Incorporating Data Concerns into Query Languages for Data Services*

Muhammad Intizar Ali[1], Reinhard Pichler[1],
Hong-Linh Truong[2], and Schahram Dustdar[2]

[1] Database and Artificial Intelligence Group, Vienna University of Technology, Vienna, Austria
[2] Distributed Systems Group, Vienna University of Technology, Vienna, Austria
{intizar,pichler}@dbai.tuwien.ac.at,
{truong,dustdar}@infosys.tuwien.ac.at

Abstract. More and more organizations provide their data on the web via data services – also referred to as Data as a Service (DaaS). Data services combine the strength of database systems and query languages on the one hand with the benefits of service-oriented architecture on the other hand. Data services are increasingly used for data integration. The data provided via data services is often associated with data concerns like privacy, licensing, pricing, quality of data, etc. Hence, data integration tools not only have to mitigate the heterogeneity in data formats and query languages. In addition, also the various data concerns should be preserved when data is published and utilized. Moreover, data service selection and data selection should be based on these data concerns. Current Data Integration systems using data services lack the ability to preserve data concerns while querying multiple services in an integrated environment. In this paper, we design a new querying system which takes data concerns into account. To this end we discuss several models of data concern aware querying and select the best suited one for our system. We describe a querying system where data concern awareness is integrated directly into the XQuery language. We also report on an implementation and experimental evaluation of this system.

1 Introduction

More and more data providers are reaping the benefits of Web 2.0 technology and provide their data on the web either through web services, APIs (REST/SOAP), or data services – also referred to as *Data as a Service* (DaaS) [1,2]. Data services combine the strength of database systems and query languages on the one hand with the benefits of service-oriented architecture on the other hand. Many big companies have started to publish their data via query interfaces (rather than simple forms in html) so that the data can be easily reused, composed and integrated with other data sources. Amazon Public Data Sets on AWS [1], Google Squared [2] and UN data API's [3] are some prominent examples of publicly available data services.

* This work was supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032.
[1] http://aws.amazon.com/publicdatasets/
[2] http://www.google.com/squared
[3] http://www.undata-api.org/

Data services are increasingly used for data integration. Many tools and techniques are available to dynamically compose, integrate and execute different data services or sources[3,4]. These tools help to create situational applications by composing existing data services. The data thus published and processed is often associated with data concerns like privacy, licensing, pricing, quality of data, etc. Hence, data integration tools not only have to mitigate the heterogeneity in data formats and query languages. In addition, also the various data concerns should be preserved when data is published and utilized. Moreover, data service selection and data selection should be based on these data concerns. Consider for example, a meta-search query (a query that is posed against many data sources and selects the best possible integrated results among them). A user query will be executed on multiple data services registered at the integrated application. After integrating the results of all the data services, usually top-k results (where k is a constant value defined by the application) are returned. Now consider that different users have different priorities for the data selection, e.g., one user may be more interested in quality of data while another user is more concerned about the pricing. There is a clear need for an explicit system that (semi)automatically selects the most appropriate data service as well as data items for each user according to various data concerns.

Some data concerns like data quality, privacy, and quality of service (QoS) have long been studied in their respective domains of databases, data mining and web services. However, data services are different. Recently, the importance of distinguishing data services from web services has been recognized [5]. For instance, while licensing and quality of data are usually *static* for web services, they are dynamic for data services. Indeed, as the data gets "older", the licensing and the data quality (of which the up-to-dateness may be an important aspect) will most probably change. Moreover data concerns can be dynamically updated. Hence static information about usage permission or privacy cannot deal with the requirements of the dynamic integration application created by composing data services on the fly. Hence, new techniques are required to integrate data concerns into data services.

Among the various data concerns, *privacy* has received most attention. It has been studied in many different areas like data integration [6,7], data mining [8], and web services [9,10]. In [11] and similarly in [12], privacy has been studied in the context of data services. However a systematic integration of data concern awareness into data services is still missing to date. Moreover, previous approaches of dealing with data concerns like privacy do not directly integrate the data concern awareness into the query language, even though this would be very important for enabling the querying system to select the best suited data source for various parts of a given query. The goal of this work is to design a querying system (i) which can take arbitrary data concerns into account, (ii) which integrates the data concern awareness into the query language, and (iii) which automatically selects the appropriate data sources depending on current context, user requirements and data concerns.

Structure and Summary of Results. The main results of this paper are as follows.

• ***Roles and Concerns.*** In Section 2, we lay the foundations of our study by identifying the actors involved in data services and their specific concerns. In contrast to common web services, there are normally *three* actors involved, namely consumer, service provider, and also a data provider.

- ***Models.*** In Section 3, we present four possible models of data concern aware querying. We discuss the characteristics and virtues of each model and select the best suited one for our system.
- ***System.*** In Section 4, we come up with our data concern aware querying system. We describe how meta-data is organized and stored by this system and how data concern awareness is integrated directly into the XQuery language.
- ***Implementation and Evaluation.*** We report on an implementation of our data concern aware querying system and an evaluation on benchmark tests in Section 5.
- ***Conclusion.*** Finally, in Section 6, we give a conclusion and point out directions for future work.

Related Work. In [5], the authors give an overview of data concerns and discuss the different parties and their roles in data service creation and utilization. As mentioned above, there have been attempts to incorporate privacy concerns into data services [11,12]. Data source selection has been long discussed in the database and information retrieval community and different algorithms have been developed for optimal selection of the database [13]. Different frameworks and techniques are available for the best service selection in a web service environment [14]. In [15], the authors consider user preferences for data source selection while [16] used QoS attribute for the service selection. Selection of data services based on data concerns has not been considered so far. A query language extension method has been used to provide additional functionality for the integrated applications. A framework for data quality aware queries is presented by extending SQL query language [17] while privacy aware querying language has been designed for preserving privacy in distributed query evaluation [18]. To the best of our knowledge, there is no querying system available for data concern aware querying to integrate data services.

2 Data Services: Roles and Data Concerns

The concept of data services is based upon the service-oriented architecture (SOA), which includes standardized processes for accessing data "where it resides" irrespective of the platform. Data services take advantage of service-oriented architecture to offer users a mediator for integrating information from database systems and other structured or non-structured data sources. Data services thus realize a layer of software between the physical, distributed data sources and the applications or services which want to access the data. The data is exposed to the customer via a virtual data model. It is the responsibility of the data service to connect to the back-end data sources via the available interfaces and to map the physical data schema to the virtual data model. The applications and services using the data service leverage this virtual data model to access the required information and the data service software handles the collection and distribution of the data as needed from the physical instances of the data. Figure 1 gives an overview of the conceptual architecture of the data services.

The traditional approach to the design of data services is that the service provider and consumer agree on how the service shall be used. The terms of this agreement

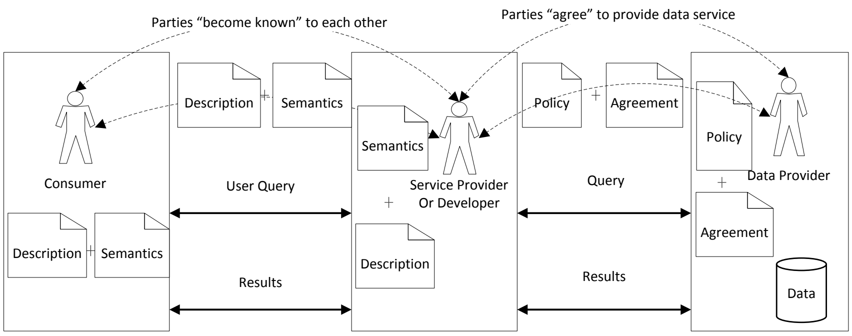
Parties "become known" to each other
Parties "agree" to provide data service
Consumer
Description
Semantics
User Query
Results
Service Provider Or Developer
Query
Policy
Agreement
Data Provider
Data

Fig. 1. Conceptual architecture of the data services

are usually static and are not altered over time. Typically, data services in the form of integration server products have connectors or adaptors built to connect applications together. A number of them have added or will add new connectors for data service applications such as salesforce (see `http://salesforce.com`). However, the static nature of agreements on data concerns is a severe drawback of the existing technology – in particular in dynamic data integration scenarios using mashups or cloud strategy. Moreover, several important aspects of data integration applications are not addressed in these solutions, such as (1) automatic updates, (2) changes of regulations, and (3) changes of the location of the data.

2.1 Roles of the Data Service

As can be seen in Figure 1, there are three actors involved in a data service architecture, namely Data Provider (DP), Service Provider (SP), and Consumer. Below we discuss each of them briefly.

Data Provider. The Data Provider is responsible to provide the actual data for the service. The DP is not necessarily the owner of the data, since data can also be outsourced. But it is the responsibility of the DP to ensure the availability of the data for the usage by the data service. All the concerns related to the data must also be communicated to the service provider. Some data concerns for the data provider like the privacy and permission concerns can vary from data item level to the whole data source level.

Service Provider. The Service Provider entity is the person or organization that provides an appropriate agent to implement a particular service. Service Provider is owner of the data service. After making the arrangement with the DP, the SP defines the functionality of the service. It is responsibility of the SP to make sure that all the concerns defined by the various data providers are preserved while the service is used.

Consumer. The consumer is a person or organization that wishes to make use of the data service. It will use a requester agent to exchange messages with the provider agent.

Table 1. Some data concerns associated with data services

Category	Scope	Data Concern	Description
Data Quality	Data Level	Timeliness	Defines the life time and freshness of the data
		Accuracy	Data correctness and consistency
		Completeness	Missing information in terms of null values etc.
		Availability	Defines possible access limitations
Quality of Service	Service Level	Performance	Defines performance of the data service, e.g. execution time, response time
		Reliability/availability	What is the failure probability and what is the recovery time in case of failure
		Dependability/Trust	Reputation, how trustful is the data service
		Service Location	Location of the service execution
License	Data Level	Usage Permission/Rights	How a data service can be used
		Data Location	Defines where the data resides
		Usage Fee	Defines the fee associated with the usage of the data or data service
		Law Enforcement	Defines laws which are used to deal with the use of the service

In most cases, the requester agent interacts with the provider agent to exchange messages. The requester entity and provider entity agree on the service description (e.g. a WSDL document) and the semantics that will govern the interaction between the requester agent and the provider agent.

2.2 Data Concerns for Data Services

Data services have different requirements and concerns compared with traditional web services. Data provided through data services is usually associated with many data concerns [5,19]. These concerns must be precisely described, organized and stored in such a way that the SP is capable of preserving these concerns while querying a data source.

For the sake of simplicity we have chosen a few of the most important concerns as shown in Table 1. Each concern can be categorized into a particular category depending on the type of the concern. We categorized the chosen data concerns into three categories namely (1) Data Quality, (2) Quality of Service, and (3) Licensing. Every category of the data concerns has its scope which will be either service level or data level (for further details, see Section 4.1). For example in Table 1, data concerns belonging to the category of quality of service have service level scope while data concerns belonging to the categories of data quality and licensing have data level scope.

2.3 DaaS Architecture with Data Concerns

Figure 2 describes the activities in a data service system with data concerns. As an initial step, both DP and SP mutually agree to provide the facility of the data service. The data provider is required to provide accessibility to its data for the service operations. The DP should provide a mechanism to access its data and additionally provide data

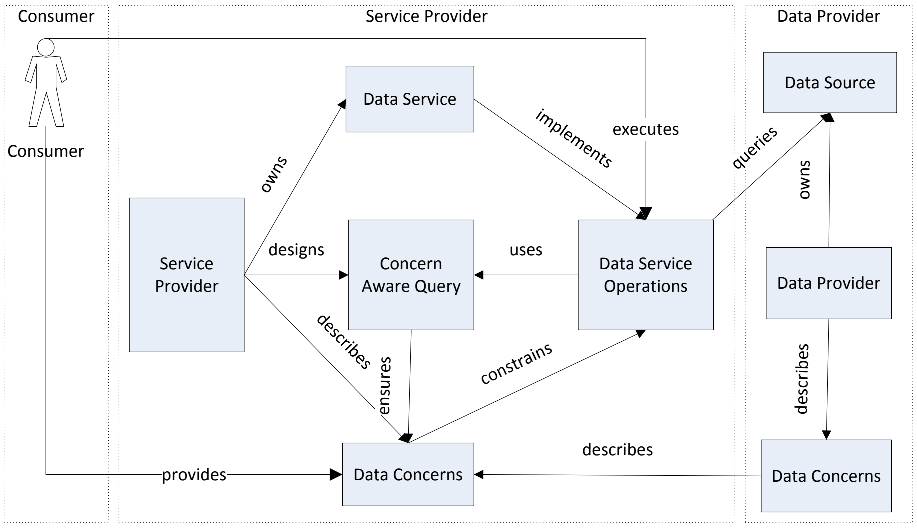

Fig. 2. Activities in data concern aware data service systems

concerns associated with the data. The user can also contribute to the data concerns meta-data by providing her preferences. Such information can be easily managed using profiling techniques. The service provider describes the service operations to access the data sources. This access can be realized via a query language, REST/SOAP or a parameterized query for structured data. The SP has to make sure that all the concerns defined by the DP are preserved.

3 Models of Data Concern Aware Querying

Figure 3 depicts four possible models for querying data services with concerns. As already discussed in Section 2 there are three actors/roles involved. For the sake of simplicity we assume that Data Provider and Service Provider have already agreed on the storage format of the data concerns: we assume that this information is stored in the form of an XML document whose schema will be described in Section 4.1.

The main difference between the models presented in Figure 3 is due to the way how the data concerns are associated with the queries that are posed against the data sources: The models in Figure 3(a) and (b) use the *"Querying With Concerns"* paradigm. In this case, the system sends a pair $\langle Q, C \rangle$ to the data source, where Q is the query for the data service and C is the collection of data concerns associated with the query. Each data service must be capable of preserving these concerns by using its own individual querying system. The models in Figure 3(c) and (d) are based on *"Concern Aware Querying"*. The service provider has the concerns either stored locally or fetched dynamically and is capable of writing queries in such a way that they preserves all the data concerns associated with the particular query and its relevant data source.

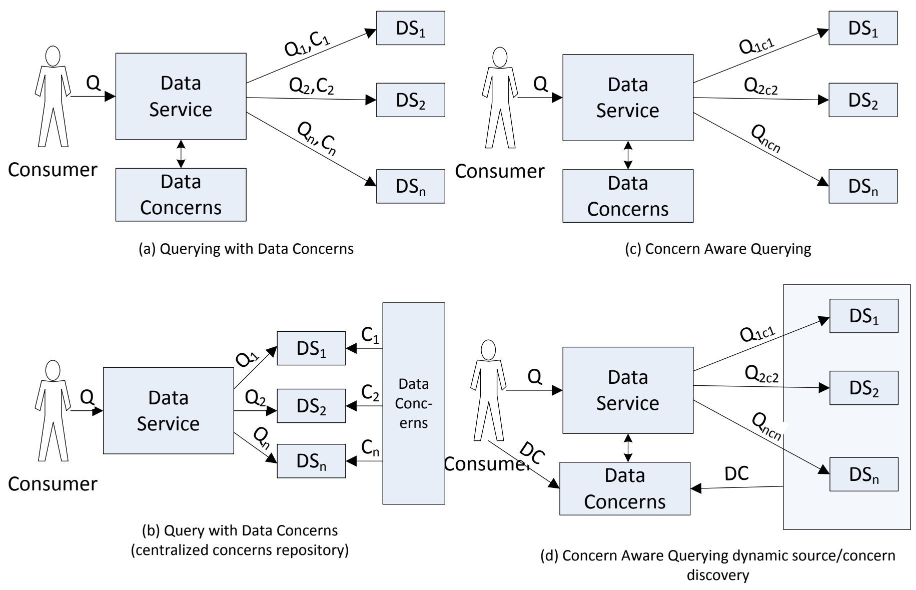
Consumer
Q
Data Service
Data Concerns
Q1,C1
Q2,C2
Qn,Cn
DS1
DS2
DSn
(a) Querying with Data Concerns
Q1c1
Q2c2
Qncn
(c) Concern Aware Querying
Q1
Q2
Qn
C1
C2
Cn
Data Conc-erns
(b) Query with Data Concerns (centralized concerns repository)
DC
(d) Concern Aware Querying dynamic source/concern discovery

Fig. 3. Data concerns aware querying models

Below we briefly discuss each of the models shown in Figure 3

Model (a): Querying with Data Concerns. Figure 3(a) shows the basic model of querying with data concerns. DP and SP have mutually agreed and have accumulated their data concerns. These data concerns are stored by the SP. A user query Q is subdivided into multiple sub-queries for multiple data sources. The SP must be capable of retrieving the data concerns meta-data of a particular data source and attaching it to the query. The data provider must be capable of taking care of the supplied concerns while executing the query and returning the results.

Model (b): Query with Data Concerns with Centralized Repository. This model deals with the generic data concerns stored in a centralized repository. Such a model is best suited for an inter-organization data integration system, where data concerns are homogeneous for all of the data sources of the organization. As shown in Figure 3(b) this model is very similar to the existing data integration systems. The SP sends user queries to multiple data sources and assumes that the data sources are capable of preserving the data concerns by accessing a centralized repository of data concerns.

Model (c): Concern Aware Querying. The concern aware querying model of Figure 3(c) is a data service capable of re-writing user queries in such a way that all the concerns associated with a particular data service are incorporated into the query itself. All the data sources and their concerns are already registered at the SP and the required meta-data information is locally stored by the SP. The SP divides a user query into multiple sub-queries in such a way that all the applicable data concerns (which are stored within meta-data file locally stored at the SP) are incorporated into these sub-queries.

Model (d): Concern aware Querying Model with Dynamic Discovery. Contrary to the legacy data integration systems for databases, data services usually have no prior knowledge of the schema or the data source. Mostly data sources are discovered dynamically and the most suitable data source has to be selected. Of course, storing meta-data information of all data sources at internet scale is out of the question. Figure 3(d) shows the model for concern aware querying with dynamic discovery of data sources and their associated data concerns. The SP stores data concerns meta-data as in the concern aware querying model, but at the same time it is capable of discovering data services dynamically and creating meta-data of data concerns by fetching them dynamically. These concerns are then applied to queries for a particular data source. This model also allows the consumer to provide her concerns and preferences within the query.

4 Concern Aware Querying

For our system, we have chosen the concern aware querying model as shown in Figure 3(d) since it is the most flexible and most powerful of the models discussed in the previous section. Our system stores the meta-data information of the concerns of all three stake holders of the data service and is capable of querying data concerns dynamically. All the three actors of a data service have the possibility to contribute their concerns to the meta-data. Each data provider provides its data concerns associated with its data along with its data. Similarly each consumer has the option to add her preferences/concerns with her query. The service provider can add its concerns to form an aggregated data concerns meta-data which will be utilized by the query language to write concern aware queries. A data concern tree (dct) is generated from the available meta-data and is associated with each of the data sources.

We have extended the XQuery language to make it concern aware, with the introduction of special keywords for mentioning data concerns within the query. The query parser looks into the data concern trees attached to the available data sources and executes the query by selecting the most suitable data sources and data items for the particular user query. Below we discuss the concern aware querying system in detail.

4.1 Data Concerns Collection

For storing the meta-data information of data concerns of a particular data source, we have decided to use XML because of its standardization and platform independence. Figure 4 shows the schema definition for the meta-data for the concern aware querying. A data concern tree will be generated by using the meta-data information available in the attached XML file. The schema shown in Figure 4 describes the data concerns as shown in Table 1. The data concern collection file can be expanded to any number of concerns depending on the type and requirement of the SP or DP. The data concern tree can be subdivided into two categories based on the scope of the data concerns.

Service Level Concerns. Data concerns whose scope is the entire data service are called service level concerns. The data concerns tree of a data service contains exactly one value for any of the service level concerns. For instance, consider the performance concern of QoS category of a service. Clearly, a data service can have only one calculated value for its performance data concern.

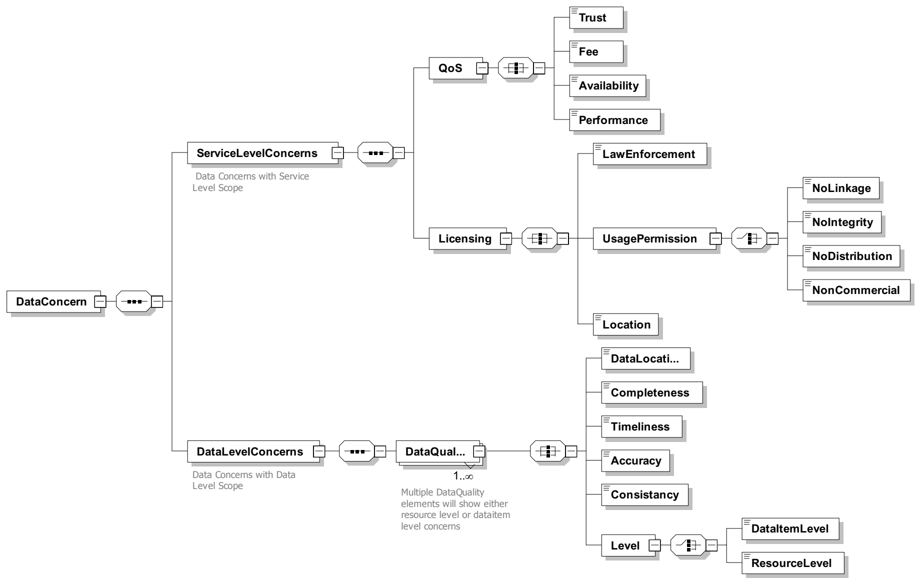

Fig. 4. Data Concerns Tree

Data Level Concerns. Data concerns whose scope is a single data item or a collection of data items are called data level concerns. Data level concerns can have multiple values in the data concern tree attached to some data source. If a data source returns a node or a list of nodes of an XML document as a result of a user query (written in XQuery), then a data level concern can be described for each of the nodes in this result and for each element in the subtrees rooted at these nodes.

The data concern tree is populated by the values of data concerns depending on the type of each data concern. There are two possible types of data concerns based on their value, namely (i) *boolean data concerns*, which can have either the value true or false, e.g. usage permission for commercial purposes can either be allowed or restricted by the data provider, and (ii) *value based data concerns*, which can have any value within a specific range e.g. the performance concern of QoS of a data service or the completeness concern of a data item can have a specific value inside a pre-described range. Different algorithms are available for the calculation of QoS values of a data service or data quality measurements for a data item, which is outside the scope of this paper. We assume that after applying any of the available algorithms and techniques a fixed value (positive integer) is provided for the value based data concerns within a range between 1 and 10, where 10 is the highest level.

4.2 Query Processing

If concern aware querying is used and a query contains data concerns, then the SP iterates through the data concern trees of all available data sources. Using data concern information provided in the query and the meta-data tree of data concerns, the data concern aware querying system is able to select the most suitable data service for a

```
input : Concern Aware Query Q, DS = $\{ds_1, ds_2,...,ds_n\}$
output: DS$' \subseteq$ DS
DS$'$ = $\{\phi\}$;
for each $ds_i \in DS$ do
    flag = true;
    for each $Q.SLC_b$ do
        if $Q.SLC_b \neq ds.SLC_b$ then flag = false
        end
    end
    if flag == true then
        for each $Q.SLC_b$ do
            if $Q.SLC_v \geq ds.SLC_v$ then flag = false
            end
        end
    end
    if flag == true then DS$' \leftarrow ds_i$
    end
end
Return DS$'$;
```

Algorithm 1. Data Source Selection based on SLC

particular query. Algorithm 1 shows the data source selection by assuring both types (i.e., boolean and value based) of service level concerns. A concern aware query and a set of available data sources is provided as input. After evaluating the Algorithm 1, the most suitable data source or set of data sources (in case multiple data sources assure the desired concerns) is returned for data querying. Once the input is provided, the algorithm starts iterating through the available data sources. For each of the service level concerns mentioned inside a concern aware query the system verifies whether that particular service level concern is also supported by the data source. If a particular service level concern is supported by a certain data source, as a next step the system compares the required value of service level concern mentioned inside the query with the value of service level concern of the data source in its data concern tree. If both conditions are met, a data source is included in the list of suitable data sources for that particular query. Similarly, DLCs are also evaluated using the same technique mentioned in Algorithm 1 for data selection to assure data level concerns.

4.3 Concern Aware XQuery

We have chosen XQuery because it is the de facto standard language for XML data and because of its capability to execute distributed queries over heterogeneous data sources [20]. Now consider a dynamic data sources integration system as described in Figure 3(d), where a user query can be executed on multiple data sources which perform the same task and one data concern tree is attached to each data source. Current query languages for semi-structured data like XQuery or SPARQL need to provide the URI or location of the data source manually. In order to select the most appropriate data source from the available data sources which perform the same task, we have extended the XQuery syntax by providing additional keywords to make it concern aware.

For service level concerns we introduce a new keyword “SLC”. A comma separated list of service level concerns inside square brackets will immediately follow the “doc” function of the XQuery language or wherever the URI for the location of data sources is provided. The syntax can be illustrated as

doc(“xmldoc.xml”)
SLC [ServiceLevelConcern1 op value, ... , ServiceLevelConcernN op value]

where N is a finite number of service level concerns in a concern aware query (formulated in XQuery) and op can be any of the basic comparison operators.

For data level concerns we used the simple notation of attaching each data level concern with the variables defined within concern aware XQuery. Variables having data level concerns will be written inside square brackets with one data level concern attached with the variable. The syntax can be illustrated as

[$varA.DataLevelConcern op value]

where $varA can be any variable defined within XQuery and op can be any of the basic comparison operators.

5 Implementation and Evaluation

5.1 Implementation

We have implemented a data concern aware XQuery tool to support the idea of concern aware querying. To store XML data sources, we use the open source native XML database system eXist-db[4] (release 1.4.0). Our concern aware querying tool prototype is implemented in Java (JDK 6) on top of the XQuery processing facility provided by eXist-db. When a concern aware query is submitted to the tool it implements the methods described above and selects the most suitable data sources for a particular query. Once the most suitable data sources have been selected, our tool excludes concern aware querying clauses and sends standard XQuery to the selected data sources.

Figure 5 shows a sample query over the XMARK benchmark data of a web application for auctions, which return a list of person names and items bought by them within Europe. A comma separated list of all the service level concerns is described within square brackets using additional keywords defined for the service level concerns immediately after the URL/location of the data source mentioned in XQuery. In the above example two service level concerns are evaluated. The statement “performance $>$ 7” will select only those data services which have a performance attribute value greater than 7 in their data concern tree, while statement “CommercialUsagePermission=true” will make sure that data returned as a result from the query will have no restriction on the commercial usage of the data. For the purpose of simple illustration we allow using data level concerns related to particular data elements within an XQuery [QName.DLC]

[4] `http://exist.sourceforge.net/`

```
let $auction :=
doc("auction.xml") SLC[performance > 7,
CommercialUsagePermission = true]
 return let [$ca.timeliness] :=
$auction/site/closed_auctions/closed_auction return
let
    [$ei.completeness > 4] := $auction/site/regions/europe/item
for $p in $auction/site/people/person
let $a :=
  for $t in $ca
  where $p/@id = $t/buyer/@person
  return
    let $n := for $t2 in $ei
    where $t/itemref/@item = $t2/@id return $t2
    return <item>{$n/name/text()} </item>
return <person name="{$p/name/text()}">{$a}</person>
```

Fig. 5. Sample Concern Aware XQuery

as part of concern aware querying. For example the statement, "$ei.completeness > 4" in the above example of Figure 5 will make sure that only those values of the variable "ei" are selected which have completeness data concern value greater than 4. Boolean data concerns can have either a true or false value. Basic comparison operators can be applied for the calculation of value based data concerns.

5.2 Experimental Application: Distributed Extended XQuery for Data Integration (DeXIN)

Our data concern aware XQuery tool is built upon the DeXIN system [20,21], which is a web based system to integrate data over heterogeneous data sources. DeXIN extends the XQuery language to support SPARQL queries inside XQuery, thus facilitating the integration of data modeled in XML, RDF, and OWL. DeXIN supports the data integration of XML and RDF data without the need of transforming large data sources into common format. It is a powerful tool for knowledgeable users or web applications to facilitate querying XML data and reasoning over Semantic Web data simultaneously.

To build our data concern aware XQuery system, we have incorporated the data concern awareness into DeXIN. Now DeXIN can integrate heterogeneous distributed data sources while preserving their individual data concerns. It is worth mentioning that by incorporating data concerns into the DeXIN system not only XQuery capabilities are enhanced for data concern assurance but SPARQL is also enhanced with data concern awareness using the DeXIN tool.

5.3 Evaluation

In order to evaluate the performance and concreteness of our concern aware querying tool, we have conducted tests with realistically large data sets. As a proof of concept we

Table 2. Data sources with varying size and data concerns

Data Source Name	File Size	No. of Copies	No. of SLC	No. of DLC
Auction1.xml	30 MB	20	3	5
Auction2.xml	70 MB	30	3	8
Auction3.xml	100 MB	10	3	4

have evaluated our system on XML benchmark data. We used the XMARK [5] benchmark data set for experimental analysis. XMARK is a popular XML benchmark and models an internet auction application. We made three subsets of varying size of auction data provided by XMARK. Table 2 shows the details of the data services used for experimental analysis. We made further copies of the subset of the XMARK auction data and defined each as a data service. Each data service assures a varying number of service level concerns and data level concerns. The resulting data services were constructed with the same functionality but with different concerns.

Due to the unavailability of data services which support data concerns, we randomly generated data concern tree meta-data for each data service and assigned different values to both service and data level concerns. To assure the distribution of the data services we set up a testbed which includes 3 computers (Intel(R) Core(TM)2 CPU, 1.86 Ghz, 2GB RAM), one running SUSE Linux with kernel version 2.6 while the other two running Windows XP. The machines were connected over a standard 100Mbit/S network connection. An open sources native XML database eXist is installed on each system to store XML data. We utilize the eXist XQuery processor to execute XQuery queries.

We used 20 different sample queries [6] provided with the benchmark and executed each of them with different data concern values. There was no reported failure in the concern aware query execution and all the provided data concerns were assured, which proves the suitability of our tool and the potential for its incorporation into any data service integration application.

6 Conclusions and Future Work

In this work, we have designed a querying system which is capable of taking several kinds of data concerns into account. We have provided a basic model in which we concentrate on three concerns, namely data quality, quality of service, and licensing. However, our approach is generic in the sense that one can incorporate arbitrary data concerns. Indeed, one item on our agenda for future work will be to integrate further data concerns like pricing, data security, auditing model, etc. Another important goal for future work is the integration of our querying system into a powerful mash-up tool. So far, our querying system is designed to access data sources via XQuery. In the future, we want our system to access also data sources which expose their data via web services.

[5] `http://www.xml-benchmark.org/`

[6] `http://www.ins.cwi.nl/projects/xmark/Assets/xmlquery.txt`

References

1. Dan, A., Johnson, R., Arsanjani, A.: Information as a service: Modeling and realization. In: Proc. SDSOA 2007. IEEE Computer Society (2007)
2. Hacigümüs, H., Mehrotra, S., Iyer, B.R.: Providing database as a service. In: Proc. ICDE 2002. IEEE Computer Society (2002)
3. Mykletun, E., Tsudik, G.: Aggregation Queries in the Database-As-a-Service Model. In: Damiani, E., Liu, P. (eds.) DBSec 2006. LNCS, vol. 4127, pp. 89–103. Springer, Heidelberg (2006)
4. Virtuoso universal server, `http://virtuoso.openlinksw.com/`
5. Truong, H.L., Dustdar, S.: On analyzing and specifying concerns for data as a service. In: Proc. APSCC 2009, pp. 87–94. IEEE (2009)
6. Bhowmick, S.S., Gruenwald, L., Iwaihara, M., Chatvichienchai, S.: Private-iye: A framework for privacy preserving data integration. In: Proc. ICDE Workshops 2006, p. 91. IEEE Computer Society (2006)
7. Clifton, C., Kantarcioglu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A.K., Suciu, D.: Privacy-preserving data integration and sharing. In: Proc. DMKD 2004, pp. 19–26. ACM (2004)
8. Zhang, N., Zhao, W.: Privacy-preserving data mining systems. IEEE Computer 40, 52–58 (2007)
9. Kobsa, A.: Tailoring Privacy to Users' Needs. In: Bauer, M., Gmytrasiewicz, P.J., Vassileva, J. (eds.) UM 2001. LNCS (LNAI), vol. 2109, pp. 303–313. Springer, Heidelberg (2001)
10. Creese, S., Hopkins, P., Pearson, S., Shen, Y.: Data Protection-Aware Design for Cloud Services. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 119–130. Springer, Heidelberg (2009)
11. Mrissa, M., Tbahriti, S.E., Truong, H.L.: Privacy model and annotation for daas. In: Proc. ECOWS 2010, pp. 3–10. IEEE Computer Society (2010)
12. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proc. SIGMOD 2009, pp. 19–30. ACM (2009)
13. French, J.C., Powell, A.L., Callan, J.P., Viles, C.L., Emmitt, T., Prey, K.J., Mou, Y.: Comparing the performance of database selection algorithms. In: Proc. SIGIR 1999, pp. 238–245. ACM (1999)
14. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing 8, 84–93 (2004)
15. Boulakia, S.C., Lair, S., Stransky, N., Graziani, S., Radvanyi, F., Barillot, E., Froidevaux, C.: Selecting biomedical data sources according to user preferences. In: ISMB/ECCB 2004, pp. 86–93 (2004)
16. Liu, Y., Ngu, A.H., Zeng, L.Z.: Qos computation and policing in dynamic web service selection. In: Proc. WWW Alt. 2004, pp. 66–73. ACM (2004)
17. Yeganeh, N.K., Sadiq, S.W., Deng, K., Zhou, X.: Data Quality Aware Queries in Collaborative Information Systems. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 39–50. Springer, Heidelberg (2009)
18. Farnan, N.L., Lee, A.J., Yu, T.: Investigating privacy-aware distributed query evaluation. In: Proc. WPES 2010, pp. 43–52. ACM (2010)
19. Truong, H.L., Dustdar, S.: On evaluating and publishing data concerns for data as a service. In: Proc. APSCC 2010, pp. 363–370. IEEE Computer Society (2010)
20. Ali, M.I., Pichler, R., Truong, H.L., Dustdar, S.: DeXIN: An Extensible Framework for Distributed XQuery over Heterogeneous Data Sources. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 172–183. Springer, Heidelberg (2009)
21. Ali, M.I., Pichler, R., Truong, H.L., Dustdar, S.: On Using Distributed Extended XQuery for Web Data Sources as Services. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 497–500. Springer, Heidelberg (2009)