UNIVERSITY OF LYON

DOCTORAL SCHOOL OF COMPUTER SCIENCE AND MATHEMATICS

Thesis Advancement Report

Trusted SLA Guided Data Integration on Multi-Cloud Environments

PhD Student: Daniel Aguiar da Silva Carvalho - Centre de Recherche Magellan Jean Moulin Lyon3 University

Advisors:
Prof. Chirine Ghedira-Guegan - LIRIS, Magellan Research Center Jean Moulin Lyon3 University
Prof. Genoveva Vargas-Solar - CNRS-LIG, Grenoble
Prof. Nadia Bennani - LIRIS, INSA-Lyon

March 2017

Contents

Chapter 1

Identification

Contents

1.1 PhD student identification

Name:	Daniel Aguiar da Silva Carvalho
Date of birth:	16/09/1988
Nationality:	Brazilian
Professional mail:	daniel.carvalho@univ-lyon3.fr
Personal mail:	danielboni@gmail.com

1.2 Laboratory and Supervision

Current I am working in the *Magellan Research Center* (Lyon3) and attached to the *InfoMaths* doctoral school (Lyon1).

The thesis project is being developed under the supervision of Prof. Chirine GHEDIRA-GUEGAN, Prof. Genoveva VARGAS-SOLAR and Prof. Nadia BENNANI.

1.3 Subject and Funding

The thesis is entitled «*Trusted SLA-Guided Data Integration on Multi-Cloud Environments*». It is funded by the project ARC 6 2014.

Chapter 2

Professional and Scientific Aspects

Contents

2.1 Courses

The list of courses followed during the project are presented below:

- *Français pour non francophones niveau 1 beginner (2014-2015).* Code E. Transversal.
- *Français pour non francophones niveau 1 beginner/débutant, élémentaire (2015-2016).* Code E. Transversal.
- *Français pour non francophones niveau 2 intermédiaire (2015-2016).* Code E. Transversal.
- *Writing a scientific paper step by step (2015-2016).* Code E. Transversal.
- *E1a Français langue étrangére-niveaux débutant à élémentaire - French as a Foreign language-beginner to elementary levels (2016-2017).* Code E. Transversal.

2.2 Scientific production

Daniel Carvalho, Nadia Bennani, Genoveva Vargas-Solar, Chirine Ghedira, Placido Souza Neto. **Can Data Integration Quality be Enhanced on Multi-cloud using SLA?**. DEXA 2015, Sep 2015, Valencia, Spain. Lecture Notes in Computer Science 9262, Springer 2015.

D. A. S. Carvalho, P. A. Souza Neto, G. Vargas-Solar, N. Bennani, C. Ghedira, **Rhone: a quality-based query rewriting algorithm for data integration**, Short paper, 20th East-European Conference on Advances in Databases and Information Systems, ADBIS 2016.

2.3 Research activities

Paper proposal presentation. **Trusted SLA-Guided Data Integration on Multi-cloud Environments**. Seminar presented to the Information System group from the Magellan's Research Center. In March 5th 2015.

Participating on to the **1st French Brazilian School on Smart cities and Big Data** at the University of Grenoble Alpes (http://fr-br-school.imag.fr). April 2015.

Poster presentation. **SLA-Guided Data Integration on Multi-cloud Environments**. Poster presented in the « Journée Scientifique de l'Arc 6 » held in 20th November 2015 at Grenoble, France.

PhD proposal and query rewriting algorithm presentation. **Trusted SLA-Guided Data Integration on Multi-cloud Environments**. Seminar presented to the Service-Oriented Computing team meeting from LIRIS, Lyon 1. In 3rd December 2015.

Query rewriting algorithm and ongoing works presentation. **A Service-based Query Rewriting Algorithm**. to the Information System group from the Magellan's Research Center. In 4th February 2016.

Paper presentation. **Rhone: a quality-based query rewriting algorithm for data integration** at the 20th East-European Conference on Advances in Databases and Information Systems, ADBIS 2016.

Poster presentation. **SLA-Guided Data Integration on Multi-cloud Environments**. Poster presented in the « Journée Scientifique de l'Arc 6 » held in 24th November 2016 at Lyon, France.

Chapter 3

Thesis Advancement

Contents

3.1 Context and problem statement

Data integration is a widely studied issue in the database domain. It consists in merging data from different databases and providing a unified view of this data to the user [?]. Commonly, data integration is referred in the literature as a problem of answering queries using views. Many authors have reported their algorithms for this purpose [?]. Levy *et al.* proposed the *bucket algorithm* [?]. Duschka and Genesereth introduced the *inverse-rules algorithm* [?]. Pottinger and Halevy presented the *MiniCon algorithm* in [?]. [?] extended some ideas of [?], and developed an implementation more efficient than the others. In general, algorithms in this domain share the same performance problem while combining views to produce a rewriting.

Current data integration implies consuming data from different data services and integrating the results while meeting users' quality requirements. Such requirements include the data that is retrieved and integrated, but also the properties of the data, its producers and the conditions in which such data is produced and processed. For example, whether the user accepts to pay for data, its provenance, veracity and freshness and how much is the user ready to pay for the resources necessary for integrating her expected result. Data services provide data according to specific APIs that specify method headers with input parameters describing the data to be retrieved and the type of results they can produce. Moreover data provision can be done by services according to different data quality measures. Such measures describe the conditions in which a service can provide or process data. These measures can be expressed in a service level agreement (SLA). An SLA states, what the user can expect from a service or system behavior. For example, whether it implements an authentication process, if it respects data consumer's privacy and the quality of the data the service can deliver, like freshness, veracity, reputation and other non-functional conditions like the business model that controls data delivery.

Data provision and data processing services may need a considerable amount of storage, memory and computing capacity that can be provided by cloud architectures. Furthermore, users could need to integrate the data provisioned by services in a homogeneous and general result. The integration process can also require important computing resources that can be obtained from the cloud too. Data provision and processing services can be deployed in the cloud. Their SLA includes the measures about the cloud services that they require to execute their requests. The cloud, itself exports a general SLA that specifies the conditions in which users can access the services (infrastructure, platform and software) deployed in it. A user willing to use the cloud services establishes a contract with the cloud provider guided by an economic model that defines the services she can access, the conditions in which they can be accessed (duplication, geographical location) and their associated cost. Different cloud providers have different possible contracts to establish with users (i.e., platinum, silver, gold, ivory users). Thus, for a given requirement, a user could decide which cloud services (from one or several cloud providers) to use for retrieving, processing and integrating data according to the type of contracts she can establish with them.

Data integration can be seen in the cloud computing as a service composition problem. In our previous work [**?**], we have identified that QoS aspects has started to considered while integrating data, and the cloud has become a popular environment to perform data integration. Moreover, data integration combined with SLA is an open issue in the cloud. Barhamgi *et al.* proposed a query rewriting approach which processes queries on data provider services [**?**]. Benouaret *et al.* introduced a service composition framework to answer preference queries [**?**]. In that approach, two algorithms based on [**?**] are presented to rank the best rewritings based on previously computed scores. Ba *et al.* presented an algorithm based on *MiniCon* that produces and order rewritings according to user preferences [**?**]. As in the database domain, these approaches requires an important amount of resources to process the rewriting and integration.

In consequence, data consumption is determined by quality constraints specified by the user (data consumer) and different contracts (i.e., SLA's) of the clouds providing the required services. User's constraints define the storage and computing capacity of the device that consumes the data, the data transmission bandwidth and cost, whether the data consumption is critical (time constraints) and energy consumption. The SLA of the cloud determines the type of quality a user can expect from its services, according to the contract (subscription) signed between her and the cloud provider. The user profile, her quality requirements and her execution context determine the conditions in which she is expecting to consume data by using such cloud services.

Thus, the **first challenge** is to compute what we call an integrated SLA that matches the user's integration preferences (including quality constraints and data requirements) with the SLA's provided by cloud services, given a specific user cloud subscription. The user may have general preferences depending on the context she wants to integrate her data such as economic cost, bandwidth limit, free services,

and storage and processing limits. The SLA's associated to the cloud services can be of different types: user - data service, data service - cloud provider, data provision service - data processing service, and cloud provider - cloud provider. In this context, matching the user integration preferences with the services that can contribute to produce a result can lead to search and identify in the chain of SLAs. Probably it is possible to find an incompatibility between the preferences and a SLA in the chain, in this case it is necessary to propose a strategy to solve the problem.

Furthermore, in order to fulfill requirements and satisfy user expectations, it is possible to have a collaboration between different clouds. This collaboration implies the agreement through SLAs between services deployed in different cloud providers. In consequence, matching user preferences with SLA's can lead to deal with heterogeneous SLA specifications (different schemata, different measures semantics and granularities). Computing an integrated SLA can imply dealing with heterogeneous SLA specifications and SLA-preferences incompatibilities.

The **second challenge** is to guide data integration taking into consideration the integrated SLA. Here, the data integration process includes (i) looking up services that can be used as data providers, and for services required to process retrieved data and build an integrated result; (ii) performing data retrieval, processing and integration and (iii) deliver results to the user considering her preferences (quality requirements, context and resources consumption). The integrated SLA can guide services filtering in the look up phase; it can help to control the amounts of data to retrieve and process according to consumption rights depending on the user subscription to the participating cloud providers and how to deliver data considering the user's context.

Several researches have reported their studies on SLA in different domains [**?**]. In the cloud context, Rak *et al.* proposed an approach to specify security requirement and to associate them to cloud services [**?**]. Mavrogeorgi *et al.* introduced a SLA management framework that allows the creation and enforcement of customized SLAs [**?**]. Leitner *et al.* presented an approach to monitor and predict SLA violations before they impacted the provider's SLA [**?**]. In general, proposals regarding SLAs in the deployment of services focus on two aspects: (*i*) approaches focusing on the life cycle of the SLA mainly interested in the contract negotiation phase between the cloud and the service consumer; and (*ii*) works monitoring contracts and cloud resources in order to avoid SLA violations, and consequently penalties due to its violation. In this sense, to the best of our knowledge, we have not identified any other approach that proposes the use of SLA associated to a data integration solution in a multi-cloud environment.

This thesis project intends to address data integration in a multi-cloud hybrid context. The originality of our approach consists in guiding the entire data integration solution taking into account (i) user preferences statements; (ii) SLA contracts exported by different cloud providers; and (iii) several QoS measures associated to data collections properties (for instance, trust, privacy, economic cost). The objective is to propose data integration strategies adapted to the vision of the economic model of the cloud.

In our work we consider an example from the domain of energy management. My directors are working on two national projects in this domain. So for instance, we assume we are interested in queries like: Give a list of energy providers that can provision 1000 KW-h, in the next 10 seconds, that are close to my city, with a cost of 0,50 Euro/KW-h and that are labeled as green? The question is how can the user efficiently obtain results for her queries such that they meet her QoS requirements, they respect her subscribed contracts with the involved cloud provider(s) and such that they do not neglect services contracts? Particularly, for queries that call several services deployed on different clouds. This work is part of an international collaboration with the DiMAp, Federal University of Rio Grande do Norte.

The project development is occurring naturally and well. In fact, this happens due to several aspects: (i) the close relationship and the easy access to my directors; (ii) the good frequency of meetings. Normally, we have at least one meeting for a week in order to evaluate the status of the work; (iii) the research center infrastructure and team; and (iv) the monthly group meetings where we can discuss about the development of the projects with other colleagues. In addition, there are meetings in another laboratory, LIRIS. These moments are important because we can have an external view and comments about our research.

3.2 Objectives

3.3 Synthesis of the work

During the first year of project, we have been working on the state of the art. The idea is to be aware of all types of publications close related to the thesis proposal. To reach this, we proceeded with a literature analysis using a systematic mapping methodology. Briefly, the methodology consists in retrieve papers from scientific databases using the same search string. These papers are filtered according to an inclusion and exclusion criteria that should be defined based on the research interests. The papers will be classified in different categories (called facets) and for each facet in a specific dimension. The facets and dimension are defined based on the authors' knowledge and interests. Taking the final papers collection, the abstracts should be read in order to classify each paper into the dimensions for each facet. The final objective of this first work is to show the research trends of data integration as a result of the emergence of the cloud and the characteristics associated to Big Data that require resources in order to be processed. In other terms, this methodology allowed us to identify trends and open issues regarding our research topic and proposing an approach that fills some gaps and proposes an original data integration solution according to current trends in the area. This work performed, we thus have written an article to be submitted to the 26th International Conference on Database and Expert Systems applications (DEXA 2015) in the beginning of March 2015. This work has been made in collaboration with the Informatics' Lab in Grenoble (co-supervisor Genoveva Vargas-Solar) and LIRIS Lab at INSA

(co-supervisor Nadia Benani). The table below presents our results. We retrieved 1718 papers. 114 papers were selected based on our inclusion and exclusion criteria. This final data collection builds the state of the art to the thesis and, in the next step, we will perform an in-depth analysis of theses papers in order to (i) analyze what have been made and (ii) propose my approach.

During the second year, we have been working on the development of our data integration approach. Given a query and a set of user preferences associated to it, the query execution process is divided in three phases. The figure 1 illustrates our data integration approach.

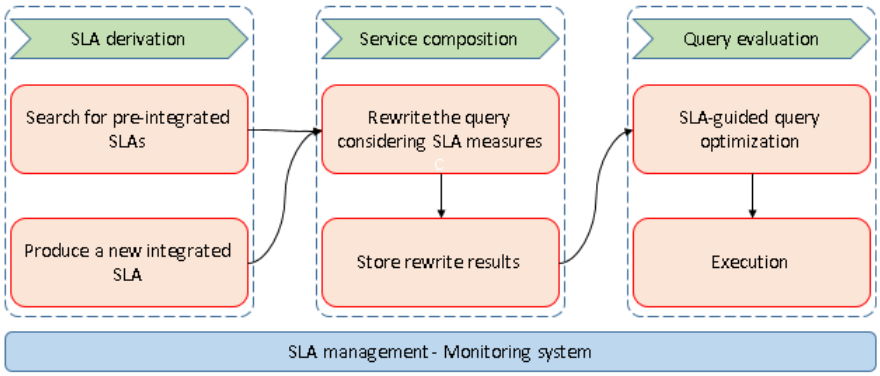


Figure 3.1: SLA-guided data integration approach

The first phase is the SLA derivation in which a SLA for the user request is created. It consists in looking for a (stored, integrated) SLA derived for a similar request. If a similar SLA is found, the request is forwarded to the query evaluation phase. Otherwise, a new SLA to the integration (called integrated SLA) is produced. The query is expressed as a service composition with associated user preferences. In the second phase, service composition, the query is rewritten in terms of different services considering the user preferences and the SLAs of each service involved in the composition. The rewriting result is stored for further uses. Finally, in the query evaluation phase, the query is optimized in terms of user preferences and SLAs concerning the consumed resources and the economic cost of the query. Once optimized, the query processed in the execution engine. In addition, we are assuming a SLA management module and monitoring system responsible to verify if the SLA contracts are being respected. Firstly, we have worked on the phases two in order to have an algorithm that will allow us to run important experiments to evaluate our approach.

[?] have proposed an algorithm for refining services composition. Their goal was to use user preferences to select and rank services in order to avoid the exponential problem while combining and producing compositions. Composition are incrementally produced until to reach a (predefined) desired number. In collaboration with our colleagues in Brazil (authors in [?]), we have worked on an adapted version of [?] to our data integration solution extending their data structure to map services

to the query, and adding the concepts of user preferences to the query and quality measures to the services. However, while performing the integration we have identified some design issues that made it useless to our approach: (i) their concept of user preferences are scores associated to services previous defined by the user while, for us, user preferences are quality requirements expected by the user concerning the whole integration; (ii) their algorithm accepts rewriting including calls to services that are not interest for the composition. Assuming that on the cloud each service has a price associated to its request, these composition that calls useless services produces an extra cost to the user. This adaptation process helped us to identify important issues to be applied to our own implementation such as developing an better approach to produce the combinations of services.

Based on the related work, we have developed and formalized the *Rhone* service-based query rewriting algorithm guided by service level agreements (SLA). The *Rhone* assumes that there are a set of quality measures associated to services which we suppose they are previously extract from their SLA. These measures will guide the service selection and the entire rewriting process. Our work address this issue and proposes the algorithm with two original aspects: (i) the user can express her quality preferences and associated them to his query; and (ii) service's quality aspects defined in SLAs guide the service selection and the whole rewriting process taking into consideration that services and rewritings should meet the user requirements, and the different cases of incompatibilities of SLAs, uncompleted SLA and the integration SLA. Given a set of abstract services, a set of concrete services and a user query (both defined in terms of abstract services), and a set of user quality preferences, the *Rhone* derives a set of service compositions that answer the query and that fulfill the quality preferences regarding the context of data service deployment. The algorithm consists in four steps: (i) *Selecting concrete services.* Similar to [**?**, **?**] our algorithm selects services based on the abstract services that exists in the query, but it includes two differences: first, a concrete service cannot be select if it contains an abstract service that is not present in the query; and second, the service' quality aspects (extracted from its SLA) must be in accordance with the user quality preferences; (ii) *creating mappings from concrete services to the query (called concrete service description (CSD))* inspired in [**?**] including also the information concerning the services' SLA; (iii) *combining CSDs*; and (iv) *producing rewritings* until fulfilling the user requirements according to the services' SLA. Each phase of the algorithm and each concept (query, concrete services, mapping rules, for instance) were formally specified and described.

In order to evaluate our approach and the *Rhone* algorithm, we began configuring a multi-cloud environment. We have searched for open source solutions instead of privates once they are (i) quite expensive; and (ii) do not allow to extend and access directly the different level of SLAs. The OpenStack was selected as our technology. We have installed and configured the different modules necessaries to the OpenStack. However, we have some issues: (i) the configuration and deployment of cloud infrastructure require important technical skills while configuring the network resources; (ii) it requires a powerful machine. Due to these reasons we have

configured a simulation of cloud run our experiments.

The first version of the *Rhone* was implemented using Java according to its formal definition. The algorithm was tested in a cloud simulation containing 35 services in its service registry. We have tested different types of query varying on the size and on the number of user preferences. Although our algorithm shares the same time performance problem as the previous approaches while combining compositions, the preliminary experiments have shown that the Rhone can enhance the quality in data integration by considering the user preferences and service's quality aspects extracted from service level agreements. Our approach avoid selecting and using services to produce compositions that are not interest to the user once they do not fulfill his quality requirements. In addition, as result we have submitted short paper to EDBT 2016.

We developed an improved version of the algorithm that better manages the manner in which lists of objects are managed. We have applied this version to a new set of experiments running 100 concrete services. With the results obtained from this experiment and the final version of our formalization, we are working on a new paper that included an extensive description of the algorithm and its evaluation to be submitted to ADBIS 2016 (deadline 27th March).

Currently, we have been working on the SLA model to data integration, on the schema for user SLA, cloud SLA and integration SLA, and on a scenario description to illustrate our approach. The result of this work is going to be described in a VLDB PhD workshop (to be submitted until 4 April).

Appendix A

Can Data Integration Quality be Enhanced on Multi-cloud using SLA?

Can Data Integration Quality be Enhanced on Multi-cloud using SLA?

Daniel A. S. Carvalho[1], Plácido A. Souza Neto[3], Genoveva Vargas-Solar[4], Nadia Bennani[2], Chirine Ghedira[1]

[1] Université Jean Moulin, Lyon 3 MAGELLAN, IAE – France
daniel.carvalho@univ-lyon3.fr, chirine.ghedira-guegan@univ-lyon3.fr
[2] CNRS INSA-Lyon, LIRIS, UMR5205 – France
nadia.bennani@insa-lyon.fr
[3] Instituto Federal do Rio Grande do Norte, Natal – Brazil
placido.neto@ifrn.edu.br
[4] CNRS, LIG-LAFMIA, Saint Martin d'Hères – France
genoveva.vargas@imag.fr

Abstract. This paper identifies trends and open issues regarding the use of SLA in data integration solutions on multi-cloud environments. Therefore it presents results of a Systematic Mapping [8] that analyzes the way SLA, data integration and multi-cloud environments are correlated in existing works. The main result is a classification scheme consisting of facets and dimensions namely (i) data integration environment (cloud; data warehouse; federated database; multi-cloud); (ii) data integration description (knowledge; metadata; schema); and (iii) data quality (confidentiality; privacy; security; SLA; data protection; data provenance). The proposed classification scheme is used to organize a collection of representative papers and discuss the numerical analysis about research trends in the domain.

Keywords: Systematic Mapping, Service Level Agreement, Data Integration, Multi-cloud Environment.

1 Introduction

The emergence of new architectures like the cloud opens new opportunities for data integration. The possibility of having unlimited access to cloud resources and the "pay as U go" model make it possible to change the hypothesis for processing big data collections. Instead of designing processes and algorithms taking into consideration limitations on resources availability, the cloud sets the focus on the economic cost implied when using resources and producing results.

Integrating and processing heterogeneous huge data collections (i.e., Big Data) calls for efficient methods for correlating, associating, and filtering them according to their "structural" characteristics (due to data variety) and their quality (veracity), e.g., trust, freshness, provenance, partial or total consistency.

Existing data integration techniques must be revisited considering weakly curated and modeled data sets provided by different services under different quality conditions. Data integration can be done according to (i) quality of service (QoS) requirements expressed by their consumers and (ii) Service Level Agreements (SLA) exported by the cloud providers that host huge data collections and deliver resources for executing the associated management processes. Yet, it is not an easy task to completely enforce SLAs particularly because consumers use several cloud providers to store, integrate and process the data they require under the specific conditions they expect. For example, a major concern when integrating data from different sources (services) is privacy that can be associated to the conditions in which integrated data collections are built and shared [11]. Naturally, a collaboration between cloud providers becomes necessary [4] but this should be done in a user-friendly way, with some degree of transparency.

In this context, the main contribution of our work is a classification scheme of existing works fully or partially addressing the problem of integrating data in multi-cloud environments taking into consideration an extended form of SLA. The classification scheme results from applying the methodology defined in [8] called *systematic mapping*. It consists of dimensions clustered into facets in which publications (i.e., papers) are aggregated according to frequencies (i.e., number of published papers). According to the methodology, the study consists in five interdependent steps including (i) the definition of a research scope by defining research questions; (ii) retrieving candidate papers by querying different scientific databases (e.g. IEEE, Citeseer, DBLP); (iii) selecting relevant papers that can be used for answering the research questions by defining inclusion and exclusion criteria; (iv) defining a classification scheme by analyzing the abstracts of the selected papers to identify the terms to be used as dimensions for classifying the papers; (v) producing a systematic mapping by sorting papers according to the classification scheme.

The remainder of this paper is organized as follows. Section 2 describes our study of data integration perspectives and the evolution of the research works that address some aspects of the problem. Section 3 gives a quantitative analysis of our study and identifies open issues in the field. Section 6 concludes the paper and discusses future work with reference to the stated problem.

2 Data integration challenges: classification scheme

The aim of our bibliographic study using the systematic mapping methodology [8] is to (i) categorize and quantify the key contributions and the evolution of the research done on *SLA-guided data integration in a multi-cloud environment* and (ii) discover open issues and limitations of existing works. Our study is guided by three research questions:

RQ1: *Which are the SLA measures that have been mostly applied in the cloud?* This question identifies the type of properties used for characterizing and evaluating the services provided by different clouds.

RQ2: *How have published papers on data integration evolved towards cloud topics?* This question is devoted to identify the way data integration problems addressed in the literature started to include issues introduced by the cloud.

RQ3: *In which way and in which context has data integration been linked to Quality of Service (QoS) measures in the literature?* The objective of this question is to understand which QoS measures have been used for evaluating data integration and to determine the conditions in which specific measures are particularly used.

2.1 Searching and screening papers

According to our research questions and our expertise in data integration we chose a set of keywords to define a complex query to be used for retrieving papers from four target publication databases: IEEE [5], ACM [6], Science Direct [7] and CiteSeerX [8]. We used the following conjunctive and disjunctive general query which was completed with associated terms from a thesaurus and rewritten according to the expression rules of advanced queries in each database:

("Service level agreement" AND ("Data integration" OR "Database integration") AND ("Cloud" OR "Multi-cloud "))

We retrieved a total of 1832 publications. As a result of the filtering process proposed by the systematic mapping methodology [8] we excluded 1718 publications. The number of papers included for building the final collection were 114 publications [9].

2.2 Defining classification facets

We analyzed the titles and abstracts of the papers derived in the previous phase using information retrieval techniques to identify frequent terms. We used these terms for proposing a classification scheme consisting of three facets that group dimensions. The following lines define the facets and dimensions of the classification scheme we propose.

Data Integration Environment: This facet groups the dimensions that characterize the architectures used for delivering data integration services (*data warehouse* and *federated database*) and architectures used for deploying these services (*cloud* and *multi-cloud*).

Data Integration Description: This facet groups the dimensions describing the approaches used for describing the databases content in order to integrate them. Data integration can be done by using *meta-data, schema*, and *knowledge*.

[5] http://ieeexplore.ieee.org/
[6] http://dl.acm.org/
[7] http://www.sciencedirect.com/
[8] http://citeseerx.ist.psu.edu/
[9] List of references available in: https://github.com/danielboni/DEXA-2015-Can-Data-Integration-Quality-be-Enhanced-on-Multi-cloud-using-SLA.git

Data Quality: This facet groups the dimensions representing data quality measures. Measures can be related directly to data for instance *confidentiality, privacy, security, protection and provenance* and to the conditions in which data is integrated and delivered (i.e., dimension *SLA*).

The original vision of our classification scheme is that of adding the notion of *quality* to data integration represented by the facets *data quality* and *SLA*. With these facets our classification scheme shows the aspects that must be considered when addressing data integration in the cloud taking into account (i) the quality of data, (ii) the systems that integrate data and (iii) the quality warranties that a data consumer can expect expressed in SLAs.

3 Quantitative Analysis

This section discusses the quantitative analysis presented in bubble charts that combine different facets. In order to observe the evolution of the publication trends we defined a time screen between the years 1998 and 2014 (see Figure 1). SLA has emerged when Cloud issues started to be addressed around 2009. The number of publications has increased as cloud infrastructures have become more popular and accessible. It seems that data integration is an open issue when it is combined with SLA and cloud trends. Less recent papers seem to be devoted to the way data is described under schemata or knowledge representation strategies. This could be due to the fact that these strategies are consolidated today and to the emergence of NoSQL approaches with their schema-less philosophy [9].

We combined facets for answering the research questions proposed for guiding our study. The following lines discuss the answers.

RQ1: Which are the SLA measures that have been mostly applied in the cloud?

The facets SLA expression, data integration description and contribution give elements for determining which SLA measures have been applied to the cloud (Figure 2). The resulting bubble chart shows that most contributions propose SLA models and that *privacy* and *security* (11 papers - 9.65%) are the most popular measures considered by SLA models for the cloud. These measures concern the network, information, data protection and confidentiality in the cloud. Most contributions propose SLA models (53 papers - 46.49%) but some languages (8 papers - 7.02%) have also emerged. *Data provenance* is also a measure that emerges but only in papers dealing with multi-cloud environments. Data integration is merely addressed by using schemata (12 papers - 10.53%) and meta-data (4 papers - 3.51%) particularly through models (34 papers - 29.82%) and tools (25 papers - 21.93%). Still, some works propose surveys (8 papers - 7.02%).

RQ2: How have published papers on data integration evolved towards cloud topics?

Combining the facets data integration environment, contribution and research it is possible to observe the evolution of publications on data integration towards the cloud (Figure 3). *Data warehouse* environments are the most

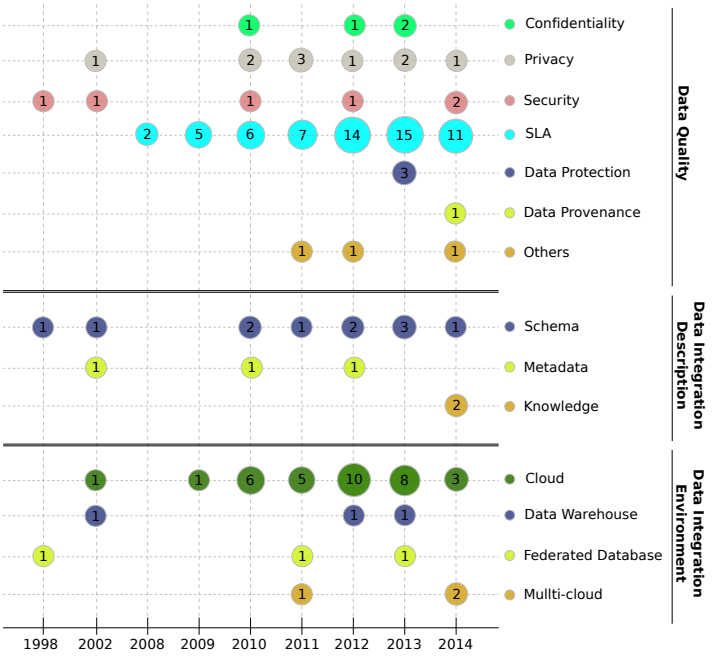


Fig. 1: Publications Per Year

common architecture. This can be explained by the increase of scientific and industrial applications needing to build integrated data sets for performing analysis and decision making tasks. The proposals are delivered as *models* (14 papers - 12.27%) and *tools* (18 papers - 15.78%) used for facilitating data integration, mostly done in the *cloud*. The most popular deployment environment of recent papers is the *cloud*. Given the importance and crucial need of data integration most papers present concrete solutions as algorithms, methods and systems (31 papers - 27.19%).

RQ3: In which way and in which context has data integration been linked to QoS measures in the literature?

We answered RQ3 by combining the facet *data quality* with the facets *data integration environment* and *data integration description* (Figure 4). Data integration and QoS measures are associated within environments like cloud (9.68%) and multi-cloud (4.39%).

According to our quantitative analysis we observe that QoS has started to be considered for integrating data. The cloud is becoming a popular environment to perform data integration in which security issues are most frequently addressed. We identify a promising research area concerning the need of studying SLA which is currently addressed for the cloud as a whole [7] but that needs to be specialized for data integration aspects. Therefore, it is important to iden-

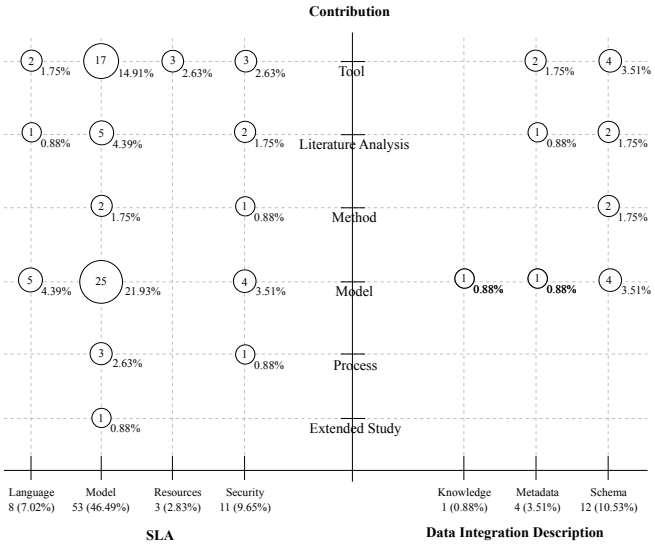


Fig. 2: Facets Contribution, SLA and Data Integration Description

tify the measures that characterize the quality of data and the quality measures associated to different phases of data integration. These phases include selecting data services, retrieving data, integrating and correlating them and building a query result that can be eventually stored and that must be delivered. The data integration phases are implemented by greedy algorithms and generate intermediate data that can be stored for further use. Therefore they consume storage, computing, processing and communication resources that have an associated economic cost. These resources must ensure some QoS guarantees to data consumers. This problem seems to be open in the domain, and we believe that it must be part of a new vision of data integration. We believe that it is possible to add and enhance the quality of data integration by including SLAs.

4 Enhancing Data Integration on Multi-Cloud environments with SLA

In order to illustrate our vision, let us consider an example from the domain of energy management. We assume we are interested in queries like: *Give a list of energy providers that can provision 1000 KW-h, in the next 10 seconds, that are close to my city, with a cost of 0,50 Euro/KW-h and that are labeled as green?* We consider a simplified SLA cloud contract inspired in the cheapest contract provided by Azure: *cost of $0,05 cents per call, 8 GB of I/O volume/month, free data transfer cost within the same region, 1 GB of storage.* Suppose that the user is ready to pay a maximum of *$5 as total query cost*; she requests

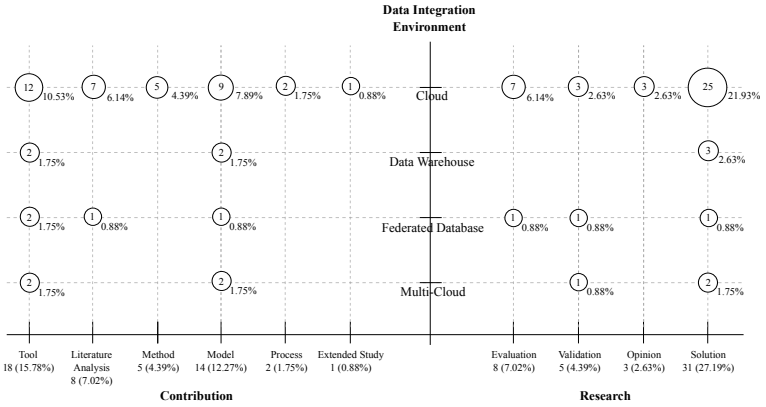


Fig. 3: Facets Data Integration Environment, Contribution and Research

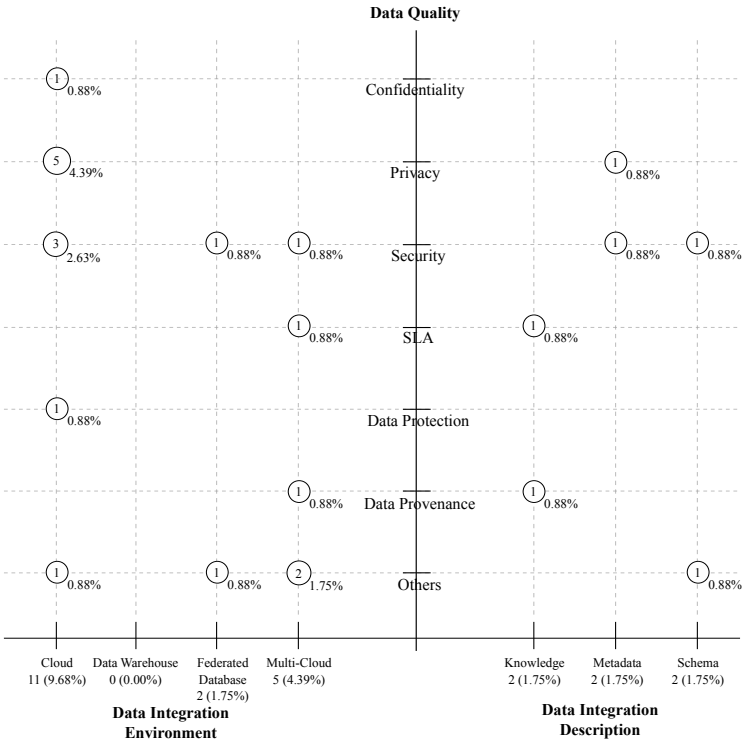


Fig. 4: Facets Data Quality, Data Integration Environment and Data Integration Description

that only *green* energy providers should be listed (provenance), with at least *85%* of precision of provided data, even if they are not fresh; she requires an availability rate of at least 90% and a response time of *0,01 s*. The question is how can the user efficiently obtain results for her queries such that they meet her QoS requirements, they respect her subscribed contracts with the involved cloud provider(s) and such that they do not neglect services contracts?

According to our classification scheme that resulted from our systematic mapping, we propose a new vision of data integration. This vision includes the description of the context in which data integration is done in modern environments. It also identifies the phases of the data integration process with their associated problems and challenges when they must include SLAs and QoS preferences expressed by data consumers.

4.1 Data integration context

We assume that data integration is done on a (multi)-cloud service oriented environment shown in Figure 5. We consider that data integration is done under new conditions with respect to the type of data sources, the environment where it is performed and the preferences of data consumers and the SLA.

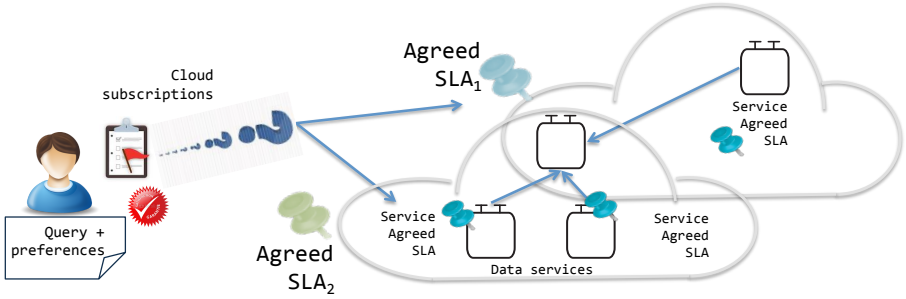


Fig. 5: New data integration context

There are data providers that are services possibly deployed in clouds and available through their API or in a REST architecture. We assume that each service exports an agreed SLA that specifies the economic cost per call, the maximum number of calls that can be done per day, the availability of the service, the average response time when a method is called, the reliability, the privacy of the produced data (whether they can be stored or not), the precision of their responses, freshness and provenance of the produced data.

Cloud providers define also their SLA contracts expressing subscription contracts that specify, the cost per request (cost/request), the volume of data that can be exchanged per month (I/0 volume/month), the cost of transferring data or applications within the same data centre or between data centres (datatransferCost/region), and storage space (storageSpace). For example some cloud providers

enable the customer to choose the zone to install PaaS services and deploy applications (e.g. zone 1 is Europe). If the customer wishes to deploy services in zone 1 but store data in zone 2 the transfer cost will change.

Some of these measures (cost/call, maxCall/day) are static and explicitly specified by the service provider. In contrast, the other measures should be computed by monitoring the conversations between the service and the applications that contact it.

In our vision a query expressed in an SQL-like language is associated to a set of QoS preferences expressing the requirements of the user. For example, the economic cost she is ready to pay for executing the query, the provenance of the data, the reputation of data services and the expected time response. The answer of such a query is the result of integrating data from different services according to a series of phases described in the following section.

4.2 SLA guided data integration

Given a query, its associated QoS preferences, cloud providers and services that can potentially be data providers (see Figure 5), SLA guided data integration can consist in four steps.

Generating a derived SLA The key and original aspect of our proposed data integration and provision process is to define a vertical mapping of user QoS preferences and agreed SLAs. This leads to a *derived SLA* that guides the evaluation of a query.

A query has associated preferences expressed as macroscopic constraints (i.e. user preferences statement): execution time, pay / no pay, data reliability, provenance, freshness, privacy, partial/full results, delivery mode. These constraints are coupled with the profile of the user which is in general stated in her cloud subscription (amount of assigned storage space, number of requests, I/O transferred Mega bytes, etc.).

Given agreed SLA's and a user preferences statement the challenge is to compute a *derived SLA* that maps SLA measures and preferences attributes. The derived SLA is defined as a set of measures that correspond to the user preferences computed as a function of different static, computed and hybrid measures. The *derived SLA* will guide the way the query will be evaluated, and the way results will be computed and delivered. Therefore, we propose to classify SLA measures to represent the relationship between fine grained measures used by agreed SLAs and coarse grained measures used in user preferences statements. It is also necessary to specify how to compute coarse grained measures with fine grained ones. For example, data precision will be computed as a function of availability, freshness and provenance exported by data services.

Filtering data services The derived SLA is used for filtering possible data services that can be used for answering the query. This is done using a set of matching algorithms based on graph structures and RDF specifications. This step may lead either to the rejection of integration in case of total incompatibility, or to a

negotiation between SLA which will lead to the proposal for a negotiated SLA integration and thus the need for an adaptive setting.

Query rewriting Given a set of data services that can potentially provide data for integrating the query result, we compute possible data service compositions that give partial or exhaustive results according to the derived SLA and the agreed SLA of each data service. The objective is to generate a number k of service compositions, combining as much as possible the services available such that the constraints of the derived SLA are verified.

Integrating a query result The service compositions are executed in one or several clouds where the user has a subscription. The execution cost of service compositions must fulfill the derived SLA (that expresses user requirements). In this phase we generate an execution plan considering the derived SLA and the subscription of the user to one or several clouds. We consider for example the economic cost determined by the data to be transferred, the number of external calls to services, data storage and results delivery costs and we decide how to use clouds resources for executing the composition. A first approach for performing this phase has been addressed in [5].

The first phase is an open issue for dealing with SLAs and particularly for adding quality dimensions to data integration. The problem is complex because SLA describe different elements participating in the data integration process: data services, cloud services at the different levels of the architecture (i.e., IaaS, PaaS, SaaS), data consumers subscriptions to cloud providers. The SLAs contain measures related to the way services are provided but also related to the data they provide. All these aspects must be considered for matching resources (i.e., services) with data consumers preferences. As shown in the following section and in our study SLA models and languages have been proposed. In contrast efficient preferences and SLAs matching algorithms need to be proposed to compute derived SLAs. Concerning the other data integration phases, they have been partially addressed by existing works, where some quality dimensions are considered (e.g., data privacy). In our vision there are open issues to be addressed in order to have solutions that consider SLA in order to enhance data integration in multi-cloud environments.

5 Related Works

Existing works addressing data integration can be grouped according to two different lines of research that correspond to the facets of the classification scheme that we propose: *(i)* data integration and services; and *(ii)* service level agreements and data integration.

5.1 Data integration and services

As shown in our classification scheme data integration description is a major topic. Existing works address knowledge oriented approaches for addressing the

problem. For example, [2] proposes a query rewriting method for achieving RDF data integration using SPARQL. The principle of the approach is to rewrite the RDF graph pattern of the query using data manipulation functions in order to: (i) solve the entity co-reference problem which can lead to ineffective data integration; and (ii) exploit ontology alignments with a particular interest in data manipulation. [3] introduces the Service Oriented Data Integration based on MapReduce System (SODIM) which combines data integration, service oriented architecture and distributed processing. SODIM works on a pool of collaborative services and can process a large number of databases represented as web services. The novelty of these approaches is that they perform data integration in service oriented contexts, particularly considering data services. They also take into consideration the requirement of computing resources for integrating data. Thus, they exploit parallel settings for implementation costly data integration processes.

A major concern when integrating data from different sources (services) is privacy that can be associated to the conditions in which integrated data collections are built and shared. [11] focusses on data privacy based on a privacy preserving repository in order to integrate data. Based on users' integration requirements, the repository supports the retrieval and integration of data across different services. [10] proposes an inter-cloud data integration system that considers a trade-off between users' privacy requirements and the cost for protecting and processing data. According to the users' privacy requirements, the query plan in the cloud repository creates the users' query. This query is subdivided into sub-queries that can be executed in service providers or on a cloud repository. Each option has its own privacy and processing costs. Thus the query plan executor decides the best location to execute the sub-query to meet privacy and cost constraints. As said before, the most popular "quality" property addressed in clouds when dealing with data is privacy. The majority of works addressing data integration in the cloud tackle security issues. We believe that other SLA measures need to be integrated in the data integration solutions if we want to provide solutions that cope to the characteristics of the cloud and the expectations of data consumers.

5.2 Service level agreement and data integration

Service level agreement (SLA) contracts have been widely adopted in the context of Cloud computing. Research contributions mainly concern (i) SLA negotiation phase (step in which the contracts are established between customers and providers) and (ii) monitoring and allocation of cloud resources to detect and avoid SLA violations.

[6] proposes a data integration model guided by SLAs in a Grid environment. Their architecture is subdivided into four parts: (i) a *SLA-based Resource Description Model* that describes the database resources; (ii) a *SLA-based Query Model* that normalizes the different queries based on the SLA information; (iii) an *SLA-based Matching Algorithm* selects the databases and finally (iv) a *SLA-based Evaluation Model* to obtain the final query solution. Considering our previous

work [1], to the best of our knowledge, we have not identified more proposals concerning the use of SLAs combined with a data integration approach in a multi-cloud context.

6 Conclusion and final remarks

This paper introduces the challenge of integrating data from distributed data services deployed on different cloud providers guided by service level agreements (SLA) and user preferences statements. The data integration problem is stated as a continuos data provision problem that has associated SLAs and that uses techniques for ensuring different qualities of delivered data (fresh, precise, partial). The problem statement was derived from a classification scheme that resulted from a study of existing publications identified by applying the systematic mapping method. Our contribution is the definition of a classification scheme that shows the aspects that characterize a modern vision of data integration done in multi-cloud environments and that can be enhanced by including SLAs in its process.

Current big data settings impose to consider SLA and different data delivery models. We believe that given the volume and the complexity of query evaluation that includes steps that imply greedy computations. It is important to combine and revisit well-known solutions adapted to these contexts. We are currently developing the strategies and algorithms of our vision applied to energy consumption applications and also to elections and political campaign data integration in order to guide decision making on campaign strategies.

References

1. N. Bennani, C. Ghedira-Guegan, M.A. Musicante, and G. Vargas-Solar. Sla-guided data integration on cloud environments. In *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, pages 934–935, June 2014.
2. Gianluca Correndo, Manuel Salvadores, Ian Millard, Hugh Glaser, and Nigel Shadbolt. SPARQL query rewriting for implementing data integration over linked data. In *Proceedings of the 1st International Workshop on Data Semantics - DataSem '10*, page 1, New York, New York, USA, March 2010. ACM Press.
3. Ghada ElSheikh, Mustafa Y. ElNainay, Saleh ElShehaby, and Mohamed S. Abougabal. SODIM: Service Oriented Data Integration based on MapReduce. *Alexandria Engineering Journal*, 52(3):313–318, September 2013.
4. Mohamad Hamze, Nader Mbarek, and Olivier Togni. Self-establishing a Service Level Agreement within autonomic cloud networking environment. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–4. IEEE, May 2014.
5. Carlos-Manuel Lopez-Enriquez, Victor Cuevas-Vicenttin, Genoveva Vargas-Solar, Christine Collet, and Jose-Luis Zechinelli-Martini. A planning-based service composition approach for data-centric workflows. In *First International Workshop on Knowledge Aware Service Oriented Applications, KASA 2014, Paris, France, November 3, 2014. Proceedings*, 2014.

6. Tiezheng Nie, Guangqi Wang, Derong Shen, Meifang Li, and Ge Yu. Sla-based data integration on database grids. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 2, pages 613–618, July 2007.
7. Carlos Pedrinaci, Jorge Cardoso, and Torsten Leidig. Linked USDL: A vocabulary for web-scale service trading. In *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pages 68–82, 2014.
8. Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK, 2008. British Computer Society.
9. Pramod J Sadalage and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012.
10. Yuan Tian, Biao Song, Jimupimg Park, and Eui nam Huh. Inter-cloud data integration system considering privacy and cost. In Jeng-Shyang Pan, Shyi-Ming Chen, and Ngoc Thanh Nguyen, editors, *ICCCI (1)*, volume 6421 of *Lecture Notes in Computer Science*, pages 195–204. Springer, 2010.
11. Stephen S. Yau and Yin Yin. A privacy preserving repository for data integration across data sharing services. *IEEE T. Services Computing*, 1(3):130–140, 2008.

Appendix B

Rhone: a quality-based query rewriting algorithm for data integration

Rhone: a quality-based query rewriting algorithm for data integration.

Daniel A. S. Carvalho[1], Plácido A. Souza Neto[2], Chirine Ghedira-Guegan[3], Nadia Bennani[4], and Genoveva Vargas-Solar[5]

[1] Université Jean Moulin Lyon 3, Centre de Recherche Magellan, IAE, France
daniel.carvalho@univ-lyon3.fr
[2] Instituto Federal do Rio Grande do Norte - IFRN, Natal, Brazil
placido.neto@ifrn.edu.br
[3] Université de Lyon, CNRS, IAE - Université Lyon 3, LIRIS, UMR5205, France
chirine.ghedira-guegan@univ-lyon3.fr
[4] LIRIS, INSA-Lyon, France
nadia.bennani@insa-lyon.fr
[5] CNRS-LIG, Grenoble, France
genoveva.vargas@imag .fr

Abstract. Nowadays, data provision is mostly done by data services. Data integration can be seen as composition of data services and data processing services that can deal with to integrate data collections. With the advent of cloud, producing service compositions is computationally costly. Furthermore, executing them can require a considerable amount of memory, storage and computing resources that can be provided by clouds. Our research focuses on how to enhance the results on the increase of cost on data integration in the new context of cloud. To do so, we present in this paper our original data integration approach which takes into account user's integration requirements while producing and delivering the results. The service selection and service composition are guided by the service level agreement - SLA exported by different services (from one or more clouds) and used by our matching algorithm (called *Rhone*) that addresses the query rewriting for data integration presented here as proof of concept.

Keywords: Data integration. Query rewriting. Query rewriting algorithm. Cloud computing. SLA.

1 Introduction

Current data integration implies consuming data from different data services and integrating the results according to different quality requirements related to data cost, provenance, privacy, reliability, availability, among others. Data services and data processing services can take advantage from the on-demand and *pay-as-you-go* model offered by the cloud architecture. The quality conditions and

penalties under which these services are delivered can be defined in contracts using Service Level Agreements (SLA).

Cloud services (data services, data processing services, for instance) and the cloud provider export their SLA specifying the level of services the user can expect from them. A user willing to integrate data establishes a contract with the cloud provider guided by an economic model that defines the services he/she can access, the conditions in which they can be accessed (duplication, geographical location) and their associated cost. Thus, for a given requirement, cloud services (from one or several cloud providers) are chosen to retrieve, process and integrate data according to the type of contracts he/she established with them.

In this context, data integration deals with a matching problem of the user's integration preferences which includes quality constraints and data requirements, and his/her specific cloud subscription with the SLA's provided by cloud services. Matching SLAs can imply dealing with heterogeneous SLA specifications and SLA-preferences incompatibilities. Moreover, even with the possibility of having an unlimited access to cloud resources, the user is limited to the resources and to the budget agreed by his/her cloud subscription. Inspired by these problems and Carrying on the ideas presented in a previous work [3], the aim of this paper is to introduce our service-based query rewriting algorithm guided by user preferences and SLAs which enhances the quality on the results integration in a multi-cloud context.

This paper is organized as follows. Section 2 discusses related works. Section 3 describes the *Rhone* algorithm and its formalization. Experiments and results are described in the section 4. Finally, section 5 concludes the paper and discusses future works.

2 Related works

In recent years, the cloud have been the most popular deployment environment for data integration [5]. Researches have proposed their works addressing this issue [6, 8]. Moreover, once query rewriting is strictly related to data integration, rewriting algorithms have been presented [1, 2, 4].

In [6], the authors introduced a system (called SODIM) which combines data integration, service-oriented architecture and distributed processing. The novelty of their approach is that they perform data integration in service oriented contexts, particularly considering data services. A major concern when integrating data from different sources (services) is privacy that can be associated to the conditions in which integrated data collections are built and shared. [8] proposed an inter-cloud data integration system that considers a trade-off between users' privacy requirements and the cost for protecting and processing data. According to the users' privacy requirements, the query plan in the cloud repository creates the users' query. Thus, the query plan executor decides the best location to execute the sub-query to meet privacy and cost constraints. This work is mostly interested in privacy and performance issues forgetting other users' integration requirements.

The main aspect in a data integration solution is the query rewriting. In the database domain, the query rewriting problem using views have been widely discussed [7]. Similarly, data integration can be seen in the service-oriented domain as a service composition problem in which given a query the objective is to lookup and compose data services that can contribute to produce a result. Generally, data integration solutions on the service-oriented domain deal with query rewriting problems. [2] proposed a query rewriting approach which processes queries on data provider services. [4] introduced a service composition framework to answer preference queries. Two algorithms are presented to rank the best rewritings based on previously computed scores. [1] presented an refinement algorithm based on *MiniCon* that produces and order rewritings according to user preferences and scores used to rank services that should be previously define by the user. Furthermore, they do not take into consideration user's integration requirements which can lead to produce rewritings that are not satisfactory in terms of quality requirements and constraints imposed by the user and the cloud environment. We assume that these requirements and constraints be expressed on SLAs. In the next section, we introduce our query rewriting algorithm that deals with SLAs while selecting, filtering and producing results.

3 Rhone service-based query rewriting algorithm

This section describes our quality-based query rewriting algorithm called *Rhone*. It is guided by user requirements and constraints, and services' quality features extracted after structuring service level agreements (SLA). Our algorithm has two original aspects: *first*, the user can express his/her quality requirements and constraints, and associate them to his/her queries; and, *second*, service's quality features defined on Service Level Agreements (SLA) guide service selection and the whole rewriting process.

3.1 Preliminaries

The idea behind our algorithm consists in deriving a set of service compositions that fulfill the users' integration requirements and constraints concerning the context of data service deployment given a set of *abstract services*, a set of *concrete services*, a user' *query* defined hereafter and a set of user' *integration requirements and constraints*.

Definition 1 (*Abstract service*). An *abstract service* describes the small piece of function that can be performed by a cloud service. For instance, retrieve patients, DNA information and personal information. The *abstract service* is defined as A ($\overline{I}$; $\overline{O}$) where: A is a name which identifies the *abstract service*; and $\overline{I}$ and $\overline{O}$ are a set of comma-separated input and output parameters, respectively. The table 1 exemplifies *abstract services* in a medical scenario. The decorations ? and ! differentiate input and output parameters, respectively.

***Abstract service* name**	**Description**
$A1$ ($d\_name?$; $p\_id!$)	Given a disease name d_name, $A1$ returns a list of infected patients' id p_id.
$A2$ ($p\_id?$; $p\_dna!$)	Given a patient id p_id, $A2$ returns her DNA information p_dna.
$A3$ ($p\_id?$; $p\_info!$)	Given a patient id p_id, $A3$ returns her personal information p_info.
$A4$ ($d\_name?$; $regions!$)	Given a disease name d_name, $A4$ returns the most affected region $regions$.

Table 1: List of *abstract services*.

Definition 2 (*Concrete service*). A *concrete service* is defined as a set of *abstract services*, and by its *quality features* extracted from its SLA according to the following grammar:

$$S(\overline{I}_h;\overline{O}_h) := A_1(\overline{I}_{1l};\overline{O}_{1l}), A_2(\overline{I}_{2l};\overline{O}_{2l}), .., A_f(\overline{I}_{fl};\overline{O}_{fl})[M_1, M_2, .., M_g]$$

The left-hand of the definition is called the *head*; and the right-hand is the *body*. A *concrete service* S includes a set of input $\overline{I}$ and output $\overline{O}$ variables, respectively. Variables in the *head* are identified by $\overline{I}_h$ and $\overline{O}_h$, and called *head* variables. They appear in the *head* and in the *body* definition. Variables appearing only in the *body* are identified by $\overline{I}_l$ and $\overline{O}_l$, and are called *local* variables. *Head* variables can be accessed and shared among different services. On the other hand, *local* variables can be used only by the service which define them.

Concrete services are defined in terms of *abstract services* $(A_1, A_2, .., A_n)$, and they include a set of service's quality features $(M_1, M_2, .., M_g)$ that are extracted from the SLA exported by the service S. M_i is in the form $x \otimes c$, where x is a special class of identifiers associated to the services; c is a constant; and $\otimes \in \{\geq, \leq, =, \neq, <, >\}$.

Let us consider the following *concrete services* specified using the *abstract services* previously presented to be used as examples to the algorithm:

```
S1(a?;b!) := A1(a?;b!) [availability > 98%, price per call = 0.2$]
S2(a?;b!) := A1(a?;b!) [availability > 98%, price per call = 0.1$]
S3(a?;b!) := A2(a?;b!) [availability > 99%, price per call = 0.1$]
S4(a?;b!) := A1(a?;p!), A2(p?; b!)
                       [availability > 98%, price per call = 0.1$]
S5(a?;b!) := A3(a?;b!) [availability > 98%, price per call = 0.0$]
S6(a?;b!,c!) := A1(a?;p!), A2(p?;b!), A3(p?;c!)
                       [availability > 99%, price per call = 0.2$]
S7(a?;b!) := A4(a?;b!) [availability > 99%, price per call = 0.2$]
```

For instance, $S1$ is written using the *abstract service* $A1$. a and b are *head* variables. *Availability* and *price per call* are identifiers associated to $S1$'s quality features with an associated constant value extracted from $S1$'s SLA.

Definition 3 (*User query*). A user *query* Q is defined as a set of *abstract services*, a set of *constraints*, and a set of *user integration requirements* in accordance with the following grammar:

$$Q(\overline{I}_h;\overline{O}_h) := A_1(\overline{I}_{1l};\overline{O}_{1l}), A_2(\overline{I}_{2l};\overline{O}_{2l}), .., A_n(\overline{I}_{nl};\overline{O}_{nl}), C_1, C_2, .., C_m[P_1, P_2, .., P_k]$$

The *query* definition is similar to a *concrete service* concerning the variables and *abstract services*. In addition, queries have constraints over the input or output variables ($C_1, C_2, .., C_m$). Constraints are used while querying the databases ($i.e$ in the "*where*" clause). The user *integration requirements* over the services or over service compositions are specified in $P_1, P_2, .., P_k$. C_i and P_j are in the form $x \otimes c$, where x is an identifier; c is a constant; and $\otimes \in \{\geq, \leq, =, \neq, <, >\}$.

User requirements can be of two types, single and composed. Single are associated directly to each service involved in the composition. Composed are linked to the entire composition. They are defined in terms of single requirements. For instance, the total response time is a composed preference obtained by adding the response time of each service involved in the composition.

Let us suppose a query specification based on a medical scenario in which doctor *Marcel* wants to query the personal and DNA information from patients that were infected by *flu*, using services with availability higher than 98%, price per call less than 0.2$ and integration total cost less than 5$. To achieve his needs, the *abstract services* $A1$, $A2$ and $A3$ should be composed as follows.

```
Q(dis?;dna!,info!) := A1(dis?;p!), A2(p?;dna!), A3(p?;info!), d = "flu",
             [availability > 98%, price per call < 0.2$, total cost < 5$]
```

The *Marcel's query* plan begins by retrieving infected patients ($A1$). This operation returns patients' ids p. The *abstract services* $A2$ and $A3$ use patient ids to return their DNA and personal information (dna and $info$). The *query* contains a constraint dis (disease name) equal to flu, and three identifiers define the user' *integration preferences* with their associated constant value.

The input data for the *Rhone* is a query and a set of concrete services. The result is a set of rewriting of the query in terms of concrete services, fulfilling the user preferences. The main function of the algorithm is divided in four steps: selecting candidate concrete services, creating candidate service descriptions and combining and producing rewritings. In the next sections, each step of the algorithm will be described.

3.2 Selecting candidate concrete services

While selecting services, the algorithm deals with three matching problems: (i) *quality features* matching, each *feature* in a query should be found in a concrete service. Moreover, the evaluation of a *feature* in a concrete service must satisfy the evaluation of a *feature* in the query; (ii) *abstract service* matching, abstract services can be matched if they have the same abstract function name and if the number and type of variable are equivalent; and (iii) *concrete service* matching, all abstract services in the concrete service must exist in the query, and

all of them should satisfy the *feature* and *abstract service matching* problems. Compared to [1], our algorithm includes the *features* matching and extends the *concrete service* matching by not accepting *concrete services* that covers useless *abstract services* to the query rewriting.

3.3 Candidate service description creation

After producing the set of candidate concrete services, the next step creates candidate service descriptions (CSDs). A CSD maps abstract services and variables of a concrete service into abstract services and variables of the query.
Definition 6 (candidate service description). A CSD is represented by an n-tuple:

$$\langle S, h, \varphi, G, P \rangle$$

where S is a *concrete service*. h are mappings between variables in the *head* of S to variables in the *body* of S. φ are mapping between variables in the *concrete service* to variables in the *query*. G is a set of *abstract services* covered by S. P is a set of *quality features* associated to the service S.

A CSD is created according to 4 rules: (*i*) for all head variables in a concrete service, the mapping h from the head to the body definition must exist; (*ii*) Head variables in concrete services can be mapped to head or local variables in the query; (*iii*) Local variables in concrete services can be mapped to head variables in the query; and (*iv*) Local variables in concrete services can be mapped to local variables in the query if and only if the concrete service covers all abstract services in the query that depend on this variable. The relation "depends" means that an output local variable is used as input in another abstract service.

Given the query Q and a list of candidate concrete services $\mathcal{L}_{\mathcal{S}}$, a list of CSDs $\mathcal{L}_{\mathcal{CSD}}$ is produced. A CSD is created only for candidate concrete services in which the mappings rules are being satisfied.

Given the candidate concrete services `S2`, `S3`, `S4` and `S5` selected in the previous step. The algorithm builds CSDs to `S2`, `S3` and `S5` once they satisfy all the mapping rules as follows. For instance, CSD_2 is produced to `S2` as follows: $\langle S2,\ h = \{a \rightarrow a,\ b \rightarrow b\},\ \varphi = \{a \rightarrow dis,\ b \rightarrow p\},\ G = \{A1\},\ P = \{availability > 98\%,\ price\ per\ call = 0.1\$\}\rangle$. However, a CSD for `S4` is not build because it violates the rule for local variables. It contains a local variable (p) mapped to a local variable in the query. Consequently, `S4` must cover all abstract services in the query depending on this variable, but the abstract service $A3$ is not covered.

3.4 Combining and producing rewritings step

Given the list of CSDs $\mathcal{L}_{\mathcal{CSD}}$ produced, the *Rhone* produces all possible combinations of its elements. Building combinations deals with a NP hard complexity problem. The effort to process combinations increases while the number of CSDs and abstract services in the query increases.

The last step identifies rewritings matching with the query and fulfilling the user preferences. The *Rhone* algorithm verifies if a given CSD list p is a rewriting of the original query. The function return *true* if (i) the number of abstract services resulting from the union of all CSDs in p is equal to the number of abstract services in the query; and (ii) the intersection of all abstract services in each CSD on p is empty. It means that is forbidden to have abstract services replicated among the set p.

Let us consider CSD_2, CSD_3 and CSD_5 are CSDs that refer to the concrete services `S2`, `S3` and `S5`, respectively. The *Rhone* produces combinations taking into account the part of the query covered by the service as follows:

$p_1 = \{CSD_2\}$
$p_2 = \{CSD_2, CSD_3\}$
$p_3 = \{CSD_2, CSD_3, CSD_5\}$

Given the combinations, the *Rhone* checks if each one of them is a valid rewriting of the original query.

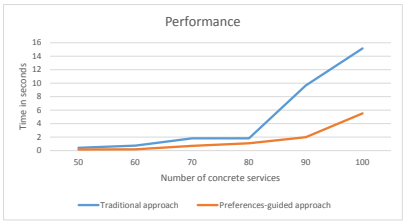
- p_1 and p_2 are not valid rewritings; their number of abstract services do not match with the number of abstract services in the query.
- p_3 is a valid rewriting; the number of abstract services matches and there is no repeated abstract service.

4 Evaluation

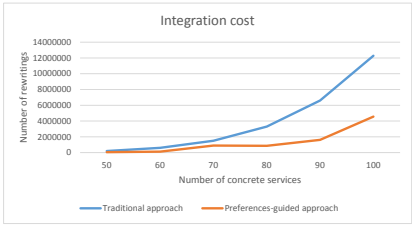
Different experiments were produced to analyze the algorithm's behavior. The *Rhone* [6], so far, was evaluated in a local controlled environment simulating a mono-cloud including a service registry of 100 concrete services. Some experiments were produced to analyze the its behavior concerning performance, and quality and cost of the integration. The experiments include two different approaches: (i) the *traditional approach* in which user preferences and SLAs are not considered; and (ii) the *preference-guided approach* (P-GA) which considers the users' integration requirements and SLAs. Figures 1a and 1b summarize our first results.

The results P-GA are promisingly. Our approach increases performance reducing rewriting number which allows to go straightforward to the rewriting solutions that are satisfactory avoiding any further backtrack and thus reducing successful integration time (Figure 1a). Moreover, using the P-GA to meet the user preferences, the quality of the rewritings produced has been enhanced and the integration economic cost has considerable reduced while delivering the expected results (Figure 1b).

[6] The *Rhone* algorithm is implemented in Java and it includes 15 java classes in which 14 of them model the basic concepts (*query*, *abstract services*, *concrete services*, etc), and 1 responsible to implement the core of the algorithm.



(a) Performance evaluation.



(b) Integration cost.

Fig. 1: *Rhone* execution evaluation.

5 Final Remarks and Future Works

Rhone still need to be tested in a large scale case and in a context of parallel multi-tenant to test efficacy. However, the results can show that the *Rhone* reduces the rewriting number and processing time while considering user preferences and services' quality aspects extracted from SLAs to guide the service selection and rewriting. We are currently performing a multi-cloud simulation in order to evaluate the performance of the *Rhone* in such context.

References

1. Ba, C., Costa, U., H. Ferrari, M., Ferre, R., A. Musicante, M., Peralta, V., Robert, S.: Preference-driven refinement of service compositions. In: Int. Conf. on Cloud Computing and Services Science. Proceedings of CLOSER 2014 (2014)
2. Barhamgi, M., Benslimane, D., Medjahed, B.: A query rewriting approach for web service composition. Services Computing, IEEE Transactions on Services Computing (2010)
3. Bennani, N., Ghedira-Guegan, C., Musicante, M., Vargas-Solar, G.: Sla-guided data integration on cloud environments. In: 2014 IEEE 7th International Conference on Cloud Computing (CLOUD). pp. 934–935 (June 2014)
4. Benouaret, K., Benslimane, D., Hadjali, A., Barhamgi, M.: FuDoCS: A Web Service Composition System Based on Fuzzy Dominance for Preference Query Answering (2011), 37th International Conference on Very Large Data Bases (VLDB 2011)
5. Carvalho, D.A.S., Souza Neto, P.A., Vargas-Solar, G., Bennani, N., Ghedira, C.: Can Data Integration Quality Be Enhanced on Multi-cloud Using SLA?, pp. 145–152. Springer International Publishing (2015)
6. ElSheikh, G., ElNainay, M.Y., ElShehaby, S., Abougabal, M.S.: SODIM: Service Oriented Data Integration based on MapReduce. Alexandria Engineering Journal (2013)
7. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10(4), 270–294 (Dec 2001)
8. Tian, Y., Song, B., Park, J., nam Huh, E.: Inter-cloud data integration system considering privacy and cost. In: ICCCI. Lecture Notes in Computer Science, vol. 6421, pp. 195–204. Springer (2010)