HTWK Leipzig

OS: Datenbanksysteme - Aktuelle Trends

Datenbanken und Semantic Web

Autor:
Christian Wagner

Betreuer:
Prof. Thomas Kudrass

7. Juli 2014

Datenbanken und Semantic Web am Beispiel von SPARQL

Das Web flankiert in der heutigen Zeit den raschen Übergang von einer Industrie- zu einer Handelsgesellschaft. Es bietet hierfür eine Infrastruktur für einen qualitativ hochwertigen Umgang mit der Beschaffung und Bereitstellung für Informationen.
Das Web wird in allen möglichen vorstellbaren Größenordnungen kommerzialisiert.
Der durchschnittliche Alltag ist ohne dem Web gar nicht mehr vorstellbar. Es wird zur Verwaltung (eGovernment), Bildung (eLearning), Social-Networking, Suche nach Neuigkeiten oder Nachrichten usw. Dieser Datenumfang bringt einige Probleme mit sich. Sämtliche Informationen sind auf den Menschen als Endbenutzer ausgerichtet. Da das Suchen nach Informationen in heutigen Suchmaschinen (Google, Yahoo, etc.) stichwortbasiert abläuft, kann nicht effektiv nach logischen und semantischen Zusammenhängen zwischen verschiedenen Fakten gesucht werden.
Um diesem Problem entgegen zu kommen, stehen Methoden aus dem KI-Bereich (ad hoc) zur Verfügung oder die Web-Informationen werden strukturiert, um eine automatische programmatische Auswertung zu erleichtern (a priori). Auf die zweite Methode bezieht sich das Semantic Web.

Um solch eine Strukturierung umzusetzen sind zwei Voraussetzungen essenziell: Im Ersten die standardisierte Beschreibung von Informationen und im Zweiten eine Methodologie, um die Informationen zu gewinnen. Unstrukturiertem Text/ Informationen muss dazu zusätzliche Informationen gegeben werden (Metadaten). Dazu wird der zu annotierende Text in Tags eingeschlossen. Die, durch die Tags dargereichten Zusatzinformationen können dann durch verarbeitende Programme oder Bibliotheken gelesen werden.

Im Kontext zur Anwendung stehende Ressourcen sollen eindeutig referenzierbar sein. Das wird durch URIs (Uniform Resource Identifier) erreicht. Eine Ressource ist dabei alles, was eine „Identität“ besitzt. URIs erweitern das URL-Konzept.
Die Bereitstellung der URIs und Informationen soll über RDF (Resource Description Framework) Dateien erfolgen. RDF ist ein graphenbasiertes Dateiformat (gerichtete Graphen) und eine offizielle W3C Recommendation. RDF war ursprünglich ein Datenmodell, um Metadaten für Web-Ressourcen bereitzustellen, jedoch ist es inzwischen allgemeiner. Durch RDF können Informationen strukturiert und in einem maschinenlesbarem, universellen Austauschformat bereitgestellt werden.
Ein RDF Graph besteht aus URIs zur eindeutigen Referenzierung von Ressourcen, Literalen zur Beschreibung von Datenwerten und leeren Knoten. Leere Knoten erlauben Aussagen über die Existenz von Ressourcen, ohne diese benennen zu müssen. Literale werden als Zeichenketten dargestellt. Die Interpretation durch die Anwendung erfolgt dabei anhand des angegebenen Datentyps. Ist kein Datentyp angegeben, wird das Literal als pure Zeichenkette interpretiert. Ein RDF Graph ist praktisch eine Liste von Tripeln (immer: Knoten-Kante-Knoten). Ein solches Tripel besteht aus Subjekt, Prädikat und Objekt. Ein Subjekt ist dabei entweder eine URI oder ein leerer Knoten. Ein Prädikat

(Property) muss in jedem Fall durch eine URI dargestellt werden. Ein Objekt kann durch eine URI, ein Literal oder einen leeren Knoten dargestellt werden. Da die Bezeichner für Knoten und Kanten eindeutig sind, kann ein ursprünglicher Graph jederzeit aus der Tripel Liste rekonstruiert werden.
Die Tripelauflistung kann in verschiedenen Formaten erfolgen. Die gebräuchlichsten sind Turtle, da es eine sehr intuitive und menschenlesbare Syntax hat und XML, da es für XML die beste Toolunterstützung gibt. Im Turtle-Format werden zuerst die Präfixe mit einem vorgegangenen „@“ angegeben. Bei einem Präfix „ex“ für die URI „http://example.com“ würde eine Zeile im Turtle-Format so aussehen: `@prefix ex: <http://example.com/>`.
Nach dem Auflisten der Präfixe folgt das Auflistung der Tripel. Die Tripel werden stets nach dem Prinzip `<Subjekt> <Prdikat> <Objekt>.` aufgelistet.
Beispiel: —ex:Larry_Page ex:istCEO ex:Google.—
Tripel mit gleichem Subjekt können abgekürzt werden. Dabei werden die Teilstatements mit einem „;“ getrennt.
Tripel mit gleichem Subjekt und Prädikat werden durch Komma getrennt.

In XML Dateien haben RDF-eigene Tags einen festgelegten Namensraum (mit dem Bezeichner „rdf“. Präfixe werden durch ein voranstehendes `<rdf:RDF xmlsn:prefix="exampleURI">` beim jeweiligen Element bzw. Vorgängerelement angefügt.
Subjekte werden in einem „Description“ Tag kodiert, die URI wird dabei im „about“ Attributwert angegeben.
Prädikate werden einfach in Tags gesetzt. Objekte werden analog zu Subjekten behandelt.
Minimalbeispiel:

```
<rdf:RDF xmlns:rdf="http://example.com">
<rdf:Description
       rdf:about="http://example.com/Larry_Page">
   <ex:istCEO>
       <rdf:Description
              rdf:about="http://example.com/Google">
       </rdf:Description>
   </ex:istCEO>
</rdf:Description>
```

Um mehr Semantik innerhalb der Literale zu erhalten können Datentypen angegeben werden, welche durch ihre URI referenziert werden. Standardisiert verwendet man meist xsd-Datentypen.
Turtle-Syntax: `"Datenwert"^^Datentyp-URI`, z.B.: `"Google"^^xsd:string.` wobei das Präfix xsd definiert sein muss.
XML-Syntax: `< Prdikat rdf:datatype="Datentyp-URI"`

Darüber hinaus ist es wünschenswert Aussagen über generische Mengen an Ressourcen treffen zu können, wie zum Beispiel „Nur Dozenten halten Lehrveranstaltungen“ oder „Nur Menschen schreiben Bücher“.
Dazu können Ressourcen innerhalb von RDF als Klassen instanziert werden. Klassenzugehörigkeit ist hier nicht exklusiv, d.h. ein Subjekt kann mehreren Klassen angehören.
In Turtle Syntax: `ex:Google rdf:type ex:Aktiengesellschaft.`
Da es syntaktisch zwischen Instanzen und Klassen keinerlei Unterschiede gibt, können Klassen direkt als diese gekennzeichnet werden: `ex:Aktiengesellschaft rdf:type rdfs:Class`. Es können hier auch Unterklassen erstellt werden. Das funktioniert intuitiv mittels `ex:Aktiengesellschaft rdfs:subClassOf ex:Firma.`

Man kann also das Prädikat `rdf:type` mit dem Elementsymbol $\in$ vergleichen, wobei das Prädikat `rdfs:subClassOf` der Teilmenge $\subseteq$ entspricht. `subClassOf` ist reflexiv und transitiv.

Analog dazu können auch verschiedene Prädikate in Verbindung gebracht werden. Das Prädikat `rdf:subPropertyOf` ist analog zum `rdfs:subClassOf` bei Subjekten. Das Erstellen von Unterprädikaten erlaubt Schlussfolgerungen.

Für bestmögliche Semantik können Prädikate eingeschränkt werden.
So kann gelten, dass b aus einem Tripel `a ex:praedikat b` folgt. Ebenso darf gelten: $a \in X \land b \in Y$. In Turtle ausgedrückt: `a rdf:type X. b rdf:type Y.`
Durch RDFS kann das direkt beschrieben werden:
`ex:praedikat rdfs:domain X; rdfs:range Y.`

Um strukturierte, in RDF vorliegende Daten abzufragen, wird eine Anfragesprache benötigt. Hierzu dient SPARQL, eine graphenbasierte Anfragesprache.
SPARQL steht für SPARQL Protocol And RDF Query Language und ist eine Anfragesprache zur Abfrage von Instanzen aus RDF-Graphen. Anfragen werden an SPARQL-Endpoints gesendet.

Die Syntax von SPARQL erinnert stark an eine Mischung aus SQL und Turtle. Im Kopf einer Anfrage werden Präfixe definiert. Anschließend folgt eine `SELECT`-Anweisung. Der Hauptbestandteil ist das `WHERE`-Anfragemuster. Es dürfen Variablen enthalten sein, die mit einem vorangehenden „?“ oder „$“ gekennzeichnet werden. Innerhalb dieses Anfragemusters gilt die Turtle Syntax.
Minimalbeispiel:

```
PREFIX ex: <http://example.com/>

SELECT ?titel ?autor
```

```
WHERE {
    ?buch ex:verlegtVon <http://springer.com/verlag> .
    ?buch ex:titel ?titel .
    ?buch ex:autor ?autor .
}
```

Diese Anfrage würde als Ergebnis eine Tabelle liefern, die den Titel und Autoren von Büchern enthält, die beim Springer-Verlag verlegt wurden.

Hier können auch mittels dem Schlüsselwort `FILTER` Informationen gefiltert werden. Innerhalb einer Filter-Anweisung sind arithmetische und boolesche Operationen zulässig. Es können auch mehrere Filter mittels booleschen Operatoren verknüpft werden, da jede Filter-Anweisung einen Wahrheitswert liefert.

SPARQL bietet auch eine Vielzahl von Spezial- und Aggregierungsfunktionen, wie zum Beispiel das Bilden eines Durchschnittswertes, das Sortieren der Ergebnise, das Überprüfen von Datentypen etc.

Um Tripel in einer Datenbank zu speichern gibt es verschiedene Ansätze. Vom Grundprinzip her werden alle Statements, Subjekte, Prädikate und Objekte mit einer eindeutigen ID versehen und die IDs und eigentlichen Werte der Ressourcen werden in getrennten Tabellen abgelegt. Den Zusammenhang bildet die ID.
Anfrage-Engines die das ermöglichen wären beispielsweise ARQ, Tracker und Virtuoso.