PAIRSE: A Privacy-Preserving Service-Oriented Data Integration System

Djamal Benslimane
LIRIS, Lyon 1 University
69622 Villeurbanne, France
djamal.benslimane@univ-lyon1.fr

Mahmoud Barhamgi
LIRIS, Lyon 1 University
69622 Villeurbanne, France
mahmoud.barhamgi@univ-lyon1.fr

Frederic Cuppens
TELECOM Bretagne
35576 Rennes, France
frederic.cuppens@telecom-bretagne.eu

Franck Morvan
IRIT, Paul Sabatier University
31062 Toulouse, France
franck.morvan@irit.fr

Bruno Defude
TELECOM SudParis
91011 Evry, France
bruno.defude@it-sudparis.eu

Ebrahim Nageba
Claude Bernard University
69622 Villeurbanne, France
ebrahim.nageba@univ-lyon1.fr

ABSTRACT

Privacy is among the key challenges to data integration in many sectors, including healthcare, e-government, etc. The PAIRSE project aims at providing a flexible, loosely-coupled and privacy-preserving data integration system in P2P environments. The project exploits recent Web standards and technologies such as Web services and ontologies to export data from autonomous data providers as reusable services, and proposes the use of service composition as a viable solution to answer data integration needs on the fly. The project proposed new composition algorithms and service/composition execution models that preserve privacy of data manipulated by services and compositions. The proposed integration system was demonstrated at EDBT 2013 and VLDB 2011.

1. INTRODUCTION

Data integration has been a long-standing challenge for the database community. This is motivated by the number of contexts in which the need for a flexible data integration mechanism has become critical, including Web and enterprise data integration, scientific data exploration, data exchange in government agencies, etc.

Much of the literature on data integration across autonomous data sources has tacitly assumed that data on the side of each data source can be revealed and shared with other sources. In practice, however, data integration scenarios are often hampered by legitimate and widespread data privacy concerns. In the healthcare application domain for example, medical data are subject to many legislations (e.g., [2, 1]) around the world that restrict collection, processing, and disclosure of personal data, and hold data holders accountable for any unintended data disclosure or misuse.

The PAIRSE project addresses the challenge of flexible and privacy-preserving data integration in peer-to-peer environments. Driven by the recent trends of using SOA-oriented architectures for data integration in modern enterprises, PAIRSE assumes that data sources are exposed to the data sharing environment as Web services. This type of services is commonly known as data services [11], where data services provide a well documented, platform (and source) independent, interoperable method of interacting with data. PAIRSE proposes a service composition-based approach for on-demand data integration; i.e., heterogeneous data services from autonomous service providers are selected and composed on the fly to answer users' queries. Data privacy preservation is a key objective of PAIRSE. Users in PAIRSE are allowed only to access the information they are entitled to for a given purpose.

PAIRSE focuses on modeling, discovering, selecting and composing data services to efficiently answer users' queries. The contributions of PAIRSE, which was demonstrated at EDBT 2013 [5] and VLDB 2011 [9], are summarized as follows:

- *Semantic description model for data services*: The semantics of data services should be explicitly represented to automate their discovery, selection and composition. We modeled data services as "*RDF Views*" over domain ontologies to formally define their semantics [7]. The service description files (e.g., WSDLs) are annotated with these RDF views.
- *Query resolution by automatic service composition*: Queries in PAIRSE are resolved by *automatically* selecting and composing data services. We exploited mature query rewriting techniques to devise a novel service composition algorithm [7, 9]. The algorithm relieves users from having to manually select and com-

pose services, tasks that would generally require important programming skills. We proposed also an efficient algorithm to locate relevant services in a P2P environment [14].

- *Privacy preservation*: We proposed a privacy preserving composition model [5, 8, 16]. Our model allows services providers to locally enforce their privacy and security policies when their services are invoked. In addition, it prevents services in a composition from learning any information about the data that each other holds, beyond what is permitted.

The rest of the paper is organized as follows. Section 2 gives an overview of our integration system. Section 3 describes our semantic modeling of data services. Section 4 presents our composition approach. Section 5 presents our techniques to privacy preservation. Section 6 applies our work in two application domains, and summarizes obtained results. Section 7 concludes the paper.

2. PAIRSE'S ARCHITECTURE

The PAIRSE data integration system has a hybrid peer-to-peer infrastructure [14], where peers form communities of interest, called *Virtual Organizations VOs*. Each VO has a common domain ontology modeling its expertise, and peer members that may have relations with members from other VOs. Relations between peers exist only if there is a mapping between the ontologies of their respective VOs. PAIRSE does not impose any constraint on the topology graph formed by the ontologies and the different mappings. Peers export their (sharable) data sources as data services.

PAIRSE follows a *declarative* approach to compose data services (Figure 1). Data services in each peer are modeled as *RDF Views* over domain ontologies to explicitly define their semantics. Users formulate their queries on domain ontologies using SPARQL query language. Then, our system exploits the defined RDF views (added as annotations to service description files) to select and compose the relevant services using an RDF query rewriting algorithm that we have devised for that purpose. Queries may necessitate the use of remote data services, in which case an efficient P2P service discovery algorithm [14] is used to locate and retrieve the descriptions of relevant services from remote peers. The system generates then an execution plan for the composition and executes it to provide the user with the requested data. As data services may manipulate privacy-sensitive information, PAIRSE proposed new service and composition execution models to preserve privacy.

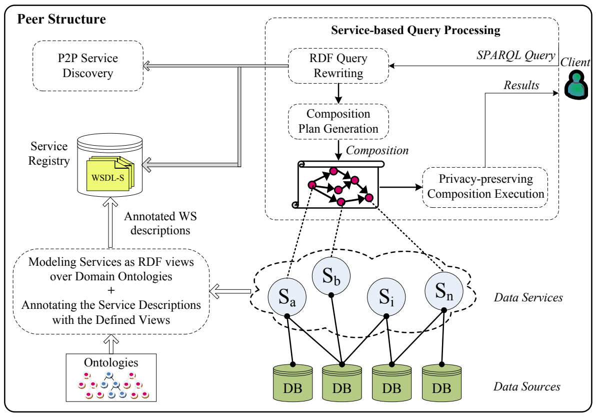

Peer Structure
Service-based Query Processing
P2P Service Discovery
RDF Query Rewriting
SPARQL Query
Client
Results
Composition Plan Generation
Composition
Service Registry
WSDL-S
Privacy-preserving Composition Execution
Annotated WS descriptions
Modeling Services as RDF views over Domain Ontologies + Annotating the Service Descriptions with the Defined Views
S_a
S_b
S_i
S_n
Data Services
Ontologies
DB
DB
DB
DB
Data Sources

Figure 1: Peer Structure

3. SEMANTIC MODELING AND SERVICE QUERYING

In this section, we explain our service composition based approach to query resolution.

3.1 Semantic Modeling of Data Services

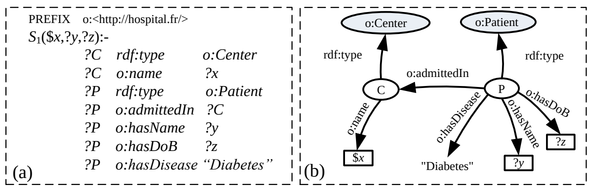
Modeling and explicitly specifying the semantics of data services are the first step towards the automation of service selection and composition.

In PAIRSE, we proposed in [7] to model data services as *RDF Parameterized Views* (RPVs) over domain ontologies. A parameterized RDF view uses *concepts* and *relations* whose meanings are formally defined in domain ontologies to define the semantic relationships between input and output parameters sets of a data service. A parameterized view is a technique that has been used to describe content and access methods in Global-as-View (GaV) integration architectures [13]. Figure 2 shows an RPV of a service returning the personal information (i.e., name and dates of birth) of patients admitted in a given medical center. Note that input parameters are prefixed with the symbol "\$" and output parameters are prefixed with the symbol "?".

RDF views may also specify constraints to characterize the data manipulated by their corresponding services. These constraints may have different forms, including *simple interval constraints* (e.g., $X \in [a, b]$, where X is a variable used in an RDF view), and *fuzzy constraints* interpreted according to a fuzzy membership function (e.g., the medications returned by a service have "*High*" concentration of hydroxypropy1-β-cyclodextrin; i.e., X *is High*, where the fuzzy term "*High*" is interpreted by a membership function specifying for each value of X the degree to which it is high).

We adopted an approach similar to SAWSDL[1] to

[1]http://www.w3.org/2002/ws/sawsdl/



```
PREFIX   o:<http://hospital.fr/>
S1($x,?y,?z):-
          ?C   rdf:type        o:Center
          ?C   o:name          ?x
          ?P   rdf:type        o:Patient
          ?P   o:admittedIn    ?C
          ?P   o:hasName       ?y
          ?P   o:hasDoB        ?z
          ?P   o:hasDisease    "Diabetes"
```

Figure 2: (a) a parameterized RDF view; (b) its graphical representation

associate data services with their RPVs. We exploited the extensibility feature of the WSDL standard to annotate the WSDL files with RPVs.

3.2 Service-based Query Resolution

In PAIRSE, users' queries are resolved by composing relevant data services on the fly. Each virtual organization in PAIRSE's hybrid P2P architecture has a DHT (Distributed Hash Table) to index its published services [14]. Services are indexed according to the ontological concepts used in their RPVs.

When a query is issued at a given peer, relevant services are first sought in the same VO where the query is posed, then the service discovery request is propagated to connected VOs. The descriptions of discovered services are then sent back to the initial peer, where the relevant services will be selected and composed. Furthermore, for each discovered service we return the mapping path between the ontologies associated with the expertise domains (i.e., VOs) of the discovered service and the initial peer. This mapping path allows the translation of RPV views.

We proposed a *query rewriting* based service composition algorithm to select and compose data services on the fly [7, 9]. The algorithm, given a SPARQL query, and a set of data services represented by their RPVs, rewrites the query in terms of calls to relevant services. Our algorithm extends earlier works on query rewriting and data integration [13] in the following aspects:

Compliance with the RDF/S data models: while most of previous work has focused on relational and XML data integration [13, 17], we considered the case of RDF/RDFS data integration. Specifically, our query rewriting algorithm takes into account RDF schema constraints such as *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain*, and *rdfs:range* when comparing RPVs to queries. The consideration of RDFS constraints is important as allows our system to infer more results than the previous rewriting techniques. For example, suppose there is a statement in an RDFS ontology specifying that *:Medication rdfs:subClassOf :Drug*. Given a data service S returning the medications administered to a given patient, and a query Q for the drugs administered to a given patient, our algorithm automatically infers that S can be used to generate rewritings for Q.

Answering parameterized queries: while previous data integration systems have focused on answering specific queries, PAIRSE has focused on answering *parameterized queries*. The key focus was on constructing compositions of services (i.e., parameterized integration plans) that are independent of a particular input value. For example, assume a parameterized query $Q(\$x, ?y)$ for the medications y that may interact with a given medication x. Assume also two data services:

S_1(\$x,?y), where $x \in$ [1, 5] and $y \in$ [100,150],
S_2(\$x,?y), where $x \in$ [6,10] and $y \in$ [150,200]

If Q was a specific query ($Q_{x=2}$), then S_2 would not be considered in the rewriting (i.e., composition) as x=2 is not covered by S_2. In contrast, both of S_1 and S_2 are usable for Q, to cover as much as possible of the potential values of x. Our composition algorithm extends the previous ones with: (i) a probabilistic subsumption test to determine in a polynomial time the minimum number of services required to satisfy the value constraints that may be specified on query's parameters [6], and (ii) a mechanism to optimize the generated composition plans based on value constraints specified in service descriptions [7].

Inclusion of user's preferences: often the number of candidate compositions that may be used to answer the same query is very large. We proposed an approach [9] to compute the top-k compositions based on user preferences. In our approach, we modeled user's preferences using fuzzy sets. We match the (fuzzy) constraints of the relevant services to those of the query and determine their matching degrees using a set of matching methods from the fuzzy set theory. We then rank-order candidate services based on a fuzzification of *Pareto dominance* and compute the top-k compositions.

4. PRIVACY PRESERVATION IN PAIRSE

In this section, we briefly present our models to preserve the privacy of manipulated data at the service and the composition levels.

4.1 Privacy-preserving Service Execution Model

Data returned by a data service may be subject to different security and privacy concerns. For example, different people may have different access rights over the same data item; data subjects[2] may

[2]We use the term *data subject* to mean the individual whose information is manipulated by data services.

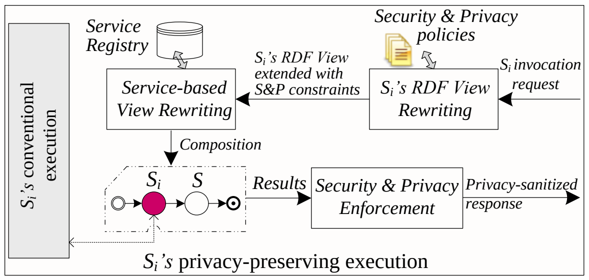

Service Registry
Security & Privacy policies
S_i's RDF View extended with S&P constraints
S_i invocation request
Service-based View Rewriting
S_i's RDF View Rewriting
Composition
S_i
S
Results
Security & Privacy Enforcement
Privacy-sanitized response
S_i's conventional execution
S_i's privacy-preserving execution

Figure 3: A Privacy-preserving Service Execution Process

have different preferences about the disclosure of their data, etc. A common approach in the database field to handle such concerns is to push them to the underlying DBMS by rewriting the query to include these constraints [15]. However, this may not be applicable to data services as the same service may access a multitude of heterogeneous data sources that may not be necessarily managed by a DBMS. An alternative approach is to enforce privacy and security policies at the application level [4], by modifying, in our case, the source code of data services. However, this may not always be applicable, as most of current data service creation platforms (e.g., *AquaLogic* [11]) provide data services as *black boxes* that cannot be modified. Even if the code was modifiable, this approach often leads to privacy leaks [15].

We proposed a secure, privacy-preserving execution model for data services allowing service providers to enforce their privacy and security policies without changing the implementation of their data services (i.e., services are seen as black boxes). Our model is inspired by the database approach to "*declaratively*" handle the security and privacy concerns. It involves the following steps (refer to Figure 3):
Step 1: *View rewriting to integrate security and privacy constraints.* When a data service is invoked, our model rewrites its corresponding RDF view to take into account applicable security and privacy rules from the service's associated policies, which are expressed using the OrBAC and PrivOrBAC models over domain ontologies and take into account the data *recipient* (i.e., service consumer), his *purpose* for requesting the data, and the *consents* of data subjects [16]. The soundness and correctness of our algorithm are demonstrated in [16, 8].
Step 2: *Rewriting the extended view in terms of data services.* The extended RDF view $v_{extended}$ may include additional data items (denoted by $\Delta v = v_{extended} - v_{original}$) required to enforce security and privacy constraints. In this step, we find the data services covering Δv, and rewrites $v_{extended}$ in terms of these services along with the initial service.
Step 3: *Enforcing security and privacy constraints.* Services selected in the previous step are composed and executed using the conventional service execution process. The composition returns (i) the data items returned by the invoked service along with (ii) the data items necessary to evaluate the security and privacy constraints. We defined a privacy filter that evaluates the privacy constraints of the different items that are subject to privacy constraints in the view. *Null* values will be returned for items whose privacy constraints evaluate to *False*.

We demonstrated the validity of our model by extending the architecture of the famous service container AXIS[3] 2.0 with a new module implementing our privacy-preserving service execution model.

4.2 Privacy-preserving Composition Execution Model

Executing compositions may disclose confidential information to component services. Assume, for example, a composition of two services: S_1 returns HIV patients in a given city, and S_2 checks whether a given patient has been treated for psychiatric disorders. Such composition could be needed (by a pharmaceutical researcher) to investigate the connection between a chemical component present in HIV medicines and the development of severe psychiatric disorders. Assume also *Bob* is a common patient to S_1 and S_2. If S_2 is invoked with Bob's identifier, and the provider of S_2 has an access to the composition plan (i.e., he knows that Bob was outputted by S_1), then he will infer that Bob is an HIV patient. On the other hand, if the data returned by S_1 were completely privacy-sanitized (e.g., by removing identifiers and sensitive information), then the composition could not be executed.

We proposed a privacy-preserving composition execution model in [5] that limits the information disclosed to services in a composition about the data that each other holds. Our model distinguishes between the following entities: (i) the services in the composition, (ii) the execution engine, and (iii) the recipient of final results. It relies on two key ideas: First, data services use the same order-preserving encryption scheme OPES [3] to encrypt the identifier attributes that are needed to connect data subjects across the different services. They are still free to protect non-identifier attributes with their own techniques (e.g., anonymization, etc.). This way the execution engine has only access to protected data and can still link data subjects across services using the encrypted identifier attributes (note that OPES allows for applying equality queries on encrypted

[3]http://axis.apache.org/axis2/java/core/

data). By the end of the composition execution, it removes from the final results the encrypted identifier attributes before returning them to the recipient, who will thus get only privacy-sanitized data. Second, we proposed a algorithm to allow the execution engine to generalize the encrypted value v_e received from a service S_i before proceeding with the invocation of the subsequent service S_j in the composition, such that the generalized value $\mathrm{Gen}(v_e)$ corresponds to k input encrypted values for which S_j has outputs; e.g., the identifier of Bob is generalized to cover k-1 other patients for which S_2 has an output (i.e., S_2 will not be able to distinguish between Bob and k-1 other patients).

5. IMPLEMENTATION AND EVALUATION

We evaluated our different techniques and algorithms in the healthcare and bioinformatics application domains. These domains have widely embraced Web standards, such as XML and Web services [12, 10], and are characterized by the need for a flexible and privacy-preserving data integration approach.

The cardiology hospital of Lyon provided us with access to two medical databases. The identities of patients in these databases were changed. We also generated synthetic medical data about the same patients. We implemented about /400/ data Web services on top of our real and synthetic data. Services were all deployed on an extended version of AXIS 2.0 implementing our service execution model. We built a medical ontology based on the building blocks and the data types defined in the HL7 standard, and used it for the annotation of service description files. To evaluate our techniques in the bioinformatics domain, we used a set of /300/ services from the *BioCatalogue registry*[4].

Figure 4 (part *a*) shows the query interface to PAIRSE. Users are assisted in formulating their SPARQL queries over domain ontologies. The figure shows (in part *b*) also the composition plan of a selected composition, along with the privacy-sanitized results (part *c*).

We conducted exhaustive experiments to evaluate the performance of our integration system. We summarize below obtained results[5]:

Composition construction and execution: Our experiments in [7] showed that our composition algorithm can handle hundreds of data services in a reasonable time. For example, for chain queries [13] and RPVs with a length of 3 or 4 object properties the algorithm was able to handle up to 400 services in less than 4 seconds.In the context of parameterized queries, our experiments in [6] showed that our algorithm to find the minimum set of services introduced only a small cost at the composition construction time (i.e., in all experiments the algorithm required less than 10% of the time needed to rewrite the query), and improved substantially the composition execution time (i.e., in all experiments the composition execution time was reduced to less than 0.75% of the time needed without optimization), as it removes redundant services. In the context of preferences queries, our experiments in [9] considered that services can be grouped in classes. The experiments showed that the top-k compositions can be computed efficiently. For instance, for classes containing about 400 services, the top-k compositions are computed in less than 4 seconds.

Security and privacy preservation: The conducted experiments in [8] showed that our secure and privacy preserving service execution model added only a small increase to the service execution time. In all experiments, the cost incurred in the enforcement of security and privacy constraints did not exceed 10% of the time required to execute the service with ignoring these security and privacy constraints altogether. The conducted experiments for the evaluation of our composition execution model [5] showed that the time required to execute the composition with privacy preservation is at most three orders of magnitude of the time required without privacy preservation (K_i was set to 4 in all tests). We were able to cut down that cost to two orders of magnitude by *reusing* the values of the protocol parameters that were computed in past invocations of the same services (and during the same composition execution).

6. CONCLUSION

The goal of the PAIRSE project was to develop new methods and techniques for flexible and privacy-preserving data integration. We have evaluated our composition-based approach in the healthcare and the bioinformatics domains. The obtained results [5, 9, 7, 14, 8, 16, 6] are promising.

7. ACKNOWLEDGMENTS

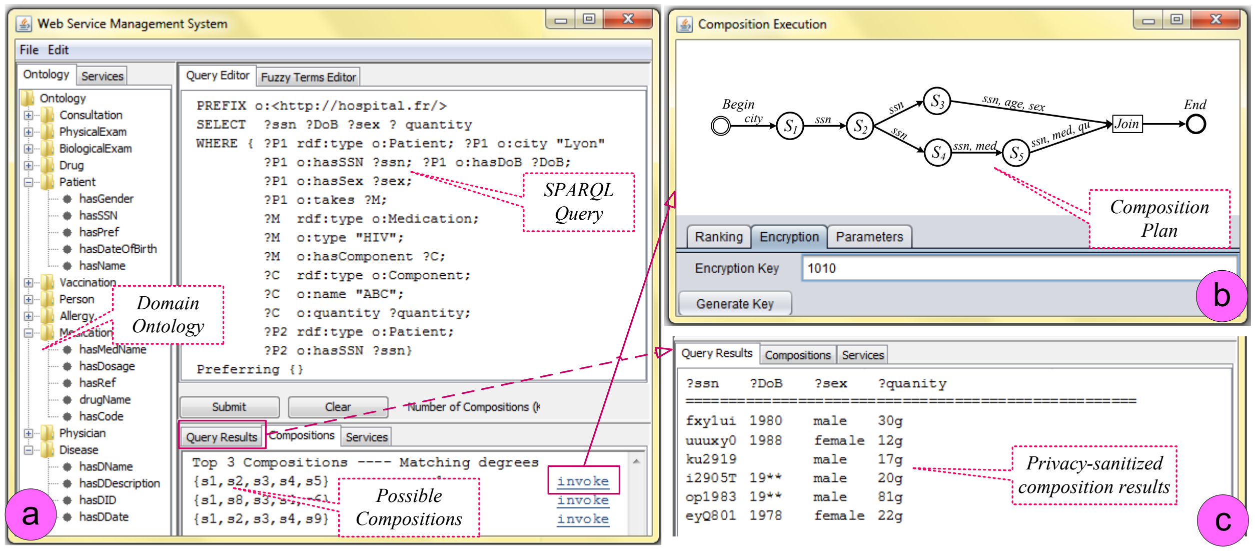
PAIRSE is funded by the French National Research Agency under the grant number ANR-09-SEGI-008

8. ADDITIONAL AUTHORS

Michael Mrissa (Lyon 1 University, `michael.mrissa@univ-lyon1.fr`), Francois Paulus (Semsoft Company, `francois.paulus@semsoft-corp.com`), Stephane Morucci (Swid Company, `stephane.morucci@swid.fr`), Nora Cuppens (Telecom-Bretagne, `nora.cuppens@telecombretagne`

[4]http://www.biocatalogue.org/

[5]For detailed information about the considered settings in each cited experiment, please refer to the corresponding paper.



```
PREFIX o:<http://hospital.fr/>
SELECT  ?ssn ?DoB ?sex ? quantity
WHERE { ?P1 rdf:type o:Patient; ?P1 o:city "Lyon"
        ?P1 o:hasSSN ?ssn; ?P1 o:hasDoB ?DoB;
        ?P1 o:hasSex ?sex;
        ?P1 o:takes ?M;
        ?M  rdf:type o:Medication;
        ?M  o:type "HIV";
        ?M  o:hasComponent ?C;
        ?C  rdf:type o:Component;
        ?C  o:name "ABC";
        ?C  o:quantity ?quantity;
        ?P2 rdf:type o:Patient;
        ?P2 o:hasSSN ?ssn}
Preferring {}
```

```
Top 3 Compositions ---- Matching degrees
{s1,s2,s3,s4,s5}
{s1,s8,s3,s4,s6}
{s1,s2,s3,s4,s9}
```

```
?ssn    ?DoB    ?sex    ?quanity
==================================================
fxylui  1980    male    30g
uuuxy0  1988    female  12g
ku2919          male    17g
i2905T  19**    male    20g
op1983  19**    male    81g
eyQ801  1978    female  22g
```

Figure 4: (a) Query interface; (b) The selected composition; (c) Obtained results

.eu), Chirine Ghedira (Lyon 3 University, chirine.ghedira-guegan@univ-lyon3.fr), Riad Mokadem (Paul Sabatier University, Riad.Mokadem@irit.fr), Said Oulmakhzoune (Telecom-Bretagne, said.oulma khzoune@swid.fr) and Jocelyne FAYN (INSA-Lyon, Jocelyne.Fayn@insa-lyon.fr).

9. REFERENCES

[1] E. u. directive on data protection, official journal of the european communities, 23 november 1995.

[2] Health insurance portability and accountability act of 1996, united states public law 104-191.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD Conference*, pages 563–574, 2004.

[4] P. Ashley and D. Moore. Enforcing privacy within an enterprise using ibm tivoli privacy manager for e-business. In *VLDB*, pages 108–119, 2003.

[5] M. Barhamgi, D. Benslimane, Y. Amghar, N. Cuppens-Boulahia, and F. Cuppens. Privcomp: a privacy-aware data service composition system. In *EDBT*, pages 757–760, 2013.

[6] M. Barhamgi, D. Benslimane, C. Ghedira, S.-E. Tbahriti, and M. Mrissa. Optimizing daas web service based data mashups. In *IEEE SCC*, pages 464–471, 2011.

[7] M. Barhamgi, D. Benslimane, and B. Medjahed. A query rewriting approach for web service composition. *IEEE Transactions on Services Computing*, 3(3):206–222, 2010.

[8] M. Barhamgi, D. Benslimane, S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, M. Mrissa, and H. Taktak. Secure and privacy-preserving execution model for data services. In *CAiSE*, pages 35–50, 2013.

[9] K. Benouaret, D. Benslimane, A. HadjAli, and M. Barhamgi. Fudocs: A web service composition system based on fuzzy dominance for preference query answering. *PVLDB*, 4(12):1430–1433, 2011.

[10] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, and M. Roos. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research*, 38(5):689–694, 2010.

[11] M. J. Carey, N. Onose, and M. Petropoulos. Data services. *Commun. ACM*, 1(1):2–9, 2012.

[12] A. Dogac. Interoperability in ehealth systems. *PVLDB*, 5(12):2026–2027, 2012.

[13] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[14] I. Ketata, R. Mokadem, and F. Morvan. Resource discovery considering semantic properties in data grid environments. In *Globe*, pages 61–72, 2011.

[15] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, and Y. Xu. Limiting disclosure in hippocratic databases. In *VLDB*, pages 08–19, 2004.

[16] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, and S. Morucci. fquery: Sparql query rewriting to enforce data confidentiality. In *DBSec*, pages 146–161, 2010.

[17] C. Yu and L. Popa. Constraint-based xml query rewriting for data integration. In *SIGMOD Conference*, pages 371–382, 2004.