

# Enterprise Linked Data as Core Business Infrastructure

Steve Harris and Tom Ilube and Mischa Tuffield

**Abstract** This chapter describes Garlik’s motivation, interest, and experiences of using Linked Data technologies in its online services. It describes the methodologies and approaches that were taken in order to deploy online services to hundreds of thousands of users, and describes the trade-offs inherent in our choice of these technologies for our production systems. In order to help illustrate and aid the arguments for the adoption of Semantic Web technologies this chapter will focus on two of our customer facing products, DataPatrol, a consumer-centric personal information protection product, and QDOS a Linked Data service that is used to measure peoples’ online activity.

## 1 Introduction

This chapter presents how and why the online identity and privacy startup Garlik<sup>1</sup> built its own Semantic Web infrastructure to power its core business. The chapter will aim to present insight into how Semantic Web technologies are utilised internally, highlighting the advantages gained over other more traditional technology stacks and whilst providing motivating factors to our early adoption. The chapter will also describe our two primary technological developments in terms of storage solutions for data represented in the Resource Description Framework (RDF)<sup>2</sup> [6], namely

---

Steve Harris

Garlik Ltd, 1-3 Halford Road, London, TW10 6AW e-mail: [steve.harris@garlik.com](mailto:steve.harris@garlik.com)

Tom Ilube

Garlik Ltd, 1-3 Halford Road, London, TW10 6AW e-mail: [tom.ilube@garlik.com](mailto:tom.ilube@garlik.com)

Mischa Tuffield

Garlik Ltd, 1-3 Halford Road, London TW10 6AW e-mail: [mischa.tuffield@garlik.com](mailto:mischa.tuffield@garlik.com)

<sup>1</sup> Garlik Ltd, Online Identity Experts <http://www.garlik.com/>

<sup>2</sup> Resource Description Framework <http://www.w3.org/RDF/>

the open-source 4store<sup>3</sup> (see section 5.1) and its proprietary successor 5store<sup>4</sup> (see section 5.2).

In order to help illustrate and aid the arguments for the adoption of Semantic Web technologies this chapter will focus on two of our customer facing products, DataPatrol<sup>5</sup>, a consumer-centric personal information protection product, and QDOS<sup>6</sup> a Linked Data [1] service that is used to measure peoples' online activity. These two systems are presented in section 3.

DataPatrol is a system which scans various sources of information, databases, online resources, open web data, and data held in closed communication systems such as IRC traffic. These sources of information are scanned for instances of customer data which is either appearing in places where it should not, such as an IRC channel known to be used for the sale of personal information, or in combinations which expose the user to a risk of financial fraud.

As the nature and methods used in personal information trading change frequently it is necessary for any system attempting to solve this problem to be easily adaptable to changing environments, and new sources of information.

The rest of this chapter will be broken down as follows. Following this introduction Garlik's core motivating factors for the adoption of Semantic Web technologies are listed in section 2. Section 3 details two of Garlik's main services, these are introduced to as real-world applications of Semantic Web technologies. This is followed by an example of Garlik makes use of Semantic Web Technology to deploy schema driven Software Development that allows for new functionality to be deployed whilst avoiding service down time (see section 4). The motivating factors coupled with the characteristics of the two services described are used to illustrate our arguments for the adoption of Semantic Web technologies, and are revisited in the conclusions and future work section of this paper (see section 6).

## 2 Motivations

Garlik's use and development of Semantic Web technologies have been driven by a number of core motivational factors that are summarised by the following points. These are used to contextualise the decisions made during the development of Garlik's core technologies, and will be revisited throughout the rest of the chapter.

- **Flexibility** Given that Garlik's core business revolves around the capture of personal information, whether it be specific data sources provided by partners or information collected from the World Wide Web, the ability to adapt and index new information in an agile manner is key. DataPatrol (section 3.1) needs to be

---

<sup>3</sup> 4store scalable RDF storage <http://4store.org/>

<sup>4</sup> 5store scalable RDF storage <http://4store.org/trac/wiki/5store>

<sup>5</sup> DataPatrol <http://www.garlik.com/products.php>

<sup>6</sup> QDOS <http://qdos.com/>

able to index new sources of data as and when they become available. The ability to react to the existence of a new source of compromised data in order to effectively protect our customers' personal information from being abused by criminals is presented as one of the key motivating factors to our development and deployment of Semantic Web technologies. The flexibility of RDF to allow for arbitrary knowledge to be encoded in a triple format is presented as the key reason for why Semantic Web technologies are presented as core to DataPatrol's success.

- **Scalability** In order to meet the demands of a high-volume consumer facing service both DataPatrol and QDOS are required to be able to index and query large amounts of RDF data in real time. As a result of this requirement Garlik was motivated to produce its own RDF storage technology to support the flexibility promised by the RDF data format. The ability to index and run SPARQL<sup>7</sup> queries across high volumes of RDF data in real-time have led to the development of Garlik's two clustered RDF database solutions (these are presented in section 3).
- **Data Centric: Ontology/Schema Driven** The core services provided by Garlik, are all data-centric, and as discussed in the first point the ability to be flexible to search and integrate new forms of information has been fundamental to our success. Garlik makes use the expressive power of RDF by implementing schema driven software solutions that allow for functionality to be added and removed by virtue of adding and removing triples from an RDF schema. An example of such an interaction is presented in section 4, this allows for new functionality to be added to a system, often without the need to redeploy any pre-existing systems. This ability to add new services as desired is said to be core to Garlik's desire to be able to react quickly to the need to index new information.
- **Use and Promotion of Standards** This point relates to all of the previous ones. The adoption and promotion of Web Standards have always been high on Garlik's agenda, this is evident by the decision to release and support 4store under the GPLv3 open-source license<sup>8</sup>. The promises presented by the energetic Linked Data community, that can be seen in ever growing Linked Data Cloud<sup>9</sup> has driven Garlik's desire to produce scalable RDF storage solutions. 4store was released into the community to help developers consume the vast amounts of RDF being produced by the Linked Data community. Furthermore, our needs regarding the ability to update RDF via the SPARQL language has motivated Garlik to be at the forefront of the design of SPARQL 1.1<sup>10</sup>, the next iteration of the query language, at the W3C. The decision to release 4store as an open-source project is seen as an example of how Garlik is promoting and furthering the state of the art. The ability to easily setup development environments for coding purposes is another good reason for deciding to work with an many standards as possible. As it stands Garlik's developers can choose to prototype systems on any operating system

---

<sup>7</sup> SPARQL Query Language for RDF <http://www.w3.org/TR/rdf-sparql-query/>

<sup>8</sup> GNU General Public License v3 <http://www.gnu.org/licenses/gpl-3.0.html>

<sup>9</sup> Linked Data Cloud <http://richard.cyganiak.de/2007/10/lod/>

<sup>10</sup> SPARQL Working Group [http://www.w3.org/2009/sparql/wiki/Main\\_Page](http://www.w3.org/2009/sparql/wiki/Main_Page)

they wish and are not limited to work on UNIX-like environments, though all production systems run the CentOS<sup>11</sup> Linux distribution. Given our heavy use of standards, such as HTTP<sup>12</sup>, and the SPARQL Protocol<sup>13</sup>, developers are free to build software and prototype using any RDF store that implements the SPARQL Protocol. These include but are not limited to: Jena<sup>14</sup>, or Sesame<sup>15</sup>.

### 3 Garlik's System Architectures

In this section we present two of Garlik's live services, DataPatrol (section 3.1), and QDOS (section 3.2). DataPatrol is used to help describe some of the software design decisions employed within Garlik. QDOS will be used to illustrate publicly accessible examples of some of the techniques used within DataPatrol. From a business perspective the information stored within DataPatrol's and QDOS's respective knowledge-bases (KBs) are treated extremely differently, to begin with they are on separate physical networks, one stores private highly-sensitive data, and the other stores facts pertaining to users, obtained from the public Web. One thing both systems share is that they make use of Semantic Web technologies: DataPatrol is presented as a Semantically Driven Web Application, and QDOS a Public Linked Data Service. As described below the ability to flexibly add and remove data, whilst maintaining the ability to query the data stores in a live environment has been central to all of our design decisions.

DataPatrol's requirement to be able to add new data sets easily, coping with constantly changing data, whilst maintaining provenance information used to gather business-critical Management Information (MI) was one of the reasons why the Resource Description Framework (RDF)<sup>16</sup> was selected as the data format of choice. The flexibility that RDF provides when used to represent data, namely that of being able to express arbitrary graphs through the adoption and reuse of unique identifiers, the encoding of knowledge in the form of triples, allows Garlik the ability to add new data sources in an ad-hoc basis, without the need to constantly make changes to one's database schema. This is due to the fact that RDF can be used as a schemaless data representation format.

Another core driver to the development of services within Garlik is the exploitation and adoption of open web standards, as proposed by the World Wide Web (W3C) consortium<sup>17</sup>. The concept of developing software as services has matured over the past few years, that coupled with the power of RDF as a knowledge repre-

---

<sup>11</sup> Community ENTERprise Operating System <http://centos.org/>

<sup>12</sup> Hypertext Transfer Protocol <http://www.w3.org/Protocols/>

<sup>13</sup> SPARQL Protocol for RDF <http://www.w3.org/TR/rdf-sparql-protocol/>

<sup>14</sup> Jena - A Semantic Web Framework for Java <http://jena.sourceforge.net/>

<sup>15</sup> Sesame - An open-source framework for storing and querying RDF data <http://www.openrdf.org/>

<sup>16</sup> RDF <http://www.w3.org/RDF/>

<sup>17</sup> The W3C <http://w3c.org/>

sensation language helped Garlik decide that it would adopt and actively participated in the development of open-sourced tools and open standard to help the realisation of the Semantic Web. RDF is not the only piece of the Semantic Web puzzle that Garlik employs, SPARQL [7], SPARQL-Protocol, HTTP, RESTful APIs [3], and ontologies are but a few of them.

From the many adopted serialisations of RDF, i.e. RDF/XML, n-triples, turtle, Garlik tends to make use of the “turtle”<sup>18</sup> serialisation. Turtle is a human-friendly serialisation and makes debugging our software much easier. At this point it is important to note that both 4store and 5store do not perform any automatic inferencing whatsoever. Unlike their predecessor 3store<sup>19</sup> [4] which implemented RDFS reasoning upon the RDF data it stored, a decision was made to simply store RDF triples as imported. This decision is based on the desire to be as efficient, and as scalable as possible. Furthermore deleting triples from an RDF graph that supports reasoning is a computationally expensive task, and did not fit in with our motivation of being to index, add, and remove data in an agile manner. This does not mean that Garlik does not apply reasoning methodologies to the RDF it stores, an example of how reasoning is performed via SPARQL is presented in section 3.2.1.

Finally, both QDOS and DataPatrol are presented as Linked Data Applications, insofar as they make use of HTTP resolvable URIs<sup>20</sup>, and run off of SPARQL stores – as apposed to being mere transforms on top of relational data. All of the publicly available user pages on QDOS allow for content negotiation to be performed in order to get back RDF describing a given person. Such requests make SPARQL CONSTRUCT<sup>21</sup> queries to the 4store instance and the result is returned to the issuer. The key difference between the two systems is that QDOS is an Linked Open Data resource, which publishes data to the Web whereas, DataPatrol’s Linked Data resources are used for internal Customer Service, provenance, and debugging purposes and it not exposed to the Web.

### 3.1 DataPatrol

DataPatrol is focused on delivering real-time information to its customers protecting them from identity theft by harvesting and searching through both publicly available and privately acquired data about individuals. There have been a number of fundamental business driven requirements that have dictated the design and development of our systems and our Semantic Web platforms, DataPatrol influenced all of them. Key to our consumer focused DataPatrol service is the requirement of a highly available software platform, where both flexibility to index new data upon discovery and service uptime were both central. DataPatrol runs on low-cost networked servers

---

<sup>18</sup> Turtle - Terse RDF Triple Language <http://www.dajobe.org/2004/01/turtle/>

<sup>19</sup> 3store RDF storage <http://threestore.sourceforge.net/>

<sup>20</sup> Uniform Resource Identifiers <http://www.ietf.org/rfc/rfc2396.txt>

<sup>21</sup> SPARQL’s CONSTRUCT verb <http://www.w3.org/TR/rdf-sparql-query/#construct>

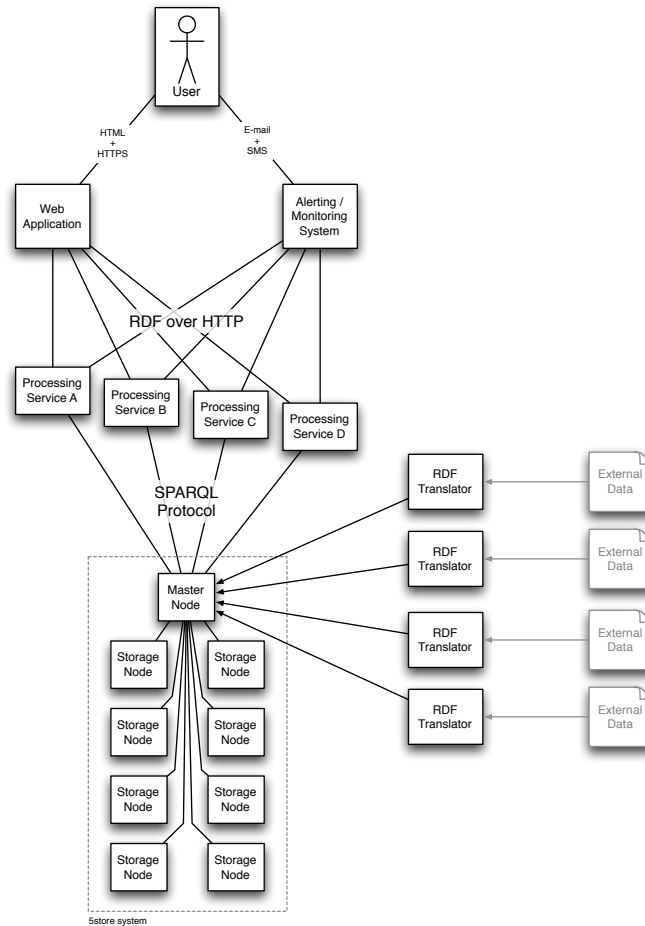
and has to support around the clock, 24x7, operation for hundreds of thousands of customers. It should be noted that DataPatrol is currently live and operational in the United Kingdom, Germany, and the United States of America – and is constantly expanding into new geographical regions.

DataPatrol is a Web based consumer focused service powered by Garlik’s 5store (see section 5.2), and is built using the Java programming language. Figure 1 presents an overview of DataPatrol’s software architecture, which in short is made up of a number of web services interacting via HTTP and that use RDF as a data exchange language. Firstly, it is important to stress that end users never come across RDF, or any mentions of Semantic Web Technology in their day-to-day interactions with DataPatrol. The end users need not know about the underlying technology, and from their point of view DataPatrol’s user interface is like any other Web Application.

DataPatrol in short indexes information about people, whether it comes from UK registries or whether it is compromised personal information collected by Garlik, and then acts as a notification system for its users. Users’ are alerted when DataPatrol deems that there is too much information about them available in the public and digital domain and suggests a course of action which should be taken by the user in order to safe guard there identity.

DataPatrol integrates information from a number of different sources, these are presented in the figure 1 as “RDF Translators”. The UK versions of DataPatrol indexes the following non-exhaustive list of data sources: The Companies House Register, The Births, Deaths, and Marriages Register, Euro Direct Marketing Database, The Royal Mail Redirects Database, as well as harvesting personal information acquired by accessing compromised data distributed via botnet networks, as well as data found on the World Wide Web. The aforementioned sources of information are converted into RDF, and stored within a 5store cluster which is subsequently queried via SPARQL over HTTP. In the architectural diagram (figure 1) these services that issue SPARQL queries over the RDF indexed by DataPatrol are illustrated as “Processing Services”. These “Processing Services” are nothing more than simple web services, that when called via a HTTP GET request, issue SPARQL queries to 5store and if they come back with a hit a small RDF document is returned. This RDF document identifies the nature of the information found, and includes provenance information regarding the source of the data and the time in which the data was indexed by DataPatrol. This notion of outputting results in RDF, is illustrated by the label “RDF over HTTP” in DataPatrol’s architectural diagram.

This use of “RDF over HTTP” (Turtle being the specific serialisation of RDF adopted by Garlik), and the “SPARQL Protocol” allows for flexibility in terms of the rapid development of new “Processing Services” given the identification of new sources of information deemed critical to protecting individuals from identity theft, and other online fraud. The two parts of the DataPatrol system that interact with users are the “Web Interface” and the “Alert/Monitoring Systems”, these components receive RDF from the various “Processing services” or *searchers* and subsequently inform the user about how their personal information may have been compromised. Garlik has written its own lightweight RDF Parser which parses “turtle”



**Fig. 1** DataPatrol Architecture

and then makes a decision to alert the customer if the customer is deemed to not have been informed of the alert before; this parser is incorporated into all of the components which may interact with a user. This model of bringing together the various searchers in a service orientated architecture allows for the easy setup and addition of new services. As long as the services can be called via the HTTP protocol, and as long as they return their results in RDF, against the DataPatrol alert schema they can be easily integrated.

The web-service based nature of the DataPatrol makes it easy to debug, allowing for developers to make various HTTP calls to the different services available, in case there is ever an issue. This means that all of the backend services can be invoked individually via HTTP GET requests and they do not require complicated test scripts that make use of Object Models or other database abstraction techniques.

Another advantage of deploying the various backend components as HTTP services makes it easy to add redundancy to the live system. Given that the various processing services are merely web services, if a machine were to crash due a hardware fault or similar the requests will fail over to a duplicate service, and it is but a mere configuration change of adding the location of an additional HTTP service to regain full strength.

There are a numerous other advantages to developing software as HTTP services, such as the fact that RESTful APIs are well established in the developer community and a lot easier for to work with than technologies like SOAP<sup>22</sup>. Mature open-source Web hosting tools, such as Apache<sup>23</sup> and Tomcat<sup>24</sup> incorporate a number of existing features load balancing, caching, proxies which all make managing a busy system with multiple users more straight forward. HTTP services also help get around the issue of connection pooling, as each HTTP based interface ensures that connections to backend services are short lived.

The data which DataPatrol searches over comes from a variety of different sources, some of which Garlik has contractual obligations regarding the manner in which that the data is used. For example, DataPatrol indexes certain pieces of information that has to be discarded after a fixed period of time, this is supported by the inclusion of provenance information relating to the origin and the time in which new data is added to DataPatrol's 5store infrastructure. This provenance information is attached to the Named Graph [2] of every RDF document imported into DataPatrol, and as a result all of the RDF imported is held as quads within 5store. This use of Named Graphs along with the SPARQL query language allows us to track the provenance of all of the data held within it, this is advantageous given that Garlik has strict reporting requirements from our various business partners and data providers. Some of our partners require that data only be accessed within a given duration of time, sometimes DataPatrol needs to count how many times results are found, and sometimes Garlik has to pay per time the data is queried. The ability to be agile when it comes to generating MI is made possible due to RDF's ability to attach triples providing provenance to Named Graphs. Given the numerous third-party data sources integrated within DataPatrol, the fact that 5store indexes all data imported to it, in the form of quads this caters for the simple generation of MI information, whereas certain queries would not be feasible if the data was held in a Relational Database Management System (RDBMS). Traditionally system run off of RDBMSs tend to have to require a separate data warehouse system in which data is imported into, 5store's quad based index relieves DataPatrol of such a subsystem.

---

<sup>22</sup> SOAP Specification <http://www.w3.org/TR/soap/>

<sup>23</sup> Apache HTTP Server <http://httpd.apache.org/>

<sup>24</sup> Apache Tomcat Java Servlet Server <http://tomcat.apache.org/>

### 3.2 QDOS

We all have a presence in the online world whether we use the internet or not and protecting yourself from identity fraud is just one side of the story. Our digital presence also increasingly opens up new opportunities and influences real world decisions made about us. We now have a means of measuring and therefore managing the way we look online, we call it digital status. QDOS was developed as an exemplar Linked Data application whose aim was to provide a means of measuring and therefore managing ones' digital status. QDOS measures digital status by calculating how active one is in the online world. A given QDOS score is made up of four different components:

- **Popularity** Who you know and the extent of your online network.
- **Impact** How many people listen to what you say online.
- **Activity** What you do online e.g. shop, chat, blog.
- **Individuality** How easy you are to find online according to your name, your age etc.

Recent research has shown that more and more British citizens are making decisions based on digital status<sup>25</sup>. Already 16% have chosen their new home based on how their prospective neighbours appear online. 1 in 5 (20%) have researched a prospective boss online before accepting a job and 32% have searched online to find out more about trades people and professionals, from plumbers to lawyers, before hiring them to do a job.

As illustrated in figure 2 QDOS's architecture is very similar to that of DataPatrol insofar as it is powered by a clustered RDF store, namely 4store. One major difference between QDOS and DataPatrol is that QDOS exposes the contents of its RDF store as publicly available Linked Data<sup>26</sup>, via SPARQL endpoints<sup>27</sup>, and via a number of RESTful APIs as described in section 3.2.1. QDOS is written in PHP<sup>28</sup> and Perl<sup>29</sup>, and makes use of SPARQL, the SPARQL protocol, and HTTP.

As it stands the QDOS 4store instance holds approximately 10 million RDF documents describing people and their social graphs. These RDF documents are of the type `foaf:PersonalProfileDocuments` an RDF class that is defined by the Friend-Of-A-Friend ontology (FOAF)<sup>30</sup>. More information regarding the FOAF based services provided by QDOS can be found in the following section 3.2.1. Work is currently underway to collect more personal profile documents from the Web. A decision will also be made to port QDOS over from 4store to 5store, and given that

---

<sup>25</sup> Garlik commissioned PCP market research consultants to conduct research among 2000 UK adults in November 2007

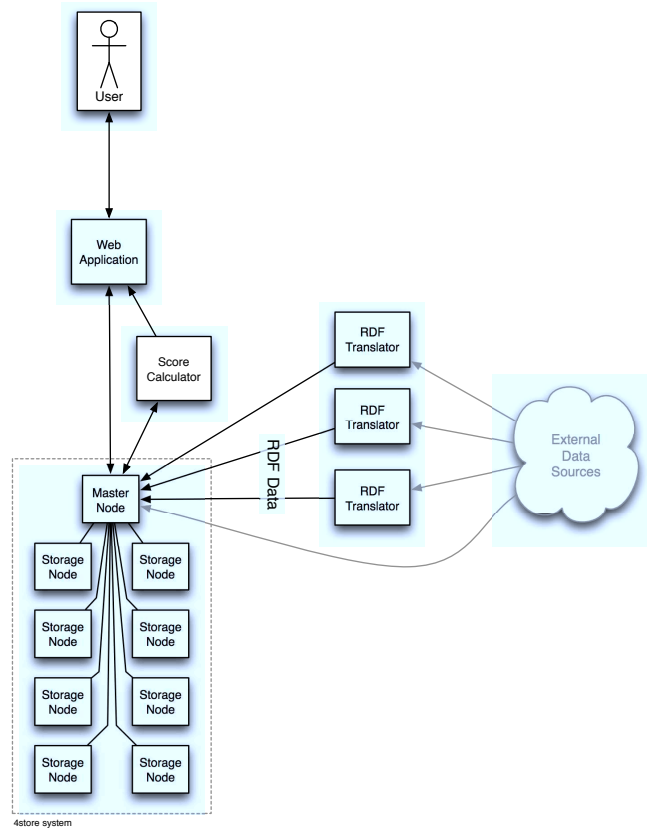
<sup>26</sup> See for example <http://qdos.com/user/5acc361496df109a7c2967760d5d9792/rdfxml>

<sup>27</sup> Once logged in as a QDOS user a SPARQL endpoint can be found at <http://qdos.com/query>

<sup>28</sup> Hypertext Preprocessor <http://php.net/>

<sup>29</sup> Perl Programming Language <http://www.perl.org/>

<sup>30</sup> FOAF ontology <http://xmlns.com/foaf/0.1/>



**Fig. 2** QDOS Architecture

both of them implement the SPARQL protocol, and all interacts with the RDF stores are performed via HTTP, this should be a trivial upgrade.

In figure 2 one can see that as in DataPatrol, QDOS is primarily a Web Application, whereby users interact with the system via a web-frontend. QDOS has a module known as the “Score Calculator” which is a background batch process that is constantly refreshing its cache of personal web pages, and publicly available FOAF documents from social networking sites such as, Twitter<sup>31</sup>, Flickr<sup>32</sup>, Lastfm<sup>33</sup>, Facebook<sup>34</sup>, and so on. This constant recalculation of individuals’ QDOS scores has meant that 4store has had to be able to cope with constantly updating RDF, which as it stands averages approximately 10 writes per second. Just like in DataP-

<sup>31</sup> Twitter Social Networking Site <http://twitter.com/>

<sup>32</sup> Flickr Photo-Sharing Site <http://www.flickr.com/>

<sup>33</sup> Lastfm Online, Social Radio Site <http://www.last.fm/>

<sup>34</sup> Facebook Social Networking Site <http://www.facebook.com/>

atrol, all of the interactions between the QDOS system and its RDF store are via the SPARQL protocol, and are performed over HTTP.

The “Score Calculator” iterates through all of the users with QDOS profile pages, and looks at QDOS’s RDFS<sup>35</sup> schema<sup>36</sup> in an attempt to classify the various accounts held by a QDOS user in order to extract numbers that are in turn used to calculate the given user’s final score. This mechanism used to process each of the individuals within the QDOS system is described in further detail in section 4 and is presented as a public example of how Garlik makes use of schema driven software.

### 3.2.1 foaf.qdos.com

foaf.qdos<sup>37</sup> is the experimental part of QDOS, and has a number of publicly available services<sup>38</sup>. The services provided under the foaf.qdos domain run on top of the same 10 million individual strong 4store instance that powers QDOS. foaf.qdos presents application developers and Semantic Web enthusiasts with a number of services that can be used to help make the world of FOAF data a more interconnected experience. There are a large number of FOAF emitting sites which can be found on the Social Web<sup>39</sup> but as it stands, FOAF documents in isolation are not said to be of much use. FOAF documents provide information about an individuals, the `foaf:primaryTopic`, and information regarding people which they claim to know. In order for a social graph to be useful, a list of people claiming to know an individual is said to make all the difference, and this is not possible without an index of FOAF documents. foaf.qdos.com is presented as a perfect example to why, Linked Data on its own, will not suffice, the ability to be able to store high volumes of RDF data, in a quad-format, preserving the document URIs from the data was extracted is presented as key to being able to find information from the Linked Data Cloud – FOAF is only useful when one can both find information about who a given `foaf:Person` knows, and information regarding who claims to know the given `foaf:Person`<sup>40</sup>.

The nature of the information stored within the QDOS 4store instance provides a perfect example of why Garlik’s motivation to not include reasoning to our RDF stores was a sensible one. FOAF documents harvested from the public Web come with a number of errors, and reasoners do not react well to scruffy data. Examples of such scruffiness include, cut and paste errors, whereby people cut and paste RDF fragments from each others FOAF files, which in turn would make the two individuals merged into one. Other examples include the fact that many peoples FOAF files’ include the URL `http://www.google.com/` as their `foaf:homepage`

---

<sup>35</sup> RDF Schema (RDFS) <http://www.w3.org/2000/01/rdf-schema#>

<sup>36</sup> QDOS Schema <http://qdos.com/schema>

<sup>37</sup> foaf.qdos <http://foaf.qdos.com/>

<sup>38</sup> QDOS’s service list <http://qdos.com/apps>

<sup>39</sup> List of FOAF Emitting Site <http://esw.w3.org/FoafSites>

<sup>40</sup> Making FOAF useful slides: <http://foaf.qdos.com/slides/london011209/index.html>

properties, which in turn is an Inverse Functional Property (IFP)<sup>41</sup> of the FOAF ontology, and the semantics of the property state that a `foaf:homepage` is a page describing a `foaf:Person`. If a RDF store with reasoning was used, all of the `foaf:Persons` with the same, incorrect, `foaf:homepage` would be merged into one person. This does not go to say that foaf.qdos does not do any reasoning, it just performs all of its reasoning at query time using the SPARQL query language. Querying are executed over QDOS's 4store instance, that in turn merge `foaf:Persons` based if they share IFPs or if they are said to be `owl:sameAs`<sup>42</sup> of another `foaf:Person` – this is referred to as IFP Triangulation and it is this triangulation that powers most of the services listed below.

The services described below are practical examples of an applications which run off of our 10 million individual strong FOAF based 4store. The current size of our QDOS 4store instance is approximately 2GT.

- **FOAF Index**<sup>43</sup> Provides an ability to search through the 10 million FOAF files harvested, searches can be made for any of FOAF's IFPs<sup>44</sup>.
- **FOAF builder**<sup>45</sup> Is a user-interface that allows for people to merge their various FOAF profiles into one, and then to save the RDF data back to their servers, or to a Garlik server if the user does not have access to their web-space.
- **FOAF Validator**<sup>46</sup> Allows for FOAF files to be validated against the FOAF ontology.
- **FOAF Reverse Search**<sup>47</sup> Is a RESTful API, which allows one to find out who claims to know a given FOAF Person.
- **FOAF Forward Search**<sup>48</sup> Is a RESTful API, which allows one to find out who a given FOAF Person claims to know.
- **FOAF SameAs Search**<sup>49</sup> Is a RESTful API, which allows one to find out which other FOAF People are the same as given FOAF Person.
- **Social Verification**<sup>50</sup> This is a RESTful API<sup>51</sup> that can help you use FOAF data to act as a whitelist for blog, email, and other online activity.

---

<sup>41</sup> Inverse Functional Properties <http://esw.w3.org/InverseFunctionalProperty>

<sup>42</sup> `owl:sameAs` relationship from the OWL ontology <http://www.w3.org/TR/owl-ref/#sameAs-def>

<sup>43</sup> FOAF Index <http://foaf.qdos.com/>

<sup>44</sup> FOAF's IFPs <http://mmt.me.uk/blog/2009/09/07/foaf-ifps/>

<sup>45</sup> FOAF Builder <http://foafbuilder.qdos.com/>

<sup>46</sup> FOAF Validator <http://foaf.qdos.com/validator/>

<sup>47</sup> FOAF Reverse Search <http://foaf.qdos.com/reverse>

<sup>48</sup> FOAF Forward Search <http://foaf.qdos.com/forward>

<sup>49</sup> FOAF SameAs <http://foaf.qdos.com/sameas>

<sup>50</sup> FOAF Social Verification <http://foaf.qdos.com/verify-about>

<sup>51</sup> FOAF Social Verification Demo <http://foaf.qdos.com/verify-about>

## 4 Schema Driven Software Deployment

As mentioned earlier on the chapter, both DataPatrol and QDOS make use of what we will refer to as “Schema Driven Software Deployment”. This technique is employed within Garlik’s services in order to be able to deploy new functionality without having incurring any service downtime. Given DataPatrol’s requirement of being agile, insofar as being able to search through and index new sources of compromised data as and when they are discovered, allowing for the highest level of protection for its users, the ability to add search rules in an ad-hoc fashion is seen as core to its success. DataPatrol’s “Processing Service” (see figure 1) that specialises in finding our customers’ personal information in DataPatrol’s feeds of compromised data, performs its searches based on an RDF schema that defines the list of data types it wishes to identify. This means that given the identification of a new source of compromised personal information, all that is needed is for a new “RDF Translator” to be implemented, which would capture this new source of potentially harmful data by converting it to RDF, and adding it to DataPatrol’s 5store instance. This would be followed by a change to the RDF schema that dictates the functionality of the “Processing Service”. Once the change has been made to the search schema, and the new search rules have been added to the 5store instance, the “Processing Service” will then start to search for the newly added information.

A public example of this can be found in the QDOS schema, which in turn operates in a similar fashion. The QDOS schema, which can be found at the following URL <http://qdos.com/schema> contains definitions of a number of service properties<sup>52</sup>, e.g. <http://qdos.com/schema#twitter>, each of which have a canonical URI pattern<sup>53</sup>, and a parsing module<sup>54</sup> associated to them. Each of the modules are also described in the QDOS schema, whereby the statistical data in which a given service property generates is listed<sup>55</sup>. As each of the QDOS users are processed by the “Score calculator” backend process, all of their online accounts are matched against the QDOS schema, where the backend process identifies which “RDF Translator” module should be used to gather statistics relevant to that user’s given online account. The RDF generated by the various “RDF Translators” is publicly accessible and can be fetched by concatenating `-ext/rdfxml` to the end of given QDOS user’s URI, e.g. <http://qdos.com/user/5acc361496df109a7c2967760d5d9792-ext/rdfxml>. It is this statistical data that is used to calculate a given QDOS user’s digital status.

The process described above allows Garlik to deploy and add new functionality to its services without the need to restart or re-deploy any of its existing software allowing for maximum service uptime. This gives our developers greater freedom to rapidly deploy software updates, allowing Garlik the ability to react, in an agile manner, to new business requirements as and when they come about.

---

<sup>52</sup> QDOS Service Property <http://qdos.com/schema#serviceProperty>

<sup>53</sup> QDOS Canonical URI Pattern <http://qdos.com/schema#canonicalUriPattern>

<sup>54</sup> QDOS Parsing Module <http://qdos.com/schema#module>

<sup>55</sup> This is listed via the <http://qdos.com/schema#producesDatum> property

## 5 Technology and the Need to Scale

Garlik's RDF stores provide facilities over the HTTP protocol to allow access to information held within them. We have already discussed the SPARQL protocol which allows queries to be run over HTTP, but in addition 4store and 5store allow data to be added and removed using the HTTP `PUT` and `DELETE` verbs<sup>56</sup>.

Using existing, widely implemented HTTP features for data management means that updates can be made by lightweight, asynchronous, loosely coupled parser processes which have little knowledge of the rest of system. An example of this can be seen in figure 1.

The dataset required to run DataPatrol for the current userbase comprises around fifteen billion triples (15 GT) of RDF data. This data comes from a variety of sources, with a variety of different conditions on its use, so tracking of the use of this data is key requirement. When data is imported into the DataPatrol RDF store, meta-data relating to its provenance is imported along with it.

The use of this meta-data for enforcing licencing conditions and providing MI reporting has been previously discussed in section 3.1 however there are additional uses. As new data sources, and new releases of the software do not necessarily happen in synchronisation queries can be written target specific data sources, or exclude ones that are not yet ready for production use.

The schema-less nature of our use of RDF allows these new data sources to be imported into the RDF store without changes to any schemas, or affecting the operation of existing processes. RDF namespacing is used to distinguish new types of data, so existing queries, designed before the inclusion of the new data items will not be presented with new results.

### 5.1 4store

When Garlik was founded, around 2006, it was realised that RDF data representation would be appropriate for the data problems that the company would face. The expected business requirements were for a system capable of storing one billion RDF triples, and answering queries over this dataset in under ten milliseconds.

In order to meet these goals it was felt it was necessary to engineer a new RDF store (4store [5]), based on clustered software in order to allow access to enough main memory to allow the indexes for the RDF data to be held in RAM, in order to meet the processing speed requirements.

For a RDF dataset consisting of one billion triples, a sustained import speed of seventy thousand triples per second (70kT/s) is required in order to allow the dataset to be restored from backup data in four hours. This meant that import speed was a major consideration as it must be possible to restore any business-critical system with a minimum of downtime.

---

<sup>56</sup> Information on 4store's SPARQL server <http://4store.org/trac/wiki/SparqlServer>

Similarly, it must be possible to make the system tolerant of hardware, or operating system faults, and so the software was made resilient against the loss of a node in the cluster.

As it happens the requirement for one billion triples was optimistic, and the final production system running under 4store contained approximately fifteen billion triples.

In order to keep hardware costs to a reasonable level 4store was designed to run on tens of low cost x86-64 servers running the Linux operating system. Using commodity servers it is possible to run at a density of around 500 million triples per cluster storage node, while maintaining the required import and query performance.

Since its initial development 4store has been replaced within Garlik by a new clustered store with even greater scalability and efficiency. The 4store source code has been made available under the GNU General Public Licence version 3. The ANSI C99 source code and documentation can be found in the project's source code repository<sup>57</sup>.

## 5.2 5store

As 4store's capacity was being stretched by holding fifteen billion triples, and the import performance, of around 150 kT/s was felt to be insufficient it was decided to develop a new generation of RDF store for internal use in Garlik. This new store is named 5store, and has been in production use inside Garlik since June 2009.

5store was developed with extreme scalability in mind. It is designed to operate with clusters of many hundreds of machines, and approaches double the triples per machine density of 4store. Its strengths are performance, scalability and stability in the domain of RDF storage and SPARQL queries. It does not attempt to implement additional features beyond this.

As both 4store and 5store adhere closely to the SPARQL standards, upgrading from one to the other was reassuringly straightforward. No software changes were required within the application layer, and all the data generated by the conversion processes could be used without change. This contrasts starkly with our experiences of working with differing RDBMS storage systems, where migrations from one system to another brings a host of incompatibility issues.

### 5.2.1 Platform

5store was designed from scratch for clusters of up to 1,000 machines and scale of over 1 Trillion ( $10^{12}$ ) triples should be easily achievable given a large enough cluster. Given the new indexing algorithms deployed, there is no effective limit on the number of triples stored, rather it is constrained by the practicalities of cluster

---

<sup>57</sup> 4store source code repository <http://github.com/garlik/4store>

size, network speed and so on. 5store is implemented in ANSI C99 and C++. It makes use of custom discovery and clustering technologies and proprietary, novel indexing algorithms. It is primarily intended to run on 64-bit Linux platforms with high RAM density.

### 5.2.2 Performance

5store maintains similar query performance to 4store, but even greater emphasis is placed on import performance. With increasing sizes of production RDF stores the time taken to reimport data into a system is becoming even more important. 5store comfortably achieves sustained import speeds of a million triples per second on a 16 node cluster, allowing a ten billion triple store to be restored in under three hours.

## 6 Conclusions

- **Practicality** The capabilities of the SPARQL 1.0 query language are somewhat limited compared to typical SQL<sup>58</sup> implementations. This often means that more processing work is required in the application layer, and can mean that more queries are required to answer the same question, compared to a SQL-based implementation. However, this is offset by the convenience of the high degree of standardisation between stores, and the high degree of flexibility in data representation.
- **HTTP+RDF** All of our services communicate using RDF. The consequences of this are that we achieve a high degree of flexibility with respect to staged deployments incorporating new capabilities.
- **Cost Effectiveness** Despite the high engineering effort involved in developing two cluster-based RDF stores, we feel that overall this effort was well justified. The data volumes that can be held by 5store in particular are similar to those held in Key-Value stores, but yet we have access to a standard, and relatively sophisticated query language that supports very efficient join operations.
- **Reliability and Uptime** There is inevitably some concern when deploying software on top of a newly developed database system, however we have been operating our production system on 4store and 5store for several years now, while retaining high uptime, and without data loss. Engineering of the storage system for robustness, and the use of redundant clusters has helped in this area, and we do not feel that we suffered any more downtime than if we had gone for a more traditional RDBMS.

---

<sup>58</sup> Standard Query Language <http://en.wikipedia.org/wiki/SQL>

## 7 Future Work

Future work in this area includes further work in the 5store RDF store, increasing the storage density, scalability and query performance. Increasing the efficiency has a direct impact on the operating costs of the business and is a high priority.

The increasing use of RDF data in public sector data publishing<sup>59</sup> has the potential to be a great benefit to the business, and we will be pursuing opportunities to make use of this data within the company in the near future.

We are also exploring the implementation of the SPARQL 1.1 standard, while retaining the high performance and scalability that is essential to the operation our business.

**Acknowledgements** This work described in this chapter has been undertaken by Garlik Limited, Garlik was founded by Mike Harris, founding CEO of Egg plc, former Egg CIO Tom Ilube, and former British Computer Society president Professor Nigel Shadbolt. As the first company to develop a web-scale commercial application of Semantic Technology, Garlik enables consumers to find and understand what personal information is in the public domain about them and manage how their identities appear online.

Garlik appointed world-class technology experts to advise the business, including Professor Dame Wendy Hall CBE from the University of Southampton and Sir Tim Berners-Lee.

Garlik is backed by three of the UK's leading blue chip investment firms, Encore/DFJ Esprit, Doughty Hanson, and Noble Venture Finance.

## References

1. Bizer, C., Cyganiak, R., Heath, T.: How to publish linked data on the web. Web page (2007). URL <http://www4.wiwiw.fu-berlin.de/bizer/pub/LinkedDataTutorial/>. Revised 2008. Accessed 07/08/2009
2. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW '05: Proceedings of the 14th international conference on World Wide Web, pp. 613–622. ACM, New York, NY, USA (2005). URL <http://portal.acm.org/citation.cfm?doid=1060745.1060835>
3. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. 2(2), 115–150 (2002). DOI 10.1145/514183.514185. URL <http://dx.doi.org/10.1145/514183.514185>
4. Harris, S., Gibbins, N.: 3store: Efficient bulk RDF storage. In: PSSS'03, pp. 1–15. Sanibel, FL (2003)
5. Harris, S., Lamb, N., Shadbolt, N.: 4store: The design and implementation of a clustered RDF store. In: ISWC 2009 SSWS Workshop 09. Washington DC (2009). URL <http://4store.org/publications/harris-ssws09.pdf>
6. Manola, F., Miller, E.: RDF primer: W3C recommendation (2004). URL <http://www.w3.org/TR/rdf-primer/>
7. W3C: SPARQL query language for RDF, working draft. Tech. rep., World Wide Web Consortium (2005). URL <http://www.w3.org/TR/rdf-sparql-query/>

---

<sup>59</sup> UK Public Sector Data <http://data.gov.uk/>