Thesis Proposal:

Exploiting Recurring Retrieval Needs in Querying Heterogeneous and Dynamic Graph Dataspaces

Francesco De Fino

Abstract

Querying data sources containing heterogeneous and dynamic information is a complex task: high quality answers need to be produced fast to cope with the dynamicity of the environment. A common solution to reduce query processing time is to rely on approximate processing approaches, which provide a faster answer at the cost of a lower accuracy. On the other hand, user involvement in the interpretation of the request [30], is not adequate in dynamic contexts, where urgent requests hamper user interaction.

In order to cope with the difficulties raised by the heterogeneity and dynamic nature of the considered environments, in [18] it has been claimed that new solutions should be devised, relying either on additional knowledge about the current execution (e.g., query context or user profile) or previous executions of similar requests.

The goal of the thesis is to exploit information related to requests recurring over time for efficiently and effectively process, in an approximate way, complex requests on heterogeneous and dynamic data spaces, possibly available on a highly distributed environment, while guaranteeing user satisfaction and limiting as much as possible user interactions. The idea is to take advantage of prior processing in order to obtain shortcuts to different points in the query processing stack. The approach we envision generalizes smart caching methods allowing various kinds of information to be associated with cached queries in order to improve query processing even further. More precisely, the thesis will first propose a general framework for recurring queries representation and management, as well as two specific instantiations, related to two particular application contexts, namely source selection for linked data and mapping selection for pay-as-you-go data integration. The role of request contexts and data quality in developing such solutions will also be investigated.

1 Motivation and goals

The last few years have been characterized by the growth of highly heterogeneous and dynamic data sources, shared across the network, ranging from knowledge bases and struc-

tured HTML information to unstructured contents from social networks and microblogs. These sources usually contain heterogeneous, strongly correlated and often highly dynamic data and can be represented according to graph-based data models [5], which model the reference domain using the notions of nodes (entities) connected by links (relationships). Notable examples of application domains where data is naturally represented in graph-based form are, besides knowledge bases, Web-scattered data, social networks, biological and chemical databases, healthcare data, personal information management (PIM) and enterprise information management (EIM) systems, just to mention a few.

Heterogeneous and dynamic datasets available on the network are often exploited much below their potential due to difficulties in accessing them. Indeed, users can specify their request only vaguely because the format and the structure of the data encoding the relevant information is unknown. As an example, consider a smart city explorer, that is, a set of information published by a municipality to users that may retrieve, e.g., information about attractions, points of interests, and shops. In this specific context, data may come from different, heterogeneous and very dynamic datasets. At different instants of time, also user position or the environment itself (including data) may be changed.

Processing complex requests on diverse and dynamic sources requires: (i) request interpretation; (ii) source selection; (iii) actual processing of the request on the relevant, and usually big in size, sources. Besides being costly, the overall process may not guarantee the user is satisfied by the obtained result due to various problems: the request could be incorrectly interpreted or processed on inaccurate, incomplete, unreliable data; additionally, processing time could be inadequate to the query urgency. A common solution to reduce query processing time is to rely on approximate processing approaches, which provide a faster answer at the cost of a lower accuracy. On the other hand, user involvement in the interpretation of the request [30], is not adequate in dynamic contexts, where urgent requests hamper user interaction. In [18], the authors claimed that, to cope with the difficulties raised by the heterogeneity and dynamic nature of the considered environments, user profiles and request contexts, data and processing quality, and similar requests recurring over time could be exploited.

In this thesis, we focus on *similar requests recurring over time* for improving efficiency and effectiveness of the approximate processing of requests, assumed to be already interpreted and represented according to a graph-based formalism [17]. Recurring retrieval needs have been considered in query processing for a very long time in terms of *materialized views* [21, 22]. The idea is to precompute the results of some recurring queries as materialized views, select some of such views (view selection problem) and re-use them (view-based query processing) for the execution of new requests. Unfortunately, the usage of materialized views in the contexts described above suffers from some problems: (i) views associate a result with a given query, however in the reference contexts other or additional information could be of interest (e.g., the set of used data sources or, under pay-as-you-go integration approaches [6], query-to-data mappings); (ii) view updates are very frequent in dynamic environments, reducing the efficiency of the overall system; (iii) view-based query processing techniques usually rely on a precise semantics while heterogeneity tends to favour approximation-based approaches. Recurring retrieval needs are also at the basis of *smart caching* approaches. Here the main issue concerns the definition of suitable cache replacement policies. Usually the most of the proposed approaches rely on precise query matching [44, 35] and similarly to materialized views, cached queries are usually associated with query results.

The goal of the thesis is to investigate how information about similar requests recurring over time can be exploited in order to efficiently and effectively process complex requests on

diverse and dynamic data spaces, possibly available on a highly distributed environment, while guaranteeing user satisfaction and limiting as much as possible user interactions, getting over the limitations of current materialized views and smart caching approaches. The idea is to take advantage of prior processing in order to obtain *shortcuts* to different points in the query processing stack. The approach we provide generalizes smart caching methods allowing various kinds of information to be associated with cached queries in order to improve query processing even further.

More precisely, the thesis will first propose a general framework for recurring queries representation and management, as well as two specific instantiations, related to two particular application contexts, namely source selection for linked data and mapping selection for pay-as-you-go data integration.

2 Reference research area and relevance of the thesis goal

The current proposal lies in the reference area "Query processing in advanced and unconventional architectures", with the aim of exploiting processing information related to recurring queries, executed in heterogeneous and dynamic environments, to make request processing more efficient and effective.

As pointed out before, recurring queries are at the basis of well-known mechanism like smart caches and materialized views. The approach we propose differ from other existing solutions for the following aspects:

- Existing approaches mainly rely on a precise semantics to perform matches between the query to be executed and queries executed in the past while heterogeneity tends to favour approximation-based approaches. For this reason, the approach we plan to develop will exploit approximation to improve query processing performance as well as user satisfaction.

- In modern processing environments, the traditional query processing stack has been enriched with new phases. All the existing approaches exploiting recurring queries associate past executed queries with the computed result, thus linking a query with the bottom layer of the query processing stack. On the other hand, the approach we envision will exploit shortcuts to different points of the stack, leading to a more flexible and adaptive solution for unconventional query processing scenarios.

- Additional information like request context and data quality can be taken into account to deal with heterogeneity and dynamism of the considered datasets. These two aspects have been already analyzed in literature but they have not been explored yet in the settings we consider. This information can contribute to improve both the query processing time and the quality of the answers.

3 State of the art

In this section, we present related work that we consider relevant for the proposed research project. In order to facilitate reading, the discussion has been organized into six main sections, corresponding to the main concepts introduced in Section 1 and 2. The first one discusses graph matching approaches, the second introduces some existing caching methods, the third analyzes techniques based on views, the fourth explores the source selection problem for linked data, the fifth briefly discusses the mapping selection problem, and the sixth provides information on context and quality estimates.

3.1 Graph data and graph matching

Graph databases emerged due to the needs of very complex applications handling highly related data, as, for example, social network data, biological and chemical databases, healthcare data, just to cite a few.

A node-labeled, directed graph is defined as $G = (V, E, L)$, where (i) V is a set of nodes; (ii) $E \subseteq V \times V$ is the set of edges, in which (v, v') denotes an edge from node v to v'; (iii) for each $v \in V$, $L(v)$ is the label of v. The label $L(v)$ may indicate a variety of real life semantics, a list of attributes, or even a set of predicates.

Our research proposal relies on a graph-based representation of data spaces and user retrieval needs [46]. In particular, for the sake of simplicity and for guaranteeing a tractable combined and data complexity (fundamental requirements for dealing with massive in size graph databases), we assume the data space is represented in terms of a set of *graph data sources*, i.e., graphs where nodes can also be labeled with variables, for representing situations in which the node identity is not known [6], due to the heterogeneous nature of data and the possible lack of information. Furthermore, we assume a graph query is specified as a directed, labeled graph with variables belonging to the set of *Unions of Acyclic Conjunctive Queries* (UACQ) [6].

In most applications, one may want to find a correspondence between two objects represented by graph structures. This problem is knows as *graph matching problem*. Examples of such applications are: web page classification and web mirror detection, object identification, schema matching, plagiarism and spam detection, social matching, web service composition, etc.

The graph matching problem can be stated as follows: given two graphs G_1 and G_2, a metric for measuring the similarity of the two graphs, the problem is to find a (partial) mapping from the nodes (edges) of G_1 to the nodes (edges) of G_2, such that the mapping satisfies the metric.

In general the problem can be divided in two different sub-problems: *exact* and *approximate* graph matching.

Exact graph matching. Given two graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_1)$, with $|V_1| = |V_2|$, the exact matching problem is to find a one-to-one mapping $f : V_1 \to V_2$ such that $(u, v) \in E_1$ iff $(f(u), f(v)) \in E_2$ and $\forall u \in V_1$, $L_1(u) = L_2(f(u))$. When such a bijective function f exists, this is called an *isomorphism*, and G_1 is said to be isomorphic to G_2.

As a class of graph matching problem, *pattern matching* is to find for a given (small) *pattern graph* all the matches in a (large) *data graph*, based on a matching metric.

Since a bijective function is often too restrictive to identify matching, the graph matching problem is also addressed in terms of *graph simulation*.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, G_1 matches G_2 via *graph simulation*, denoted by $G_2 \prec G_1$, if there exists a total match relation M, i.e., for each $u \in V_1$, there exists $v \in V_2$ such that $(u, v) \in M$.

Both graph isomorphism and graph simulation problems are known to be NP-complete. For this reason various heuristics have been proposed to solve this problem efficiently [41].

Existing solutions differs on the algorithms used for computing the isomorphism: most of them rely on indexing techniques which are based on graph features selection in order to reduce the matching cost [45, 20, 19]. For example, in [45] are proposed two indexes, one for the dataset and one for the query, for sub or supergraph query processing. In [20] the index called FG*-index consisting of three components.

Approximate Graph Matching. The approximate graph matching problem can be defined as follows: given two graphs G_1, G_2 and a metric for measuring the *similarity* of the two graphs, the problem is to find a (partial) mapping from the vertices (edges) of G_1 to the vertices (edges) of G_2 s.t. the mapping satisfies the metric. If such mapping exists, G_2 is a match of G_1 [48].

Various approximated approaches have been proposed, which differ mainly on the method chosen and on the algorithm used for the problem resolution.

The approximated graph matching problem is mainly dealt with in terms of subgraph isomorphism. In the approximate problem, differently from the precise matching, is also computed a similarity value between the two graphs.

The approximate graph matching problem is also NP-complete, for this reason some heuristics have been proposed, among which vertex-similarity graph matching [], feature-based graph matching [] and edge-to-path graph matching [].

The algorithms proposed for the existing solution are mainly based on two techniques: *indexing methods* and *neighborhood similarity*. In [51] both approaches are used together to create a candidate set used by the heuristic solution to the subgraph isomorphism problem. In [40] a neighborhood index is created to identify the important nodes in the graph. In [1, 31] vertex neighborhood is used to compute the similarity between the graphs.

3.2 Views

Recurring retrieval needs have been considered in query processing for a very long time in terms of *materialized views* [21, 22].

A view V can be defined as a query associated with a name. A view is materialized if the query result is pre-computed and stored. We refer to the query result $V(G)$ of a view V upon a dataset G as the *view-related extension* for V in G or simply as a *view* [23].

Two main problems can be identified: the *view selection problem* and the *view-based query processing*.

The *view selection problem* can be stated as follows: given a query Q, given a set of views $\mathcal{V} = \{V_1, ..., V_n\}$, the problem is to find another query A such that: (i) A only refers to views $V_i \in \mathcal{V}$ and their extensions $\mathcal{V}(G) = \{V_1(G), ..., V_n(G)\}$ in G, and A is equivalent to Q i.e., for all data graphs G, $A(\mathcal{V}(G)) = Q(G)$, where $A(\mathcal{V}(G))$ denotes the result of A over $\mathcal{V}(G)$. If such a query A exists, the query Q can be answered using views $\mathcal{V}$ [21].

As regards the *view-based query processing*, the problem is to compute the answer to a query by relying solely on a set of selected views.

For the *view-based query processing* some algorithms have been proposed, based mainly on precise techniques for the match between the query and the views.

For example, in the approach proposed in [21], a subset of views is selected according to the request (view selection problem) and all the views selected are merging together following a mapping between them and the query (view-based query processing). This process ends when a fixpoint is reached.

3.3 Caching

Recurring retrieval needs are at the basis of caching approaches. Existing caching methods [35, 44, 27] rely on the exploitation of precomputed results of previously executed queries to be used for the execution of new requests in order to improve query processing performance.

The main issues of caching approaches are:

1. how to update the cache content;

2. how to use the cache during query processing.

In order to address the first problem, two decisions have to be taken: (i) which information should be stored together with queries inside the cache; (ii) how the cached items should be replaced in the cache.

Typically, the existing approaches populate the cache by adding the executed query together with its computed results. As a consequence of the limited size of the cache, replacement policies need to be defined in order to avoid cache pollution [45].

The second issue, i.e. the selection of cached results during query processing, can be stated as follows: given a query, select the cached item that best matches it. This is clearly a matching problem, so it depends on the matching approach chosen. In case of exact matching, only cached queries that exactly match the request are selected. In the approximate scenario, the choice is done by relying on a similarity measure between the request and the cached queries. Most of the proposed approaches rely on precise query matching [44, 35].

Caching policies depend on the considered data. In our research we will focus on graph-based data. In this context, some caching approaches have been proposed, focusing mainly on RDF and SPARQL queries [35, 27], but also methods for generic graph queries [44] and linked open data [25, 47, 29] have been proposed.

The smart caching framework proposed in [35] provides an example of caching applied to SPARQL queries. This approach addresses the caching query selection problem using a precise matching. A Cache Controller process monitors all the requests issued by the queries, enabling the detection of cross-query profitable subgraphs and triggering their execution and caching. The whole framework is based on a Dynamic Programming Planner which issues cache requests for all query subgraphs of a given query by relying on a canonical label associated with the SPARQL query. The canonical label is generated using a revised approach based on a tool called Bliss [26].

3.4 Source selection

A growing amount of Linked Data − graph-structured data accessible at sources distributed across the Web − enables advanced data integration and decision-making applications.

The publication of Linked Data on the WWW is based on the following four principles, which have become known as the "Linked Data principles" [10]: (i) use URIs as names for things; (ii) use HTTP URIs so that people can look up those names; (iii) when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL); (iv) include links to other URIs, so that they can discover more things.

Several general options for querying the Web of Data exist [24]. In the simplest case, an application may access the SPARQL endpoint provided by a particular data publisher. However, usually, we are interested in approaches for executing queries over the virtual union of Linked Data from multiple providers, relying on either data warehousing or federated query processing. In the first cases, linked data sources are integrated in a centralized repository: in this way, it is possible to provide almost instant query results but query results may not reflect the most recent data and users can only benefit from the portion of the Web of Data that has been copied into the repository. On the other hand, federated query processing approaches distribute query execution over the SPARQL endpoints that publishers provide for their Linked Data sets. The problem in this case is that we cannot assume that each publisher provides a SPARQL endpoint for its Linked Data.

An alternative approach for Linked Data query execution relies on an online execution of queries for which the query execution system relies only on the Linked Data principles.

This means that, in order to get Linked Data potentially relevant for answering a query, URIs are looked up during the query execution process itself, thus providing a *live query processing* approach, quite relevant in all the contexts in which freshness and discovery of results is more important than the generation of the answer in a very short time.

One of the main problems for live exploration approaches is *source-selection*. The goal is to identify the data sources that possibly provide results for the given query or, in other words, to eliminate sources from the query plan that do not contribute to the result.

Source selection is a crucial step that affects the efficiency of the execution of queries and three main groups of solutions have been proposed [24]:

1. *Live Exploration*: as described above, this approach makes use of the characteristics of the Web of Data, in particular, the existence of data links. Live-exploration-based systems perform a recursive URI lookup process during which they incrementally discover further URIs that can be scheduled for lookup.

2. *Index-Based Approaches*: this method relies on a pre-populated index which is used for identifying URIs to look up during query execution time and uses data structures that index URIs as pointers to data. Source selection using such an index is based on *relevance*: A URI is relevant for a given query if the data retrieved by looking up the URI contributes to the query result.

3. *Hybrid Approaches*: this approach combines an index-based approach with a live exploration approach and, thus, aim to achieve the advantages of both approaches without inheriting their respective shortcomings [24].

In our research we focus on an index-based method proposed in [42] which relies on *data summaries* for determining relevant sources during query evaluation. Data summaries have been proposed to efficiently determine which sources may contribute answers to a query in live distributed query systems. They approximately describe the data provided by a source in an aggregated form, in much more detail than schema-level indexes. The QTree, specifically, is a data summary over Linked Data sources, seeing the data items (RDF triples) in the sources as points of a three dimensional numerical space, by applying hash functions to the triple components. Like the R-tree, a QTree is a tree structure consisting of nodes defined by minimal bounding boxes (MBBs). An MBB de- scribes the multidimensional region in the data space that is represented by the node the subtree underneath. Leaf nodes in a QTree, however, rather than containing the data items that are contained in their MBBs as in R-trees, are buckets containing statistical information (e.g., count) that approximate the data items contained in their MBBs.

3.5 Data orchestration and mapping selection

Usually in data management scenarios, users have to deal with collection of heterogenous data sources, of which they have limited understanding of both structure and semantics. A fundamental aspect in querying such data spaces is the issue of orchestrating the sources for querying, i.e. aligning and exchanging data between sources. Many solutions have been proposed towards a common goal which is query answering in the overall set of data sources, by relying on *mappings* for representing both associations of similar entities in different datasets and associations of similar entities between the query to be executed and the source datasets.

The management of mappings for the orchestration of data sources has been taken into account mainly from a data-centric perspective, with the aim of discovery stable data mappings or one-time, pay-as-you-go data mappings [9]. In line with the dataspace philosophy of providing the benefits of classical data integration while reducing upfront costs, pay-as-you-go integration and curation approaches [38, 50] elaborate data incrementally, also in response to specific user feedback [8], while indexes can dynamically self-organize their structure on the basis of the workload [4]. Flexible entity-based data models [38] are usually adopted to represent various forms of heterogeneous knowledge and indexes over this kind of data can rely on the expressive power of the adopted query languages at instance level [36]. More recent paradigms for schema mapping specification use data examples for an interactive design and refinement of mappings [3, 37]. Queries can be specified without exactly knowing the schema and structure of the data by indicating allowed approximations [28, 49] and also by means of examples [15, 34].

Nowadays, in very dynamic scenarios, like those taken into account in this proposal, such an approach does not work and is in general replaced by an iterative and dynamic approach where new query results inform and guide further analysis, in an open-ended interaction between mapping discovery and data orchestration []. In this context, information about mapping computed in prior query executions could be quite useful to improve efficiency and effectiveness of the processing of the query at hand.

3.6 Context dependency and quality estimates

Processing complex requests on diverse and dynamic sources, besides being costly, may not guarantee the user is satisfied by the obtained result due to various problems: the request could be incorrectly interpreted or processed on inaccurate, incomplete, unreliable data; additionally, processing time could be inadequate to the query urgency.

In order to cope with the difficulties raised by the heterogeneity and dynamic nature of the considered environments, user profiles and request contexts, as well as data and processing quality, can be exploited.

Context dependency. Contexts are the surroundings, circumstances, environment, background or settings that determine, specify, or clarify the meaning of an event or other occurrence. In a query processing environment, context information can be associated, as metadata, with both user requests and data collections. Usually, context metadata capture information about [2]: *(i) who*: information about the user submitting the request (e.g., a user profile); *(ii) what*: the type of the requested or described information; *(iii) where*: spatial location and type of environment; *(iv) when*: temporal location; *(v) why*: request or data motivations.

Context information can be either *static* or *dynamic*. Static context information remains stable along time. By considering a context associated with a user request, metadata related to *who*, *what*, and *why* are typically static metadata. On the other hand, metadata related to *where* and *when*, i.e., location and temporal context properties, often constitute dynamic context information, since their value may dynamically change along time. Dynamic context information make the context acquisition process more complex since the need arises of capturing context information as fast as possible before the context changes.

More generally, a context can be represented in terms of various metadata (*dimensions*), each representing information at a given level of detail (*granularity*). Several context dimensions have been proposed by previous studies [14], among them we recall time, space, absolute/relative space and time, context history, subject and user profile. The overall set of dimensions to be used for context representation and the considered granularity levels

give rise to a *context model*, based on which contexts, to be associated with user requests and data, can be represented in a hierarchical way. Context dimensions and granularity facilitate data exploration and searching since they provide a flexible mechanism for comparing contexts associated with different entities, based on the context hierarchy. In dynamic context, where data sources are heterogeneous and dynamic, three main coordinates have been identified as relevant [18]: (a) user profile and request context, (b) data and processing quality and (c) similar requests repeated over time.

Quality estimates. Data quality characterizes diverse data sources. It is highly variable and can be represented by metadata describing or allowing to derive data accuracy, completeness, consistency, and update rate [7, 6]. Quality is very subjective and finding a canonic, universal criterion is simply impossible [32]. Thus, different dimensions and assessment techniques should be considered to capture the different quality aspects. Completeness, consistency, accuracy, update frequency, copy-paste issues [32, 11] are just few examples of quality dimensions that can be taken into account. Additionally, data may have different levels of credibility. Unreliability of the data source as well as the lack of knowledge of the data publisher on the subject may result in data with poor credibility.

Quality indicators assess the adequacy of data items and data sources for a specific kind of goal. Relevance of quality indicators depends on the application domain and on the quality dimensions to be assessed. Bizer et Al. [12, 11] classify assessment metrics into three categories, according to the type of information exploited to obtain quality indicators: *(i) content-based metrics*, which use the content itself, e.g., the content subject; *(ii) context-based metrics*, which rely on the conditions in which data was created/accessed/last modified; *(iii) rating-based metrics*, which rely on explicit ratings or feedbacks about data itself, data sources, or data providers. Implicit metadata can be extracted from data also, as in [13]. Provenance metadata are quite relevant for computing context-based metrics. They provide information about the lineage or history of how data objects are generated or transformed [39]. From the quality point of view, it is crucial to know the creator of a piece of data, the creation time, and to track changes. Thus, provenance allows the evaluation of freshness and credibility of sources and content. A survey of provenance information usage in databases and in the Semantic Web can be found in [33].

4 Objectives, expected results and methodology

4.1 Objectives and expected results

The overall goal of this research proposal is the definition of an innovative solution for finding (approximate) answers to recurring and complex information needs, operating on the full spectrum of relevant content in a data space of highly heterogeneous, dynamic and poorly controlled sources. We organize the work into four main objectives, each leading to specific results. The aim of objective 1 (reference model) is to analyze the state of the art approaches on graph matching, caching techniques and view selection and to define a general framework for representing and managing recurring queries together with information about their previous execution. Objective 2 (source selection for linked data) is aimed at analyzing state of the art approaches on linked data and live query processing and at providing a first instantiation of the proposed framework. The aim of objective 3 (mapping selection in pay-as-you-go environments) is to analyze the state of the art approaches mapping selection in data integration contexts and to provide a second instantiation of the proposed framework. Finally, the objective 4 aims at analyzing

additional information, like requests context and data quality, for improving further the query processing.

More precisely, the objectives of the proposal are the following:

Objective 1: reference models.

1.1 *Analysis of the state of the art approaches*, related to (i) the graph matching problem, both for exact and approximate methods; (ii) caching techniques, for graph-based data and (iii) views and view selection approaches for graph data.

1.2 *Definition of a framework for recurring queries execution in dynamic and diverse dataspaces*, exploiting information associated with previously executed requests for improving efficiency and effectivity of query processing in such complex environments. The goal is to define a general framework for representing and managing recurring queries. The idea is to take advantage of prior processing in order to obtain shortcuts to different points of the query processing stack (not only to query results, but also to the result of previous query processing stack). We will also identify the key elements of the framework which can lead to significant instantiations, with a special reference to:
(i) the considered data and query model (e.g., generic graphs or linked data graphs);
(ii) the step of the query processing stack the shortcut refers to (e.g., final result computation, source selection, mapping selection);
(iii) the architectural level at which previously generated processing information can be stored and managed (e.g., the physical level, as for caches, or the external one, as for views).

Objective 2: source selection for linked data.

2.1 *Analysis of the state of the art concerning linked data and live query processing*, with the aim of investigating in details the source selection problem in live query processing for Linked Data, in order to consider this context as the first instantiation of our framework. We also plan to select as suitable source selection approach to be used as a baseline in the experimental evaluation of our first framework instantiation. At the time being, we plan to consider the Data Summary method proposed in [43].

2.2 *Instantiation of the proposed framework on linked data*, addressing the source selection problem for recurring query execution. The considered instantiation will consider: (i) data expressed as linked data graphs and SPARQL queries; (ii) shortcut to the source selection step; (iii) physical architectural level, exploiting a cache mechanism for instantiating the framework. We plan to design a specific caching architecture for the problem at hand, as well as specific algorithms for cache maintenance and cache-based query processing. We also plan to implement the proposed approach and experimentally evaluate in according to some baseline approaches (like [43]).

Objective 3: mapping selection in pay-as-you-go environments.

3.1 *Analysis of the state of the art concerning mapping selection in data integration contexts*, focusing on how source-to-source mappings and query-to-source mappings are typically processed in pay-as-you-go approaches in order to consider this context as the second instantiation of our framework. We plan to select a suitable mapping selection approach to be used as a baseline in the experimental evaluation of our second framework instantiation.

3.2 *Instantiation of the proposed framework on graph data and data integration*, with the aim of addressing the issue of mapping selection under pay-as-you-go integration environments. Specifically, referring to the three elements underlined in 1.2, we consider: (i) query expressed as graphs; (ii) shortcut to the data integration step; (iii) physical architectural level, exploiting a cache mechanism for instantiating the framework. We plan to design a specific caching architecture for the problem at hand, as well as specific algorithms for cache maintenance and cache-based query processing. We also plan to implement the proposed approach and experimentally evaluate in according to some baseline approaches.

Objective 4: query context and result quality.

Analysis of the role of context and quality information in the developed solutions, in order to take into account result quality of past query executions as well as contexts and profiles associated with data sources and user requests to improve user satisfaction. This problem rise from the fact that the request could be incorrectly interpreted or processed on inaccurate, incomplete, unreliable data; additionally, processing time could be inadequate to the query urgency. Exploiting these information in dynamical environment will further improve the query processing. The outcome of this step will be an analysis of how techniques we propose can be extended, in order to take into account query context and data quality as well some preliminary solutions to this problem.

4.2 Methodology

In the following, we describe the methodology we envision for achieving each project objective described in Subsection 4.1.

Objective 1: reference models. The analysis of the state of the art will be carried out by classified methods (related to graph matching, caching and views) with respect to precise and approximate approaches. Among approximate approaches, we will analyze in detail the used similarity measure. Indexing approaches for graph matching will also carefully revised. For what concern the framework definition, we plan to investigate three main problems:

1. given a request as input, find an equivalent query which can be executed relying on a set of recurrent queries.

2. determine how to execute this query relying on information associated with recurring queries.

3. how to update the set of recurring queries adding a new query considered recurrent or deleting a query that is no longer recurrent.

Objective 2: source selection for linked data. We will analyze caching approaches for graphs and we will focus on one baseline method. We will revise it in order to take into account information about sources used for recurring queries past execution. Measures used for selecting the best item in the cache will be also revised. In case of the selected caching approach is based on precise matching, we will modify in order to allow approximate graph matching. The method proposed will be implemented and experimented, comparing the results with the chosen baseline approach for the source selection problem [43].

Objective 3: mapping selection in pay-as-you-go environments. We will analyze the best architectural level at which store the mappings. We plan to analyze the existing mapping selection approaches and choose the best method to be the baseline solution for our evaluation. We will develop a second instantiation of our framework that will consider shortcut to the data integration step instead of the source selection, as in the first implementation. The method proposed will be implemented and experimented, comparing the results with the chosen baseline approach for the mapping selection problem.

Objective 4: query context and result quality. We want to further improve the query processing taking into account query context and result quality information. We plan to include these information in our framework associating them to recurring queries in order to experimentally evaluate the benefit of using such information, revising the proposed implementations.

5 Research plan

We aim at organizing our research in the three years according to eight activities, listed in Table 1. Each activity corresponds to a thesis objective or to the thesis writing. In particular, Table 1, for each activity, points out each sub-objective and the time units dedicated to it, while Table 2 presents their schedule in the three years.

Table 1: Workplan

Activity	Sub-objective	Months	Color
1	Analysis of the state of the art on graph matching, caching and views	12	
2	Definition of a framework for recurring query execution in dynamic and diverse dataspaces	6	
3	Analysis of the state of the art on linked data and live query processing	4	
4	Instantiation of the framework on linked data and source selection	10	
5	Analysis of the state of the art on mapping selection in data integration	4	
6	Instantiation of the framework on graph data and data integration	10	
7	Analysis of the role on context and quality information in the developed solutions	8	
8	PhD thesis writing	6	

Table 2: Schedule

Activity	Year 1	Year 2	Year 3
1			
2			
3			
4			
5			
6			
7			
8			

6 Preliminary results

In this section we will present the activities done and the results achieved in the first year.

1. *Analysis of the state of the art on graph matching, caching and views*: we have analyzed some of the most promising graph matching approaches. Since requests are often incomplete, it is hard to find precise matches between two queries. So we focused on some approximate graph matching approaches. We provide a table which summarizes the methods analyzed since now. The table is shown in Figure 1.

2. *Definition of a framework for recurring queries execution in dynamic and diverse dataspaces*: we proposed a general framework for representing and managing recurring queries in terms of *Profiled Graph Query Pattern*. Each PGQP corresponds to a graph, associated with data or metadata related to the past evaluation of the queries it represents. For each PGQP is also stored an accuracy value which quantifies the accuracy of associating the query with the information.

   The PGQP-based framework is shown in Figure 2 and it is based on three main phases:

   - **Phase 1: PGQP-based query decomposition**. Given a query Q and a set of PGQP, we want to determine whether it is possible to rely on the PGQPs for executing Q. This is possible if Q can be approximated by a new query, obtained by composing a set of Q subgraphs, which best match PGQPs, with the portion of Q which cannot be represented in terms of the PGQPs, called *residual query*.
   - **Phase 2: PGQP-based query processing**. Each subgraph of the query Q matched with a PGQP can be executed relying on information associated with the PGQPs selected. To this aim, specific PGQP-based processing algorithms should be designed, depending on the information associated with PGQPs. The evaluation of each subgraph of Q returns in general an approximate result, for which an accuracy value should be computed. This value is influenced mainly by the similarity between Q and the each selected PGQP, the usage of precise or approximate query processing algorithm and the result cardinality. All the partial results collected by the execution of each subgraph of Q, included the residual query which will be executed based on a traditional graph query processing algorithm, will be merged together using a fusion operator.

Ref	Type	Problem	Algorithm	Dataset Type	Similarity Function	Title	Authors	Year
ZZSJHQ16	Approximate	Subgraph Isomorphism	Indexing	Large Graph	Nodes (two score functions)	Approximate Subgraph Matching Query over Large Graph	Zhao, Y., Zhang, C., Sun, T., Ji, Y., Hu, Z and Qiu, X.	2016
KWAY13	Approximate	Subgraph Isomorphism	Minimum matching cost	Large Graph	Nodes	NeMa: Fast graph search with label similarity.	Khan, A., Wu, Y., Aggarwal, C. C., and Yan, X.	2013
TP08	Approximate	Subgraph Isomorphism	Indexing	Large Graph	Nodes	Tale: A tool for approximate large graph matching.	Tian, Y., and Patel, J. M.	2008
YZYH06	Approximate	Subgraph Isomorphism	Indexing	Large Graph	Edges	Feature-based similarity search in graph structures.	Yan, X., Zhu, F., Yu, P. S., and Han, J.	2006
MGMR02	Approximate		Fixpoint computation	Depends on the input	Nodes	Similarity flooding: A versatile graph matching algorithm and its application to schema matching.	Melnik, S., Garcia-Molina, H. and Rahm, E.	2002
ZLGZ09	Approximate/ Exact	Subgraph Isomorphism	Indexing	Multiple Graphs organized into one with GPTree structure		A novel approach for efficient supergraph query processing on graph databases.	Zhang, S., Li, J., Gao, H., and Zou, Z.	2009
FWW16	Exact	Simulation	Materialized Views	Large Graph		Answering Pattern Queries Using Views	Fan, W., Wang, X. and Wu, Y.	2016
WNT2016	Exact	Subgraph Isomorphism	Indexing	Large Graph		Indexing Query Graphs to Speedup Graph Query Processing	Wang, J., Ntarmos, N. and Triantafillou, P.	2016
FWW13	Exact	Subgraph Isomorphism, Simulation	Incremental	Large Graph		Incremental graph pattern matching.	Fan, W., Wang, X., and Wu, Y.	2013
SWWSL12	Exact	Subgraph Isomorphism	Parallel Computing	Large Graph		Efficient subgraph matching on billion node graphs.	Sun, Z., Wang, H., Wang, H., Shao, B., and Li, J.	2012
MCHW12	Exact	Simulation		Large Graph fragmented		Distributed graph pattern matching.	Ma, S., Cao, Y., Huai, J., and Wo, T.	2012
CYDPW08	Exact	Reachability conditions	Join based	Large Graph		Fast graph pattern matching.	Cheng, J., Yu, J. X., Ding, B., Philip, S. Y., and Wang, H.	2008
CKN08	Exact	Subgraph Isomorphism	Indexing	Large Graph		Efficient Query Processing on Graph Databases	Cheng, J., Ke, Y. and Ng, W.	2008
CYYHZG07	Exact	Subgraph Isomorphism	Indexing	Large Graph		Towards Graph Containment Search and Indexing	Chen, C., Yan, X., Yu, P. S., Han, J., Zhang, D.Q. and Gu, X.	2007
CGK05	Exact	Subgraph Isomorphism	DAGs	Large Graph		Stack-based algorithms for pattern matching on dags.	Chen, L., Gupta, A., and Kurul, M. E.	2005
CFSV04	Exact	Subgraph Isomorphism	Feasibility Rules	Large Graph		A (sub) graph isomorphism algorithm for matching large graphs.	Cordella, L. P., Foggia, P., Sansone, C. and Vento, M.	2004
BS98	Exact	Subgraph Isomorphism	Graph distance	Large Graph		A graph distance metric based on the maximal common subgraph.	Bunke, H., and Shearer., K.	1998

Figure 1: Report table on graph matching approaches

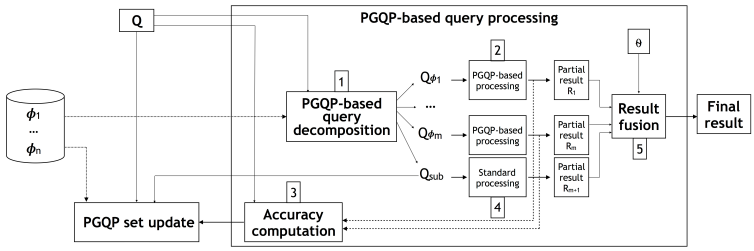

Figure 2: PGQP-based framework

- **Phase 3: PGQP set update**. Since the PGQP-based query processing of a query produces various information that need to be reflected into the PGQP set to keep it up-to-date. More precisely, we envision two main groups of updates, namely online updates, executed as side effects of query processing which produces accuracy values for each PGQP selected, and offline updates, which periodically delete or refresh PGQPs which are considered unreliable for three main reasons: (i) the PGQP is no longer recurrent; (ii) the accuracy value of the PGQP is under a threshold; (iii) the associated information may become imprecise due to the variability of the data space.

The results of this work have been already published in [17, 16].

3. *Analysis of the state of the art on linked data and live query processing*: we have analyzed existing approaches on source selection for live query processing. We have selected as baseline approach the Data Summaries method for linked data [43] since it is proved to be the best approach for selecting sources based on indexes.

4. *Instantiation of the framework on linked data and source selection*: we are currently working on a first instantiation of the framework. Starting from the baseline smart caching approach [35], we have exploited the caching application for SPARQL queries they provide.

   We are currently working on revising the cache tree proposed in [35], exploiting a new measure to select the best cache entry. This measure is derived from the baseline source caching approach [43] and it identifies the number of triple patterns of a certain source that contribute to the query result. As a consequence, we are working on the implementation proposed by [35], revising the code in order to adapt the application for our purpose. In this way, the cache will contain recurring queries associated with the (ranked) list of sources used for their previous execution. We want to test our software in various scenarios, for example when the cache is empty or when the cache already contains some entries, and compare it with the implementation of the Data Summaries approach provided by the authors [43].

References

[1] C. C. Aggarwal A. Khan Y. Wu and X. Yan. "Nema: Fast graph search with label similarity". In: *Proceedings of the VLDB Endowment*. Vol. 6. 3. VLDB Endowment. 2013, pp. 181–192.

[2] G. D. Abowd and E. D. Mynatt. "Charting past, present, and future research in ubiquitous computing". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 7.1 (2000), pp. 29–58.

[3] Bogdan Alexe, Balder Ten Cate, Phokion G Kolaitis, and Wang-Chiew Tan. "Designing and refining schema mappings via data examples". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM. 2011, pp. 133–144.

[4] Gunes Aluc, M Tamer Ozsu, Khuzaima Daudjee, and Olaf Hartig. "Executing queries over schemaless RDF databases". In: *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE. 2015, pp. 807–818.

[5] R. Angles and C. Gutierrez. "Survey of graph database models". In: *ACM Computing Surveys (CSUR)* 40.1 (2008), p. 1.

[6] P. Barceló Baeza. "Querying graph databases". In: *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*. Ed. by R. Hull and W. Fan. ACM, 2013, pp. 175–188. DOI: 10.1145/2463664.2465216.

[7] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. "Methodologies for data quality assessment and improvement". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 16.

[8] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, and C. Hedeler. "Incrementally improving dataspaces based on user feedback". In: *Information Systems* 38.5 (2013), pp. 656–687.

[9] S. Bergamaschi, D. Beneventano, F. Mandreoli, R. Martoglia, F. Guerra, M. Orsini, L. Po, M. Vincini, G. Simonini, S. Zhu, L. Gagliardelli, and L. Magnotta. "From Data Integration to Big Data Integration". In: *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*. 2018, pp. 43–59. DOI: 10.1007/978-3-319-61893-7_3. URL: https://doi.org/10.1007/978-3-319-61893-7_3.

[10] T. Berners-Lee. "Linked data-design issues". In: *http://www. w3. org/DesignIssues/LinkedData. html* (2006).

[11] C. Bizer and R. Cyganiak. "Quality-driven information filtering using the WIQA policy framework". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7.1 (2009), pp. 1–10.

[12] C. Bizer and R. Oldakowski. "Using context-and content-based trust policies on the semantic web". In: *WWW*. ACM. 2004, pp. 228–229.

[13] C. Böhm, F. Naumann, Z. Abedjan, D. Fenz, T. Grütze, D. Hefenbrock, M. Pohl, and D. Sonnabend. "Profiling linked open data with ProLOD". In: *ICDE Workshops*. IEEE Computer Society, 2010, pp. 175–178.

[14] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. "A data-oriented survey of context models". In: *ACM Sigmod Record* 36.4 (2007), pp. 19–26.

[15] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. "Learning path queries on graph databases". In: *18th International Conference on Extending Database Technology (EDBT)*. 2014.

[16] B. Catania, F. De Fino, and G. Guerrini. "Exploiting Recurrent Retrieval Needs in Querying Heterogeneous and Dynamic Graph Dataspaces." In: *SEBD*. 2017.

[17] B. Catania, F. De Fino, and G. Guerrini. "Recurring Retrieval Needs in Diverse and Dynamic Dataspaces: Issues and Reference Framework." In: *EDBT/ICDT Workshops*. 2017.

[18] B. Catania, G. Guerrini, A. Belussi, F. Mandreoli, R. Martoglia, and W. Penzo. "Wearable queries: adapting common retrieval needs to data and users". In: *Proceedings of the 7th International Workshop on Ranking in Databases*. ACM. 2013, p. 7.

[19] C. Chen, X. Yan, P. S. Yu, J. Han, D.-Q. Zhang, and X. Gu. "Towards graph containment search and indexing". In: *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment. 2007, pp. 926–937.

[20] J. Cheng, Y. Ke, and W. Ng. "Efficient query processing on graph databases". In: *ACM Transactions on Database Systems (TODS)* 34.1 (2009), p. 2.

[21] W. Fan, X. Wang, and Y. Wu. "Answering graph pattern queries using views". In: *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE. 2014, pp. 184–195.

[22] F. Goasdoué, K. Karanasos, J. Leblay, and I. Manolescu. "View selection in semantic web databases". In: *Proceedings of the VLDB Endowment* 5.2 (2011), pp. 97–108.

[23] A. Y. Halevy. "Theory of answering queries using views". In: *ACM SIGMOD Record* 29.4 (2000), pp. 40–47.

[24] O. Hartig. "An overview on execution strategies for Linked Data queries". In: *Datenbank-Spektrum* 13.2 (2013), pp. 89–99.

[25] O. Hartig. "How Caching Improves Efficiency and Result Completeness for Querying Linked Data." In: *LDOW*. 2011.

[26] T. Junttila and P. Kaski. "Engineering an efficient canonical labeling tool for large and sparse graphs". In: *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2007, pp. 135–149.

[27] J. Lorey and F. Naumann. "Caching and prefetching strategies for sparql queries". In: *Extended Semantic Web Conference*. Springer. 2013, pp. 46–65.

[28] Federica Mandreoli, Riccardo Martoglia, and Wilma Penzo. "Approximating expressive queries on graph-modeled data: The GeX approach". In: *Journal of Systems and Software* 109 (2015), pp. 106–123.

[29] M. Martin, J. Unbehauen, and S. Auer. "Improving the performance of semantic web applications with SPARQL query caching". In: *The Semantic Web: Research and Applications* (2010), pp. 304–318.

[30] Y. Mass, M. Ramanath, Y. Sagiv, and G. Weikum. "IQ: The Case for Iterative Querying for Knowledge." In: *CIDR*. 2011, pp. 38–44.

[31] S. Melnik, H. Garcia-Molina, and E. Rahm. "Similarity flooding: A versatile graph matching algorithm and its application to schema matching". In: *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE. 2002, pp. 117–128.

[32] P. N. Mendes, H. Mühleisen, and C. Bizer. "Sieve: Linked data quality assessment and fusion". In: *EDBT/ICDT Workshops*. Ed. by Divesh Srivastava and Ismail Ari. ACM, 2012, pp. 116–123.

[33] L. Moreau. "The foundations for provenance on the web". In: *Foundations and Trends in Web Science* 2.2-3 (2010), pp. 99–241.

[34] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. "Exemplar queries: Give me an example of what you need". In: *Proceedings of the VLDB Endowment* 7.5 (2014), pp. 365–376.

[35] N. Papailiou, D. Tsoumakos, P. Karras, and N. Koziris. "Graph-aware, workload-adaptive sparql query caching". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, pp. 1777–1792.

[36] F. Picalausa, G. H. L. Fletcher, J. Hidders, and S. Vansummeren. "Principles of guarded structural indexing". In: *stud* 4 (2014), p. 5.

[37] Carlos R Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo. "MostoDEx: A tool to exchange RDF data using exchange samples". In: *Journal of Systems and Software* 100 (2015), pp. 67–79.

[38] A. Das Sarma, X. Dong, and A. Halevy. "Bootstrapping pay-as-you-go data integration systems". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM. 2008, pp. 861–874.

[39] Y. L. Simmhan, B. Plale, and D. Gannon. "A survey of data provenance techniques". In: *Computer Science Department, Indiana University, Bloomington IN* 47405 (2005).

[40] Y. Tian and J. M. Patel. "Tale: A tool for approximate large graph matching". In: *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE. 2008, pp. 963–972.

[41] J. R. Ullmann. "An algorithm for subgraph isomorphism". In: *Journal of the ACM (JACM)* 23.1 (1976), pp. 31–42.

[42] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres. "Comparing data summaries for processing live queries over linked data". In: *World Wide Web* 14.5-6 (2011), pp. 495–544.

[43] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres. "Comparing Data Summaries for Processing Live Queries over Linked Data". In: *World Wide Web* 14.5-6 (2011), pp. 495–544.

[44] J. Wang, N. Ntarmos, and P. Triantafillou. "GraphCache: a caching system for graph queries". In: (2017).

[45] J. Wang, N. Ntarmos, and P. Triantafillou. "Indexing Query Graphs to Speed Up Graph Query Processing". In: (2016).

[46] G. Weikum. "Data and knowledge discovery". In: *GRDI 2020* (2011).

[47] G. T. Williams and J. Weaver. "Enabling fine-grained HTTP caching of SPARQL query results". In: *International Semantic Web Conference*. Springer. 2011, pp. 762–777.

[48] Y. Wu. "Extending graph homomorphism and simulation for real life graph matching". In: (2011).

[49] S. Yang, Y. Wu, H. Sun, and X. Yan. "Schemaless and structureless graph querying". In: *Proceedings of the VLDB Endowment* 7.7 (2014), pp. 565–576.

[50] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. "Lenses: An on-demand approach to etl". In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1578–1589.

[51] Y. Zhao, C. Zhang, T. Sun, Y. Ji, Z. Hu, and X. Qiu. "Approximate Subgraph Matching Query over Large Graph". In: *International Conference on Big Data Computing and Communication6s*. Springer. 2016, pp. 247–256.