What VIVO Is and What It's Not

An expansion on the introductory talk from Cornell for VIVOweb Indiana F2F, 11/12/09, attached to this page IUIntro.2009-11-12.pptx.

Will VIVO be a System of Record (SOR)?

Vivo could be the sole source (and hence SOR) for some data provided by individuals (e.g., research keywords, short bio statements or research overviews). However, it makes more sense to leverage existing SORs and focus on VIVO's strength as a platform for integrating data and exposing on the web and for sharing with other institutional websites. Trying to harden VIVO so that it could store sensitive institutional data such as salary history would soak up huge amounts of developer time and contribute nothing to what NIH wants from the project.

Will VIVO be able to archive data?

This was the topic for an extended discussion the afternoon of 11/12 that branched off into an exploration of 4 different ways that data *could* be timestamped and/or time-based filtered views provided (rather than addressing whether this is a good idea or not).

Taking snapshots of the RDF database at specific intervals would satisfy the need for an archive, as would extracting VIVO pages as PDF documents to store in an institutional repository once a year. Building timestamps into the data itself requires changes to the way the data are modeled, typically by *reifying* each basic subject-predicate-object triple statement (adding statements about each statement specifying the source, time it was entered, etc.) This approach adds complexity that is likely not warranted, and we instead recommend that use cases be developed to identify where retaining information that is no longer current should be explicitly modeled.

The approach we recommend is to add **context nodes** to the model in much the same way that join tables hold intermediate information between 2 related primary tables in a relational database. One example where we know we need this is to model authorship of publications to retain author order.

Will VIVO hold private information?

Here again we recommend staying with simplicity rather than introducing new layers of complexity and questions of scalability. Marking each individual triple (subject-predicate-object statement) as public or private introduces the same reification requirement as adding a time stamp to it; an alternative would be to use a separate graph for private data from what is publicly visible, including the necessary interface controls to enable users to understand the difference (and implementation teams to set policies). Without strong use cases being voiced for maintaining personal information, we feel the compressed time line of the VIVOweb project compels us to avoid unnecessary new complexity.

There have been use cases described for enabling individual people to control whether certain publications are included in their VIVO web profile (especially for senior faculty with hundreds of publications) – so we do plan to support a data model and interface to allow limited control over the visibility of data on the VIVO website. We do not intend to support filtering data that is discoverable via RDF except in ways that apply across the whole model via the ontology (as a way to limit which data are harvested from an institutional VIVO to the national network.

Will VIVO log user entries?

In a limited way. We can add a log to the file system to indicate when any self-editor logs in, presumably to change his or her profile.

However, tracking the source and time of every statement inserted and deleted from the model will expand both the size and complexity of the database and software beyond what we feel is reasonable.

Will it be possible to track the provenance of statements?

The RDF additions and retractions loaded to VIVO through ingest procedures can at the option of local implementation teams be stored in a separate RDF database as a warehouse of data added and removed. If a question of the origin of any particular statement arises, it would be possible to run a query against this database to determine the batch of data a statement came from or that it did not come from batch input, so must have been added or removed interactively.

Where use cases in favor of or requiring provenance exist, we can also model the source of data explicitly, as has been done with data imported from Cornell's 3rd party faculty reporting system. See the page on Context nodes for more information.