Currency Conversion the Linked Data Way

Alex Stolz and Martin Hepp

E-Business and Web Science Research Group, Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, D-85577 Neubiberg, Germany
{alex.stolz,martin.hepp}@ebusiness-unibw.org

Abstract. In many business applications that could be built on the basis of Linked Open Data, the conversion of monetary amounts from one currency to another is a much-needed functionality. For example, hotel prices in a comparison shopping application may have to deal with prices given in differing currencies. While currency conversion APIs exist on the Web, their integration into operations over RDF data is still burdensome and requires proprietary code. In this paper, we propose to integrate currency conversion functionality from open Web APIs into the Linked Open Data (LOD) cloud in a conceptually clean, scalable way that (1) adheres to the LOD design guidelines, (2) removes the need for proprietary code, (3) can be accessed from client-side JavaScript, and (4) works with any standard SPARQL processor that is able to retrieve a RDF representation by dereferencing a resource URI. We argue that our solution serves as a good generic pattern for integrating Web APIs into the LOD cloud, beyond the practical relevance of the concrete implementation.

Keywords: Exchange rates, currency conversion, ISO 4217, linked data, LOD, RDF, SPARQL, JavaScript, Web APIs

1 Introduction

The integration of multiple data sources on the Web of Linked Data implies collecting data with heterogeneous representations in terms of data formats, structures and semantics. Structured data on the Web may appear in different formats such as RDFa, Microdata, Turtle, N-Triples or RDF/XML. Furthermore, it may represent similar facts by using distinct Web vocabularies, and draw on varying element codes within standards (meter vs. feet, gram vs. kilogram, U.S. dollar vs. Euro, etc.). Accordingly, unit codes published on the Web often show regional and application-specific differences. In the context of currency conversions for example, while financial transactions within countries are typically conducted using domestic currencies (e.g. Yen in Japan, Swiss franc in Switzerland), international trade is predominated by monetary exchanges based on global/world currencies, namely U.S. dollar and Euro.

For better interoperability between applications, an internationally recognized representation of currencies was established, i.e. the ISO 4217 standard [12]. The three-letter currency codes defined in that standard are well known and widely accepted by businesses and business applications. Euros, for example, are represented by the cur-

rency code EUR, whereas GBP describes British pounds and USD denotes U.S. dollars. According to the currency code table[1] published in ISO 4217 [12], as of today at least 180 accepted currencies exist world-wide. Currency codes, if they co-occur with price values, serve to disambiguate monetary amounts. In particular, they can help to distinguish prices by independent bodies such as companies operating in different countries and/or trading with different partners. The value of a currency with respect to another currency is determined by the currency exchange rate. On the foreign exchange market, the exchange rate is determined by a currency pair (e.g. USD/EUR) that consists of a *base* (primary, transaction, quoted) currency and a *counter* (secondary, reference, quote) currency. Large institutions, such as Citibank, publish latest exchange rates on a regular basis. Currency conversion services on the Web then take advantage of these exchange rates, offering translations of monetary amounts available in different currencies. Providers of currency conversion services encompass online portals like OANDA, XE or Reuters, but also search engines like Google and Yahoo. Furthermore, dedicated Web APIs[2] for currency conversion are available.

In many business applications that could be built on the basis of Linked Open Data [2, 4, 10], the conversion of monetary amounts from one currency to another is a much-needed functionality. If a comparison shopping website of hotel prices is required to compile a list of the *n* cheapest hotel room offers relative to the preferred currency of the user, it will likely have to deal with prices declared in a number of currencies. Despite the existence of currency conversion APIs on the Web, their integration into operations over RDF data is still burdensome and requires proprietary code. Up to now, SPARQL support for external Web services is poor. Even if some query engines could be extended to invoke external Web services using proprietary functions, SPARQL implementations in general do not offer a straightforward and generic way to realize on-the-fly currency conversions over RDF data.

In this paper, we propose to integrate currency conversion functionality from open Web APIs into the Linked Open Data (LOD) cloud in a conceptually clean, scalable way. In particular, we present an online service for exchange rates that (1) adheres to the LOD design guidelines, (2) removes the need for proprietary code, (3) can be accessed from client-side JavaScript, and (4) works with any standard SPARQL processor that is able to retrieve a RDF representation by dereferencing a resource URI. We argue that our solution serves as a good generic pattern for integrating Web APIs into the LOD cloud, beyond the practical relevance of the concrete implementation. We show how our approach enables to easily convert between prices in RDF, thus making products and services comparable irrespective of the currencies used.

The rest of the paper is organized as follows: Section 2 presents our approach, addressing its most important features and implementation details; in Section 3 we show how our proposal can be used with real datasets; Section 4 discusses our contribution with respect to relevant previous efforts to provide currency and unit conversions in the context of the Semantic Web; finally, Section 5 concludes the paper.

---

[1] http://www.currency-iso.org/content/dam/isocy/downloads/dl_iso_table_a1.xml currently lists 182 distinct currencies.

[2] https://openexchangerates.org/

2 Exchange Rates for Linked Open Data

In this section, we propose a RESTful [8] Web service for currency exchange rates intended for the integration into the LOD cloud. Our approach is based on two main building blocks. Firstly, we propose an OWL ontology that provides the means to model currency exchange rates in RDF. Secondly, we present a Web service that we put online to serve RDF representations populated with exchange rates, constantly refreshed by means of a Web API for currency conversions. We also present the supported URI patterns, content negotiation, and how we ensure reliable caching.

2.1 Exchange Rate Ontology

The schema (vocabulary, data dictionary, ontology) underlying our approach is defined in terms of the Exchange Rate Ontology (XRO, prefixed with *xro:*), which language description is available online[3]. The UML diagram depicted in Fig. 1 describes the conceptual model of the vocabulary.

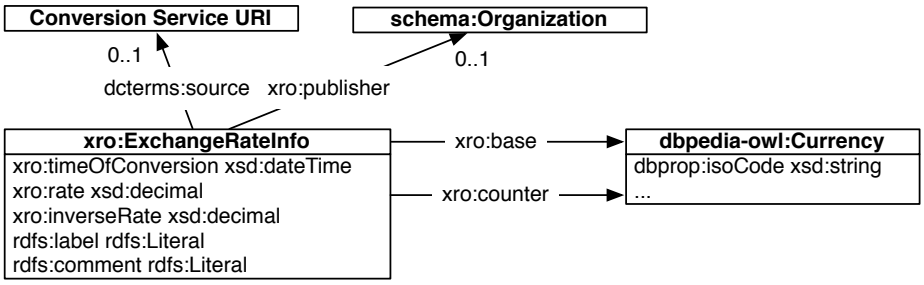

Fig. 1: UML class diagram of the Exchange Rate Ontology

ExchangeRateInfo represents the core class of the Exchange Rate Ontology and provides the exchange rate to a given currency pair. Every currency pair consists of a *base* and a *counter* currency, as represented in the model by two properties pointing to the *Currency* class of DBPedia[4]. Harnessing currency instances defined in DBPedia can provide useful additional information about currencies that could be queried. The *ExchangeRateInfo* class consists of three attributes, namely *timeOfConversion*, *rate* and *inverseRate*. The attribute *rate* describes the actual exchange rate value of two currencies, whereas *inverseRate* is a convenience attribute and embodies the inverse value of *rate*. The timestamp value pointed at by *timeOfConversion* can be used to determine the creation date of the class instance. Human-readable descriptions using *rdfs:label* and *rdfs:comment* may further refine the *ExchangeRateInfo* class. In order to supply basic provenance information about the conversion, one could optionally provide a URI of the conversion API and/or supply the publishing institution (e.g. Federal Reserve Bank, Citibank, etc., using the corresponding DBPedia resource

[3] http://purl.org/xro/
[4] http://dbpedia.org/

URI) of the exchange rate. As an extension to the model outlined above, one might also consider to take into account the place of trade (e.g. New York Stock Exchange, Philadelphia Stock Exchange), type of transaction (e.g. cash or non-cash), or type of market (interbank, spot, forward, etc.) with potential future prices and delivery dates. However, for the sake of simplicity, the current model does not cover such aspects.

The following listing gives an example of the vocabulary usage in N3 format, populated with Euro and U.S. dollar as the base and counter currencies:

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xch_EUR: <http://www.currency2currency.org/EUR#> .
@prefix xro: <http://purl.org/xro/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

xch_EUR:USD a xro:ExchangeRateInfo;
 rdfs:label "Euros to U.S. dollar"@en;
 rdfs:comment "1 EUR = ? USD"@en;
 xro:base dbpedia:Euro;
 xro:counter dbpedia:United_States_dollar;
 xro:rate "1.31010"^^xsd:decimal;
 xro:inverseRate "0.7633010"^^xsd:decimal;
 dcterms:source <http://www.google.com/ig/calculator?hl=en&q=1EUR=?USD>;
 xro:timeOfConversion "2013-04-11T00:00:02Z"^^xsd:dateTime .
```

The example in the listing above indicates that we used the *source* property of the *dcterms* vocabulary to add provenance information to the RDF dataset, thus allowing to keep track of the Web API that generated the exchange rates. The *dterms:source* property turned out to be appropriate for our provenance needs, being defined as a "related resource from which the described resource is derived" [6].

2.2 Web Service for Exchange Rates

Fig. 2 outlines the conceptual architecture of the exchange rate service for Linked Open Data. The service is based on the Exchange Rate Ontology, and equipped with sophisticated storage and caching strategies for resource-saving and fast delivery. It has been developed as a Google App Engine application and was deployed online[5].

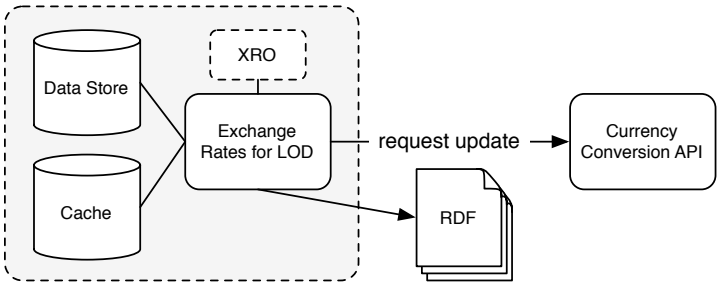

Fig. 2: Conceptual architecture of the Exchange Rates service for LOD

[5] http://www.currency2currency.org/

Our Web service pulls its exchange rates from the unofficial[6] Google Calculator service that, again, procures the exchange rates from Citibank. The currency conversion service returns a JSON (JavaScript Object Notation) [5] string in response to the URI pattern provided by the following example, which calculates the exchange rate of Euro with respect to U.S. dollar:

```
http://www.google.com/ig/calculator?hl=en&q=1USD=?EUR
```

At present, exchange rates for roughly 80 currencies are supported. Due to the symmetry between exchange rates, only about 80 service requests to the currency conversion API are necessary to calculate all possible combinations of exchange rates. The explanation for this is that given a base currency (*BASE*), the exchange rate between currencies *A* and *B* (*A2B*) can be derived from the ratio of the exchange rates of the two currencies relative to the base currency (*A2BASE* and *B2BASE*), as characterized by the formula given below:

$$rate_{A2B} = \frac{rate_{A2BASE}}{rate_{B2BASE}} \quad (1)$$

A cron job scheduled for everyday at midnight (server time zone) takes care of periodic updates of the currency exchange rates. Old exchange rates are properly archived and can be requested any time, as we will see in the upcoming Section 2.3. The subsequent sections will then discuss content negotiation and caching.

2.3 URI Pattern

The service distinguishes between multiple URI patterns, each serving a different purpose. The supported URIs obey the basic idea of cool URIs [1, 18], as such being

- *Simple*: Mnemonic, memorable names for URIs,
- *Stable*: Hiding implementation details, subject to change as technology advances,
- *Manageable*: Manageable, stable URIs over time.

The service uses three basic URI patterns: the first delivers the exchange rate between two particular currencies; the second lists all exchange rates with respect to a given base currency; and the third yields a full dump of all exchange rates.

The most efficient and lean method of taking advantage of the service is to request the exchange rate of a specific currency pair. The placeholders *base* and *counter* indicated in the URI pattern below have to be replaced by valid ISO 4217 currency codes:

```
http://www.currency2currency.org/<base>/<counter>
```

A more costly, yet more powerful way is to list all exchange rates with respect to a given base currency. This method will meet most requirements, since all possible combinations of exchange rates can be computed out of it. The respective URI pattern is composed as follows:

[6] Google Calculator service is not officially sanctioned. However, the specific calculator API is irrelevant for the proper functioning of our service and could be replaced easily.

```
http://www.currency2currency.org/<base>
```

The omission of any currency codes in the URI pattern produces a complete RDF dump. The huge RDF graph consists of a materialization of all possible combinations of currency pairs, i.e. n^2 currency exchange rates for n currencies:

```
http://www.currency2currency.org/
```

The three basic URI patterns can be expanded with an optional date string (*YYYYMMDD*, e.g. *20130411*), which causes the service to return a snapshot for that particular date. This can be useful for looking up historical data about exchange rates:

```
<uri>/YYYYMMDD
```

The two URIs indicated below both give the same result if retrieved on April 11, 2013. However, if accessed on different dates, they will describe distinct datasets:

```
http://www.currency2currency.org/USD/EUR
http://www.currency2currency.org/USD/EUR/20130411
```

The base URIs used for identifying the instances of *ExchangeRateInfo* in RDF are irrespective of the request URI. The URI pattern is hash-based of the form

```
http://www.currency2currency.org/<base>#
```

For instance, the exchange rate between U.S. dollar and Euro is described by

```
http://www.currency2currency.org/USD#EUR
```

2.4 Content Negotiation

The service uses HTTP content negotiation ([7], Section 12) to serve various RDF serializations via a single URI, negotiated based on the media type supplied in the HTTP request header. A client application can use proper HTTP *Accept*-headers ([7], Section 14.1), potentially including quality factors for ranking preferences, to indicate its preferred RDF materialization of exchange rates. The following media types are currently supported: HTML, and a series of RDF serializations, namely RDF/XML, N3, Turtle, RDF/JSON and N-Triples. Table 1 shows correspondences between media types and serialization formats as recognized by the Web service. The second column denotes the media types that the service accepts, the third column the returned content types. Generally, if a media type preference is invalid, unsupported or missing, the service will return HTML with its corresponding content type. Accordingly, entering the URIs of the types presented in Section 2.3 into a Web browser results in a human-readable HTML description of the respective pages together with usage instructions.

The following *curl* command fetches exchange rates relative to Swiss francs in N3:

```
curl -H "Accept: text/n3;q=1.0" http://www.currency2currency.org/CHF
```

The HTTP response header returned by the server then looks as follows:

Table 1: Mapping of serialization formats and media types

Serialization format	**Media types accepted** *Accept:*	**Content type delivered** *Content-Type:*
HTML	*not available* text/html application/xhtml+xml	text/html
RDF/XML	application/rdf+xml application/xml	application/rdf+xml
N3	text/n3 text/rdf+n3 application/n3	text/n3
Turtle	text/turtle application/x-turtle	text/turtle
RDF/JSON	application/json text/rdf+json text/javascript	application/json
N-Triples	text/plain	text/plain

```
HTTP/1.1 200 OK
Content-Location: http://www.currency2currency.org/CHF
Access-Control-Allow-Origin: *
Vary: Accept
Cache-Control: max-age=3600, must-revalidate
Content-Type: text/n3
Content-Length: 31602
Date: Thu, 11 Apr 2013 10:31:55 GMT
```

The header determines that the page could be retrieved successfully. The content type is *text/n3* that corresponds to the expected media type according to Table 1. Furthermore, the header field *Vary* indicates to the client that the value of the request header field *Accept* supplied by the client was considered to select among multiple representations ([7], Sections 14.1 and 14.44). In that sense it constitutes a nice example of the *identity* axiom in Web architecture [3]. The identity axiom states that the identity for a URI is determined by the owner of the resource and not imposed by the technology, and that the existence of different representations for the same URI should be signaled appropriately, i.e. using the *Vary* response header field. Another notable header information returned by the server is that cross-origin resource sharing (CORS) [13] is enabled (*Access-Control-Allow-Origin:* *), meaning that JavaScript clients can take advantage of the response message irrespective of the same origin security policy enforced by most Web browsers.

If a requested page does not (yet) exist, the service will reply with a plain error page and indicate it with a status code *404 Not Found* in the HTTP response header.

2.5 Caching and Expiry

As already depicted in Fig. 2, the service takes advantage of a data store (comparable to a database, but schemaless) and advanced caching of datasets at three levels,

namely client cache, memcache, and file storage (cf. Table 2). The type of caching strategy chosen addresses the peculiar natures of the different-sized RDF datasets, which ensures an efficient dealing with subsequent client requests; it fosters timely responses and reduces the server load.

Table 2: Comparison of storage and caching mechanisms

	Data store	**Client cache**	**Memcache**	**File store**[7]
Type	NoSQL (schemaless) object data store	Local cache of client application	Distributed memory object caching system	Data object store for large files
Expiry	Never	After 1 hour	After 6 hours	Never
Update frequency	Daily	Cache lifetime expired	Cache limits or lifetime reached	Daily
Memory limits	Limited to storage capacity of used data type	Unlimited, application-dependent	1 Megabyte	Limited to data transfer limit
Intended usage	Daily updated list of exchange rates	Downloaded page contents	Small-sized, partial RDF serializations	Complete RDF dumps

The data store records exchange rates of a particular date, while the caches take care of a timely and resource-saving delivery of the respective RDF [15] representations. While smaller-sized datasets (i.e. exchange rates relative to a base currency or between a currency pair) are kept in memcache for a specific time, bigger datasets (complete RDF dumps) that easily exceed the maximum memcache size (e.g. one megabyte) are pre-fetched on a daily basis, immediately after the update of exchange rates becomes available, and then stored in the data cloud. Caching on client side complements caching at the server by providing HTTP response headers with cache lifetime values (cf. Section 2.4) that instruct client applications to cache contents for an indicated time period, e.g. an hour (3600 seconds).

```
Cache-Control: max-age=3600, must-revalidate
```

The records in the data store and the permanent storage of dump files permit to cache historic data for a longer period, thus to keep track of exchange rates over time.

3 Usage Scenario

In this section, we present an example in SPARQL 1.1 [9] that uses our LOD exchange rate service to turn arbitrary price values into corresponding prices based on a particular currency. The formula that applies for this purpose is provided by

$$price_A = rate_{A2BASE} \cdot price_{BASE} \quad (2)$$

where *BASE* denotes the base currency, and *A* represents an arbitrary currency A.

[7] In the context of Google App Engine, it is called *Blobstore*.

For our example, we took a collection of Web shops that offer products and services in GoodRelations [11], expressed using different prices and currencies. The goal is to normalize prices, i.e. to provide a means to operate with prices based on a common currency. This will enable us to sort items based on price values and to return the *n* cheapest offers (cf. Section 1). We assume that all Web shop data and exchange rates related to U.S. dollar[8] have already been loaded into the SPARQL endpoint. However, SPARQL 1.1 [9] added federated query support to facilitate queries that include data of remote SPARQL endpoints. Some SPARQL endpoints, e.g. Virtuoso Open Source[9], actually provide means for fetching remote RDF data on the fly, even though this feature is commonly discouraged for production use due to security risks. The following listing shows the SPARQL query for currency conversion:

```
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX xro: <http://purl.org/xro/ns#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpprop: <http://dbpedia.org/property/>

SELECT DISTINCT ?price ?code (?price/?rate AS ?base_price) ?base_code
WHERE {
  ?s a gr:Offering; gr:hasPriceSpecification ?pspec .
  ?pspec gr:hasCurrency ?code; gr:hasCurrencyValue ?price .
  ?xrate xro:rate ?rate;
    xro:base ?base_currency; xro:counter ?counter_currency .
  ?base_currency dbpprop:isoCode ?base_code .
  ?counter_currency dbpprop:isoCode ?counter_code .
  FILTER(str(?counter_code) = str(?code))
}
ORDER BY ?base_price LIMIT 5
```

The query seeks price values and currency codes of items in the dataset, for both the original and the converted price. For that it selects proper exchange rates from the RDF graph, which as we know are all relative to *USD*. In particular, it matches the currency code of every item with the corresponding currency codes of DBPedia instances. They, again, can be used to pick the associated exchange rates from the dataset. Table 3 lists the five results compiled from a real dataset[10] by executing the query outlined above. The results returned by the endpoint contrast original price and currency (U.S. dollar and Romania Leu) with target price and currency (U.S. dollar).

Table 3: SPARQL query results of a currency conversion

price	code	base_price	base_code
0.0	USD	0.0	USD
1.29	RON	0.38123885	USD
1.93	RON	0.5703806	USD
2.58	RON	0.7624777	USD
3.13	RON	0.92502147	USD

[8] USD exchange rates as of 10.04.2013: http://www.currency2currency.org/USD/20130410
[9] http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/
[10] The data was extracted from a collection of Web shops with RDFa markup and is privately maintained by the E-Business and Web Science Research Group.

The SELECT query in the example could easily be altered to CONSTRUCT queries over the whole dataset, i.e. to normalize all available prices and return them for being consumed by other RDF-capable applications.

4 Discussion

In this section, we review and compare existing alternatives with our own proposal.

4.1 Related Work

Despite the availability of numerous Web APIs for currency conversion on the Web, the most relevant works to our proposal are conversion approaches aimed for the Semantic Web. To the best of our knowledge, there exist no RDF-based Web services dedicated to exchange rates. However, modeling of currency exchange rates was sometimes addressed as part of other modeling tasks, e.g. comprised in vocabularies for units of measurement. QUDT[11] is a vocabulary for quantities, units, dimensions and types, but also includes currency units for all of the world's currencies. The vocabulary subset for currencies is similar to our approach. QUDT is targeted for use with the SPIN[12] (SPARQL Inferencing Notation) engine. Accordingly, currency conversion is done using the most recent exchange rates that are retrieved by invoking a SPIN function to call an external API [14]. Another effort to model quantities and units of measurement in OWL is the Ontology of Units of Measure and Related Concepts (OM) [17]. However, unlike QUDT, OM does not support currency units. In the context of the European LOD2 project[13], a dataset with exchange rates relative to Euro was published in RDF. The currency exchange rates are available from a SPARQL endpoint[14] and contain historical data about exchange rates.

4.2 Our Contribution

As opposed to existing approaches, which are mostly limited to provide a data model for currency conversion and units of measurement, we suggest a fully-fledged framework that consists of a Web vocabulary for exchange rates and a RDF-based service for currency conversions based on current or historical exchange rates. Our service could serve as an authoritative source of exchange rates compliant with Linked Open Data. For example, it adheres to the Linked Data design principles as stated in [2]: (1) every exchange rate entity obtains a named URI, (2) exchange rates can be looked up easily because relying on HTTP URIs, (3) when someone looks up a URI of an exchange rate, useful information is displayed to humans and machines

[11] http://www.qudt.org/
[12] http://spinrdf.org/
[13] http://lod2.eu/
[14] http://linked.opendata.cz/sparql

alike, enabled by content negotiation that returns either HTML or RDF descriptions, and (4) the dataset provides links to other datasets, e.g. to currency instances of DBPedia, to discover additional things.

In contrast to traditional Web APIs, our approach is ready-to-use for the Web of Data, i.e. users are not expected to provide proprietary code and can integrate the RDF representation instantly into their RDF-aware applications. The service is accessible to JavaScript applications by sending proper CORS HTTP headers.

Our proposal is not limited to the domain of currency conversion. It could likewise be applied to other fields relevant for inclusion in the LOD cloud. For that purpose it would be sufficient to replace the domain model and the Web service of our approach by equivalents in the target domain.

4.3 Future Extensions

Our proposal could of course benefit from future extensions. On the conceptual side, we envision to extend the Exchange Rate Ontology model with additional domain-related properties and concepts, as already indicated in Section 2.1.

On the technical part, our approach could benefit from hosting the currency exchange rate data in a RDF store, which would empower SPARQL-1.1-capable endpoints to execute federated SPARQL queries without having to import the currency exchange rate dataset first.

Furthermore, the service could be made operable with different Web services, thereby introducing an additional level of trust and better accuracy by taking advantage of multiple currency converters. The provenance information about the source API contained in the RDF graph could provide a helpful starting point for this. The planned model enhancements could then contribute additional granularity.

5 Conclusion

It is difficult to ensure interoperability across multiple data sources. This is also true for prices and currencies in the Linked Open Data (LOD) cloud. However, in order to do useful product matchmaking [16], it is indispensable to have a uniform view on such data, since prices are among the most decisive buying criteria in e-commerce. While currency conversion APIs exist on the Web, their integration into operations over RDF data still involves significant manual effort.

To overcome this limitation, we proposed a way to integrate currency conversion functionality from open Web APIs into LOD. For this purpose we introduced a conceptual schema for currency exchange rates and complemented it with a RESTful Web service for exchange rates in RDF, which again takes advantage of freely available currency converter APIs on the Web. The benefits of our approach can be summarized as follows: our service adheres to LOD design principles, it eliminates the need for writing proprietary code, it is accessible to latest Web technologies (e.g. JavaScript), and compatible with standard SPARQL processors. In addition, it employs caching and storage of exchange rates while keeping track of historical data. Thus we

suggest that our approach could serve as a generic pattern for integrating open Web APIs into LOD, such as services for unit conversion or product review data that could add important value to better e-commerce matchmaking. As a proof of concept for our proposal, we showed how easy it is to accomplish currency conversions in SPARQL environments by taking advantage of our service.

Acknowledgments. Parts of the work presented in this paper have been supported by the German Federal Ministry of Research (BMBF) by a grant under the KMU Innovativ program as part of the Intelligent Match project (FKZ 01IS10022B).

References

1. Berners-Lee, T.: Cool URIs don't change, http://www.w3.org/Provider/Style/URI
2. Berners-Lee, T.: Linked Data - Design Issues, http://www.w3.org/DesignIssues/LinkedData.html
3. Berners-Lee, T.: Universal Resource Identifiers -- Axioms of Web Architecture, http://www.w3.org/DesignIssues/Axioms.html
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems. 5, 3, 1–22 (2009)
5. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON), http://www.ietf.org/rfc/rfc4627.txt
6. Dublin Core Metadata Initiative Usage Board: DCMI Metadata Terms, http://dublincore.org/documents/2012/06/14/dcmi-terms/
7. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt
8. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis. University of California, Irvine (2000)
9. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language, http://www.w3.org/TR/sparql11-query/
10. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)
11. Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008). pp. 332–347. Springer, Acritezza, Italy (2008)
12. ISO: ISO 4217:2008: Codes for the representation of currencies and funds. (2008)
13. Van Kesteren, A.: Cross-Origin Resource Sharing, http://www.w3.org/TR/cors/
14. Knublauch, H.: Currency conversion with the Units Ontology, SPARQLMotion and SPIN, http://composing-the-semantic-web.blogspot.it/2009/09/currency-conversion-with-units-ontology.html
15. Manola, F., Miller, E.: RDF Primer, http://www.w3.org/TR/rdf-primer/
16. Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: A System for Principled Matchmaking in an Electronic Marketplace. Proceedings of the 12th International World Wide Web Conference (WWW 2003). pp. 321–330. ACM, Budapest, Hungary (2003)
17. Rijgersberg, H., Van Assem, M., Top, J.L.: Ontology of units of measure and related concepts. Semantic Web. 4, 1, 3–13 (2013)
18. Sauermann, L., Cyganiak, R.: Cool URIs for the Semantic Web, http://www.w3.org/TR/cooluris/#cooluris