QUEPA: QUerying and Exploring a Polystore by Augmentation

Antonio Maccioni
Roma Tre University
Rome, Italy
maccioni@inf.uniroma3.it

Edoardo Basili
Roma Tre University
Rome, Italy
basili@inf.uniroma3.it

Riccardo Torlone
Roma Tre University
Rome, Italy
torlone@inf.uniroma3.it

ABSTRACT

Polystore systems (or simply polystores) have been recently proposed to support a common scenario in which enterprise data are stored in a variety of database technologies relying on different data models and languages. Polystores provide a loosely coupled integration of data sources and support the direct access, with the local language, to each specific storage engine to exploit its distinctive features. Given the absence of a global schema, new challenges for accessing data arise in these environments. In fact, it is usually hard to know in advance if a query to a specific data store can be satisfied with data stored elsewhere in the polystore.

QUEPA addresses these issues by introducing *augmented search* and *augmented exploration* in a polystore, two access methods based on the automatic enrichment of the result of a query over a storage system with related data in the rest of the polystore. These features do not impact on the applications running on top of the polystore and are compatible with the most common database systems. QUEPA implements in this way a lightweight mechanism for data integration in the polystore and operates in a plug-and-play mode, thus reducing the need for ad-hoc configurations and for middleware layers involving standard APIs, unified query languages or shared data models. In our demonstration audience can experience with the augmentation construct by using the native query languages of the database systems available in the polystore.

1. INTRODUCTION

Polystores are the result of the "one size does not fit all" philosophy in a world where taking care of the peculiarities of different kind of data became necessary [5, 8]. Let us consider, as a practical example of a polystore environment, the databases of a company called ACME selling music online. As shown in Figure 1, each ACME department uses a storage system that best fits its specific business objectives: (i) the sales department requires ACID transactions and uses a relational database for the *purchases* and the *inventory* of products, as provided by sellers, (ii) a marketing department uses a document store for a music and customer *catalogue*, where each item is represented by a JSON document, and (iii) a business intelligence department uses a graph database for representing *similarities* among items. In addition, there exists a key-value store containing the products currently on *clearance*, which is shared among the three departments above.

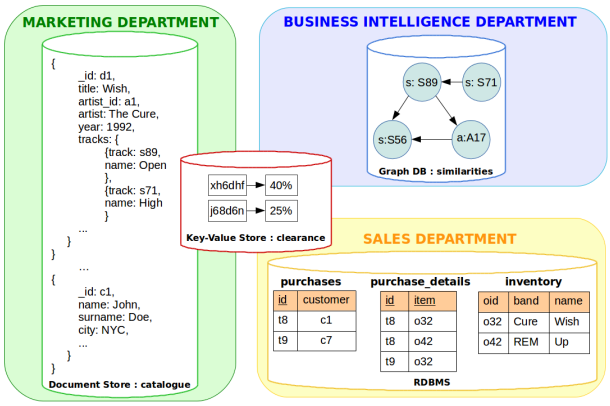

Figure 1: A polystore environment

In such scenario it is common that a user is only aware of a single database of the polystore but does not know anything about other databases (neither the content, nor the way to query them and, sometimes, not even their existence). This clearly poses new challenges for accessing and integrating data in an effective way. To recall a relevant discussion about polystores, the issue is that: *"if I knew what query to ask, I would ask it, but I don't"* [8].

QUEPA is a tool that shows a possible contribution to this problem by introducing *augmented search* and *augmented exploration*, two new methods to access a polystore based on the automatic enrichment of data extracted from a database with data that belong to the rest of the polystore.

Augmented search consists in the expansion of the result of a query over a local database with data that are relevant for the query but are stored elsewhere in the ploystore. For example, if Lucy, an employee of the sales department who only knows SQL, needs all the information available in ACME on the album "Wish", she submits the following query in *augmented* mode:

```
SELECT * FROM inventory
WHERE name like '%Wish%'
```


SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA

ISBN 978-1-4503-3531-7/16/06...$15.00

DOI: http://dx.doi.org/10.1145/2882903.2899393

Taking advantage from a repository of metadata which stores the relationships between objects in different datastores, QUEPA returns the following augmented tuple revealing details about the product that are not in the database of the sales department, including the fact that such item is currently on clearance sale with a discount of 40%.

```
< o32, Cure, Wish >   ⇒   (clearance:  40%)
                ⇓
        (catalogue:  {title:  Wish,
                      artist_id:  a1,
                      artist:  The Cure,
                      year:  1992,
                      tracks:  { ...}
                     }
        )
```

Augmented exploration exploits the same infrastructure and provides a more interactive and flexible way to access data, which consists in a guided expansion of the result of a query over a local database with related data stored elsewhere in the polystore. For example, if Lucy submits the same query above in *exploratory* mode, QUEPA returns the following tuple, in which the links suggest that further information, related to the returned tuple, is available elsewhere and allows Lucy to decide which component of the query result to augment.

< o32, Cure, Wish >

This process is iterative and provides a method of database exploration [4], where the user can freely find her way through the polystore, similarly to what happens when we surf the Web. QUEPA is also able to suggest an initial result to start with, based on the most frequent items accessed in the past.

The majority of solutions to the problem of accessing data in polystores introduce a middleware approach, where a unified language, a common system interface or a universal data model is provided to address data heterogeneity [1, 2, 3]. These approaches introduce a generalization that not only adds computational overhead at runtime, but also hides the specificity and functionality these systems were adopted for [8]. Indeed, a middleware approach tends to vanish the expected benefits of a polystore and introduces an intrinsic complexity that increases when new database systems take part to the polystore. Conversely, QUEPA is a lightweight tool that does not add abstraction layers over a polystore, thus, it has a minimal impact on the applications running on top of the data sources. Moreover, the augmentation technique of QUEPA provides a soft mechanism for data integration in polystores that complements other approaches based on cross-db joins [5], it keeps data in the original format, allows the use of the original query languages or APIs and avoids any query translation. Finally, QUEPA is compatible with the most popular categories of modern database systems and it is easily extensible to work with additional systems.

Although it can be easily embedded on existing platforms that perform complex data operations over a polystore (e.g., [5]), we have implemented QUEPA as a stand-alone tool that exposes a set of primitives and connects to several database systems. We demonstrate this stand-alone version accessible through a web user interface.

In brief, the demonstration of QUEPA aims at showing the audience:

- the novel methods of augmented search and augmented exploration for accessing data scattered among heterogeneous database systems in a polystore;
- the advantages of the "Bring Your Own Language" approach for accessing a polystore, which allows the user to submit queries without the need to learn a new query language;
- the value of guaranteeing the exploitation of the complete functionality of each DBMS, which is the main motivation for adopting a polystore approach;
- the simplicity of operating, in this framework, in a plug-and-play mode, without ad-hoc configurations or additional code.

2. SYSTEM OVERVIEW

We briefly show our approach by explaining how each component of QUEPA works. As shown in Figure 2, the logical architecture of QUEPA relies on three main components: (a) an *augmenter*, which augments the result of local queries, (b) a link *repository*, which keeps track of meaningful relationships among data objects belonging to different stores, and (c) a link *collector*, which gathers such relationships. Each component is analyzed in more detail in the following of this section.

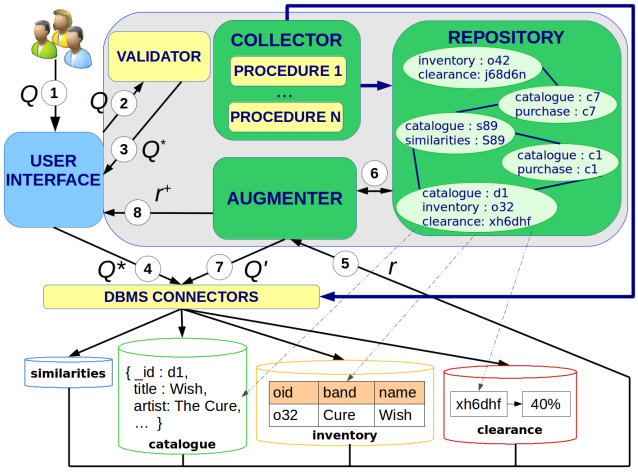

Figure 2: Architecture of QUEPA.

QUEPA exposes a set of primitives for a programmatic use of the augmentation construct. However, we have developed a light user interface that allows to use QUEPA via a web browser. Figure 3 and Figure 4 show two pages of this interface. The former allows users to submit an (augmented) query, specifying the target database in the polystore, and to visualize the augmented result composed of several data objects. The latter allows users to explore the polystore, by augmenting a single answer at a time.

Repository. QUEPA is based on a collection of relationships among related data objects in the polystore. Each database has one or many data collections (e.g., tables in a relational database), which are identified in the polystore

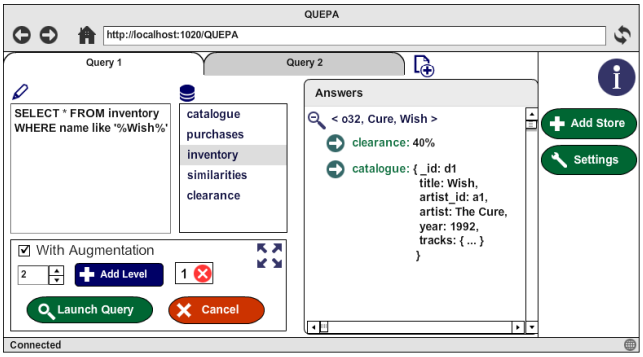

Figure 3: Interface for augmented search.

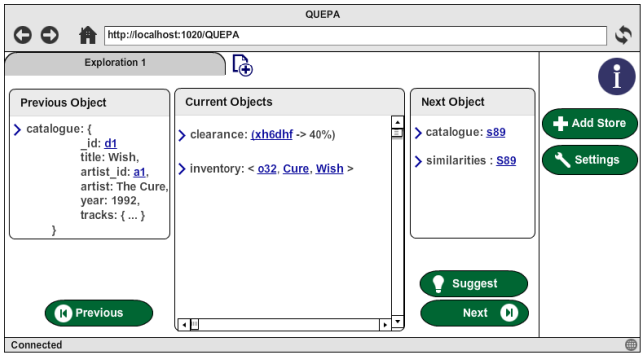

Figure 4: Interface for augmented exploration.

with a unique name. A data object is an atomic value set that can be uniquely identified inside a data collection. As a concrete example, tuples and JSON documents are data objects in relational databases and document stores, respectively. Each data object is represented in QUEPA with a key-value pair, where the key is the name of the data collection and the value is the (unique) identifier of the object, used in that data collection. A relationship indicates that two or more objects are related to each other. Relationships among objects capture either an identity or a reference. It turns out that the approach behind QUEPA is compatible with every database system whose data objects can be uniquely identified with a name and a key.

The set of all relationships in QUEPA forms an undirected graph. The nodes of the graph represent relationships that aim to capture identities in the polystore and are called *level 0* relationships. All key-value pairs that have been found to refer to the same conceptual object are stored in the same node. For example, in Figure 2 we have five *level 0* relationships, one of these is defined among three data objects: the tuple with key "o32" in the inventory, the document of the catalogue with id "d1" and the pair with key equal to "xh6dhf" in the clearances.

References among objects are represented by connecting with an edge the corresponding nodes of the graph. In this case we call them *level 1* relationships. Generalizing, the level of a relationship between two data objects is the length of the shortest path between the corresponding nodes in the graph. For example, the object with id "c1" in the *catalogue* is involved in a *level 3* relationship with the object "j68d6n" in the *clearance* database. Hence, the level of a relationship can be used to measure the distance between the result of a regular query and the data objects retrieved by the augmentation. For this reason, the augmentation is done with respect to a set of levels, as we will clarify below.

In the graph of relationships each data object of the polystore is represented in at most one *level 0* relationship (i.e. a key-value pair is unique over the entire graph). The graph of relationships is currently stored in the graph database management system Neo4j, which allows to easily manage a high number of relationships, but we are investigating different solutions for its implementation.

Collector. A fundamental task in QUEPA is the identification of the meaningful relationships among data objects in the polystore. This is a long standing problem, which is made even harder by the heterogeneity present in a polystore. Existing techniques usually require a *schema alignment* phase, where attributes of different schema are matched [7], followed by a *record linkage* phase (also known as entity resolution or duplicate detection), which serves to identify those objects that represent the same entity. Unfortunately, in a polystore, we cannot fully rely on state-of-the-art approaches because of the lack of necessary information (some of the stores are schema-less, others may not have any metadata associated to themselves) and of the different data models of the various systems involved. Therefore, we have opted for a combination of different techniques in the literature, with the goal of overcoming the limitations of each in our framework. The relationships are found transparently to the user and are progressively stored offline in the link repository by enforcing the constraints mentioned above. This aspect is, however, outside the scope of this demonstration.

Augmenter. When a user submits a query Q in augmented mode (step ① in Figure 2), QUEPA first checks, via a *validator* (step ②), whether the query involves only aggregate functions. In this case Q is computed without augmentation. In addition, the validator can rewrite Q into Q^* (step ③) by just adding all the identifiers of the data objects involved in the query that are not explicitly mentioned in Q, before its execution over the target database (step ④).

The result r of Q^* is then processed by the *augmenter* that computes the actual augmentation according to the level of relationship with the data objects in r (step ⑤). The default is 0 (that is, an augmented search returns, for each data object o in r, all the data objects in a *level 0* relationship with o) but, as also showed in Figure 3, the level of augmentation can be fixed by the user. In general, the *augmenter* is in charge of identifying all the relationships of the given level in the repository (step ⑥) and retrieving all the involved data objects by means of suitable queries to the corresponding databases in the polystore (step ⑦). This component has to account non-trivial algorithms for minimizing the number of accesses to the polystore needed to generate the current augmentation. Finally, the augmented answer r^+ is returned to the user (step ⑧).

In case of augmented exploration, the *augmenter* is in charge of retrieving and caching, in each step of the interaction with the user, all the data objects having either a *level 0* or a *level 1* relationship with the data objects in the current result (here the level of augmentation is transparent to the user). This information is suitably used by the user interface to build the output and to generate the next result according to the request of the user.

3. DEMO OUTLINE

In our demonstration we simulate the scenario of the running example in Figure 1. Specifically, we use a polystore for the ACME company that includes a MongoDB database for the marketing department, a MySQL database for the sales department, a Neo4j database supporting the business intelligence department and a shared Redis database. We have chosen these database systems because they are widely used within their respective category but, as previously mentioned, other database systems can be added to our framework with limited effort.

We have used the Last.fm dataset[1], included in The Million Song Dataset, to populate the *catalogue* and the *similarities* databases with 943.347 songs and 6.639.415 similarities , respectively. The MusicBrainz web services have been used to populate both the catalogue and the inventory with roughly 100.000 albums each. For the purpose of the demonstration, the content of these two databases does not overlap. In the generation of the albums, we have found many ambiguities due to different releases, special editions, etc. These ambiguities have been stored only in one of the two databases storing the albums. We have also used synthetic-data generators for populating the databases with user's profiles, products on clearance and purchases. We have generated in this way 20.000 user profiles, 40.000 products on clearance, 50.000 purchases of about 200.000 single albums sold.

We propose to the audience the following sequence of use cases that allow a comprehensive vision of the augmented construct and of the main functionality of QUEPA.

Scenario A: Plug-and-play. In this scenario we show how QUEPA can be used with a very simple configuration, which only requires to set the IP address of the underlying DBMSs. Audience can experience in this way the ability of QUEPA to easily collect a bunch of relationships, thus allowing to solve augmented queries without further human intervention. We then show more advanced configurations of the system that allow the user to gather more relationships and obtain a richer augmentation.

Scenario B: Bring Your Own Language (BYOL). This demo scenario focuses on query augmentation. A participant to the demonstration can choose the DBMS she/he is more expert in, among those available. As a result, QUEPA shows the structure of the chosen database so that she/he can submit meaningful queries. It is then possible to compare the results of the query with and without augmentation. Finally, we show the different results that can be obtained by varying the level of augmentation.

Scenario C: Augmented exploration. In this scenario, we ask the participant to submit a query in exploratory mode and after getting the results, we ask him/her to freely click on the links made available by the system to retrieve further information on that component of the result. This interaction can proceed as long as new data is found. During this exploration the system keeps track of the history of the navigation, so that the participant to the demo can go back to the previously visited data objects (see Figure 4).

We also demonstrate the ability of QUEPA to suggest the most promising links to follow according to the number of data objects that can be reached from them. Suggestions are also available without submitting a query, so that users who are totally unaware of the content of the polystore or do not know any of the local query languages can still interact with the system [4, 6].

In this scenario the level of augmentation is not visible to the user and the exploration proceeds, in each step, through the *level 0* and *level 1* relationships of the data objects in the current result.

Scenario D: Promotions of relationships. This last scenario serves to show one of the methods for collecting relationships. This procedure is currently the most involved among those implemented, and we believe that, since it requires human interaction, it can be more appealing to the audience. It relies on a simple, yet effective, adaptive learning mechanism, based on crowdsourcing, that allows the promotion of a *level n* relationship in the link repository to a *level m* relationship (with $m < n$).

More specifically, when a user interaction involve an augmentation, either in a search or in an exploration, she/he is following a path in the graph of the relationships. QUEPA keeps track of the most visited paths by the crowd. When a path is visited more times than a threshold (which we set to a low value for the demonstration purposes), the system connects the sink with the source of that path. As an example, in Figure 5 we have four nodes that are, supposedly, visited in sequence many times until reaching the given threshold. It can be concluded that this path of the graph is meaningful for the users and for this reason the dashed line is added to the link repository.

Figure 5: Promotion of relationships.

It follows that, in QUEPA, the graph of relationships evolves over time according to the behaviour of the users and so does the augmented result of the queries. This feature is demonstrated by showing the audience how the graph of the relationships evolves as soon as queries are performed and how these changes affect the augmentation process of QUEPA.

[1] http://labrosa.ee.columbia.edu/millionsong/lastfm

4. REFERENCES

[1] Apache MetaModel. http://metamodel.apache.org/, (accessed January, 2016).

[2] UnQL: Unstructured Data Query Language. http://www.couchbase.com/press-releases/unql-query-language, (accessed January, 2016).

[3] P. Atzeni, F. Bugiotti, and L. Rossi. Uniform access to nosql systems. *Inf. Syst.*, 43:117–133, 2014.

[4] M. Buoncristiano et al. Database challenges for exploratory computing. *SIGMOD Record*, 44(2):17–22, 2015.

[5] J. Duggan et al. The BigDAWG polystore system. *SIGMOD Record*, 44(2):11–16, 2015.

[6] K. Morton, M. Balazinska, D. Grossman, and J. D. Mackinlay. Support the data enthusiast: Challenges for next-generation data-analysis systems. *PVLDB*, 7(6):453–456, 2014.

[7] R. Pottinger and P. A. Bernstein. Schema merging and mapping creation for relational sources. In *EDBT*, pages 73–84, 2008.

[8] M. Stonebraker. The case for polystores. http://wp.sigmod.org/?p=1629, July, 2015.