Kai Eckert

Die Provenienz von Linked Data

Linked Open Data öffnet die Datenmodelle der Bibliotheken für die Nutzung von außen und verheißt im Gegenzug einfachere Möglichkeiten, die eigenen Daten durch andere Daten anzureichern. Zu Recht besteht deshalb der Wunsch, die Herkunft aller Daten festzuhalten und darüber Auskunft geben zu können, welche Daten nun genau in den eigenen Datenbestand eingegangen sind und wer gegebenenfalls dafür verantwortlich zeichnet. Leider ist die Frage der Darstellung von Provenienzinformationen zu Linked Data bisher nur unbefriedigend und uneinheitlich gelöst. In diesem Kapitel stellen wir die grundsätzliche Problematik und mögliche Lösungsansätze vor und beleuchten aktuelle Entwicklungen in den Arbeitsgruppen des World Wide Web Consortium (W3C) und der Dublin Core Metadata Initiative (DCMI), die auf eine einheitliche Lösung in naher Zukunft hoffen lassen.

Einleitung

Laut Duden meint Provenienz den „Bereich, aus dem jmd., etwas stammt; Herkunft, Ursprung“ (Baer, 2001). Der Begriff Provenienz wird unter anderem im Kunsthandel verwendet, wenn es um die Herkunft eines Kunstwerks geht. Sowohl für die Authentizität des Werkes selbst, als auch für die Legitimation des Besitzes ist die vollständige Kenntnis der Provenienz wichtig.

Die Provenienz ist allerdings nicht nur für Kunstwerke von Bedeutung; es gibt viele Gründe, sich für die Herkunft beliebiger Ressourcen zu interessieren. In der Automobilindustrie liegen zum Beispiel Informationen darüber vor, wann und wo jedes einzelne Auto hergestellt wurde. Nur so kann im Fall eines Rückrufs klar ermittelt werden, welche Autos von einem Fehler im Herstellungsprozess betroffen sind, bzw. aus Kundensicht, welche Autos ganz sicher nicht betroffen sind. Im Linked Data Web ist der Ursprung von bestimmten Daten entscheidend für die Bewertung ihrer Korrektheit und ob sie gegebenenfalls in einer Anwendung wiederverwendet werden können. Informationen zur Provenienz sind eine Voraussetzung für Vertrauen. Provenienz ist nicht nur bei wenigen, wertvollen Objekten wichtig; gerade Daten können erst durch Provenienzinformationen wertvoll werden.

Eine allgemeine Definition für Provenienz, die für die Datenprovenienz anwendbar ist, stammt von der W3C Provenance Incubator Group (W3C Provenance Incubator Group, 2010):

> Die Provenienz einer Ressource ist ein Datensatz, der Entitäten und Prozesse beschreibt, die an der Produktion oder Auslieferung der Ressource beteiligt waren oder sie anderweitig beeinflusst haben. Die Provenienz liefert die entscheidende Grundlage, um die Authentizität zu bewerten und Vertrauen und Reproduzierbarkeit zu ermöglichen. Provenienzinformationen sind eine Form von Metadaten und können selbst zu wichtigen Datensätzen mit eigener Provenienz werden. (Übersetzung durch den Autor)

Bemerkenswert ist die Einbeziehung und Beschreibung der Prozesse, die zur Herstellung oder Veränderung einer Ressource geführt haben. Das unterscheidet Provenienzinformationen gemäß dieser Definition von einfachen Aussagen zum Ersteller oder Besitzer einer Ressource, wiewohl eben auch diese oft als Provenienzinformationen bezeichnet werden. Auf diese Unterscheidung werden wir im Verlauf dieses Kapitels noch weiter eingehen. Ebenso wichtig ist die Feststellung, dass Provenienzinformationen Metadaten sind und dass derartige Metadaten eigene Provenienzinformationen haben können. Denn genau darin liegt die Herausforderung im Linked Data Web: Wie erfasst man die Provenienz von Linked Data?

Hier müssen wir zunächst kurz klären, was eigentlich Metadaten sind. Oft werden Metadaten als Daten über Daten bezeichnet. Das ist jedoch irreführend, gerade im Bibliotheksumfeld, wo Katalogdaten über Bücher als Metadaten bezeichnet werden. Man könnte Bücher als eine Form von Daten sehen, aber spätestens bei der Beschreibung von anderen Ressourcen, wie dem Inventar eines Museums, passt die Definition nicht mehr. Die griechische Vorsilbe „Meta-“ bedeutet unter anderem „über, mit, nach“. Es ist zielführender, Metadaten als „Über-Daten“ zu sehen, also Daten über etwas. Meta impliziert auch, dass die Daten vom beschriebenen Etwas getrennt sind und sich auf einer anderen Ebene befinden, der Metaebene. Wir definieren daher wie folgt:

> Metadaten sind strukturierte Daten, die die Eigenschaften einer Ressource beschreiben.

Daten im Linked Data Web sind grundsätzlich Metadaten – beschreibende Daten über Ressourcen. Die Daten sind in RDF repräsentiert, dem Resource Description Framework. Eine Ressource kann dabei fast alles sein. Eine Ausnahme sind aber RDF-Daten selbst. Zumindest derzeit ist nicht klar, wie man innerhalb von RDF über RDF spricht. Das ist allerdings notwendig, um Metadaten zu RDF-Daten liefern zu können, also auch zur Repräsentation der Provenienz von RDF-Daten. Im Verlauf dieses Kapitels werden wir einige Entwicklungen auf diesem Gebiet näher beleuchten und auch auf verschiedene Praktiken eingehen, wie derzeit schon Provenienzinformationen im Linked Data Web dargestellt werden können.

Provenienz

Die Erfassung von Provenienzinformationen ist eine wichtige Anforderung in vielen Bereichen, die Anwendungen reichen von der Softwareentwicklung (Davies, German, Godfrey, & Hindle, 2011) über Datenbanken (Buneman, Khanna, & Wang-Chiew, 2001; Green, Karvounarakis, & Tannen, 2007) zu wissenschaftlichen Prozessen (Davidson & Freire, 2008) und vielen anderen.

Für die Darstellung von Provenienzinformationen wurden spezielle Vokabulare und Datenmodelle entwickelt, wie das Open Provenance Model (OPM)[1], Provenir[2] oder das Provenance Vocabulary.[3] Eine gute Übersicht – auch als Forschungsgebiet – findet sich im Abschlussbericht der W3C Provenance Incubator Group (W3C Provenance Incubator Group, 2010) und bei Moreau, 2010. Im Folgenden gehen wir auf zwei sehr unterschiedliche Vertreter ein: zum einen Dublin Core als sehr einfaches, allgemeines Metadatenvokabular, zum anderen PROV, das Provenienzmodell der W3C Provenance Working Group.

Dublin Core als einfaches Provenienzvokabular

Die Dublin Core Metadata Initiative (DCMI) stellt ein einfaches Metadatenvokabular zur Verfügung, das allgemein als Dublin Core bezeichnet wird. Ursprünglich bestand es aus 15 Elementen, die nach wie vor verfügbar sind und als Element Set bezeichnet werden (DCMI, Dublin Core Metadata Element Set, Version 1.1, 2010). Die Elemente sind sehr breit definiert, insbesondere haben sie keine Angabe zum Wertebereich, können also mit beliebigen Werten auf Objektseite arbeiten. Diese Elemente wurden spezifischer gefasst und typisiert. Dieses speziellere Vokabular wird als Dublin Core Terms bezeichnet, von denen es derzeit 55 gibt (DCMI, DCMI Metadata Terms, 2010).

Die Dublin Core Elemente gelten als veraltet und ihr Einsatz wird nicht mehr empfohlen. Die Elemente und die Terms haben verschiedene Namensräume, eine gängige Konvention ist es, das `dc`-Präfix für die Elemente zu verwenden und `dcterms` oder `dct` für die Terms.

1 http://openprovenance.org/
2 http://wiki.knoesis.org/index.php/Provenir_Ontology.
3 http://purl.org/net/provenance/

Ein typischer Datensatz, der Dublin Core nutzt, sieht so aus:[4]

```
ex:doc1     dct:title         "A mapping from DC..." ;
            dct:creator       ex:kai ;
            dct:created       "2012-02-28" ;
            dct:publisher     ex:w3c ;
            dct:issued        "2012-02-29" ;
            dct:subject       ex:dublincore ;
            dct:replaces      ex:doc2 ;
            dct:format        "HTML" .
```

Hier wird deutlich, dass sich nicht alle Aussagen auf die Provenienz der beschriebenen Ressource beziehen. So sind zum Beispiel `dct:title`, `dct:subject` und `dct:format` direkte Beschreibungen der Ressource. Sie liefern keinerlei Informationen darüber, wie die Ressource erstellt oder in der Vergangenheit verändert wurde.

Auf der anderen Seite lassen sich aus manchen Aussagen Informationen zur Provenienz der Ressource ableiten, zum Beispiel bezieht sich `dct:creator` auf den Autor und impliziert, dass die Ressource erstellt wurde. In ähnlicher Weise impliziert `dct:issued`, dass die Ressource veröffentlicht wurde. Das lässt sich auch aus der Aussage über den `dct:publisher` ableiten. Zuletzt setzt `dct:replaces` die Ressource in Beziehung zu einer weiteren Ressource und es kann davon ausgegangen werden, dass diese Ressource einen gewissen Einfluss auf unsere Ressource hatte, was uns weitere Informationen zur Provenienz liefert.

Aus diesen Überlegungen wird ein Muster ersichtlich, das allgemein auf Metadaten anwendbar ist: Man kann zwischen der Beschreibung einer Ressource und der Provenienz einer Ressource unterscheiden. Präziser definieren wir Provenienz-Metadaten als Metadaten, die Informationen zur Provenienz gemäß obiger Definition liefern, und Beschreibungsmetadaten als alle anderen Metadaten.

Nach dieser Definition können die DCMI Terms wie folgt klassifiziert werden:

Beschreibung: abstract, accessRights, accrualPeriodicity, accrualPolicy, alternative, audience, bibliographicCitation, conformsTo, coverage, description, educationLevel, extent, format, identifier, instructionalMethod, isRequiredBy,

4 Wir verwenden die gut lesbare Turtle-Syntax für RDF-Aussagen. Eine Aussage besteht aus Subjekt, Prädikat und Objekt, abgeschlossen durch einen Punkt. Durch ein Semikolon getrennt können mehrere Prädikate und Objekte aneinandergereiht werden, wodurch die ständige Wiederholung des Subjekts vermieden wird.

language, license, mediator, medium, relation, requires, rights, spatial, subject, tableOfContents, temporal, title, type.

Provenienz: accrualMethod, available, contributor, created, creator, date, dateAccepted, dateCopyrighted, dateSubmitted, hasFormat, hasPart, hasVersion, isFormatOf, isPartOf, isReferencedBy, isReplacedBy, issued, isVersionOf, modified, provenance, publisher, references, replaces, rightsHolder, source, valid.

Damit beziehen sich 26 der 55 Terme auf die Provenienz einer Ressource. Wir können dabei noch nach verschiedenen Aspekten der Provenienz unterscheiden:

Wer? *(contributor, creator, publisher, rightsHolder)* Alle Terme haben als Wertebereich `dct:Agent`, also eine Ressource, die handelt oder die Macht zum Handeln hat. Beitragende, Autoren und Verleger beeinflussen eindeutig eine Ressource und sind deshalb für ihren Ursprung wichtig. Für den Rechteinhaber mag das nicht unmittelbar einsichtig sein, allerdings verändert der Umstand, dass eine Ressource einen Besitzer hat, der Rechte an der Ressource ausübt, auch Eigenschaften der Ressource, beispielsweise ihre Verfügbarkeit.

Wann? *(available, created, date, dateAccepted, dateCopyrighted, dateSubmitted, issued, modified, valid)* Datumsangaben zählen typischerweise zu den Provenienzinformationen. Auch hier kann gefragt werden, inwiefern manche Daten die Ressource beeinflussen, zum Beispiel das Datum der Veröffentlichung. Auch hier gilt, dass der Einfluss nicht zwingend physischer Art sein muss, es kann auch die Änderung einer Eigenschaft sein, zum Beispiel der Eigenschaft, ob eine Ressource veröffentlicht ist. Die Unterscheidung, ob ein Datum relevant für die Provenienz ist, kann von Fall zu Fall unterschiedlich sein. Ein Beispiel dafür sind die Terme *available* und *valid*: Sie können Datumsbereiche zur Verfügbarkeit, bzw. zur Gültigkeit einer Ressource angeben. Oft ist zum Beispiel die Gültigkeit einer Ressource inhärent und von Anfang an bekannt, wie im Fall einer Kreditkarte oder eines Reisepasses. Dann würde man die Gültigkeit zur Beschreibung der Ressource zählen. Wurde eine Ressource allerdings bewusst als ungültig erklärt, zum Beispiel weil sich die Ressource als fehlerhaft herausgestellt hat, dann zählt diese Information zur Provenienz der Ressource.

Wie? *(isVersionOf, hasVersion, isFormatOf, hasFormat, references, isReferencedBy, replaces, isReplacedBy, source, hasPart, isPartOf, accrualMethod)* Ein wichtiger Teil der Provenienzinformationen bezieht sich darauf, wie die Ressource entstanden ist. Sie kann von anderen Ressourcen abgeleitet worden sein oder aus anderen Ressourcen bestehen. Streng genommen muss man auch hier unter-

scheiden, ob eine Veränderung der Ressource beschrieben wird. Für die inversen Terme ist das unter Umständen nicht der Fall, zum Beispiel bei *hasVersion*. Wir beziehen hier trotzdem alle Terme mit ein, da sie generell genutzt werden können, um die Beziehungen zwischen voneinander zumindest in einer Richtung abhängigen Ressourcen zu beschreiben.

Damit bleibt ein spezieller Term übrig: *provenance*. Er ist definiert als „Aussage über jedwede Veränderungen in Besitz und Obhut der Ressource seit ihrer Erstellung, die bedeutsam für ihre Authentizität, Integrität und Interpretation ist" (Übersetzung durch den Autor). Das passt ausgezeichnet zu unserer Definition von Provenienz, trotzdem dürften die Informationen aus knapp der Hälfte der verfügbaren Terme mehr über die Provenienz der Ressource aussagen, als dieser einzelne Term.

Zusammenfassend ist festzustellen, dass die DCMI-Terme, bzw. Metadaten, die diese nutzen, eine Menge an Informationen zur Provenienz einer Ressource beinhalten, insbesondere darüber, wann eine Ressource in der Vergangenheit beeinflusst wurde, von wem, und wie.

W3C Provenance Working Group

Die W3C Provenance Working Group entwickelt momentan Spezifikationen für den interoperablen Austausch von Provenienzinformationen in heterogenen Umgebungen wie dem Web (W3C Provenance Working Group, 2012). Die Veröffentlichung einer W3C Recommendation ist für Januar 2013 geplant. Insofern sind die Informationen in diesem Kapitel nur als vorläufig anzusehen. Änderungen sind allerdings nur noch im Detail zu erwarten, die Grundlagen, um die es in diesem Artikel geht, können als stabil angesehen werden.[5]

Die Spezifikationen, die von der Arbeitsgruppe entwickelt werden, werden unter dem Namen PROV zusammengefasst. PROV besteht aus dem Datenmodel (PROV-DM) und der Ontologie (PROV-O). Beide sind in eigenen Dokumenten beschrieben, die derzeit als Working Drafts vorliegen (W3C Provenance Working Group, 2012; W3C Provenance Working Group, 2012). PROV-DM ist in einer formalen Sprache formuliert, die in PROV-O nach RDF übertragen wurde, wobei die

5 Anmerkung der Herausgeber: PROV wurde am 30. April 2013 als Recommendation veröffentlicht. Der Text ist davor entstanden und bezieht sich auf die im Literaturverzeichnis angegebenen Working Drafts, die der Veröffentlichung als Recommendation vorausgingen.

Ontologiesprache OWL2 zum Einsatz kommt (W3C OWL Working Group, 2009). Im Folgenden beziehen wir uns auf PROV-O.

Der wesentliche Unterschied zwischen PROV und anderen Provenienzmodellen wie OPM sowie Dublin Core besteht darin, dass es Aktivitäten in den Mittelpunkt stellt, die die beschriebene Ressource beeinflussen. Dadurch wird eine handelnde Person (Agent) nicht direkt in Beziehung zur Ressource gesetzt, wie bei `dct:creator`, stattdessen ist sie mit einer Aktivität verknüpft, die zur Erstellung der Ressource geführt hat.

Das folgende Beispiel, dem PROV-O Working Draft entnommen, soll das verdeutlichen. Hier geht es um ein Balkendiagramm (`ex:bar_chart`), das von Derek (`ex:derek`) erstellt wurde. Die Erstellung war eine Aktivität (`ex:illustrationActivity`). Aus dieser Aktivität ist das Balkendiagramm hervorgegangen (`prov:WasGeneratedBy`), die Aktivität war verknüpft mit (`prov:wasAssociatedWith`) Derek. Für die Erstellung wurde ein Datensatz (`ex:aggregatedByRegions`) verwendet (`prov:used`), was letztlich bedeutet, dass das Balkendiagramm von diesem Datensatz abgeleitet wurde (`prov:wasDerivedFrom`).

```
ex:bar_chart7
    a6  prov:Entity;
    prov:wasGeneratedBy ex:illustrationActivity;
    prov:wasDerivedFrom ex:aggregatedByRegions.
ex:illustrationActivity
   a prov:Activity;
   prov:used ex:aggregatedByRegions;
   prov:wasAssociatedWith ex:derek.
```

Das ist nur der erste Teil des Beispiels, im Original wird eine vollständige Kette von Provenienzinformationen bis zu den ursprünglichen Daten dargestellt.[7] Hier soll es nur darum gehen, die grundsätzliche Idee zu vermitteln.

Im Beispiel ist zu sehen, dass es neben der indirekten Verbindung zwischen dem Datensatz und dem Balkendiagramm über die Aktivität auch die direkte Aussage gibt, dass das Diagramm vom Datensatz abgeleitet wurde. Es gibt weitere derartige „Abkürzungen“ in PROV-O, wie zum Beispiel die Eigenschaft `prov:wasAttributedTo`, die eine handelnde Person direkt mit einer Entität verbindet, ähnlich wie `dct:creator` in Dublin Core.

6 a ist eine Kurzform für rdf:type.

7 Hier ist eine interessante Verbindung zum Thema Forschungsdaten zu sehen, das in diesem Sammelband von Ritze et al. behandelt wird.. Durch PROV lassen sich Zusammenhänge zwischen Daten, Aufbereitungen und daraus resultierenden Publikationen detailliert darstellen.

Abbildung 1 verdeutlicht den Unterschied in der Informationsrepräsentation zwischen Dublin Core und PROV. Deutlich ist die höhere Komplexität zu erkennen, die durch die Aktivität entsteht. Die untere linke Kante ist dabei die direkte

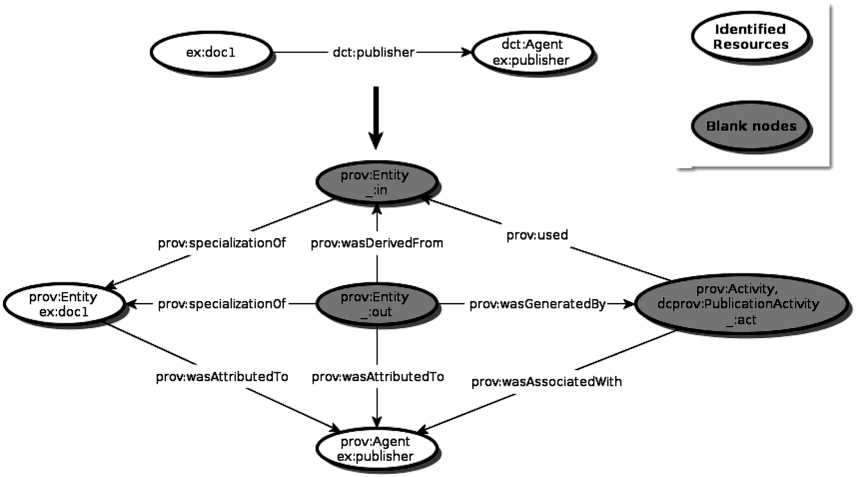

Abbildung 1: Dublin Core im Vergleich zu PROV.

Entsprechung zur dct:publisher Aussage, die in PROV redundant ist.

In Abbildung 1 sind zwei weitere Besonderheiten erkennbar, die für die Modellierung von Bedeutung sind.

Zum einen sieht man, dass die Ressource, um die es eigentlich geht – `ex:doc1` – gar nicht direkt mit der Aktivität verbunden ist. Stattdessen gibt es zwei weitere Ressourcen, die hier als Blank Nodes realisiert sind und die die Grundlage, bzw. das Ergebnis der Aktivität repräsentieren. Diese Unterscheidung ist wichtig, wenn man die vollständige Kette der Entstehung und Veränderung einer Ressource beschreiben will. In PROV werden alle Ressourcen, insbesondere auch diese „Zwischenergebnisse", als Entitäten (`prov:Entity`) bezeichnet. Entitäten sind eigenständige Ressourcen, können aber je nach Anwendung auch als Zustände einer übergeordneten Entität verstanden werden. Diese Beziehungen zwischen Entitäten werden in PROV durch die Prädikate `prov:alternateOf` und `prov:specializationOf` beschrieben. Im Allgemeinen sollten die einzelnen Entitäten natürlich auch identifiziert und nicht nur als Blank Nodes repräsentiert werden. Ein einfaches Beispiel soll das verdeutlichen, die Entstehung des Artikels über den Eiffelturm in Wikipedia. Dieser Artikel ist die Ressource, die beschrieben wird, identifiziert durch `http://de.wikipedia.org/wiki/Eiffelturm`. Betrachtet man die Versionsgeschichte dieses Artikels, so sieht

man, dass er vor rund 10 Jahren angelegt wurde und es seitdem über 1.600 verschiedene Versionen des Artikels gab, er also jedes Mal durch eine Aktivität verändert wurde. Alle diese Versionen mit `http://de.wikipedia.org/wiki/Eiffelturm` zu bezeichnen, wäre sicher nicht sehr hilfreich. Stattdessen gibt es Identifier für jede einzelne Version, die erste Version wird zum Beispiel identifiziert durch `http://de.wikipedia.org/w/index.php?title=Eiffelturm&oldid=3583`.

Die zweite Besonderheit liegt in der fehlenden Spezifikation der Aktivität, durch die der Status einer Ressource geändert wurde. In der Abbildung soll es um eine Veröffentlichungsaktivität gehen, aber die Umsetzung in PROV sähe für die Erstellung einer neuen Version genauso aus. Es gibt zu dieser Unterscheidung in PROV das Konzept von Rollen, außerdem können die Aktivitäten natürlich innerhalb einer konkreten Anwendung weiter spezialisiert werden. Die Ausgestaltung liegt aber beim Anwender, es werden keine konkreten Rollen und Spezialisierungen vorgegeben. Hier ist PROV also allgemeiner gehalten als Dublin Core.

Mit Dublin Core und PROV haben wir zwei grundlegend verschiedene Vertreter von Ontologien vorgestellt, mittels derer die Provenienz von Ressourcen repräsentiert werden kann. Fast die Hälfte der Terme in Dublin Core bezieht sich mehr oder weniger auf die Provenienz der Ressource. Trotz der allgemeinen Ausrichtung von Dublin Core werden dabei schon sehr spezielle Formen der Beeinflussung einer Ressource vorgegeben, die sich weitgehend am Entstehungsprozess schriftlicher Veröffentlichungen orientieren. Dem gegenüber steht mit PROV eine Ontologie zur Verfügung, um feingranular beliebige Provenienzketten zu repräsentieren, die Ausgestaltung für einen konkreten Einsatzbereich liegt dabei allerdings beim Anwender. Je nach Anwendungsfall ist das eine oder das andere zu bevorzugen. Wenn es darum geht, die wichtigsten Aussagen zur Entstehung einer Ressource zusammenzufassen, dann ist Dublin Core klar zu empfehlen. Ist die gesamte Provenienzkette abzubilden, mit der Möglichkeit, weitere Informationen zum darunterliegenden Entstehungsprozess und dem Lebenszyklus der Ressource hinzuzufügen, dann liefert PROV dafür eine solide Basis.

Die Provenienz von Linked Data

Bis hierhin haben wir festgestellt, dass Provenienzinformationen Metadaten sind und wir wissen nun auch, wie Provenienzinformationen *als* Linked Data repräsentiert werden können. Im Folgenden geht es nun darum, Provenienzinformationen *für* Linked Data bereitzustellen.

Auf den ersten Blick ist nicht einsichtig, warum die Darstellung von Provenienzinformationen für Linked Data speziell oder anders sein sollte, als die Darstellung von Provenienzinformationen für andere, beliebige Ressourcen. Das Problem ist, dass Metadaten im Allgemeinen, aber eben auch Linked Data, oft nicht als eine eigenständige Ressource gesehen werden. Metadaten sind einfach „da“, sie beschreiben andere Ressourcen. Die Gründe sind vielfältig, die folgenden Faktoren können aber Hinweise darauf liefern, wieso das so ist.

Metadaten entstehen in Anwendungen: In einer typischen Anwendung werden Daten über Ressourcen gespeichert. Weitere Informationen über die Daten werden oft nicht gebraucht. Man denke zum Beispiel an ein Dateisystem, das Metadaten zu Dateien speichert, wie Zeitpunkt der Erstellung, letzten Änderung, und den Besitzer. Warum sollten diese Metadaten weiter beschreibbar sein? Eine typische Datenbankanwendung enthält Daten, zum Beispiel über Kunden oder im Fall eines Bibliothekskatalogs über Bücher. Es ist ein klarer Schnitt erkennbar zwischen dem was beschrieben wird und der Beschreibung. Die Möglichkeit, die Beschreibung selbst zu beschreiben, würde weitreichende Änderungen im Datenbankentwurf und der Softwarearchitektur erzwingen.

Metadaten gehören zu Ressourcen: Oft sind Metadaten mehr oder weniger feste Bestandteile der Ressourcen. Zum Beispiel können PDF-, JPEG- oder MP3-Dateien Metadaten über den Ersteller des Dokuments, Bildes oder Liedes enthalten. Auch hier ist unmittelbar klar, was die Ressource ist und was die Beschreibung.

„Schlampiger“ Einsatz von Provenienzinformationen: Trotz dieser fundamentalen Trennung wird manchmal eben doch die Provenienz von Metadaten benötigt. Leider wird das dann oft unsauber umgesetzt und die Provenienzinformationen werden mit den Metadaten vermischt. Ein typisches Beispiel ist eine Spalte „Letzte Änderung“ in einer Datenbank. Alle Spalten in einer Kundentabelle beschreiben den Kunden, diese Spalte tut es offensichtlich nicht. Stattdessen beschreibt es die Daten über den Kunden, die in der Datenbank gespeichert sind und gibt an, wann diese Daten (und nicht der Kunde) zuletzt geändert wurden. Diese Unsauberkeit kommt oft vor und lässt sich ebenfalls durch die tief verwurzelte Wahrnehmung der unterschiedlichen Ebenen der Beschreibung erklären: Warum solle man präziser bei der Modellierung sein, es ist doch klar, was zur Beschreibung und was zur beschriebenen Ressource zählt.

Synonymer Einsatz von Beschreibung und beschriebener Ressource: Innerhalb eines Systems entspricht die Beschreibung einer Ressource oft der Ressource. In Bibliothekskatalogen existieren Identifikatoren für die einzelnen Kata-

logisate und es ist verführerisch, sie als Identifier für die Bücher selbst zu sehen. Dagegen ist auch nichts zu sagen, wenn die Bedeutung der Identifier klar ist und sie konsistent genutzt werden. Aber auf keinen Fall können sie sowohl als Identifier für die Bücher, als auch als Identifier für das Katalogisat genutzt werden. Leider wird genau das oft vernachlässigt, was letztlich zum schlampigen Einsatz von Provenienzinformationen führt, wie oben beschrieben. Die einzige Lösung besteht darin, konsequent zwei Identifier einzuführen, einen für die beschriebene Ressource und einen für die Beschreibung. Nur so können sie konsistent unterschieden werden.

Zusammengefasst: Um für Metadaten Provenienzinformationen bereitstellen zu können, müssen diese losgelöst von der beschriebenen Ressource sein und eigenständig identifizierbar. So werden Metadaten zu eigenständigen Ressourcen. Abbildung 2 verdeutlicht den Prozess. Zunächst sind Ressourcen und Beschreibungen gemischt. Sie sind unterscheidbar, die Ressourcen sind durch Bilder dargestellt, die Beschreibungen durch Text. Das entspricht der intuitiven Unterscheidung von Ressourcen und Metadaten, wie wir sie die ganze Zeit vornehmen, in unserer Wahrnehmung und in unseren Anwendungen. Indem wir die Metadaten identifizierbar machen, wechseln wir von einer intuitiven zu einer expliziten Unterscheidung.

Die Trennung von Beschreibung und Ressource ist in RDF gelöst. Selbst in eingebetteten Versionen von RDF wie RDFa ist die Beschreibung klar von der Ressource unterscheidbar. Leider gibt es (noch) keine allgemein akzeptierte Form in RDF, die Metadaten selbst identifizierbar zu machen. Gleichwohl gibt es verschiedene Ansätze und Best-Practice-Empfehlungen, die wir im Folgenden kurz vorstellen wollen. Derzeit arbeitet die W3C RDF Working Group an der nächsten Version von RDF und es ist davon auszugehen, dass dort der Identifizierbarkeit von RDF-Datensätzen Rechnung getragen wird. Unabhängig vom gewählten Ansatz geht es immer darum, RDF-Aussagen einzeln oder als Gruppe identifizierbar zu machen. In Eckert, Pfeffer, & Stuckenschmidt, 2009 haben wir den Begriff „Metametadaten“ geprägt, um die Besonderheit bei der Beschreibung von Metadaten gegenüber beliebigen anderen Ressourcen zu verdeutlichen.

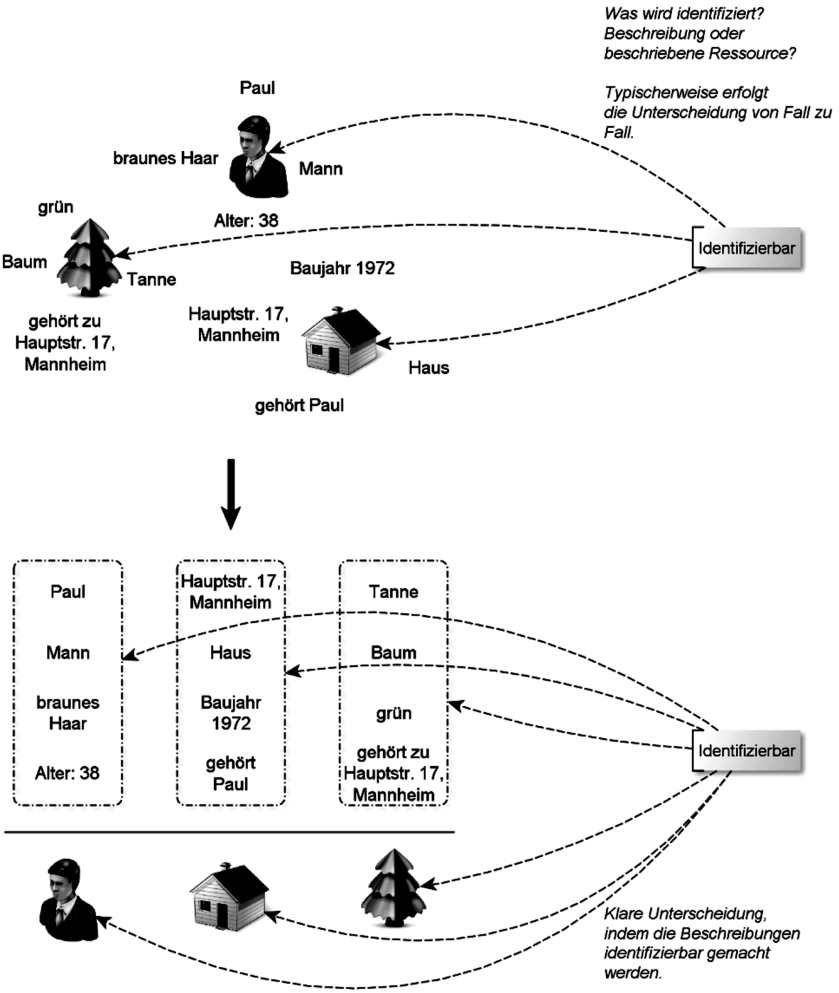

Abbildung 2: Von Beschreibungen zu Ressourcen.

Linked Metadata

Der erste Ansatz nutzt ganz einfach die Linked Data Prinzipien (W3C SWEO Interest Group, 2008; Heath & Bizer, 2011), um Metadaten zu Linked Data bereitzustellen. Immer, wenn Linked Data im Web veröffentlicht wird, muss ein URI geprägt werden, der die Daten zugänglich und damit auch identifizierbar macht, genau wie für jede andere Ressource. Als Beispiel dient wieder der Eiffelturm, über den

Daten als Linked Data veröffentlicht werden (Abbildung 3). Der Eiffelturm selbst wird dabei durch `ex:eiffeltower` identifiziert. Wird `ex:eiffeltower` dereferenziert, d. h. in einen Browser eingegeben oder auf andere Weise angefordert, leitet der Server per 303 redirect auf eine URL weiter, unter der Daten über diese Ressource gefunden werden können, in diesem Fall `ex:eiffeltower-meta`. Damit ist `ex:eiffeltower-meta` eine eigene Ressource, die ebenfalls beschrieben werden kann, entweder mit RDF-Aussagen, die direkt mitgeliefert werden oder unter einer eigenen URI, zum Beispiel `ex:eiffeltower-meta-meta`.

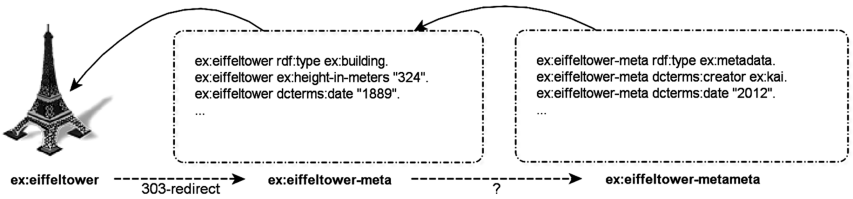

Abbildung 3: Linked Metadata

Im ersten Fall wäre es nicht möglich, weitere Aussagen über die Metametadaten zu treffen, da sie nicht durch einen URI identifiziert werden. In letzterem Fall bleibt die Frage, wie die Metametadaten abgerufen werden können. Ein 303-Redirect kommt hier nicht in Frage, da ja Daten unter `ex:eiffeltower-meta` vorliegen. Normalerweise teilt die Anwendung mit, in welchem Format sie eine Antwort erwartet, so dass man zum Beispiel bei einem Bild entscheiden kann, ob das Bild selbst oder eine Beschreibung des Bildes gewünscht ist. Das kann hier nicht funktionieren, da sowohl Ressource als auch Beschreibung als RDF vorliegen.

Eine mögliche Lösung wäre die Einführung eines neuen HTTP-Request-Headers, der angibt, dass Metadaten zur Ressource gewünscht sind. Immerhin existiert bereits das Gegenstück für die HTTP-Response: der Server kann einen Link-Header schicken, der angibt, dass Metadaten für die angeforderte Ressource unter einer anderen URL vorliegen, z. B.:

```
Link: <http://example.org/eiffeltower-metameta>; rel=meta
```

Die empfohlene Vorgehensweise ist deshalb, die Header für eine Ressource gezielt abzufragen (Head-Request) und zu prüfen, ob ein Link-Header auf zusätzliche Metadaten verweist.

Ein anderer Ansatz, der für RDF funktioniert, wäre, auf die Beschreibung der Daten innerhalb der ausgelieferten RDF-Aussagen zu verweisen. Zum Beispiel könnte man die folgende Aussage hinzufügen:

```
ex:eiffeltower-meta rdfs:seeAlso ex:eiffeltower-metameta.
```

Leider gibt es kein allgemein anerkanntes Prädikat für diesen Zweck – eine andere Variante wäre die Nutzung von `foaf:page`. Beide sind sehr allgemein und insofern unbefriedigend für diesen grundlegenden und speziellen Zweck.

Diese ersten Überlegungen zeigen schon, dass die Darstellung der Provenienz von Linked Data nicht ganz einfach ist. Es ist klar, dass die Provenienzinformationen nahtlos mit den eigentlichen Daten integriert werden müssen. Der Vorteil bei dem hier vorgestellten Ansatz ist, dass er lediglich die Linked-Data-Prinzipien unabhängig vom Typ der Ressource umsetzt und daher auf jeden Fall gültig ist. Es gibt aber eben Nachteile, die den Ansatz nicht allgemein anwendbar machen:

1. Er ist nur geeignet für die Erfassung von Provenienzinformationen auf Datensatzebene, da pro identifizierbare Teilmenge von RDF-Aussagen eine Anfrage an den Server gestellt werden muss. Es gibt aber Forderungen, zum Beispiel durch Hillmann, Dushay, & Phipps, 2004, nach Provenienzinformationen für einzelne Aussagen.
2. Es ist notwendig, die Response-Header und das Verarbeiten von Request-Headern zu beeinflussen, das bedeutet, der volle Zugriff auf den Web-Server wird benötigt.
3. Es gibt immer noch mehrere Freiheitsgrade bei der Implementierung, die die Interoperabilität beeinträchtigen, zum Beispiel, ob Header und/oder Prädikate genutzt werden und ggf. welche, bzw., ob die Provenienzinformationen zusammen mit den Daten ausgeliefert werden oder als eigenständige Ressource.
4. Es ist nicht klar, wie die Provenienzinformationen innerhalb von RDF-Anwendungen repräsentiert und organisiert werden. Zumindest muss erfasst bleiben, über welche URL welche Aussagen abgerufen wurden. Das wird bereits in vielen Triple-Stores umgesetzt, doch selbst dann geht die Information aus den Link-Header verloren und die Verbindung zwischen den Provenienzinformationen und den beschriebenen Ressourcen kann nur noch aus dem Inhalt der Aussagen abgeleitet werden.

In diesem Sammelband beschreibt Pascal Christoph in seinem Beitrag über Datenanreicherung auf LOD-Basis erste Ansätze, Provenienzinformationen in lobid.org als Linked Metadata umzusetzen.

Reification

Schon seit der ersten Version von RDF existiert ein Mechanismus, um auf Aussagenebene Metainformationen auszudrücken: Reification. In Eckert, Pfeffer, & Stuckenschmidt, 2009 haben wir gezeigt, wie mit Reification grundsätzlich auch Provenienzinformationen dargestellt werden können.

Reification erlaubt es, einzelne RDF-Aussagen zu beschreiben. Dazu muss eine Aussage zunächst zu einer Ressource werden, genauer gesagt zu einer Instanz der Klasse `rdf:Statement`. Zur Beschreibung des Statements gibt es drei Prädikate: `rdf_subject`, `rdf:predicate` und `rdf:object`. Die Reification der Aussage „`ex:eiffeltower ex:height-in-meters „324"^^xsd:integer.`" sieht dabei wie folgt aus:

```
ex:stmt1 rdf:type rdf:Statement;
         rdf:subject ex:eiffeltower;
         rdf:predicate ex:height-in-meters;
         rdf:object "324"^^xsd:integer.
```

Nun lassen sich weitere Aussagen über diese Aussage treffen, zum Beispiel können wir angeben, wer die Aussage getroffen hat: „`ex:stmt1 dct:creator ex:kai.`"

Trotz der allgemeinen Verwendbarkeit von Reification für die Darstellung von Provenienzinformationen auf Aussagenebene ist die Reification kaum verbreitet und es wird sogar diskutiert, sie in der nächsten Version von RDF nicht mehr offiziell zu unterstützen und von ihrem Einsatz abzuraten (Hawke, 2011).

Grund dafür ist unter anderem die umständliche Darstellung einer Aussage, die zu einer Tripel-Explosion führt: Vier Aussagen sind notwendig, um eine Aussage zu reifizieren – vier zusätzliche Aussagen wohlgemerkt, da die Reification die Aussage selbst nicht beinhaltet. Darüber hinaus muss jede einzelne Aussage reifiziert werden und zu anderen Aussagen in Bezug gesetzt werden, falls die Aussagen gemeinsam beschrieben werden sollen.

Trotzdem führen wir die Reification hier noch der Vollständigkeit wegen auf, zumal es derzeit der einzige standardisierte Weg ist, über einzelne RDF-Aussagen innerhalb von RDF zu sprechen. Es gibt aber bereits andere Ansätze, wie die im Folgenden beschriebenen Named Graphs, die bereits eine viel höhere Akzeptanz in der Community haben und als Grundlage für die nächste RDF-Version gesehen werden können.

Named Graphs

Named Graphs wurden von Carroll, Bizer, Hayes, & Stickler, 2005, als eine kleine Erweiterung von RDF eingeführt. Sie entwickelten Named Graphs ausgehend von URI-Referenzen für RDF-Dateien, wie oben beschrieben. Named Graphs sind nicht Teil von RDF, wurden aber in SPARQL aufgenommen, der RDF-Anfragesprache (RDF Data Access Working Group, 2008). Ein Named Graph ist ein normaler RDF-Graph, der mit einem URI versehen wurde, der als Name fungiert. Der Name kann zum Beispiel die URI-Referenz der RDF-Datei sein, in der der Graph gespeichert ist, aber auch jeder beliebige andere URI. TriG (Bizer & Cyganiak, 2007) ist eine mögliche Serialisierung von Named Graphs. In TriG sieht der obige Beispielgraph wie folgt aus:

```
ex:eiffeltower-meta {
   ex:eiffeltower rdf:type ex:building.
   ex:eiffeltower ex:height-in-meters "324".
   ex:eiffeltower dcterms:date "1889".
   ...
}
```

Durch die Aufnahme von Named Graphs in SPARQL unterstützen die meisten Implementierungen heute Named Graphs.

In Eckert, Pfeffer, & Völker, 2010, haben wir gezeigt, dass Named Graphs für die Repräsentation von Provenienzinformationen gemäß den in Eckert, Pfeffer, & Stuckenschmidt, 2009, eingeführten Anwendungsfälle einsetzbar sind.

Sowohl Reification, als auch Named Graphs sind grundsätzlich geeignet, RDF-Aussagen oder Mengen von RDF-Aussagen zu identifizieren. Named Graphs sind dabei intuitiver, da sie sich an Datensätzen orientieren. Allerdings sind sie nicht Teil des RDF-Standards. Reification ist (noch) Teil des Standards, aber der Einsatz ist umständlich und kann nicht mehr empfohlen werden.

Aktuelle Entwicklungen in RDF

Zurzeit arbeitet die W3C RDF Working Group an der nächsten RDF-Version und die Entwicklung eines standardisierten Mechanismus zur Identifizierung von Graphen ist Teil ihrer Aufgabe.[8] Zunächst führte die Arbeitsgruppe dazu vorläufige Bezeichnungen für verschiedene Graph-Konzepte ein, die die Diskussion erleichtern:

8 http://www.w3.org/2011/01/rdf-wg-charter

G-Box: Eine G-Box ist ein Graph-Container, der RDF-Tripel enthält. Verschiedene G-Boxes können identische Tripel enthalten. Der Inhalt einer G-Box kann sich verändern.

G-Snap: Ein G-Snap ist eine idealisierte Momentaufnahme einer G-Box. Ein G-Snap enthält demnach ebenfalls RDF-Tripel, aber der Inhalt eines G-Snap kann sich nicht ändern. Ein G-Snap wird durch seinen Inhalt definiert, ein geänderter Inhalt führt zu einem anderen G-Snap. Entsprechend sind zwei G-Snaps, die den gleichen Inhalt haben, in Wirklichkeit derselbe G-Snap.

G-Text: Ein G-Text ist eine bestimmte Folge von Zeichen oder Bytes, die einen G-Snap in einer bestimmten Sprache repräsentieren, zum Beispiel Turtle oder RDF/XML. Der exakte Inhalt einer bestimmten G-Box lässt sich durch einen G-Text darstellen. Der G-Text beinhaltet den G-Snap, der dem gegenwärtigen Status, bzw. Inhalt der G-Box entspricht.

Diese Begriffe verdeutlichen, dass man Linked Data auf verschiedenen Ebenen identifizieren kann. Eine identifizierte G-Box entspricht dabei am ehesten einem im Web publizierten RDF-Datensatz oder einem Named Graph. Die Möglichkeit, dass der Inhalt sich ändert, ist normal im Web und etwas, mit dem man zusätzlich umgehen muss. Auch dazu gibt es viele Überlegungen und Entwicklungen, auf die wir hier aber nicht weiter eingehen wollen. Für die Darstellung von Provenienzinformationen dürfte vor allem der Begriff der G-Box von Bedeutung sein und die Frage, wie nahtlos er sich in die Web-Architektur und die RDF-Implementierungen integrieren wird.

Zur endgültigen Benennung einer G-Box und zur genauen Spezifikation ihrer Bedeutung gibt es zahlreiche Diskussionen innerhalb und außerhalb der Arbeitsgruppe. Man darf gespannt sein, wie die nächste RDF-Version aussehen wird, die Ende 2013 fertig sein soll.

In diesem Zusammenhang ist interessant, dass es auch in PROV das Konzept eines Provenienz-Datensatzes gibt, der wahrscheinlich *Bundle* heißen wird. Beide Arbeitsgruppen arbeiten zusammen, um sicherzustellen, dass das Konzept eines Bundles eine passende Entsprechung in RDF hat.

OAI-ORE

Das Fehlen eines standardisierten Mechanismus, um Informationen über RDF-Daten auszudrücken und die Nachteile der Reification haben dazu geführt, dass Entwickler nach anderen Lösungen suchen mussten, um unter anderem Pro-

venienzinformationen darzustellen. Ein Ansatz, der ohne direkte Metaebene auskommt, ist das *Object Reuse and Exchange* (ORE) Framework, das von der Open Archives Initiative (OAI) angeboten wird (Open Archives Initiative, 2008). OAI-ORE wurde ursprünglich entwickelt um ein etwas anderes Problem zu lösen, für das es ebenfalls keine standardisierte Lösung in RDF gibt: Wie kann man über Ressourcen Aussagen tätigen, die nur in einem bestimmten Kontext gültig sind? Ein typisches Beispiel ist eine Zusammenstellung von Ressourcen, zum Beispiel eine Sammlung von Artikeln. Die Aussage, dass ein Artikel der erste Artikel in der Sammlung ist, ist nur für diese Sammlung gültig, der Artikel ist nicht grundsätzlich der erste Artikel.

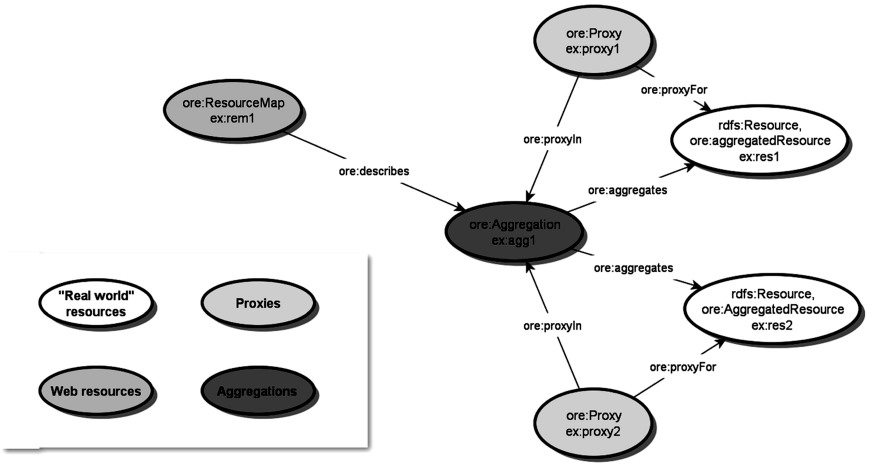

Abbildung 4: OAI-ORE.

Abbildung 4 zeigt das verwendete Datenmodell. ORE führt sogenannte Proxies ein (`ore:Proxy`). Ein Proxy ist eine spezielle Ressource, die als Platzhalter für die ursprüngliche Ressource (`ore:AggregatedResource`) innerhalb einer Zusammenstellung (`ore:Aggregation`) dient. Aussagen über den Proxy sind Aussagen über die ursprüngliche Ressource, aber eben nur gültig im Kontext der Zusammenstellung[9]. Aussagen über die Zusammenstellung selbst und ihre Verbindung zu den aggregierten Ressourcen werden über eine spezielle Ressource zur Verfügung gestellt, der Resource Map (`ore:ResourceMap`). Abgesehen von

9 Proxies sind in ORE eigentlich optional, Ressourcen können auch direkt aggregiert werden. Sie sind aber erforderlich, um verschiedene Beschreibungen auseinanderhalten zu können, eine Voraussetzung für Provenienzinformationen.

den Proxies folgt OAI-ORE damit genau den Linked-Data-Prinzipien, durch die Spezifikation von Resource Maps wird sogar vorgegeben, dass die Beschreibungen der Zusammenstellungen eigenständige Ressourcen mit eigener URI sind.

OAI-ORE wird für den Zweck der Provenienzerfassung insbesondere im aktuellen Europeana Datenmodell (EDM) verwendet. Ein grundsätzliches Problem beim Einsatz von OAI-ORE besteht darin, dass die Semantik von der üblichen Semantik in RDF abweicht, wo Aussagen direkt über Ressourcen getroffen werden. Eine Anwendung muss ORE „verstehen", um Aussagen über Proxies richtig interpretieren zu können. Das wird auch in der Dokumentation von ORE entsprechend berücksichtigt: Ein ORE-Server muss Anwendungen, die nicht „ORE aware" sind, unterschiedlich behandeln und via 303 Redirect direkt auf die aggregierte Ressource verweisen.

Eine nicht ORE-Anwendung stößt sonst auf semantische Probleme beim Versuch, ORE-Daten zu verstehen. Soll eine aggregierte Ressource zum Beispiel durch Dublin Core beschrieben werden (wie im EDM der Fall), so finden sich `dct:creator` Aussagen, die als Subjekt den Proxy verwenden, der ja als Platzhalter für die eigentliche Ressource fungiert. Das Prädikat `dct:creator` ist aber so definiert, dass das Subjekt die erstellte Ressource identifiziert und das Objekt die handelnde Person oder Einrichtung (Agent), die die Ressource erstellt hat. Wenn nun aber das Subjekt, also der Proxy, die erstellte Ressource ist, was ist dann die ursprüngliche Ressource im Datenmodell? Letztlich kann man aus den Aussagen folgern, dass alle Proxies identisch mit der ursprünglichen Ressource sind, was natürlich der Intention zuwiderläuft, verschiedene Aussagen über eine Ressource voneinander zu trennen.

Ein grundlegendes Problem, dass immer auftritt, wenn man versucht, Aussagen über Aussagen ohne die Nutzung eines zusätzlichen Levels auszudrücken, sind die resultierenden komplexen Datenmodelle.

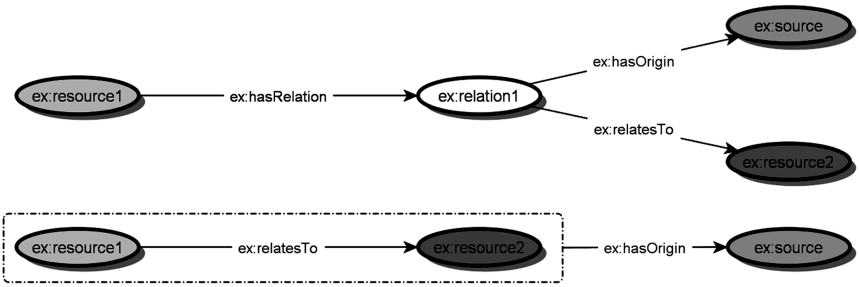

Abbildung 5: Zusätzliche Knoten oder die Einführung einer Metaebene

Abbildung 5 soll das verdeutlichen. Die klassische Vorgehensweise, in RDF (wie auch beim Modellieren relationaler Datenmodelle oder objektorientierter Architekturen) zusätzliche Informationen zu einer Aussage (Relation, Objektbeziehung) hinzuzufügen, ist die Einführung einer Zwischenebene, also eines zusätzlichen Knotens (Tabelle, Objekt). David Wheeler hat das treffend formuliert: „Alle Probleme in der Informatik können durch das Einfügen einer Zwischenebene gelöst werden… bis auf das Problem zu vieler Zwischenebenen.“[10]

In diesem Fall wird also eine zusätzliche Ressource eingeführt, die die Prädikatszuweisung in der ursprünglichen Aussage repräsentiert. Das Ziel der ursprünglichen Aussage, also das Objekt, wird entsprechend identifiziert, der neue Knoten wird über ein anderes Prädikat an das ursprüngliche Subjekt gehängt. Nun hat man die Möglichkeit, weitere Informationen, zum Beispiel zur Provenienz der Aussage, an die neue Ressource zu hängen, im Beispiel durch das Prädikat `ex:hasOrigin` dargestellt.

Nutzt man eine Metaebene, die es erlaubt, direkt Aussagen über andere Aussagen zu machen, ist nicht nur das Modell einfacher, die Interpretation ist auch viel intuitiver. Insbesondere bleibt die ursprüngliche Aussage erhalten und wird nicht durch die Zwischenebene auseinandergerissen. Analog zum Beispiel der Proxies in EDM ist ein solches Auseinanderreißen semantisch oft gar nicht möglich, weil die Definition der verwendeten Prädikate vorsieht, dass Subjekt und Objekt direkt verbunden werden. Dieser Umweg über eine Zwischenebene kommt also eigentlich nur für neu erstellte Ontologien in Betracht und kann nicht allgemein für vorhandene Daten verwendet werden.

Zusammenfassend kann man sagen, dass es schon jetzt verschiedene Möglichkeiten gibt, Provenienzinformationen für Linked Data darzustellen. Allerdings liegt die Tücke im Detail und letztlich ist keine Lösung uneingeschränkt anwendbar und interoperabel, insbesondere mangels Berücksichtigung im RDF-Standard. Es ist aber davon auszugehen, dass sich das mit der nächsten Version von RDF ändern wird.

DCAM: Eine Ontologie für (Meta-) Datensätze?

Zum Abschluss dieses Kapitels befassen wir uns noch mit Überlegungen, was noch benötigt werden könnte, sobald ein Mechanismus in RDF zur Identifikation von Graph-Containern zur Verfügung steht. Graph-Container werden dann zu eigenständigen Ressourcen und können damit beliebig beschrieben werden.

10 “All problems in computer science can be solved by another level of indirection … except for the problem of too many levels of indirection.” (Spinellis, 2007).

Sie werden für verschiedenste Zwecke zum Einsatz kommen und gegebenenfalls in Anwendungen dynamisch erzeugt, um beliebige Verwaltungsinformationen oder ähnliches strukturiert festhalten zu können. Damit bekommen Graph-Container eine anwendungsbezogene Bedeutung, die sich letztlich in verschiedenen *Klassen* von Graph-Containern niederschlagen wird.

Eine solche Klasse dürfte der Metadatensatz sein, sozusagen die Rückkehr des im Linked-Data-Umfeld totgesagten Records.[11] Die Anwendungsfälle sind vielfältig und kommen vor allem aus dem Bereich des Metadatenmanagements. Zum Beispiel kann man die Konformität der enthaltenen Daten mit bestimmten Standards oder Regelwerken festhalten, oder ihre Vollständigkeit im Bezug auf bestimmte Anforderungen.

Für derartige Szenarien bietet die Dublin Core Metadata Initiative Ansatzpunkte mit dem Dublin Core Abstract Model (DCAM) als Basis zur Beschreibung von Metadatensätzen und den Application Profiles (DCAP), einer formalisierten Angabe, welche Metadatenelemente für eine bestimmte Anwendung wie genau zu verwenden sind.

DCAM wird derzeit ebenfalls vor dem Hintergrund aktueller Entwicklungen überarbeitet, insbesondere Anforderungen aus dem Bereich der Metadatenprovenienz spielen dabei eine Rolle. Auch hier wird darauf geachtet, dass das Ergebnis kompatibel zu den Ergebnissen der W3C Provenance und RDF Arbeitsgruppen ist. Eine zu Named Graphs kompatible Erweiterung von DCAM speziell für die Erfassung von Provenienzinformationen haben wir dazu in Eckert, Garijo, & Panzer, Extending DCAM for Metadata Provenance, 2011, vorgestellt. Die Entwicklung eines Application Profiles für Provenienzinformationen ist geplant, sobald die Grundlagen in DCAM und RDF gelegt sind.

Zusammenfassung

In diesem Kapitel haben wir einen Überblick zur Darstellung von Provenienzinformationen im Linked-Data-Web gegeben. Insbesondere haben wir dabei aktuelle Entwicklungen aus verschiedenen Arbeitsgruppen zusammengefasst und sie zu gängigen Praktiken und tatsächlichen Anwendungen in Beziehung gesetzt. Das bedeutet natürlich, dass einige Aussagen in diesem Kapitel nur als vorläufig

11 Da ist sie wieder, die Karteikarte. Wieder einmal besteht die Möglichkeit, sie zum zentralen Element eines Datenmodells zu erklären. Allerdings wächst der Nutzen des Inhalts der Karteikarte beträchtlich, da die Inhalte (hoffentlich) mehr untereinander verknüpft sind, als mit der Karteikarte. Sie dient nur noch dem Zweck, der nie in Frage gestellt war: In einem Kontext zusammengehörige Informationen zusammenzufassen.

anzusehen sind. Wir haben aber versucht, uns auf im Wesentlichen feststehende Grundlagen zu beschränken und auch deswegen auf allzu tiefgreifende Beispiele und Details verzichtet.

Ein Gebiet, das wir vollkommen ausgeklammert haben, ist die Versionierung von Linked Data. PROV kann hier eine wichtige Rolle spielen, wenn es darum geht, die Veränderungen einer Linked-Data-Ressource festzuhalten. Schon als einfache Metadaten zur Entstehung einer Resource, aber erst Recht in Form einer vollständigen Provenienzkette, sind Provenienzinformationen die Basis für Vertrauen in Daten aus dem Web. Das Vertrauen, der sogenannte Trust-Layer, ist ein wichtiges Arbeitsgebiet im Bereich Semantic Web und Linked Data und wir können gespannt sein, wie sich die Entwicklungen zur Provenienz von Daten hier auswirken.

Zum Abschluss dieses Kapitels ist es angebracht, noch auf einen besonderen Aspekt der Provenienz einzugehen: Eine (aber nur eine) Motivation für die Bereitstellung von Provenienzinformationen kann eine rechtliche Verpflichtung sein, das heißt, der Besitzer der veröffentlichten Daten verlangt es. Wie wir gesehen haben, stellt eine adäquate Bereitstellung von Provenienzinformationen eine große technische Hürde dar, es ist fraglich, ob das überhaupt bis in letzter Konsequenz erreicht werden kann. Eine Möglichkeit, das Problem zu umgehen, ist die Freigabe der Daten in die Public Domain, das heißt, der Besitzer gibt alle Rechte an den Daten auf. Das ist die empfohlene Praxis für Daten aus der öffentlichen Hand, insbesondere auch für Bibliotheksdaten. Europeana verlangt die Freigabe aller übermittelten Daten, nicht zuletzt, um damit Interoperabilitätsproblemen wegen der rechtlichen Beschränkungen aus dem Weg zu gehen. In der Tat ist das auch nach wie vor der beste (wenn nicht einzige) Weg und das wird auch vom Autor dieses Kapitels nachdrücklich empfohlen.

Benötigt man dann Provenienzinformationen, wenn die Daten doch alle frei sind? Die Antwort ist ja, denn natürlich brauchen wir trotzdem die Information, aus welcher Quelle bestimmte Informationen stammen und wie die Beziehung zwischen verschiedenen Metadatensätzen ist. Nur so entsteht Vertrauen in die Daten und damit letztlich auch Vertrauen in die Anwendungen, die auf diesen Daten aufbauen. Und schließlich geht es ja bei der Freigabe von Daten nicht darum, dass man den Ersteller von wertvollen Daten nicht nennen will. Beim Weiterverarbeiten der Daten darf man nur nicht dazu verpflichtet sein, diese Information uneingeschränkt und in jedem Fall beizubehalten.

Die Forderung nach freien, offenen Daten wird also durch die Entwicklung geeigneter Mechanismen zur Provenienzerfassung nicht ungültig. Würde man die Angabe der Provenienz erzwingen (zum Beispiel durch eine CC-BY-Lizenz oder ähnliches), so müsste die volle Provenienzkette von allen Daten für alle Zeiten aufbewahrt und zur Verfügung gestellt werden. Daten sind keine Ressourcen, die

einfach nur konsumiert werden. Daten werden ständig gemischt, transformiert, integriert, angereichert und verbessert, zum Wohle der Anwendungen. Würde man die Provenienzinformationen ständig mitführen wollen, wäre der Anteil der Provenienzinformationen weit höher als der Anteil der Daten selbst. Die Daten würden schlicht und ergreifend unbenutzbar werden. Deswegen müssen Daten rechtlich offen zugänglich und frei sein.

Mit anderen Worten und als abschließende Zusammenfassung: Der Bedarf an Provenienzinformationen muss von den Anwendungen und Datennutzern getrieben sein, nicht von den Datenanbietern.

Danksagung

Dieses Kapitel basiert auf Teilen meiner Masterarbeit mit dem Titel „Metadata Provenance in Europeana and the Semantic Web“, die ich im Rahmen des Studiengangs Bibliotheks- und Informationswissenschaft an der Humboldt-Universität zu Berlin geschrieben habe. Die Grundlage dafür wiederum war meine Arbeit in der DCMI Metadata Provenance Task Group, zusammen mit Daniel Garijo und Michael Panzer. Ihnen sei hiermit in besonderem Maße gedankt.

Literaturverzeichnis

Baer, D. (Hrsg.). (2001). *Duden, Fremdwörterbuch* (7. Ausg.). Mannheim; Leipzig [u.a.] : Dudenverl.

Bizer, C., & Cyganiak, R. (2007). *The TriG Syntax*. Von http://www.wiwiss.fu-berlin.de/suhl/bizer/TriG/Spec/ abgerufen

Buneman, P., Khanna, S., & Wang-Chiew, T. (2001). Why and Where: A Characterization of Data Provenance. In J. Van den Bussche, & V. Vianu (Hrsg.), *Database Theory - ICDT 2001* (Bd. 1973, S. 316-330). Heidelberg: Springer. Von http://dx.doi.org/10.1007/3-540-44503-X_20 abgerufen

Carroll, J. J., Bizer, C., Hayes, P., & Stickler, P. (2005). Named Graphs, Provenance and Trust. *Proceedings of the 14th International Conference on World Wide Web (WWW) 2005, May 10-14, 2005, Chiba, Japan*, (S. 613-622).

Davidson, S. B., & Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (S. 1345-1350). New York, NY, USA: ACM. Von http://doi.acm.org/10.1145/1376616.1376772 abgerufen

Davies, J., German, D. M., Godfrey, M. W., & Hindle, A. (2011). Software bertillonage: finding the provenance of an entity. *Proceedings of the 8th Working Conference on Mining*

Software Repositories (S. 183-192). New York, NY, USA: ACM. Von http://doi.acm.org/10.1145/1985441.1985468 abgerufen

DCMI, U. B. (2010). *DCMI Metadata Terms*. Dublin Core Metadata Initiative. Von http://dublincore.org/documents/2010/10/11/dcmi-terms/ abgerufen

DCMI, U. B. (2010). *Dublin Core Metadata Element Set, Version 1.1*. Dublin Core Metadata Initiative. Von http://dublincore.org/documents/2010/10/11/dces/ abgerufen

Eckert, K., Garijo, D., & Panzer, M. (2011). Extending DCAM for Metadata Provenance. *DC-2011: International Conference on Dublin Core and Metadata Applications*.

Eckert, K., Garijo, D., & Panzer, M. (2011). Metadata Provenance: Dublin Core on the Next Level. *DC-2011: International Conference on Dublin Core and Metadata Applications*.

Eckert, K., Pfeffer, M., & Stuckenschmidt, H. (2009). A Unified Approach For Representing Metametadata. *DC-2009: International Conference on Dublin Core and Metadata Applications*.

Eckert, K., Pfeffer, M., & Völker, J. (2010). Towards Interoperable Metadata Provenance. *Proceedings of the Second International Workshop on the role of Semantic Web in Provenance Management (SWPM 2010)*. Von http://ceur-ws.org/Vol-670/ abgerufen

Green, T. J., Karvounarakis, G., & Tannen, V. (2007). Provenance semirings. *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (S. 31-40). New York, NY, USA: ACM. Von http://doi.acm.org/10.1145/1265530.1265535 abgerufen

Hawke, S. (2011). RDF-ISSUE-25 (Deprecate Reification): Should we deprecate (RDF 2004) reification? [Cleanup tasks]. *RDF-ISSUE-25 (Deprecate Reification): Should we deprecate (RDF 2004) reification? [Cleanup tasks]*. Von http://lists.w3.org/Archives/Public/public-rdf-wg/2011Apr/0164.html abgerufen

Heath, T., & Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space* (1st Ausg.). Morgan \& Claypool. Von http://linkeddatabook.com/editions/1.0/ abgerufen

Hillmann, D. I., Dushay, N., & Phipps, J. (2004). Improving Metadata Quality: Augmentation and Recombination. *DC-2004: Proceedings of the International Conference on Dublin Core and Metadata Applications*. Dublin Core Metadata Initiative. Von http://hdl.handle.net/1813/7897 abgerufen

Moreau, L. (2010). The Foundations for Provenance on the Web. *Foundations and Trends in Web Science, 2*(2--3), 99-241. Von http://eprints.soton.ac.uk/271691/ abgerufen

Open Archives Initiative. (2008). *Open Archives Initiative - Object Reuse and Exchange: ORE User Guide - Primer*. (C. Lagoze, H. van de Sompel, P. Johnston, M. Nelson, R. Sanderson, & S. Warner, Hrsg.) Open Archives Initiative. Von http://www.openarchives.org/ore/1.0/primer abgerufen

RDF Data Access Working Group. (2008). *SPARQL Query Language for RDF*. (E. Prud'hommeaux, & A. Seaborne, Hrsg.) W3C. Von http://www.w3.org/TR/rdf-sparql-query/ abgerufen

Spinellis, D. (2007). Another level of indirection. In A. Oram, & G. Wilson (Hrsg.), *Beautiful code* (S. 279-291). Sebastopol, CA, USA: O'Reilly and Associates. Von http://www.dmst.aueb.gr/dds/pubs/inbook/beautiful_code/html/Spi07g.html abgerufen

W3C OWL Working Group. (2009). *OWL 2 Web Ontology Language Document Overview*. W3C. Von http://www.w3.org/TR/owl2-overview/ abgerufen

W3C Provenance Incubator Group. (2010). *Provenance XG Final Report -- W3C Incubator Group Report 08 December 2010*. Tech. rep., W3C. Von http://www.w3.org/2005/Incubator/prov/XGR-prov/ abgerufen

W3C Provenance Working Group. (2012). *PROV Model Primer (Working Draft)*. (Y. Gil, & S. Miles, Hrsg.) W3C. Von http://www.w3.org/TR/2012/WD-prov-primer-20120503/ abgerufen

W3C Provenance Working Group. (2012). *PROV-DM: The PROV Data Model (Working Draft)*. (L. Moreau, & P. Missier, Hrsg.) W3C. Von http://www.w3.org/TR/2012/WD-prov-dm-20120503/ abgerufen

W3C Provenance Working Group. (2012). *PROV-O: The PROV Ontology (Working Draft)*. (T. Lebo, S. Sahoo, & D. McGuinness, Hrsg.) W3C. Von http://www.w3.org/TR/2012/WD-prov-o-20120503/ abgerufen

W3C SWEO Interest Group. (2008). *Cool URIs for the Semantic Web: W3C Interest Group Note 03 December 2008*. (L. Sauermann, & R. Cyganiak, Hrsg.) W3C. Von http://www.w3.org/TR/cooluris/ abgerufen