CHAPTER 8

Big Data Applications

The sustainability of huge and ever-growing data pools using different formats that cannot be processed with traditional software tools is the next big challenge for web designers, Internet marketers, and software engineers and requires new technologies and practices. One of the approaches to cope with Big Data is to use Semantic Web technologies, especially machine-interpretable metadata and Linked Data. Implementing the Resource Description Framework (RDF) and RDF-based standards ensures that data and its meaning are encapsulated, and concepts and relationships can be managed while connecting diverse data from various data sources. Graph representations, such as Facebook's Open Graph, add context to and visualize Big Data for data analysis. The Service-Oriented Architecture (SOA) infrastructure over Big Data makes it possible to update Big Data in real time. Data can be automatically classified, relationships associated, and new relationships found, so that data can be collected and integrated without worrying about schemas and data descriptions, yet providing a data description. Big Data applications on the Semantic Web include, but are not limited to, next-generation Search Engine Result Pages, social media graphs, analysis of natural language content, publishing factual data about massive world events, interlinking BBC's online content, as well as high-performance data storage and processing.

Big Semantic Data: Big Data on the Semantic Web

Big Data refers to any high-volume, high-velocity datasets too large and complex to process using traditional data processing tools, applications, and database systems. Representing petabytes of data, such datasets store billions of hidden values unavailable for efficient and automatic machine processing. Big Data is characterized by four *V*s:

- **Volume**: Huge amounts of data stored in, and retrieved from, massive datasets. The challenge is to achieve a reasonable processing speed, especially in real-time applications.
- **Velocity**: High-rate data flow. The challenge is the streaming data processing.
- **Variety**: Different forms of data. The challenge is to deal with the different data structures, data formats, and serializations.
- **Veracity**: Uncertainty of data. The challenge is to handle trust issues, determine accuracy, and cope with poor data quality.

One of the promising approaches to address the issues associated with Big Data is to implement Semantic Web technologies in order to build systems that can efficiently handle Big Data and evolve with the growing data processing needs.

Google Knowledge Graph and Knowledge Vault

One of the best known Big Data applications on the Semantic Web is the *Google Knowledge Graph*, which was introduced in 2012. The Google Knowledge Graph is a semantic knowledge base to enhance traditional Search Engine Result Pages (SERPs) with semantic search information gathered from a wide variety of sources. The data sources used by the Knowledge Graph include pages indexed by Google, objects on GoogleMaps, public data sources such as Wikipedia, LOD datasets such as DBpedia, the CIA World Factbook, and the FDA datasets, as well as subject-specific resources such as Weather Underground and World Bank, for meteorological information and economic statistics, respectively. The result of a Knowledge Graph search is not only relevant information far more accurate that what you would find with traditional searches but also related information, such as similar resources people search for the most. For example, if you search for Leonardo da Vinci, you not only get facts about him and his famous works like *Mona Lisa* and *The Last Supper*, but Google will also suggest other notable painters of the same era, such as Jan van Eyck, Dürer, Raphael, and Michelangelo (see Figure 8-1).

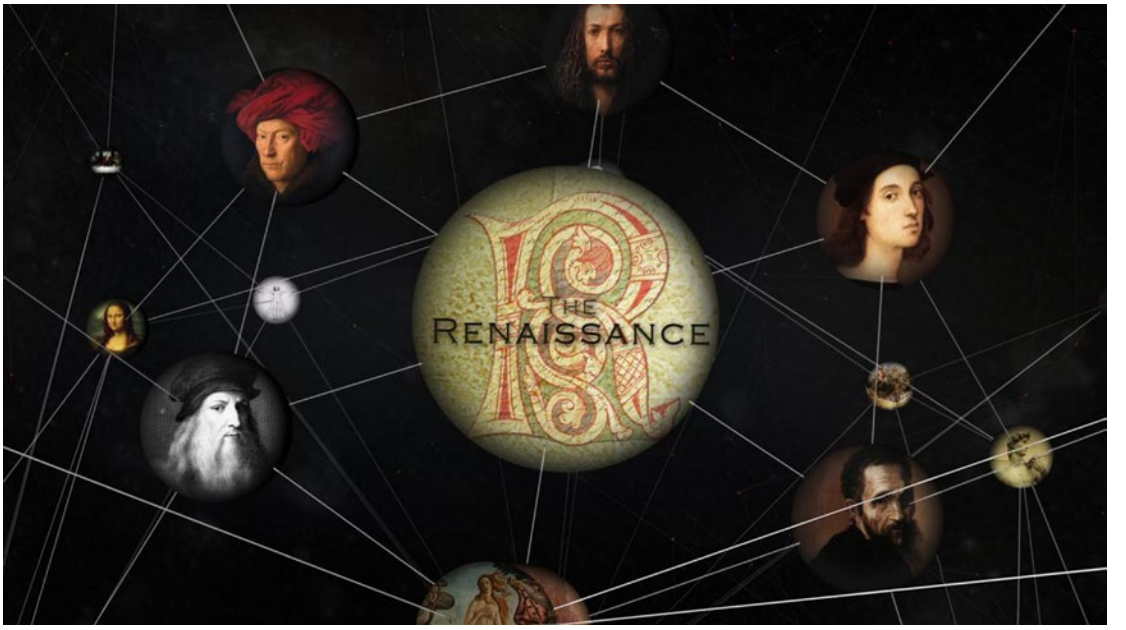

Figure 8-1. *The Google Knowledge Graph finds data resources related to your search phrase [1]*

Similarly, if searching for the title of an action movie, the results will include similar movies, while searching for a particular inventor will disclose additional inventors, with similar research fields and awards. The Knowledge Graph contains more than half a billion objects and over 18 billion facts about relationships between different objects that help software agents "understand" the meaning of the search keywords, and these figures are constantly growing.

Depending on the search phrase used, the search results retrieved from the Google Knowledge Graph are represented in two ways. The first one, called the *Google Knowledge Panel*, is displayed on the right-hand side of the Search Engine Result Pages, next to the organic search results. Searching for persons or brand names typically results in a Knowledge Panel, as shown in Figure 8-2.

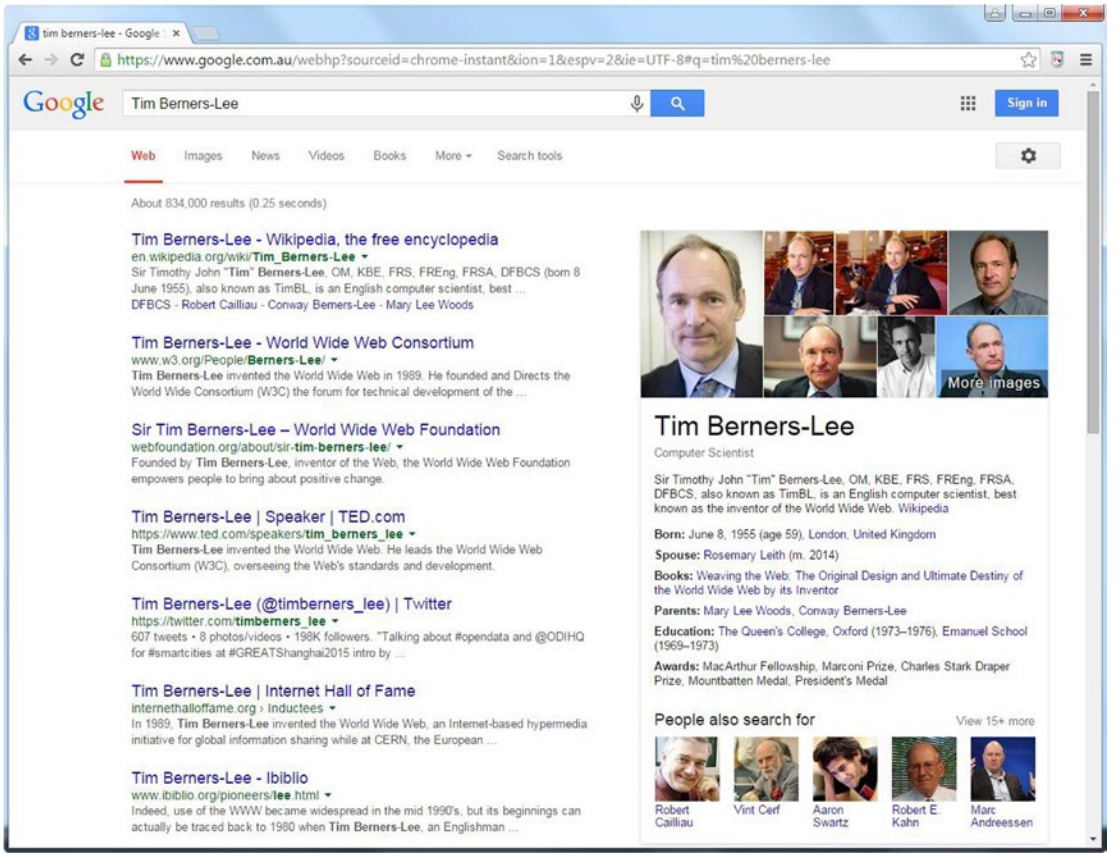

Figure 8-2. *Facts about Tim Berners-Lee shown by the Google Knowledge Panel*

If the object has a social media presence on Facebook, YouTube, Twitter, Instagram, Google+, etc., links to those pages will also be displayed. The most related links are shown under "People also search for," which can be extended by clicking the "View more" link. If you search for a living musician, you might also see "Upcoming events" on the Knowledge Panel, containing the venue and date of upcoming concerts.

The second type of data representation for data retrieved from the Google Knowledge Graph is the Google Knowledge Carousel, which shows entities related to the search phrase. For instance, if you search for Clint Eastwood's filmography, Google will provide the organic results, a Knowledge Panel on Clint Eastwood, as well as a Knowledge Carousel about his most famous movies, such as *The Good, the Bad and the Ugly*, *Unforgiven*, *The Outlaw Josey Wales*, *Gran Torino*, *A Fistful of Dollars*, *Dirty Harry*, and so on (see Figure 8-3). The Carousel is also used when you click certain link types on the Knowledge Panel. For example, if you click an upcoming concert of a musician displayed on the Knowledge Panel, all the upcoming concerts of the musician will be shown on a Carousel at the top.

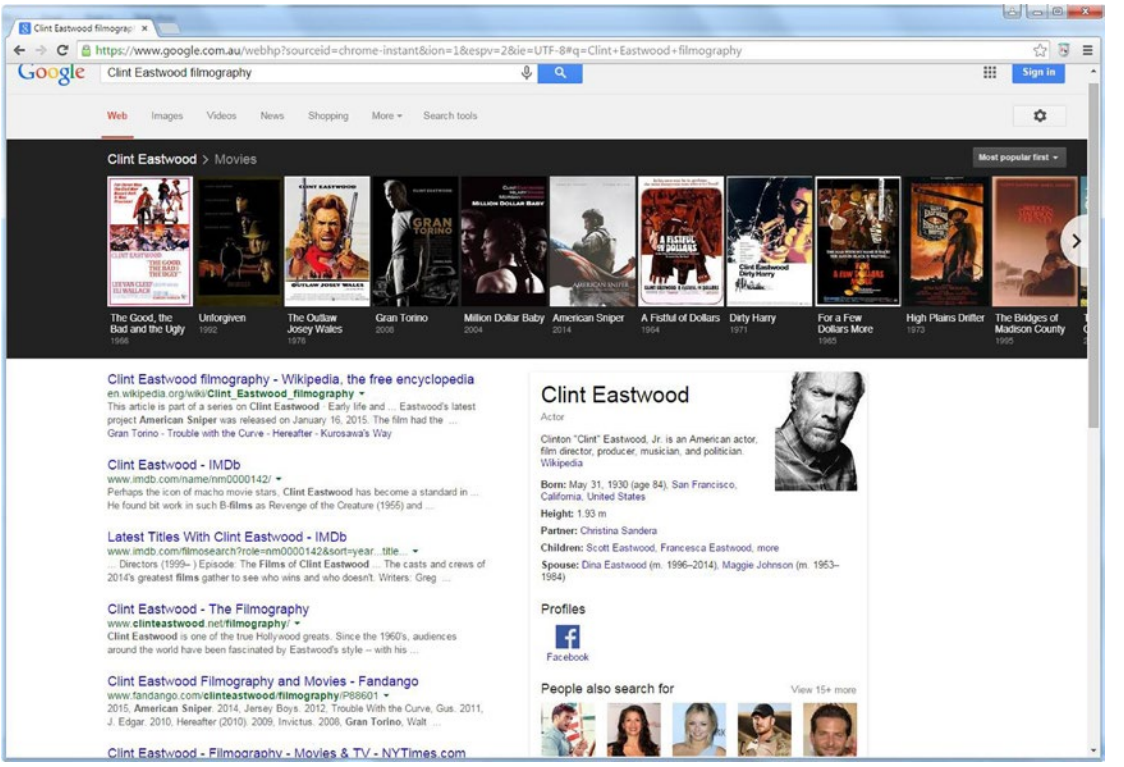

Figure 8-3. *Searching for an actor's filmography gives both a Knowledge Carosel and a Knowledge Panel*

The *Google Knowledge Vault* combines data from conventional web sites, including unstructured text, DOM trees, and tables, as well as structured data from Freebase. It is a large database of automatically extracted structured data. The amount of information users can retrieve depends on the structure and correctness of the queries.

The Knowledge Vault derives much of its data from the Knowledge Graph and the sources thereof, as well as harvesting its own data, ranking its reliability and compiling all results into a database of over 1.6 billion facts collected by machine learning algorithms. There are no more ambiguous natural language queries about Jaguar (car make or animal) or Taj Mahal (monument, musician, or casino), as Google knows exactly what is the difference between these "things."

Get Your Company, Products, and Events into the Knowledge Graph

If you describe your company, products, services, or events on your web site using controlled vocabulary terms, either as HTML5 Microdata or JSON-LD, they will be considered to be included in the Google Knowledge Graph. The Schema.org terms can be used to define features and relationships such as the surname of a person (`http://schema.org/familyName`), the genre of a music album (`http://schema.org/music/artist/album`), or the opening hours of a store (`http://schema.org/openingHours`). The more precise category you use, the better. For example, if you have a concert, use `http://schema.org/MusicEvent` rather than `http://schema.org/Event`, or if you have a soccer match, use `http://schema.org/SportsEvent` instead of `http://schema.org/Event`. Event organizers, for example, can describe upcoming events by

adding structured data to the markup as a separate code block in JSON-LD, so that Google might include them on the Knowledge Graph (see Listing 8-1). The vocabulary is defined using `@context` and `@type`, as discussed earlier, in Chapter 3.

Listing 8-1. JSON-LD Annotation of a Band in the Markup

```
<script type="application/ld+json">
  {
    "@context"  : "http://schema.org",
    "@type"     : "MusicEvent",
    "name"      : "Nice Band Live",
    "startDate" : "2015-09-18T20:00",
    "url"       : "http://www.nicebandexample.com/tour/150918",
    "location"  : {
                    "@type"   : "Place",
                    "name"    : "The Oval",
                    "address" : "1234 Blackwood Plaza",
                    "sameAs"  : "http://www.xyzoval.com"
                  },
    "performer" : {
                    "@type"   : "MusicGroup",
                    "name"    : "Nice Band",
                    "sameAs"  : "http://www.nicebandexample.com"
                  },
    "offers"    : {
                    "@type" : "Offer",
                    "url"   : "http://www.exampleticketseller.com"
                  }
  }
</script>
```

Similarly, online retailers and shops can describe products using Schema.org terms. To identify which properties you can use, go to http://schema.org/Book and check the property list. Here, we added the web site of the book with `url`, defined the author by referring to the corresponding DBpedia page, declared the available formats (paperback and e-book) using Schema.org terms, among other properties (see Listing 8-2).

Listing 8-2. JSON-LD Annotation of a Product Description

```
<script type="application/ld+json">
  {
    "@context": "http://schema.org",
    "@type": "Book",
    "url": "http://www.lesliesikos.com/web-standards-mastering-html5-css3-and-xml-second- ↩
     edition/",
    "author": "http://dbpedia.org/resource/Leslie_Sikos",
    "bookFormat": "http://schema.org/Paperback",
    "bookFormat": "http://schema.org/EBook",
    "datePublished": "2014-12-24",
    "image": "http://www.lesliesikos.com/img/web-design-book.jpg",
    "inLanguage": "English",
    "isbn": "1484208846",
```

```
    "name": "Web Standards: Mastering HTML5, CSS3, and XML",
    "numberOfPages": "524",
    "offers": {
      "@type": "Offer",
      "availability": "http://schema.org/InStock",
       "price": "39.89",
       "priceCurrency": "USD"
      },
    "publisher": "http://dbpedia.org/resource/Apress",
    "about": "http://dbpedia.org/resource/Web_design"
  }
</script>
```

Product offerings can be annotated using GoodRelations, covering properties such as the web page of the advertisement on eBay or Gumtree, the accepted payment methods, the item price and currency, the category of the product, the official vendor web page describing the product, and the description of the product (see Listing 8-3).

Listing 8-3. JSON-LD Annotation of a Product Offering in the Markup

```
{
  "@context": {
    "gr": "http://purl.org/goodrelations/v1#",
    "pto": "http://www.productontology.org/id/",
    "schema": "http://schema.org/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "schema:url": {
      "@type": "@id"
    },
    "gr:acceptedPaymentMethods": {
      "@type": "@id"
    },
    "gr:hasBusinessFunction": {
      "@type": "@id"
    },
    "gr:hasCurrencyValue": {
      "@type": "xsd:float"
    }
  },
  "@id": "http://www.ebay.com/itm/ExampleAd-Giant-TCR-Advanced-1-Road-Bike-/21621444051",
  "@type": "gr:Offering",
  "gr:acceptedPaymentMethods": "gr:Cash",
  "gr:description": "Want to sell my Giant TCR Advanced 1 Road Bike as I'm moving ➥
   interstate",
  "gr:hasBusinessFunction": "gr:Sell",
  "gr:hasPriceSpecification": {
    "gr:hasCurrency": "USD",
    "gr:hasCurrencyValue": "1350"
  },
```

```
  "gr:includes": {
    "@type": [
      "gr:Individual",
      "pto:Racing_bicycle"
    ],
    "gr:name": "Giant TCR Advanced 1",
    "schema:url": "https://www.giant-bicycles.com/enus/bikes/model/tcr.advanced.1.force/ ↩
     14797/66271/"
  },
  "gr:name": "Used Giant Road Bike"
}
```

To add structured data to your HTML5 markup about your local business, you can use the LocalBusiness vocabulary from Schema.org. Make sure that you use the most specific type for your business (http://schema.org/Library for libraries, http://schema.org/ShoppingCenter for shopping centers, http://schema.org/AutomotiveBusiness for garages, http://schema.org/FinancialService for financial planners and banks, etc.). A school or a sports club should use http://schema.org/Organization instead, while http://schema.org/Corporation is more suitable for enterprises. The most commonly used LocalBusiness properties are name, description, address, and telephone (see Listing 8-4). The physical address details can be embedded to PostalAddress.

Listing 8-4. LocalBusiness Annotated with Microdata

```
<div itemscope="itemscope" itemtype="http://schema.org/LocalBusiness">
  <h1><span itemprop="name">The Blue Cafe</span></h1>
  <span itemprop="description">A nice cafe on the beach with a friendly atmosphere.</span>
  <div itemprop="address" itemscope="itemscope" itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">123 Esplanade</span>
    <span itemprop="addressLocality">Nice Beach</span>,
    <span itemprop="addressRegion">CA</span>
  </div>
  <p>
   Phone: <span itemprop="telephone">123-456-7890</span>
  </p>
</div>
```

Depending on what you want to display as human-readable data on your site and what you want to add as machine-readable only, you can use different markup elements and style sheets. For those data you add for software agents only, use the attribute values on the `meta` element.

Social Media Applications

Excellent examples for Big Data implementations on the Semantic Web are the social media graphs, such as the Facebook Social Graph, the Twitter Interest Graph, the Twitter Follow Graph, the LinkedIn Professional Graph, or the LinkedIn Economic Graph.

Facebook Social Graph

The Facebook Social Graph is the largest social graph in the world, containing tens of petabytes of structured data about approximately 1 billion users. Because every object is a graph node, and every relationship is a graph edge on the Facebook Social Graph (see Figure 8-4), any object can easily be accessed directly in the browser as a user and programmatically from Facebook apps.

Figure 8-4. *On the Facebook Social Graph, every object is a node and every connection is an edge*

In fact, the easy access of this vast user data is exploited well beyond Facebook, as the social connections and links of the Facebook Social Graph are also used by other social networking portals, such as Pinterest and Last.fm (*social bootstrapping*).

Have you ever wondered how Facebook recommends friends? Using the edges of the Facebook Social Graph, it is straightforward to identify those people who have at least one friend in common (see Figure 8-5).

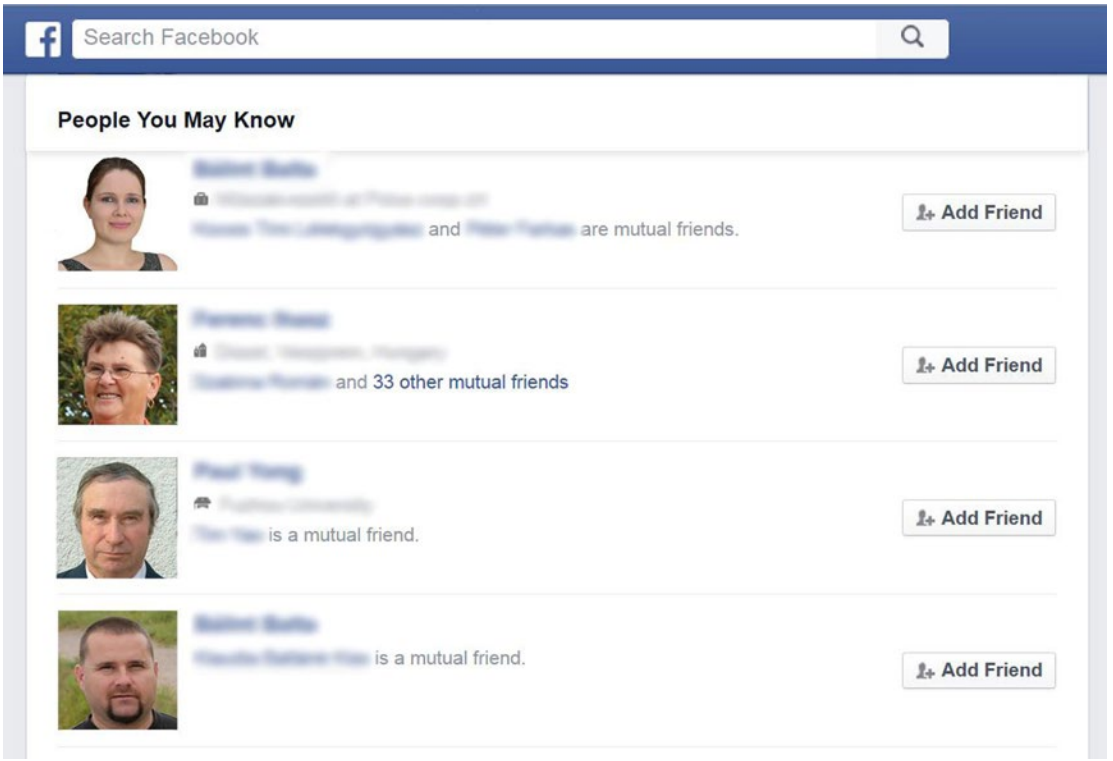

Figure 8-5. *The edges of the Facebook Social Graph make it possible to suggest people you may know*

The Facebook Graph API

The *Facebook Graph API* is the core of the Facebook Platform, enabling developers to read data from and write data into Facebook user profiles. The Graph API represents the current state of the Facebook Social Graph through graph objects such as people, photos, events, and pages, as well as the connections between them, such as friend relationships, shared content, and photo tags. In other words, the Graph API makes it possible to programmatically access user objects and connections from the Facebook Social Graph, which can be used for Facebook apps.

The Graph API can not only query data but also post new stories, publish Open Graph stories, read information about a Facebook user, upload photos, update information in the Social Graph, and perform similar tasks used by Facebook apps. All the objects of the Facebook Social Graph (users, photo albums, photos, status messages, pages, etc.) have a unique identifier, which is a positive integer and makes it possible to refer to any node or edge.

Originally, the Graph API provided data to applications exclusively in JSON. The two different key/value pair sets of JSON are the objects (where keys are strings) and arrays (which represent the set of keys as a finite, counting sequence of nonnegative integers). The values can be JSON objects, arrays, or primitives (strings, numbers, Boolean values, and null).

Because the Graph API is a RESTful JSON API, you can access it in your browser. The web interface of the Graph API is called the *Graph API Explorer*, which is available at `https://developers.facebook.com/tools/explorer/`. With this tool, you can use and traverse the Facebook Social Graph. You have to have a Facebook account and log in to use the Graph API Explorer. Once you are logged in and visit the Graph API Explorer, you can see a JSON object on the right, with two properties, the identifier and the name of the

current user, because these are the two fields selected by default (displayed under the `me` node on the left). If you unselect the two check boxes, and click Submit, the Graph API Explorer will reveal more information about the user (see Figure 8-6). How much detail is provided depends on your privacy settings. These field values are the default data to be returned when no fields are specified in your query.

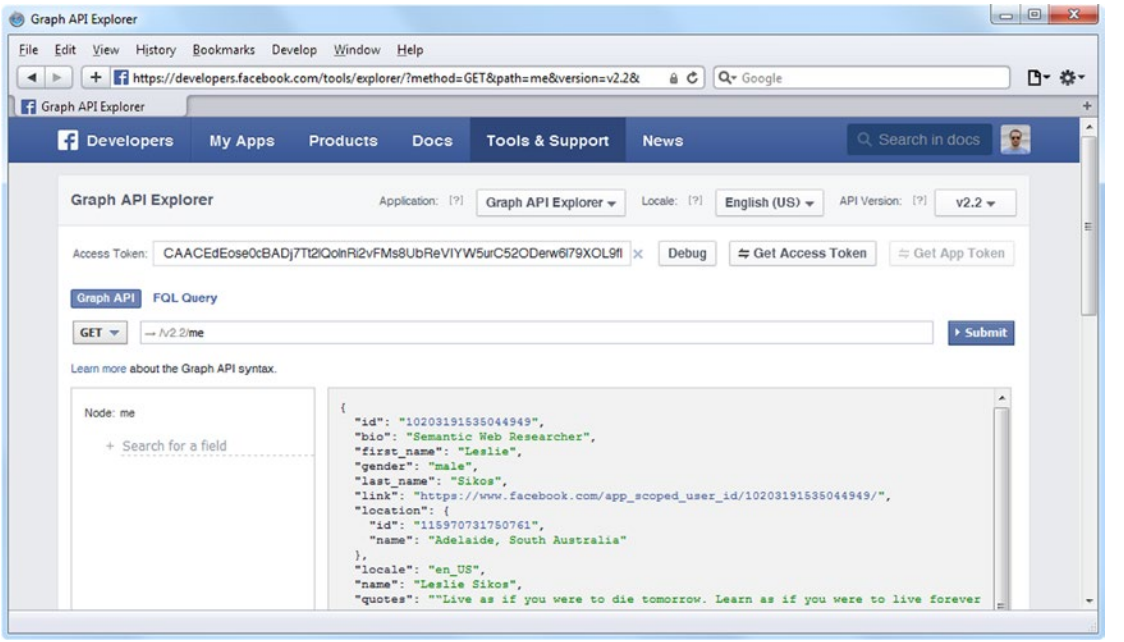

Figure 8-6. *With the Graph API Explorer, you can access fields of a node in the Facebook Social Graph*

If you need more data about the node, you might have to generate an access token (Get Access Token), select the additional fields of your choice, and give permission to the Graph API to access those data. The Graph API Explorer performs simple HTTP `GET` requests in the background, as demonstrated in Listing 8-5, and provides a drop-down on the left, with the `GET`, `POST`, and `DELETE` choices. The default value is `GET`. To initiate a new request, you have to click Submit on the right.

Listing 8-5. HTTP `GET` Request Through the Graph API Explorer

```
GET /v2.2/me HTTP/1.1
Host: graph.facebook.com
```

While every node in the Facebook Social Graph has an identifier, you can refer to any node either by the numeric ID or the username (if the object has one). For example, if you change the request after `GET` from the default `me` value to your user ID or your username and hit Submit, you get the same results for your queries. This works even for Facebook pages.

Due to the HTTP `GET` requests used under the hood, every query can also be performed directly. For instance, to get information about the Facebook page of this book (`http://facebook.com/SemanticWebBook`), you can directly open `http://graph.facebook.com/SemanticWebBook` in your browser to retrieve the JSON output. This makes it possible to access any node or edge of the Facebook Social Graph programmatically, with software libraries that handle HTTP requests and the JSON format. To make it even easier, Facebook provides SDKs for popular languages and platforms such as PHP (see Listing 8-6), JavaScript (see Listing 8-7), iOS (see Listing 8-8), and Android (see Listing 8-9).

Listing 8-6. Make an API Call from PHP

```
$request = new FacebookRequest(
  $session,
  'GET',
  '/me'
);
$response = $request->execute();
$graphObject = $response->getGraphObject();
/* result handler */
```

Listing 8-7. Make an API Call from JavaScript

```
FB.api(
    "/me",
    function (response) {
      if (response && !response.error) {
        /* result handler */
      }
    }
);
```

Listing 8-8. Make an API Call from iOS

```
[FBRequestConnection startWithGraphPath:@"/me"
                    completionHandler:^(
                        FBRequestConnection *connection,
                        id result,
                        NSError *error
                    ) {
                        /* result handler */
                    }];
```

Listing 8-9. Make an API Call from Android

```
new Request(
    session,
    "/me",
    null,
    HttpMethod.GET,
    new Request.Callback() {
        public void onCompleted(Response response) {
            /* result handler */
        }
    }
).executeAsync();
```

Since 2011, Facebook provides data retrieved from the Social Graph, not only in JSON, but also in a semantically enriched RDF serialization format, to include Linked Data URIs. The implementation is meant to be flexible and robust, so the Turtle format has been chosen, although JSON-LD has also been considered. The JSON output to Turtle translation is accessible via HTTP content negotiation. A URI or blank node is assigned to a JSON object or array as the subject of RDF triples. The predicate and object of RDF triples are derived from the key-value pairs of JSON objects and arrays. The JSON key is translated into a URI, while the value is converted to an RDF term, such as a meaningful literal, a URI, or a blank node.

The primitive values correspond to RDF literals, and the conversion is made by applying heuristics to determine the most suitable RDF datatype URI for literals. The JSON strings that form URIs are translated to URIs. The most frequently used datatype URIs of the JSON-Turtle conversions are `xsd:Boolean`, `xsd:dateTime`, `xsd:decimal`, `xsd:double`, and `xsd:integer`. The object identifiers remain strings, even if they seem to be integers. Facebook implemented the RDF 1.1 convention for strings, namely that regular strings are left as plain literals (implicitly handled as `xsd:string`) rather than explicitly typing them as `xsd:string`. To comply with httpRange-14 (to ensure that HTTP `GET` requests won't lead to an undefined domain), fragment identifiers are preferred over slash URIs. Because the output represents an isolated graph that is not connected to external resources, the resulting Linked Data is four-star Linked Data only (see Chapter 3). Still, the RDF/Turtle output is semantically richer than the JSON output, due to explicit semantics accessible as ontologies utilizing the RDFS and OWL vocabularies.

The Linked Data can be accessed the same way you perform an HTTP `GET` request directly, i.e., using the `http://graph.facebook.com` or `https://graph.facebook.com` base URI, followed by a slash and the Facebook username or Facebook page name. As an example, the Linked Data URIs can be used to augment a person's FOAF profile, as shown in Listing 8-10.

Listing 8-10. FOAF Profile Augmentation

```
@base <http://graph.facebook.com/> .
<http://www.lesliesikos.com/datasets/sikos.rdf#sikos>
owl:sameAs </1105249544#> ;
rdfs:seeAlso </1105249544?metadata=1> ;
foaf:depiction </1105249544/picture> ;
foaf:account <http://www.facebook.com/sikos> .
```

Facebook Module of Apache Marmotta's LDClient Library

The Facebook Module of Apache Marmotta's LDClient library represents the Facebook Graph API objects and connections as RDF triples using Schema.org, Dublin Core, FOAF, SIOC, and SKOS terms. Whenever feasible, the mapping will use Schema.org terms. The Facebook Module of Marmotta (ldclient-provider-facebook) registers an endpoint to handle all URIs starting with `http://graph.facebook.com` and `http://www.facebook.com`. The Facebook Module can be used in your Apache Maven project by adding a dependency, as demonstrated in Listing 8-11.

Listing 8-11. Dependency for the Facebook Module in Maven

```
<dependency>
  <groupId>org.apache.marmotta</groupId>
  <artifactId>ldclient-provider-facebook</artifactId>
  <version>3.3.0</version>
</dependency>
```

Each Facebook object has at least a category (mapped to Schema.org terms using `rdf:type` whenever possible), an identifier (mapped to `dcterms:id`), a name (mapped to `schema:name`), a description (mapped to `schema:description`), and a Facebook web page (mapped to `foaf:homepage`). This allows programmatic access to Facebook objects from Maven projects.

Facebook Open Graph Protocol

Inspired by Microformats and RDFa, the *Open Graph Protocol* makes it possible for developers to integrate their pages into the Facebook Social Graph.

The Open Graph vocabulary can be used to annotate your web site markup with metadata that can be associated with Facebook properties. For example, a book can be described as shown in Listing 8-12. The namespace of Open Graph is http://opengraphprotocol.org/schema/.

Listing 8-12. Open Graph Annotation in the Markup

```
<meta property="og:title" content="Web Standards: Mastering HTML5, CSS3, and XML" />
<meta property="og:type" content="book" />
<meta property="og:url" content="http://www.masteringhtml5css3.com" />
<meta property="og:image" content="http://www.masteringhtml5css3.com/img/ ↩
 webstandardsbook.jpg " />
<meta property="og:site_name" content="Web Site of the Book Web Standards: ↩
 Mastering HTML5, CSS3, and XML" />
<meta property="og:description" content="A book describing web standardization to create ↩
 optimized, device-independent web sites with cutting-edge technologies." />
```

Twitter Cards

Similar to Facebook's Open Graph annotation, Twitter provides the so-called *Twitter Cards* annotation to add structured data to your markup about Twitter objects [2]. Because the Twitter Cards are based on the same conventions as the Open Graph protocol, the tags are similar, and you can generate a Twitter Card intermixing Open Graph and Twitter Card annotation without tag or data duplication. While it is recommended that users specify the og RDFa Core 1.1 CURIE prefix mapping for Open Graph on the `html` element (`<html prefix="og: http://ogp.me/ns#">`), Twitter Cards do not require similar markup; however, they can use the `twitter:` prefix as the value of the `name` attribute of the `meta` element. Another difference is that while the Open Graph protocol specifies the use of property and content attributes for markup (e.g., `<meta property="og:image" content="http://example.com/ogimg.jpg"/>`), Twitter Cards use `name` and `content`. The parser of Twitter will understand the `property` and `content` attributes of existing Open Graph markup. To define a summary card (the default Twitter Card type), you can mix Twitter Card and Open Graph annotation on the `meta` element, as shown in Listing 8-13.

Listing 8-13. Twitter Card Annotation in the Markup

```
<meta name="twitter:card" content="summary" />
<meta name="twitter:site" content="@lesliesikos" />
<meta name="twitter:creator" content="@lesliesikos" />
<meta property="og:url" content="http://www.lesliesikos.com/linked-data-platform-1-0 ↩
 standardized/" />
<meta property="og:title" content="Linked Data Platform 1.0 Standardized" />
<meta property="og:description" content="The Linked Data Platform 1.0 is now a W3C ↩
 Recommendation, covering a set of rules for HTTP operations on Web resources, including ↩
 RDF-based Linked Data, to provide an architecture for read-write Linked Data on the ↩
 Semantic Web." />
<meta property="og:image" content="http://www.lesliesikos.com/img/LOD.svg" />
```

IBM Watson

IBM Watson's *DeepQA* system is a question-answering system originally designed to compete with contestants of the *Jeopardy!* quiz show, where three contestants compete against one another in answering open-domain questions. While the system famously won the contest against human grand champions, it has applications well beyond *Jeopardy!*, namely, natural language content analysis in both questions and knowledge sources in general [3]. Watson's cognitive computing algorithms are used in health care, to provide insight and decision support, and give personalized and instant responses to any inquiry or service issue in customer service. Developers can use the *IBM Watson Developers Cloud*, a collection of REST APIs and SDKs for cognitive computing tasks, such as natural language classification, concept expansion, machine translation, relationship extraction, speech to text, and visual recognition.

Among other technologies, Watson uses triplestores such as Sesame, ontologies, and inference. The information resources of DeepQA consist of unstructured, semistructured, and structured data, of which the last one plays an important role, mainly by implementing RDF in IBM's DB2 database platform [4]. DeepQA uses three types of structured data, including online structured data repositories such as DBpedia, GeoNames, YAGO, and movie datasets; structured data extracted from unstructured data; and curated data that provide additional information to be stored in the triplestore, such as question-and-answer types. Watson performs reasoning over the data, using Semantic Web technologies [5]. DeepQA uses a variety of technologies to produce candidate answers to each question. The subsequent candidate answer ranking components are primarily powered by Semantic Web technologies. In particular, Linked Data sources are used to provide typing evidence for potential answers [6]. Linked Open Data plays a crucial role in the DeepQA architecture, not only in generating candidate answers, but also to score the answers while considering multiple points of view, such as type coercion and geographic proximity [7]. The data extracted from DBpedia also supports entity disambiguation and relation detection. YAGO is used for entity type identification, wherein disjointness properties are manually assigned to higher level types in the YAGO taxonomy. To be able to answer questions from a variety of domains, Watson implements relation detection and entity recognition on top of the Natural Language Processing (NLP) algorithms that process factual data from Wikipedia [8].

BBC's Dynamic Semantic Publishing

The British Broadcasting Corporation (BBC) has implemented RDF since 2010 [9] in web sites such as the World Cup 2010 web site [10] and the London 2012 Olympics web site [11]. Today, BBC News [12], BBC Sport [13], and many other web sites across the BBC are authored and published using Semantic Web technologies. The BBC News articles use automatic metadata tagging and interlinking, and the different BBC domains (brands, locations, people, and general subjects) are integrated by a hierarchy categorizing the terms expressed in the Simple Knowledge Organization System (SKOS) [14]. The identifiers of the content categorization system (CIS) of the BBC Programmes web site [15] are mapped to DBpedia concepts. The BBC Music web site [16] is also built on Linked Data, and each artist is represented in RDF. The BBC Earth web site [17] is powered by RDF and its own Wildlife Ontology.

The publishing platform of the BBC curates and publishes XHTML and RDF aggregations based on embedded Linked Data identifiers, ontologies, and inference. The in-house content management system of the BBC (called the Content Management/Production System, or CPS, for short) supports static metadata entry through a WYSIWYG (What You See Is What You Get) editor. The manual content administration processes are completed by dynamic semantic annotations, yielding automated metadata, rich content relationships, and semantic navigation. The publishing platform automatically aggregates and renders links to relevant stories and assets [18].

The Library of Congress Linked Data Service

The US Library of Congress, the largest library in the world, featuring more than 160 million items [19], publishes subject heading taxonomies as Linked Data for cataloging. It provides the LC Linked Data Service to present data in RDF [20], and where appropriate, in SKOS, as well as using its own ontology for accurate description of classification resources and relationships. All records of the Library of Congress are available individually via content negotiation as XHTML+RDFa, RDF/XML, N-Triples, and JSON [21]. To address the limitations of MAchine-Readable Cataloging (MARC), a standard initiated by the Library of Congress, MARC records have been mapped to BIBFRAME vocabulary terms [22] to leverage Linked Data benefits. Other controlled vocabularies used by the library are Dublin Core and MARC Relator Terms (which uses the marcrel prefix). For example, the contributor term of the Dublin Core vocabulary has been refined using the correspondent term of the MARC Relator Terms vocabulary, which allows catalogers to specify the role that an individual played in the creation of a resource, such as illustrator, calligrapher, or editor, in RDF [23].

High-Performance Storage: The One Trillion Triples Mark

AllegroGraph (`http://franz.com/agraph/allegrograph/`), the industry-leading graph database discussed in Chapter 6, constantly sets new records in loading and querying huge amounts of RDF triples. In 2004, the first version of AllegroGraph was the first graph database that loaded and indexed 1 billion triples using standard x86 64-bit architecture. In 2008, 10 billion quads were loaded with AllegroGraph on the Amazon EC2 service. In June 2011, it successfully loaded 310 billion RDF triples in just over 78 hours, using an eight-socket Intel Xeon E7-8870 processor-based server system configured with 2TB RAM and 22TB physical disk space [24]. Two months later, AllegroGraph was the first graph database in the world to load, infer, and query more than 1 trillion (!) triples. This mind-blowing, unmatched performance would be enough to store [25]

- 6,350 facts about each of 158 million items in the Library of Congress, the largest library in the world
- 1000 tweets for every one of the 1 billion Twitter users
- 770 facts about every one of the 1.3 billion Facebook users
- 12 facts about every one of the 86 billion neurons in the human brain.
- 400 metabolic readings for each of the 2.5 billion heartbeats over an average human lifetime
- Five facts about every one of the 200 billion stars in the Milky Way

AllegroGraph is designed for maximum loading speed and query speed. Loading triples and quads through its highly optimized RDF/XML and N-Quads parsers is very powerful, particularly with large files. The 1,009,690,381,946 triples were loaded in just over 338 hours, with an average rate above 800,000 triples per second (see Table 8-1).

Table 8-1. *AllegroGraph's Performance in Triple Loading and Indexing*

Configuration	Triple Count	Time	Load Rate (T/s)
2×4-core Intel E5520@2.26 GHz, 48GB RAM, CentOS 5.3	1.106 billion	48 m 30 s	379,947
32-core Intel E5520@2.0 GHz, 1TB RAM, Red Hat Enterprise Linux 6.1	1.106 billion	36 m 49 s	500,679
32-core Intel E5520@2.0 GHz, 1TB RAM, Red Hat Enterprise Linux 6.1	22.120 billion	12 h 18 m 16 s	499,188
64-core Intel x7560@2.27 GHz, 2TB RAM, 22TB disk, Red Hat Enterprise Linux 6.1	310.269 billion	78 h 9 m 23 s	1,102,737
240-core Intel x5650, 2.66GHz, 1.28TB RAM, 88TB disk, Red Hat Enterprise Linux 6.1	1.009 trillion	338 h 5 m	829,556

Oracle Spatial and Graph with Oracle Database 12c reached the 1 trillion triples mark in processing and indexing RDF triples in September 2014. Oracle has a maximum load rate of 1,420,000 quads loaded and indexed per second (see Table 8-2).

Table 8-2. *Oracle Spatial and Graph's Performance in Triple and Quad Loading and Indexing*

Configuration	Count	Time	Load Rate*
64-core SPARC64 VII+@3GHz, 512GB RAM, 160 drives in dual F5100 flash arrays, Oracle Database 11.2.0.2.0	1.1 billion triples	28 m 11 s	650,500 TLIPS
64-core SPARC64 VII+@3GHz, 512GB RAM, 160 drives in dual F5100 flash arrays, Oracle Database 11.2.0.2.0	3.4 billion triples	105 m	539,700 TLIPS
40-Core Intel E7-4870@ 2.4GHz with 1TB RAM on one Sun Server X2-4 node, dual node Sun ZFS 7420 storage	27.4 billion quads	13 h 11 min	273,000 QLIPS
192-core Oracle Exadata Database Machine X4-2 High Capacity full rack, 2TB RAM, 44.8TB flash cache, eight-tray dual controller ZS3-2 storage, Oracle Database 12.1.0.1	605.4 billion quads	115.2 h	1,420,000 QLIPS

**TLIPS: Triples Loaded and Indexed Per Second; QLIPS: Quads Loaded and Indexed Per Second*

Summary

In this chapter, you saw Big Data applications using Semantic Web technologies in search engines, social media, and high-performance storage. You learned how to add structured data to your web site, to be considered by Google for inclusion in the Knowledge Graph, whose data will be used to show additional data on the Knowledge Panel and Knowledge Carousel. You know by now how to write semantic annotation for Facebook and Twitter objects in the markup.

The final chapter will demonstrate step-by-step use cases for a variety of real-life situations.

References

1. Google (2015) Introducing the Knowledge Graph. www.google.co.uk/insidesearch/features/search/knowledge.html. Accessed 9 March 2015.
2. Twitter (2015) Getting Started with Cards. https://dev.twitter.com/cards/. Accessed 12 March 2015.
3. Gliozzo, A., Patwardhan, S., Biran, O., McKeown, K. (2013) Semantic Technologies in IBM Watson. www.cs.columbia.edu/nlp/papers/2013/watson_class_acl_tnlp_2013.pdf. Accessed 23 April 2015.
4. Gucer, V. (2013) IBM is embracing Semantic technologies in its products. In: 5 Things To Know About Semantic Technologies. www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_the_semantic_technologies?lang=en. Accessed 23 April 2015.
5. Le Hors, A. (2012) Interview: IBM on the Linked Data Platform. www.w3.org/blog/2012/05/interview-ibm-on-a-linked-data/. Accessed 23 April 2015.
6. Welty, C. (2013) Semantic Web and Best Practice in Watson. In: Proceedings of the Workshop on Semantic Web Enterprise Adoption and Best Practice (WaSABi 2013), Sydney, Australia, 22 October, 2013. http://ceur-ws.org/Vol-1106/keynote2.pdf. Accessed 23 April 2015.
7. Unger, C., Freitas, A., Cimiano, P. (2014) An Introduction to Question Answering over Linked Data. In: Reasoning Web: Reasoning on the Web in the Big Data Era. Lecture Notes in Computer Science 2014, 8714:128–130, http://dx.doi.org/10.1007/978-3-319-10587-1. Accessed 23 April 2015.
8. Gliozzo, A. M., Kalyanpur, A., Welty, C. (2011) Semantic Web Technology in Watson. Tutorial at the 10th International Semantic Web Conference, Bonn, Germany, 23–27 October 2011. http://iswc2011.semanticweb.org/tutorials/semantic-web-technology-in-watson/. Accessed 23 April 2015.
9. Shotton, D. (2012) A major user of RDF linked data—the BBC. http://jats.nlm.nih.gov/jats-con/2012/presentations/shotton_jatscon2012.pdf. Accessed 24 April 2015.
10. BBC (2010) World Cup 2010. http://news.bbc.co.uk/sport2/hi/football/world_cup_2010/. Accessed 24 April 2015.
11. BBC (2012) London 2012 Olympics. www.bbc.com/sport/0/olympics/2012/. Accessed 24 April 2015.
12. BBC (2015) BBC News. www.bbc.co.uk/news/. Accessed 24 April 2015.
13. BBC (2015) BBC Sport. www.bbc.co.uk/sport. Accessed 24 April 2015.
14. Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R. (2009) Media Meets Semantic Web—How the BBC Uses DBpedia and Linked Data to Make Connections. Lecture Notes in Computer Science 2009, 5554:723–737, http://dx.doi.org/10.1007/978-3-642-02121-3_53. Accessed 24 April 2015.
15. BBC (2015) BBC Programmes. www.bbc.co.uk/programmes. Accessed 24 April 2015.

16. BBC (2015) BBC Music. www.bbc.co.uk/music. Accessed 24 April 2015.

17. BBC (2015) BBC Earth. www.bbc.com/earth/uk. Accessed 24 April 2015.

18. BBC (2012) Sports Refresh: Dynamic Semantic Publishing. www.bbc.co.uk/blogs/legacy/bbcinternet/2012/04/sports_dynamic_semantic.html. Accessed 24 April 2015.

19. Library of Congress (2015) Fascinating Facts. www.loc.gov/about/fascinating-facts/. Accessed 24 April 2015.

20. Library of Congress (2015) LC Linked Data Service. http://id.loc.gov/. Accessed 24 April 2015.

21. Ford, K. (2010) ID.LOC.GOV, 1½ Years: Review, Changes, Future Plans, MADS/RDF. http://id.loc.gov/static/presentations/kefo_dlf_id.pdf. Accessed 24 April 2015.

22. Library of Congress (2015) Vocabulary (Bibliographic Framework Initiative Technical Site). http://bibframe.org/vocab/. Accessed 24 April 2015.

23. Harper, C. A., Tillett, B. B. (2007) Library of Congress Controlled Vocabularies and Their Application to the Semantic Web. Cataloging & Classification Quarterly 2007, 43(3-4):47–68. http://dx.doi.org/10.1300/J104v43n03_03. Accessed 24 April 2015.

24. Franz Inc. (2011) Franz's AllegroGraph Sets New Record on Intel Xeon E7 Platform. http://franz.com/about/press_room/Franz-Intel_6-7-11.lhtml. Accessed 12 March 2015.

25. Oracle (2014) Oracle Spatial and Graph: Benchmarking a Trillion Edges RDF Graph. http://download.oracle.com/otndocs/tech/semantic_web/pdf/OracleSpatialGraph_RDFgraph_1_trillion_Benchmark.pdf. Accessed 3 February 2015.