Scaling the Tower of Babel: Data Integration and Warehousing with the WWW

Susan B. Davidson
University of Pennsylvania

The Web can be thought of as one large, disorganized database. Within it there are several related files (web pages), some of which are functions that take input from the user and provide another web page as output. For example, query interfaces to databases such as GenBank and SWISS-PROT can be thought of as such functions. Some of the web pages contain explicit pointers (hot links) to other web pages; other contain implicit pointers (for example, accession numbers) to components of other web pages.

However, the Web as a database is disorganized (no integrity constraints are enforced) and highly heterogeneous. Researchers wishing to query the web cannot do so easily, and two primary approaches have been taken within Bioinformatics to deal with this:

1) Developing federations of databases accessible over the Web, as with SRS.

2) Developing tools to allow database integration over (and often under) the Web,as with Kleisli/K2.

The second approach can either be used in a dynamic sense, in which queries are executed on-the-fly agains the underslying databases, or in a static sense by creating data warehouses. Warehousing has the benefit of allowing cleansing to occur (enforcing certain forms of integrity constraints), as well as the addition of annotation and new information.

Creating and maintaining these "value-added" warehouse databases raise a number of problems:

1. How can we detect when data in the underlying data sources has changed?

2. How can we automate the refresh process?

3. How can we track the origins or "provenance" of data?

Determining exactly what changes have occurred to the source databases is complicated by the fact that updates to biomedical databases are typically propagated in one of three ways:

1. Producing periodic new versions which can be uploaded by the user community;

2. Timestamping data entries so that users can infer what changes have occurred from the last time they accessed the data; and

3. Keeping a list of additions and corrections; each element of the list is a complete entry. The list of additions can be uploaded by the user community. None of these methods precisely describe the minimal changes that have been made to the data, resulting in potentially very expensive update techniques.

Part of the second problem, automating the refresh process, has received a lot of attention by the database community in the context of updating materialized views of relational databases. However, the results have not been applied to biomedical databases. The result is that the update process for many secondary databases rely on hand-written scripts, and are therefore quite expensive to write and are difficult to modify as the primary or secondary data source schemas evolve. This affects the periodicity with which refreshing occurs, since the need for information that is as current as possible must be balanced with the expense of keeping the view current.

The last problem, tracking the origin or provenance of data, is also very important. For example, suppose that one form of annotation in our warehouse is to assign function to sequences based on similarity (e.g. using BLAST

0-7803-6449-X/00/$10.00 © 2000 IEEE

searches). This annotation could then be transitively inherited by other sequences. If the original annotation is determined to be incorrect through experimentation, all subsequent annotations would also have to be undone. It is therefore important to track the origins of the annotation by keeping detailed information about what information the annotation was based on.

To realize the full potential of the WWW, we must have protocols, procedures, exchange formats, and tools to facilitate the above problems.