



Einführung in die XML Codierung von RDF

Einleitung

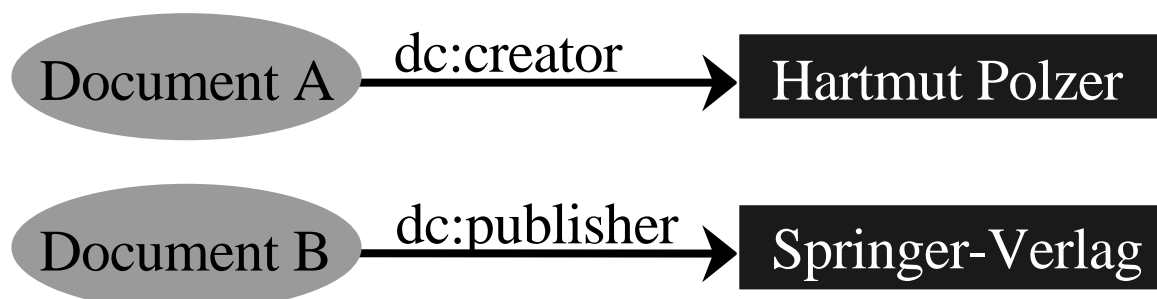
Vorausgesetzt wird im folgenden die Kenntnis, wie selbst komplexe Sachverhalte durch RDF-Graphen repräsentiert werden können. Diese Art der Repräsentation von RDF hat jedoch aufgrund der enthaltenen graphischen Elemente im Vergleich zu textorientierten Formaten deutliche Nachteile. Obwohl sich die Darstellung von Aussagen mit Hilfe von Graphen sehr intuitiv gestaltet, ist sie nur mit vergleichsweise großem Aufwand maschinenverstehbar.

Als Ausweg soll hier nun in eine weitere RDF-Repräsentationsform eingeführt werden: Neben Tripelschreibweise und Graphen-Darstellung lassen sich im RDF formulierte Aussagen auch innerhalb eines XML-Dokuments unterbringen.

Bei dieser Einführung soll anhand von Beispielen ausgehend von einfachen Statements schrittweise hin zu komplexen Strukturen (z.B. Reifizierung) vorgestellt werden, wie einzelne RDF-Konstrukte mit Hilfe der XML-Syntax darstellbar sind. Dabei werden auch einige alternative Darstellungsmöglichkeiten innerhalb der XML-Syntax diskutiert. Die hier gezeigten Möglichkeiten erheben keinen Anspruch auf Vollständigkeit, mit ihnen lassen sich jedoch sämtliche RDF-Konstruktionen realisieren.

back to the roots: einfache Statements

Um in RDF formulierte Aussagen in ein XML-Dokument einzubinden, muß man sich eines geeigneten Vokabulars bedienen. Zu diesem Zweck steht ein jedem RDF-Parser bekannter Namespace zur Verfügung, der dennoch innerhalb des RDF-Abschnittes des XML-Dokumentes deklariert werden muß.



```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.0/" >

  <rdf:Description about="http://www.iuk-initiative.de/iwi/Document_A">
    <dc:creator>Hartmut Polzer</dc:creator>
  </rdf:Description>

  <rdf:Description about="http://www.iuk-initiative.de/iwi/Document_B">
    <dc:publisher>Springer-Verlag</dc:publisher>
  </rdf:Description>

</rdf:RDF>

```

Die ersten drei Zeilen bilden zusammen mit der letzten gewissermaßen den Rahmen für die eigentliche Aussagen

„Hartmut Polzer ist der Autor von http://www.../Document_A.“
 und „Springer-Verlag ist der Publisher von http://www.../Document_B.“

Durch die Anweisung `Description` wird jeweils ein neuer Knoten generiert, der im folgenden näher beschrieben wird. Das Attribut `about` ist optional. Fehlt es, wird eine anonyme Resource generiert, über die dann jedoch außerhalb des aktuellen `Description`-Blockes keine zusätzlichen Aussagen mehr gemacht werden können. Um sich dagegen auch außerhalb des `Description`-Blockes auf die neue Resource beziehen zu können, wird ihr durch das Attribut `ID="..."` explizit ein Name zugewiesen.

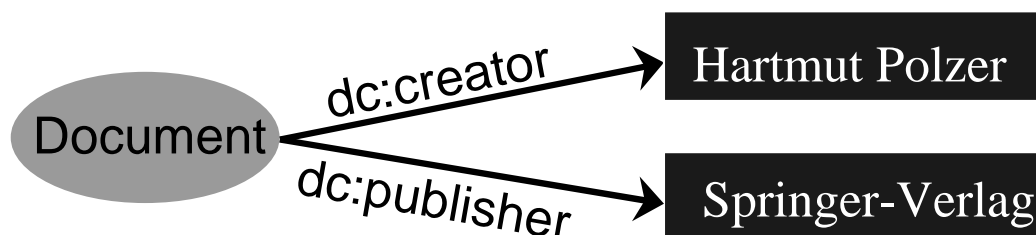
Achtung: Von zwei Ausnahmen abgesehen, die in dieser Einführung später noch thematisiert werden, gilt stets, dass in RDF keine Konstruktion zulässig ist, bei der im RDF-Graphenmodell mehr als ein Pfeil auf eine anonyme Resource zeigt.

Die Prädikate `creator` und `publisher` werden dem Dublin-Core-Vokabular entnommen, das vorher im Kopf des RDF-Blockes deklariert wurde.

Für ein RDF-Dokument ist die Reihenfolge, in der Ressourcen in einem XML-Dokument eingeführt werden, irrelevant. Die einzige Ausnahme von dieser Regel wird später bei der Container-Klasse "Sequence" besprochen.

Mehrere Statement beziehen sich auf dieselbe Resource

Beziehen sich mehrere Statements auf dieselbe Resource, gibt es verschiedene äquivalente Arten der XML-Codierung:



Neben der im ersten Beispiel dargestellten Version, bei der sich nun beide Description-Blöcke durch den Parameter about auf dieselbe Resource beziehen würden, gibt es hier eine weitere Möglichkeit der Beschreibung:

```
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:creator>Hartmut Polzer</dc:creator>
  <dc:publisher>Springer-Verlag</dc:publisher>
</rdf:Description>
```

Wie oben gezeigt, lassen sich auch mehrere Statements in einem Description-Block unterbringen, wobei auch hier die Reihenfolge der einzelnen Zeilen keine Rolle spielt. Bezieht man sich nicht auf eine externe Resource, sondern generiert eine eigene, gibt es ebenfalls verschieden äquivalente Schreibweisen:

a)

```
<rdf:Description ID="mein_Dokument">
  <dc:creator>Hartmut Polzer</dc:creator>
  <dc:publisher>Springer-Verlag</dc:publisher>
</rdf:Description>
```

b)

```
<rdf:Description ID="mein_Dokument">
  <dc:creator>Hartmut Polzer</dc:creator>
</rdf:Description>

<rdf:Description about="#mein_Dokument">
  <dc:publisher>Springer-Verlag</dc:publisher>
</rdf:Description>
```

c)

```
<rdf:Description ID="mein_Dokument"/>

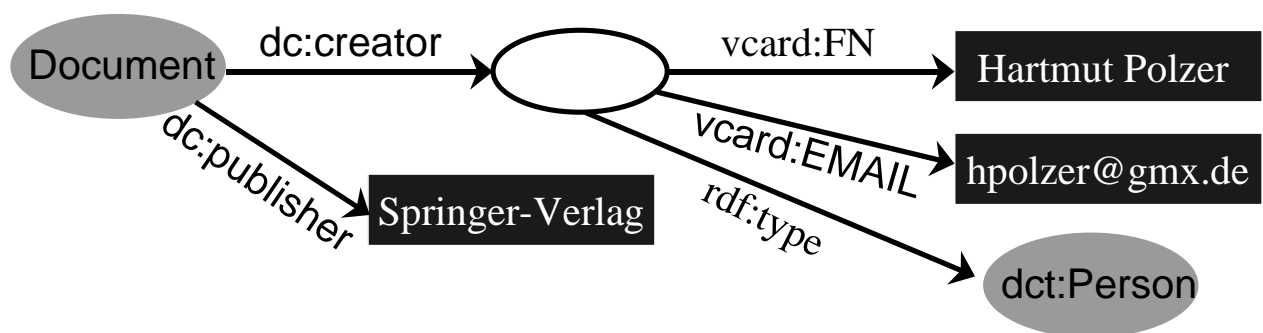
<rdf:Description about="#mein_Dokument">
  <dc:creator>Hartmut Polzer</dc:creator>
  <dc:publisher>Springer-Verlag</dc:publisher>
</rdf:Description>
```

In a) wird die Verwendung des Attributes ID gezeigt. Hierbei werden über die neu generierte Resource, die den Namen *mein_Dokument* zugewiesen bekommt, zwei Aussagen gemacht. Ist eine Resource erst einmal mit einem Namen versehen worden, so lassen sich wie in b) auch ausserhalb dieses Description-Blockes Aussagen über die Resource machen. Bei der Benutzung von about wird eine URI-Referenz erwartet, so dass durch das Voranstellen des Lattenkreuzes die nachfolgende ID als relativ zum aktuell verarbeiteten Dokument interpretiert wird und damit zusätzliche Angaben über die im ersten Description-Block mit *mein_Dokument* benannte Resource möglich werden. In

c) ist einer Resource sogar zunächst lediglich ein Name zugewiesen worden. Würden anders als im Beispiel im folgenden keine Angaben über diese Resource gemacht, so hätte diese Zeile letztlich keine Wirkung.

Bäume / Verschachtelungen

Über die Formulierung einfacher Statements hinaus lassen sich Statements auch verknüpfen.



XML-Realisierung:

```
<rdf:Description ID="Document">
  <dc:publisher>Springer-Verlag</dc:publisher>
  <dc:creator>
    <rdf:Description>
      <vCard:FN>Hartmut Polzer</vCard:FN>
      <vCard:EMAIL>hpolzer@gmx.de</vCard:EMAIL>
      <rdf:type rdf:resource="http://purl.org/dc/terms/1.0/Person"/>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

Im Beispiel wird nach dem Verlag der Autor angegeben, der dann näher beschrieben wird. Hierzu wird mit Description eine neue (anonyme) Resource generiert, die die innerhalb dieses Description-Blockes aufgeführten Eigenschaften hat.

Nachdem bisher als Objekte nur Literals eingesetzt wurden, wird für die Aussage, dass es sich bei dem Autor um eine Person handelt, auf eine andere Resource verwiesen. Hierbei wird dem Prädikat stets das Attribut `rdf:resource` (`rdf` ist hierbei die Kennung für den im Kopf des RDF-Blockes vereinbarten RDF-Namespace, die auch im folgenden weiter verwendet wird) samt URI der entsprechenden Resource mitgegeben.

Variationen in der Schreibweise

Selbst wenn der RDF-Graph eindeutig ist, kann es verschiedene äquivalente XML-Formulierungen für ihn geben. Über die Möglichkeit des Vertauschens einzelner Description-Blöcke und das Vertauschen einzelner Aussagen innerhalb eines Description-Blockes wurde bereits gesprochen. Darüber hinaus lässt sich häufig das folgende Fragment

```
<rdf:Description ID="Document">
  <dc:publisher>Springer-Verlag</dc:publisher>
  <dc:creator>
    <rdf:Description>
      <vCard:FN>Hartmut Polzer</vCard:FN>
      <vCard:EMAIL>hpolzer@gmx.de</vCard:EMAIL>
      <rdf:type rdf:resource="http://purl.org/dc/terms/1.0/Person"/>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

ersetzen durch

```
<rdf:Description ID="Document">
  <dc:publisher>Springer-Verlag</dc:publisher>
  <dc:creator rdf:parseType="Resource">

    <vCard:FN>Hartmut Polzer</vCard:FN>
    <vCard:EMAIL>hpolzer@gmx.de</vCard:EMAIL>
    <rdf:type rdf:resource="http://purl.org/dc/terms/1.0/Person"/>

  </dc:creator>
</rdf:Description>
```

Durch die Benutzung von `rdf:parseType="Resource"` als Attribut eines Prädikates wird der Parser explizit angewiesen, für das Objekt des Statements eine anonyme Resource zu erzeugen, für die die innerhalb des zugehörigen Prädikat-Blocks gemachten Aussagen gelten. Damit ergibt sich eine Möglichkeit, einen Knoten zu generieren, ohne `rdf:Description` zu schreiben. Ob das eleganter erscheint, bleibt jedem selbst überlassen. Allerdings ist dieses Konstrukt im Hinblick auf die Existenz von `Description` im Prinzip überflüssig und kann andererseits `Description` nicht überall ersetzen, weil z.B. das Attribut `about` hier nicht zugelassen ist.

Content-Hiding

Mitunter wird es sinnvoll sein, RDF-Abschnitte innerhalb anderer Einheiten in ein XML-Dokument einzubetten. Insbesondere XHTML-Dokumente sind hierbei interessant.

Diese können dann von einer XML-Anwendung verarbeitet, also z.B. von einem Browser angezeigt werden, wobei der RDF-Abschnitt natürlich nicht dargestellt werden soll. Hierbei taucht allerdings das Problem auf, dass gemäß der XHTML-Specification eine XHTML-Anwendung den Inhalt von Tags, die sie nicht kennt, anzeigen MUSS. Als Resultat würden also im folgenden Beispiel die markierten Inhalte durch den Browser angezeigt.

```
<rdf:Description ID="Document">
  <dc:publisher>Springer-Verlag</dc:publisher>
  <dc:creator>
    <rdf:Description>
      <vCard:FN>Hartmut Polzer</vCard:FN>
      <vCard:EMAIL>hpolzer@gmx.de</vCard:EMAIL>
      <rdf:type rdf:resource="http://purl.org/dc/terms/1.0/Person"/>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

Um dieses Problem zu umgehen, müsste man sämtliche Inhalte innerhalb der Tags selber unterbringen.

Auch dieses lässt sich innerhalb der XML-Syntax realisieren.

Das folgende Beispiel ist in der sog. "Basic Abbreviated Syntax" (vgl. [MSS] 2.2.2.) formuliert und mit dem oben gezeigten semantisch äquivalent:

```
<rdf:Description ID="Document"
  dc:publisher="Springer-Verlag">
  <dc:creator>
    <rdf:Description
      vCard:FN="Hartmut Polzer"
      vCard:EMAIL="hpolzer@gmx.de">
      <rdf:type rdf:resource="http://purl.org/dc/terms/1.0/Person"/>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

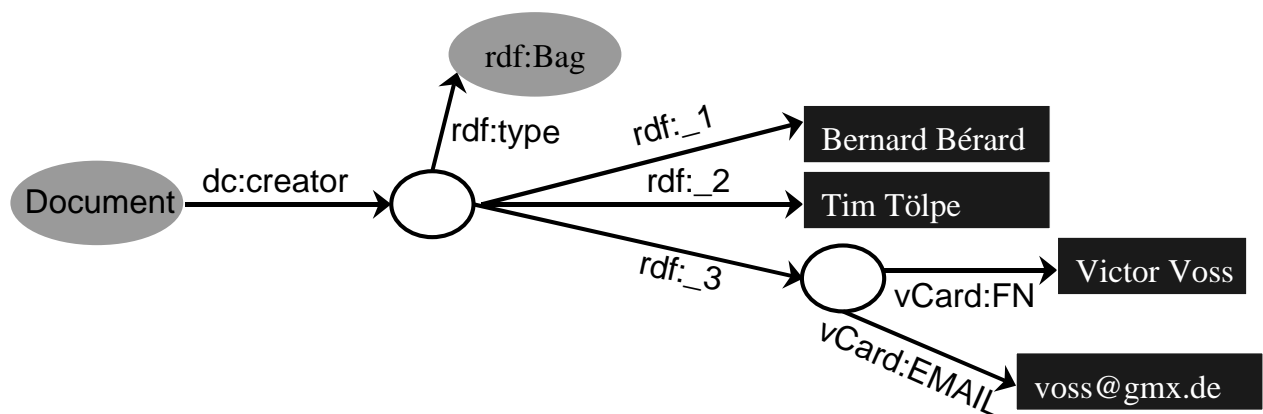
Bei dieser Syntax werden einzelne Prädikate als Attribute in die einzelnen Description-Tags mit hineingezogen. Dabei definiert der Name des Attributs das Prädikat, während das zugehörige Objekt, sofern es sich um ein Literal handelt, in Anführungszeichen angefügt wird. Hierbei ist zu beachten, dass alle Objekte, die Literals sind, bereits innerhalb des einleitenden Description-Tags behandelt werden müssen. Auf Objekte, die keine Literals sind, kann dagegen wie bisher innerhalb von Tags, die für einzelne Prädikate stehen, mit `rdf:resource` verwiesen werden. Andererseits muss man bedenken, dass sich nicht alle Konstruktionen in dieser kompakten "Basic Abbreviated Syntax" formulieren lassen. Da die wiederholte Auflistung eines Attributes innerhalb eines Tags nicht zulässig ist, versagt diese Darstellung bei Resources, bei deren Beschreibung einzelne Prädikate wiederholt werden.

Container

Nachdem die Formulierung einfacher und verschachtelter Statements sowie verschiedene mögliche Schreibweisen hierfür behandelt wurden, wird es im folgenden um Container gehen. Dabei werden die Beispiele aus Gründen der Übersichtlichkeit lediglich in der Basic Serialization Syntax (vgl. [MMS] 2.2.1.) formuliert. Außerdem wird bei Bedarf der umgebende RDF-Block weggelassen.

Container-Elemente (Bag, Seq, Alt) kommen z.B. zum Einsatz wenn die gemeinsame Autorenschaft zum Ausdruck gebracht bzw. auf verschiedene Zugangsmöglichkeiten hingewiesen werden soll.

Im folgenden Graph wird z.B. die gemeinsame Autorenschaft beschrieben, wobei die Reihenfolge der Autoren ausdrücklich keine Rolle spielen soll.



XML-Realisierung:

```
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:creator>
    <rdf:Bag>
      <rdf:li>Raoul B&#233;rard</rdf:li>
      <rdf:li>Tim T&#246;lpe</rdf:li>
      <rdf:li>
        <rdf:Description>
          <vCard:FN>Viktor Voss</vCard:FN>
          <vCard:EMAIL>voss@gmx.de</vCard:EMAIL>
        </rdf:Description>
      </rdf:li>
    </rdf:Bag>
  </dc:creator>
</rdf:Description>
```

Das zum Prädikat `dc:creator` gehörende Objekt ist vom Typ `Bag`. Dies wird in der XML-Syntax durch den Einschub `rdf:Bag` realisiert. Durch den von `<rdf:Bag>` und `</rdf:Bag>` umschlossenen Block wird eine neue Resource vom Typ `Bag` generiert.

Die Objekte dieses Bags werden dann nur noch aufgeführt. Die Aussage, die man über sie mittels `rdf:li` macht ist: Sie sind Objekte eines Containers (in diesem Fall eines Bags). Bei `rdf:li` handelt es sich wie bei `rdf:Description` um ein XML-Beschreibungselement für RDF und NICHT um ein Prädikat. Die "zugehörigen" Prädikate `rdf:_1`, `rdf:_2` usw. finden sich in diesem Fall erst nach dem Parsen in den Tripeln. Durch die Reihenfolge der Auflistung der einzelnen Objekte werden dabei die konkreten Prädikate vergeben.

Bei der Benutzung von Containern ist die Reihenfolge der einzelnen `<rdf:li>`-Abschnitte also für die Vergabe der Prädikate sehr wohl von Bedeutung, auch wenn diese Reihenfolge semantisch letztlich nicht bei allen Containern eine Rolle spielt.

Im Zusammenhang mit der z.B. für Content-Hiding benutzten kompakten Schreibweise, bei der die Prädikate als Attribute in einzelne Tags mit hinein gezogen werden, tritt hier eine Besonderheit auf: Da `rdf:li` kein Prädikat ist und daher nicht aufgelistet werden kann, müssen die Attribute `rdf:_1=...` `rdf:_2=...` usw. ggf. wie im folgenden Beispiel explizit aufgeführt werden.

```
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:creator>
    <rdf:Bag rdf:_1="Raoul B&#233;rard" rdf:_2="Tim T&#246;lpe"/>
  </dc:creator>
</rdf:Description>
```

Allerdings ist in diesen Fällen, ein zusätzliches Aufführen von weiteren Objekten mittels `rdf:li` nicht zulässig. Aus diesem Grund lässt sich der obige Graph auf diese Weise nicht durch die kompakte Schreibweise ausdrücken, weil das dritte Objekt dann nicht beschrieben werden kann.

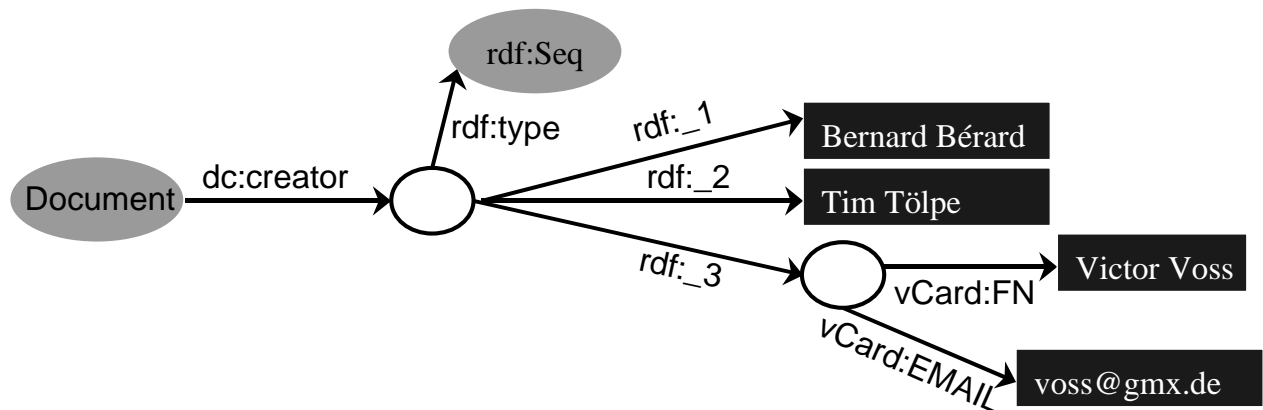
Die im Beispiel gezeigte Schreibweise `<rdf:Bag>` hat über die Container hinaus generelle Bedeutung. In RDF können in externen Schemata (vgl. [SS]) Klassen definiert werden, die gewissermassen bestimmte Objekttypen repräsentieren. Die einzelnen Container stellen z.B. Klassen dar. Durch `<ns:KLASSE>` wird stets ein neuer Knoten generiert, von dem mittels `rdf:type` auf eine durch den Namespace `ns` definierte Resource verwiesen wird, die die jeweilige Klasse beschreibt.

Genau genommen handelt es sich bei dem oben besprochenen XML-Ausschnitt also um eine Kurzschreibweise für folgenden Ausschnitt, der den RDF-Graphen unmittelbarer widerspiegelt, und bei dem die Prädikate `rdf:_1`, `rdf:_2` usw. explizit vergeben werden. Darüber hinaus lassen sich damit auch die oben beschriebenen Eigenheiten bei der Benutzung der kompakten Schreibweise umgehen.

```
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:creator>
    <rdf:Description>
      <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"
      <rdf:_1>Raoul B&#233;rard</rdf:_1>
      <rdf:_2>Tim T&#246;lpe</rdf:_2>
      <rdf:_3 rdf:resource="http://www.voss.de"/>
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

An diesem Beispiel ist außerdem zu sehen, wie mit Sonderzeichen verfahren wird, die wie auch in HTML möglich als Entity ausgedrückt werden.

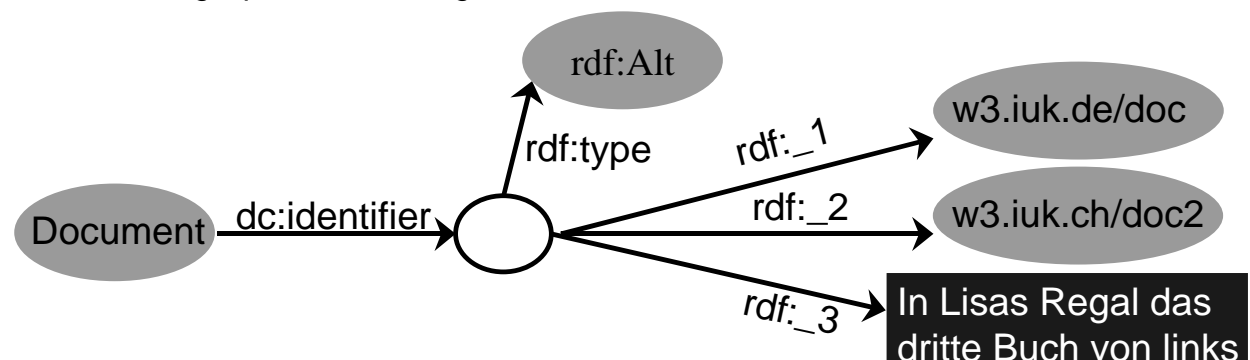
Die Beschreibung eines Containers vom Typ Sequence verläuft völlig analog. Falls die Reihenfolge der einzelnen Autoren ein Rolle spielt, ergibt sich z.B. folgender RDF-Graph:



Auch bei der XML-Syntax ändert sich nur die Container-Klasse:

```
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:creator>
    <rdf:Seq>
      <rdf:li>Raoul B&#233;rard</rdf:li>
      <rdf:li>Tim T&#246;lpe</rdf:li>
      <rdf:li>
        <rdf:Description>
          <vCard:FN>Viktor Voss</vCard:FN>
          <vCard:EMAIL>voss@gmx.de</vCard:EMAIL>
        </rdf:Description>
      </rdf:li>
    </rdf:Seq>
  </dc:creator>
</rdf:Description>
```

Für die dritte und letzte Container-Klasse Alternative ändert sich syntaktisch ebenfalls nur die bereits angesprochene Kleinigkeit:

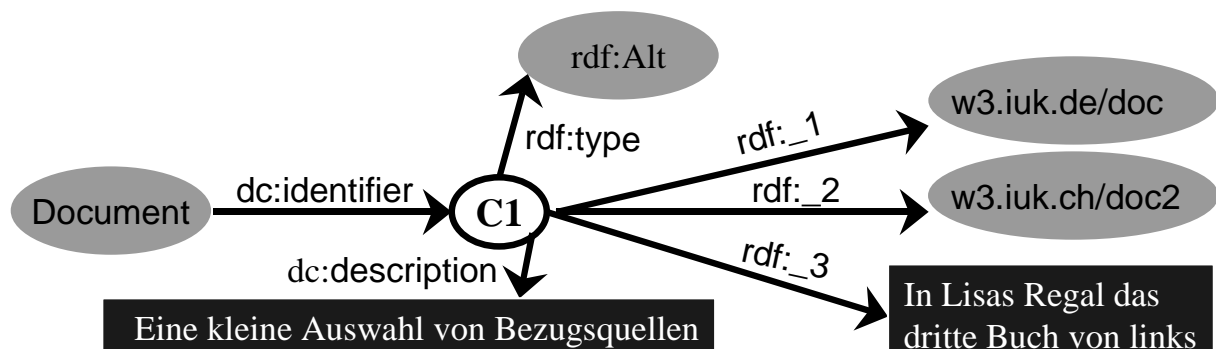


```

<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:identifizier>
    <rdf:Alt>
      <rdf:li rdf:resource="w3.iuk.de/doc"/>
      <rdf:li rdf:resource="w3.iuk.ch/doc2"/>
      <rdf:li>In Lisas Regal das dritte Buch von links</rdf:li>
    </rdf:Alt>
  </dc:identifizier>
</rdf:Description>

```

Über die bisher besprochenen Möglichkeiten hinaus, lassen sich für Container auch IDs vergeben, um auch außerhalb des jeweils aktuellen Description-Blockes Aussagen über ihn bzw. die in ihm enthaltenen Objekte machen zu können. Den folgenden RDF-Graph erhält man beispielsweise mit dem darunter stehenden XML-Abschnitt.



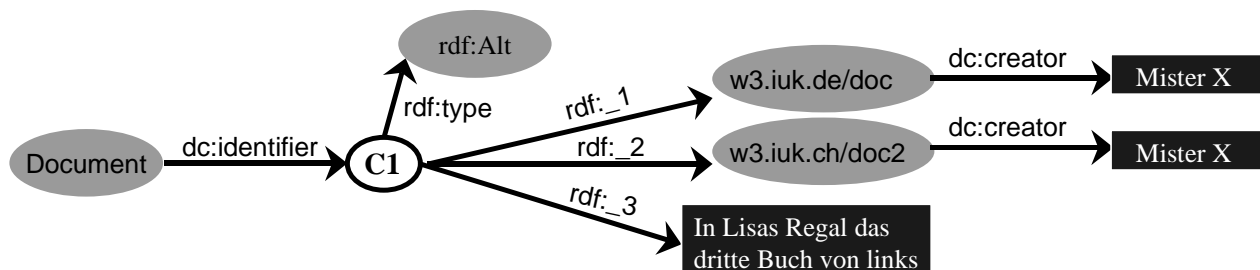
```

<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:identifizier>
    <rdf:Alt ID="C1">
      <rdf:li rdf:resource="w3.iuk.de/doc"/>
      <rdf:li rdf:resource="w3.iuk.ch/doc2"/>
      <rdf:li>In Lisas Regal das dritte Buch von links</rdf:li>
    </rdf:Alt>
  </dc:identifizier>
</rdf:Description>
<rdf:Description about="#C1">
  <dc:description>Eine kleine Auswahl von Bezugsquellen</dc:description>
</rdf:Description>

```

Durch die Benutzung von `about` Mit dieser Konstruktion werden keinerlei Aussagen über die Objekte in diesem Container gemacht; lediglich der Container bekommt eine weitere Eigenschaft.

Mit Hilfe des Schlüsselwortes `aboutEach` können dagegen auch Aussagen über Objekte innerhalb des Containers gemacht werden.



Dieser RDF-Graph lässt sich z.B. durch folgenden Abschnitt realisieren.

```

<rdf:Description about="http://www.iuk-initiative.de/iwi/Document">
  <dc:identifier>
    <rdf:Alt ID="C1">
      <rdf:li rdf:resource="w3.iuk.de/doc"/>
      <rdf:li rdf:resource="w3.iuk.ch/doc2"/>
      <rdf:li>In Lisas Regal das dritte Buch von links</rdf:li>
    </rdf:Alt>
  </dc:identifier>
</rdf:Description>
<rdf:Description aboutEach="#C1">
  <DC:creator>Mister X</DC:creator>
</rdf:Description>

```

In diesem Fall beziehen sich die Aussagen, im zweiten Description-Block auf jedes einzelne Objekt des Containers. Vorsicht ist jedoch geboten, falls sich im Container auch Literals befinden. Für diese sind natürlich keine weitergehenden Beschreibungen zugelassen.

Mit diesem Beispiel soll das Kapitel Container abgeschlossen werden.

Reifizierung

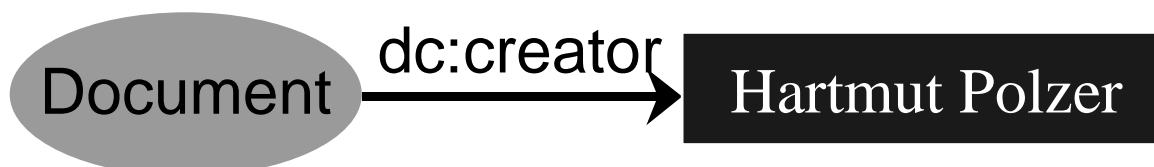
Eine Aussage lässt sich in RDF zum einen auf die herkömmliche Art mit einem Tripel beschreiben. In XML-Syntax könnte dies z.B. wie folgt aussehen.

```

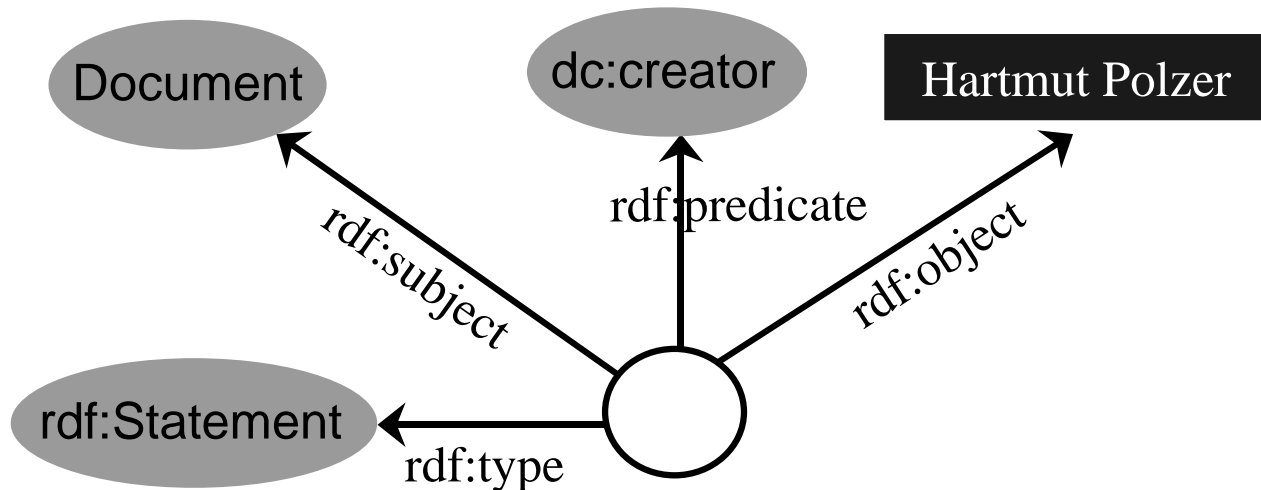
<rdf:Description about="http://www.iuk-initiative.de/iwi/Document_B">
  <dc:publisher>Springer-Verlag</dc:publisher>
</rdf:Description>

```

Diese Formulierung führt zu einem einzelnen Statement.



Unter Reifizierung eines Statements wird in RDF die Generierung einer neuen Resource vom Typ *Statement* und der die explizite Benennung von Subjekt, Prädikat und Objekt des Statements als solche verstanden. Dies würde zu folgendem RDF-Graph führen.



Außer den drei Statements, die die Bestandteile des ursprünglichen Statements bezeichnen, muss außerdem die neue Resource typisiert werden.

Hierzu wird eine Resource generiert die vom Typ *Statement* ist, und die außerdem noch folgende Eigenschaften hat: Subjekt ist Dokument, Prädikat ist `dc:creator` und Objekt ist das Literal "Hartmut Polzer".

Ein XML-Ausschnitt, der den obigen Graphen ausdrückt, könnte z.B. wie folgt aussehen, wobei für die mit Description neu generierte Resource mit ID= natürlich auch ein Name vergeben werden kann:

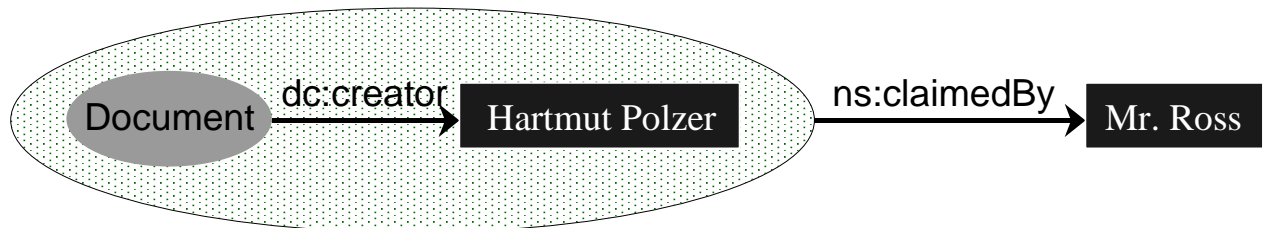
```
<rdf:Description>
  <rdf:type rdf:resource="http://www.w3.org/.../22-rdf-syntax-ns#Statement"/>
  <rdf:subject rdf:resource="http://www.iuk.org/Dokument"/>
  <rdf:predicate rdf:resource="http://purl.org/dc/elements/1.0/creator"/>
  <rdf:object>Hartmut Polzer</rdf:object>
</rdf:Description>
```

Da es sich bei `rdf:Statement` um eine Klasse handelt, kann man hier, wie bereits bei den Container-Klassen gesehen, auch eine kürzere Schreibweise benutzen:

```
<rdf:Statement>
  <rdf:subject rdf:resource="http://www.iuk.org/Dokument"/>
  <rdf:predicate rdf:resource="http://purl.org/dc/elements/1.0/creator"/>
  <rdf:object>Hartmut Polzer</rdf:object>
</rdf:Statement>
```

Allerdings sieht die RDF-Specification lediglich eine Kontrolle auf syntaktischer Ebene vor, so dass unvollständige Statements hier ebenso erlaubt sind wie Subjekt oder Prädikat vom Typ Literal oder sogar mehrere Prädikate, die von einer Resource ausgehen.

Aussagen höherer Ordnung: (Statements über Statements) lassen sich machen, indem man das ursprüngliche Statement als Subjekt oder Objekt weiterer Statements begreift.



In diesem Fall wird über die Resource vom Typ Statement also noch eine zusätzliche Aussage gemacht, die somit einfach zusätzlich in den Description-Block eingefügt werden kann.

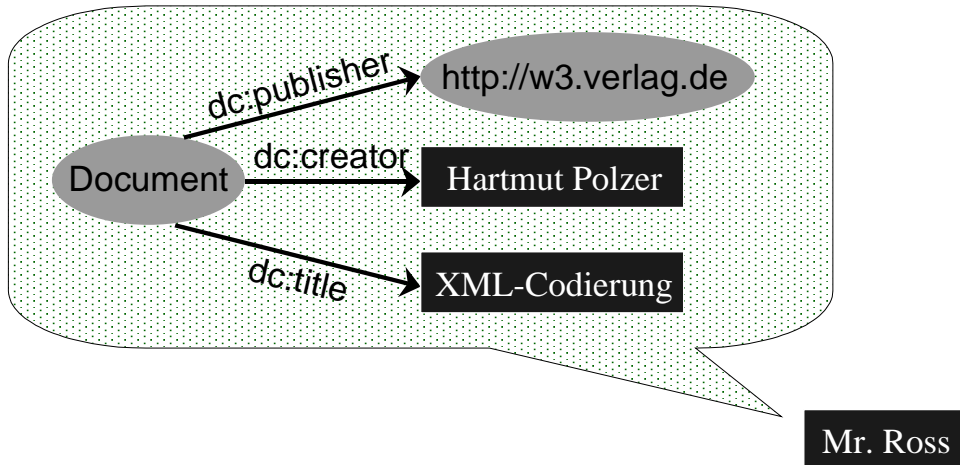
```
<rdf:Description about="http://www.iuk.org/Dokument" />
  <dc:creator>Hartmut Polzer</dc:creator>
</rdf:Description>
```

```
<rdf:Description ID="stat1">
  <rdf:type rdf:resource="http://www.w3.org/.../22-rdf-syntax-ns#Statement" />
  <rdf:subject rdf:resource="http://www.iuk.org/Dokument" />
  <rdf:predicate rdf:resource="http://purl.org/dc/elements/1.0/creator" />
  <rdf:object>Hartmut Polzer</rdf:object>
```

```
  <ns:claimedBy>Mr. Ross</ns:claimedBy>
</rdf:Description>
```

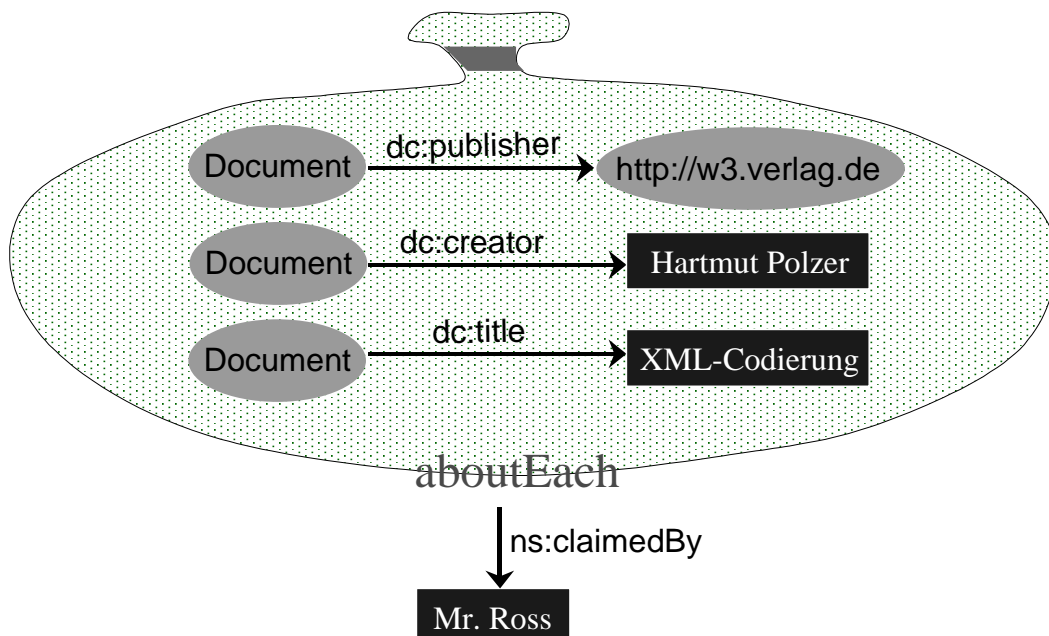
Will man in einem anderen Description-Block Aussagen hinsichtlich des reifizierten Statements machen, muss man den für das Statement neu generierten Knoten durch Vergabe einer ID benennen. Im obigen Beispiel wäre das aber nicht nötig gewesen. Wichtig ist es dabei, zu unterscheiden, ob das ursprüngliche Statement, bereits eine Tatsache ist. In diesem Fall muss das zugehörige Tripel wie im Beispiel separat generiert werden.

Diese explizite Art der Reifizierung wird allerdings recht aufwendig, sobald die Anzahl der Aussagen grösser wird:



In diesem Fall müssten drei reifizierte Statements wie oben gezeigt erzeugt werden, die außerdem noch die Eigenschaft haben, dass Mr. Ross ihre Richtigkeit behauptet. Allerdings lässt sich u.U. eine andere XML-Schreibweise angeben, hinter der die folgende Idee steckt:

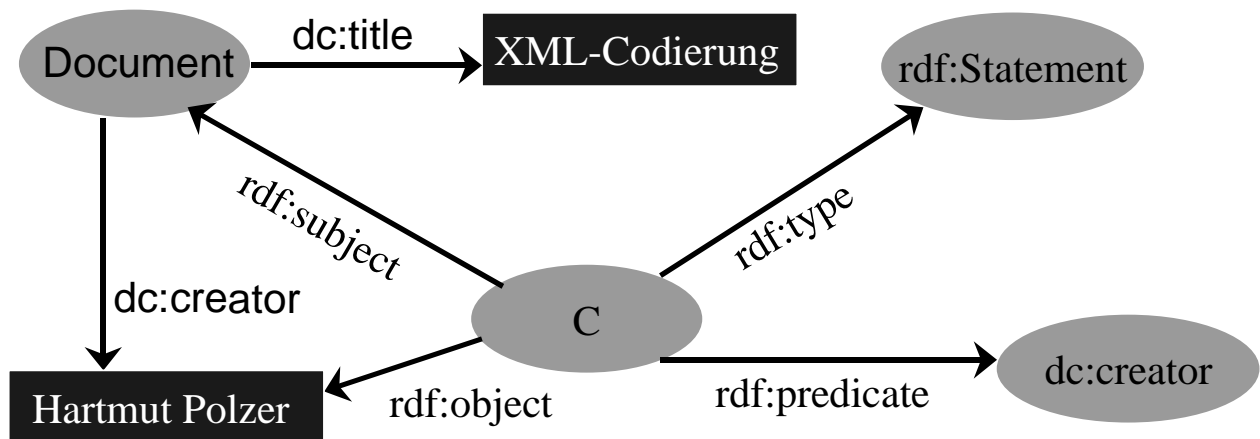
Man steckt sämtliche reifizierten Statements in einen Bag, gibt diesem eine ID und kann später mit Hilfe von *aboutEach* über jedes Objekt Aussagen machen. In diesem Fall kann man das Reifizieren der einzelnen Statements dem Parser überlassen:



Die zweite Möglichkeit ergibt sich durch die Vergabe von ID-Attributen an Properties, also die Benutzung von ID="id_name" innerhalb von Tags, die für Prädikate stehen.

```
<rdf:Description about="http://www.iuk.org/Document">
  <dc:creator ID="C">Hartmut Polzer</dc:creator>
  <dc:title>XML-Codierung</dc:title>
</rdf:Description>
```

In diesem Fall wird die Aussage "Hartmut Polzer ist der Autor von Document" reifiziert. Gleichzeitig bleibt das ursprüngliche Statement als Tatsache bestehen. Allerdings wird nicht wie beim Setzen einer bagID auch ein Bag generiert. Damit würde der Graph wie folgt aussehen:



Diese Art der Reifizierung bietet damit eine weitere Möglichkeit, einzelne Statements praktisch vom Parser reifizieren zu lassen. Allerdings muss dabei im Vergleich zur Benutzung von bagIDs für jedes so zu reifizierende Statement explizit eine eigene ID vergeben werden, während im anderen Fall nur eine einzige ID für den Container benötigt wird.

Auch hier kann der Fall eintreten, dass zwei Pfeile auf eine anonyme Resource zeigen, denn würde im obigen Graphen das Literal "Hartmut Polzer" wie im folgenden Abschnitt gezeigt durch eine anonyme Resource ersetzt, würden ebenfalls zwei Pfeile auf das Objekt des Statements zeigen.

```
<rdf:Description about="http://www.iuk.org/Document">
  <dc:creator ID="C">
    <rdf:Description>
      <vCard:FN>Hartmut Polzer</vCard:FN>
      <vCard:EMAIL>hartmut@gmx.de</vCard:EMAIL>
    </rdf:Description>
  </dc:creator>
  <dc:title>XML-Codierung</dc:title>
</rdf:Description>
```

ANHANG

Bekannte Fehler der gebräuchlichsten RDF-Parser

Die hier beschriebenen Probleme bzw. Fehler beziehen sich auf die aktuellen (Stand: 31. Mai. 2000) Online-Versionen der genannten Parser.

SiRPAC

Probleme mit <ns:Klasse>

Die Aussagen die mittels der kürzeren Schreibweise über ein Objekt einer Klasse gemacht werden werden nicht diesem zugeordnet, sondern einer weiteren neu generierten Ressource

Probleme mit aboutEach:

Die Aussagen, die über die im Container enthaltenen einzelnen Objekte gemacht werden, werden zwar mit diesen in Verbindung gebracht. Allerdings wird nicht sauber unterschieden zwischen Ressourcen und Literals.

Obwohl über Literals keine weiteren Aussagen gemacht werden dürfen, werden dafür anonyme Ressourcen generiert, die dann die beschriebenen Eigenschaften haben.

Probleme mit about:

Bezieht man sich mit einer Description mit dem Attribut about auf eine bereits vergebene ID (z.B. mit about="#Buch42"), werden die in diesem Description-Block aufgezählten Prädikate nicht der ursprünglichen Resource zugeordnet, die diese ID trägt, sondern einer neu generierten.

Pro-Solutions-Parser

Probleme bei der Reifizierung:

Werden in dem Teil, in dem Aussagen über Statements gemacht werden, komplexere Datenstrukturen (z.B. Bäume) benutzt, tauchen im Parser-Output CDATA-Teile auf .

Probleme mit about:

Bezieht man sich mit einer Description mit dem Attribut about auf eine bereits vergebene ID (z.B. mit about="#Buch42"), werden die in diesem Description-Block aufgezählten Prädikate nicht der ursprünglichen Resource zugeordnet, die diese ID trägt, sondern einer neu generierten.

Probleme mit aboutEach:

Die Aussagen, die über die im Container enthaltenen einzelnen Objekte gemacht werden, werden nicht mit diesen in Verbindung gebracht. Statt dessen werden diese

Aussagen jeweils zusätzlich generierten anonymen Ressourcen zugeordnet, deren Anzahl der der im Container enthaltenen Objekte entspricht.

Probleme mit unbekanntem Tags:

Lt. Spezifikation, muss ein RDF-Parser, wenn er auf ihm unbekannte Tags aus dem rdf-namespace ("<http://www.w3.org/TR/REC-rdf-syntax>") trifft, dieses Element, und dessen kompletten Inhalt unberücksichtigt lassen. Auf diese Weise kann man also innerhalb des RDF-Blocks in einem XML-Dokument auch andere Inhalte unterbringen.

Vom Pro-Solutions-Parser werden dagegen auch solche Tags vollständig geparkt.

Quellen

[MSS]	RDF Model and Syntax Specification http://www.w3.org/TR/REC-rdf-syntax
[SS]	RDF Schema Specification http://www.w3.org/TR/rdf-schema
SiRPAC	http://www.w3.org/RDF/Implementations/SiRPAC
Pro-Solutions-Parser	http://www.pro-solutions.com/rdfdemo