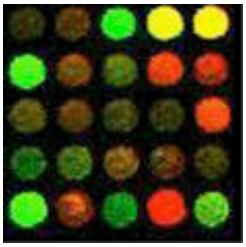

DATA-GRID

W3C Semantic Web in Life Sciences

— a position paper

Data Grid, Inc.
1413 Sandpiper Spit
Point Richmond, CA 94801

Main: (510) 233-1782

contact:
Tom Atwood (415) 732.6155
thomas.atwood@data-grid.com

A longer form of this position paper is available on line at

www.data-grid.com/standards/W3C_Position.pdf

[1] Data Management in Biotech/Pharma

Some of the data management needs of biotech firms and pharmaceutical companies are well served by commercial relational database management systems (RDBMS). Others — particularly modeling the complex structure and function of genes, proteins, macromolecular machines, gene expression regulation networks, and biological pathways — need DBMS with more semantic power. Data-Grid is building such a DBMS. We call it 'BioStore'. Its data model is based on OWL-DL.

DBMS requirements in Life Sciences research include:

1. New functionality, heretofore unavailable in commercial DBMS products
   - High level semantic model (first-order logic at a minimum) capable of capturing the semantics of the structure and function of biological entities and processes
   - Inference capability over these models
   - Support for probabilistic data and probabilistic schemas
   - Ability to handle a wide range of data type efficiently: objects, text, image data, 2D, 2.5D, 3D models, process models
   - Support for dynamic schema evolution as we understand the biology better
   - Collaboration among geographically distributed research teams, each with their own (slightly or dramatically) divergent database schemas
2. Unprecedented scale: terabytes of data; schemas of 40,000+ concepts
3. Very high performance: domain-specific search/match problems that require high-performance computing environments

'First order logic', 'ontology', 'inference', 'probabilistic data' — none of these terms occur in the user manuals for today's relational DBMS. Life Science needs a radically different DBMS — one explicitly designed to support research in a complex and rapidly evolving scientific domain, and one designed to run in the HPC environments characteristic of today's genomics, proteomic and systems biology labs.

DATA-GRID is building a DBMS that attempts to meet these requirements:

- Its data model is based on the description logic OWL
- It handles probabilistic data, and evolving partial data models
- It is built to support high-throughput computing characteristic of analytic work in the Life Sciences: grid clusters of hundreds of processors, and multi-terabyte storage fabrics.

Today: Home-brew Knowledge Representation Systems

Bioinformatics members of research teams often develop their own Knowledge Representation Systems (KRS). Commercial companies like Ingenuity, package the knowledge they sell into an in-house developed KRS.These KRS systems are broadly similar to the AI knowledge representation systems that were built at universities and Bell Labs in the mid' 90s — Classic, KL-One, Protégé. And they have some of the same problems — (i) similar but often slightly idiosyncratic semantics that make data interchange between systems difficult, and (ii) poor scaling. The scaling and stability

problems are often simply due to the fact that these systems are built by teams of 2-3 bioinformatics programmers as a stop-gap solution for a reseach project that may last only 12-18 months; they simply don't have the time or manpower to build a commercial-grade DBMS system.

Tomorrow: OWL-based KBMS

Data-Grid is in the process of building a Knowledge Base Management System (KBMS) that has the semantic power of a KRS, but the scale, performance and stability of a commercial DBMS product.

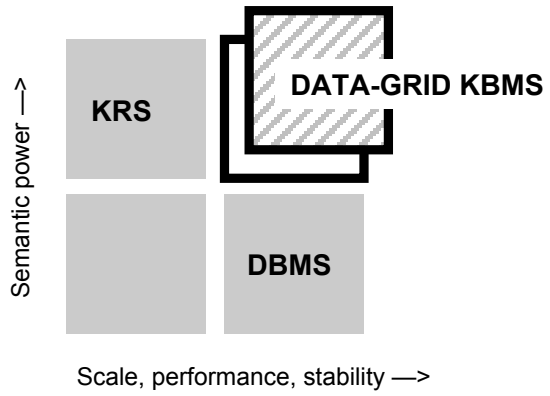

Figure 1. DATA-GRID KBMS — 'BioStore'

— Semantics

The 'frame'/'slot'/'facet' primitives of a typical KRS are representation-level constructs used to represent higher-level semantic concepts: classes, objects, attributes of objects, relationships between objects, subclass relationships between classes, etc. BioStore directly supports these higher-level concepts. It's data model is defined by OWL.

KSR primitive	used to represent	OWL primitive
frame	classes	class
	objects — instances of those classes	object — instance of a class
slot	properties of an object	property
literal-valued	attributes	data-valued (attributes)
frame-valued	relationships between objects	object-valued (relationships between objects)
facet	attributes of properties	attributes of properties, subclasses of relationships

Figure 2. OWL directly supports concepts KRS primitives used to represent

— Performance

Most DBMS and KBMS were designed in an era when main memories were small, the database was on disk, and the central design issue was minimizing the number of disk accesses. On today's clusters, this is no longer the issue. On a 100-node cluster with 4GB of memory per node, 2GB/node can be allocated to a cluster-wide distributed shared memory (DSM). That gives us a 200GB memory in which to cache the database. Most databases — even most databases in molecular biology — will fit comfortably

into that size cache. The central design problem in building high-performance database today has become maximizing locality of reference on the nodes of a cluster. BioStore has been explicitly designed to run on grid clusters. It maps the entire database into a cache-coherent shared memory which is distributed across processors in the cluster.

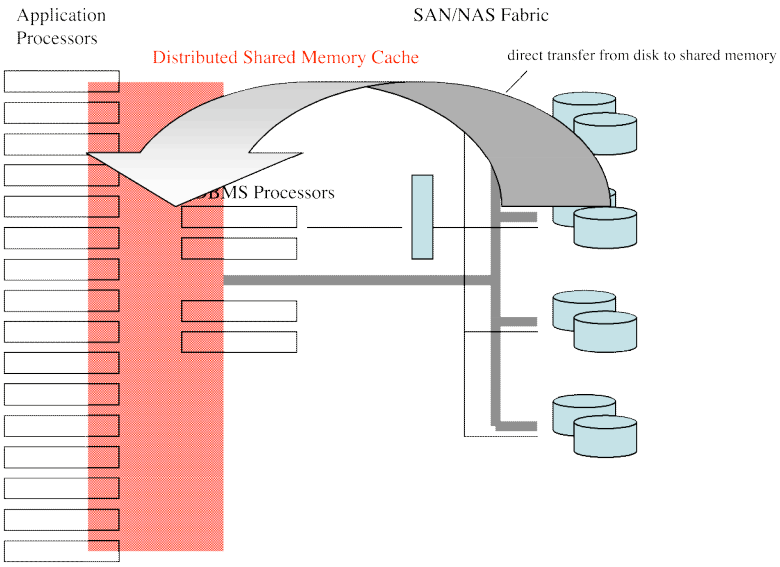

Figure 3. BioStore in a HPC cluster/SAN

[2] OWL as the ground for a DBMS semantic model

OWL was defined as an ontology representation language — a vehicle for describing a taxonomy of terminology within a particular domain of discourse. One use is as a markup language within documents published to the web. A second use is as a schema-definition language for a DBMS. BioStore uses OWL in this second way; it provides the semantic model for the DBMS.

In the balance of this section we outline a set of thoughts on (1) how OWL compares to the semantic models implicit in object programming lanagues and object DBMS; (2) what features that are high-value in a DBMS that have not been included in the current definition of OWL; (3) deeper seated differences.

OWL versus Object-Programming Languages and Object-DBMS

OWL is reasonably close to the type systems behind object programming languages such as Java and C#. This has the pragmatic advantage that an OWL DBMS can be accessed through an API similar to the JDO API now going through the Java Community Process. A noteworthy addition in OWL is support for subtypes of relationships as well as subtypes of objects.

OWL is also broadly similar to the type systems defined in Object DBMS schemas. These databases add notions of object lifetimes that may be longer than those of the process that created them, associative identification of objects of interest (a 'query

language'), and controlled sharing of the database in the face of concurrent access by multiple users or multiple threads operating on behalf of the same user.

Additional Features Suggested

There is a set of features in object database management systems and application modeling languages (UML) that experience has shown to be useful, but which are currently not represented in OWL. They include:

- Collections; at a minimum: set, list, bag, map.
- Keys, simple and composite
- Default values for instance properties
- Defeasible inheritance — exceptions.
- Methods: operations defined on object classes: name, and typed arguments, returns, exceptions.[1]

Default values and defeasible inheritance both step into the realm of non-monotonic logic and have been carefully kept out of description logics in order to keep inference tractable. That may be the wrong tradeoff for an OWL DBMS. One of the main purposes of inference in the DL literature is to deduce subtype relationships from the properties defined on classes. In most DBMS applications, the subtype relationships are specified by the schema designer; they do not need to be deduced; in fact, having the user explicitly draw the DAG representing the subtype lattice is one of the key ways of keeping the schema definition process understandable. Removing it is not a help.[2]

Conversely, the strict subtype notions built into many description logics (including OWL-DL), are an oversimplification of the type lattices that we extract from natural kinds. They may in fact work only for disciplines like mathematics. They certainly don't work for molecular biology. The GO schema almost doubles in size because of the inability to support exceptions. At each level there is a split between <normal X> and <exceptional X> . Object programming languages with a basic strict-subtyping model, always include a pragmatic way of handling exceptions, e.g., on Penguin you redefine the Fly operation inherited from Bird to be a no-op. Exceptions need to be given serious thought in subsequent versions of OWL.

Deeper differences

There two assumptions in OWL that seem rationally motivated for its use as a document markup language in the web, but are at odds with the assumptions traditionally made in database management systems:

- Open-world assumption
- Non-unique identity assumption

Three other deep issues bear discussion as well:

- Second-order versus first-order logic
- Extensional versus intentional logic
- Layering higher level logics on top of first-order-logic: probabilistic logic, context logic.

---

[1] For HPC environments, we also needs ways of declaring methods that can be run in parallel, and how they are to be synchronized.

[2] Subtype inference may be useful in schema design tools and in schema integration from different databases

We comment briefly on the second of these below.

— Extensional versus Intentional Semantics

The semantics of OWL is a classical Tarski-style extensional semantics. Two classes are the same if their extents are the same. This has some unfortunate consequences. To repeat an example from Lakoff[3], it means that the class Unicorn and the class Bald_Kings_of_France are the same class. The extent of both classes is the same — the null set. Somehow that seems wrong. Unicorns and Kings, bald or otherwise, just don't seem like objects of the same type. OWL currently makes a token gesture in the direction of keeping intentional and extensional semantics straight — it mentions the 'extent' of a class. However the current OWL specification, like many current object programming languages, uses a mixed notion called 'Class'. It does not distinguish Type (intension) versus Extent (extension) rigorously or consistently. This impacts complex class descriptions. Although the syntax of complex class definition is compositional, the semantics is not. Class expressions in the current definition of OWL can be used to define things which are not classes (in the sense of Types); they are better described as collections, e.g., the class description 'Astronauts AND Moon_rocks_we_are_taking_back'. The things in this collection are of interest because they all contribute to lift-off weight, but there is no type which is the union of Astronaut and Rock. Astronauts are People and people are animate; rocks are inanimate. Near the top of type hierarchy we probably have something that says that animate and inanimate objects form a disjoint partition of the extent of their supertype.

We suggest a small step in the direction of keeping intentional and extensional semantics clear: for each Class, make a default definition of its Extent which is the plural of the class name. So for the class 'Gene' we have the extent 'Genes'. Move what are now Boolean class expressions — unionOf, intersectionOf, complimentOf, — onto Extents. 'Class' will then be a clean intentional notion, 'extent' its extensional analog. I can now say that Unicorns and Kings are disjoint classes, but that their extents are the same, namely the null set.

[3] Why are we at the W3C Life Sciences / Semantic Web colloqium?

To Ask For Help.

We are going to put a lot of our investor's money ($30-40M) into building an OWL DBMS for Life Sciences. We'd like to get it right. We invite and ask for your help and collaboration. If I can be more overt: We would actively welcome teaming arrangements with companies or university research groups who are interested in becoming development partners.

[3] George Lakoff, Women Fire and Dangerous Things, University of Chicago Press, 1987