

## **SECO: Mediation Services for Semantic Web Data**

Andreas Harth

DERI Technical Report 2004-04-19

April 2004

**DERI Galway**

University Road  
Galway  
IRELAND  
[www.deri.ie](http://www.deri.ie)

**DERI Innsbruck**

Technikerstrasse 13  
A-6020 Innsbruck  
AUSTRIA  
[www.deri.at](http://www.deri.at)

# SECO: Mediation Services for Semantic Web Data

Andreas Harth, NUI Galway/DERI  
andreas.harth@deri.ie

## Abstract

*The Semantic Web has motivated grassroots efforts to develop and publish ontology specifications in RDF. So far, the large amount of RDF data available online has not been utilized to the extent possible. SECO, the application presented in this paper, aggregates, integrates, and displays RDF data obtained from the Semantic Web. SECO collects the RDF data available in files using a crawler, and also utilizes RDF repositories as sources for data. Integration tasks over the various data sources, such as object consolidation and schema mapping, are carried out using a reasoning engine and are encapsulated in mediators to which software agents can pose queries using a remote query interface. SECO includes a user interface component that emits HTML, which allows for human users to browse the integrated data set.*

# 1 Introduction

The Semantic Web has motivated grassroots efforts to develop and publish ontology specifications, and netizens have provided hand-crafted instance data in RDF online or have exported RDF from within applications. Examples for community-specified ontologies include FOAF (Friend Of A Friend) that is used in the Semantic Web community to describe people and their relationships, and RDF Site Summary 1.0 (RSS) that is used in the weblog community to syndicate news items. Both formats are encoded in RDF, are based on relatively compact ontologies, and enjoy wide community support. Most of the FOAF files are connected via links and therefore represent a web of data that has been built by a collaborative effort of thousands of people.

## Sidebar: Data on the Web

### Friend of a Friend

The Friend of a Friend (FOAF) project is about creating a Web of machine-readable homepages describing people, the links between them and the things they create and do. Netizens put FOAF files typically in flat files of a few kilobytes of size and make them accessible via HTTP, similar to HTML documents on the HTML web. In FOAF, some descriptions about a person only contain the bare minimum: a name and an email address. Other FOAF files are more extensive and contain — besides the minimal properties — also descriptions about a person's affiliation, homepages, workplace, pictures, and relationships with other people. The example RDF file shown in Figure #FOAF describes the author, his email address and homepage, and provides links to other RDF files via the `rdfs:seeAlso` property. FOAF is mainly created manually or via the foaf-a-matic form at <http://www.ldodds.com/foaf/foaf-a-matic.html>. The FOAF ontology specification is online at <http://xmlns.com/foaf/0.1/>.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">

  <foaf:Person>
    <foaf:name>Andreas Harth</foaf:name>
    <foaf:firstName>Andreas</foaf:firstName>
    <foaf:surname>Harth</foaf:surname>
    <foaf:nick>aharth</foaf:nick>
    <foaf:mbox rdf:resource="mailto:andreas.harth@deri.ie"/>
    <foaf:homepage rdf:resource="http://www.harth.org/andreas/" />
    <foaf:phone rdf:resource="tel:+353-91-512-651" />
    <foaf:workplaceHomepage rdf:resource="http://www.deri.ie/" />
    <rdfs:seeAlso rdf:resource="http://www.isi.edu/~aharth/foaf.rdf" />
    <foaf:knows>
      <foaf:Person>
        <foaf:name>Stefan Decker</foaf:name>
        <foaf:mbox rdf:resource="mailto:mailto:stefan@deri.ie" />
```

```
<rdfs:seeAlso
rdf:resource="http://www.isi.edu/~stefan/foaf.rdf"/>
</foaf:Person>
</foaf:knows>
</foaf:Person>
</rdf:RDF>
```

*Figure FOAF. An RDF file using FOAF to describe the author; links to other FOAF files are provided using `rdfs:seeAlso`.*

## RDF Site Summary

RDF Site Summary (RSS) is a lightweight multipurpose extensible metadata description and syndication format. News sites such as Slashdot and weblog sites use RSS to provide a machine-readable description containing metadata about the news items on the sites. Some sites only publish a title and a short description of the news item. Other sites provide richer metadata to describe the news items, such as dates, category information, author, and other metadata. Although RSS 1.0 is defined in a specification, blog and site management software use a variation of properties for the RDF they write (date for example, `rss:description` vs. `dc:description`), thus making integration of the various RSS flavors challenging. There are several versions of RSS in use (0.9x, 1.0, 2.0), of which versions 0.9x and 2.0 are based on XML, and RSS 1.0 on RDF. The RSS 1.0 specification can be found at <http://www.purl.org/rss/1.0/>.

## iCalendar

The iCalendar format aims at providing interoperable calendaring services. Calendaring tools such as Apple's iCal or Mozilla Calendar use the iCal format to describes calendar entries. There exist converters from/to iCalendar to automatically import/export calendar entries from personal information manager applications. The iCal ontology is not yet as widely used as FOAF and RSS, but has potential to become widely adopted. RfC 2445 describes the iCalendar format, and the iCal RDF specification is online at <http://www.w3.org/2000/10/swap/pim/ical.rdf>.

In addition to the close-knit homogeneous group of RDF instance data in FOAF and RSS vocabularies, there are small amounts of instance data available in other vocabularies as well, for example the iCalendar format in RDF. Some database-driven sites even automatically export their user data in dynamically created files. Few sites go even further and provide access to their RDF repositories using query languages such as RDQL or SERQL. These repositories contain data such as telephone lists of a department or descriptions about project partners in proprietary vocabularies. The advantage of the repositories is that one can issue queries against them instead of having to access flat files.

A lot of metadata is emerging online in different sizes and shapes: a community-build web of data in FOAF and RSS, small amounts of RDF instance data in proprietary vocabularies that are only linked from within HTML pages, and query-able RDF repositories in proprietary vocabularies typically created by a single organization. The

Semantic Web of today has means to describe data, but lacks the facilities to combine data from different sources. So far, the RDF data on the Web has been utilized to the extent possible, mostly because it is difficult to locate and combine the files and query the resulting information since the RDF files are dispersed across several thousand web sites. Similar to documents on the HTML web, the data on the Semantic Web is of varying quality in terms of syntactic correctness, accuracy, and detail. To take full advantage of the information in all sources it is necessary to access and integrate the data. All these data sources contain a vast amount of potentially useful information, trapped in isolated ontology islands that cannot be queried as a whole, which is a serious shortcoming of the current Semantic Web infrastructure.

To illustrate how to make use of the type of information available online consider the following example: In a project, a meeting is supposed to be held in a few weeks. As a project participant, you know that person A will participate in the meeting. You can look up detailed information (contact details, his social network) about that person that is encoded in FOAF. In the person's weblog (made available in RSS) he posted information about a document or a presentation that will be subject of discussion at the meeting. You find the ideas in the document interesting and plan to further discuss the document with that person. Finally, by seeing the arrival and departure dates made available through his calendar (in iCalendar format) to which the project participants have access to, you have the ability to send an email to the person to schedule a personal meeting. In this paper, we describe the infrastructure that enables agents to access data that is potentially scattered across the web in a uniform way.

SECO, the application described in this paper, utilizes RDF data by combining techniques from information retrieval and semistructured databases by applying Semantic Web technologies. By performing range of operations on the available RDF data, both human users and applications can have an integrated view onto the Semantic Web. SECO includes components that aggregate RDF files from the Web, integration mediators that can be queried by software agents, and a user interface component that constructs HTML from the integrated data view provided by the integration mediators.

## **2 SECO Components**

The components described in the following can be seen as mediators [[Wiederhold 1992](#)]. We combine warehousing the data with providing virtual integration once all data sources are queryable, because needed data is fetched from the repositories and integrated on demand. The scutter component can be seen as an aggregation mediator that gathers RDF files from the Web, aggregates them, and enables software agents to query the RDF data set using a remote query interface. Part of the system are other mediator services that consolidate instances and perform schema mapping based on a inference engine. The user interface is used to browse the integrated data. You find an overview of the architecture in Figure 1.

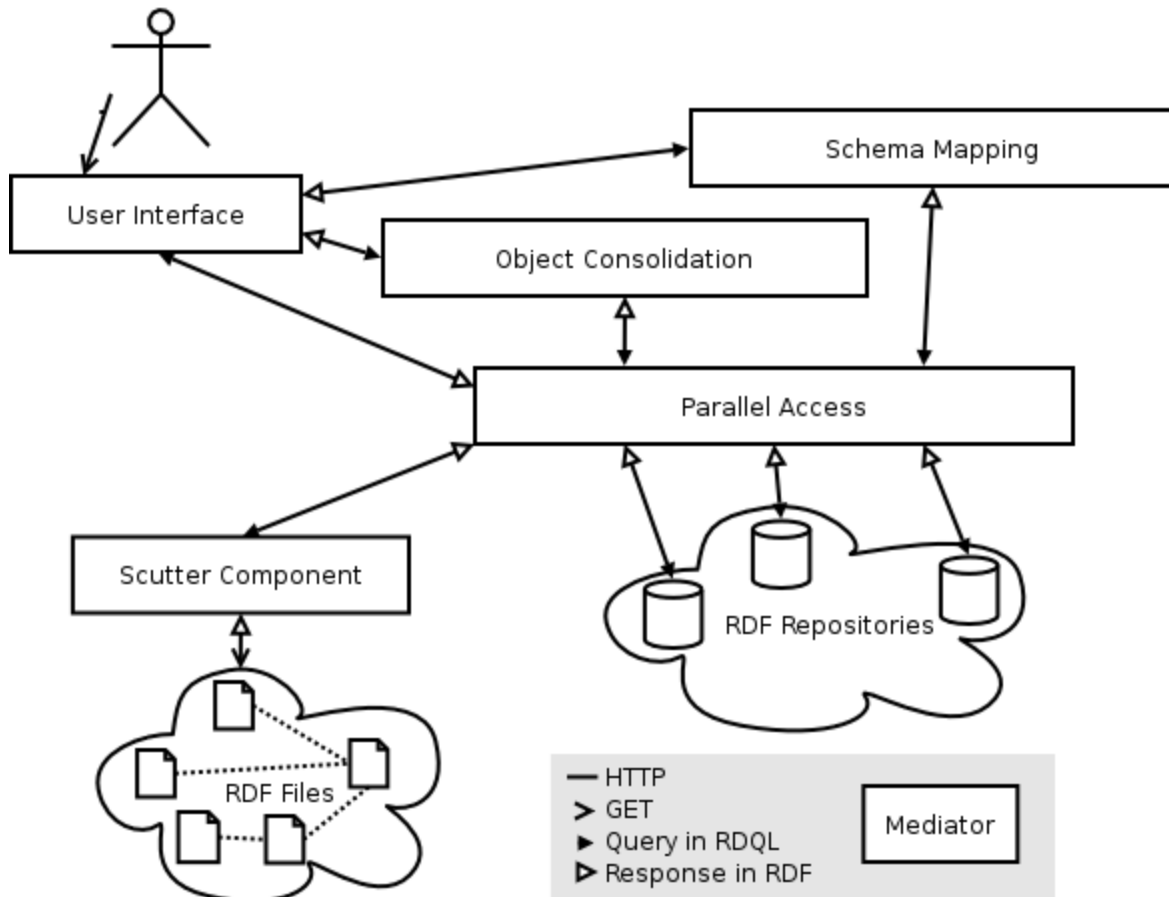


Figure 1. Architecture of SECO

SECO applies techniques known from HTML search engine technology to solve the document-centric part of the problem, and techniques from database integration research to map schemas and provide query facilities over the diverse set of data sources. Other than in integration projects targeted at one organization, data integration on the web has to take into account variations of data quality. Dealing with the dynamic and distributed nature of an open environment such as the web in the context of database integration is challenging.

Components are self-contained web applications that expose their functionality via RDQL over HTTP, except the user interface which emits HTML. Two components have internal data repositories: the scutter for storing collected files together with associated metadata, and the user interface for storing site usage data. The integration mediators only parse and interpret a query in a subset of RDQL and pass the transformed query along to other components. You can find an example query that selects name and e-mail address properties in Figure 2. The result of RDQL queries that is transmitted is the subgraph needed to answer the query [Seaborne 2002]. The reason for returning the subgraph as opposed to variable/value pairs is to be able to process the result of the query further using RDF tools.

```
SELECT ?name, ?mbox
WHERE (?uri, <foaf:name>, ?name),
      (?uri, <foaf:mbox>, ?mbox)
USING foaf FOR <http://xmlns.com/foaf/0.1/>
```

Figure 2. RDQL query that selects *foaf:name* and *foaf:mbox* (email address) properties

## 2.1 Scutter Component

At first, the FOAF and RSS files have to be discovered and collected, and made available for query. Because the structure of FOAF and RSS files are very similar to the document-centric HTML web, we employ a crawling component to harvest RDF files from the web. The design of the scutter is very similar to HTML crawlers [[Cho and Garcia-Molina 2002](#)] and uses similar mechanisms for crawling the web and resolving cycles. Crawling is currently carried out simply in a breadth-first manner without optimizations, but mechanisms used in HTML crawlers can be used to perform more sophisticated crawling.

The *rdfs:seeAlso* links connect the FOAF files scattered across the web into a web of data. In contrast to most RDF files in other vocabularies which are only linked from within HTML pages via a *href* hyperlink, using the *rdfs:seeAlso* property enables software agents to harvest the distributed FOAF files and put them into a repository to allow for querying. Connecting RDF files among each others facilitates writing RDF crawlers without having to parse and extract information out of HTML. We counted more than 5000 RDF files on the Semantic Web that are linked using *rdfs:seeAlso*. Figure 3 shows the graph of RDF files, connected via *rdfs:seeAlso*, that we were able to harvest from the web. The center of the clusters are mainly scutterplan files, files that only contain *rdfs:seeAlso* links to jump-start the web of data.

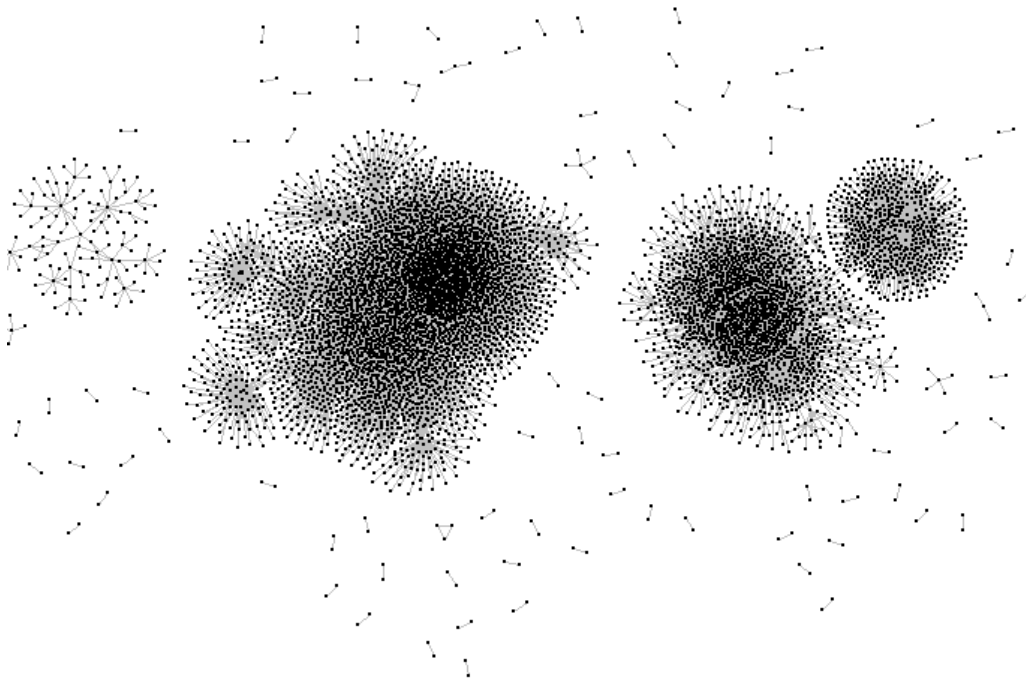


Figure 3. The web of data: dots denote RDF files, connected via *rdfs:seeAlso* links

Data associated with the operation of the scutter (e.g. when a file was last visited, robots.txt, status information, etc) is stored in RDF. RDF files — regardless of what vocabulary used — that have been transmitted successfully and syntactically correct are stored. To keep track of provenance of RDF files and thus to be able to perform updates in case a file has changed, statements are stored using reification. The scutter accepts content type application/rdf+xml, although a lot of RDF files are just being served as text/plain or text/html types. Files that contain `syn:updateFrequency` and `syn:updatePeriod` properties as defined in the RSS specification are re-crawled accordingly. For example, some RSS newsfeeds are updated four times an hour, and have to be visited as often by the scutter, whereas FOAF files seldom change and are therefore only visited less often. The scutter understands the HTTP 304 "not modified" header to prevent re-fetching of files that have not changed since the last visit, and honors the robots exclusion protocol. Large database-generated site or even RDF repositories that export FOAF in flat files are excluded manually to prevent the scutter from replicating huge repositories.

## 2.2 Parallel Access Mediator

Next, a mediator provides parallel access to all available data repositories both the scutter and RDF data repositories on the Internet can be queried. The integration mediators can access all the underlying data sources via a single access point. The parallel access mediator stores descriptions of the various repositories and their contents very broadly (i.e. what properties are stored there), which makes it possible to locate the RDF statements needed to answer a query. The query is split into triples, and then the metadata for the data repositories is used to decide what repositories to query. For example, for there might be a repository that only contains FOAF data, therefore we don't need to pose a query about RSS against the repository. The queries are then posted to the various data repositories. Results are combined in a first step based on URI, or on bNode references in case the statements originate from the same repository.

## 2.3 Integration Mediators

Then, integration mediators can be used to process the data further. The integration mediator expose three operations that provide the essential data needed for constructing the user interface. The integration mediators have to consolidate objects, map queries from the mediated to the native schema, and map the resulting RDF accordingly. The two integration mediators can be combined or chained if required. In the current architecture, however, only either one or the other is used in a query.

### Object Consolidation

Different places can make statements about the same thing. In case the same subject URI is used, combining those statements is straightforward and is carried out in the parallel access mediator. For FOAF, a unique identifier for persons can be their email addresses or chat ids. The object consolidation is currently carried out on those properties, but the

constraints can be relaxed so that consolidation is also carried out based on other things such as `foaf:name`, which can increase recall but lower precision.

## Schema Mapping and Alignment

For tasks such as constructing the user interface we need to perform schema mapping. For now, we only want to be able to display the gathered information, so we define a very simple mapped schema that consists of four properties: title, description, date, and rank. Table 1 shows the mappings from the mediated ontology to native terms. Queries that are posed to the mapping mediator are split into subqueries, and the subqueries are then translated to the native schema. The resulting RDF returned from the parallel access mediator is mapped from the native schema to the mediated schema. All four properties have to be present to construct the user interface, so default values are assumed in case a property is missing.

Mediated term	Native term
title	dc:title, rss:title, foaf:name, ical:summary
desc	dc:description, rss:description, foaf:mbox, ical:description
date	dc:date, ical:value
rank	seco:rank

*Table 1. Description of mappings*

The schema mapping in SECO utilizes the global-as-view (GAV) approach as described in [Levy 1999] since the interactions among the sources are small. An RDF file provides the mappings from different vocabularies into a mediated schema. Only properties are mapped in our example, but mapping of classes follows a similar scheme. The user interface uses data in the mapped schema for generating a HTML page providing a list view.

First, split the query posed in native terms into S,P,O subqueries. Then, translate the SPO subquery in the mediated ontology that only consists of four properties (title, description, data, rank) into subqueries in native terms. For example, the subquery for `title` results in four subqueries in native vocabulary: `dc:title`, `rss:title`, `foaf:name`, and `ical:summary`. Pose the subqueries to the parallel access mediator, which has knowledge about the location of triples to discard senseless queries.

Once the parallel access mediator completed, the results in native vocabulary have to be mapped to the mediated schema. The schema mapping is carried out using TRIPLE [Sintek and Decker 2002]. The rules in Figure 4 assume default values for missing properties and perform the mapping of the results from the native schema into the mediated schema. Variations in the instance data such as missing properties are dealt with by assuming default values for `date` and `rank`. The results are still in native terms, therefore we need to map accordingly. The query in the mediated schema is now posed against the mediated model (`@mediated` in Figure 4), and we end up with the answer to

the original query in the mediated schema. The resulting RDF model is returned to the original caller, the user interface.

```
// copy statements from native model to mediated model and apply mappings
FORALL A, B, X, Y A[X->B]@mediated <- X[mapsTo->Y] AND A[Y->B]@native.

// add mandatory rank and date if not already there
FORALL X,Y,Z X[rank->1]@mediated <- X[title->Z]@mediated AND NOT
X[rank->Y]@mediated.
FORALL X,Y,Z X[date->"2000-01-01T00:00:00+00:00"]@mediated <- X[title->Z]@mediated AND NOT X[date->Y]@mediated.
```

*Figure 4. Rules for applying default values and mappings.*

## 2.4 User Interface

Finally, there have to be means for human users to navigate the integrated data. The key idea for the user interface is to get the portion of the whole graph from the mediators that is important for the generation of the final page, extract a tree structure with the needed information via query, and use the tree to generate a HTML page to display the mapped and aligned portion of the original graph. We declaratively specify the structure of the pages of the site using a query, and transform the results of the query to HTML. The user interface supports three fundamental operations: a list view, keyword search functionality, and a page containing all available statements of an instance.

The list presentation, similar to how items in weblogs or search engines are presented, is used for displaying news items and results of keyword searches. The pages for the list view are constructed by posing a query in the mediated schema. In case of keyword search, the query for keywords is restricted to `title` and `desc`. The items in the list view can be sorted according to rank or date. Figure 5 shows a screenshot of the front page displaying news items originating from the UK news site "The Register", news for nerds site Slashdot, and German national TV news show Tagesschau. From the list view, one is able to select a certain instance and display the statements associated with an instance — including links to other instances — on the details page in native terms. The details page is constructed by posing a query to the object consolidation mediator.

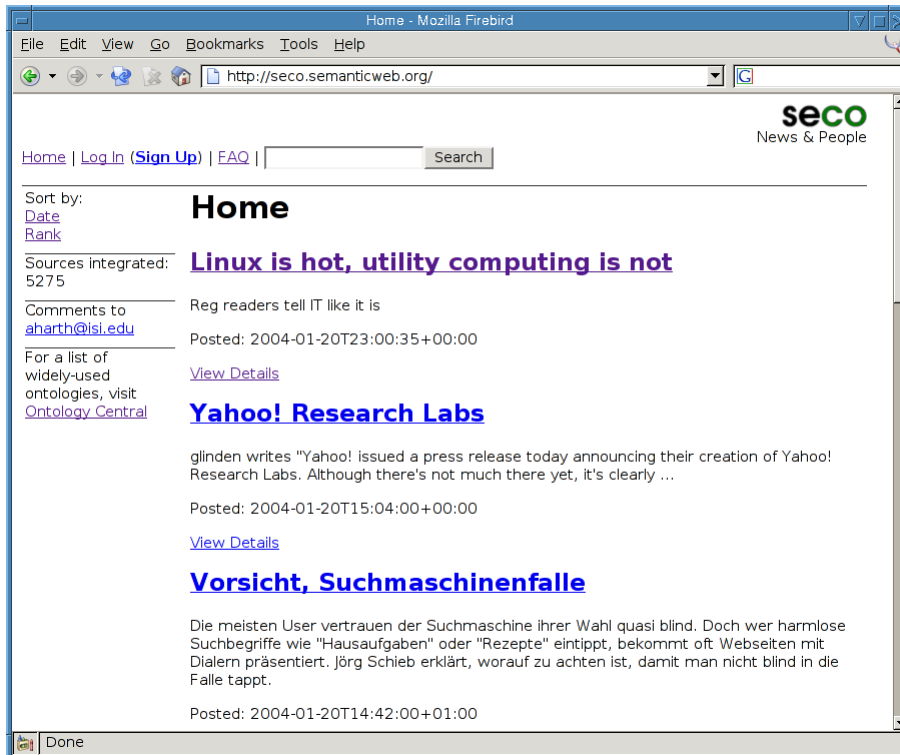


Figure 5. Screenshot of the HTML user interface in list mode, sorted by date

The user interface takes the data returned by the integration mediators to construct a user interface in HTML using a three stages (Figure 6). First, an RDQL query is posed to the model returned by the integration mediator to extract the required data. Second, extract the relevant pieces from the returned graph and format the result as XML. Third, the XML document is transformed into a presentation-oriented HTML page using an XSLT stylesheet. Thus the process of creating HTML from RDF as depicted in Figure 6 comprises three stages: from content over structure to presentation. Having the three-step approach simplifies the generation of the user interface significantly, because only a query and a set of XSLT stylesheets are needed to generate a new set of pages.

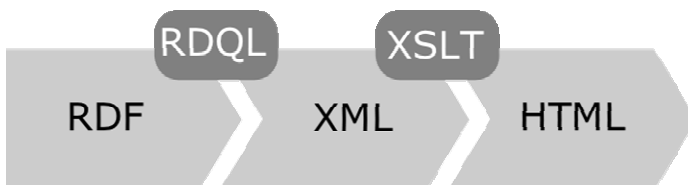


Figure 6. Creating HTML pages from RDF using XML as intermediate representation

### 3 Related Work

The user interface generation works using a similar principle as employed in the Strudel site management system [Fernandez et al. 1998]. Strudel used the separation of content, structure, and presentation using proprietary technologies for processing a centrally created data set. SECO uses off-the-shelf components and leverages standards such as

RDF, RDQL, and XSLT; more importantly, we apply the mechanism to data collected and integrated from the Semantic Web.

We borrow techniques from information retrieval as well as database research: from an IR perspective, we warehouse documents similar to how a search engine collects files and builds an inverted index. From database research, we utilize the "virtual integration" approach that has better scalability properties than warehousing all data, but requires the data sources to exhibit some "intelligence", meaning that the data sources must have a query interface.

The Ontobroker initiative [[Decker et al. 1999](#)] showed the principal feasibility of the idea to collect and integrate data from heterogeneous data sources in an intelligent way. The main difference to Ontobroker is SECO's distributed architecture. Furthermore, SECO is a working system that integrates data from the web which has been created by a large number of people.

The mediators as described in this paper can be seen as web services. SECO doesn't seek to automatically compose the different services, but has a fixed scheme of how the various mediators interact. Also, SECO is less ambitious than the Edutella project [[Nejdl et al. 2002](#)] which is based on a peer-to-peer network that requires special p2p software to be installed for nodes to be part of an overlay p2p network. We just use the web infrastructure and the mediator notion, however, in a sense the different mediators and data repositories could be viewed as peers. A complete peer-to-peer approach for SECO would have to add capabilities to deal with services coming and going, which is needed to include peers that use dial-up Internet connections.

There are existing aggregators for news and FOAF data available online. News aggregators such as Google News scrape the news items out of HTML pages but are not using RSS feeds, RSS aggregators such as AmphetaDesk are client applications, and FOAF aggregators such as <http://plink.org/> only provide support for FOAF. All aggregators to date are specialized on one vocabulary and lack a query interface for software agents to access the integrated data, which means everybody who wants to perform queries over the Semantic Web has to employ own scutters and storage infrastructure.

## 4 Conclusion

This paper presented a set of mediators that collect, integrate, and display RDF data from the web. Making use of the RDF data either in flat files or in data repositories that constitute today's Semantic Web using SECO involves a series of steps. At first, the scutter component discovers and collects RDF files, and makes the resulting data set available for querying. Next, a mediator provides parallel query access to the scutter component's and other data repositories' data. Then, integration mediators can perform object consolidation or schema mapping and alignment. Finally, a user interface allows users to browse the integrated data set in the mediated schema, perform keyword searches, and view instances in the native schemas.

SECO has applied the mediator architecture known in databases to data from the open web environment. Agents can now use the mediators to query a large chunk of the current Semantic Web without the need to crawl and warehouse the data themselves. Agents can search about information for people, and have information about the same thing integrated and combined from different sources in varying granularity depending on which mediator is used.

SECO tracks provenance information about RDF statements to be able to update files that have changed. Different vocabularies have different update semantics: where RSS usually provides a stream of news items that should be accumulated over time, changes in FOAF files mean that the previous version should be replaced by the current. Because vocabularies can be mixed in the same file, determining what update semantics to apply for a certain file is difficult. There exist no standardized mechanisms for updating RDF files or RDF repositories.

Scalability will become increasingly a concern as the number of available data sources and ontologies grows. Scuttering can be parallelized, and a mediator already provides parallel access to a large number of data sources. The integration mediators perform better if less data gets fetched from the data repositories. Thus, joins and constraints in queries should be propagated to the original RDF repositories to minimize the amount of RDF transmitted. Having knowledge about contents and capabilities of RDF repositories is mandatory to perform optimization on queries and combine mediators automatically.

## **Acknowledgements**

Thanks to Jose-Luis Ambite and Stefan Decker for their helpful comments.

## Bibliography

[Wiederhold 1992] Gio Wiederhold. "Mediators in the Architecture of Future Information Systems". IEEE Computer, March 1992, pages 38-49.

[Decker et al. 1999] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999.

[Miller et al. 2002] Libby Miller, Andy Seaborne and Alberto Reggiori. "Three Implementations of SquishQL, a Simple RDF Query Language". In Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, June 2002, Lecture Notes in Computer Science (LNCS 2342), pages 423-435.

[Nejdl et al. 2002] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmer, Tore Risch: EDUTELLA: A P2P Networking Infrastructure Based on RDF. In Proceedings of the 11th International World Wide Web Conference 2002 (WWW2002), Hawaii, 7-11 May 2002.

[Fernandez et al. 1998] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy and Dan Suciu. "Catching the Boat with Strudel: Experiences with a Web-Site Management System". In Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA, pages 414-425.

[McBride 2000] Brian McBride. "Jena: Implementing the RDF model and syntax specification". Technical Report, HP Labs at Bristol, UK, 2000. [www.hpl.hp.com/people/bwm/papers/20001221-paper/](http://www.hpl.hp.com/people/bwm/papers/20001221-paper/)

[Seaborne 2002] Andy Seaborne. "An RDF Net API". HP Technical Report, 2002. <http://www.hpl.hp.com/techreports/2002/HPL-2002-109.html>

[Levy 1999] Alon Y. Levy. "Logic-Based Techniques in Data Integration". Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14-16, 1999.

[Cho and Garcia-Molina 2002]. Junghoo Cho and Hector Garcia-Molina. "Parallel Crawlers". In Proceedings of the 11th International World Wide Web Conference 2002 (WWW2002), Hawaii, 7-11 May 2002.

[Sintek and Decker 2002]. Michael Sintek and Stefan Decker. "TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web". In Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, June 2002, Lecture Notes in Computer Science (LNCS 2342), 2002.