

Implementierung einer RDF-Datenstromverarbeitung mit SPARQL

André Bolles, Marco Grawunder

Universität Oldenburg
Department für Informatik, Abteilung Informationssysteme
[andre.bolles|marco.grawunder]@uni-oldenburg.de

Zusammenfassung

Diese Arbeit betrachtet die Verarbeitung von RDF-Datenströmen. Im Gegensatz dazu betrachten viele andere Arbeiten lediglich relationale oder XML-Datenströme (vgl. [ABW03, Krä07] u. a.). Zur Verarbeitung von Anfragen auf RDF-Datenströmen ist ein Entwurf zur Erweiterung bestehender Anfragekonzepte notwendig. Mit Stream-SPARQL wurde in Oldenburg eine Möglichkeit geschaffen, basierend auf der W3C-Empfehlung SPARQL, Anfragen auf RDF-Datenströmen zu verarbeiten. Hierzu wurde SPARQL in vier Schritten erweitert: Spracherweiterung, Definition der Semantik von Algebra-Operatoren auf Datenströmen, Definition möglicher Operatorrealisierungen und Erzeugung von Anfrageplänen aus SPARQL-Datenstromanfragen. Die Umsetzung dieser SPARQL-Erweiterung erfolgte dabei im ODYSSEUS-Framework, einem Datenstrommanagementsystem der Abteilung Informationssysteme der Universität Oldenburg. Diese Arbeit stellt die Struktur der SPARQL-Erweiterung in ODYSSEUS vor und erläutert, wie dadurch die Verarbeitung von RDF-Datenströmen ermöglicht wird.

1 Einleitung

Datenströme spielen heute in vielen Anwendungen eine Rolle. Beispielsweise werden moderne Windenergieanlagen (WEA) mit Sensoren ausgestattet, die über Datenströme Informationen über den Zustand einer WEA bereitstellen. Datenströme sind potentiell unendlich. Daher werden in der Regel nur Ausschnitte eines Datenstroms in Form von Fenstern betrachtet und Datenstromelemente werden mit Gültigkeiten, bspw. über Zeitintervalle (vgl. [Krä07]), versehen. Der Standard 61400-25 der IEC [Int06] bietet eine Möglichkeit zur standardisierten Überwachung und Steuerung von WEA. Ähnlich wie der IEC Standard 61970 könnte dieser Standard zukünftig auch in RDF modelliert werden (vgl. [UG07]). Daher untersuchen wir Möglichkeiten zur Verarbeitung von RDF-Datenströmen (vgl. [BGJ08]). Diese Arbeit beschreibt die Umsetzung einer Erweiterung der W3C Empfehlung SPARQL für RDF-Datenströme im Oldenburger **DynaQuest Datastream Query System** (ODYSSEUS).

Der Aufbau dieser Arbeit ist an die einzelnen Schritte zur SPARQL-Erweiterung angelehnt. Der folgende Abschnitt 2 betrachtet die SPARQL-Spracherweiterung. Daran anschließend wird auf logische Algebra-Operatoren eingegangen, bevor im Abschnitt 4 die Realisierung dieser Algebra-Operatoren in einer physischen Algebra beschrieben wird. Der Abschnitt 5 erläutert die Erzeugung von ausführbaren Anfrageplänen, die aus SPARQL-Datenstromanfragen gewonnen werden können. Diese Arbeit schließt mit einer Darstellung verwandter Arbeiten in Abschnitt 6 und einem Ausblick auf weitere Forschungsvorhaben in Abschnitt 7 ab.

2 SPARQL-Spracherweiterung

Im ersten Schritt wurde die eigentliche Anfragesprache SPARQL so erweitert, dass auch die Referenzierung von Datenströmen in einer Anfrage möglich wird. Außerdem können in Anlehnung an den SQL:2003 Standard¹ Fenster auf diesen Datenströmen ausgedrückt werden. Das folgende Listing 1 zeigt ein Beispiel einer SPARQL-Datenstromanfrage:

¹<http://savage.net.au/SQL/sql-2003-2.bnf.html#window%20clause>

```

SELECT ?x ?y ?z
FROM STREAM <http://iec.org/td.rdf>
  WINDOW RANGE 200 SLIDE
WHERE { ?x wtur:StrCnt ?y .
  { ?x wtur:StopCnt ?z .
    WINDOW RANGE 450 FIXED } }

```

Listing 1: Beispiel einer SPARQL-Datenstrom-Anfrage

Die Verarbeitung einer solchen Anfrage geschieht in einem ersten Schritt durch das Parsen mit anschließender Überführung in einen logischen Anfrageplan. Mit ARQ [Sea07] existiert bereits eine Referenzimplementierung von SPARQL, in der die SPARQL-Grammatik mit JavaCC² umgesetzt wird. Auf Basis dieser Referenzimplementierung wurde mit unserer Arbeit das Parsen von SPARQL-Datenstromanfragen ermöglicht. Dazu haben wir die JavaCC-basierte Grammatik so erweitert, dass die neuen SPARQL-Datenstrom-Konstrukte mit einem von JavaCC erzeugten Parser erkannt werden. Außerdem erweiterten wir die Klassenstruktur von ARQ, um eine Repräsentation einer SPARQL-Datenstromanfrage im Hauptspeicher vorhalten zu können. Die folgende Abbildung 1 zeigt diese erweiterte Klassenstruktur der ARQ-Engine.

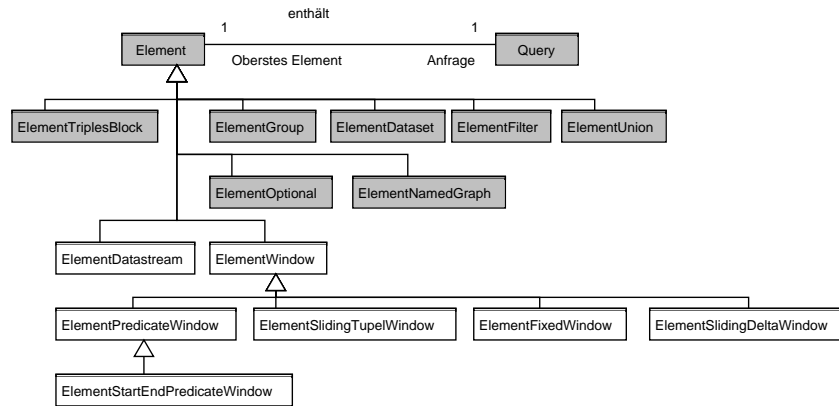


Abbildung 1: Erweiterte Klassenstruktur in ARQ

In dieser Abbildung sind sowohl die Klassen zu erkennen, die bereits von ARQ zur Übersetzung gewöhnlicher SPARQL-Anfragen benötigt werden (grau hinterlegt) als auch die neu entworfenen Klassen zur Übersetzung von SPARQL-Datenstromanfragen. Insbesondere wurden Klassen für die Repräsentation verschiedener Fenstertypen entworfen, die auf Datenströmen definiert werden können. Objekte der von `Element` abgeleiteten Klassen werden vom Parser beim Übersetzen erzeugt und zu einem Abstract Syntax Tree (AST) zusammengeführt. So kann bspw. ein Objekt der Klasse `ElementGroup`, welches ein Basic-Graph-Pattern (BGP) in einer SPARQL-Anfrage repräsentiert, als Kindknoten ein Objekt der Klasse `ElementFixedWindow` erhalten, welches ein festes Zeitfenster innerhalb eines BGP repräsentiert (vgl. Listing 1). So entsteht beim Parsen einer SPARQL-Datenstromanfrage-Zeichenkette bereits eine erste logische Repräsentation der Anfrage mit grundlegenden Metainformationen. Um jedoch auch die Semantik einzelner Operatoren zu beschreiben, ist die Erzeugung eines logischen Anfrageplans notwendig, worauf der nächste Abschnitt eingeht.

²<https://javacc.dev.java.net/>

3 Logische Operatoren

Zur Definition der Semantik von SPARQL-Datenstromanfragen haben wir uns an dem allgemeinen Vorgehen bei der Entwicklung von Anfrageverarbeitungssystemen orientiert und eine logische Algebra analog zu [Krä07] definiert. Neben der Semantik stellen die logischen Algebra-Operatoren einfache Metainformationen wie die Fenstergröße bereit und erlauben die Berechnung von Ausgabeschemata, um diese nicht während der Anfrageverarbeitung auf der physischen Ebene mitführen zu müssen. Diese Schemaberechnung ist in Abb. 2 dargestellt.

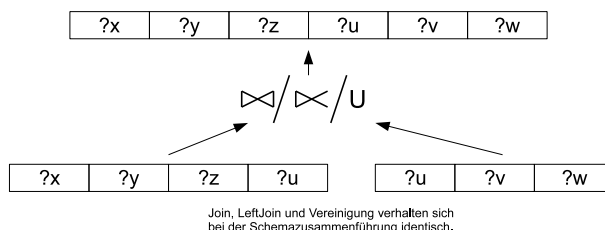


Abbildung 2: Schemamitführung in ODYSSEUS

In dieser Abbildung ist zu erkennen, dass bei einem Join, LeftJoin und der Vereinigung bereits im Vorfeld der Anfrageverarbeitung das Ausgabeschema dieser Operatoren bestimmt wird, indem zunächst die Attribute der linken Eingabe in das Ausgabeschema übernommen und dann die fehlenden Attribute der rechten Eingabe angehängt werden. Dabei entsprechen in diesem Beispiel die Attribute den Variablen, die in einer SPARQL-Anfrage spezifiziert und durch ein Basic-Graph-Pattern-Matching in sogenannten SPARQL-Lösungen zusammengefasst werden (vgl. [PS08]). Die Schemazusammenführung wird in SPARQL auch bei der Vereinigung benötigt, da hier, anders als in SQL, auch Datenströme mit unterschiedlichen Schemata vereinigt werden können. Die in Abbildung 3 dargestellte Klassenstruktur für logische Anfragepläne ermöglicht eine solche Vorausberechnung der Ausgabeschemata einzelner Operatoren.

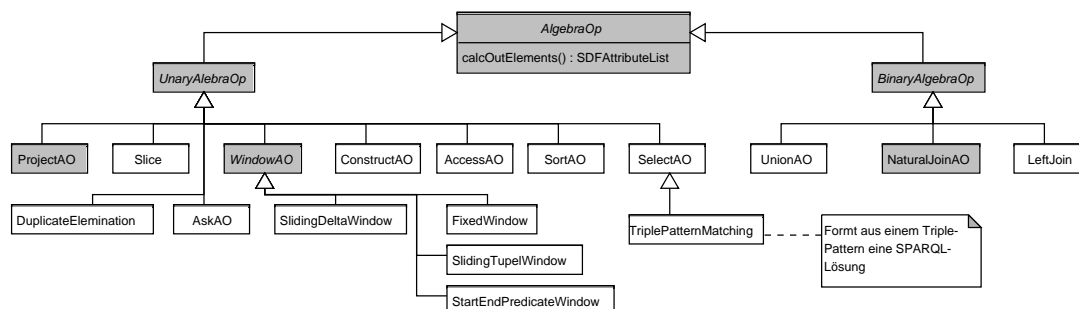


Abbildung 3: Klassenstruktur für logische Anfragepläne

Alle Klassen sind von einer abstrakten Klasse `AlgebraOp` abgeleitet und müssen damit eine Methode `calcOutElements` bereitstellen. Mit Hilfe dieser Methode ist es möglich die Ausgabeschemata einzelner Operatoren in Form einer `SDFAttributeList` zu bestimmen (Source Description Framework, s. [Gra05])³. Neben der Berechnung von Ausgabeschemata können logische Anfragepläne dazu dienen Optimierungen durchzuführen, ohne verschiedene Operatorrealisierungen berücksichtigen zu müssen. Um dennoch eine Äquivalenz zu physisch ausführbaren Anfrageplänen zu gewährleisten, haben wir Transformationsfunktionen zwischen logischen und physischen Anfrageplänen analog zu [Krä07] definiert. Für Details sei daher auf [Krä07] verwiesen.

³Von ODYSSEUS bereitgestellte Klassen sind grau hinterlegt

4 Physische Operatoren

Während auf der logischen Ebene die Semantik einzelner Operatoren definiert wird, enthalten physische Operatoren die eigentlichen Realisierungen. Dabei arbeiten diese Operatoren auf physischen Datenströmen, in denen die Gültigkeit von Elementen über Zeitintervalle ausgedrückt wird (Intervallansatz, vgl. [Krä07]). Die Abbildung 4 zeigt die Klassenstruktur für physische SPARQL-Datenstrom-Operatoren in ODYSSEUS.

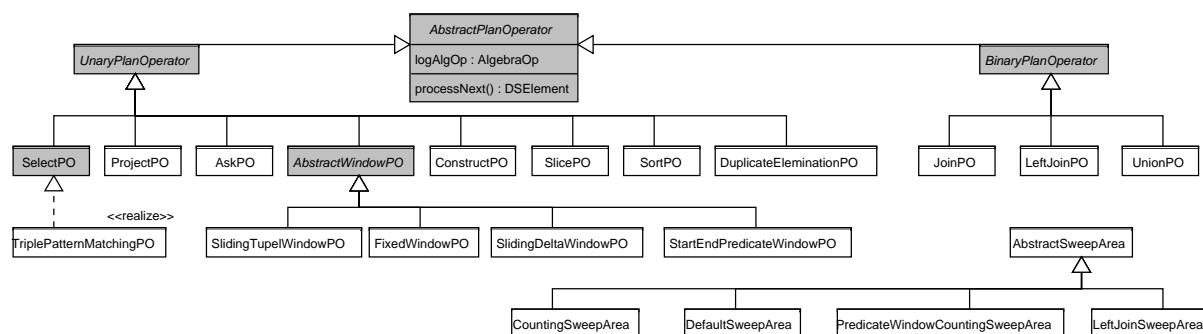


Abbildung 4: Klassenstruktur für physische Anfragepläne

Alle Klassen erben von einer abstrakten Klasse `AbstractPlanOperator`. Die eigentliche Realisierung der Operatoren wird jeweils in der Methode `processNext` implementiert⁴. Die *SweepAreas* [Krä07] werden von zustandsbehafteten Operatoren benötigt. Sie enthalten jeweils die aktuell gültigen Elemente. Je nach Operator werden evtl. besondere Implementierungen benötigt, so dass eine Klassenhierarchie für *SweepAreas* entwickelt wurde. Die Metadaten erhalten physische Operatoren aus ihren logischen Gegenstücken, die sie bei der Erzeugung physischer Anfragepläne als Attribute übergeben bekommen.

5 Erzeugung physischer Anfragepläne

Um aus einer SPARQL-Datenstromanfrage-Zeichenkette einen ausführbaren Anfrageplan zu erhalten ist zunächst die Erzeugung eines logischen Anfrageplans notwendig. Dies wird dadurch realisiert, dass der von ARQ erzeugte AST rekursiv durchlaufen wird und alle Objekte der von `Element` abgeleiteten Klassen durch Objekte von entsprechenden Klassen der logischen Operatoren ersetzt werden. So wird bspw. ein Objekt der Klasse `ElementUnion` durch ein Objekt der Klasse `UnionAO` ersetzt. Anschließend werden zunächst die Ausgabeschemata der einzelnen Operatoren berechnet und dann für jeden logischen Operator eine mögliche physische Realisierung ausgewählt wird. Auch hier wird der logische Anfrageplan rekursiv durchlaufen und die logischen Operatoren werden durch die ausgewählten physischen Gegenstücke ersetzt. Bei der Ersetzung werden gleichzeitig die logischen Metadaten in den physischen Plan integriert. Die eigentliche Ausführung des Anfrageplans geschieht nach dem Open-Next-Close-Ansatz (ONC-Ansatz) (vgl. bspw. [Gra05]). Die in [Gra05] definierte Ausführungseengine ist dabei so generisch gehalten, dass mit den zuvor beschriebenen Klassenstrukturen, diese Engine ohne Anpassung übernommen werden kann und eine Anfrageausführung möglich wird.

6 Verwandte Arbeiten

Auf konzeptioneller Ebene ist diese Arbeit durch [Krä07] inspiriert worden. Insbesondere der Intervallansatz, der Gültigkeiten von Datenstromelementen über Zeitintervalle ausdrückt, und

⁴Von ODYSSEUS bereitgestellte Klassen sind grau hinterlegt

die SweepArea wurden aus dieser Dissertation übernommen. Weiterhin basiert die Implementierung der Konzepte natürlich auf [Gra05], da das dort entwickelte DYNAQUEST-Framework die Grundlage für ODYSSEUS darstellt. So konnte durch die Wiederverwendung bestehender Klassenhierarchien (`AlgebraOp`, `AbstractPlanOperator` der ONC-Ansatz aus [Gra05] übernommen werden. Außerdem wurden viele Ansätze von SPARQL [PS08] und ARQ [Sea07] übernommen.

7 Ausblick

Die SPARQL-Datenstrom-Erweiterung wurde vollständig in ODYSSEUS implementiert. Es wurden bereits erste Ergebnisse mit einer Evaluierung von SPARQL-Datenstrom-Anfragen auf Simulationsdatenströmen erzielt, die jedoch noch nicht repräsentativ sind. Daher wird eine weitere Evaluierung auf Realdatenströmen, bspw. im Rahmen des Alpha-Ventus-Projektes⁵, erfolgen. Weiterhin wird die SPARQL-Erweiterung um Aggregationsoperatoren angereichert und es werden zukünftig Alternativen zum Intervallansatz, wie bspw. der positiv-/negativ-Ansatz (vgl. [GHM⁺04]), untersucht werden.

Literatur

- [ABW03] ARASU, Arvind ; BABU, Shivnath ; WIDOM, Jennifer: An Abstract Semantics and Concrete Language for Continuous Queries over Streams and Relations. In: *9th International Workshop on Database Programming Languages* Stanford University, 2003, S. 1 – 11
- [BGJ08] BOLLES, Andre ; GRAWUNDER, Marco ; JACOBI, Jonas: Straming SPARQL - Extending SPARQL to process data streams. In: BECHHOFFER, Sean (Hrsg.) ; HAUSWIRTH, Manfred (Hrsg.) ; HOFFMANN, Joerg (Hrsg.) ; KOUBARAKIS, Manolis (Hrsg.): *Proceedings of the 5th European Semantic Web Conference*, 2008. – Veröffentlichung im Juni 2008 erwartet
- [GHM⁺04] GHANEM, Thanaa M. ; HAMMAD, Mustafa A. ; MOKBEL, Mohamed F. ; G.AREF, Walid ; ELMAGARMID, Ahmed K.: Query Processing using Negative Tuples in Stream Query Engines / Purdue University. 2004. – Forschungsbericht
- [Gra05] GRAWUNDER, Marco: *DYNAQUEST: Dynamische und adaptive Anfrageverarbeitung in virtuellen Datenbanksystemen*, Universität Oldenburg, Diss., 2005
- [Int06] INTERNATIONAL ELECTROTECHNICAL COMMISSION TECHNICAL COMMITTEE 88: 61400-25 Communications for monitoring and control of wind power plants / International Electrotechnical Commission. 2006. – Forschungsbericht
- [Krä07] KRÄMER, Jürgen: *Continuous Queries over Data Streams - Semantics and Implementation*, Universität Marburg, Diss., 2007
- [PS08] PRUD'HOMMEAUX, Eric ; SEABORNE, Andy: SPARQL Query Language for RDF / World Wide Web Consortium. <http://www.w3.org/TR/rdf-sparql-query/> letzter Zugriff: 21. Januar 2008, Januar 2008. – W3C Recommendation
- [Sea07] SEABORNE, Andy: *ARQ - A SPARQL Processor for Jena*. <http://jena.sourceforge.net/ARQ/>, letzter Zugriff: 15. Oktober 2007, 2007
- [UG07] USLAR, Mathias ; GRÜNING, Fabian: Zur semantischen Interoperabilität in der Energiebranche: CIM IEC 61970. In: *Wirtschaftsinformatik* 49 (2007), September, Nr. 4, S. 295 – 303

⁵<http://www.alpha-ventus.de>