# Knowledge Cubes - A Proposal for Scalable and Semantically-Guided Management of Big Data

Amgad Madkour
Purdue University
West Lafayette, USA
Email: amgad@cs.purdue.edu

Walid G. Aref
Purdue University
West Lafayette, USA
Email: aref@cs.purdue.edu

Saleh Basalamah
Umm Al-Qura University
Makkah, KSA
Email: smbasalamah@uqu.edu.sa

***Abstract*—A Knowledge Cube, or cube for short, is an intelligent and adaptive database instance capable of storing, analyzing, and searching data. Each cube is established based on semantic aspects, e.g., (1) Topical, (2) Contextual, (3) Spatial, or (4) Temporal. A cube specializes in handling data that is only relevant to the cube's semantics. Knowledge cubes are inspired by two prime architectures: (1) *Dataspaces* that provides an abstraction for data management where heterogeneous data sources can co-exist and it requires no prespecified unifying schema, and (2) *Linked Data* that provides best practices for publishing and interlinking structured data on the web. A knowledge cube uses Linked Data as its main building block for its data layer and encompasses some of the data integration abstractions defined by Dataspaces. In this paper, knowledge cubes are proposed as a semantically-guided data management architecture, where data management is influenced by the data semantics rather than by a predefined scheme. Knowledge cubes support the five pillars of Big Data also known as the five V's, namely Volume, Velocity, Veracity, Variety, and Value. Interesting opportunities can be leveraged when learning the semantics of the data. This paper highlights these opportunities and proposes a strawman design for knowledge cubes along with the research challenges that arise when realizing them.**

***Index Terms*—big data architecture; semantics; data management**

## I. Introduction

Today, we are surrounded by a plethora of information usually coined under the term *Big Data*. Big data requires exceptional technologies to efficiently process the large quantities of data within tolerable elapsed times. The types of data include social media shared content, web pages, blogs, photos, videos, and transaction records, to name a few. We need systems that can provide answers to questions that have been considered beyond our reach in the past, given the amount of information that is required to be analyzed.

We describe two prime architectures for data management that inspire our proposal, namely *Dataspaces* [1] and *Linked Data* [2] and illustrate how a hybrid architecture could be created around them that tackles the challenges of Big Data [3].

Franklin, Halvey, and Maier coin the term *Dataspaces* [1], which is an abstraction for data bases that attempts to avoid data integration problems. Dataspace systems use a Dataspaces Support Platform (DSSP) to provide interrelated services over its components. Dataspaces consist of six logical components: (1) *Catalog* for maintaining information about data sources, (2) *Local store and index*, (3) *Search and query* mechanism, (4) *Administration*, (5) *Discovery* for locating new data sources and maintaining existing ones, and (6) *Source extension* that embeds participants with additional capabilities, e.g., schema, catalog, keyword search, and update monitoring. It is expensive to model an integration schema upfront especially with large-scale integration scenarios involving Big Data scale. In order to mitigate this, the data is not integrated by default, and hence the system can answer only simple queries. If more data needs to be integrated, then it can follow a pay-as-you-go model. Dataspaces are best suited for small-scale, loosely coupled heterogeneous data sources [11]. It relies on a centralized catalog that is a key component in the success of the Dataspaces support platform. Given the Big Data scale, it would not be possible to rely on a central repository for materializing web data given its dynamic nature [11].

Tim-Berners Lee first coined the term Linked Data [2] in 2006. Linked Data is about using the Web to connect related data that has not been previously linked. It is a way of exposing, sharing, and connecting loosely coupled pieces of data and creating new knowledge. Unlike Dataspaces, one of the main principles of Linked Data lies in its decentralized setting. that facilitates distributed publishing and maintenance of the combined data. It uses a rich data model, namely RDF [4], to express the data through statements of equivalence, subclasses, and sub-property relations.

Many organizations and governments have adopted Linked Data as a way to publish their data. This has created a global data space referred to as the Web of Data [5]. The Web of Data contains billions of RDF statements from a variety of sources covering various topics. In fact, the Web of Data can be considered a Dataspace with the difference that it is now distributed and spans a more global scale [6].

Another architecture for learning from web-scale data is NELL [7]. NELL realizes a never-ending learning agent that starts with an initial knowledge base of predicates and relations. NELL grows the knowledge base continuously by reading and learning from the web using various components that make uncorrelated errors, learning multiple types of interlinked knowledge, using coupled semi-supervised learning methods to leverage constraints among predicates, distinguishing high-confidence beliefs from lower ones, and using a uniform knowledge base representation to capture facts. NELL needs human intervention to assert the correctness of identified

entities to avoid aggregating the mistakes further.

We propose Knowledge Cubes, a data management architecture for the Web of Data that relies on semantics to define how the data is fetched, organized, stored, optimized, and queried. Knowledge cubes use RDF to store data. This allows knowledge cubes to store Linked Data from the Web of Data. Knowledge cubes are envisioned to break down the centralized architecture into multiple specialized cubes, each having its own index and data store. When a user query relies on data from multiple cubes, these specialized cubes communicate to satisfy the user's query. A cube is intelligent in identifying when to update its data based on query predicates and the frequency of the requested items.

In the following sections, we introduce knowledge cubes, and discuss the motivation for each of their architectural components. Then, we illustrate how knowledge cubes cater to the needs of the five pillars of Big Data [8]. Finally, we provide a strawman's design for the knowledge cubes architecture and highlight research challenges in its realization.

## II. Motivation

Handling Big Data scale is a fundamental challenge for the Web of Data. Given the growing availability of Linked Data, we are in need of systems capable of storing RDF's and evaluating complex queries over these large silos of interlinked datasets. To be efficient, these systems should maintain a reasonable ratio of RAM and database size [9]. They also need to have one index per database that can be updated as needed. The system describes its data with a well-defined representation, e.g., RDF, that contains rich constructs to interlink Big Data resources. We describe the following three motivational points that drive our knowledge cubes proposal.

### A. Semantically Partitioned Data

We propose to partition and manage data based on semantics rather than on a predefined partitioning scheme. Linked Data with its rich semantics provides an excellent foundation for understanding and linking interrelated data. With semantically partitioned data, one can forward questions directly to specific knowledge cubes and get fine-grained answers in contrast to contacting a heterogeneous data source or multiple data sources in order to get an answer. An advantage of this approach is that each specialized knowledge cube stores and continuously updates data from the web that is related only to the topic of interest to the knowledge cube, and thus resulting in a notion of specialization. These specialized knowledge cubes can inter-communicate to provide in-depth answers to users' queries. This approach also allows breaking the large data silo into meaningful and interrelated sets.

### B. Unsupervised Learning from Big Data Sources

To harness the richness of Big Data, knowledge cubes need to harvest as much information as possible from Big Data sources to enrich the current Linked Data of the knowledge cube. This needs to be performed in an unsupervised fashion due to the immense amount of data that needs to be processed and hence resulting in some uncertainties. The aim is to accommodate as much information as possible in the system instead of creating a high-quality information repository. Similar to Dataspaces [1], high-quality information can be obtained using a pay-as-you-go model over the data of interest.

### C. Understanding Query Semantics

We need to understand user intent from the queries to provide the best matching results. This calls for a rich query language capable of expressing that intent. Semantic Web query languages, e.g., SPARQL [10], should be tailored to express important aspects including spatial, temporal, textual, and relational. While some of these features are supported, others, e.g., structured, keyword, and meta-data queries, are not efficiently supported [11]. The system should disambiguate a query to find its true meaning. Linked Data provides the necessary constructs that can help understand true user intent.

## III. Proposal

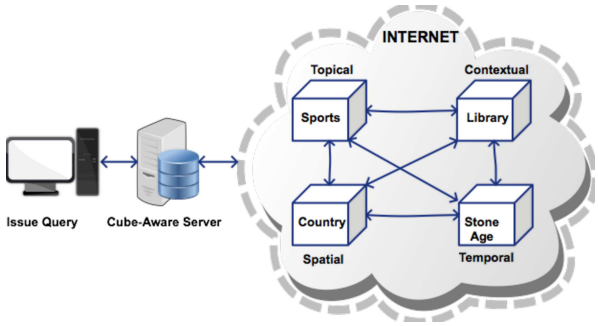


Fig. 1. Knowledge Cubes - The Big Picture

Initially, a knowledge cube is created based on one or more semantic aspect, such as topical, contextual, spatial, or temporal. For example, a knowledge cube can focus on a sport (topical), a library in a university (contextual), a country (spatial), the stone age (temporal), or a combination of these or other semantic aspects. Then, the knowledge cube extracts relevant information based on its own semantic aspects from existing Linked Data datasets. This creates a notion of specialization based on the rich Linked Data information at the cube level.

The data extracted should conform to Linked Data principles and should be stored in the RDF model [12]. The cube fetches further resources from the Web of Data that matches the cube's semantics. Once a query is issued, a *cube-aware server* identifies which knowledge cube is suitable to answer the query. If a knowledge cube needs further information in order to disambiguate or enhance the query answer, it communicates with other knowledge cubes. Figure 1 illustrates the big picture of querying the system.

## IV. Founding Principles

In this section, we present the founding principles that guide the knowledge cubes architecture.

### A. Structural Evolution

As in Figure 1, a knowledge cube captures the notion of specialization, be it spatial, temporal, topical, contextual, or otherwise. Its granularity level adapts to represent finer grained knowledge cubes. For example, the "Sports" knowledge cube can be broken into multiple cubes, e.g., "Basketball" and "Baseball". These two sub-cubes would still be inter-linked together. This inter-linking is important as it can provide more insightful information. For example, an interesting relationship would be that "Michael Jordan" who is a Basketball player (and hence, belonging to the Basketball cube) has a "played" relationship with the Baseball cube. This means that the decoupling of information does not mean breaking some relationships, but instead decouples the main information into partitions. This structural evolution creates space to accumulate further information relevant to the semantics of the cube on both sides that in turn provides scalability to the whole system.It also allows minimizing the communication overhead between knowledge cubes. We refer to this behavior as the *structural evolution* of a knowledge cube.

A knowledge cube may evolve based on its newly attained size or semantic aspect by re-partitioning dynamically in an unsupervised fashion. The evolution of knowledge cubes involves a number of aspects (1) *Merging and Splitting*, where a cube merges or splits based on the semantic aspects (e.g., sports into basketball and football, or vice versa) (2) *Arbitration and Prioritization*, where a knowledge cube gives precedence to certain structural changes based on the overall status of the system. The structural evolution leads to the creation of further specialized cubes that are capable of learning more insightful information relating to their semantic aspects. The semantics of user queries also dictates if there is a need for a structural evolution in case it forces the system to contact more than one knowledge cube.

### B. Temporal Evolution

A knowledge cube organizes it own data temporally using *a time-roadmap*. The time-roadmap can be identified by (1) the entities involved in the data units, (2) the timestamps of the data units, and (3) the relationships of the entities involved and their respective time-span. For example, a user may be interested in a query related to the activities of the U.S. president who held office during a specific time-span. The time-roadmap will display entries for all the U.S presidents that held office in that time period in addition to any activities that they performed that overlap with the requested time-span. Knowledge cubes operate on Linked Data that originate from well-structured data that progressively evolve over time using other structured or unstructured data. Each data unit or entity has a belief value (degree of truthfulness) associated with it.

### C. Analytic Distribution

An important principle is the distribution of the analytic load across multiple knowledge cubes and then communicating the results back according to relevance. The analytic load is composed of extracting the data, determining its level of uncertainty, and linking it with the existing Linked Data in the cube. The advantage of distributing the analytic load is to allow for the simplification in development and maintenance. The knowledge cube approach is orthogonal to the data cube approach in [13] as knowledge cubes focus on one specific dimension only. Also, the knowledge cube's RDF data model allows it to support graph-based analytics, e.g., random walks, regular expression and reachability queries, distance oracles, and community searches.

## V. Strawman Design

In this section, we present a strawman design of the components needed for the realization of knowledge cubes.

### A. Cube-awareness

This component is responsible for making a knowledge cube aware of all the other cubes relevant to it. Cube-awareness provides structural or data-level updates to the hosting cube. This component guarantees to the hosting instance the most updated information that other cubes possess. The cube-awareness component appears in two states, (1) the *stand-alone* state, where it resides on a cube-aware server so that when it receives a user's query, it forwards the query to the appropriate knowledge cube, and (2) the *Embedded* state, where the cube-aware component provides each knowledge cube with the necessary information regarding other relevant knowledge cubes. Relevance among knowledge cubes is defined by the set of common relations they share. These relations are defined based on the Linked Data of each cube.

### B. Catalog

The catalog maintains all the information related to the data sources it fetches [1]. It maintains a list of the data elements that each source captures. For example, each catalog maintains a list of all named entities, e.g., countries, people, buildings, etc., for each source. In contrast to Dataspaces [1], the catalog maintains information about the other knowledge cubes' catalogs. Knowledge cubes coordinate with each other the content acquired in order to maximize partitioning, and hence increase their specialization. A cube-aware server acts as a dispatcher for data sources, where it indicates to each knowledge cube what data source to handle.

The catalog addresses the *variety* aspect of Big Data. Each cube fetches heterogeneous data sources related to the cube's semantic focus. For example, a news source publishes an article that can also be discussed over social media channels, e.g., twitter or facebook. Exhibiting this behavior over multiple specialized cubes, each with its separate view of heterogeneity, allows introducing *variety* into the knowledge cube.

### C. Information Extraction

This component employs text analysis techniques in order to extract and learn from structured and unstructured sources. Systems, e.g., NELL [7], contain a free-text extractor termed Coupled Pattern Learner (CPL) [14] that couples the semi-supervised training of extractors of different categories and

relations. NELL is capable of extracting patterns in the form of "X playsIn Y" and "president of X". Given the architecture of knowledge cubes, we suggest that such extractors could be trained with the initial Linked Data of the cube since the training patterns are similar to the RDF model of Linked Data. We believe that having an extractor per cube with an initial limited set of fine-tuned labeled data that evolves over time should lead to a better extraction accuracy for the cube rather than a globally trained extractor [15]. Unlike NELL [7], our proposed extraction patterns rely on the structural evolution of the cube and not on the initial seed only.

The Information Extraction component tackles the *veracity* aspect of Big Data. The use of Linked Data creates a rich platform for understanding the query and thus creating added *value* out of the underlying data. The *veracity* of the identified entities has to be verified before it can be used for generating valuable results or linked to existing data.

### D. Search and Querying

This component provides a rich set of constructs that understand the semantics of the query terms specified by the user. A knowledge cube uses its underlying Linked Data in order to disambiguate and discover interesting relations that in turn provides more insight to the results. The search and querying component supports search-based methods that include keyword, textual, spatial, temporal, and semantic-based methods that rely on similarity functions and database-level methods. For example, we need to understand the semantics of queries like "Seafood restaurant by the ocean" or "Cafe with a view on the Eiffel tower". The query is converted into a triple that is matched against the knowledge cubes data that is also formatted as triples.

The search and querying component is responsible for determining the best possible way to communicate with other cubes to satisfy the user query. This requires understanding the communication topology and determining the best routing technique to minimize query time.

The search and querying component tackles the *value* aspect of Big Data. The use of Linked Data creates a rich platform for understanding the query and thus creating added *value* out of the underlying data.

### E. Data Store and Indexing

An integral component that defines the feasibility of the knowledge cubes architecture is the storage and indexing mechanisms over the Linked Data. We need highly efficient and scalable storage and indexing mechanisms that can operate over billions of RDF triples that are available over the Web of Data. Various data stores have been proposed that include native, relational, or hybrid data stores. Using the semantic aspects can aid in creating efficient storage schemes given the RDF data model. Existing proposals need to be modified to consider this aspect.

Another integral factor in the knowledge cubes architecture is its ability to answer queries without accessing the actual data sources. This requires having a descriptive and up-to-date index in order to satisfy the user's query. Given that each cube indexes its own data, the index needs to maintain information about what other cubes are capable of providing and what can aid in satisfying the user query. Another aspect is the ability of the index to create efficient associations among data objects of different sources for the same cube [1].

Query Optimization provide facilities, e.g., query simplification, indexing, and operator implementations. Interesting operators include spatio-temporal top results. SPARQL 1.1 provides rich facilities [16] that allow further optimization by having no centralized coordinating unit for query evaluation between sources.

Similar to Dataspaces, in knowledge cubes, there will be support for high availability and recovery, the ability to trace back to the origin of a token (e.g., from a text file, an XML file, etc.), caching by building additional indexes to support efficient access, caching for generalization of join index (token appears in multiple data sources), caching to reduce the query load on participants that cannot allow ad-hoc external queries [1].

The data store and indexing components tackle the *volume* and *velocity* aspects of Big Data, respectively. The proposed architecture for knowledge cubes suggests that *learning semantics should allow the storage and indexing to scale better than a centralized or distributed mechanism that do not consider the semantics of the data*. Neglecting the scalability factor incurs a higher overhead on the index and incurs a rise in the communication overhead within the distributed system.

### F. Discovery of Data Sources

Each knowledge cube can have its own discovery mechanism for the list of Big Data sources relevant to its semantic aspects. One source that indexes Linked Data is Sindice [17], where it provides relevancy over its indexed Linked Data sources based on the Universal Resource Identifier (URI) or the keyword-based query. Sindice only indexes sources that conform with Linked Data formats. Knowledge cubes can use a web search engine to search for relevant web sources that match the semantic aspects of a cube. This is done by issuing a keyword-based search query with keywords that best describe a cube. One method of determining these keywords can be by using the most frequent ones that appear in the data that the cube holds. The knowledge cube uses the interlinking of relevant documents to retrieve further information that is relevant to that document and to the cube.

The Data Sources Discovery component is responsible for maintaining and creating relationships among data sources and locating new Big Data sources in a semi-supervised fashion. It is also capable of proposing new relationships. Data Sources and Discovery will also monitor the contents of the cube and propose additional relationships over time [1].

### G. Data Sources Update and Extension

Given an initial seed of RDF's, the time-roadmap is responsible for maintaining the updates for the data in the cube. If new information is identified and it relates to existing RDF

entries, a time-oriented snapshot is taken and amended to the current RDF. The time-roadmap guarantees "freshness" of the results by providing or linking information with the latest updates found from sources. The newly acquired data is integrated in an unsupervised manner based on the characteristics of the current RDF data. This data includes the subjects that can help guide the acquisition process. For example, in the news domain, we may have a cube representing the Presidents of the United States. In this case, the RDF entities can be the names of Presidents and can further be used to harvest all information from Big Data sources related to US presidents.

We can integrate newly discovered information and entities that relate to the initial set of identified Presidents based on how much this information is coupled. This will require that the cube perform text analysis on the data to determine the entities involved and then determine whether the identified information relates to the Linked Data of the cube or not. This information can also be shared with other knowledge cubes that may find this analyzed information of relevance.

Similar to Dataspaces, this component can amend data sources that have limited capabilities with additional ones such as schema, catalog, keyword, search, and update monitoring [1]. This allows the heterogeneous sources to share similar services thus allowing the system to harness their information.

## VI. Challenges

This section presents common challenges that knowledge cubes share with the Web of Data. Also, we indicate a set of new challenges that are spawned from the knowledge cubes architecture. We demonstrate how the knowledge cubes architecture can be used to address these challenges.

### A. Challenge 1: Semantic Interpretation

A key issue in learning on Big Data scale lies in understanding the true meaning of the data. This is known as *semantic interpretation* [15]. It deals with the interpretation of ambiguous or imprecise data. Web-scale data is believed to be part of the solution [15]. Given the amount of available structured data, it can resolve semantic heterogeneity. We share the same view as the one in [15] that the challenge lies in choosing a representation that can use unsupervised learning over plenty of unlabeled data rather than relying on limited labeled data.

The semantic interpretation challenge is also apparent in the spatial dimension. One study proposes a set of semantic features derived from web contextual information to create a location disambiguation system for identified locations [18]. Although the study illustrates that the system achieves better results than the state-of-the-art industrial systems, it still does not operate well over heterogeneous data sources. Knowledge cubes can provide a rich platform for semantic aspects that can enhance information extraction systems with features to efficiently disambiguate locations.

The proposed knowledge cubes architecture deals primarily with a heterogeneous set of Big Data sources that may not include any structure or labeled annotations to start with. Knowledge cubes need to rely on aspects beyond the labeled data in order to correctly interpret the data. We believe that endorsing semantic aspects in the architecture of the knowledge cubes can provide an interesting platform for tackling this challenge. For example, using the spatial aspect, we can disambiguate identified organizations or people in a document or query based on other entities identified in the same context and we can use the relations of the cube to disambiguate the identified entities further.

### B. Challenge 2: Uncertainty

Even with highly accurate named-entity extraction and disambiguation systems, there is a need to attach a truth value to the extracted data. This allows the system to select the most reliable information to present in response to a user query. This issue is vital when new data is "linked" to existing data in the cube. We need to attach data with a specific uncertainty value.

Reynolds [19] presents several aspects to the uncertainty challenge. One aspect is the ambiguity resulting from Linking date where the same identified entities occurring in different datasets do not have the same universal resource identifier (URI). Datasets use an imperfect mechanism, namely the "owl:sameAs" relation to co-reference similar entities with heuristic weighting. The same challenge exists in knowledge cubes, specifically, when an entity is identified and needs to be linked to existing identifiers. Knowledge cubes extend this challenge further with regards to entities that map into multiple cubes. The question becomes: Which cube should the entity belong to? The "owl:sameAs" relation is one mean of having the entity mentioned in two cubes. In this case, the relation can act as a connector for both. This raises a bigger concern of duplication of entities across cubes.

The precision of certain values for entities is another important aspect [19]. For example, one entity may have an attribute (e.g., population) captured at different time-spans that is derived from various readings. This raises the concern that when entities are merged, there can be a conflict in values, and hence affecting the merging process. One way is to use the time-roadmap of a knowledge cube to track the attribute value changes over time. A challenge that this suggestion raises is: what is the best way to represent a time-roadmap? and which attributes are to be maintained? We need techniques to identify time-related events upon which the time-roadmap can be built that also may need to consider other factors, e.g., the spatial aspect. We need to maintain an uncertainty value for the attribute in case the event does not include an explicitly specified date and should be implied from the event, e.g., World War 2.

There is also uncertainty associated with the data source. This issue is apparent in the Linked Data settings, especially given the cross linking and the absence of provisioning. In turn, this raises the concern of veracity of the included data source. We suggest that knowledge cubes include well-established provisioning mechanisms in the catalog component in order to tackle this issue.

In [19], Reynolds suggests that the data source selection process should be selective in terms of the data needed. Given the scale of the data handled by the knowledge cubes architecture, the proposed suggestion by Reynolds aligns with knowledge cubes. Reynolds suggests two types of vocabularies to express the data (1) Link vocabulary to provide a common representation for links and (2) Imprecise value vocabulary to provide a common representation for common vocabulary. Given the knowledge cubes architecture, the vocabulary proposal by Reynolds can be a viable solution where each cube includes its own vocabulary. However, we need to allow for cross-talk, e.g., by maintaining relations among existing vocabulary terms from the various vocabularies and providing enough provisioning in order for the relations to still hold. The spatial and temporal dimensions of the knowledge cube data form a rich facility in solving some uncertainty challenges. For example, we can disambiguate two named entities if both belong to the same spatial location or time slot.

### C. Challenge 3: Data Partitioning Scheme

Defining an efficient scheme for partitioning depends on the semantics of the data. For example, specific snapshots of the data may mandate that the data be partitioned according to the named entities contained in the data. Another scheme might see that partitioning based on people who are politicians or soccer players might generate a good scheme. More sophisticated schemes might involve both entities and relations or a combination of both. An important aspect to consider is how big the data is given each scheme and whether there is a form of overlapping among the generated partitions.

We believe that the data is the main driver of the partitioning scheme. Employing some form of data analysis over the acquired data may reveal interesting properties that can aid in defining efficient partitioning schemes. For example, semantic relationships can be learned in an unsupervised fashion from the search queries and their results [20]. This means that the statistics gathered based on user queries can be an asset that aids in identifying an efficient partitioning scheme. In turn, this requires that the system starts with an initial scheme that can evolve over time based on gathered statistics.

### D. Challenge 4: Storage and Indexing

The database and semantic web communities have a number of solutions for storing and querying Linked Data ranging from centralized to distributed data stores [3]. We describe four main categories that have been adopted by both communities and discuss the challenges associated with each of them.

The first category, called Triplestores, stores RDF triples in a single relational table. The attributes specified are either triples (subject, predicate, object) or quadruples (subject, predicate, object, context). Even though disk space for Triplestores is not a major concern in terms of size, a real concern is in maintaining a reasonable RAM and database ratio [9], which is exemplified in the case of Big Data scale.

The second category is vertically-partitioned tables [21] where systems maintain a table per property as many triple patterns have fixed properties. One variation is to use a column store that captures the column entries of the same type. The objective is to avoid reading an entire row into memory in the event that only a few attributes are accessed per query. This approach has scalability problems when the number of properties in an RDF is high [22], [23]. The study in [22], [23] suggests that Triplestores outperform the vertically-partitioned approach when having a larger number of properties.

The third category is schema-specific systems that capture subjects with overlapping properties. These systems have a query processing advantage as the subjects may be accessed together. In [24], Levandoski and Mokbel claim that property tables can outperform Triplestores and vertically-partitioned tables for RDF data. One concern is the storage overhead because subjects may contain multiple objects. The second and third categories also raise an important concern regarding the increased cost of integration quality.

An open question is: Which of the three categories is best suited for embodying the semantic aspects proposed by the knowledge cubes architecture? One proposed direction is to adopt all three categories and apply the most optimum based on the current semantics of the data. This step occurs as the knowledge cube evolves over time.

Defining a suitable index over the data is also a crucial challenge. An index may need to include spatial, temporal, relational, or textual aspects. An advantage of knowledge cubes is that the index would be defined over well-structured information with rich semantics. In turn, this aids in defining a more meaningful indexing structure. The structural integrity of the index would also affect the query processing performance.

### E. Challenge 5: Communication among Cubes

There are many forms of communication in knowledge cubes. One is when a cube needs to know further information about a query that it is processing. Another is when a cube needs to be updated about what other cubes have. Defining an efficient communication protocol that considers the overhead of contacting and retrieving content from other knowledge cubes is vital. Knowledge cubes may co-exist on the same machine (i.e., embedded communication) or on separate machines (i.e., stand-alone). A knowledge cube should be intelligent enough to decide when to *cache* content that it usually retrieves from other knowledge cubes. In turn, this should significantly decrease the communication overhead. Adaptive Replacement Cache (ARC) [25] is one candidate algorithm that can be used to keep track of frequently used and recently used data. It can also be used to maintain historical data regarding eviction for both scenarios. A challenge would be to see if the algorithm will adapt to both data and structural evolution of the knowledge cube.

### F. Challenge 6: Spatial and Temporal Extraction

Published articles or web pages will need to be segmented into individual units of information, termed statements. For each statement, we need to identify the topic of the statement,

the subject, the person, the event, the time interval, or the location that this statement is about, who authored the sentence, where, and when this statement is made, and whether that statement is extracted from some other internet source, etc. For example, in the time dimension, we need to capture two notions of time for a given statement; (1) when that statement is published or written (i.e., its transaction time), and (2) when the facts in the statement took place (i.e., its valid time). The same applies for the space dimension. Two space dimensions need to be extracted: (1) the physical location of the person who published the statement (i.e., the statement's transaction location), and (2) the location where the facts in the statement took place (i.e., its valid location). The same is true for the "person" who wrote the statement and the person that the statement states facts about, etc.

### G. Challenge 7: Data Change Frequency

It is important to identify when a knowledge cube updates its Linked Data. Linked Data is very dynamic in nature where its resources are added, removed, or updated very frequently especially in the Web of Data. Therefore, it is important to investigate measures of "freshness" for the content and interlinked resources. For example, the Linked Data of the cube may indicate certain properties for triples that might conflict with others that are later retrieved as new data. Identifying these conflicts and creating policies of how to handle them is a crucial issue. Adding provenance can be one solution to this issue. However, it raises another question of what version of the data to present as an answer to the query. A study by Umbrich et. al identifies the characteristics of change over Linked Data and compares the change frequency in the Web of Data versus the traditional Web (HTML documents) [26] .

## VII. Conclusion

We present a vision for the knowledge cube architecture that motivates understanding the semantics of the data as a means for solving some of the research challenges we face today, specifically in Big Data. We motivate the use of Linked Data as a mean for achieving this target given the rich semantics that Linked Data provides. We present a strawman design that sketches the possible components of a knowledge cubes system. Each presented component conveys a high-level overview about its potential usage scenarios. We illustrate some of the challenges and potential solutions associated with the knowledge cubes proposal. Finally, we demonstrate how the proposed architectural components can be used to tackle the five pillars of Big Data.

## Acknowledgment

This research was partially supported by the National Science Foundation under Grants III-1117766, and IIS-0964639, and IIS-0916614.

## References

[1] M. J. Franklin, A. Y. Halevy, and D. Maier, "From databases to dataspaces: a new abstraction for information management," *SIGMOD Record*, vol. 34, no. 4, pp. 27–33, 2005.

[2] T. Berners-Lee, "Linked data - design issues," http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[3] A. Harth, K. Hose, and R. Schenkel, "Database techniques for linked data management," in *SIGMOD Conference*, 2012, pp. 597–600.

[4] O. Lassila and R. Swick, "Resource description framework (RDF) model and syntax specification," http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, 1999.

[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[6] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, ser. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. H. Jr., and T. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI*, 2010.

[8] K. Thirunarayan and A. Sheth, "Semantics-empowered approaches to big data processing for physical-cyber-social applications," *Kno.e.sis Technical Report*, pp. 29–53, May 1997.

[9] O. Erling, "Directions and challenges of semdata," in *VLDB Industry position paper*, 2010.

[10] E. Prud'hommeaux and A. Seaborne, "Sparql query language for rdf," http://www.w3.org/TR/rdf-sparql-query/, Tech. Rep., 2008.

[11] J. Umbrich, M. Karnstedt, J. X. Parreira, A. Polleres, and M. Hauswirth, "Linked data and live querying for enabling support platforms for web dataspaces," in *ICDE Workshops*, 2012, pp. 23–28.

[12] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *Int. J. Semantic Web Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[13] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 29–53, Jan. 1997.

[14] A. Carlson, J. Betteridge, R. Wang, E. H. Jr., and T. Mitchell, "Coupled semi-supervised learning for information extraction," in *WSDM*, 2010, pp. 101–110.

[15] A. Y. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[16] C. B. Aranda, M. Arenas, and Ó. Corcho, "Semantics and optimization of the sparql 1.1 federation extension," in *ESWC (2)*, 2011, pp. 1–15.

[17] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: a document-oriented lookup index for open linked data," *IJMSO*, vol. 3, no. 1, pp. 37–52, 2008.

[18] T. Qin, R. Xiao, L. Fang, X. Xie, and L. Zhang, "An efficient location extraction algorithm by leveraging web contextual information," in *GIS*, 2010, pp. 53–60.

[19] D. Reynolds, "Uncertainty reasoning for linked data," in *URSW*, 2009, pp. 85–88.

[20] D. Scott and H. Uszkoreit, Eds., *22nd International Conference on Computational Linguistics, Proceedings of the Conference*, 2008.

[21] D. J. Abadi, A. Marcus, S. Madden, and K. Hollenbach, "Sw-store: a vertically partitioned dbms for semantic web data management," *VLDB J.*, vol. 18, no. 2, pp. 385–406, 2009.

[22] L. Sidirourgos, R. Goncalves, M. L. Kersten, N. Nes, and S. Manegold, "Column-store support for rdf data management: not all swans are white," *PVLDB*, vol. 1, no. 2, pp. 1553–1563, 2008.

[23] D. C. Faye, O. Cure, and G. Blin, "A survey of RDF storage approaches," *ARIMA J.*, vol. 15, pp. 11–35, 2012.

[24] J. Levandoski and M. Mokbel, "Rdf data-centric storage," in *ICWS*, 2009, pp. 911–918.

[25] N. Megiddo and D. S. Modha, "Arc: A self-tuning, low overhead replacement cache," in *Second USENIX Conference on File and Storage Technologies*, ser. FAST '03, 2003, pp. 115–130.

[26] J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres, and S. Decker, "Towards dataset dynamics: Change frequency of linked open data sources," in *LDOW*, 2010.