TAXONIC

LINKED DATA IN THE ENTERPRISE

How information enlightenment reduces the cost of IT-systems by orders of magnitude

Second Edition

With a Foreword by Irene Polikoff

LINKED DATA IN THE ENTERPRISE. How information enlightenment reduces the cost of IT-systems by orders of magnitude. Second Edition. With a foreword by Irene Polikoff

A Taxonic whitepaper, by Taxonic B.V., Utrecht, The Netherlands

http://taxonic.com

Foreword by Irene Polikoff

In the beginning, software processing was about automating paper-based business processes. The processes modeled were reasonably static and the data often fit into the tabular format. Hugely successful, this approach resulted in the digitization of immense amounts of information creating a plethora of siloed data sources.

As data grew, understanding the connections between these disparate sources became as important or, arguably, even more important than each source alone. Enabling connectivity requires a method for resolving identity and meaning of information elements across the sources. Today, we are increasingly interested in new types of data that are more diverse in structure and less predictable in nature. We capture interrelated information about all things happening around us and have to implement systems that are considerably more dynamic than those of the past. Each year, the speed of change in business, and in software systems that support everything we do, accelerates. Using established traditional technologies to answer these needs has been driving up the cost and complexity of enterprise IT systems.

These drivers have led to the rise of new technologies, referred to as NoSQL (Not Only SQL). While they are called schema-less, this does not literally mean that the data they manage has no structure. It means that the data is stored in containers that are much more fluid than the relational model permits. This is useful when data lacks uniformity. It is also critical when its structure and content can change unpredictably and frequently. For example, as a result of new legislation, new business initiatives or new policies.

The NoSQL technology landscape in some way still resembles the "wild west" of the past with many proprietary approaches that have evolved from the initial implementations by the web giants like Google, Facebook and Amazon. At the same time, relevant standards have been developed by the W3C (World Wide Web) consortium. They offer a standard approach for describing rich and flexible data models and for querying the model and data alike. Importantly, they also offer a way to uniquely identify, connect and access data across many diverse sources. This standards-based approach is becoming known as "Linked Data" as it enables the interconnection of rich networks of data. A growing number of standards-compliant products offer a interoperable alternative to using proprietary technologies that is much more 'future-proof' in terms of enabling unanticipated changes, additions and dynamic interconnections among data sources

Jan Voskuil's paper starts with an excellent introduction to the concepts and technologies involved in turning enterprise data into Linked Data. Jan follows

this introduction by talking about how one may build applications that take advantage of Linked Data. He describes real examples of how this approach reduces the cost of data integration and implementation of agile and flexible systems needed to support modern enterprises. I am certain that reading the paper will help you gain key insights into using Linked Data technologies to achieve the promise of enterprise interoperability.

Irene Polikoff
CEO and co-founder of TopQuadrant

TopQuadrant was founded in 2001 and is a leading world-wide vendor of enterprise metadata management and data integration solutions based on Linked Data and semantic technologies.

Contents

Adjusting IT-systems often leads to destructive change. In a changing environment, this is the source of problems and massive cost. Each piece of new legislation, each change in policy results in a cascade of highly expensive modifications. The inertia of IT-systems might be significantly alleviated with the uptake of Linked Data. This will increasingly enable creative change. IT-systems will not only be modifiable at low cost, organizations will have the ability to leverage their creativity by "playing" with their systems and discover new ways of creating value. This is a dream that certainly will be reality soon.

Linked Data is a method of storing, managing and sharing data that fundamentally differs from the classic methods: the relational database and SOA-based web services. The technology is not new as such — the first iteration of some of the key underlying standards were established by the W3C consortium ten years ago. Developments in supporting tools and a growing uptake have resulted in a number of remarkable successes. In this paper we answer the question as to how Linked Data will reduce the cost of modifying IT-systems and data integration.

The intended audience is anyone with an interest in the cost of IT and some curiosity in the above question. We won't go into much detail — let alone technical detail. We explain just enough to understand the answer at the conceptual level.

We start with a short description of how Linked Data works. Next, we discuss why relational databases are a major source of inertia and cost. We then contrast Linked Data and relational databases to bring out the difference and understand how Linked Data makes creative change a possibility. We discuss some exciting real world business cases, primarily in the domain of public administration in the Netherlands — where cost of IT is a perennial issue. In the final chapter, we draw some conclusions.

LINKED DATA: A MANAGEMENT SUMMARY

Referential Semantics

The essential difference between Linked Data and relational databases is the way they handle the meaning of data. So that is where our exposé must start.

The question as to what the meaning of a symbol really is has puzzled philosophers, linguists and scientists through the millennia. One of the simplest theories of meaning is called referential semantics: meaning is what is referred to. Thus, if you know what the phrases 'James Bond' and 'MI6' refer to, and you know what the phrase 'works for' refers to, then you know what "James Bond works for MI6" means.

For this theory to work it is crucial that each symbol uniquely refers to a thing in our world, namely, James Bond, MI6 and the relation of working for, respectively. The relation between symbol and thing in the world does not need to be unique the other way around. Put differently, you may refer to the same thing using different symbols. Once you know two symbols refer to the same thing, you can start making inferences. Thus, if you know that the phrase '007' refers to the same thing as 'James Bond', you can, based on the previous statement, draw the conclusion that "007 works for MI6".

This is an elegant and simple theory of meaning. The theory is too simplistic to account for many of the subtleties of human language. However, it provides a solid basis for the purpose of managing data. It treats the meaning of data in a much more effective way than other methods. It is precisely this theory of referential semantics that underpins Linked Data.

Global Identity

The first step in developing a method for managing data based on referential semantics is finding a way to uniquely refer to things. This is, at first blush, not an easy task. In the world of databases, uniqueness of identification is localized: each row in each table has a unique key value. But this kind of uniqueness is defined in terms of the table in which it occurs. In another table, in another database, the same number may identify something completely different. In other words, identity is locally defined, and the meaning of the key differs across contexts. This will not do.

So, we need globally unique symbols. The problem of global uniqueness has been solved many times. Before the Web, this most often took the form of a global authority handing out IDs. An example is the ISBN-number assigned to books. With the advent of the Web, however, a broadly used, systematic method has emerged.

The Web rests on a decentralized way of defining global uniqueness. Each domain has a unique owner, who delegates ownership of subdomains to their respective owners, and so forth. In any case, we can be sure that the (fictitious) email address james.bond@mi6.gov.uk uniquely refers to one and only one

mailbox. Of course, there could be aliases referring to the same mailbox, such as 007@mi6.gov.uk. Email addresses are a special case of Unique Resource Identifiers, or URI for short. Another special case of a URI is the web address or URL, which locates a resource such as a web page on the Web. The fundamental insight here is that a URI is a globally unique symbol that can be used to refer to a unique thing.

In Linked Data, we call this thing a "resource". Everything is a resource: James Bond (a.k.a. 007), MI6, and the relation we call "works for", the USA, the milky way, the gene sequence characterizing the fruit fly, the fruit fly itself, its fondness of fruit, the girl next door and her being your neighbor — to mention just some examples. It may strike you as odd to use the term 'resource' in this way, but that's just the way it is. Internet pages and mailboxes are also resources, which happen to reside on the Web, but from the perspective of Linked Data these special cases are not especially interesting. The point is that we now have a mechanism to define globally unique symbols to uniquely refer to resources, that is, to things in the world.

Representing Facts as Triples

Now that we have URIs to refer to resources, we can use them to represent facts. Suppose we define the URI http://corporate-data.mi6.gov.uk/JamesBond to uniquely refer to James Bond. We could only "mint" this URI in this way if we were given the mandate to do so by the owner of the domain — let's suppose that we are, and that we mint similar URIs to refer to 'work for' and 'MI6'. Among web technologists it is common to abbreviate the first part of a URI (up to the rightmost slash) using some convenient namespace prefix — we will use "sis:" for this purpose. We could then represent the facts that James Bond and Moneypenny work for MI6 as follows:

```
sis:JamesBond sis:worksFor sis:MI6
sis:Moneypenny sis:worksFor sis:MI6
```

Each of these two assertions consists of exactly three URIs: a subject, a predicate, and an object. Such an assertion is called a triple.

Triples are stored in triple stores. One can think of a triple store as a table with three columns. From a technical perspective, that's all there is in the way of structure. By atomizing all information into factoids of this type, the number of triples in a triple store will soon become large: hundreds of millions of triples in a triple store is the normal situation. Enterprise-grade triple stores are expressly designed to deal with such numbers.

To work with triples, components are needed on top of the triple store. Collectively, these components are often called "semantic middleware" — in this paper, we call it tripleware for short. These components do a variety of things: enabling a triple store to be queried over the Web, supporting federated querying, making different kinds of inferences, and much more. Importantly, there is tripleware available for combining Linked Data and relational databases, so that these can be used alongside each other. It is even possible to make a relational database look like a triple store to the outside world. In the next section, we summarize four central tasks that tripleware can perform.

Tripleware

Tripleware can do a number of things. First of all, it allows users (and user programs) to create, retrieve, and delete triples in a triple store. Second, it provides a service for querying. Like a traditional database, it answers questions by assembling query results, using a standardized query language. In the traditional database world, this is SQL. With Linked Data, it is called SPARQL. SPARQL has the additional function of defining an interaction protocol, so that a triple store is said to present a "SPARQL-Endpoint" on the Web. Clients can fire queries at the endpoint and receive the answer.

The third and fourth things tripleware can do are drawing inferences and federate data stores. Let us discuss these in combination. Since the structure of a triple is so trivial, one can write a SPARQL–query and query multiple triple stores on the Web simultaneously. Suppose that we have a fan club that hosts a triple store in its own domain that contains the following triples, where we use "fc:" as the namespace abbreviation for the fan club's domain:

```
fc:007 fc:hasEliminated fc:Goldfinger
fc:007 owl:sameAs sis:JamesBond
```

The second assertion in the triple store is interesting in that it contains two URIs defined in a domain other than "fc". One of these is the URI referring to James Bond that we minted in the "official" (though fictional) SIS-domain. Since a URI is globally unique (and ownership is explicit), its meaning does not depend on context: the URI can be used anywhere and be part of a triple in any triple store — it always means the same, no matter in which context it occurs.

The other "alien" URI in the assertion is owl:sameAs — a URI that the W3C standards committee has minted as referring to the relation of, well, being the same resource as. The triple uses this predicate to assert that the URIs sis:JamesBond and fc:007 refer to the same resource. Provided we have an internet connection to both triple stores, we can now fire off SPARQL-queries and

receive the following triples back as answers. A query that yields the first of these is for instance "return all triples of which sis:JamesBond is the subject and fc:hasEliminated is the predicate." A similar query yields the second:

sis:JamesBond fc:hasEliminated fc:Goldfinger
fc:007 sis:worksFor sis:MI6

The two triples shown here are not as such present in either one of the two triple stores. They can be inferred, however, based on the assertion that the URIs sis:JamesBond and fc:007 refer to the same resource. Deducing new facts from existing ones based on knowing that two expressions have identical references may seem hardly impressive. SQL-experts could certainly achieve similar results. The difference is, however, that true inference engines simply apply general rules of logic without further instruction. We will briefly revisit inferencing below.

This example shows that it is utterly simple to combine triples from different sources. The simple, standardized structure of triples makes them technically interoperable by nature. The principle of combining URIs from different domains in one triple makes datasets interoperable at the semantic level. The example shows that this capability can be leveraged to issue federated queries against multiple triple stores on the Web. While the possibility of federated queries poses new challenges — for instance, with respect to performance —, the power to combine data from different sources constitutes a quantum leap. Ultimately, as we will see, it is this ability that soon will make IT-systems responsive to change.

We now almost know enough to understand how Linked Data will lead to more adaptable IT-systems. Yet, there are two additional points that warrant discussion here. We do so in the next two paragraphs.

Literals and Tabular Data

The first point is that the object in a triple can be what is called a literal — that is, a data value that is not a URI, but just a string of symbols. Just three examples are the following:[1]

sis:JamesBond sis:dateOfBirth "1920-11-11"
sis:Moneypenny sis:dateOfBirth "1933-12-21"

[1] The source of Bond's date of birth is Wikipedia, which also offers an alternative view. Moneypenny's date and place of birth are made up by the present author by way of example.

sis:Moneypenny sis:placeOfBirth “London”

It makes sense that a date is just a literal: there seems little point in defining a — globally unique — URI that uniquely refers to it. With Moneypenny’s place of birth, this is perhaps not so clear. It may be quite useful to have a URI for London. While there is no obligation to do so, treating the city as a literal instead of a resource does have consequences. For instance, we can’t say things about London in our triple store: literals can only be the object, not the subject in a triple. Choices such as this are at the heart of the modeling process with Linked Data.

The good news is that the use of literals gives us a simple recipe for translating back and forth between Linked Data and tabular data. Consider the following table:

EMPLOYEE		
ID	**Date of birth**	**Place of birth**
James Bond	1920-11-11	
Moneypenny	1933-12-21	London

To translate this to Linked Data, we define a URI for each row based on the cell containing the row’s ID. Next we define a URI for each column header, and treat the values in the corresponding cells as literals. Applying this to the table yields the three above triples.

The recipe is presented here in a simplified form limited to literals, but the bottom line is: a translation is always possible. As a result, using tripleware, relational data can be accessed as if it was stored in RDF. Thus, we can leverage relational data in Linked Data architectures without having to migrate all of it into triple stores. Conversely, using SPARQL, we can query our triple stores and generate the above table as a tabular view on the data contained in them, and export the data to a spreadsheet or relational database.

Knowledge Models and Deductions

The second point is that Linked Data has standards that allow us to say abstract, logical things about the information in a triple store. The most basic of the Linked Data standards is RDF, an abbreviation for Resource Description Framework, most of which has been presented in the previous paragraphs.

On top of RDF, there is a number of additional W3C standards, notably RDFS (“RDF Schema”), OWL (“Web Ontology Language”), and SPIN (“SPARQL

Inferencing Notation"). SPIN is still in the process of being ratified but already widely used. These additional standards cater for assertions about class membership, cardinality, relationships between values of properties and other such things. For example, we could say:

```
sis:Moneypenny rdf:type sis:Secretary
sis:Secretary rdfs:subClassOf sis:Staff
```

From this, we can query for Moneypenny as someone who is of type secretary or (through the subclass statement) as someone of type staff, even though the latter is not as such asserted in the triple store. Importantly, the additional standards do not add extra technical structure. Everything we say in Linked Data is said in the form of triples and can be added to the triple store. Taken together, RDF, RDFS, OWL and SPIN constitute an extremely powerful tool set to create knowledge models.

Suppose you cultivate vegetables and you maintain a database of which crops are cultivated where and sold to customers in which areas. Instead of referring to each kind of crop using a symbol made up by your IT-department — such as the strings "spinach" and "Toulouse" —, you could instruct them to use URIs defined by some authority in a published list. Such lists are called taxonomies or vocabularies. This would be an important step towards making your data interoperable with the outside world. If the vocabularies maintained by the authority (or plural, vocabularies maintained by different authorities) contains enough logic, fine grained and useful deductions can be made by combining your triples with knowledge from outside: management information on steroids. The marketing department could, without any IT-effort, ask which crops are most often bought by customers living in a place (all this information is in your organization's data sources) such that that place is known to be a city with more than 500,000 inhabitants (this information could be obtained at query time from, for instance, DBPedia, the public triple store based on Wikipedia).

In fact, Linked Data is broadly used for linking different sources for precisely that purpose. Before closing the chapter on Linked Data, let us briefly expand on that theme.

Vocabularies for Enriching Data with Metadata

In some areas, such as the life sciences and pharmaceuticals, vocabulary usage along the lines just mentioned is massive and done at Web scale, with hundreds of ontologies containing millions and millions of concepts and relations. The concepts in these ontologies are used to annotate research data, and to answer questions such as "which gene sequences occur in which species", or "which

markers are associated with which cancers". Since the amount of concepts is so vast, it is a practical necessity that the vocabularies are distributed over different triple stores and are maintained by different organizations.

Normal operation of entire segments of the industry and research community — such as genomics or cancer research— would simply be unthinkable without this technology. In fact, while the rest of us are pondering the introduction of Linked Data, folks in the life sciences domain have started developing the next generation of this technology: platforms for brokering between hundreds of ontologies. The European OntoCAT-project is an interesting and widely publicized example of this.[2]

A growing number of ontologies and vocabularies for enriching data becomes available in other domains too. A well-known and very early example is Dublin Core, which provides an RDF-vocabulary for describing publications and media files. By using the URI dc:creator to indicate the author property, the description of your media file becomes interoperable at Web scale. Because its meaning remains constant in different contexts, the URI is an ideal vehicle to carry metadata information used by many different organizations to share data.

Two Boxes and Some Philosophical Questions

The philosopher and logician Ludwig Wittgenstein (1889-1951) made a point of distinguishing expressions that say something about the world from those about the language we use. For instance, when I say that the table is round, I make a factual assertion: its truth depends on the world. Conversely, when I say that a round object has no corners, my statement is about how to use the concepts "round" and "corner." Wittgenstein called such expressions logical or grammatical statements.

In the late 1990's, the nascent Linked Data community distinguished the so-called "A-box" ("assertion box") and "T-Box" ("terminology box"). The triple representing that Moneypenny works for MI6 would be an A-box assertion, while the one that says that secretaries form a subclass of staff is a T-box assertion. The first is about the world, the latter is about structuring our conceptualization of it: it is a "grammatical" statement.

---

[2] See the OntoCAT website. This article on Biomedcentral provides a description: Tomasz Adamusiak et al (2013). "OntoCAT -- simple ontology search and integration in Java, R and REST/JavaScript".

Nowadays, many consider these terms artificial. For one thing, the only difference between the triples is semantic — the triples can live alongside each other in the same triple store, blurring the distinction. For another, the distinction can be quite subtle. According to the latest insights in phylogenetics a bird is a kind of reptile. Where would this statement fit? This is an invitation to explore fascinating and rewarding philosophical questions, but one that leads away from practical issues in a business setting.

Yet, it is useful to remember two things. First, with Linked Data there is no technical difference between data and metadata, only a subtle semantic one. Second, it is common and very useful to use URIs from external ontologies to enrich your data. Before moving on, let us take stock of what we have.

Linked Data in 147 Words

We started out with the theory of referential semantics, according to which we know what a symbol means if we know what resource the symbol refers to. We saw how URIs can be used as globally unique symbols that uniquely refer to a resource. Their meaning remains constant across different contexts. URIs are owned by whoever owns the URI's domain. Some resources are things, other resources are relations. Based on this, we defined the notion of triple. Triples are stored in triple stores. Triple stores are easy to combine because they all work with the same simple, standardized, atomic data structure. A number of W3C-standards allow us to make logical assertions about resources, such as sameness and class membership, in a logically rigorous way. Data can be enriched using metadata from controlled vocabularies. These features enable tripleware to make inferences and integrate data from disparate sources.

THE PROBLEM WITH RELATIONAL DATABASES

Structural Commitment to Capture Meaning

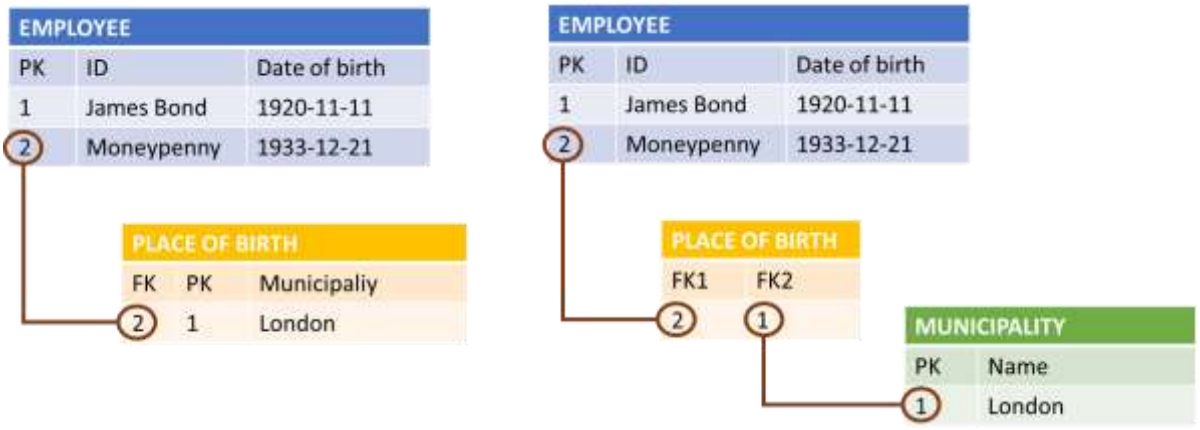
In the world of relational databases, which has shaped the field of IT for the past 40 years or so, things are not so simple. Data are stored in tables, and it is the structure of the tables that defines meaning. Without the structure, the data is meaningless. If you see the number 27 in a database or spreadsheet, you can deduce that it indicates someone's age by looking at the column header. The table structure, including the headers, tells you what is what. The consequence of this method of representing meaning is that before you can record any data, you have to define a structure first. The same meanings can be reflected in a literally endless amount of structures: you can represent age as column 5 of

table 3, but just as well as column 3 in table 5. The chance that the same meaning is represented in the same way in two spreadsheets — let alone two databases — is infinitesimally small. There are no rules to predict the data structure from what the data means.

Let us discuss some examples to illustrate the problem. Take as point of departure the table presented in the previous chapter. Probably, in a database we would add a primary key field (denoted PK) with a unique number assigned by the database, as follows:

EMPLOYEE			
PK	**ID**	**Date of birth**	**Place of birth**
1	James Bond	1920-11-11	
2	Moneypenny	1933-12-21	London

This is one way of representing our data in a relational database. Alternatively, we could promote the concept of "place of birth" from attribute to a separate entity. A foreign key field links each instance of Place of birth to the corresponding Employee. We now have two tables. As a third alternative, however, we could also model Place of birth as a relation between Employee and yet a new entity, Municipality. The relation table Place of birth links the two using a foreign key into each.



The choice between representing place of birth (1) as attribute *or* (2) as an entity *or* (3) as a "relation table" is unavoidable in designing the database structure: it has to be made, one way or another. To represent the handful of triples we have seen so far, there are many of such choices. Take the relation "works for" which we used in our first triples in this paper. The same three alternatives are possible. Similar reasoning holds for the triples asserting that Moneypenny is a secretary and that the set of secretaries is a subset of staff. Representing tree structures of types and sets in databases is a notorious source of complexity, with many possible solution patterns.

The amount of valid but distinct structures grows exponentially in proportion to the number of relations in the underlying vocabulary. This causes a combinatorial explosion of valid alternatives. Real-world databases contain dozens, if not hundreds of tables, each having handfuls of attributes, primary keys and foreign keys. And each such database only represents one structural possibility out of a million valid alternatives. Once you have chosen the structure, however, you are committed to it. This is called structural commitment.

Difficulties Resulting From Structural Commitment

Structural commitment is how relational databases handle meaning. It gives rise to four important problems. First, you have to understand the structure to be able to use the database. You cannot retrieve Moneypenny's place of birth without knowing in which tables these two data points reside and how they are connected. This is in stark contrast with a triple store: there, you have to know which URIs are used to refer to Moneypenny and place of birth, but there is no such thing as a data structure beyond that that you need to understand.

The problem of having to be aware of the exact structure of a database is exacerbated by the second problem: structures tend to become large and complicated, so that it becomes almost impossible to find your way. As long time Linked Data pioneer and visionary Dave McComb quips: if the databases in your enterprise contain more than 100,000 distinct attributes, the result is unstructured data.

The third problem is that relational databases are resistant to change. There are three reasons for this:

- Changing a complex structure in one place often leads to a change in some other place, and so on: the rippling effect
- Application programs have to reflect the structure of the database, so they need to change as well
- The data in the database needs to be migrated from the old to the new structure

The third point, data migration, poses a serious and often underestimated challenge in terms of effort and money. Changing the structure with two tables in the figure above to the structure with three tables, we need to unload the data from the old database, convert them to match the new structure with three tables, and load them into the new database. In real world situations, this is often a costly, error-prone activity. Moreover, the application software that uses the database needs to be refurbished so that it can use the new structure,

adding project dependencies to the challenge. In many IT-projects, migration is a serious cost component.

The fourth problem stemming from structural commitment is that in order to combine data from different databases, it is necessary to translate one structure to the other first. Next, you have to translate the data to the translated structure. This is so expensive that it is only done when really, really necessary.

The Relational Database Paradox

The problems associated to structural commitment have led software architects to the consensus that databases should as much as possible be isolated from the outside world using a layer of software. This layer of software provides data services to the outside world and concentrates the structural commitment inherent in using the database in one spot.

This is a good thing. However, while a services layer concentrates the problems in one spot, it doesn't solve them. Moreover, the architectural pattern leads to a paradox. In the early days of relational databases, most data used to reside inside application software. One of the most important selling points of relational databases was: free your data from the confines of an application. Put the data in a central repository available to all stakeholders across the organization. Let different applications share the same data.

Hiding the database behind a services layer makes the data less responsive to varying needs. Data can be used only through a limited set of services — much more limited than the set of queries one could run against the database directly. Without a service layers, combining relational databases over the internet would be extremely cumbersome because of the difficulties that arise from mapping one complex structure onto another. But with a service layer isolating the databases, database federation becomes completely impossible. The only way data from different sources can be combined is through a limited amount services, architected at design time, long before the system is ready for use.

The relational database paradox is that it wants to free data from the bounds of application programs, but has to lock the data up into a service layer to alleviate the problems stemming from its structural commitment.

THE PARADIGMS COMPARED

Let us now compare Linked Data and the relational database. The most basic difference is that Linked Data relies on referential semantics instead of structural

commitment to give data meaning. Secondly, like other Web technologies, Linked Data is fully and rigorously standardized at the technology level.[3] From these two points, most of the other differences follow automatically. There are also some similarities, however.

Similarities

An important similarity is that in both approaches, access control is based on partitioning the data. In relational databases, tables constitute the main vehicle for partitioning. Triples can similarly be partitioned into named graphs. For example, we could say that name and birth date triples belong to "personal data" Access control can then be based on this named graph. As you may have guessed, named graph membership is simply another triple and can be changed dynamically, offering more flexibility. At the core, however, access control is based on the same approach.

Another similarity pertains to the way restrictions can be imposed. Using RDFS and especially SPIN — the standard briefly mentioned earlier — it is possible to express logical constraints on RDF data, such as cardinality constraints. We could add triples to our Secret Service triple store stating that a resource of type "Person" can have one and only one birth date, but may bear a "worksFor" relation to zero or more resources of type "Organization". However, triple stores do not enforce such constraints. Thus, the storage level stays flexible and accepts any data — ahead of development of schemas and ahead of validation of the data constraints once they have been put in place.

Does this mean that using a triple store will inexorably lead to chaos? On the contrary. Data can be validated for conformance to constraints after it is written into a triple store. Exceptions can be identified and either fixed through transformations or reported on. While triple stores may not do such validation natively, widely used tripleware products add this capability and work directly with triple stores. With such products, not only data constraints can be expressed in RDF, but also (other) business rules. With this, applications can query not only for data, but also for rules and constraints. Tripleware can help enforce business rules making custom application code thinner, faster to develop and more responsive to change.

---

[3] SQL is an ISO standard, but implementations are not always consistent with the standard. See http://troels.arvin.dk/db/rdbms/. More important, however, is the fact that Linked Data is standardized all the way from persistence behavior up to and including interaction between the persistent store and clients, exchange over the Web using Web protocols, semantics of inferencing, and so forth.

In addition, it always possible to control what is written into the triple store. The business rules governing constraints on data may be enforced in the application software through which users enter data. This is not much different from how things work in the relational database world. Even though the structure of relational database will prevent storing two birth dates for one person, the application software must have its own awareness of the underlying business rule, and guide the user in such a way that double entry is impossible. It is a good software architecture practice to funnel incoming data through business logic before storing it. In that sense, on the application side, there is not so much difference between the two worlds after all.

Differences

It is on the data storage side that Linked Data makes a huge difference. With Linked Data, the lack of structural commitment causes application software to be loosely coupled to the data store. To see what this means, we discuss how this solves the four problems with relational databases presented in the previous chapter:

- To work with a database, you must understand its structure
- This structure is complex
- This structure is resistant to change
- Data in different databases cannot be combined without translating one structure to the other

The first two are easy. As explained earlier, there is no such thing as a data structure that you need to understand when working with Linked Data. Therefore, there is no problem either of structures becoming complex and difficult to follow.

Linked Data also solves the problem of inertia:

- You can simply add new triples to a triple store using existing or new URIs. There is no rippling effect
- Application programs can remain as they are: they simply don't "see" the new triples, and continue to operate normally
- With Linked Data, new information does not lead to data migration

A real-world example of this is the Linked Data information platform for fire fighters, provided by Netage.nl.[4] It is developed in cooperation with the

[4] I am grateful to Bart van Leeuwen, fire fighter at the Amsterdam Fire Department and founder of Netage, for providing the information in this example.

Amsterdam Fire Department, and used by fire fighters across the world. The platform is hooked up to a growing number of data sources. Fire fighters use a so-called monitor — a piece of application software that selects and presents data from the platform — to acquire actionable data about a location during transport to the fire. On departure from the fire station, hardly more than an address is known. Every bit of data added to that before going in may be crucial, life-saving information: type of building, previous incidents, location of doors and windows — to mention some examples. The platform is under continuous development. Each time a new data source is added, a new monitor is created that broadens the scope compared to previous versions. The old monitors continue to work as they did before. When an older monitor is not accessed anymore by users, it is simply taken off the air. A typical budget for upgrading the platform with new data sources and creating a new monitor is 15,000.– Euro.

The methodology of continuously upgrading the data store and the applications that use it would be completely impossible using a relational database — let alone with that type of budget.

The fourth problem, that of combining data from different sources, is the *raison d'être* of Linked Data. Triple stores are both technically and semantically interoperable. We have seen how this works, and it is the basis for combining data sources in the fire-fighting example above.

A REAL WORLD BUSINESS CASE

The claim that adoption of Linked Data reduces the cost of IT-change and the combining of data from different sources by orders of magnitude is supported by facts. We have seen one business case with numbers already in the previous chapter: the fire-fighting example. In this chapter, we add a few more.

Introducing Authentic Data Sources

The Dutch legislation compels public sector organizations to maintain and use a system of core registers. These contain so-called "authentic data" about citizens, buildings, real estate values, incomes, organizations, vehicles, geographic locations and more. When you move house all you do is register your new address with the pertinent municipality. The act of registration causes the authentic data source containing your residential address to be updated. Tax authorities, agencies handing out benefits and other governmental bodies will automatically send mail to your new address. The idea behind this system of authentic sources is that the basic elements of information management by the

government are of a legally guaranteed quality level and are available in one unique repository: the authentic source. The data in the different authentic sources are linked. The so-called BSN-number uniquely identifying your person links you to the car registered to you.

Currently, the system is based on relational database technology and SOA-based web services. One architectural pattern used in organizations that work with authentic data is that the consuming agency builds a local repository for storing the data they need in the structure they need, as part of their IT-landscape. For each piece of data — a person, a car —, they register a subscription. Each time the data is updated in the authentic source, the agency receives a web service call triggering an update in the local store. In the process, the data exchanged is translated from one structure to another. This is a costly way of sharing data.

Cost Reduction through Linked Data

In a recent article, Ria van Rijn and Arjen Santema point out that the budget of Logius, the agency responsible for hosting, maintenance and administration of the authentic sources, is in the order of magnitude of 50M Euro *per year*.[5] This does not include application software for data entry and maintenance of the data content, which is the responsibility of other organizations, such as municipalities. This does also not include the expenditures that organizations consuming the data have to make — these latter must be massive.

With Linked Data, completely different integration patterns spring to mind. Van Rijn and Santema argue convincingly that using Linked Data would lead to a cost reduction of tens of millions of Euro's per year.

Realizing the Business Case: Recent Developments

In fact, the Dutch government is already setting steps in the direction of making a transition, showing that the business case has more than theoretical value.

One initiative is the development of a national URI-strategy for Linked Data. This is a set of conventions for minting URIs. The current state is described in a recent article, and the first proposals have been submitted to Bureau Forum

[5] Ria van Rijn and Arjen Santema (2013), "Een nieuwe wereld, een nieuwe informatiearchitectuur." Available in the Pilod book, pp 211-221.

Standaardisatie, the national standards committee in the Netherlands.[6] The hierarchical nature of domain ownership underpinning the structure of the URI requires careful design of naming conventions. The government of the UK, the first public administration to adopt Linked Data at a significant scale, is currently revising its strategy, which is not free from problems. Minting URIs is something you want to get right at the start as much as possible.

Secondly, there are some interesting developments more directly related to the system of authentic sources. The so-called Stelselcatalogus is a catalogue of all the concepts used in these sources: street address, person, date of birth, house number, vehicle ID, and so on. Recently, the first part of the 2.0-version of the catalogue has gone life.[7] It is fully based on Linked Data technology and mints URIs for all the concepts that are part of the so-called semantic core of the system. It links each of these to relevant legislation, which is also in the process of becoming available as Linked Data.

Kadaster, the Netherlands' Cadastre, Land Registry and Mapping Agency, has already created an vocabulary representing the semantic core of the authentic source under its authority, the so-called BRK. A Linked Data version of this register will be launched soon. In a similar vein, the Ministry of Infrastructure and the Environment has announced that it will make its authentic data available as Linked Data as of 2015.

It is interesting to mention the fact that Stelselcatalogus 2.0 and the Kadaster's vocabulary overlap. The legal term for lot in Dutch is *perceel*. In the current situation, there are two URIs referring to this concept: brk:Perceel (minted by the Kadaster) and scbr:Perceel (minted by Stelselcatalogus).[8] A next step could be to deprecate one of the URIs and give preference to using the other. Or both URIs can be upheld and one can be asserted to be the owl:sameAs the other.

---

[6] See Hans Overbeek (KOOP) and Linda van den Brink (Geonovum), (2013) "Aanzet tot een nationale URI-Strategie voor Linked Data van de Nederlandse Overheid." Available in the Pilod book, pp 178-190.

[7] I am indebted to Joop Rasser, project lead of the Stelselcatalogus, for discussion. See also the homepage of the Stelselcatalogus and the project wiki.

[8] The full URI's are http://brk.kadaster.nl/def/gegevenselement/Perceel# and http://scbr.data.overheid.nl/brk/id/concept/Perceel. The first of these resolves to a URL that locates a webpage that provides a textual description of the concept referred to. Put differently, you can paste the URI in the address field of your browser and the information is shown right away — which is considered good practice by many in the Linked Data community. Textual information about the URI in the scbr-namespace can be found here.

This shows no less than four important points. First of all, it is another demonstration of the fact that overlapping ontologies can coexist without problems. Secondly, it shows that — because of this — vocabulary development can be done in a bottom-up fashion. You don't have to wait until the whole world agrees on one single vocabulary to be used exclusively. Third, it exposes once again the importance of a national URI strategy, including clear statements as to who owns which domains and is given the authority to mint which concepts.

The fourth point is more subtle. The system of authentic sources was architected 20 years ago. In a world view based on relational database technology, "11-11-1921" is data, and "Date of birth" is metadata. At the time, it was only natural that data were in the authentic sources, and a catalogue of metadata in the Stelselcatalogus. As we have seen, with Linked Data there is no technical difference between data and metadata. Therefore, the introduction of Linked Data compels us to rethink the function and position of the Stelselcatalogus, and the structure of the information contained in it.

Reflections

A business case for using Linked Data in the context of public administration is easy to make. Massive amounts of data are created and shared between different agencies. The network of organizations is large. The data travel in large amounts through countless edges linking them. This alone will account for great cost savings in transitioning to Linked Data.

In addition, the non-hierarchical nature of the organizational network prevents simplistic solutions for agreeing on one standardized vocabulary for all domains. To deal with this kind of complexity, it is much more effective to have many vocabularies alongside each other. Linked Data is expressly designed to support mixing and matching of metadata vocabularies. This adds strong qualitative force to the business case.

When it comes to vocabularies, there is a lot of low hanging fruit to harvest. An example is the Thesaurus of Legal Terms published by the WODC, the research institute of the Ministry of Justice in the Netherlands. The thesaurus is regarded as one of the best in its kind in Europe, because of its rich and fine-grained structure. It uses the SKOS vocabulary, one of the most central RDF-vocabularies around and expressly recommended by the W3C as the standard representation vocabulary for thesauri. Publishing it in the form of Linked Data will enable countless organizations in the legal domain to enrich their data and make the data interoperable at the semantic level. Given that all the elements are there except for the SPARQL-endpoint, this could be done at practically no

cost at all. There is no doubt that there are many more examples of high quality, trustworthy vocabularies waiting to be used.

All this does not mean, of course, that the business case for Linked Data works better within the public sector than elsewhere. Quite the opposite is true. We have discussed the massive use of this technology in the domain of life sciences and pharmaceuticals. The largest uptake of Linked Data is currently in industry and research domains. A recent and quite spectacular case is the BBC: all content for all its websites are served by a humongous triple store that serves up to 2000 SPARQL-requests per second, with an up-time worthy of the reputation of its owner.

MAKING THE PLAN COME TOGETHER

Linked Data will reduce the cost of modifying IT-systems and data integration by orders of magnitude. It caters for realization methodologies that are simply impossible with classical technology. We have seen how the Amsterdam Fire Department has its data store enriched with new data, introducing new information with new meanings. Yet, existing application software continues to work as if nothing has happened.

Apart from reducing cost, this will enable organizations to take a more creative stance to their IT-landscape. When changes can be made without much cost, it becomes possible to try different alternatives and discover what works best. In this way, change becomes a source of creativity instead of a destructive force of disinvestment.

The transition to Linked Data can be made in a step-wise, non-intrusive fashion. As the example of the Stelselcatalogus shows, Linked Data can peacefully coexist with relational database technology. The transition will never be trivial, however. While the technology is simple and solid, there are still enough challenges to warrant a careful approach. When the technical challenges are met, one also has to make sure due attention is given to the semantics of the information. Analysis and modeling with Linked Data are just as important — if not more so — as with relational database technology.

When Tim Berners-Lee invented Linked Data, he wanted to replace the Web of documents by a Web of data. His dream was that the data in all repositories on the Web could be combined and integrated in endless arrangements, without barriers — technological, semantic or otherwise. The Linked Open Data community is working hard to make this dream come true. The number of open

SPARQL-endpoints on the Web is currently experiencing an exponential growth comparable to what happened in the early days of the Web of documents.

As a side effect of this call for information enlightenment, new architectural patterns begin to emerge that will reduce the cost of IT-system inertia by orders of magnitude. Perhaps, the next big wave of uptake of the technology will be in the enterprise — not primarily to realize Berners-Lee's vision, but to make creative change possible. If that happens, it may be not so clear anymore what is a side effect of what. Either way, change is in the air.

Acknowledgments

I am indebted for information and inspiration to many. Numerous points mentioned in this paper derive from sometimes prolonged and always pleasant conversations with Bart van Leeuwen (Amsterdam Fire Department and founder of Netage), Marc van Opijnen (KOOP, Ministry of the Interior and Kingdom Relations), Joop Rasser (ICTU), Hayo Schreijer (KOOP), Arjen Santema (Kadaster), Hans Overbeek (KOOP) and Jan Verbeek (Be Informed). Thanks to Erich Gombocz (IO Informatics) for enlightening me on the subject of bioinformatics. I am indebted to Tomasz Adamusiak (Medical College of Wisconsin) helping me getting the description of OntoCAT right. Special thanks is due to Dave McComb (Semantic Arts), long time pioneer and visionary of Linked Data and a great teacher on top of that. Following his workshops and seminars has been an exciting way of getting introduced to the field.

About a year and a half after publishing the first edition, Irene Polikoff of TopQuadrant drew my attention to some issues, leading to long, deep and pleasant discussions. This resulted in a second edition with major improvements. I am grateful for this support. Of course, any remaining errors are my own.

About Taxonic

Taxonic helps organizations creating more value from their information flow. Linked Data-technologies are an essential ingredient of that. We consult in technology selection, provide expertise in tendering, and assist in the design, delivery, implementation and exploitation of solutions.

About the author

As CEO of Taxonic, **Jan Voskuil** is responsible for answering clients' business challenges with innovative solutions. After obtaining a PhD in theoretical linguistics, Jan worked for several start-ups in the field of artificial intelligence. Before co-founding Taxonic, Jan worked as senior solution architect at Logica and was involved in several large-scale, high-profile innovation programs.

Taxonic BV

Janssoniuslaan 80

3528 AJ Utrecht

The Netherlands

T +31 (0) 88 240 42 00

info@taxonic.com

www.taxonic.com

kvk 54529190

rabobank 161959660

btw NL851339803B01