# Linked Sensitive Data

Steve Harris and Mischa Tuffield: Garlik Ltd, October 2011 (Acknowledgements)

This paper expresses some of the issues faced by Garlik Ltd when processing commercially and legally sensitive data, using Linked Data principles within an enterprise environment. This input can be used for the W3C Workshop on Linked Enterprise Data Patterns (6-7 December 2011, MIT, Cambridge, MA, USA).

## Garlik's Position Summary

This paper describes some of the challenges faced by Garlik Ltd when storing and querying sensitive data organised using Linked Data principles. These challenges are listed below:

- Pay-per-Query Datasets
- Data Lifetime management
- Secure Erasing

## Pay-per-Query

There are often datasets and other information sources that whilst held within the enterprise's datacentre require payment for each time the data is accessed or queried.

This presents two challenges:

1. Generating accurate Management Information (MI) to raise the correct Purchase Order with the given data provider.
2. In pure Linked Data sources minimising the access to chargeable resources, to control costs.

### MI Generation

When accessing data as Linked Data via simple HTTP requests the accounting is relatively simple. Normal access accounting methods may be used though great care has to be taken to ensure that the actions of the requesting client conform to a single "Request" as per the data contract.

When accessing the data via a SPARQL interface, that may contain data from a number of sources and can be chargeable or non-chargeable, it can be difficult to determine exactly which queries accessed how much of the various datasources without generating too much overhead. Below we sketch an algorithm of the usage accounting system we implemented to help determine this information.

This usage accounting is based on which triples in a database are traversed by the queries. We use a statistical technique to estimate this, examining one in every N quads referred to by the query engine. Consequently the output figures are only accurate to the nearest N. In our tests the overhead of enabling this accounting for N=1000 was almost negligible, less than one percent increase in query execution time. Depending on the billing demands of the data provider, and the quantity and predictability of queries executed in each billing period different values of N may be appropriate, even as low as 1.

Given the Garlik systems datasets are stored under different graphs within our databases. The output of the accounting process is two columns; a count and the name of the SPARQL graph. Graphs which were not access by any queries are not listed.

In our cases the number of graphs of data belonging to each data provider with per-query billing requirements is small, so it is easy to manage this externally. If the number of graphs or data contracts were large it may be necessary to model the mapping between graph IRIs and billing contracts in some way.

## Data Lifetime

Managing Retention and Expiry

Another issue that presents itself when making use of commercial data in a Linked Data environment is tracking data retention requirements. These can cause issues both for data which must be retained for certain minimum times, beyond which they would normally be used by the system, and data which must be removed from the live system at a certain expiry time.

### Minimum Retention Requirements

Extended retention may not be an issue, except that the storage of large quantities of RDF data can be expensive, and is preferably avoided due to additional management overhead.

Retention may be addressed by maintaining regular backups or dumps of the entire SPARQL store / Linked Data set, though given the requirements in the next section this can be difficult.

An alternative approach is to mark the graphs containing information which must be preserved with some metadata, and backing up those graphs in isolation from the rest of the data.

### Maximum Retention Requirements

Some data carries with it requirements for maximum retention, for example data held under European Data Protection laws may only be held for a certain period after the person described ceases to be a user of the system.

Again this can be dealt with by annotating appropriate graphs with the information about what user they relate to, and removing the graph once that person has ceased to be a user and after some time has elapsed.

Example:

```
DELETE {
  GRAPH ?g { ?s ?p ?o }
}
WHERE {
  GRAPH ?g {
    ?g dc:subject <http://my.example/user/1234> .
    ?s ?p ?o .
  }
}
```

However, this approach requires that the information relating to specific people is placed in appropriate graphs, or that the information is divided along lines which allow the relevant RDF data to be identified.

There are also issues in some systems where deletion of triples from the database is a computationally expensive operation, suggesting batch deletion at fixed time periods, but the syntax of the SPARQL language doesn't make this especially convenient to express.

## Secure Erase of RDF data

Secure erasing of Enterprise Linked Data served over HTTP can suffer from client-side caching. Measures need to be taken to ensure that client caching of sensitive Linked Data is minimised when accessed.

Within RDF databases it is often necessary to delete portions of the data, in a way which is secure (see Data Retention) without impacting runtime performance too heavily and in ways which can be demonstrated to make the data removed unrecoverable.

An example of where secure erase would be different from a standard SPARQL DELETE operation is in resource storage. In Garlik's RDF databases quads are removed from the indices immediately but the mapping between the quad symbols, and IRI / literal values are retained for some indeterminate time. A secure erase would also require a garbage collection (of at least resources touched by the delete) to be run immediately, and complete before the secure delete command returns. On top of this the quad storage rows would have to be overwritten multiple times, with different patterns of data, as per the Gutmann Algorithm or equivalent.

So far Garlik does not have a practical solution to this problem, where secure erase is required we store the data that will be erased in a separate database, and secure erase the database index files on expiry.

## Conclusions

Within Garlik all of our Linked Data resources are assembled dynamically using queries made against SPARQL servers. Garlik has yet to find solutions to the problem of securely erasing RDF data from SPARQL stores. Nonetheless the technology stack allows for Garlik's sensitive data to be organised and partitioned in meaningful ways allowing us to meet our obligations to our data providers and to our end users.

## Acknowledgements

- Nick Lamb Garlik Ltd
- Cezary Biernacki Garlik Ltd