATA INTEGRATION

CODI (**C**ontextual **O**ntology for **D**ata **I**ntegration) is an ontology for representing *context* according to the DI issues discussed in Sections 1 and 2. In order to establish the relevant contextual elements (CEs), at first we have identified the domain entities that we needed to work with. A domain entity is anything in the real world that is relevant to describe the domain (e.g data sources, users and applications) [18]. In our work, we consider that CEs are used to characterize a given domain entity. Therefore, we determined six main domain entities around which we consider the CEs: *user*, *environment*, *data*, *procedure*, *association* and *application*. To figure out these domain entities and their related CEs, our approach has been guided by a participatory and incremental design methodology. The ontology was developed during a series of face-to-face meetings between DI experts who are concerned with issues such as schema reconciling, query processing, Data Integration Systems, PDMS and reasoning. Furthermore, we have also examined systematically in the literature some DI real systems and related problems. As a result, we draw the domain entities' concepts, their properties and more specifically the related contextual elements that would be relevant to deal with.

We present the domain entities' taxonomy as well as some contextual elements relevant to them in Figure 2. As a result, CODI is a conjunction of those domain entities and the CEs which are related to them. Moreover, Figures 3,4,5,6,7 and 8 describes the CEs that characterize each of the above mentioned domain entities. For the sake of space, we have converted the diagrams from Protégé's notation to UML[1]. In addition, we show the CEs in white and the domain entities in gray.

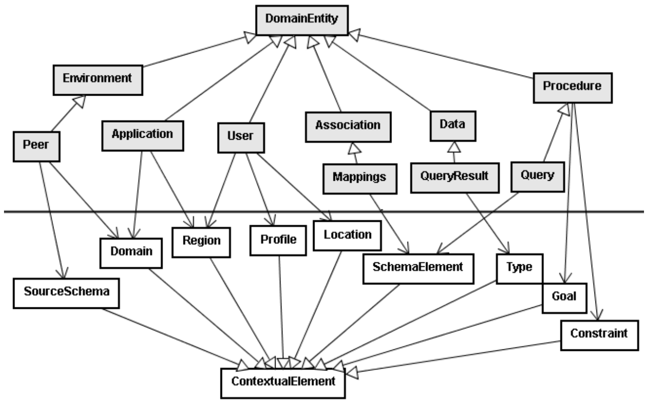


Figure 2. The Domain Entities' taxonomy and the overview of CODI's Contextual Elements

User: The CEs that make up a user's information context are concerned with his/her *profile, location*, *role*, *region* and query *interface* type (Figure 3). According to the user preferences and query interface type, the system may define, for instance, the way a query result should be presented. Also, user's CEs are used in schema reconciling to determine data types and scale for schema element representation.

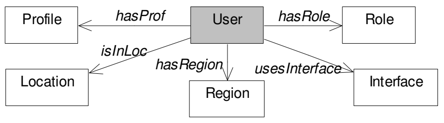


Figure 3. CEs for the User domain entity

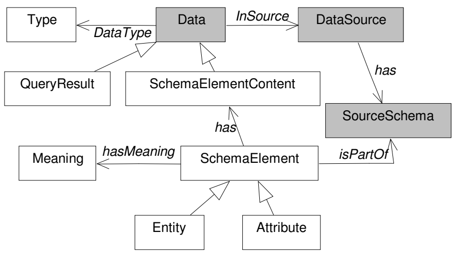


Figure 4. CEs for the Data domain entity

[1] Unified Modeling Language

Association: associations are important to characterize relations between elements and are used by tasks such as query processing (Figure 5). In our work, associations are *mappings* which represent existing relationships among schema elements and may indicate how data in one schema is to be transformed into data in another schema.

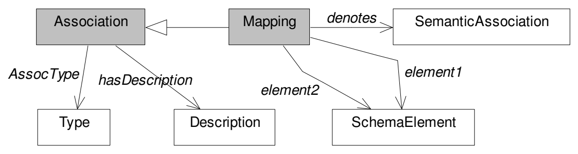


Figure 5. CEs for the Association domain entity

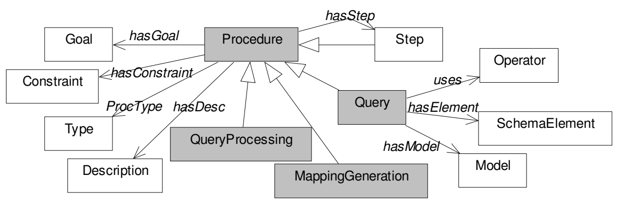


Figure 6. CEs for the Procedure domain entity

Procedure: in this case, the idea is to provide the contextualization of a procedure steps in order to help to solve a given problem. In our setting, a procedure may be the complete *mapping generation* process, a particular *Query* or the *Query Execution Process* as a whole (Figure 6). For instance, a *Query* is formulated within a search context, therefore, in addition to the inherited *Procedure*'s CEs, we have to identify: i) which kind of query *model* is being used; ii) which *schema elements* are necessary to work with; and iii) which *operators* are to be executed.

Application: each application has its particular features (Figure 7). For instance, an important CE to DI is the application *Domain*. Each *domain* has a *Vocabulary*, usually represented by a domain ontology and its specific *terms*. *Terms* and their related features are acquired from the related domain ontology.

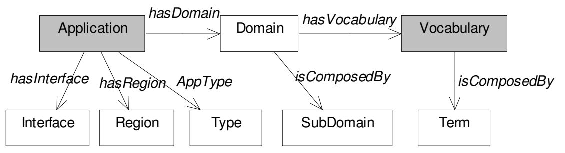


Figure 7. CEs for the Application domain entity

Environment: concerns the environment where the user interacts and the application is executed. In our work, it may be a Data Integration System (with a single global schema) or a PDMS (with mappings among peers' schemas), as shown in Figure 8. In fact, in both cases, we are dealing with dynamic and autonomous data sources that may join and leave the network at any time. Thus, environment CEs must be acquired on the fly (e.g. data source availability). In this sense, *Data Integration Systems*, *PDMS*, *data sources*, *peer* and *source schemas* are the domain entities from which the CEs will be acquired. In general, the main environment CEs are: *Type*, *Region*, *Platform* and *Condition*. Depending on the system (e.g. PDMS), other specific elements may be added or refined.

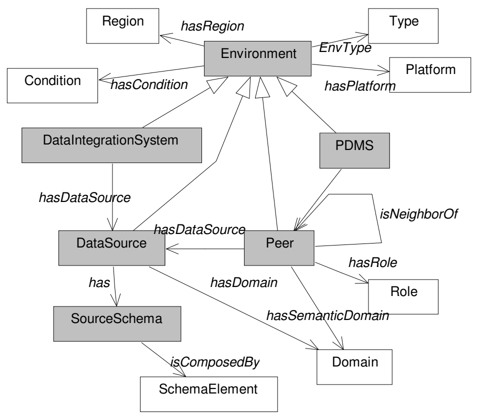


Figure 8. CEs for the Environment domain entity

CEs can either be explicit or implicit. An explicit CE is obtained from static sources, such as a profile (e.g. settings information). An implicit one is perceived in the surrounding dynamic environment or is derived through some reasoning process. For example, a spatial relationship (e.g. *touch*, *cross*, *distance*) is inferred through the analysis of two objects locations. Still, the scale a user is working with may be identified through his/her application parameters. Another illustration concerns the presentation of query results. A query's result set may contain different data representations, e.g. different unit formats that are used in the distributed data sources. Thus, depending on the context of query submission, a specific unit may be chosen and a conversion and merging process may be performed automatically. In other words, contextual information perceived or inferred through reasoning mechanisms may be used to adjust the result representation. For instance, a presentation preference for statistical results may specify different formats such as a summary table, trends diagram or a pie chart, thus a user may explicitly defines that s/he prefers a summary table rather than a trends diagram or the system may implicitly discover such information through user trajectories (historical information).

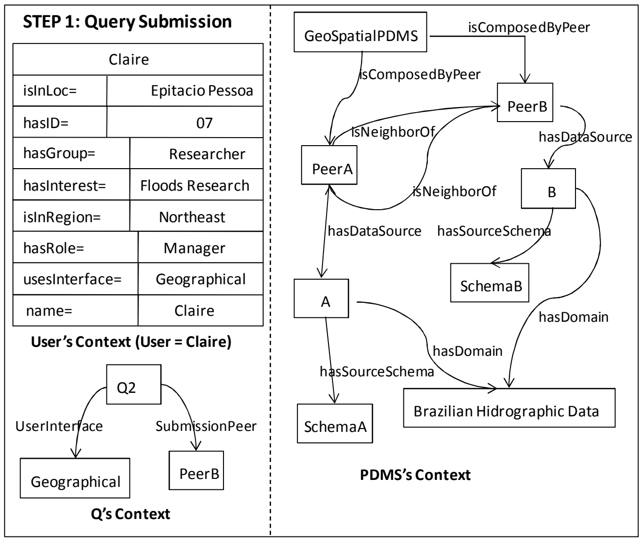
5. USING CODI

The main idea underlying our work is to use CODI to represent and to maintain the CEs related to DI. One of the advantages of using an ontology mechanism is the possibility of inferring new complex information from existing basic context. In this section, we present a query processing case study in the light of our motivating scenario. In such scenario, we assume that the inter-schema mappings have already been generated, so we are able to focus on query answering in general.

To better explain where and when the contextual information is used, we present CODI's usage for each one of the presented query execution process steps (Section 2). To this end, we provide views of CODI's instantiation which have been produced using OntoViz, a Protégé plug-in. In this format, instances are associated with their concepts through the ***io*** relationship and subtypes are associated with their supertypes through the ***isa*** relationship. The diagrams presented below are in fact fragments from the overall ontology, and do not show neither the whole class hierarchy nor the complete set of instances.

In this light, suppose that a user poses the following spatial query **Q**: "*SELECT R.Name, C.Name FROM River R, City C WHERE*

Cross(R.Shape,C.Shape)=1;". The topological spatial operator *Cross* $(geometry_1, geometry_2)$ is a Boolean operation which returns true if a $geometry_1$ intersects with another $geometry_2$. It can be applied to line/line, line/area, point/area, and point/line groups [5]. Thus, **Q**'s submission is done in Peer B and means: "For all the rivers, find the cities through which they pass".



STEP 1: Query Submission

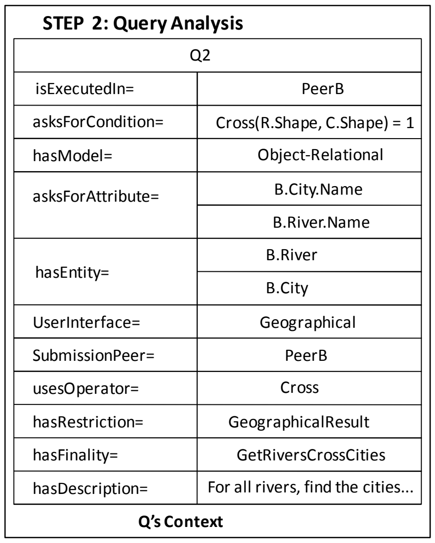
Claire	
isInLoc=	Epitacio Pessoa
hasID=	07
hasGroup=	Researcher
hasInterest=	Floods Research
isInRegion=	Northeast
hasRole=	Manager
usesInterface=	Geographical
name=	Claire

User's Context (User = Claire)

Figure 9. CEs at Query Submission Time

At submission time (step 1), some contextual information concerned with the user, the query and the environment are acquired or perceived as depicted in Figure 9. In this case, the user profile (group, role, interest, name), his/her location and the kind of interface he/she is using are CEs which are gathered. Also, information about the environment, i.e the PDMS, such as the composing peers and data sources, their schemas as well as their domain are important information that should be dealt with when the relevant peers are set. To this end, we have to know for example which peers are available, if they have a common knowledge domain and the existing elements in each peer's schema. Besides, as context of the query, it is observed where it has been submitted and what kind of interface has been used.

In step 2, the query is completely analyzed (Figure 10). Thereby, the required entities, spatial operators, attributes, constraints and conditions are gathered in order to identify the semantics of the query. As a result, this semantics will be taken into account to verify which peers are relevant for such query and how it can be better reformulated in these peers. For example, we consider that in our PDMS, when a submission peer P receives a given query Q, it identifies its semantics (through contextual elements) and creates a corresponding query graph. This graph is compared with the graphs representing the schemas of the peer's neighbors. If the query graph is subsumed by the neighbor's schema graph, then this neighbor is really relevant for such query.



STEP 2: Query Analysis

Q2	
isExecutedIn=	PeerB
asksForCondition=	Cross(R.Shape, C.Shape) = 1
hasModel=	Object-Relational
asksForAttribute=	B.City.Name
	B.River.Name
hasEntity=	B.River
	B.City
UserInterface=	Geographical
SubmissionPeer=	PeerB
usesOperator=	Cross
hasRestriction=	GeographicalResult
hasFinality=	GetRiversCrossCities
hasDescription=	For all rivers, find the cities...

Q's Context

Figure 10. CEs at Query Analysis Time

Next, in step 3, the peers that are considered relevant (in our example, *Peer A*) are also observed and their context acquired and used (Figure 11). For instance, we have to see if such peers are available for query reformulation and if they can execute the spatial operator that has been required, since not all of the DBMS are able to execute properly all the set of existing spatial operators.

Next step is reformulating query **Q** to a representation (a rewriting) that is compatible with each relevant peers' schemas. In this example, *Peer A* is relevant, so the process takes into account the mappings between *Peer A* and *Peer B* and rewrites **Q** into another query **QRef**. Figure 12 depicts some mappings which, for us, are treated as contextual information and are used to allow query rewriting. In fact, mappings are rather important in a PDMS's setting since peers may join or leave the system at free and thus, the assumption we can make about them is based on their mappings. Figure 13 presents the context of the reformulated query **QRef** in Peer A.

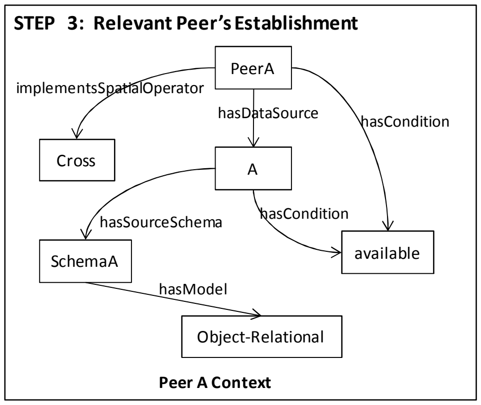
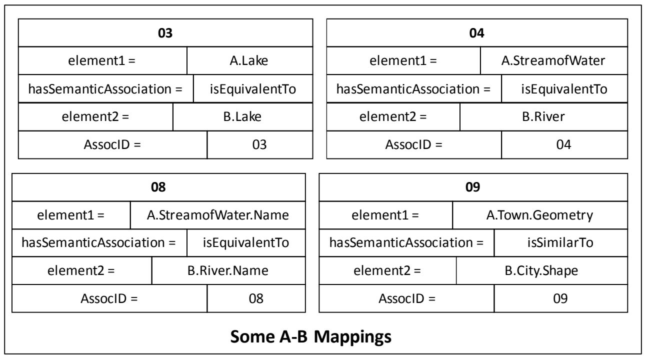


Figure 11. CEs at Relevant Peer's Establishment Time



03	
element1 =	A.Lake
hasSemanticAssociation =	isEquivalentTo
element2 =	B.Lake
AssocID =	03

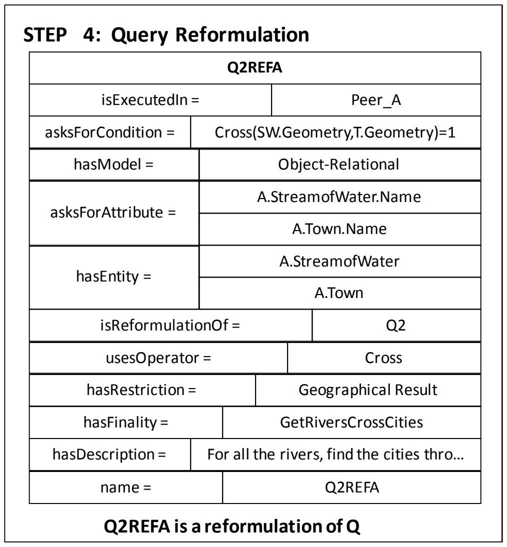
04	
element1 =	A.StreamofWater
hasSemanticAssociation =	isEquivalentTo
element2 =	B.River
AssocID =	04

08	
element1 =	A.StreamofWater.Name
hasSemanticAssociation =	isEquivalentTo
element2 =	B.River.Name
AssocID =	08

09	
element1 =	A.Town.Geometry
hasSemanticAssociation =	isSimilarTo
element2 =	B.City.Shape
AssocID =	09

Some A-B Mappings

Figure 12. Some Mappings between Schemas A and B



STEP 4: Query Reformulation

Q2REFA	
isExecutedIn =	Peer_A
asksForCondition =	Cross(SW.Geometry,T.Geometry)=1
hasModel =	Object-Relational
asksForAttribute =	A.StreamofWater.Name
	A.Town.Name
hasEntity =	A.StreamofWater
	A.Town
isReformulationOf =	Q2
usesOperator =	Cross
hasRestriction =	Geographical Result
hasFinality =	GetRiversCrossCities
hasDescription =	For all the rivers, find the cities thro...
name =	Q2REFA

Q2REFA is a reformulation of Q

Figure 13. CEs at Query Reformulation Time

It is important to note that, in this example, query **Q** will be executed both in Peer B (submission peer) and in Peer A (through a reformulation). In Step 5, when the executed queries results are assembled to produce the final answer, the system analyzes other CEs such as multi-representation and scales difference. Considering that the formulating scale is about 1:100'000, this means that the user is working with a more detailed view of the themes. Thus the graphical result will be taken from Peer B whose scale of origin is closer and whose *City*'s geometric representation (polygon) is more adequate to that level of detail. Therefore, since the user interface is able to present geographical results, the final result (step 6) will be depicted to the user both graphically and textually (e.g. in the map and in a table format). Sometimes, the final result may be produced from the answers obtained in several peers if they return complementary information, for example, when some attributes are present in one peer but are absent in another.

Representing context information using an ontology brings various benefits. It provides concept subsumption, concept consistency and instance checking (including object properties checking). Efficient implementation of these operations allows a PDMS to organize knowledge, maintain its consistency, answer semantic queries and recognize conditions that trigger rule firings. The goal of a "semantic query" is to provide answers to queries in face of incomplete information, usually stored in heterogeneous data sources that are part of a dynamic environment. To this end, the system should take advantage of the available semantic information (in our work, through contextual elements) in order to provide an enriched query execution process. Semantic queries may produce different results to different users, depending on the contextual elements that are acquired at its submission time. For example, consider the situation in which an element (e.g. a schema entity) of a given rule is not available (e.g., when a peer goes out and comes back and its mappings have not yet been updated). In this case, it is possible to exploit generalization relationships (through the ontology) between concepts to find out a concept that can be used in the rule. As a result, the rule is fired anyway, despite the lack of precise information, returning a more general, but yet meaningful result. In this sense, in a dynamic environment such as a PDMS, the query results may be considered complete according to what is available in that given period of time and taking into account the user specific needs as well.

A context ontology also allows defining constraints and reasoning rules that may be used to derive other implicit context information. In our work, for instance, we consider inter-schema mappings with their types (e.g. subsumption) as contextual information. Based on the mapping types, rules can be applied to derive new useful mappings among the peers. Thus we are currently specifying some rules to infer other kinds of mappings from the existing ones in query execution time in order to provide other possible semantic query paths. Thus, considering C_A as a concept from Peer A, C_B a concept from Peer B and C_C a concept from Peer C, and subsumption and equivalence mappings, we present an example of this kind of rule in Table 1.

Table 1. A Rule Example

Rule	Instantiation
If $C_A \sqsubseteq C_B$ and $C_B \equiv C_C$ then $C_A \sqsubseteq C_C$	If A.VisitingTeacher $\sqsubseteq$ B.Teacher and B.Teacher $\equiv$ C.Professor Then A.VisitingTeacher $\sqsubseteq$ C.Professor

In addition, in Table 2, we work with some properties that may be used to infer spatial relationships. Thus, knowing that *Brazil* is part of *South America*, we can provide users with the extra information that *Brazil* is also part of *America*. Also, if a user poses a query that needs the operation "INSIDE" but there is no available data source which realizes it, the system can search one that executes "CONTAINS", since from one we can derive the other and vice-versa.

Table 2. Some Spatial Property Rule Examples

Property	Rule	Instantiation
Part-of	If A isPartOf B and B isPartof C Then A isPartOf C;	If "Brazil" isPartOf "SouthAmerica" and "SouthAmerica" isPartOf "America" Then "Brazil" isPartOf "America";
Contains -Inside	If A contains B Then B isInside A;	If "Brazil" contains "São Paulo" Then "São Paulo" isInside "Brazil";

This is a brief description of how the use of CODI can help to enhance data integration, and, more specifically, query processing. In fact, all information from the geospatial integration world that is to be reasoned over may be dealt with as contextual information. Consequently, from explicit contextual elements, gathered from the peers, from the mappings and from the query formulation, the system can infer and derive other implicit contextual elements. Moreover, since the environment (PDMS) is highly dynamic and, for each submitted query, the whole query execution process instantiation changes completely, the context around the query (its semantics), the peers (availability), mappings (may be of different types) and the user (preferences, interface) are essential information that have to be dealt with. In this work, such information is treated as context.

By using context, the system is able to adapt and react to different users' queries and needs. Without context, query processing would be limited by not dealing with some information that can just be acquired on the fly. As an illustration, in our example, the kind of the interface where the query has been submitted and the working scale can only be acquired in such given time. Another example concerns the user preferences: not all user preferences are relevant all the time, and only those that are semantically close to the current query should be used, disregarding those ones that are out of context. We can think in the same way for the other domain entities: environment, application, data, procedure and associations. As a result, dealing with contextual information can increase the quality of query results and provide users with more complete answers.

6. RELATED WORK

In data integration systems, contextual information has been used in several ways to capture the relevant semantics related to an object, its relationships and the surrounding issues that may influence its usage. In this sense, it has been used in processes addressing data- and schema-level conflicts resolution as well as query answering, mostly taking into account the user context.

In terms of *schema reconciling*, our work is quite similar to the work of Turley *et al.* and Ram *et al.* [17, 12]. Turley *et al*. relate the contextual information necessary to improve data integration in healthcare applications defining five main types of context while Ram *et al.* provide constructs and definitions to represent data- and schema-level conflicts between original and target contexts. Our work, differently, deals with a broader range of DI entities and related CEs, and also allows to infer existing semantic relations between schema element names taking into consideration context-bound interpretation.

In *query answering*, the effective use of multiple data sources requires context and user-specific reconciliation of differences in the data semantics among them. A partial investigation of the query translation from a context and user's point of view is given in the work of Bao *et al.* [1]. Stefanidis *et al.* [14] provide a logical model for the representation of user preferences and context-related information and demonstrate how their model can be integrated in a relational DBMS using data cubes for storing context dependent preferences. Both works are really focused in the user-specific context. Differently, our work is concerned not only with the user context, but also with other existing ones (e.g. mappings, data sources, queries) that a DI or query processing scenario requires. Furthermore, Souza *et al.* [13] have proposed an ontology to represent contextual information in geospatial data integration. Such ontology intended to define meta-concepts to be used in a broad range of areas, related to DI and to the geospatial realm. CODI is an extension of Souza *et al*'s work, providing environment and application contexts and other additional related elements as well.

In summary, our proposed approach differs from the ones mentioned above in the following aspects. Firstly, we define the CEs according to domain entities that have been identified as relevant in DI (data, environment, procedure, association, application and user). Secondly, using such domain entities we are able to provide a broader range of concepts and CEs which are to be used in DI processes, making CODI a more complete context ontology. Thirdly, CODI presents two different entities – procedure and environment (which allow adapting activities on the fly) that have not been employed in DI context-aware solutions yet. Fourthly, we use an ontology as a context representation model which is, in fact, a formal framework since its underlying logical formalism is Description Logics. In this sense, we are able to clearly define reasoning services over its constructs, and provide information reuse and sharing as well.

7. CONCLUSIONS AND FURTHER WORK

Due to the ever increasing complexity of data integration environments, the concept of context is becoming more and more a necessity, instead of an optional functionality. These environments are highly dynamic and the semantics and control information around their processes (e.g. queries) is rather relevant to produce results with quality according to users' needs and environment's capabilities. In this sense, this work presented CODI – a **C**ontextual **O**ntology for **D**ata **I**ntegration which aims to assist the common tasks of a generic data integration process. This means that CODI represents CEs related to the entities involved within a DI scenario from any knowledge domain. What differentiates CODI from other approaches is that the other ones lack important aspects that should be considered in DI (e.g. procedure, environment and association) since they are usually restricted to specific integration processes and/or knowledge domains. CODI aims to structure entities and their CEs in such a way that they may be used for diverse DI processes, including schema reconciliation and query processing. In fact, CODI may be used by developers of DI solutions to identify, model and represent contextual information in their applications.

CODI was encoded in OWL DL (Web Ontology Language) using Protégé 3.3.1[2]. It was initially used by a schema integration process that merges schemas of healthcare data sources from institutions of the Brazilian public health system [2]. In this case, some preliminary tests have already been done and the initial results have shown improvements when applying context. An experiment carried out has shown that without context, terminological semantic associations that were not settled in the domain ontology were not taken into account by the schema integration process. Using context, these relationships were considered. Currently, we are working on the implementation of the example described in this paper in a PDMS environment which was already developed in Java/RMI (used for peer

[2] Protégé 3.3.1 version, protege.stanford.edu/

communication). We are using Jena (jena.sourceforge.net/) for reasoning.

Finally, the preliminary results provide evidence that the application of contextual elements and the reasoning over them has the potential to yield considerable benefits to DI processes. As further work, we will develop additional scenarios which may allow us to work with other instances, constraints, queries and rules as well as with larger datasets. We are also integrating this work with a context manager which is being developed within our research group [18].

8. REFERENCES

[1] Bao, J., Caragea, D. and Honavar, V.: Query Translation for Ontology-Extended Data Sources. In: Proceedings of the Workshop on Semantic e-Science,Vancouver,Canada (2007)

[2] Belian, R. B. : A Context-based Name Resolution Approach for Semantic Schema Integration, PhD thesis, Center for Informatics, UFPE (2008)

[3] Brézillon, P.: Representation of Procedures and Practices in Contextual Graphs. The Knowledge Engineering Review, v. 18, n. 2, pp. 147-174 (2003)

[4] Dey A.: Understanding and Using Context. Personal and Ubiquitous Computing Journal, Volume 5 (1), pp. 4-7 (2001)

[5] Egenhofer M.: Reasoning about Binary Topological Relations. In Oliver Günther, Hans-Jörg Schek (Eds.): Advances in Spatial Databases, Second International Symposium, SSD'91, Zürich, Switzerland Proceedings. Lecture Notes in Computer Science 525 Springer, ISBN 3-540-54414-3 (1991)

[6] Goh, C. H., Bressan, S., Madnick, S., and Siegel M.: Context interchange: New features and formalisms for the intelligent integration of information. ACM TIS, 17(3) (1999)

[7] Halevy A., Rajaraman A. and Ordille J. : Data integration: the teenage years. In: Proceedings of the 32nd international conference on Very large data bases - Volume 32, pp. 9 – 16 (2006)

[8] Herschel, S. and Heese, R.: Humboldt Discoverer: A Semantic P2P index for PDMS. In: Proceedings of the International Workshop Data Integration and the Semantic Web, Porto, Portugal (2005)

[9] Kashyap, V. and Sheth, A.: Semantic and schematic similarities between database objects: a context-based approach. The VLDB Journal, v. 5. Springer-Verlag, pp. 276-304 (1996)

[10] Mills J., Goossenaerts J.B.M.: Using contexts in managing product knowledge. In: E. Arai, J. Goossenaerts, F. Kimura, K. Shirase (eds) Knowledge and Skill Chains in Engineering and Manufacturing: Information Infrastructure in the Era of Global Communications, Springer, pp. 57-65 (2005)

[11] Rahm, E. and Bernstein, P.: A survey of approaches to automatic schema matching. The VLDB Journal, vol. 10, pp. 334-350 (2001)

[12] Ram, S. and Park, J.: Semantic Conflict Resolution Ontology (SCROL): An ontology for detecting and resolving Data- and Schema-Level Semantic Conflicts. Knowledge and Data Engineering, IEEE Transactions on Communications (2004)

[13] Souza, D., Salgado, A.C. and Tedesco, P.: Towards a Context Ontology for Geospatial Data Integration. Second International Workshop on Semantic-based Geographical Information Systems (SeBGIS'06), Montpellier, France (2006)

[14] Stefanidis K., Pitoura E., Vassiliadis P.: On Supporting Context-Aware Preferences in Relational Database Systems. In: Proc. of the first International Workshop on Managing Context Information in Mobile and Pervasive Environments, in conjunction with MDM 2005, Cyprus (2005)

[15] Stuckenschmidt, H., Wache, H.: Context Modelling and Transformation for Semantic Interoperability, In: M. Bouzeghoub & M. Klusch & W. Nutt & U. Sattler (Eds.), Knowledge Representation Meets Databases (KRDB 2000) (Vol. 29): CEUR Workshop Proceedings (2000)

[16] Sung, L. G. A., Ahmed, N., Blanco, R., Li, H, Soliman, M. A., and Hadaller, D.: A Survey of Data Management in Peer-to-Peer Systems, Web Data Management, Winter 2005, pp.1–50 (2005)

[17] Turley, J. and Johnson-Throop, K.: The role of context in the integration of heterogeneous healthcare databases. In: Proceedings of 6th International workshop on enterprise networking and computing in healthcare industry (2004)

[18] Vieira V., Tedesco P., Salgado A.C. and Brézillon P.: Investigating the Specifics of Contextual Elements Management: The CEManTIKA Approach. The Sixth International and Interdisciplinary Conference on Modeling and Using Context. B. Kokinov et al. (Eds.): LNAI 4635, Springer-Verlag, pp. 493–506 (2007)

[19] Wang X., Zhang D., Gu T., Pung H.: Ontology Based Context Modeling and Reasoning using OWL. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, p.18 (2004)