Semantic Information Sharing: A New Approach to Storing and Sharing Data

Combining free-form data management with semantic-web-style search.

Ian Graham, Technical Director, Trireme International

As independent consultants, it's not often that Trireme sees fit to endorse a commercial product, but this one is so different from anything we've used before that we think an exception has to be made. OneIS (One Information Store) is a novel way to store all an organization's data, whether they are structured (as they must be in a conventional database) or entirely unstructured. These data are all stored in a single repository and can be retrieved in any order, as the case study below will show, and searched in a free-form manner that most people used to Google will be very comfortable with.

So, if you want to know if you should read further and find out more, ask yourselves these questions.

- Would I like to be able to browse and search all my corporate or project data as easily as I can search the web with Google?
  ... and to do this without missing key data?
  ... and to do this with data that are totally unstructured (the majority of data in most companies)?
- Would I like to be able to create a totally extensible, working, data-centred application in hours rather than weeks?
- As soon as I search for an item, would I like to be able to see all its linked information at a glance?
- Would I like to be able to share data and documents with colleagues in a safe, version-controlled way?
- Would I like to be able to use non-IT people to create, maintain and manage the data?

OneIS (pronounced One Eye Ess.) allows you to combine large amounts of structured and unstructured data and documents in a single semantic web. Once created you have at your disposal a highly usable engine that can perform unconstrained but comprehensive searches through the complex network of documents and data that represent the knowledge assets of your organization.

Compared to Sharepoint and similar systems the admin costs are very low indeed. Compared to conventional relational databases the set-up costs are minuscule.

OneIS can be used or has been used in the following areas.

- Organizing the knowledge of a large medical practice.
- Providing information on consultant skills and project suitability for consulting firms.
- Managing curated collections of documents, artworks or museum pieces, thus helping to design exhibitions, educational programmes and organize research projects.
- Managing training course and associated materials and contacts.
- Flexible content management when data are unstructured, structured or a mixture of the two.
- Computer supported coöperative working; helping teams to share documents and search freely within them. We know of at least one geographically distributed company that runs its entire business using OneIS.
- Creating a sales contact management system that allows you to perform searches across multiple cross-reference links, such as all the companies a person has worked for.
- As a personalized address book with advanced search facilities running on a 3G mobile. As well as the searches that a database can do, you can also ask it questions such as: ‘Who’s got a London telephone number?’, ‘Who knows John Smith?’ or 'Who lives in a postal code area that starts PO1?’

OneIS is normally provided as a hosted service in the cloud with free regular upgrades. You can even access OneIS from a mobile phone or other mobile device.

OneIS has a notably friendly user interface both for searching and browsing and for application development; it is so easy that, after initial set-up, non-IT staff can happily extend and modify the system. Initial set-up involves building a simple business model of your organization and can usually be completed in a few days.

OneIS has comprehensive security features, ensuring that only authorized people can get at or modify the data or design. The granularity of permissions and ownership can be as coarse or fine grained as you like. Hosting is not outsourced and data are backed up nightly to a different data centre. Two-factor authentication tokens are available.

Case Study 1: Building a personal contact database

Database systems have been around almost since the invention of the modern programmable electronic computer in 1948 and certainly since the advent of the first commercial computer: the Lyons Electronic Office (LEO) in 1951. In those days programmers crafted their own data management systems relying on, apart from their own skills, nothing more than features built in to the operating system; later examples included the operating system keyed indexes in the Xerox Sigma machines. Later, database management systems emerged to help with productivity and standardization; at first these were hierarchical file management systems , soon to be followed by network databases and the relational systems which have come to dominate the market.

With modern XML schemata it almost seems that hierarchical file management has returned as a sort of Hegelian 'negation of the negation' in which the later are sublated (*aufgehoben*) in the former. Object databases, while not attending with great commercial success similarly sublated network databases, removed the need for queries to coded to navigate the internal structure of the data.

With relational database management systems a Rubicon was crossed: it became possible to show mathematically that all possible queries could be implemented against a relational database. However, to say to prove that it is possible is not to say that it is easy or in any way tractable. Russell and Whitehead (1927) showed in their Principia Mathematica that all Mathematics could be founded on Logic in this way, but proving that 1+1=2 takes very many pages of closely argued proof. It is the same with relational databases: When the problem is suitable they are a boon but otherwise they can stop the show. In addition to sheer tractability, relational databases can obstruct application development through their user interface, which is also designed for a certain class of relationally tractable applications.

Let me give an example Suppose you want to build a personal address book in a relational database. I will use Microsoft Access to illustrate, but first let's write a short requirements statement.

1. I want to be able to store all my friends, contacts, colleagues and relations along with organizations I might have dealings with or which are employers of my contacts..
2. I want then to be able to construct the following types of query:
   - List all the people I know with the letters BOTTOM in their name or address?
   - Who has a telephone number containing 161?
   - Who has a dialling code containing 161? (*Not* the same question as the previous one!)
   - Who lives in the Cardiff area? What are their dogs called?
   - Who has a Chinese telephone number.?
   - Who lives at the same address as Daisy Wheeler? Which of these are her relatives and when are their birthdays?
   - … and so on.
3. I don't want to have to type the same thing in twice – *ever*.

In accordance with good software development practice, we should now sketch out a data or type model.

We will need a type Person for people. Because of the requirement numbered 3 and the awkward fact that people will cohabit, there should be a type called Residence, consisting of an address and phone number. Oh dear, we also will need types for these too.

But Access comes equipped with a Contact Management template. Let's use that as a starting point at least. When you download it you will see that it only has one type: Contact. So I will be able to store people and do some – but not all – of my queries without resorting to coding or schema redesign. And for contacts that live at the same address I will have to enter that address for each one of them. Furthermore, and now we look at the supplied user interface, I can't just enter part of a field and get the records that match the partial data. OK, so let's put our own schema in. Here's what it might look like during development.

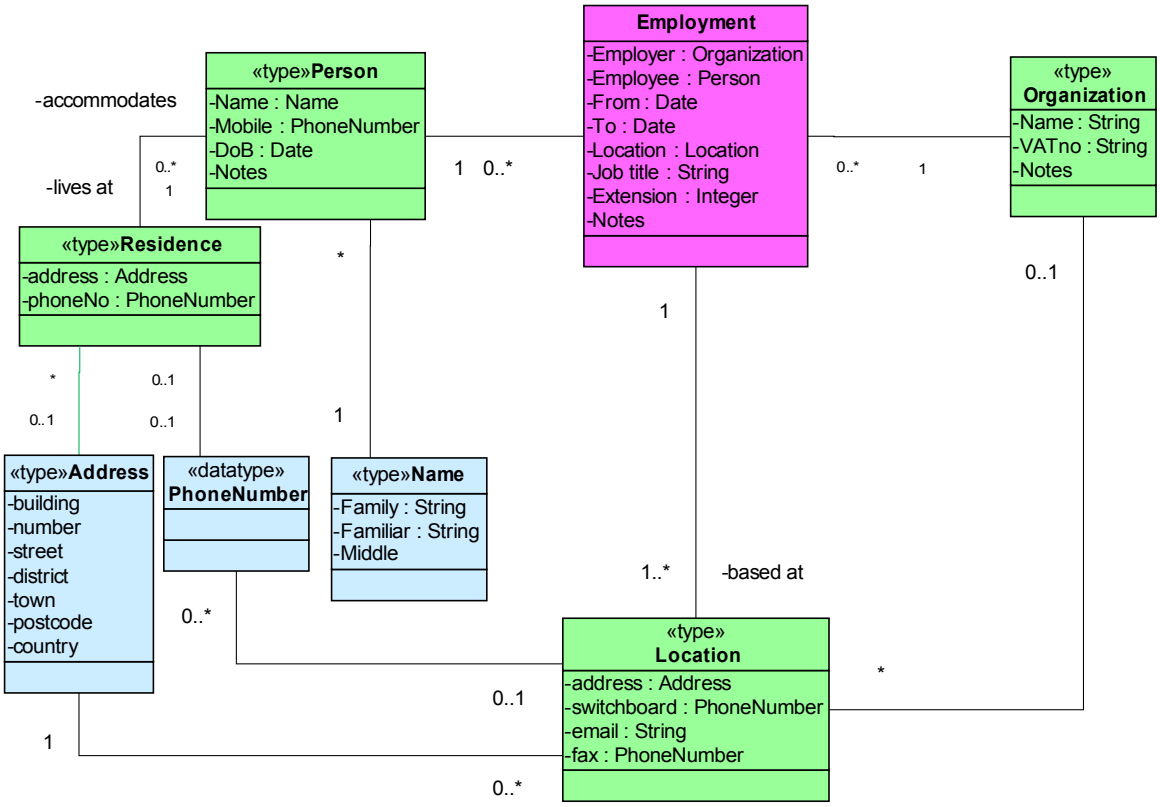

Figure 1

The colours follow the conventions of Peter Coad's pattern system (Coad *et al*., 1999): blue for descriptions or definitions, green for parties, places or things, and pink for temporally dependent episodes. Meeting requirement number 1 it fairly easy; all I need to to is set up CRUD-style queries and screens for people and organizations and cascade to the linked 'blue' types. Requirement 3 is trickier but you can just about do by creating tables as in Figure 1. The showstopper is requirement 2,

The present author spent well over two weeks trying to get Access to display a single screen with fields for all the data stored. What I then wanted to do was enter all or a fragment of a datum and have Access find a set of nearest matches that I could browse through. Of course, I expected to have to write a bit of simple Visual Basic to do this, but I never even got that far. The user interface just refused to let me out of its 'master and slave screens' paradigm. Round and round in circles I went and, eventually, gave up.

Some time later I discovered OneIS and realized that it might do the job. I obtained a license and had the application and up and running in about two hours flat – including reading the documentation! No coding was required. All that took any time was entering the data from my manual records. It was enough to convince me.

I was able to enter a part of a post code and see who lived in the area. On the same screen I could put in a fragment of a telephone number and match it to peoples names. Well, as long as I didn't start the number with a zero, since OneIS converts telephone numbers into international dialling format automatically no matter how they are entered, but once I knew about it, it wasn't an obstacle to getting the job done. I have been assured the supplier would expose the underlying search facilities for this to the user, if a client were ever troubled by it.

The multi-screen problem doesn't really arise with data entry. In that case the CRUD pattern works equally well in a relational system and in OneIS; you choose a table (or type) and enter data for instances of it. The problem arises, however, in spades when searching and browsing. In the relational world one is forced to start queries at one table and then navigate to related tables via foreign keys; in Access using a hierarchical cascade of screens.

The alternative is to construct a join of all tables, which is massively inefficient in terms of database speed and imposes an extra burden in terms of development effort. So it becomes very tricky to do what is so natural a requirement in applications like the address book: e.g. type in part of a telephone number and retrieve all the people or organizations that contain the string. And, incidentally, you get all the records that are linked to these people. In OneIS not only is this easy and natural but the user interface is already there and so you don't have the bother of defining queries and query screens.

The other point worth noting is that it was only when I started implementation in Access that I found it necessary to think about drawing a data model out formally; so much of the type model was already there in OneIS that it was easy for me to work from the rough and ready model that I had in my head. On the other hand this meant that the OneIS implementation didn't follow the formal model exactly: e.g. I didn't bother to create a type for the employment episode (moment-interval) on the first prototype. Another iteration was required to improve the OneIS model but at least I was up and running in no time at all.

The most interesting uses of OneIS occur when you add in other things, such as documents or projects, to the model. Then you can really get a complete view into what's going on in an organization.

Case Study 2: Sharing information internationally

An American corporation specializing in conversion maximization for multi-channel commerce, needed to launch its British subsidiary as a low-cost start-up. It appointed two executive directors to accomplish this. The Business Development Director was charged not only with approaching prospects but setting up the legal entity and defining office procedures. There was a need to coordinate closely with the US principles. Weekly telephone conferencing served some of this need but soon it became necessary to work collaboratively on documents and to share information about prospects and internal technical documents (some of which were highly sensitive commercially). The present author recommended a trial on OneIS to support this.

Within a few days and without any training beyond a demonstration of the system's capabilities, this director was able to create a system that allowed all parties to see all the organization's data, documents and notes of contacts. Any change or addition to the information store is logged automatically and users can check out documents such as letterheads, white papers or legal forms to work on their design or content. You can even create a simple intranet page.

The company now believes that using OneIS will make a significant contribution to the costs and the amount of work and time needed to set the company up and will be of lasting benefit to its operations as it grows.

Conclusions

Trireme is very impressed with OneIS. Our expertise is in requirements analysis and software and business modelling. Although OneIS is easy to use, you do need to analyze the data that your organization holds and the business processes that OneIS needs to support. We offer an agile business modelling service to get organizations started with OneIS, taking you through the organizational model and OneIS set-up and, in the unlikely event it is needed, user training. We can also, in collaboration with OneIS Ltd., help with data transfer from existing databases. OneIS also offer a customization service should your company require features or add-ons that are not part of the standard package. For further details see the OneIS website: http://www.oneis.co.uk/

To enquire about Trireme's requirements analysis, business modelling and data modelling services click here or call us on +44(0)1625-850839.

References

Coad, P., LeFebvre, E. and DeLuca, J. (1999) *Java Modeling in Color with UML*, Upper Saddle River NJ: Prentice Hall

Russell, B. and Whitehead, A.N. (1927) *Principia Mathematica,* Cambridge University Press