Semantic data and semantic data storages

Contents

1 Semantic Data

The Semantic Web is an extension of the World Wide Web through standards set by the World Wide Web Consortium (W3C).

The goal of the Semantic Web is to make Internet data machine-readable.

The term was coined by Tim Berners-Lee for a web of data (or data web) that can be processed by machines—that is, one in which much of the meaning is machine-readable.

Berners-Lee originally expressed his vision of the Semantic Web in 1999 as follows:

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

Berners-Lee is now the director of the World Wide Web Consortium (W3C), which oversees the development of Semantic Web standards. Since 2013, Semantic Web activities have been subsumed by Web of Data activities.

1.1 Brief history

Since the early beginning, CS has been concerned with the processing of data. Programming languages provide simple and complex datatypes to store data. Originally, the semantics of these data were hardwired in the programs in which they were interpreted and used.

Around 50 years ago, data began to become separated from the application program to be stored in databases. This allowed one to reuse the same data in different programming contexts and prevented the same data management component from being re-implemented across many applications. The fact that the meaning of the data was no longer hardwired directly into the application program led to mechanisms for representing the structure and semantics of the data being developed.

One such extremely successful structure was the relational data model.

The third area of computational semantics was founded around 1955 with the goal of enabling a computer to act intelligently as humans do, that is, generating Artificial Intelligences. The field began by implementing general problem-solving methods such as global search and theorem proving. However, after a short space of time, the numerical complexity of the tasks involved in intelligent problem solving made it apparent that a machine-understandable representation of the knowledge related to how a problem may be solved efficiently was required.

1.2 Semantics, the Science of (Meaning)

In a relational database, everything is represented in a table, and a row has a key and a column has a name. With this, even with a very simple machine, one can find the phone number of Mr. X if X is the value of the name column and the phone number is the heading of another column.

Unfortunately, with an average Web page, this is far more difficult. As mentioned earlier, hidden in various HTML tags there is a name and somewhere else a phone number (a set of integers including some special characters).

A browser is required to render the information and a human reader to understand the information based on the layout of the website. This is the solution as implemented in the Web which was introduced 20 years ago. As outlined earlier, the sheer simplicity has made the Web an incredible success story with now more than one billion users. Its simplicity also leaves room for improvement.

Semantic technology adds tags to semi-structured information as database technology adds column headings to tabular information. Let us use a small example in Fig. 1.

```
<person>
  <name>Sir Tim</name>
  <phone number>01-444444</phone number>
</person>
```

Fig. 1. A small example

These annotations allow a computer ‘‘to understand’’ that Sir Tim is a name of a person and 01-444444 is his telephone number. In a similar fashion, programs and other computational resources can be described through semantic annotations. This is the essence of Semantic Web technology.

What can be seen from this example is that one needs two things to define the semantics of information: a language such as <X>Y</X> to define the meaning of Y, and terms such as X to denote this meaning.

1.3 Logic

From an algorithmic perspective, implementing logical-reasoning systems demonstrates clearly how complex decidability and complexity are to manage. These are logical paradigms in increasing levels of complexity.

- Propositional Logic is a rather simple logic language providing propositions such as A, B, C, ... and logic connectives such as AND and OR. All interpretations are simply the enumerations of all possible false and true assignments to these propositions. Therefore, propositional logic is decidable, although, already NP-hard.

- First-Order Predicate Logic provides a richer means to define such propositions by providing terms such as c, f(c, X),... and predicate symbols that can be applied to these terms P(c), Q(c), f(c, X),... . Terms can make use of variables that can be existentially or all quantified (i.e., either there must exist a term fulfilling a formula or all terms must fulfill a formula). First-order predicate logic is still semi-decidable.
- Second-Order Predicate Logic and comparable languages drop this limitation. Here, one can apply predicates to other predicates or entire formulae and interpret variables as sets rather than as individuals of a domain of interpretation. Unfortunately, for these languages, already unification, that is, the question of whether two terms can be substituted, is semi-decidable, which means that there is not even an approach for implementing inference in these languages.
- Description Logics provide a whole family of sub-languages of first-order logic of differing complexity. Common among these languages is to restrict the formalism to unary and binary predicates (concepts and properties) and to restrict the usage of function symbols and logical connectors to build complex formulae. The different levels of complexity and the decidability of these languages follow from the precise definition of these restrictions.

Therefore, many different languages have been defined and implemented, many of which contain intractable worst-case behavior but which however still work for many practical applications.

1.4 Semantic Web Languages

HTML provides a number of ways to express the semantics of data. An obvious one is the META tag.

In the time before the wider usage of RDF, systems used the attribute of the anchor tag to encode semantic information. It is also possible to interpret the semantics of HTML documents indirectly. For example, information captured in a heading tag of level one (<H1>) may be used to encode concepts that are significantly important for describing the content of a document. Still, HTML was not designed to provide descriptions of documents beyond that of informing the browser on how to render the contents. Within efforts to stretch the use of HTML to include meaning, the term semantic HTML was created.

The Extensible Markup Language (XML) has been developed as a generic way to structure documents on the Web. It generalizes HTML by allowing user-defined tags. This flexibility of XML, however, reduces the possibilities for the type of semantic interpretation that was possible with the predefined tags of HTML.

The Resource Description Framework (RDF) is a simple data model for semantically describing resources on the Web. Binary properties interlink terms forming a directed graph. These terms as well as the properties are described using URIs.

Since a property can be a URI, it can again be used as a term interlinked to another property. That is, unlike most logical languages or databases, it is not possible to distinguish the language or schema from statements in the language or schema.

For example, in the statement <rdf:type, rdf:type, rdf:Property> it is stated that type is of type property.

Also, unlike conventional hypertext, in RDF, URIs can refer to any identifiable thing (e.g., a person, vehicle, business, or event). This very flexible data model is obviously suitable in the context of a free and open Web; however, it generates quite a headache for logicians who wish to layer a language on top.

RDF schema (RDFS) uses basic RDF statements and defines a simple ontology language. Specifically, it defines entities such as rdfs:class, rdfs:subclass, rdfs: subproperty, rdfs:domain, and rdfs:range, enabling one to model classes, properties with domain and range restrictions, and hierarchies of classes and properties. RDFS is a specific RDF vocabulary for this purpose and is simply RDF plus some more definitions (statements) in RDF.

The Web Ontology Language OWL extends this vocabulary to a full-fledged spectrum of Descriptions Logics defined in RDF, namely, OWL Lite, OWL DL, and OWL Full.

Mechanisms are provided to define properties to be inverse, transitive, symmetric, or functional. Properties can be used to define the membership of instances for classes or hierarchies of classes and of properties.

Frankly, OWL Lite is already quite an expressive Description Logic which makes the development of efficient implementations for large data sets quite challenging and, in practice, as difficult as implementing OWL DL.

However, neither of these languages can make use of full RDF, that is, some valid RDF statements are not valid in Lite or DL. This is due to the fact that logic languages such as Descriptions Logics exclude meta statements, that is, statements over statements.

For RDF and RDFS, this was not a problem since neither language provided mechanisms to define complex logical definitions.

Spoken in a nutshell, Lite and DL define a vocabulary in RDF and restrict the usage of RDF. OWL Full drops these restrictions. OWL Full provides the vocabulary of OWL DL, that is, an expressive Description Logic, and allows for any valid RDF statement. For example, in OWL Full, a class can be treated simultaneously as a set of individuals and as an individual. Therefore, OWL Full is beyond the expressive scope of Description Logic and minimally requires a theorem prover type of inference such as firstorder logic (i.e., is semi-decidable).

1.5 The Tower of Babel

The Open Systems Interconnection model (OSI model) is a product of the Open Systems Interconnection effort at the International Organization for Standardization. It is a way of sub-dividing a communications system into smaller parts called layers. A layer is a collection of conceptually similar functions that provide services to the layer above it and receives services from the layer below.

This model is widely used in designing network architectures on a global scale. A model starts with the physical layer and ends with the application layer that provides mechanisms such as the HTTP protocol.

For example, in the Internet stack, the Internet protocol components IP and TCP are at levels 3 and 4.

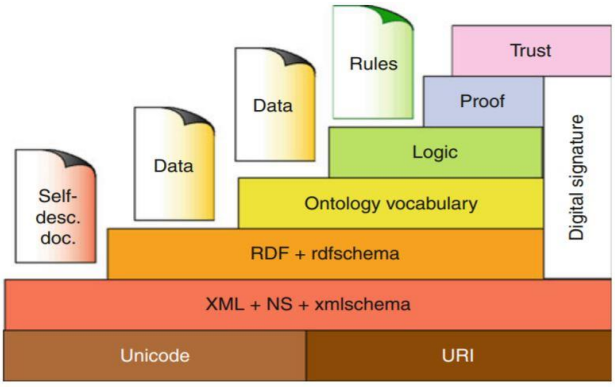

Fig. 2. The Tower of Babel to structure the Semantic Web

Tim Berners-Lee started a similar conceptual effort to structure the Semantic Web as Fig. 2 shows.

At the lowest level, Unicode is seen as a means to encode text, URIs to refer to resources, and XML with its namespace and schema mechanisms to provide syntactic descriptions of structured objects.

On top of this, he envisioned five layers of semantics: RDF, OWL, RIF, and layers for proof and trust.

This type of layering has two major functions: preventing an upper layer from re-implementing functionality provided by a layer below and allowing an application that only understands a lower layer to at least interpret portions of definitions at a higher layer.

1.6 Semantic Web as a Database

The Semantic Web as a research area saw the coming together of a number of communities including Artificial Intelligence (from agents, knowledge modeling, and logic) and the Web. For the most part, though, the research overlap between the Semantic Web and databases was minimal. This could be seen as somewhat surprising as the Web of Data is now a widely used term, but, in the early days, the emphasis was on creating knowledge structures as a platform for agents.

How can we use it as database? The main research issues that are currently beginning to emerge include the following:

- Which particular database techniques (e.g., partitioned hashes, column tables) are most applicable to high-performance RDF storage?
- How to structure benchmarks for large-scale repositories? Including what are the correct dimensions?

- When and where should reasoning be handled? For example, materialization (the precomputation and storage of inferred triples) is an expensive process which may not contribute to desired results.

2 Semantic data storages

The storage and retrieval of information are important functions of information systems (IS). These IS functions have been realized for decades, due to the maturity of the relational database technology. In recent years, the concept of Semantic Information System (SIS) has emerged as IS in which information is represented with explicit semantic based on its meaning rather than its syntax to enable its automatic and intelligent processing by computers.

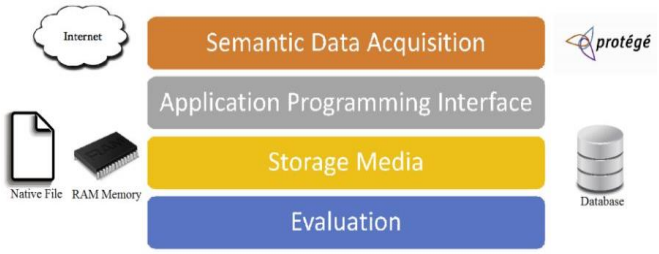

Fig. 3. the framework developed in the suggestion phase of the Design research method

Fig. 3. presents the framework developed in the suggestion phase of the Design research method to analyze the semantic data storage approaches.

- The first layer of the framework is the Semantic Data Acquisition layer. The purpose of this layer is to acquire the ontologies or semantic data that will be used by the other layers.
- The Semantic Data Acquisition layer of the Framework in Figure acquires existing ontologies that have been developed and made available publicly on the internet.
- The second layer is the Application Programming Interface (API) layer; it is used to create, edit, browse and delete ontology or semantic data. It is also used to load existing semantic data.
- The third layer which is the Storage Media, is used to physically store the semantic data in the computer memory.
- Evaluation is the last layer; it analyses and discusses the underlying structures used to store semantic data.

2. 1 Approaches for Storing Semantic Data

Three approaches are used to store ontology or semantic data, namely, in-memory, native or file systems and databases.

In the in-memory approach, the computer's central memory is used to store semantic data.

The advantage of this approach is that it provides quick query response times with small-scale semantic data.

The main drawbacks of this approach are that larger semantic data are difficult to process and the stored data are not kept permanently. In fact, in this approach, the semantic data need to be loaded in the computer memory on demand; which is inefficient and time-consuming.

The native storage approach uses files to store semantic data; this enables fast loading and query of semantic data. Processing large-scale semantic data is one of the main drawbacks of the native storage approach.

Furthermore, functionalities such as query optimization, data recovery, transaction processing, and controlled access need to be implemented separately; fortunately, these drawbacks are addressed with the database storage approach. In fact, relational databases (RDB) remain the appropriate media for storying semantic data due to the maturity of relational database technology.

The database storage of semantic data offers many functionalities including storage, query, reasoning and scalability. Two approaches are used to store semantic data in databases: generic and specific schema.

In the generic schema approach, a table is used to store semantic data in RDB; the columns of the table are the elements of RDF statements of the ontology. An improved version of the generic schema approach is called normalized triple store; it uses two more tables to store semantic data with the purpose of making join queries less expensive.

2.2 Software Platforms for Semantic Data Storage

To enable the storage and query of semantic data, several platforms have been developed. The most popular of these platforms are: AllegroGraph, Jena, Open Anzo, Minerva and Sesame (Fig. 4.).

Platform	License	Operating system	Type of Storage
AllegroGraph	Commercial/Free	Linux	Native
Jena	Free/Open Source	Windows/Linux	Memory, Native, RDB
Sesame	Free/ Open Source	Windows/Linux	Memory, Native, RDB
Open Anzo	Free/ Open Source	Windows Linux	RDB
Minerva	Free	Windows/Linux	RDB

Fig. 4. Software Platforms for Semantic Data Storage

AllegroGaph is a server application that is accessed remotely by client applications. It enables the storage and query of semantic data and provides an API for the direct access to these data without any use of queries.

Minerva is a component of the Integrated Ontology Development Toolkit; it is used as a library in Eclipse Integrated Development Environment (IDE) to store semantic data.

Open Anzo was developed by IBM; it can be used in three different modes to store and query semantic data: (1) embedded in an application, (2) installed as a server application and accessed remotely by clients or (3) run locally.

Jena API is integrated into Eclipse IDE as a library; it enables the creation and storage of semantic data in different formats.

Sesame is a Software Development Kit (SDK) that was developed in the European IST project On-to-Knowledge. It enables semantic data to be queried or exported.

3 Ontology

3.1 Ontology in Philosophy

Ontology is the branch of philosophy that studies concepts such as existence, being, becoming, and reality as Fig. 5 shows.

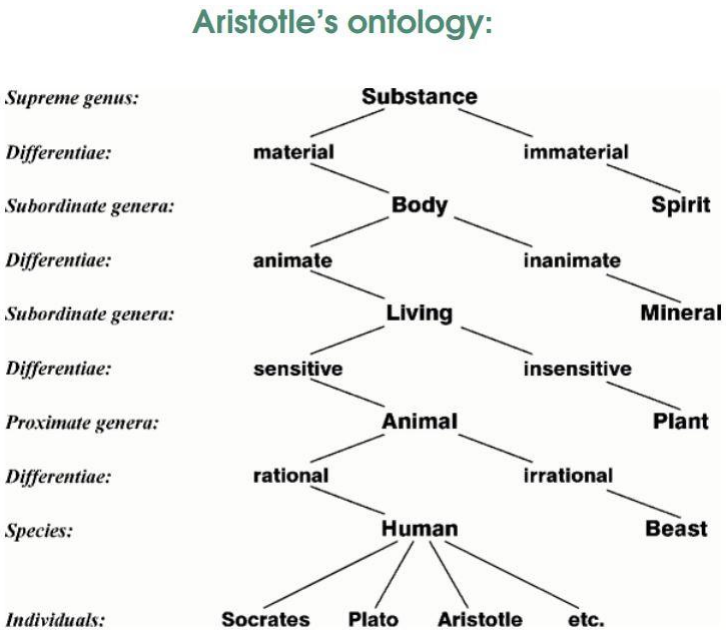

Fig. 5. Aristotle's ontology

It includes the questions of how entities are grouped into basic categories and which of these entities exist on the most fundamental level. Ontology is sometimes referred to as the science of being and belongs to the major branch of philosophy known as metaphysics.

3.2 Ontology in Computer Science

In computer science, an ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them. To enable such a description, we need to formally specify components such as individuals (instances of objects), classes, attributes and relations as well as restrictions, rules and axioms. As a result, ontologies do not only introduce a sharable and reusable knowledge representation but can also add new knowledge about the domain.

Of course, there are other methods that use formal specifications for knowledge representation such as vocabularies, taxonomies, thesauri, topic maps and logical models. However, unlike taxonomies or relational database schemas, for example, ontologies express relationships and enable users to link multiple concepts to other concepts in a variety of ways.

Since ontologies define the terms used to describe and represent an area of knowledge, they are used in many applications to capture relationships and boost knowledge management.

The adoption of ontologies helps early hypotheses testing in Pharma by categorizing identified explicit relationships to a causality relation ontology. Ontologies also enrich semantic web mining, mining health records for insights, fraud detection and semantic publishing.

3.3 Semantic Technologies at the BBC

As an example of the application of ontology, The BBC ontology is successful.

BBC Web-based service is one of the most visited websites and the world's largest news website. As of 2007, it contained over two million pages

Focus has been on separate, standalone HTML microsites that are not linked together and to other data sources on the Web. It is difficult to find everything BBC has published about any given object. We cannot navigate from a page about a musician to a page with all the programmes that have played that artist, to their biography, etc.

By using ontologies and an enterprise-ready knowledge base with the power of inference, the BBC wanted to minimize expensive editorial management of content assets. They also wanted to have a website navigation guided by what was important to the consumer (e.g., teams, countries, players, etc.). As a result, the BBC expected to see an increase in content aggregation, re-use and re-purposing without additional costs.

Thus ontology technique is used in BBC, This site provides access to the ontologies the BBC is using to support its audience facing applications such as BBC Sport, BBC Education, BBC Music, News projects and more. These ontologies form the basis of our Linked Data Platform.

The ontologies are built incrementally according to current business requirements. They are all expected to evolve as our requirements evolve. The BBC produces a plethora of rich and diverse content about the things that matter to our audiences. Linked Data gives us an opportunity to connect content together through those topics.

They use ontologies to describe the world around us, content the BBC creates, and the management, storage and sharing of these data within the Linked Data Platform.

Semantic Technologies at the BBC – Sport Ontology.

The Sport Ontology is a simple lightweight ontology for publishing data about competitive sports events. The terms in this ontology allow data to be published about:

The structure of sports tournaments as a series of events, the competing of agents in a competition, the type of discipline an event involves, the award associated with the competition and how received it...etc

Whilst it originates in a specific BBC use case, the Sport Ontology should be applicable to a wide range of competitive sporting events data publishing use cases. Care has been taken to try and ensure interoperability with more general ontologies in use. In particular, it draws heavily upon the ontology of the event.

4 Knowledge base

A knowledge base is a collection of interlinked descriptions of entities (real-world objects, events, situations or abstract concepts) interlinked in a way that enables storage, analysis and reuse of this knowledge in a machine-interpretable way.

Most people are familiar with traditional, relational databases. There are cells and tables filled with letters and numbers. Years of refinements and optimizations have ensured that organizations can manage phenomenal amounts of data. But as the American author Clifford Stoll said it best: Data is not information, information is not knowledge.

Knowledge bases abstract away from a simple database to create an organized collection of data that is closer to how the human brain organizes information. Knowledge bases add a semantic model to the data, which includes a formal classification with classes, subclasses, relationships and instances (ontologies and dictionaries), on one hand, and rules for interpreting the data, on the other.

The difference between a database and a knowledge base is that a database is a collection of data representing facts in their basic form, while a knowledge base stores information as answers to questions or solutions to problems. A knowledge base allows for rapid search, retrieval, and reuse. Information in a knowledge base is typically fully developed and ready to be applied.

5 NoSQL Database

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.

NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive which is shown in Fig. 6.

The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

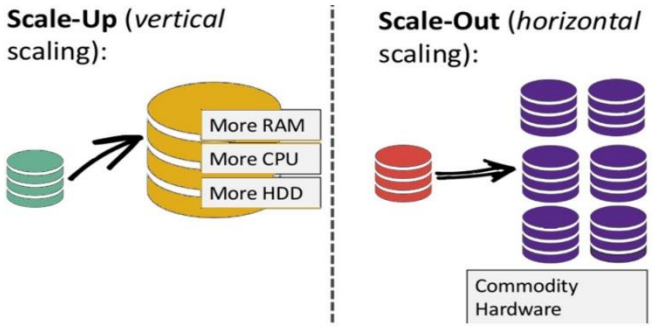

Fig. 6. "scale up" and "scaling out".

5.1 CAP theorem

CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees: Consistency, Availability, Partition Tolerance

Consistency

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability.

The database should always be available and responsive. It should not have any downtime.

Partition Tolerance.

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

The classification of NoSQL systems as either AP, CP or CA vaguely reflects the individual systems' capabilities and hence is widely accepted as a means for high-level comparisons. However, it is important to note that the CAP Theorem actually does not state anything on normal operation; it merely tells us whether a system favors availability or consistency in the face of a network partition.

5.2 NoSQL databases

Many NoSQL databases were designed by young technology companies like Google, Amazon, Yahoo, and Facebook to provide more effective ways to store content or process data for huge websites. Some of the most popular NoSQL databases include the following:

- Apache CouchDB, an open source, JSON document-based database that uses JavaScript as its query language.
- Apache Cassandra, an open source, wide-column store database designed to manage large amounts of data across multiple servers and clustering that spans multiple data centers.
- MongoDB, an open source document-based database that uses JSON-like documents and schema, and is the database component of the MEAN stack
- Redis, a powerful in-memory key value store used for session caching, message queues, and other specific applications.
- Elasticsearch, a document-based database that includes a full-text search engine.