Neue Konzepte für RDF-Managementsysteme

Ralf Heese
Humboldt-Universität zu Berlin
Institut für Informatik, Datenbanken und Informationssysteme
Unter den Linden 6, D-10099 Berlin
rheese@informatik.hu-berlin.de

Zusammenfassung

Durch die Erforschung und Standardisierung von Semantic-Web-Technologien wie dem Resource Description Framework und der Web Ontology Language werden auf lange Sicht große Mengen an semantisch annotierten Daten verfügbar sein. Daher werden Werkzeuge benötigt, die ein effizientes Verarbeiten, Transformieren und Anfragen dieser Informationen ermöglichen. Dieser Beitrag beschreibt die Operationen auf RDF-Modellen und diskutiert die Anforderungen an RDF-Managementsysteme für das Verwalten großer Mengen semantischer Daten. Daraus und aus den existierenden Ansätzen zur Verwaltung von RDF-Daten werden noch ungelöste Problemstellungen aufzeigt.

1 Einleitung

In den vergangenen Jahren etablierte sich das Internet als Medium für Datenaustausch und E-Commerce. Durch die Erforschung und Standardisierung neuer Technologien wie zum Beispiel dem Resource Description Framework (RDF) [15, 14] oder der Web Ontology Language (OWL) [13] wird die Funktionalität des Internet erweitert. Momentan dient das Internet vor allem zur Publikation von Dokumenten, die für das Lesen durch Menschen gedacht sind. Zukünftig wird ein Teil dieser Informationen auch in maschinenverarbeitbarer Form angeboten werden. [5].

Die Verfügbarkeit von Daten alleine ist hierbei nicht ausreichend, sondern es werden außerdem noch Werkzeuge benötigt, die ein effizientes Verarbeiten, Transformieren und Anfragen dieser Informationen ermöglichen. Unter anderem beschäftigt sich das Projekt InterVal–Wissensnetze[1] zur Zeit mit der Entwicklung eines Jobvermittlungsportal auf Basis semantisch beschriebener Jobanzeigen und -gesuchen, wobei ein Schwerpunkt auf dem semantischen Matching zwischen diesen liegt. Die Beschreibung erfolgt unter Verwendung von Human-Resource-Ontologien basierend auf OWL, das selbst auf dem RDF-Datenmodell aufsetzt. Hierbei wird ein RDF-Managementsystem benötigt, dass nicht nur die großen Datenmengen verwalten kann, sondern auch die Anfragen wie dem semantischen Matching effizient unterstützt. Der letzte Punkt verdeutlicht auch, dass es sich dabei nicht um eine Standarddatenbankanwendung handelt.

Nach einer kurzen Einführung in die Besonderheiten des RDF-Datenmodells untersuchen wir dazu im Abschnitt 2, welche Anforderungen an ein solches System gestellt werden. Insbesondere gehen wir auf Operationen auf RDF-Daten ein. Abschnitt 3 befasst sich anschließend mit existierenden Systeme und analysiert, inwiefern diese den Anforderungen gerecht werden. Dort werden die Vor- und Nachteile der Systeme gegenübergestellt. Abschließend zeigen wir Problemstellungen bei der Realisierung von RDF-Managementsystemen auf, die unsere Forschung adressiert.

[1] InterVal – Internet and Value Chains, Berliner Forschungszentrum für Internetökonomie (http://interval.hu-berlin.de) ist ein vom Bundesministerium für Bildung und Forschung (BMBF) gefördertes Projekt.

2 Operationen und Anforderungen

Bevor wir in diesem Abschnitt auf die Operationen auf dem RDF-Datenmodell eingehen, fassen wir kurz die Besonderheiten dieses Datenmodells zusammen. Aus dem Datenmodell und den Operationen lassen sich anschließend die Anforderungen an ein RDF-Managementsystem ableiten.

2.1 RDF-Datenmodell und Operationen

Die Basis des Resource Description Frameworks [15, 14] bildet die syntaxneutrale Darstellung von Aussagen. Zum einen lassen sich diese als gerichteter beschrifteter Graph repräsentieren, bei dem die Knoten Resourcen oder Literale und die Kanten Eigenschaften sind. Zum anderen lassen sich die Aussagen als eine Menge von Tripeln der Form *(Ressource, Eigenschaft, Objekt)* ansehen, wobei ein Objekt entweder eine Ressource oder ein Literal sein kann. Unter Ressourcen werden beliebige Dinge verstanden, die mittels einer URI adressiert und über die Aussagen getroffen werden. Eigenschaften beschreiben die Ressourcen näher, indem Beziehungen zu anderen Ressourcen oder Literalen hergestellt werden; insbesondere sind Eigenschaften selbst wieder Ressourcen. Dadurch ergeben sich komplexe Graphenstrukturen. Beispielsweise ist die Aussage des Tripels *(ex:Hans, ex:hatAusbildung, ex:Maler)*, dass Hans eine Berufsausbildung als Maler hat. Als Modell bezeichnen wir den Kontext, in dem eine Menge von Aussagen definiert wird.

Im Folgenden beschreiben wir Operationen auf dem RDF-Datenmodell, wobei wir die Tripeldarstellung zur Veranschaulichung nutzen. Die Menge der Aussagen lassen sich auf zwei Ebenen anfragen wie durch Abbildung 1 veranschaulicht wird. Die Grundlage für Operationen auf einer RDF-Datenbasis bilden die expliziten Aussagen. Darauf aufbauend lassen sich weitere Aussagen ableiten, indem zusätzlich Regeln definiert werden – der Vorgang wird auch als Inferenz bezeichnet. Als erstes beschreiben wir Operationen auf expliziten Aussagen. Danach stellen wir Operationen auf der Menge der virtuellen (deduzierten) Aussagen vor.

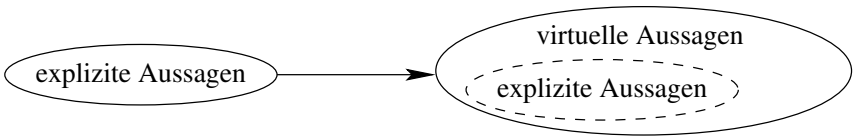

Abbildung 1: Kategorien von RDF-Aussagen

Neben dem Einfügen und Löschen von Tripeln lassen sich verschiedene Leseoperationen auf expliziten Aussagen identifizieren. Ausgehend von der Tripelform können wir einen Projektionsoperator definieren, der beliebige Kombinationen von Komponenten aus den Tripeln extrahiert. Ein Beispiel hierfür ist eine Anfrage der Form *(?x, ex:dauerBerufserfahrung, ?y)*, die als Ergebnis alle Ressourcen bzw. Literale liefert, für die ein Tripel mit der Eigenschaft *ex:dauerBerufserfahrung* existiert. Die mit einem Fragezeichen beginnenden Zeichenketten sind Variablen. Diese einfache Form lässt sich um einen Filter erweitern, der die Werte für *?y* einschränkt: *(?x, ex:dauerBerufserfahrung, ?y) AND ?y >= 5*. Durch wiederholtes Verwenden derselben Variablen in mehreren Anfragetripeln lassen sich Tripel miteinander kombinieren (Join). Beispielsweise ergibt die Anfrage *(?x, ex:dauerBerufserfahrung, "5"), (?x, ex:hatAusbildungAls, ex:Maler)* die Ressourcen (*res*), für die beide Tripel der Anfrage (*?x* mit *res* ersetzt) in der Datenbasis enthalten sind.

Ergänzen wir das Datenmodell um eine Regelmenge, so lassen sich Aussagen ableiten, die nicht explizit definiert wurden. Einige Regeln werden dabei bereits durch die RDFS- und OWL-Spezifikationen eingebracht. Eine typische Regel ist die Definition der Transitivität von Eigenschaften wie *rdfs:subClassOf*. Grundsätzlich lassen sich alle Operationen, die für explizite Aussagen definiert wurden, auch auf virtuelle anwenden, wobei sich jetzt die Einfüge- und

Löschoperationen auf Regeln beziehen. Ein großes Problem im Zusammenhang mit Selektionen ist die Berechenbarkeit aller ableitbaren Aussagen.

Darüber hinaus lassen sich weitere Operationen definieren: Validierung des Modells und Herleitung einer Aussage. Die erste Operation überprüft das Modell auf nicht eingehaltene Bedingungen. So lassen sich in RDFS beispielsweise die Definitions- und Wertebereiche von Eigenschaften spezifizieren. Beim Herleiten einer Aussage interessieren wir uns für die Sequenz der Deduktionsschritte, mit der eine Aussage aus einem Modell abgeleitet wurde.

Auf Modellebene lassen sich noch Operationen wie Vereinigung, Durchschnitt und Differenz von Modellen definieren.

2.2 Anforderungen

Die allgemeinen Anforderungen an Datenbankmanagementsysteme, wie sie in [12] beschrieben wurden, sind sicherlich hinreichend bekannt (z.B. Skalierbarkeit). Daher betrachten wir im Folgenden nur die Anforderung, die durch die Besonderheiten des RDF-Datenmodells impliziert werden. Die Grundlage bilden die oben vorgestellten Operationen und Veröffentlichungen aus dem Gebiet der RDF-Datenverarbeitung [9, 4, 17].

Anfragesprache: Die Anfragesprache eines RDF-Managementsystems ist deklarativ und berücksichtigt die Besonderheiten des RDF-Datenmodells wie die Tripelform der Aussagen, Modelle (Kontext) von Aussagen oder Namensräume.

Reifikation:[2] Die Verwendung von Reifikation ermöglicht es, Aussagen über Aussagen zu treffen (s. [15]). Bei der Reifikation einer Aussage entstehen vier Tripel, was bei vielen reifizierten Aussagen zu einer Vervielfachung der Tripel in der Datenbasis führt (triple bloat). Ein RDF-Managementsystem soll Reifikation effektiv verarbeiten können.

Effizienz: Lesende und schreibende Operationen werden effizient ausgeführt. Diese Anforderung umfasst auch ein effizientes Ausführen von Anfragen und das schnelle Importieren von RDF-Daten, die serialialisiert als XML vorliegen.

Inferenz: Die Anwendung von Inferenzmechanismen ist ein wichtiger Aspekt bei der Benutzung von RDFS und OWL. Teile der Semantik dieser Spezifikationen sind über Regeln definiert. Daher ist eine Unterstützung von Inferenz durch das RDF-Managementsystems erforderlich.

Schnittstellen: Es werden Schnittstellen benötigt, damit das RDF-Managementsystem in Anwendungen integriert werden kann.

3 Existierende RDF-Managementsysteme

Seit Beginn der Spezifizierung von RDF wurden RDF-Managementsysteme entwickelt, die fast alle entweder auf (objekt)relationale oder deduktive Technologie aufsetzen und die RDF-Daten als eine Menge von Tripeln handhaben. In diesem Abschnitt skizzieren wir die grundsätzlichen Ideen. Für weitere Details verweisen wir auf die jeweiligen Publikationen; einen Überblick gibt [3]. Abschließend fassen wir die Vor- und Nachteile dieser Technologien zusammen.

Ein Ansatz, wie er unter anderem in [6, 2, 4] Anwendung findet, verwendet ein relationales Datenbankmanagementsystem (RDBMS), in dem die Aussagen in einer Tabelle verwaltet werden. In den meisten Systemen werden dabei die URIs ganz oder teilweise in andere Tabellen ausgelagert. Dies erbringt einen Speicherplatzgewinn, da sich wenige URIs häufig wiederholen.

[2] Aus Platzmangel steht hier Reifikation stellvertretend für weitere Konzepte im RDF-Modell (bNodes, Container), die speziell vom RDF-Managementsystem unterstützt werden sollten. (vgl. [15])

In [1] wird dahingegen die Vererbunghierarchie zwischen den Eigenschaften dazu genutzt, eine Tabellenhierarchie in einem ORDBMS aufzubauen. Damit wird eine Verteilung der Tripel auf mehrere Tabellen erreicht. Nachteile bringt der erhöhte Aufwand bei Veränderungen in der Eigenschaftshierarchie, da sich diese direkt auf das Datenbankschema auswirken.

Als Anfragesprachen haben sich RDQL [7] und RQL [8] etabliert, auf die wir nicht detaillierter eingehen werden.

Im vorhergehenden Abschnitt stellten wir fest, dass Inferenz von einem RDF-Managementsystem unterstützt werden sollte. Daher wurden auch Systeme auf Basis von deduktiven Datenbanken XSB [18] oder SWI-Prolog [16] entworfen. Diese Ansätze unterscheiden sich in der Abbildung des RDF-Datenmodells auf die deduktive Datenbank.

In [11] werden die Aussagen in eine um F-Logik-Konstrukte erweiterte Hornlogik abgebildet, wobei auch Modellinformationen berücksichtigt werden: *Ressource[Eigenschaft → Objekt]@Modell*. Eine andere Abbildung wird in [10] vorgenommen, indem eine Aussage in ein Prädikat der Form *triple(Ressource, Eigenschaft, Objekt)* überführt wird.

Die Fähigkeiten der vorgestellten Ansätze sind komplementär. Die Stärken der RDBMS-basierten Ansätze liegen in der Projektion auf Komponenten der Tripel und der Einbeziehung des Sekundärspeichers. Eine Inferenzkomponente muss aber separat ergänzt werden. Nachteilig auf die Anfrageabarbeitung wirkt sich Inferenz bei den vorgestellten Datenbankschemata aus, da dabei Tripel durch teure Joinoperationen miteinander verknüpft werden (vgl. Abschnitt 2.1). Dahingegen wurden deduktive Datenbanken speziell für das Inferieren entwickelt. Ein Problem besteht bei diesen Systemen darin, dass vor der Ausführung von Inferenzanfragen alle Fakten und Regeln im Hauptspeicher vorliegen müssen.

4 Problemstellungen und Ausblick

Mit diesem Beitrag haben wir die Anforderungen an ein RDF-Managementsystem aufgezeigt. Heute verfügbare Systeme sind bisher nicht in der Lage alle diese Anforderungen gleichzeitig zu erfüllen. Das Ziel unserer Forschung ist die Verknüpfung der Vorteile relationaler Technologien mit denen von deduktiven Datenbanken unter der Berücksichtigung der Besonderheiten des RDF-Datenmodells. Hieraus ergeben sich für uns folgende Fragestellungen:

- Wie lassen sich große Aussagenmodelle effizient handhaben? Wie lassen sich RDF-Konstrukte wie reifizierte Aussagen effizient speichern und anfragen?
- Wie können Inferenzanfragen durch Indexe unterstützt werden?
- Wie können Inferenzanfragen so umgeschrieben werden, dass die Hauptlast durch das DBMS getragen wird?
- Wie verändern sich die Eigenschaften des Systems, wenn anstatt der Tripel der Graph als Konzept für die physische Speicherung verwendet wird?
- Welche anderen Technologien können zur Verbesserung von RDF-Managementsystemen beitragen?

Literatur

[1] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *Proceedings of the 2nd International Workshop on the Semantic Web*, pages 1–13, 2001.

[2] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the 1nd International Semantic Web Conference*, volume 2342 (LNCS), pages 54–68, 2002.

[3] J. Große. *Speicherverfahren und Werkzeuge für RDF/S*, Dezember 2002. http://www.xml-clearinghouse.de/reports/.

[4] S. Harris and D. N. Gibbins. 3store: Efficient Bulk RDF Storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Web Systems*, 2003.

[5] J. Hendler, T. Berners-Lee, and E. Miller. Integrating Applications on the Semantic Web. *Journal of the Institute of Electronic Engineers of Japan*, 122(10):676–680, 2002.

[6] Hewlett Packard Research Labs. *Jena2—A Semantic Web Framework*, 2004.

[7] Hewlett Packard Research Labs. *RDQL - RDF Data Query Language*, 2004. http://www.hpl.hp.com/semweb/rdql.htm.

[8] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of the 11th Internaltional World Wide Web Conference*, page 329, 2002.

[9] S. Melnik. *Storing RDF in a Relational Databases*, December 2001.

[10] J. Peer. A Logic Programming Approach to RDF Document and Query Transformation. In *Proceedings of the Workshop on Knowledge Transformation for the Semantic Web*, 2002.

[11] M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *Proceedings of the 1st International Semantic Web Conference*, volume 2342 (LNCS), pages 364–378, 2002.

[12] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

[13] W3C. *OWL Web Ontology Language—Reference*, 2004. http://www.w3.org/TR/owl-ref/.

[14] W3C. *RDF Vocabulary Description Language 1.0: RDF Schema*, 2004. http://www.w3.org/TR/rdf-schema/.

[15] W3C. *RDF/XML Syntax Specification (Revised)*, 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[16] J. Wielemaker. An overview of the SWI-Prolog Programming Environment. In F. Mesnard and A. Serebenik, editors, *Proceedings of the 13th International Workshop on Logic Programming Environments*, pages 1–16, Heverlee, Belgium, December 2003. Katholieke Universiteit Leuven.

[17] J. Wielemaker, G. Schreiber, and B. Wielinga. Prolog-based Infrastructure for RDF: Scalability and Performance. In *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 (LNCS), pages 644–658, 2003.

[18] XSB Research Group. XSB, 2004. http://xsb.sourceforge.net/.