

Universität Frankfurt am Main
Fachbereich Biologie und Informatik (15)
Institut für Informatik

Lehrstuhl für Datenbanken und Informationssysteme

Seminar WS2004
„Automatische Verarbeitung von Datenströmen“
Automatic Processing of Data Streams

Thema: easy RDF Query Language
- eRQL -

Veranstalter:

Prof. Dott. -Ing. Roberto Zicari
Dipl. -Math. Karsten Tolle
Dipl.-Inform. Peter Werner

Autor: Abu Gaber, Halil

Agenda

1. Einführung

- 1.1 Semantic Web auf Basis von RDF
- 1.2 Resource Description Framework (RDF)

2. Abfragesprachen für RDF-Daten

- 2.1 Bedarf an neuartige RDF-Abfragesprache
- 2.2 Vorderrungen an die neuartige RDF-Abfragesprache
- 2.3 Wahl der Zwischensprache

3. eRQL-Ad Hoc-Abfragen für Informationsportale

- 3.1 Eigenschaften von eRQL
 - 3.1.1 Groß-/Kleinschreibung
 - 3.1.2 Boolesche Verknüpfungen
 - 3.1.3 Ein-Wort-Abfragen
 - 3.1.4 Umgebung und Abfragemodus

4. Formale Semantik

- 4.1 Das Typsystem von eRQL
- 4.2 Die Semantikregeln von eRQL

5. Ausblick und Zusammenfassung

6. Glossar

7. Literatur

1 Einführung und Problemstellung

Mit der globalen Vernetzung und über 3 Milliarden Webseiten kam die Informationsüberflutung - auch data smog genannt. Die klassischen Internetsuchmaschinen wie z.B. Google, AltaVista usw. sind überlastet und unzureichende Werkzeuge geworden.

Internetsuchmaschinen suchen im Volltext nach Stichwörtern und daraus resultiertes Ergebnis ist vorstellbar als eine Art gigantisches Stichwortverzeichnis für das World Wide Web. Dadurch standen den Internetsuchmaschinen ausreichende und behandlungsbedürftige Probleme gegenüber. Beispielsweise werden die Bedeutungsähnlichkeiten (Synonyme) nicht berücksichtigt. Google findet bei der Suche nach „Teein“ nur noch 1760 Fundstellen, während der Suche nach dem Synonym „Koffein“ jedoch 35800 Fundstellen. Dies gilt auch für alle andere Internetsuchmaschinen.

Ebenso werden die homonymen Wörter (wie „Java“ als Programmiersprache und „Java“ als Insel) nicht unterschieden. Weitere Probleme wie Rechtschreibfehler, Wortformen und Sinnzusammenhang können die Internetsuchmaschinen nicht erkennen.

Zusätzlich haben die Suchmaschinen noch Probleme mit nahezu allen strukturierten Informationen, welche nicht in HTML kodiert sind. Dazu zählen die Dateiformate wie z.B. PDF-, DOC-, XLS- Dateien und Dateiformate wie Postscript.

Das Ergebnis einer Suche mit der Suchmaschine ist ein vollständiges Dokument bzw. ein Verweis darauf und nicht nur die gewünschte Information.

Der Hoffnungsträger „Semantic Web“ verspricht eine Lösung für solche Probleme, die eine gewöhnliche Suchmaschine gegenwärtig nicht lösen kann.

1.1 Semantic Web auf Basis von RDF

Das Semantic Web ist der Oberbegriff für zahlreiche Standards, Techniken und Ideen zur Abfrage, Übertragung, Speicherung und Verarbeitung von Daten. Das Semantic Web auf Basis von RDF versucht das Problem der Informationsüberflutung zu lösen.

Durch die Hinterlegung zusätzlicher maschinenlesbarer *Sinnesinformationen* zu den eigentlichen Dokumenten werden Suchmaschinen oder Informationssysteme in die Lage versetzt, textuelle wie nicht- textuelle Inhalte exakter zu erfassen, nämlich, anhand einer formalsprachlichen Notation, Resource Description Framework RDF.

Es existieren zahlreiche Parser und Validierungswerkzeuge für diesen Zweck. Für die Abfrage von RDF-Daten ist eine geeignete Abfrage Sprache erforderlich. Solche Sprachen existieren schon, Beispielsweise RQL, rdfDB, RDQL, SquishQL und SeRQL.

1.2 Resource Description Framework (RDF)

RDF ist eine Empfehlung des World Wide Web Consortium [W3C] zum Austausch und zur Speicherung strukturierter wie semi-strukturierter Daten im WWW. RDF-Modell ermöglicht Vermischung von Daten verschiedener und voneinander unabhängiger Anwendungen.

RDF beschränkt sich im Wesentlichen auf zwei sehr allgemein gehaltenen Vorschriften:

1. Ein RDF-Modell ist nicht anderes als eine Menge von beliebig vielen Aussagen (siehe Glossar in 6), die in einer speziellen, maschinenlesbaren Notation formuliert sind.
2. Jede Aussage (statement) eines RDF-Modells besteht aus je genau einem Subjekt (subject), einem Prädikat (property) und einem Objekt (object).

RDF ist kein Dateiformat, allerdings sieht die Empfehlung des W3C jedoch zwei Dateiformate vor, welche zur Serialisierung von RDF verwendet werden können, um eine Speicherung in Dateiform zu ermöglichen: ein erstes Dateiformat auf Basis von XML, und ein zweites auf Basis einer einfachen Textdatei.

- Solche Abfragesprachen sind kostenlos, weit verbreitet, teilweise standardisiert und haben relativ hohe Performance.

Nachteile:

- Solche Abfragesprachen verwenden formale Notation vergleichbar mit SQL oder Programmiersprache. Für das Lernen solche Sprachen benötigt der Anwender (Abfragesteller) einen großen Lernaufwand. Also sie sind für den Abfragesteller nicht einfach und er benötigt sicherlich technisches Wissen.
- Solche Abfragesprachen unterstützen Projektionen und Selektion im Sinne der relationalen Datenbank, daher ist Exaktheit der Formulierung bei der Selektion und Projektion in diesen Sprachen erforderlich.

2.1 Bedarf an neuartige RDF-Abfragesprache

Obwohl es zahlreiche Abfragesprachen für RDF- Daten existieren, mangelt es fast in allen Abfragesprachen an die Einfachheit und die Intuitivität bei den Abfragen. Diesen Sprachen sind ziemlich entfernt von den bekannten Internetsuchmaschinen wie beispielsweise Google, wenn es um Einfachheit und Vorkenntnisse geht. Wegen der oben genannten Nachteile wird eine neue Abfragesprache für RDF entwickelt, die diese Nachteile umgeht und deren Vorteile, insbesondere hohe Performance und Verfügbarkeit, mit einbezieht.

Daher soll bei dem Entwurf von neuer Abfragesprache eine der bereits existierenden Abfragesprachen als *Zwischensprache* zu verwenden, indem die Abfrage dementsprechend zunächst in eine Abfrage dieser Zwischenabfragesprache transformiert wird und dann von dem Prozessor dieser Sprache verarbeitet.

2.2 Anforderungen an die neuartige RDF-Abfragesprache

Im Vordergrund bei der Entwicklung der neuen Abfragesprache steht der Abfragesteller (Interaktion zwischen dem Informationsportal und dem Abfragesteller). Diese Sprache soll ihn helfen, die gewünschten Informationen schnell und präzise zu finden, ohne dass er technische Vorkenntnisse mitbringen zu müssen. Man kann sagen Abfragesprache für Jeder, wie es bei der Internetsuchmaschinen bekannt ist. Im Folgenden werden die Ziele gezählt, die die Abfragesprache erfüllen soll:

1. Einfachheit

Einfache Abfragen sollten ohne spezielle Vorkenntnisse möglich sein. Eine Gewöhnung an die Abfragesprache sollte schnell sein.

2. Schemaunabhängigkeit

Abfragen müssen ohne Kenntnisse über die Struktur des RDF-Modells möglich sein

3. Mächtigkeit

Mächtige Abfragen sollten durch bottom-up-Konstruktion aus einfachen Abfragen zusammensetzbar sein.

4. Domänenunabhängigkeit

Die Abfragesprache darf nicht auf eine spezielle Domäne zugeschnitten, sondern muss in jeder Domäne gleichermaßen einsetzbar sein.

5. Erweiterbarkeit

Die Syntax der Abfragesprache soll ausreichend flexibel sein, um einen freien Raum für weitere Entwicklungen möglich zu machen.

6. Schemaunterstützung

Die Abfragesprache sollte eine Schemaunterstützung mitbringen, Beispielerweise eine Unterstützung für RDF Schema, um der Abfragesteller durch den Datenbestand eines Informationsportals navigieren zu können, ohne gezielt suchen zu müssen.

2.3 Wahl der Zwischensprache

Von allen erwähnten Anfragesprachen für RDF-Daten kommt RQL als Zwischensprache in Frage. Sie bringt die besten Voraussetzungen für die neue Abfragesprache mit. RQL (RDF Query Language) ist eine funktionale und typsichere Abfragesprache für RDF-Daten, entwickelt wurde von Greg Karvounarakis in Griechenland.

Die wichtigen Argumente, die für den Wahl von RQL als Zwischensprache sprechen, sind:

- RQL bietet eine leistungsfähige Kurzschreibweise, welche der Idee der neuen Abfragesprache sehr nahe kommt. D.h. es ist eine Kurzschreibweise ohne SQL-Syntax (SELECT-FROM-WHERE) möglich, obwohl ihre Syntax an der Syntax von SQL ähnelt.
- Da RQL eine sehr mächtige Abfragesprache ist, so es ist möglich, bei der Weiterentwicklung der neuen Sprache, die Abfragen weiter in RQL transformieren lassen.
- RQL ist gut geeignet für Schemaabfragen; ihre Kurzschreibweise ist für das Navigieren durch RDF-Modell ausreichend intuitiv und für Jeden mit minimalen Kenntnissen möglich ist. Daher wird die neue Abfragesprache keine explizite Unterstützung für Schemaabfragen beinhaltet, sondern alles für RQL überlassen. Allerdings ist die Sprache jung und noch nicht standardisiert; sie ist für eine Weiterentwicklung fällig.

3 eRQL-Ad Hoc-Abfragen für Informationsportale

eRQL ist eine Kurzschreibweise für **easy RQL** (easy RDF Query Language) und wurde in der Lehrstuhl DBIS entwickelt. eRQL ist auf der bereits vorhandenen Abfragesprache RQL aufgebaut. Ein Abfrageprozessor wurde mittels Java implementiert. eRQL- Prozessor trägt den Name **eRqlEngine**; eRQL- Prozessor wandelt eRQL- Abfragen in eine oder mehrere RQL- Abfragen um, und anschließend wird diese von einem RQL- Prozessor (wird **RqlEngine** genannt und auf Basis des RDF- Parsers **VRP** implementiert) ausgewertet (siehe Abbildung 2).

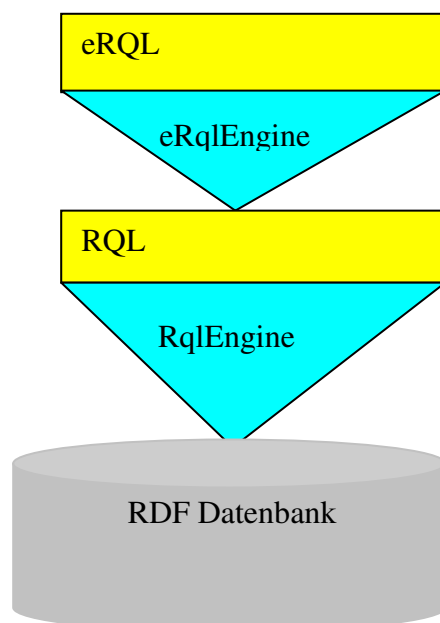


Abbildung 2- Zusammenhang zwischen eRQL und RQL

3.1 Eigenschaften von eRQL

Die wichtigsten Eigenschaften von eRQL sind grob unten gezählt; für Demonstration und Erläuterungen dieses Abschnitts wird das folgende, mittlerweile klassische, Szenario herangezogen, (siehe Abbildung 3) gegeben als RDF- Graph:

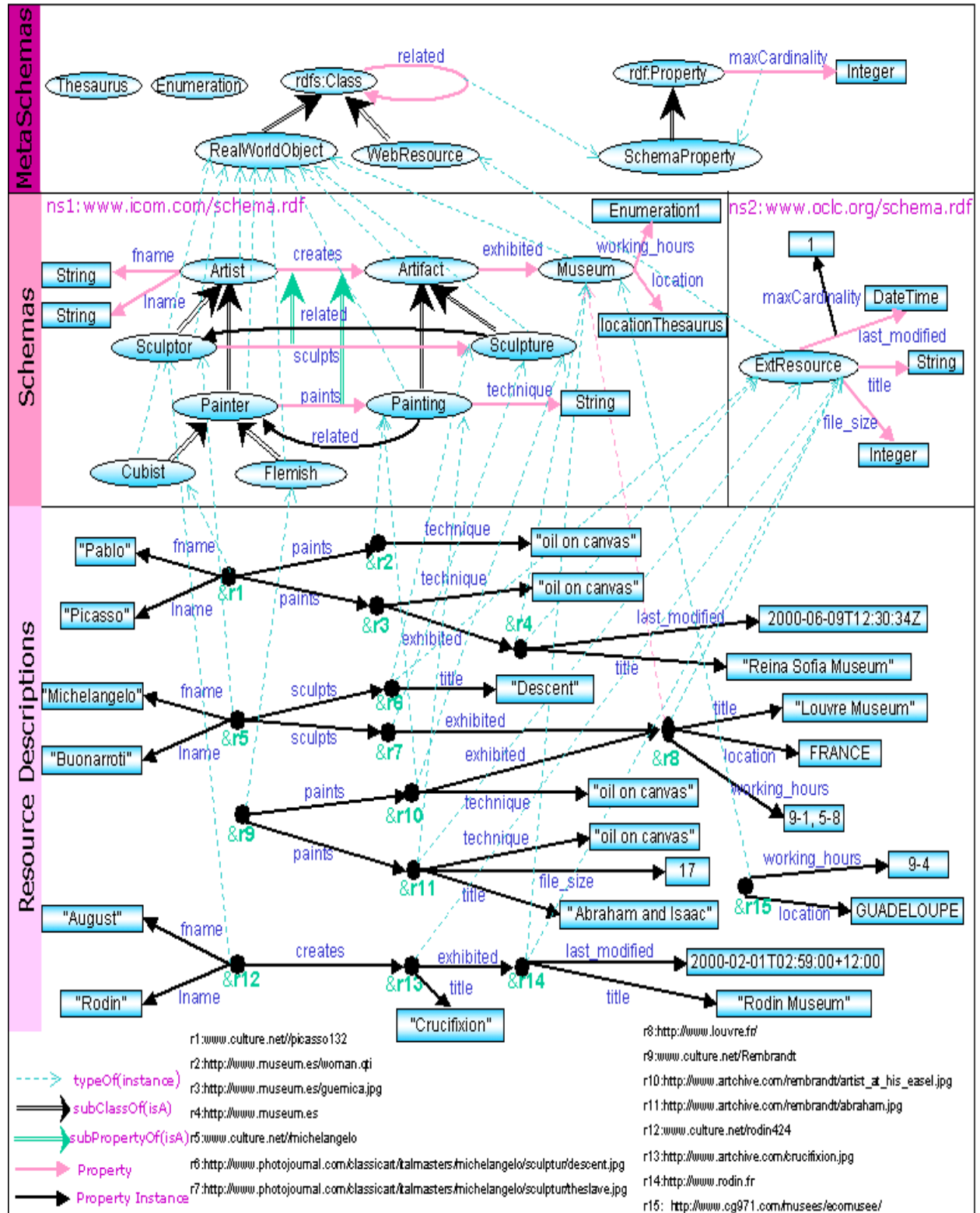


Abbildung 3- Szenario: Ein Kultur- Informationsportal

Die wichtigsten Eigenschaften sind:

- Groß-/Kleinschreibung
- Boolesche Verknüpfungen und Klammerung
- Ein –Wort -Abfrage
- Umgebung und Abfragemodus: Aussagemodus , POI-Modus und Dokumentmodus

3.1.1 Groß-/Kleinschreibung

eRQL berücksichtigt generell keine Unterschiede bezüglich der Groß-/Kleinschreibung. Dies betrifft sowohl den Vergleich von Literalen, als auch den Vergleich von Ressourcen/ URIs.

3.1.2 Boolesche Verknüpfungen

Genauso wie die Internetsuchmaschinen bekannt ist, unterstützt eRQL die boolesche Operatoren AND und OR.

3.1.3 Ein- Wort- Abfragen

eRQL erlaubt die Suche nach einem bestimmten Begriff, ähnlich wie bei Google, indem dieser als Abfrage verwendet wird; Beispielweise PICASSO.

Das Ergebnis solchen Abfragen :

- Eine Abfrage, die nur aus einem Suchbegriff besteht, liefert alle Aussagen zurück, welche diesen Suchbegriff innerhalb ihres Subjektes, Prädikates und/ oder Objektes in beliebiger Groß-/Kleinschreibung besitzen. Beispielweise PICASSO, PicAsso
- Abfragen, die aus mehreren Suchbegriffen bestehen, werden mit der Operator AND verknüpft und dann verarbeitet. Beispielweise PABLO PICASSO ↔ PABLO AND PICASSO.

Beispiel: Abfrage *Picasso* als einziges Wort. Ihre Durchführung anhand **eRqlEngine** liefert folgendes Ergebnis (siehe Abbildung 4):

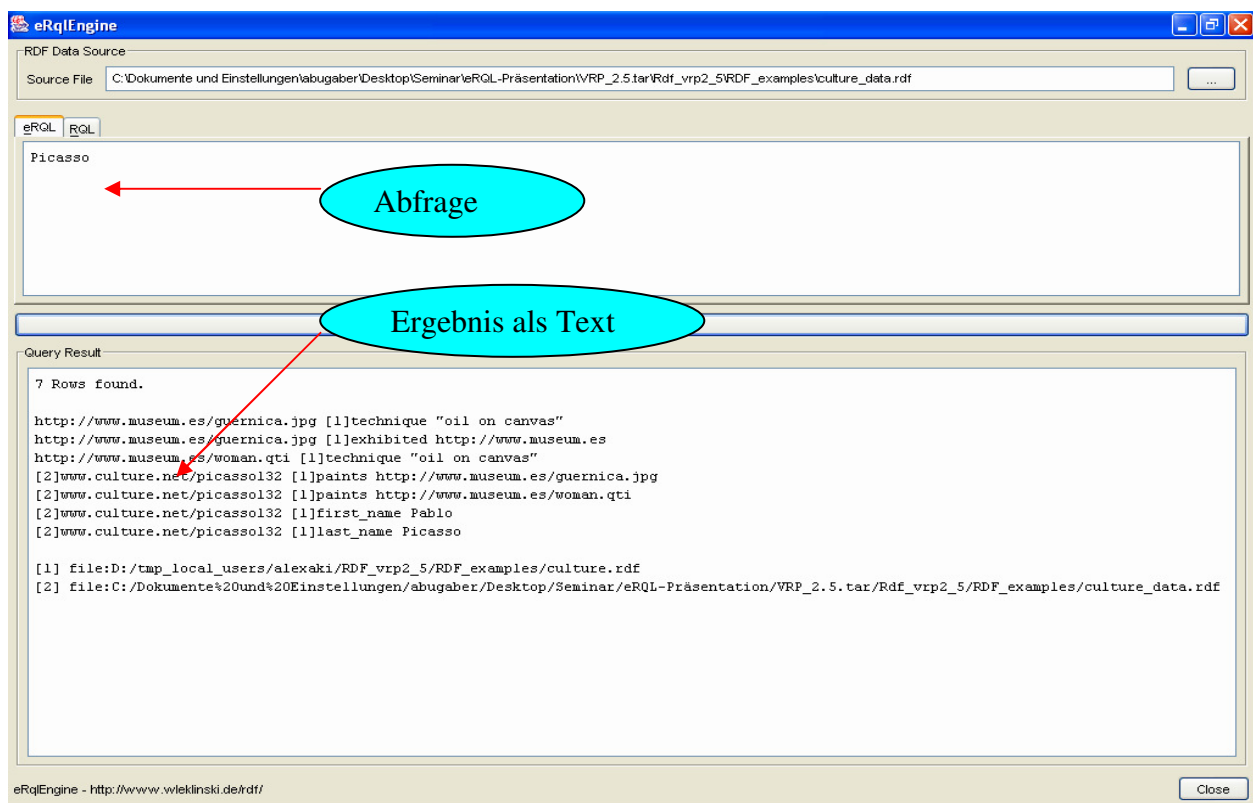


Abbildung 4- Ergebnis der Abfrage: Picasso

3.1.4 Umgebung und Abfragemodus

eRQL erweitert einzelne Ressourcen, Literale oder Aussagen automatisch um *benachbarte Aussagen*, um dem Abfragesteller zusätzliche Informationen über den Kontext der Fundstellen zurückgeben zu können. Da einzelne Aussagen, Ressourcen und Literale im Sinne einer Suche innerhalb eines Informationsportals eine zu hohe Granularität haben, erweitert eRQL daher diesen um dessen *Umgebung*. Diese Umgebung beinhaltet, neben der entsprechenden Aussage selbst, alle weiteren Aussagen des RDF-Modells, die zu dieser Aussage, bezüglich der Graphenrepräsentation, *benachbart* sind.

eRQL unterscheidet zwischen drei Arten von Abfragen, die davon die genaue Definition der Umgebung abhängig macht. Die Abfragemodus sind:

Aussagemodus

Aussagemodus wird durch den [...] -Operator aktiviert. Die *Umgebung* einer Aussage im Aussagemodus beinhaltet lediglich die Aussage selber. Die Umgebung einer Menge von Aussagen beinhaltet exakt diese Aussagenmengen an sich selbst. Die Verwendung des Aussagemodus liefert im Vergleich zu dem POI- oder Dokumentmodus die wenigsten Informationen über Fundstellen zurück.

Als Beispiel ist die Abfrage **[Picasso]**, ihre Durchführung anhand **eRqlEngine** liefert folgendes Ergebnis (siehe Abbildung 5), vier Statements:

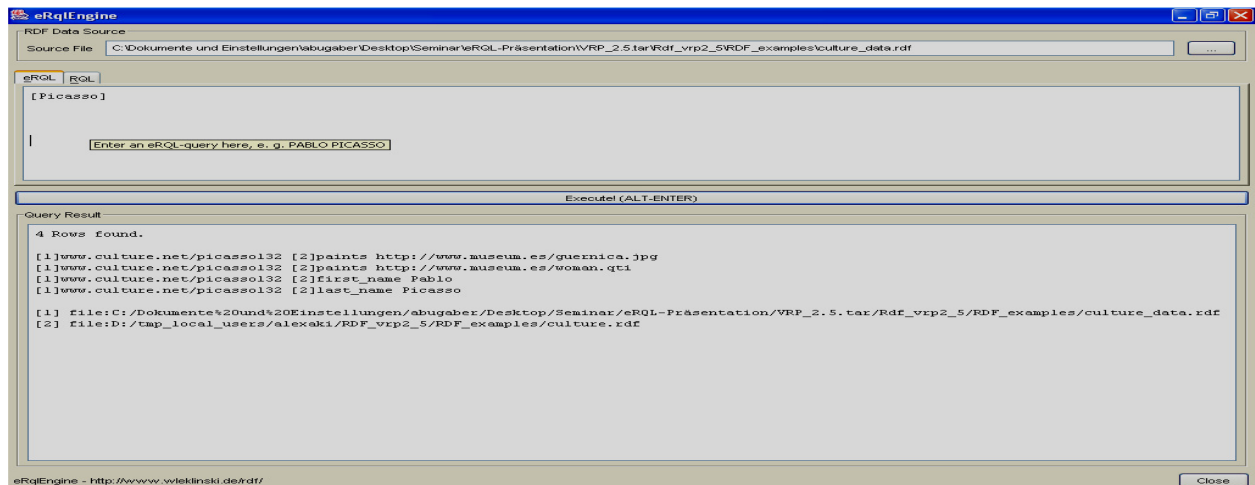


Abbildung 5- Aussagemodus- Ergebnis der Abfrage: [Picasso]

In Abbildung 6 ist das Ergebnis für die Abfrage [Picasso] graphisch dargestellt:

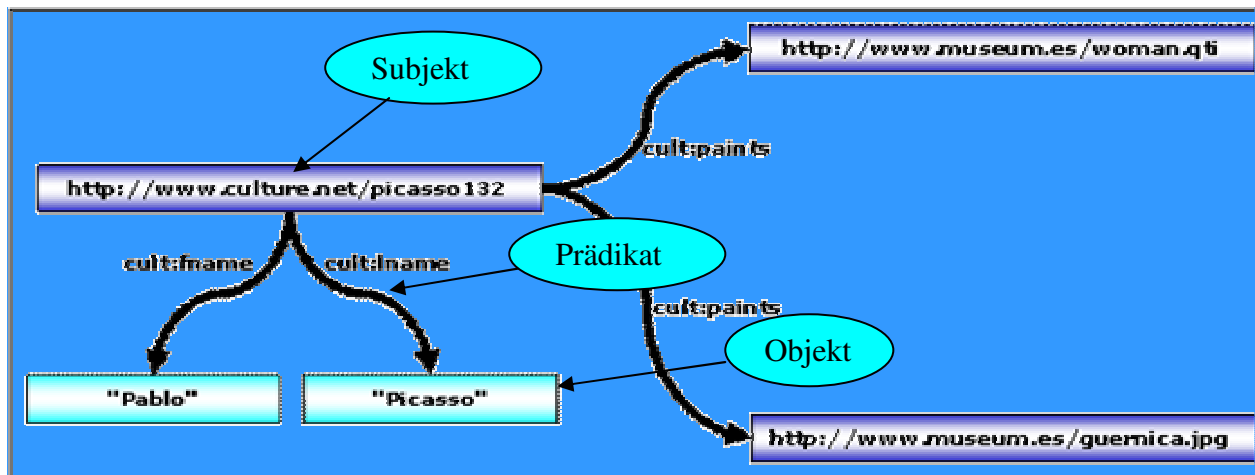


Abbildung 6- Aussagemodus- Ergebnis der Abfrage: [Picasso] in RDF Graph

Interpretation des Ergebnisses:

In dem Ergebnis sind alle Aussagen enthalten, die den Begriff PICASSO in ihrem Subjekt, Prädikat, und/oder Objekt beinhalten, ohne Berücksichtigung der Groß-/Kleinschreibung.

Point Of Interest-Modus (POI-Modus)

POI-Modus wird durch den {...}-Operator aktiviert. Die Umgebung einer Aussage im POI-Modus beinhaltet, neben der Aussage selbst, all jene Aussagen, welche diese Aussagen *berühren*.

A_1 berührt $A_2 \leftrightarrow A_i = \{s_i, p_i, o_i\}$ AND $A_1 \cap A_2 \neq \emptyset$

Die POI-Umgebung einer Ressource oder eines Literales besteht aus allen Aussagen, welche diese Ressource bzw. diese Literal in ihrem Subjekt, Prädikat und /oder Objekte beinhalten, sowie deren Umgebungen. Die Verwendung des POI-Modus liefert, im Vergleich zu dem Aussagemodus, mehr Informationen über Fundstellen zurück, jedoch weniger als der Dokumentmodus.

Bemerkung:

Im POI-Modus werden Literale automatisch ausgewertet. D.h. die Abfrage *Picasso* und *{Picasso}* sind äquivalent, ausgenommen wenn sie explizit in einem <...>- oder [...] - Operator enthalten sind.

Der POI-Modus- Operator {...} unterstützt eine einfache Notation und zwar ~ **-Operator**, als Kurzschreibweise. Daher sind die Abfragen {Picasso} \equiv Picasso äquivalent und ~Picasso \equiv {{Picasso}} \equiv ~ {Picasso} auch äquivalent.

Als Beispiel für die Verwendung von POI-Modus ist zu ermitteln, alle Informationen rund um den Begriff „Picasso“. Dafür wird folgende Abfrage getätigt:

Picasso, oder {Picasso}, oder ~[Picasso], ihre Durchführung anhand **eRqLEngine** liefert folgendes Ergebnis, (siehe Abbildung 7), sieben Statements:

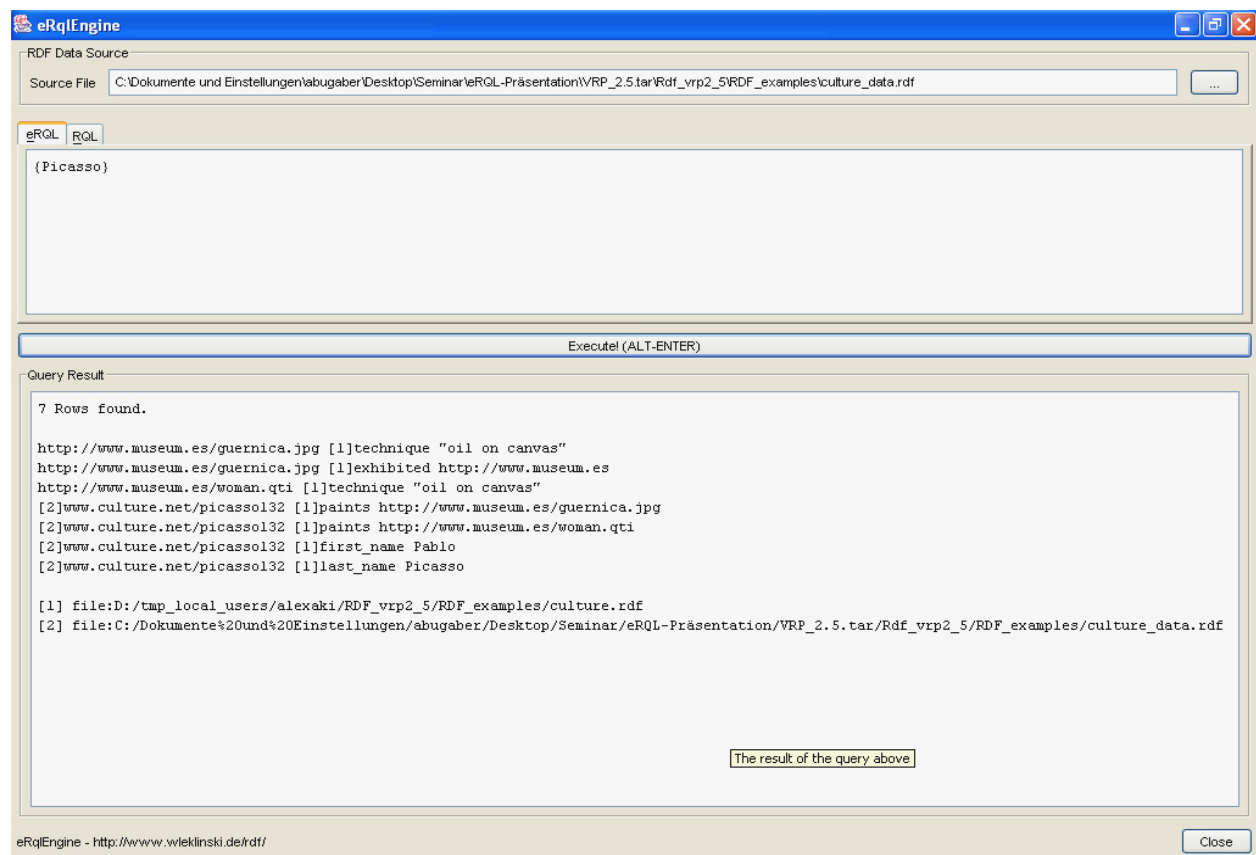


Abbildung 7- POI-Modus- Ergebnis der Abfrage: Picasso, oder {Picasso}

In Abbildung 8 ist das Ergebnis für die Abfrage **Picasso** graphisch dargestellt:

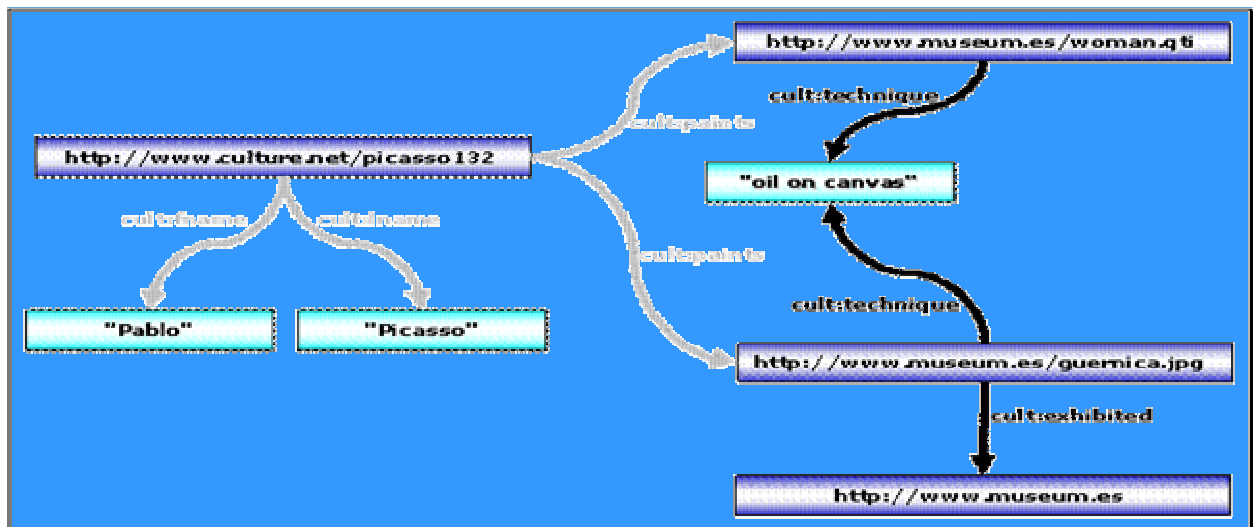


Abbildung 8- POI-Modus- Ergebnis der Abfrage: Picasso in RDF Graph

Interpretation des Ergebnisses:

Das Ergebnis dieser Abfrage enthält alle Aussagen des obigen Szenarios, welche im Aussage-modus ermittelt wurden, hinzukommen noch Aussagen, die durch Anwendung von POI-Modus-Operator gebildet sind.

Bemerkung:

Durch mehrfach Anwendung der POI- Modus-Operator lässt sich die Umgebung vergrößern, was bei den anderen Operatoren nicht machen lässt.

Erstes Beispiel: {{Picasso}} oder ~ Picasso, siehe Abbildung 9.

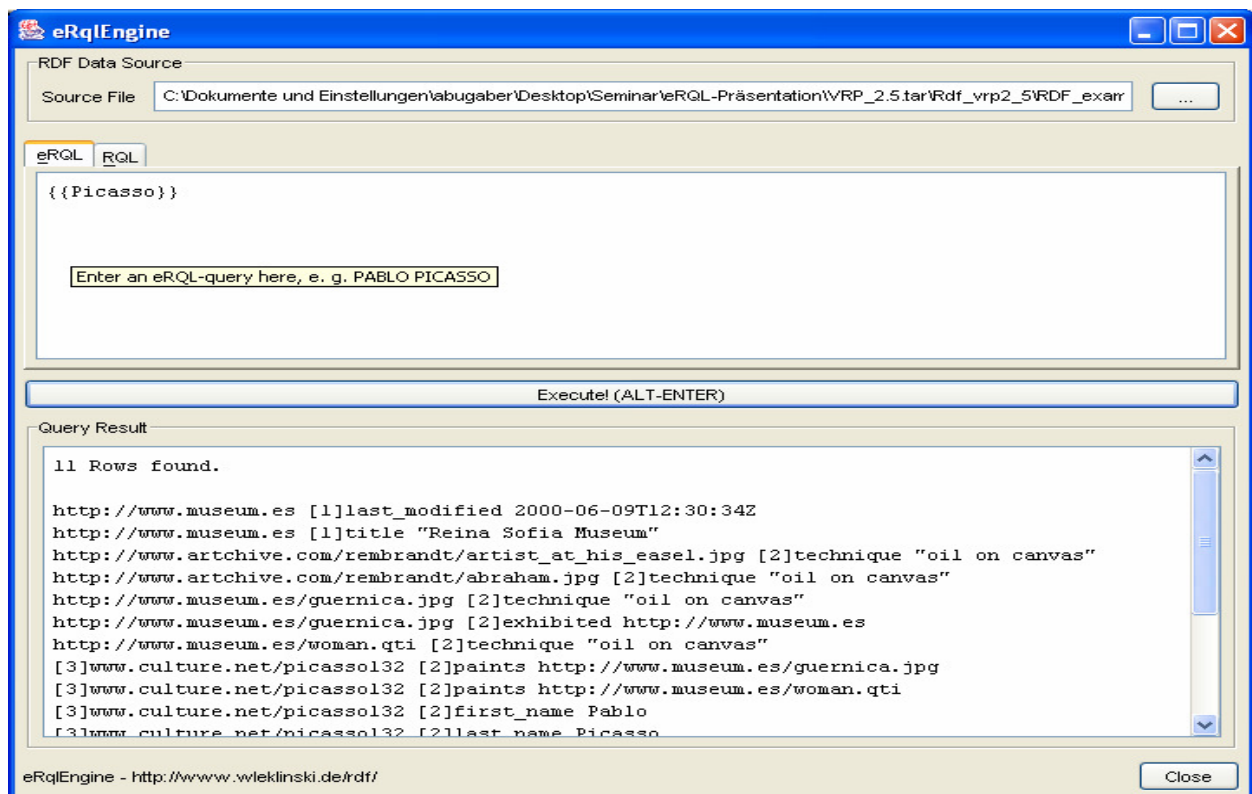


Abbildung 9- POI-Modus- Ergebnis der Abfrage: ~Picasso, oder {{Picasso}}

Zweites Beispiel: ~Picasso oder {{{Picasso}}}, siehe Abbildung 10.

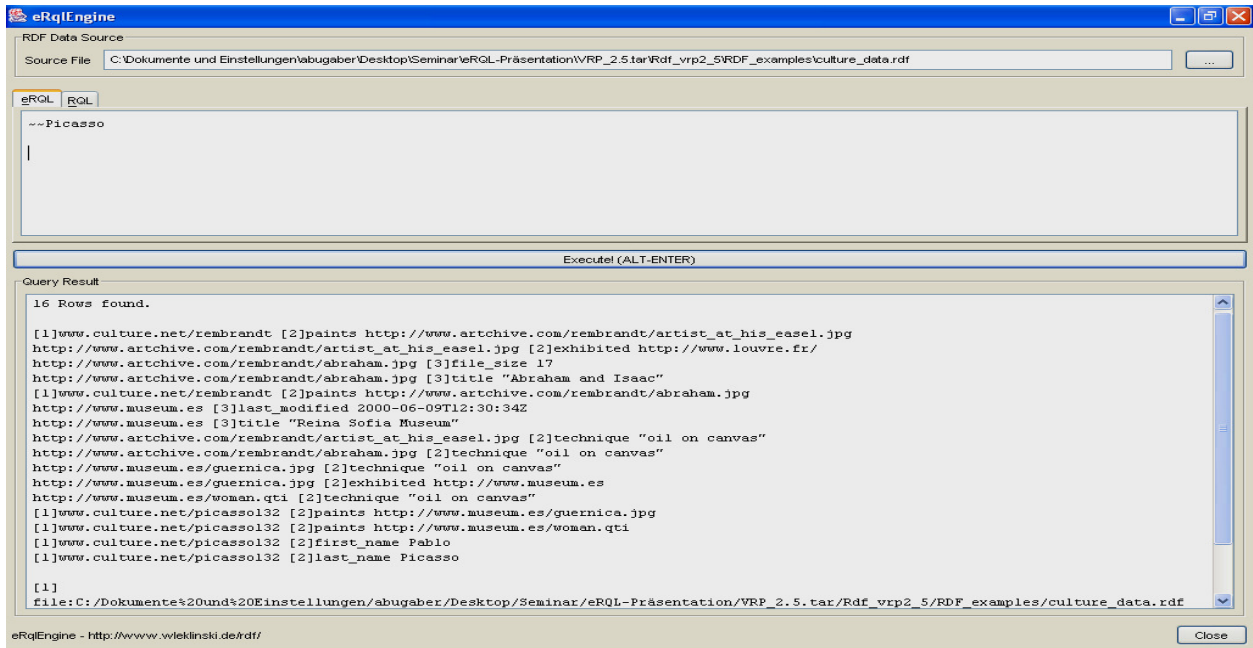


Abbildung 10- POI-Modus- Ergebnis der Abfrage: ~Picasso, oder {{{Picasso}}}

Dokumentmodus

Dokumentmodus wird durch den <...>-Operator aktiviert. Die Umgebung einer Aussage im Dokumentmodus beinhaltet, neben der Aussage selbst, all jene Aussagen, welche demselben Dokument entstammen, wie diese Aussage. Als Beispiel ist die Abfrage: <Picasso>. Ihre Durchführung anhand eRqLEngine liefert folgendes Ergebnis (siehe Abbildung 11, 30 Statements)

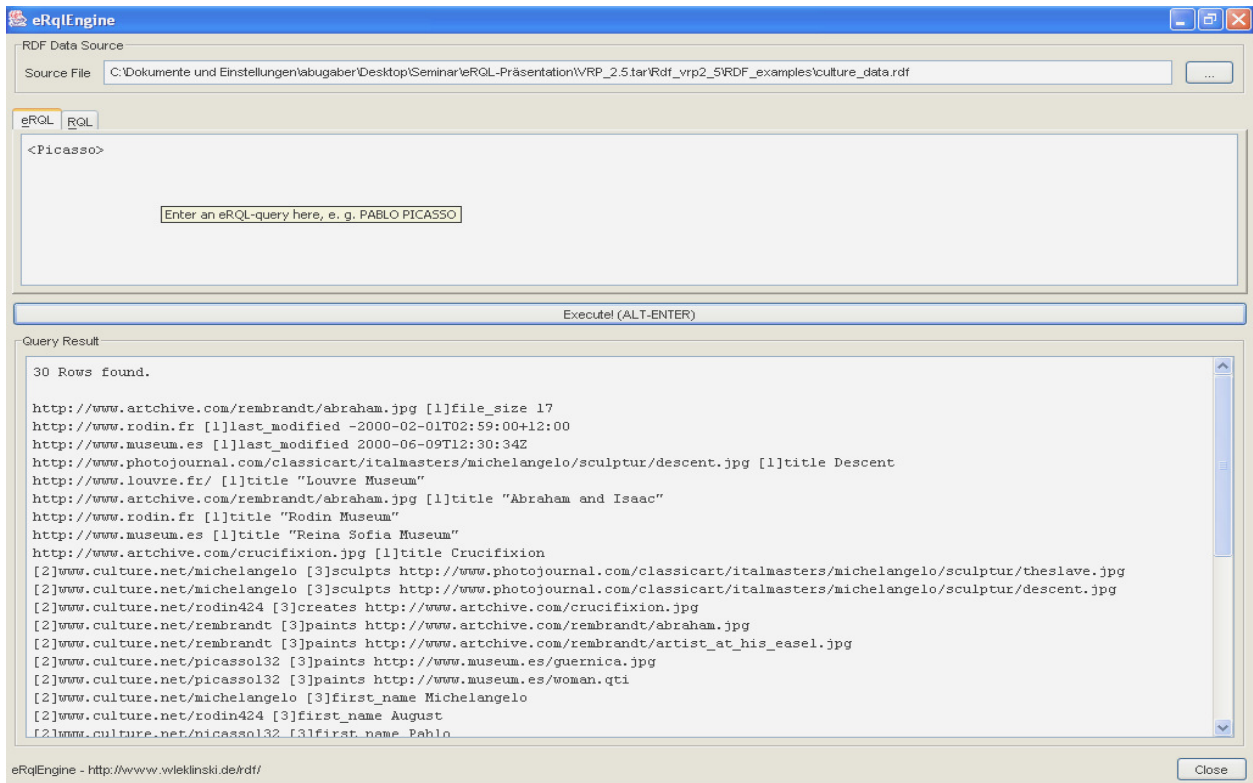


Abbildung 11- Dokumentmodus - Ergebnis der Abfrage: <Picasso>

Bis jetzt werden mittels Dokumentmodus- Operator $\langle \dots \rangle$ sämtliche Aussagen der betroffenen Dokumente zurückgegeben. In der bisherigen Situation liefert der Dokumentmodus keinerlei Informationen darüber, von welchen Dokumenten die zurückgegebenen Aussagen stammen.

Nehmen wir aber an, dass der Besucher eines Informationsportals nicht nur an der Menge der Aussagen, sondern auch an der Zugehörigkeit der jeweiligen gefundenen Aussagen interessiert ist. Für die neue Situation erwartet der Abfragesteller also nicht nur die Rückgabe der Menge der gefundenen Aussagen, sondern auch die Rückgabe von **Dokumentverweisen** z.B. in Gestalt von URIs, wie es beispielsweise bei Google üblich ist. Dafür nehmen wir weiterhin an, es existieren zwei Dokumente d_1 , und d_2 , die den Suchbegriff „Picasso“ enthalten, wobei S_i , P_i und O_i für Subjekt, Prädikat und Objekt stehen. Die modifizierte eRQL sollte dann für die Abfrage $\langle \text{Picasso} \rangle$ folgendes Ergebnis ausgeben:

(S1, P1, O1, d_1), (S1, P2, O2, d_1), (S2, P2, O2, d_1),...

...

(S1, P1, O1, d_2), (S3, P4, O6, d_2), (S2, P3, O2, d_2),...

...

Für diese Situation sollte in eRQL den Dokumentmodus so zu abwandeln, dass für jede Fundstelle sowohl die betroffenen Aussagen, als auch die URIs der Dokumente, welche sie beinhalten, zurückzugeben.

Es gibt verschiedene Gründe, warum Dokumentverweis des betroffenen Dokuments zurückgegeben werden soll; beispielsweise um Glaubwürdigkeit- Test zu ermöglichen, oder den Suchraum für weitere, möglicherweise neue, Informationen zu erweitern. Ein Ergebnis auf die Abfrage $\langle \text{Aktienanalyse} \rangle$, die neben sämtlichen Aussagen der betroffenen Dokumente auch ihren jeweiligen Dokumentenverweisen (URIs) enthält, ermöglicht den Abfragesteller die Validität des Dokumentes beziehungsweise den Aussagen zu testen. Ein URI von deutsche Börse ist ernsthafter zunehmen als ein URI von XY- Student. Auch die Aktualität des Dokuments ist ein wichtiger Aspekt bei der Informationssuche.

4 Formale Semantik

Unter formale Semantik von eRQL ist die Beschreibung der Regeln für ein eRQL- Prozessor zu verstehen, nach denen eine gültige eRQL- Abfrage in ein Abfrageergebnis umgewandelt werden kann. Es wird eine formal sprachliche Notation zur Formulierung der Regeln verwendet.

4.1 Das Typsystem von eRQL

In diesem Dokument werden Symbole wie m , D und s verwendet, um bestimmte Dinge wie ein Datenmodell, den Dokument-Typ oder eine Aussage zu bezeichnen. Es sind weiterhin Datentypen wie *Dokument*, *Aussagen*, etc nötig, um einen Zusammenhang wie „ein Dokument besteht aus Aussagen“ formal ausdrücken zu können. Datentypen werden durch Großbuchstaben bezeichnet, z.B. die Datentypen D für *Dokument*, A für *Aussage*. Der Ausdruck $d \in D$ bedeutet z.B., dass d ein Dokument ist.

4.1.1 Datenmodell

Der Begriff *Datenmodell* (m) beschreibt die Menge aller Daten, welche einem Informationsportal zur Verfügung stehen, und abgefragt werden können.

Ein Datenmodell $m \in M$ enthält beliebig viele Dokumente (d_1, \dots, d_n), welche den einzelnen Dateien, RDF-Datenbanken, oder anderen Datenquellen entsprechen, M ist die Menge aller Datenmodelle. Formal ausgedrückt:

$$\forall m \in M \Rightarrow m = \{ d_1, \dots, d_n \} \quad n \geq 0 \quad \forall 0 \leq i \leq n: d_i \in D \quad (4.1)$$

4.1.2 Dokument

Ein *Dokument* (d) enthält beliebig viele Aussagen (a), und dient als logisches Strukturelement. D ist die Menge aller Dokumente. Das Dokument besitzt keine Entsprechung in RDF, A ist die Menge aller Aussagen.

$$\forall d \in D \Rightarrow d = \{a_1, \dots, a_n\}, n \geq 0 \quad \forall 0 \leq i \leq n: a_i \in A \quad (4.2)$$

4.1.3 Aussagengruppen

Eine *Aussagengruppe* (g) ist die Zusammenfassung beliebig viele Aussagen zu einem logischen Verbund und dient ausschließlich zur Realisierung der POIs. G ist die Menge aller Aussagengruppen.

$$\forall g \in G \Rightarrow g = \{a_1 \in A, \dots, a_n \in A\} \quad (4.3)$$

4.1.4 Aussage

Eine *Aussage* (a) entspricht einer RDF-Aussage und besteht aus einem Subjekt, einem Prädikat und einem Objekt. Subjekt und Prädikat sind eine *Ressource* (r), aber das Objekt ist eine *Ressource* (r) oder ein *Literal* (l). A ist die Menge aller Aussagen, R ist Menge aller Ressourcen, L ist die Menge aller Literale.

$$\forall a \in A \Rightarrow a = (s, p, o), s, p \in R, o \in R \cup L \quad (4.4)$$

Durch die Funktion *statement(.)* kann, zu einem gegebenen *Dokument* d oder *Aussagengruppe* g , die Menge der enthaltenen Aussagen bestimmt werden:

$$\text{Statement}(d) := \{a_1, \dots, a_n\}, d \in D \quad d = \{a_1, \dots, a_n\} \quad (4.5)$$

$$\text{Statement}(g) := g \in G \quad g = \{a_1, \dots, a_n\}$$

4.1.5 Ressourcen und Literale

Ressourcen (r) und *Literale* (l) sind die kleinsten Einheiten der RDF-Modell und repräsentieren Objekte der realen Welt, wie Personen, Tätigkeiten, etc. und Objekte der virtuellen Welt, wie Websites, E-Mail-Adressen, etc.

$$\forall r \in R \Rightarrow r \text{ ist eine URI} \quad (4.6)$$

$$\forall l \in L \Rightarrow l \text{ ist eine Literal, } R \cap L = \emptyset \quad (4.7)$$

Mittels der Funktionen *values()* und *literals()* können zu einem gegebenen Dokument die Menge der enthaltenen Ressourcen bzw. Literale bestimmt werden, mittels *resources()* Ressourcen und Literale:

$$\begin{aligned} \text{resources}(a) &:= \{s, p, o\} \quad a \in A \quad a = (s, p, o) \quad o \in R \\ \text{resources}(a) &:= \{s, p\} \quad a \in A \quad a = (s, p, o) \quad o \in L \\ \text{resources}(g) &:= \bigcup_i \text{resources}(a_i) \quad g \in G \quad g = \{a_1, \dots, a_n\} \\ \text{literals}(a) &:= \{o\} \quad a \in A \quad a = (s, p, o) \quad o \in R \\ \text{literals}(a) &:= \{o\} \quad a \in A \quad a = (s, p, o) \quad o \in L \\ \text{literals}(g) &:= \bigcup_i \text{literals}(a_i) \quad g \in G \quad g = \{a_1, \dots, a_n\} \\ \text{values}(a) &:= \{s, p, o\} \quad a \in A \quad a = (s, p, o) \\ \text{values}(g) &:= \bigcup_i \text{values}(a_i) \quad g \in G \quad g = \{a_1, \dots, a_n\} \end{aligned} \quad (4.8)$$

4.2 Die Semantikregeln von eRQL

Die Semantikregeln von eRQL sind die Regeln, nach welchen eRQL-Abfragen zu Abfragergebnissen ausgewertet werden.

Infolgedessen werden Regeln für URIs und Literale, für boolesche Verknüpfungen und Klammerung und schließlich Regeln für Umschaltung des jeweiligen Modus mittels Klammerung gegeben. Alle Regeln verwenden eine formalsprachliche Notation: WENN- DANN- Notation oder Voraussetzung und Folgerung, wobei die Folgerung tritt ein, wenn jene Voraussetzung erfüllt ist und notiert als:

$$\frac{\text{Voraussetzung}}{\text{Folgerung}}$$

• URIs und Literale

Eine URI oder Literale, *LiteralOrUri*, wird zu der Menge aller Aussagen ausgewertet, welche diese URI bzw. dieses Literal enthalten:

$$\frac{\{a_1, \dots, a_n\} : \{A\} = \{a_i \mid \text{LiteralOrUri} \in a_i \in A\}}{\text{LiteralOrUri} : L \cup R \Rightarrow \{\{a_1 : A\}, \dots, \{a_n : A\}\} : \{G\}} \quad (4.9)$$

• Boolesche Verknüpfungen und Klammerungen:

$$\frac{\begin{array}{l} \text{Conjunction}_1 \Rightarrow \{g_1, \dots, g_{i-1}\} : \{G\} \\ \text{Conjunction}_2 \Rightarrow \{g_j, \dots, g_n\} : \{G\} \end{array}}{\text{Conjunction}_1 \text{ OR } \text{Conjunction}_2 \Rightarrow \left\{ \bigcup_i \text{statements}(g_i) \right\} : \{G\}} \quad (4.10)$$

$$\frac{\begin{array}{l} \text{SubQuery}_1 \Rightarrow \{g_1, \dots, g_r\} : \{G\} \\ \text{SubQuery}_2 \Rightarrow \{h_1, \dots, h_s\} : \{G\} \end{array}}{\text{SubQuery}_1 \text{ AND } \text{SubQuery}_2 \Rightarrow \left\{ \bigcup \text{statements}(g_i) \cup \text{statements}(h_i) \right\} : \{G\} \left. \begin{array}{l} \\ g_i \cap h_i \neq \emptyset \end{array} \right\}} \quad (4.11)$$

$$\frac{\text{Disjunction}_1 \Rightarrow \{g_1, \dots, g_n\} : \{G\}}{(\text{Disjunction}_1) \Rightarrow \{g_1, \dots, g_n\} : \{G\}} \quad (4.12)$$

• Umschaltung des Modus mittels Klammerung:

$$\frac{\begin{array}{l} \text{Disjunction} \Rightarrow \{g_1, \dots, g_m\} : \{G\} \\ \{a_1 : A, \dots, a_n : A\} = \bigcup_i \text{statements}(g_i) \end{array}}{[\text{Disjunction}] \Rightarrow \{\{a_1 : A\}, \dots, \{a_n : A\}\} : \{G\}} \quad (4.13)$$

$$\begin{array}{l}
\text{Disjunction} \Rightarrow \{g_1, \dots, g_m\} : \{G\} \\
\{a_1, \dots, a_n\} = \left\{ a_i \mid a_i \in A \quad \exists b \in \bigcup_j \text{statements}(g_j) : b \cap a_i \neq \emptyset \right\} \\
\hline
\{\text{Disjunction}\} \Rightarrow \{a_1, \dots, a_n\}
\end{array} \tag{4.14}$$

$$\begin{array}{l}
\text{Disjunction} \Rightarrow \{g_1 : G, \dots, g_n : G\} \\
\{d_1, \dots, d_m\} = \left\{ d_i \mid d_i \in D \quad \text{values}(d_i) \cap \bigcup_i \text{statements}(g_i) \neq \emptyset \right\} \\
\hline
\langle \text{Disjunction} \rangle \Rightarrow \bigcup_{d_i} \text{statements}(d_i)
\end{array} \tag{4.15}$$

5. Ausblick und Zusammenfassung

eRQL ist eine intuitive aber auch leistungsfähige Abfragesprache für Informationsportale. Sie erfüllt die Anforderungen in 2.2, einfach, mächtig, Domänen und Schema unabhängig, und erweiterbar.

eRQL ist so konstruiert, dass die Suchabfragen aus dem gesuchten Begriff bestehen, wie es vielen Menschen von Suchmaschinen bekannt ist. Aber auch lassen sich komplizierten Abfragen durch einfaches Hintereinanderschreiben der gesuchten Begriffe durchführen. Der Anwender von eRQL benötigt keinerlei Schemakennnisse; er muss nicht spezifizieren, ob der gesuchte Begriff als Subjekt, Prädikat oder Objekt verwendet werden muss, oder ob es um Groß-, Kleinschreibung handelt. Die Syntax von eRQL ist so spezifiziert, dass die eRQL-Abfragen ohne SELECT-FROM-WHERE formuliert werden können.

Da einzelne Aussagen, Ressourcen und Literale im Sinne einer Suche innerhalb eines Informationsportals eine zu hohe Granularität haben, erweitert eRQL daher diesen um dessen Umgebung. Diese Umgebung beinhaltet, neben der entsprechenden Aussage selbst, alle weiteren Aussagen des RDF-Modells, die zu dieser Aussage, bezüglich der Graphenrepräsentation, benachbart sind. eRQL unterscheidet zwischen drei Arten von Abfragen, Aussagemodus, POI-Modus und Dokumentmodus. Die Verwendung des Dokumentmodus liefert, im Vergleich zu dem Aussagemodus und POI-Modus, die meisten Informationen über Fundstellen zurück.

Mittels Dokumentmodus- Operator $\langle \dots \rangle$ werden nur sämtliche Aussagen der betroffenen Dokumente zurückgegeben, ohne den Abfragesteller zu informieren, von welchen Dokumenten sie stammen. Dabei wäre es wünschenswert, nicht nur die Rückgabe der Menge der gefundenen Aussagen, sondern auch die Rückgabe von Dokumentverweisen, z.B. in Gestalt von URIs, wie es beispielsweise bei Google üblich ist. Für diese Situation ist eRQL noch nicht soweit; denn es wird für eine Abfrage nur die Menge von Aussagen zurückgegeben.

6. Glossar

Wichtige Begriffe und Terminologie

Im Folgenden werden spezielle Begriffe des Semantic Web erläutert, die einer offiziellen Terminologie entstammen und für spätere Erklärungen notwendig sind.

Aussage (statement)

Eine Aussage entspricht der Kombination von Subjekt, Prädikat und Objekt, und stellt einen Sachverhalt dar.

Container (container)

Ein Container stellt gemäß der RDF-Terminologie eine spezielle Ressource dar, welche beliebig viele andere Ressourcen beinhaltet. Es gibt drei Arten von Container: Alternativen, Folgen, und Sammlungen.

Definitionsbereich (domain)

Der Definitionsbereich eines Prädikates ist die Menge der RDF-Ressourcen, welche Subjekt für dieses Prädikat sein dürfen, d.h. von welchen dieses Prädikat ausgehen darf. Der Definitionsbereich wird durch das Prädikat `rdfs: domain` spezifiziert.

Folge (sequence)

Eine Folge repräsentiert eine geordnete Menge beliebige RDF-Ressourcen.

Kante (edge)

Eine Kante repräsentiert ein Prädikat in der Graphendarstellung eines RDF-Modells, und verbindet eine Ressource mit einem Literal, oder mit einer weiteren Ressource. Kanten sind in der Graphendarstellung gerichtet, d.h. verfügen über einem expliziten Start- und Endknoten, welche im Allgemeinen nicht vertauscht werden können, ohne das Modell zu modifizieren.

Klasse (class)

Eine Klasse repräsentiert eine *class* im Sinne von RDF Schema, d.h. eine Menge gleichartiger RDF-Ressourcen.

Knoten (node)

Ein Knoten repräsentiert eine Ressource oder ein Literal in der Graphendarstellung eines RDF-Modells. Zwei Knoten können durch eine Kante (d.h. ein Prädikat) miteinander verbunden werden.

Literal (literal)

Ein Literal ist neben einer Ressource die zweite Möglichkeit, einer Aussage ein Objekt zuzuweisen. Während es sich bei Ressourcen um Dinge handelt, welche durch weitere Aussagen näher beschrieben werden können, sind Literale einfache Texte, Ziffern oder Datumsangaben.

Objekt (object)

Das Objekt ist diejenige Komponente einer RDF-Aussage, welche Endpunkt des Prädikates ist. In der Graphendarstellung eines RDF-Modells entspricht dies dem Endknoten einer jeden Kante, in der Tripeldarstellung der dritten Komponente eines jeden Tripels.

Prädikat (property)

Das Prädikat ist diejenige Komponente einer RDF-Aussage, welche Subjekt und Objekt miteinander verbindet. In der Graphendarstellung eines RDF-Modells entspricht dies der Kante zwischen je zwei Knoten, in der Tripeldarstellung der zweiten Komponente eines jeden Tripels.

Ressource (resource)

Eine Ressource ist der Oberbegriff für alles, was sich mittels RDF beschreiben lässt, und zur Beschreibung verwendet werden kann. Dabei kann es sich um beliebige Sachen aus der realen Welt handeln (Bücher, Produkte, Firmen,...), Sachen aus der virtuellen Welt (Websites, E-Mail-Adressen,...), Personen, oder beliebige andere Dinge.

Sammlung (bag)

Eine Sammlung repräsentiert eine nicht geordnete Menge beliebige RDF-Ressourcen.

Subjekt (subject)

Das Subjekt ist diejenige Komponente einer RDF-Aussage, welche Ausgangspunkt des Prädikates ist. In der Graphendarstellung eines RDF-Modells entspricht dies dem Startknoten einer jeden Kante, in der Tripeldarstellung der ersten Komponente eines jeden Tripels.

Tripeldarstellung (N-Triples)

Die Tripeldarstellung ist ein zeilenweises und textbasiertes Speicherformat für RDF-Modelle. Jede Zeile entspricht dabei einer einzelnen Aussage, d.h. ein RDF-Modell mit n Aussagen wird also zu einer Textdatei mit n Zeilen. Jede Zeile enthält dabei die drei Komponenten Subjekte, Prädikat und Objekt, in dieser Reihenfolge, separiert durch Leerraum.

Unterklasse (subclass)

Eine Unterklasse repräsentiert eine so genannte *subclass* im Sinne von RDF Schema, d.h. eine Spezialisierung einer anderen Klasse.

Unterprädikat (subproperty)

Ein Unterprädikat repräsentiert ein *subproperty* im Sinne von RDF Schema, d.h. eine Spezialisierung eines anderen Prädikats.

Wertebereich (range)

Der Wertebereich eines Prädikates entspricht der Menge der RDF-Ressourcen, welche dieses Prädikat als Objekt verwenden darf, d.h. zu welchen dieses Prädikat hinführen darf. Der Wertebereich wird durch das Prädikat *rdfs: rang* spezifiziert.

Berühren

Diese begriff ist nicht Bestandteil der offiziellen RDF-Terminologie.

Zwei Aussagen A1 und A2 berühren sich, wenn mindestens eine der Ressourcen bzw. Literale, die in A1 enthalten sind, auch in A2 enthalten ist.

Dokument

Diese begriff ist nicht Bestandteil der offiziellen RDF-Terminologie

Ein Dokument repräsentiert einen logischen Verbund von Aussagen, welche bezüglich ihrer Lokalität zusammengehören. Dabei kann es sich z.B. um eine RDF-Datei handeln, um eine spezielle RDF-Datenbank, aber auch um eine Internet-Domain.

Point Of Interest

Diese begriff ist nicht Bestandteil der offiziellen RDF-Terminologie. Ein Point Of Interest (POI) ist eine spezielle Art von Umgebung um eine oder mehrere Ressourcen (oder Literale). Die POI-Umgebung beinhaltet dabei all jene Ressourcen und Literale, welche in der Graphenrepräsentation des RDF-Modells zu den vorgegebenen Ressourcen bzw. Literalen benachbart sind.

Umgebung

Diese begriff ist nicht Bestandteil der offiziellen RDF-Terminologie.

Da einzelne Aussagen, Ressourcen und Literale im Sinne einer Suche innerhalb eines Informationsportals eine zu hohe Granularität haben, erweitert eRQL daher diesen um dessen Umgebung. Diese Umgebung beinhaltet (neben die Aussage selbst) alle Aussagen des RDF-Modells, die zu dieser Aussage, bezüglich der Graphenrepräsentation, benachbart sind.

7. Literaturverzeichnis

Diplomarbeit von Fabian Wleklinski
Suche im Semantic Web /2003

Verfügbar unter:

<http://www.dbis.informatik.uni-frankfurt.de/~tolle/RDF/eRQL/>