Knowledge Graph Exploration Systems: are we lost?

Matteo Lissandrini
matteo@cs.aau.dk
Aalborg University
Denmark

Davide Mottin
davide@cs.au.dk
Aarhus University
Denmark

Katja Hose
khose@cs.aau.dk
Aalborg University
Denmark

Torben Bach Pedersen
tbp@cs.aau.dk
Aalborg University
Denmark

ABSTRACT

Knowledge graphs (KGs) represent facts in the form of nodes and relationships and are widely used to represent and share knowledge in many different domains. However, their widespread adoption to integrate different data sources and their generation processes have made KGs very complicated and difficult to understand, leading to the advent of new *knowledge graph exploration* approaches to better understand their contents and extract relevant insights. Nevertheless, the needs of current KG exploration use cases are not met (even neglected) by existing KG data management systems. Hence, the question: are we lost? We hope not. Therefore, with the aim of fostering research on these open issues, in this position paper, we first present an overview of state-of-the-art approaches for KG exploration. Then, we identify the (currently unmet) requirements for effective KG exploration systems, and finally, we highlight promising research directions for the realization of a system able to fully support knowledge graph exploration.

1 INTRODUCTION

A multitude of diverse companies including Amazon, Bosch, Google, Microsoft, and Zalando is using the graph data model to represent and store their enterprise knowledge bases [51, 61, 65]. Moreover, an increasing amount of data is published as RDF datasets and made available as Linked Open Data in different scientific domains [22, 31]. We also see widespread adoption of resources like YAGO, DBpedia, and WikiData describing entities and facts of general encyclopedic interest, e.g., artists, books, movies, and songs. These networks of rich connections among entities are called *knowledge graphs* (KGs – see Figure 1 for an example). KGs store highly heterogeneous information and are increasingly adopted to advance more intelligent machine learning systems [6, 24, 71].

The heterogeneity of KGs constitutes both a defining characteristic and a challenge in their effective utilization [48]. As a byproduct of their widespread adoption, KGs include data from many different domains and easily become large and complex, and thus their contents become hard to grasp [25, 51, 65]. Usually, they are generated (semi-)automatically through the integration of many different data sources [31, 54], each with its own original data model, and they are updated with unprecedented volumes of data, and at unprecedented speed [61]. Thus, the contents of these KGs have become less and less familiar even to domain experts and almost impenetrable to first-time users, calling for exploratory methods for knowledge graphs [39, 50].

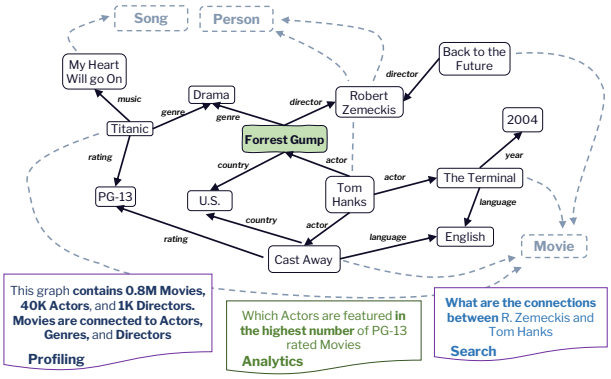

Figure 1: Simplified fragment of a knowledge graph around the entity "Forrest Gump" and three exploratory tasks.

In this context, *knowledge graph exploration* [42] is the machine-assisted and progressive process of analysis of a KG with the goal of (1) understanding the structure and nature of the graph, (2) identifying which portion of the given KG and its data (modelled through relationships and attributes) can satisfy the current (often vague and hard-to-express) information need or research question, and (3) extracting from it insights that can assist in the formulation of novel research questions and hypotheses. These goals are achieved through three main tasks (exemplified in the boxes in Figure 1 and detailed in Figure 2): (i) summarization and profiling, (ii) exploratory data analytics, and (iii) exploratory search.

The connections between the aforementioned goals and the supported tasks are not simply one-to-one (see also Figure 3 below) since some goals can be achieved by the combination of different tasks, e.g., to extract relevant data and insights, one can first inspect some specific entities of interest for relevant attributes and connections and then analyze how frequent or prevalent those are in the rest of the graph. Therefore, a KG exploration system should support all these three tasks interchangeably and at the same time.

In recent years, KG exploration in general, and the tasks of profiling, search, and analytics in particular, have received considerable attention. Thus, building upon our previous analysis [42], in this paper we first survey state-of-the-art approaches in this scientifically

 12th Annual Conference on Innovative Data Systems Research (CIDR '22). January 9-12, 2022, Chaminade, USA.

rich area (summarized in Figure 2). Then, we identify their main characteristics and the kind of insight they allow to obtain in connection with the requirements they impose on KG data management systems (DBMS). Our first finding is that the task of exploratory KG analytics has received less attention despite its importance. We observe that KG exploration in general and exploratory KG analytics in particular have special computational needs. Therefore, there is a need for novel and more efficient data management systems that are tailored for these tasks. Specifically, exploratory queries access large portions of the graph, they are generally under-specified, and they require the capabilities to provide approximate answers as well as to provide some initial answers very quickly, even if such answers are incomplete. Yet, existing systems are either designed for batch, non-interactive, processes, can support only limited ad-hoc computations, or conversely are designed for transactional queries that only return answers for highly selective queries. As a result of our analysis, in this work, we not only provide a view of exciting directions for the development of novel KG exploration techniques that will enable the exploration of highly heterogeneous knowledge graphs, but we also identify important topics for system-oriented research that can lead to the realization of efficient KG exploration systems to effectively support such techniques. Moreover, we advocate for more system-oriented research that should converge in the design and implementation of fully-fledged KG exploration systems that will support such methods in real-world day-to-day business deployments. This is of particular importance since existing experimental studies identify that state-of-the-art graph management systems fall short in supporting the needs of KG exploration, while relational DBMS cannot handle the heterogeneity of the KG model and the type of analytics they require.

In the following, we first identify the limitations of existing DBMSes in supporting the needs of KG data management (Section 2), and then provide a summarization of the area of KG exploration (Section 3). Moreover, we analyze in detail the challenges that KG exploration systems need to meet (Section 4). As a result, we identify the core operations that should be supported by systems for KG exploration (Section 5), and we conclude with a look towards future directions (Section 6).

2 PRELIMINARIES & PITFALLS

A *knowledge graph* $\mathcal{G}$ is a tuple $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ consisting of a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$, and a set of triples $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. A knowledge graph forms a network structure where entities, e.g., Forrest Gump and Tom Hanks, are connected through relationships, e.g., *actor*, to form triples or facts, e.g., (Tom Hanks, *actor*, Forrest Gump). For simplicity, in the example, we treat literal values and attributes, such as 65 in (Tom Hanks, *age*, 65) as entities. Nonetheless, there are of course important nuances involved in the precise data model adopted [35], especially but not only, when dealing with exploratory analytics and numerical or statistical values. Overall, in a KG, the information stored within the graph, i.e., the data values, are as important as the connections among them, i.e., the structure and its semantics.

At a high level, we can characterize an exploratory task as a tuple $E{=}(t, P, C)$, where $t{\in}\{\textsc{profile}, \textsc{summary}, \textsc{explore}, \textsc{analyse}, \ldots\}$ is a task type, P is a set of parameters for t, and C is a set of conditions for t. An exploratory query returns an exploratory answer $\mathbb{A}$, which consists of either triples (directly from the knowledge graphs or generated based on the data in the KG), metadata, or statistics. For instance, the profiling task in Figure 1 is the triple $E{=}(\textsc{profile}, \{\mathcal{E}, \mathcal{R}\}, \emptyset)$ that, when executed over the graph $\mathcal{G}$, returns as answer $\mathbb{A}{=}E(\mathcal{G})$, a set of statistics extracted for both sets $\mathcal{E}$ and $\mathcal{R}$. Furthermore, we talk of an exploration workload as a sequence of exploration tasks $W{=}[E_1, E_2, ..., E_k]$, where the parameters and conditions of a given task E_i are informed by the results of some subset of previous tasks $\{E_j{\in}W\ .\ j{<}i\}$. Thus, a *knowledge exploration system* is a data management system for a knowledge graph $\mathcal{G}$ that efficiently returns an answer $\mathbb{A}_i$ for each exploratory task $E_i{=}(t, P, C)$ in a given workload W and that can exploit information gained while executing the sequence of tasks in W to optimize its functions (in terms of both performance and precision).

Despite the growing need for exploratory search and analysis tools and the availability of numerous KGs, we have identified a substantial scarcity of both systems providing effective support for such analyses as well as benchmarks that can help researchers identify more clearly the limitations of those systems. The main candidates to provide support for KG exploration tasks are Relational Data Management Systems (RDBMSes) and Graph Data Management systems (GDBMSes). In the following, we revisit the pitfalls of the available data management systems when dealing with KGs.

RDBMSes. One straightforward solution for storing and querying graphs in general is to adopt traditional RDBMSes. An RDBMS offers competitive advantages on graphs with *regular and small schemas* and on *simple queries* [10, 15, 45, 52]. Data exploration in RDBMSes has as long history [28, 39] with tasks ranging from profiling [2] to progressive analytics [5, 17]. The data exploration techniques for RDBMSes are tailored to the relational model and in particular assume that the data has a predefined schema. For instance, it is not obvious how functional dependencies used in data cleaning for relational data generalize to knowledge graphs [16]. Similarly, navigating unbounded and complex paths requires recursive queries, which, while supported by RDBMSes, are harder to express and not as effective when dealing with a multitude of relation types [37]. On the other hand, specialized Graph DBMSes have recently become more common. As a result, several studies investigate the performance of Graph DBMSes in the transactional case [37, 45, 63] and some studies suggest that there is no need for specialized GDBMSes [15, 52]. Yet, such a conclusion originates from analyses of KGs with only a few relationships and for a very limited number of workloads. In particular, these analyses do not consider highly heterogeneous KGs or KG exploration use cases. As the heterogeneity of the KG increases and the queries become more complex, RDMBSes quickly face scalability issues [37].

GDBMSes. Compared to RDBMSes, existing GDBMSes provide higher flexibility with heterogeneous graphs. However, GDBMSes are optimized for point-wise queries, e.g., retrieving the immediate neighbors of a node having a specific edge label [37]. Additionally, the lack of benchmarks for KG exploration hinders the capacity of researchers and system developers to correctly identify current limitations. Nonetheless, recent experimental studies have already highlighted that also existing GDBMSes face scalability challenges

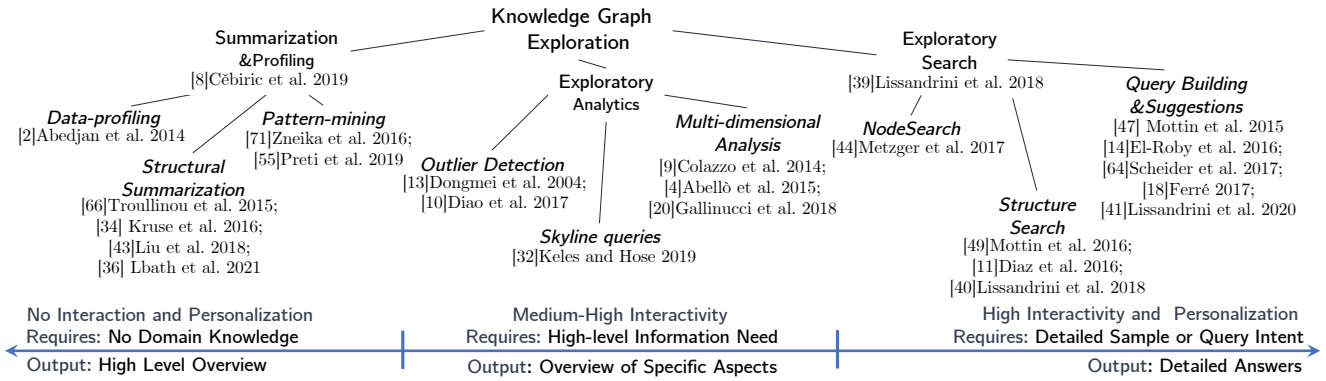

Figure 2: Taxonomy of KG Exploration techniques and their positioning on the spectrum of features.

on complex pattern or path queries [37, 45, 61]. Therefore, GDBMSes are not optimized for dealing with a mix of pointwise and analytical queries, which is instead the usual composition of exploratory workloads. As such, current systems (both relational and graph-based) fall short in supporting KG exploration and the challenges identified in Section 4.

3 KG EXPLORATION METHODS

As mentioned earlier, data exploration approaches cover three domains (summarized in Figure 2): (1) KG *profiling and summarization* to distil the most important characteristics both of the structure and the contents of a KG; (2) *exploratory search* for a gradual discovery and retrieval of the items that are pertinent to a vague or underspecified information need; and (3) *exploratory analytics* to distil salient features from different subsets of the graph. We organize these methods over a spectrum describing the required expertise in domain knowledge, the level of interactivity they expect, and the type of output they are able to produce. Methods that do not require any domain knowledge usually provide high-level information with coarse granularity and a low level of detail. Also, they are typically one-off approaches that do not account for user preferences. On the other hand, methods that require some domain knowledge, e.g., some representative elements of interest or some initial definition of the query intent, are able to produce more fine-grained answers with a high level of detail. Moreover, they exploit interaction and user feedback to adapt to diverse user needs. Below, we first present summarization and exploratory search as the extremes of the spectrum. We then describe the special role of exploratory analytics as a middle ground in this spectrum. Furthermore, the left-hand side of Figure 3 presents the connections between these three categories of approaches and the type of user goals they support.

Profiling & Summarization. *Data profiling* is the simplest form of exploration, as it computes basic statistics to determine descriptive metadata about a given dataset. For KGs, existing methods [3] perform tasks like counting the number classes (e.g., Movie, Song, or Person) and their instances or summarizing value distributions for specific attributes (e.g., averaging the release year). They also identify important descriptors of the structure of the graph, e.g., node degree distribution. Their focus is therefore on frequencies and statistical measures.

Structural summarization [8, 43] and *pattern mining* [55, 72] approaches have been applied to KGs to facilitate understanding the structure of the data as well as to obtain concise representations of the most salient features of their contents. In general, KG summaries either (i) present a compact representation of the main features of the original graph; or (ii) define a new graph derived from the original graph [8]. In the former case, the summary is directly useful for and interpretable by the end user. In the latter, the summary is used instead of the original graph to provide easier access to answers that are expensive or impractical to compute on the original graph [59]. The main graph summarization techniques extract the schema of the graph [8, 36], a meta-graph composed of high-level patterns and their most representative instances [66]; for example, *T. Hanks acted in Forrest Gump*, Figure 1 is a notable instantiation of the frequent high-level pattern *Person acted-in Movie*. Primarily, this is achieved by analyzing frequent substructures, i.e., via *pattern mining* [55], or based on node and edge types and their topology [34, 72], e.g., a *Person* can be *actor* or *director* in a *Movie*, but is never playing the role of a *Song*. For instance, we could summarize part of Figure 1 with *Person acted-in and/or directed Movie*.

Overall, *these approaches require no specific domain knowledge and they return a high-level overview of the graph.* Thus, they help in the initial exploratory stages since they can assist in evaluating whether a KG (or portion of it) matches the domain of interest, whether any data cleaning is required, and they can help in formulating new research questions. At the same time, *their computations are very expensive and require multiple passes over the entire graph.* Yet, they are usually conducted offline since their results are computed *only once* and possibly updated when the graph is updated.

Exploratory Search. Exploratory search has the goal of helping a user with a vague information need to retrieve specific portions of the graph that are, usually only in hindsight, recognized as relevant. Therefore, while KG summarization provides a high-level representation of the graph (e.g., a zoomed-out view), exploratory search instead delves into the data itself (e.g., zooming in to a subset of items of interest). Yet, in contrast to traditional search, where the desired result is well defined, exploratory search usually starts from a *tentative query* that hopefully leads to answers that are at least partially relevant and provide cues for the next queries, e.g.,

inspecting nodes connected to the node *Forrest Gump* to finally identify as relevant the path that leads to *Back to the Future*.

Hence, exploratory queries change the traditional semantics of the search input: *instead of strictly prescribing the conditions that the desired result set must satisfy, they provide a hint of what is relevant,* delegating to the system the task of identifying more precisely what could be of use. This shift in semantics has led to (i) a number of methods following the *search-by-example* paradigm [39] and (ii) methods and interfaces that help the users formalize their intent into a domain-specific query construct that is usually an expansion of the input [18, 41]. Both have the common goal to overcome one of the main challenges in enabling exploratory search: to avoid complicated declarative languages (e.g., SPARQL) and at the same time retain the flexibility and expressiveness of such languages.

Search-by-example methods [38, 39] receive as input a set of example members of the answer set (e.g., *Tom Hanks* and *Forrest Gump*). The search system then infers the entire answer set based on the given examples and any additional information provided by the underlying database [49] (e.g., other movies by *Tom Hanks*, or a list of *Drama* movies and their actors). This allows retrieving a set of entities similar to some given entities of interest [44], or complex structures matching some relevant structure known by the user [40, 49]. *Node and Entity search* allows for automatic completion of a set of seed entities (persons, organizations, places). *Example-based graph search* works similarly to node search but requires a full example (a subgraph or a tuple) to be provided as input. For instance, it is possible to support by-example reverse engineering of (SPARQL) queries from example tuples [11].

To further facilitate the user when formulating a query written in an unfamiliar language and over an unfamiliar data set, different studies have proposed *query suggestion and refinement* techniques [41, 47] and graphical user interfaces [14, 18, 64]. Yet, while by-example methods allow for rather vague information needs, query formulation interfaces are designed to help users with a clear information need in writing (relatively simple) queries about specific entities.

Exploratory search *is useful in different stages of the exploration* since it supports the user in identifying or further exploring entities, relationships, and structures of interest. They help in answering more fine-grained and specialized information needs but still take into account that the user is not familiar with the graph. For this reason, particular focus is given to approximate methods [40] and to query suggestion and query refinement techniques [41]. These methods *involve high interactivity and require fast response times*, while they usually access only small portions of the KG.

Exploratory Analytics. Exploratory analytics is an iterative, integrated process of data discovery and analytical querying on data that is not well known to the user, e.g., external data. The ability to support analytical workflows for rich KGs has recently received increased attention [4, 9, 26]. The idea is to provide functionalities typical of relational data warehouses, i.e., *multi-dimensional analysis* over knowledge graphs by describing multi-dimensional and statistical information within the KG model [20, 68]. This is also motivated by increasing interest from public and private organizations to represent business data in specialized knowledge graphs [65]. All these methods enable a similar approach: to obtain analytical insights on RDF graphs by means of "views" and aggregation operations. For instance, considering the example in Figure 1, we could materialize an aggregated view to count the number of *Movies* for every *Country* and *Genre*. Such views are themselves accessible as RDF graphs. Similarly, *skyline queries* identify entities that optimize a multi-criteria decision problem leading to a set of objects that are of interest to a user because of their *dominance* across multiple attributes [32], e.g., what are the most recent *Movies* with the highest number of *Actors* performing in it.

Finally, *outlier detection* approaches identify elements that are interesting because they are very different from the rest of the elements [13], e.g., *Actors* who have participated in an unusually high number of *Movies*. Some of the most advanced approaches, Dagger and Spade [10], inspect a triplestore and select different aggregation queries that can describe different entity types based on high-variance values, e.g., whether there is more variability in the number of movies per *Year* or *Genre* across *Countries*.

In conclusion, exploratory analytics is effective to enable users to identify high-level details w.r.t. facets of the data tailored to specific information needs. In contrast, data summarization approaches are agnostic of the user's information need and only provide a global overview of the data. On the other hand, exploratory search digs into specific data items (entities and relationships) but these searches return very large result sets instead of a more useful aggregate analysis identifying trends and common patterns. Hence, exploratory analytics techniques are a middle ground, where specific summarization methods are applied over the large results of an exploratory search. Yet, current approaches usually mimic the same operators proposed for relational data, providing no graph-centric analyses. Moreover, *in these approaches, the user is either required to be familiar with the (complex) query language, or the system is not able to accept any user input to customize the output.* Thus, analytical approaches for KGs are currently an area in need of more research.

4 THE CHALLENGES OF KG EXPLORATION

Knowledge graphs pose complex and novel challenges when it comes to satisfying typical data exploration needs. In particular, we identify 4 main challenges that graph management systems face when dealing with KG exploration, namely: heterogeneity, evolution, vagueness, and scale (listed on the right-hand side of Figure 3). Among these, heterogeneity, both in terms of data and schema, is the most particular and prominent challenge posed by KGs. Heterogeneity is usually linked with an *unstructured evolution* of schema and data, which is also derived from the need for integrating different sources with content- and vocabulary-mismatch. Moreover, exploratory use cases naturally have an inherent *vagueness* of user information needs, a problem that is also exacerbated by the mismatch between the user's mental model of the domain and the complex structure of the graph. Finally, KGs naturally aim at representing and linking data at the largest scale, e.g., all of the human knowledge. In the following, we elaborate on each of these challenges individually. These challenges exist in relation to any exploratory use case, i.e., in Figure 3, we can imagine an arrow departing from each goal and pointing to each one of the challenges. Yet, here we highlight those connections that have a more prominent impact across both tasks and operations. Moreover, we

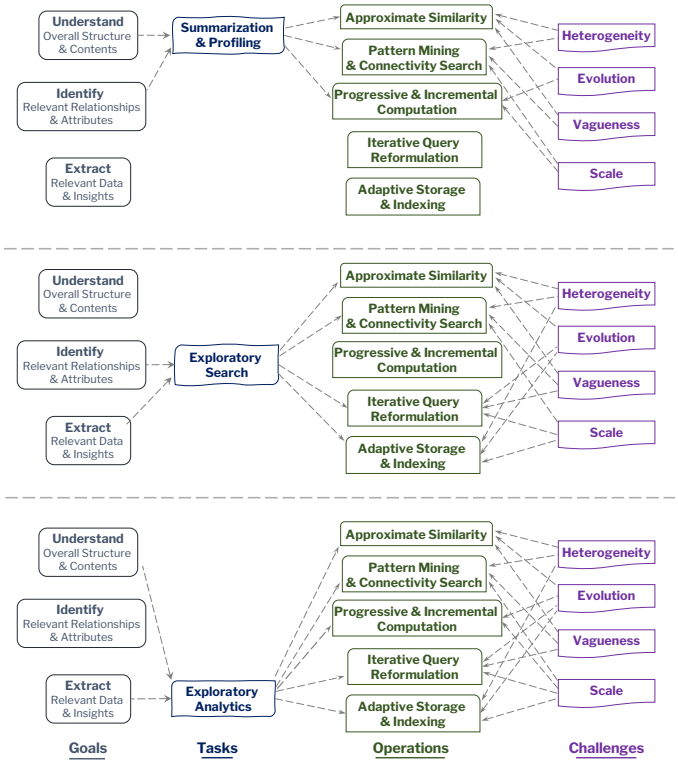

Figure 3: The connections between Tasks, Goals, Challenges, and Operations characterizing KG exploration systems.

will see later specific operations that should be supported by KG exploration systems to overcome them.

Heterogeneity. We distinguish between data and structural heterogeneity, where the latter refers to the fact that KGs have an inherent lack of a global schema. This is the price for their flexibility in handling new data and data being integrated from different sources. As such, a large number of KGs has no or very loose schemas [51]. While schema languages [56], such as RDF Schema (RDFS), ShEx, and SHACL, provide formalisms for schemas and ontologies, allowing easier data validation and cleaning, their rigidity hinders both the addition of new types and the integration of multiple sources that may represent the same type of information in different ways. Recent schema discovery methods try to automatically infer the schema of the graph either from patterns [36] or by asking expert users [23]. Nonetheless, these attempts do not assure full coverage or perfect compliance, always leaving the possibility that not all instances of the data are perfectly mapped by a given schema, or that information of the same type exists, represented by similar, but not identical structures. Therefore, a system supporting KG exploration should be flexible and forgiving enough when faced with these situations so that it can recognize and retrieve information that is modelled in a non-uniform way.

Evolution. Knowledge naturally evolves, and so should systems supporting KGs and KG exploration [53]. *Schema evolution* [58] was one of the most pressing challenges in the DB community. In KGs, the potential lack of schema makes this challenge even more prominent, with entities changing their meaning over time. This problem is known as *concept drift* [69] and has a strong relationship with the evolution of meaning in natural languages. This challenge has recently been identified as one of the most pressing challenges, as current systems and reasoning tools do not support the rapid evolution of knowledge [1]. A system for KG exploration cannot eschew from treating freshness of information as a first-class citizen and hence face the challenges of evolving knowledge head-on.

Vagueness. The lack of a schema and a continuous evolution of KGs make it almost impossible for the user to precisely and exhaustively formulate their information need. This is natural since the user does not know the exact schema, nor how the data is represented, resulting in queries that become vague and imprecise. Kersten et al. [33] argued in 2011 that *"next-generation database systems should interpret queries by their intent, rather than as a contract carved in stone for complete and correct answers."*. This is even more true when it comes to KG exploration systems. Existing methods for approximate graph queries [72] allow for imprecision in the queries and inconsistent data representations. Also, example-based search [39] allows for querying using results of an unknown query. However, these methods only partially address the issue that the user and the graph vocabulary are potentially different. Hence, approaches like active learning [46] and question answering [30] could form the basis of more flexible methods for exploratory queries for vague user needs. Yet, these approaches are far from being effectively integrated into systems that can also swiftly overcome the aforementioned challenges posed by heterogeneity and evolution.

Scale. The growing sizes of KGs alone already challenge the capabilities of existing approaches and systems in general [61, 62] and therefore complicate supporting heterogeneity, evolution, and vagueness even further. The need to integrate entities from several domains and representations renders also the scale of integrating data into KGs an even harder task. Finally, the size and richness of their structure pose also new challenges when it comes to data summarization, understanding, and visualization [7]. Visualizing the large heterogeneous graph data of KGs is particularly difficult when it comes to deciding what information to show to the user and in which form.

5 KG EXPLORATION REQUIREMENTS

Our analysis (summarized in Figure 3) started from the goals and the tasks characterizing KG exploration on the one hand and considers the existing challenges, on the other hand. These aspects are linked together and lead to the new requirements for KG exploration systems discussed here. We discuss a set of requirements and desiderata for KG exploration systems as a logical conclusion of our analysis. In particular, we identify five operations that KG systems should implement to properly support KG exploration.

Approximate similarity search. KG exploration often starts by means of a keyword search query asking for values containing at least partial matches to node and edge attributes. Yet, existing GDBMSes offer only limited support to these queries and no support for approximate query answering besides partial string matching. Often, users ask for nodes or edges similar to the ones they know. Yet, the notion of similarity varies substantially across domains and applications, e.g., short distance in a taxonomy vs. similarity

of attribute values. These operations are very costly and require specialized algorithms. Nonetheless, to date, no system provides native support for such operations.

Pattern mining and connectivity search. GDBMSes excel in queries such as finding reachable nodes given some conditions on path length and node/edge labels. An exploratory search asks for arbitrarily long paths with limited or no knowledge of edge or node labels. Oftentimes, the user would like to find the k most relevant structures given an unspecified proximity measure [21]. On another vein, graphs expose repeated structures that form patterns revealing important connections among entities. As such, pattern queries [72] constitute another important, yet loosely supported, mechanism. These exploratory pattern queries involve large search spaces due to underspecified query conditions or malformed input calling for approximations. Finally, analytical queries ask for aggregates, e.g., counting nodes matching some structures and attributes, that require advanced techniques, such as sampling, or pre-computed statistics and view materialization [19, 27, 29, 57, 67].

Progressive and incremental results. Oftentimes during exploration, users need a sample of the answers to a query to determine whether the results are relevant without requiring all results. Therefore, a system should optimize for minimizing the time required to provide the representative results first and return more results progressively as time passes. Additionally, when the data changes, continuous queries [70], e.g., those that are used to produce dashboards and visualizations, need to show updated results efficiently by only considering the latest data changes.

Iterative query modification. Exploratory search is naturally an iterative process in which the user issues sequences of queries in which each query slightly perturbs the previous one. In this query modification scenario, the system should be able to re-use even partial computations from a previous query instead of recomputing everything from scratch. Moreover, by examining both the graph and past interactions, the system could also predict the next query reformulation and either proactively compute some results or use this prediction to further guide the user in their next queries.

Adaptive storage and indexing. As the KG changes so should any index and data structure in the database. While, on the one hand, these structures are necessary to deliver fast responses, on the other hand, current indexes do not adaptively change with the data. Yet, KGs' heterogeneity and evolution, linked to the high variety of exploratory use-cases, imposes a need for adaptivity to these indexes and data representation [60]. In particular, data structures like reachability indexes or graph partitions becomes quickly obsolete if the data constantly updates. As such, KG exploration demands new types of data structures that lazily adapt to the user queries and the data updates.

The aforementioned requirements have long been identified and investigated [39, 50, 61]. Notwithstanding, system designers have only partially prioritized such requirements to support KG exploration. Designing the system from the bottom up, we first need new query operators, such as example-based queries [39] and more flexible approximate search operators, *directly integrated within the system, instead of treating them as separate applications*. Subsequently, we need query processing and optimization that are aware of the complexity of the exploratory workload and not merely targeting point-wise operators. Finally, we need elastic query processing, query caching, and partial query answering techniques supported by adaptive and on-demand data structures.

6 CONCLUSIONS AND FUTURE DIRECTIONS

This paper provides a taxonomy of KG exploration methods divided into profiling, search, and analytics. As shown, challenges of KG exploration go beyond scale and concentrate around heterogeneity, the evolution of the data, and the vagueness of the user needs. We discuss how novel KG exploration systems are required to support operations like approximate similarity search, pattern mining, and progressive, incremental, and iterative query answering using adaptive storage and indexes. In Figure 4, we show how the various components integrate and respond to a given example workload. In particular, we see the connections between the various techniques presented above and the core components of the system: query operators, query processing, and the storage layer. An analysis of the state of the art (Figure 2) and the connections between the goals and the challenges posed by KG exploration use cases (Figure 3) identify two important research avenues for KG exploration systems: (1) more extensive support *within the system* for example-based and similarity-based approximate exploratory methods and (2) enhanced interactivity, personalization, and performance through adaptive systems (which would likely require new applications of machine learning techniques).

Support for Example-based and Similarity-based methods. Example-based approaches [39] provide the unique opportunity to lift the requirement to be familiar with the structure of the data and the query language. At their core, they rely on a flexible notion of *similarity*. Despite their importance and potential in helping users explore a KG, no system natively and effectively supports them. Currently, example-based approaches exist only for exploratory search tasks and not for exploratory analytics. Consequently, *example-driven exploratory analytics for KGs* is largely unexplored. These analytics should combine data summarization and exploratory search and return statistics based on examples. Data summarization and profiling techniques should also extract context-specific insights. The *context* and information needs have to be inferred from the user-provided examples. Additionally, we can envision system-level support for example-driven visualization suggestions. These suggestions can help the user identify further analytical explorations and should be guided by the information derived by the user provided examples.

Machine learning for KG exploration systems. Data and KG exploration should account for the specific user need through *interactive* and *personalized* methods. During interaction with the user, the system improves and learns more about the user needs to enable personalization. Machine learning and active search are a promising ground [46] to learn user preferences from past interactions, e.g., analyzing the query log, and adapt to the user's needs. Yet, existing exploratory methods are mainly data-driven: they assume fixed user preferences or refinement criteria based on hardwired rules. While few exploratory methods learn a dynamic notion of *interestingness* from the user input and interaction with the system, e.g., query reverse engineering [11] or query suggestion [41], most approaches lack this feature. Personalization also

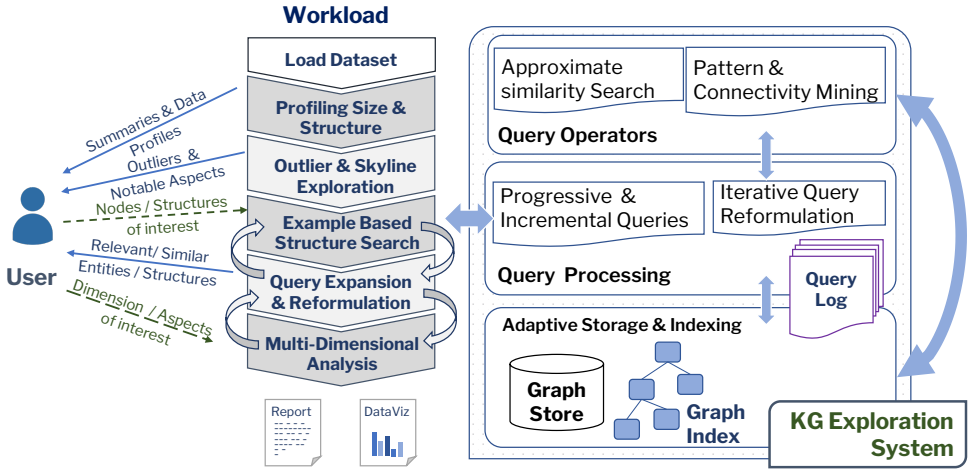

Figure 4: KG Exploration System Architecture

provides useful insights tailored to each individual user [7]. Other desiderata include the capability to learn exploration criteria (e.g., similarity measures) and computation strategies (e.g., query optimizers). Finally, we have recently seen how machine learning can be adopted for learned data representation and indexes [12], applying these techniques on-the-fly to expedite and support KG exploration workloads is a promising research direction.

Conclusions. *Are we lost?* No, but the time is ripe for investing in systems supporting KG exploration. We identified a set of challenges, opportunities, and requirements to empower current systems with exploration capabilities and support users from disparate domains and with different information needs. These, in turn, require the implementation of novel operators, data structures, and query optimizers that can explicitly handle the heterogeneity of KGs and the computational requirements of data exploration.

ACKNOWLEDGMENTS

Matteo Lissandrini is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant No 838216. Davide Mottin is supported by the Horizon Europe under the Eureka, Eurostar grant no E115712. Katja Hose is supported by the Poul Due Jensen Foundation. Torben Bach Pedersen is supported by the SEMIOTIC grant from Independent Research Fund Denmark.

REFERENCES

[1] Nacira Abbas, Kholoud Alghamdi, Mortaza Alinam, Francesca Alloatti, Glenda Amaral, Claudia d'Amato, Luigi Asprino, Martin Beno, Felix Bensmann, Russa Biswas, et al. 2020. Knowledge Graphs Evolution and Preservation–A Technical Report from ISWS 2019. *arXiv preprint arXiv:2012.11936* (2020).

[2] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. 2018. Data profiling. *Synthesis Lectures on Data Management* 10, 4 (2018), 1–154.

[3] Ziawasch Abedjan, Toni Grütze, Anja Jentzsch, and Felix Naumann. 2014. Profiling and mining RDF data with ProLOD++. In *ICDE*. IEEE, 1198–1201.

[4] Alberto Abelló, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga, Victoria Nebot, Maria Jose Aramburu, and Alkis Simitsis. 2015. Using semantic web technologies for exploratory OLAP: a survey. *TKDE* 27, 2 (2015), 571–588.

[5] Lukas Berg, Tobias Ziegler, Carsten Binnig, and Uwe Röhm. 2019. ProgressiveDB: progressive data analytics as a middleware. *PVLDB* 12, 12 (2019), 1814–1817.

[6] Anders H. Brams, Anders Langballe Jakobsen, Theis E. Jendal, Matteo Lissandrini, Peter Dolog, and Katja Hose. 2020. MindReader: Recommendation over Knowledge Graph Entities with Explicit User Ratings. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2975–2982. https://doi.org/10.1145/3340531.3412759

[7] Irène Burger, Ioana Manolescu, Emmanuel Pietriga, and Fabian Suchanek. 2020. Toward Visual Interactive Exploration of Heterogeneous Graphs. In *SEAdata Workshop*. CEUR-WS.org.

[8] Šejla Čebirić, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. 2019. Summarizing semantic graphs: a survey. *The VLDB Journal* 28, 3 (2019), 295–327.

[9] Dario Colazzo, François Goasdoué, Ioana Manolescu, and Alexandra Roatiş. 2014. RDF analytics: lenses over semantic graphs. In *The Web Conference*. 467–478.

[10] Yanlei Diao, Paweł Guzewicz, Ioana Manolescu, and Mirjana Mazuran. 2021. Efficient Exploration of Interesting Aggregates in RDF Graphs. In *SIGMOD*. 392–404.

[11] Gonzalo Diaz, Marcelo Arenas, and Michael Benedikt. 2016. SPARQLByE: Querying RDF data by example. *PVLDB* 9, 13 (2016), 1533–1536.

[12] Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. 2020. Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads. *PVLDB* 14, 2 (2020), 74–86.

[13] Dongmei Ren, Baoying Wang, and W. Perrizo. 2004. RDF: a density-based outlier detection method using vertical data representation. In *ICDM*. IEEE, 503–506.

[14] Ahmed El-Roby, Khaled Ammar, Ashraf Aboulnaga, and Jimmy Lin. 2016. Sapphire: Querying RDF Data Made Simple. *Proc. VLDB Endow.* 9, 13 (2016), 1481–1484. https://doi.org/10.14778/3007263.3007289

[15] Jing Fan, Adalbert Gerald Soosai Raj, and Jignesh M Patel. 2015. The Case Against Specialized Graph Analytics Engines.. In *CIDR*.

[16] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *SIGMOD*. 1843–1857.

[17] Jean-Daniel Fekete, Danyel Fisher, Arnab Nandi, and Michael Sedlmair. 2018. Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411). *Dagstuhl Reports* 8, 10 (2018), 1–40. https://doi.org/10.4230/DagRep.8.10.1

[18] Sébastien Ferré. 2017. Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web* 8, 3 (2017), 405–418.

[19] Luis Galárraga, Kim Ahlstrøm Jakobsen, Katja Hose, and Torben Bach Pedersen. 2018. Answering Provenance-Aware Queries on RDF Data Cubes Under Memory Budgets. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11136)*. Springer, 547–565.

[20] Enrico Gallinucci, Matteo Golfarelli, Stefano Rizzi, Alberto Abelló, and Oscar Romero. 2018. Interactive multidimensional modeling of linked data for exploratory OLAP. *Information Systems* 77 (2018), 86–104.

[21] Denis Gallo, Matteo Lissandrini, and Yannis Velegrakis. 2020. Personalized page rank on knowledge graphs: Particle Filtering is all you need!. In *EDBT 2020*. 447–450.

[22] Agneta Ghose, Katja Hose, Matteo Lissandrini, and Bo Weidema Pedersen. 2019. An Open Source Dataset and Ontology for Product Footprinting. In *ESWC Satellite Events*. Springer, 75–79.

[23] Subhasis Ghosh, Arpita Kundu, Aniket Pramanick, and Indrajit Bhattacharya. 2020. Discovering Knowledge Graph Schema from Short Natural Language Text via Dialog. In *SIGDIAL*. 136–146.

[24] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[25] Emil Riis Hansen, Matteo Lissandrini, Agneta Ghose, Søren Løkke, Christian Thomsen, and Katja Hose. 2020. Transparent Integration and Sharing of Life Cycle Sustainability Data with Provenance. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12507)*. Springer, 378–394.

[26] Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. 2014. Towards Exploratory OLAP Over Linked Open Data - A Case Study. In *Enabling Real-Time Business Intelligence - International Workshops, BIRTE 2013, Riva del Garda, Italy, August 26, 2013, and BIRTE 2014, Hangzhou, China, September 1, 2014, Revised Selected Papers (Lecture Notes in Business Information Processing, Vol. 206)*. Springer, 114–132.

[27] Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. 2016. Optimizing Aggregate SPARQL Queries Using Materialized RDF Views. In *International Semantic Web Conference (ISWC)*, Vol. 9981. 341–359.

[28] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of data exploration techniques. In *SIGMOD*. 277–281.

[29] Kim Ahlstrøm Jakobsen, Alex B. Andersen, Katja Hose, and Torben Bach Pedersen. 2015. Optimizing RDF Data Cubes for Efficient Processing of Analytical Queries. In *International Workshop on Consuming Linked Data co-located with 14th International Semantic Web Conference (ISWC 2105)*, Olaf Hartig, Juan F. Sequeda, and Aidan Hogan (Eds.), Vol. 1426.

[30] Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. 2021. Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks. In *ACL*. 850–862.

[31] Mayank Kejriwal. 2020. Knowledge Graphs and COVID-19: Opportunities, Challenges, and Implementation. *Harvard Data Science Review* (2020).

[32] Ilkcan Keles and Katja Hose. 2019. Skyline queries over knowledge graphs. In *ISWC*. Springer, 293–310.

[33] Martin L Kersten, Stratos Idreos, Stefan Manegold, and Erietta Liarou. 2011. The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB* 4, 12 (2011), 1474–1477.

[34] Sebastian Kruse, Anja Jentzsch, Thorsten Papenbrock, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Felix Naumann. 2016. RDFind: Scalable conditional inclusion dependency discovery in RDF datasets. In *SIGMOD*. 953–967.

[35] Ora Lassila, Michael Schmidt, Brad Bebee, Dave Bechberger, Willem Broekema, Ankesh Khandelwal, Kelvin Lawrence, Ronak Sharda, and Bryan Thompson. 2021. Graph? Yes! Which one? Help! *arXiv preprint arXiv:2110.13348* (2021).

[36] Hanâ Lbath, Angela Bonifati, and Russ Harmer. 2021. Schema Inference for Property Graphs. In *EDBT*. OpenProceedings.org, 499–504.

[37] Matteo Lissandrini, Martin Brugnara, and Yannis Velegrakis. 2018. Beyond macrobenchmarks: microbenchmark-based graph database evaluation. *PVLDB* 12, 4 (2018), 390–403.

[38] Matteo Lissandrini, Davide Mottin, Themis Palpanas, Dimitra Papadimitriou, and Yannis Velegrakis. 2015. Unleashing the Power of Information Graphs. *SIGMOD Record* 43, 4 (Feb. 2015), 21–26.

[39] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. 2018. *Data Exploration Using Example-Based Methods*. Morgan & Claypool Publishers.

[40] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. 2018. Multi-example search in rich information graphs. In *ICDE*. IEEE, 809–820.

[41] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. 2020. Graph-Query Suggestions for Knowledge Graph Exploration. In *The Web Conference 2020* (Taipei, Taiwan). ACM, New York, USA, 2549–2555.

[42] Matteo Lissandrini, Torben Bach Pedersen, Katja Hose, and Davide Mottin. 2020. Knowledge graph exploration: where are we and where are we going? *ACM SIGWEB Newsletter* Summer 2020 (2020), 1–8.

[43] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *Comput. Surveys* 51, 3 (2018).

[44] Steffen Metzger, Ralf Schenkel, and Marcin Sydow. 2017. QBEES: query-by-example entity search in semantic knowledge graphs based on maximal aspects, diversity-awareness and relaxation. *J. Intell. Inf. Syst.* 49, 3 (2017), 333–366.

[45] Amine Mhedhbi, Matteo Lissandrini, Laurens Kuiper, Jack Waudby, and Gábor Szárnyas. 2021. LSQB: a large-scale subgraph query benchmark. In *GRADES*.

[46] Tova Milo and Amit Somech. 2020. Automating Exploratory Data Analysis via Machine Learning: An Overview. In *SIGMOD*. ACM, 2617–2622.

[47] Davide Mottin, Francesco Bonchi, and Francesco Gullo. 2015. Graph query reformulation with diversity. In *KDD*. 825–834.

[48] Davide Mottin, Matteo Lissandrini, Senjuti Basu Roy, and Yannis Velegrakis (Eds.). 2021. *Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021) co-located with 47th International Conference on Very Large Data Bases (VLDB 2021), Copenhagen, Denmark, August 20, 2021*. CEUR Workshop Proceedings, Vol. 2929. CEUR-WS.org. http://ceur-ws.org/Vol-2929

[49] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. 2016. Exemplar Queries: A New Way of Searching. *The VLDB Journal* 25, 6 (2016), 741–765.

[50] Davide Mottin and Emmanuel Müller. 2017. Graph exploration: From users to large graphs. In *SIGMOD*. AMC, 1737–1740.

[51] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: Lessons and challenges. *ACM Queue* 17, 2 (2019), 48–75.

[52] Anil Pacaci, Alice Zhou, Jimmy Lin, and M Tamer Özsu. 2017. Do we need specialized graph databases? Benchmarking real-time social networking applications. In *GRADES*. 1–7.

[53] Olivier Pelgrin, Luis Galárraga, and Katja Hose. 2020. Towards fully-fledged archiving for RDF datasets. *Semantic Web* Preprint (2020), 1–24.

[54] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. 2020. YAGO 4: A Reason-able Knowledge Base. In *The Semantic Web*, Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez (Eds.). Springer, 583–596.

[55] Giulia Preti, Matteo Lissandrini, Davide Mottin, and Yannis Velegrakis. 2019. Mining patterns in graphs with multiple weights. *DAPD* 37 (18 Feb 2019), 1–39.

[56] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *SEMANTICS*. 32–40.

[57] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2021. Optimizing SPARQL Queries using Shape Statistics. In *EDBT*. OpenProceedings.org, 505–510.

[58] Erhard Rahm and Philip A Bernstein. 2006. An online bibliography on schema evolution. *SIGMOD Rec* 35, 4 (2006), 30–31.

[59] Tara Safavi, Caleb Belth, Lukas Faber, Davide Mottin, Emmanuel Müller, and Danai Koutra. 2019. Personalized knowledge graph summarization: From the cloud to your pocket. In *ICDM*. 528–537.

[60] Tomer Sagi, Matteo Lissandrini, Katja Hose, and Torben Bach Pedersen. 2022. A Design Space for RDF Data Representations. *The VLDB Journal* (2022).

[61] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. 2020. The ubiquity of large graphs and surprising challenges of graph processing: extended survey. *The VLDB Journal* 29, 2 (2020), 595–618.

[62] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. [n.d.]. The Future Is Big Graphs: A Community View on Graph Processing Systems. *CACM* ([n. d.]).

[63] Muhammad Saleem, Gábor Szárnyas, Felix Conrads, Syed Ahmad Chan Bukhari, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2019. How Representative Is a SPARQL Benchmark? An Analysis of RDF Triplestore Benchmarks. In *WWW* (San Francisco, CA, USA). ACM, 1623–1633.

[64] Simon Scheider, Auriol Degbelo, Rob Lemmens, Corné van Elzakker, Peter Zimmerhof, Nemanja Kostic, Jim Jones, and Gautam Banhatti. 2017. Exploratory querying of SPARQL endpoints in space and time. *Semantic web* (2017), 65–86.

[65] Juan Sequeda and Ora Lassila. 2021. Designing and Building Enterprise Knowledge Graphs. *Synthesis Lectures on Data, Semantics, and Knowledge* 11 (2021).

[66] Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, and Dimitris Plexousakis. 2015. RDF digest: Efficient summarization of RDF/S KBs. In *ESWC*. 119–134.

[67] Georgia Troullinou, Haridimos Kondylakis, Matteo Lissandrini, and Davide Mottin. 2021. SOFOS: Demonstrating the Challenges of Materialized View Selection on Knowledge Graphs. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD/PODS '21)*. Association for Computing Machinery, New York, NY, USA, 2789–2793.

[68] J. Varga, L. Etcheverry, A. A. Vaisman, O. Romero, T. B. Pedersen, and C. Thomsen. 2016. QB2OLAP: Enabling OLAP on Statistical Linked Open Data. In *ICDE*. 1346–1349.

[69] Shenghui Wang, Stefan Schlobach, and Michel Klein. 2011. Concept drift and how to identify it. *Journal of Web Semantics* 9, 3 (2011), 247–265.

[70] Lefteris Zervakis, Vinay Setty, Christos Tryfonopoulos, and Katja Hose. 2020. Efficient Continuous Multi-Query Processing over Graph Streams. In *EDBT*. 13–24.

[71] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning. *SIGIR* 43 (2020), 10 pages.

[72] Mussab Zneika, Claudio Lucchese, Dan Vodislav, and Dimitris Kotzinos. 2016. Summarizing linked data RDF graphs using approximate graph pattern mining. In *EDBT 2016*. OpenProceedings.org, 684–685.