A Protocol for Maintaining Multidatabase Referential Integrity

Peter D. Karp
Artificial Intelligence Center
SRI International, EJ229
333 Ravenswood Ave.
Menlo Park, CA 94025
voice: 415-859-6375
fax: 415-859-3735
pkarp@ai.sri.com

The bioinformatics community is becoming increasingly reliant on the creation of links among biological databases (DBs) as a foundation for DB interoperability. For example, a link might be created from a protein in one DB (such as PIR), to a gene in another DB (such as GDB), by storing the unique identifier (id) of the gene object within an attribute of the protein object. User interfaces can then support navigation from the protein to the gene, and multiDB queries can join the protein with the gene. The unique id of the gene is serving as a foreign key. However, a variety of factors, such as changes in the underlying biology, can cause object ids to become invalid, thus producing invalid links among DBs. Invalid links are a violation of multidatabase referential integrity. We propose a network protocol whereby a database administrator can provide information about changes to the identifiers of objects in their database via Internet, to allow other databases to maintain referential integrity. We request comments from the bioinformatics community for the purpose of building a consensus on the proposed protocol.

1 Introduction

The bioinformatics community is becoming increasingly reliant on the creation of links among biological databases (DBs) as a foundation for DB interoperability. For example, a link might be created from a protein in one DB (such as PIR[a]), to a gene in another DB (such as GDB), by storing the unique identifier (id) of the gene object within an attribute of the protein object. User interfaces can then support navigation from the protein to the gene, and multiDB queries could join the protein with the gene. The unique id of the gene is serving as a foreign key. However, a variety of factors, such as changes in the underlying biology, can cause object ids to become invalid, thus producing invalid links among DBs. Invalid links are a violation of multidatabase referential integrity. We propose a network protocol whereby a database administrator can provide

[a] We use particular DB names to serve as examples only, without implying that the administrators of these DBs either support, or plan to implement, this protocol.

information about changes to the identifiers of objects in their database via Internet, to allow other databases to maintain referential integrity in an automated fashion. Automation is desirable given the tens of thousands of existing DB links.

We request comments from the bioinformatics community for the purpose of building a consensus on the proposed protocol.

We also argue that the bioinformatics community would benefit from an analog of the Internet RFC repository, to promote the promulgation of standards within the community.

2 Instability of Object Identifiers

Biological names are in constant flux, and therefore do not provide a stable, nor necessarily a unique, way of identifying an object in a particular DB. For example, Robbins presents examples of genes with identical names but different functions across different species[3]. Such names therefore do not provide a solid foundation for creating links among DBs. But we can associate a machine-generated identifier with a given object that can be guaranteed to be unique, and that can be more stable than biological names. But an assumption of complete stability — that the binding between an object and its single id will not change over time — is unrealistic. Both DBs and our knowledge of the underlying biology are in constant flux because of data entry errors, experimental errors, and revisions of theory. Instability in the underlying objects implies that the identifiers must also be unstable.

More precisely, object ids are unstable in the sense that we cannot guarantee that the mapping from objects to identifiers will remain a monotonically increasing, one-to-one mapping. Biology changes over time in a number of ways, and the DBs must change with it. In addition to discovering new objects,biologists also discover that previously known objects do not actually exist (e.g., they may have been experimental artifacts). They also discover that what had been thought to be two distinct objects are in fact one and the same.

As an example of the nonexistence case, computer sequence analysis may have predicted an ORF at a given location of a genome; that gene might be assigned a unique id. Later experiments might reveal that this reading frame never generates any protein product and was a figment of the computer's imagination. Therefore, this object should be deleted from the DB. Its id should be retired in the sense that any programs that later attempt to retrieve the object using that id should receive some type of error condition. In addition, the id should *never* be reused, that is, it should never be assigned to a newly discovered object.

As an example of object merging, consider a gene coding for a bifunctional

enzyme. Historically, different workers might have discovered different mutants of the gene that had different phenotypes, and only later found that the two genes map to the same location and are in fact one and the same. Consequently, we want only one gene, not two, in the DB. But since other DBs may use both of the old ids to reference the gene, we cannot invalidate either id. We must relax the requirements of one id for one object, because now we require two ids to map to the same object. (Again, neither id should be assigned to a new gene after the merge.)

Not only might objects merge over time, they might also split (or fork). In many cases, *E. coli* has two enzymes with the same function, coded for by two different genes. Historically, it often takes time for the separate identities of the two genes to emerge. Initially, the DB described only one gene. After the discovery that one DB entry must become two, which of the external references to Gene-1 should be changed to point to Gene-2?

3 Experimental Results

How significant is this problem in practice? The following three simple experiments address this question.

Bairoch's ENZYME [1] DB lists 2809 enzyme-catalyzed reactions that have been assigned EC numbers. The DB also contains records that describe object histories: it describes 173 EC numbers that have been deleted, and 127 EC numbers that were transferred (reclassified). These 300 changes provide a measure of the volatility of EC numbers that amounts to 11% of its current entries.

A second experiment examined the *E. coli* subset of SwissProt. SwissProt entries contain links to EMBL entries. I determined how many of those links are either completely invalid because the EMBL entry no longer exists, or are outdated because the EMBL entry forked or merged. For each of the 1578 unique EMBL accession numbers in this subset of SwissProt, I queried the NCBI Entrez[b] server by *primary* accession number. That is, the query succeeded only when I supplied a primary accession number; the query returned failure if the accession number is either invalid or is a secondary accession number due to merging or forking. 42 of the 1578 queries failed, a percentage of 2.6%. I then determined that 7 of the 42 accession numbers are invalid (.4%).

In a third experiment, I asked how the number of links from *E. coli* SwissProt[c] entries to *E. coli* genes in Rudd's EcoGene [4] changed from version 31 to version 32 of SwissProt. I found that although the number of links to EcoGene in-

[b] The Entrez server contains Genbank, which contains all EMBL entries.

[c] My intent is not to single out Bairoch's databases for inspection; I simply find them familiar and easy to work with.

creased from 2618 to 2692, 2 of the links present in version 31 were not present in version 32 — presumably because those EcoGene entries were retired, forked, or merged. This experiment suggests that if roughly .1% of the objects retire, merge, or fork in *each release* of a DB, that a significant number of bad links will accumulate in the DB over a period of, say, 5 years.

4 The Protocol

A protein DB would link to the preceding set of genes by associating gene object ids with its protein objects. As the gene DB evolves, more and more of the gene ids stored in the protein DB become invalid due to the forking, merging, and retirement of gene objects. Our proposed protocol assumes that the gene DB records information about object histories, e.g., when a gene object is retired, it is marked as such in the gene DB, and when two gene objects merge, the gene DB records that a merge relationship exists among the child object and its parents.[d] Our proposed protocol is designed to provide a standardized mechanism whereby the protein DB can query the gene DB about these changes in object histories.

The proposed protocol could be implemented as an application layer that sits above TCP/IP. In our example problem, the protocol would allow a client program written by the administrators of PIR to open an Internet connection to a server program written by the administrators of GDB. The client and the server would then carry on a well-defined dialog whereby the client is able to determine which of the links that PIR holds to GDB have become invalid, and how.

I envision that the protocol would be used in the following context. A PIR client program would periodically poll the GDB server to obtain news of object changes. That information would be forwarded to a human DB annotator for PIR, who would make the actual decision as to how to change the PIR entry. If, for example, the PIR entry pointed to Gene–1, and Gene–1 had forked to produce Gene–2 and Gene–3, the annotator would have to decide which of these two genes the PIR entry should now point to.

Herein I propose three alternative protocols. Each variation has different advantages and disadvantages, and it seems preferable to solicit feedback from the bioinformatics community before deciding whether to advocate one, two, or all three of them.

All three protocols would share a common initiation phase where the client opens a connection to the server, the server identifies itself, and the client selects a DB that the remainder of the conversation will refer to. An example

[d] Object histories are currently maintained for several biological DBs including GDB, PIR, Genbank, and the LabBase DB from the MIT Whitehead Genome Center.

conversation follows for illustrative purposes. Each line begins with the name of the party (client or server) who transmits it. In this example, a PIR client converses with a GDB server.

```
client:  HELLO foo.pir.org
server:  OK bar.gdb.org
client:  SELECT GDB
server:  OK GDB selected
```

4.1 Variation 1: Validation of Individual Identifiers

After the initial phase, this first and simplest variation of the protocol would allow the client to ask the server to validate a particular identifier, and to return additional information about invalid ids. The client might ask, "is P1283 a valid identifier?" The server might reply simply "yes" or "no." It could also reply with more information such as "no, the object was retired," or "no, the object merged with another object to yield an object whose id is P2911," or, "no, the object forked to yield two objects whose ids are P2888 and P2777," or "no, and the identifier never was valid." See [2] for a discussion of object forking and merging.

Examples:

```
client:  VALIDATE P1283
server:  FORKED P2888 P2777

client:  VALIDATE P234
server:  INVALID P234
```

4.2 Variation 2: Incremental Change Queries

Another approach is to allow the client to ask the server for information about what object changes have occurred since a given time. Possible queries by the client include: "Tell me all objects that have undergone *any* change since noon on Jan 1," and "Tell me all objects that were retired, merged, or forked since noon on Jan 1." Possible replies by the server include a list of ids of changed objects, or a list of ids plus the type of change, such as, "P213 was retired; P228 was merged with P312 to yield P456."

Example (note that "RFM" stands for Retired-Forked-Merged):

```
client:  RFM-CHANGES 1-JAN-95 12:00:00
server:  OK 3 changes
server:  FORKED P2888 P2777 P2999
server:  RETIRED P213
server:  MERGED P228 P312 P456
```

4.3 Variation 3: Subscriptions

The subscription variation would allow PIR, for example, to tell GDB which GDB objects PIR holds links to, in effect subscribing to those objects. At some time after changes to those subscribed GDB objects occur, GDB can tell PIR about those changes. PIR might also tell GDB what class of change it is interested in being informed of.

Example 1: Creating subscriptions initiated by PIR client

```
client:  SUBSCRIBE P1283
client:  SUBSCRIBE P1128
```

Example 2: Updates to subscriptions initiated by GDB client

```
client:  RFM-UPDATES 3
client:  FORKED P2888 P2777 P2999
client:  RETIRED P213
client:  MERGED P228 P312 P456
```

4.4 Comparison

These alternative protocols have different relative merits. Variation 1 is easy to implement on the server side because (depending on exactly how the protocol is defined) it requires the server only to either determine what ids are valid, or to maintain a record of object history relationships.

Variations 2 and 3 are more complex to implement. Both require the server to timestamp object changes; Variation 3 imposes the additional requirement that each DB must track subscriptions. The advantage of Variations 2 and 3 relative to Variation 1 is their incremental nature: they allow synchronization in time proportional to the number of objects changed, rather than the number of links held by a DB.

4.5 Realizing the Protocol

Several steps would facilitate the widespread implementation and use of this protocol within the bioinformatics community:

1. Submission of detailed comments on each alternative protocol from bioinformaticians around the world, to the author, to establish a consensus on which variation is preferable, and what modifications are required.

2. Development of a more detailed specification of the protocol, including error handling (due to network lossage for example), with another subsequent round of comments.

3. Implementation of a prototype *object news server* that both tracks object histories, and implements the proposed protocol (both client and server implementations could be provided). If the entire protocol could be implemented as two isolated components (client and server) that could be reused at a variety of sites, all DB administrators would not be forced to implement the protocol. The object news server could also track object histories, to avoid duplicating this functionality within every DB.

5 The Case for a BioRFC Repository

A significant reason for the success of the Internet is the Internet RFC repository. It holds a set of documents called the RFCs ("Request for Comments") that are used to disseminate information, and encourage dialog, about new Internet services. For example, the Internet file transfer protocol (FTP), and email transfer protocol (SMTP) were both defined and promulgated using RFCs.

A major hindrance to interoperability in bioinformatics is the lack of shared protocols whose specifications are publicly available. Therefore I advocate that the bioinformatics community establish its own repository of RFCs.

Once it is specified in more detail, the protocol defined herein would be a candidate for inclusion in the repository. Another candidate is a simple protocol for requesting a single object from a biological DB given the identifier of the object. The object could be returned in one of several possible encodings (the protocol might allow the client to specify the desired encoding). Such a protocol would be useful both for hypertext navigation, and for synchronizing replicated copies of DBs.

Acknowledgments

I thank Peter Li for commenting on earlier versions of this manuscript, and for suggesting the subscription mechanism. This work was supported by the National Library of Medicine under grant R29-LM-05413-01A1.

References

1. A. Bairoch. The ENZYME databank. *Nucleic Acids Research*, 22:3626–3627, 1994.
2. P. Karp. Models of identifiers. In *Proceedings of the Meeting on Interconnection of Molecular Biology Databases*, 1995. Available via World Wide Web URL http://www.ai.sri.com/~pkarp/mimbd.html.
3. R. Robbins. Comparative genomics: A new integrative biology. In J. Collado, editor, *Proceedings of the First International Workshop on Integrative Approaches to Molecular Biology*. in press.

4. K. E. Rudd, G. Bouffard, and W. Miller. Computer analysis of *E. coli* restriction maps. In K. E. Davies and S. M. Tilghman, editors, *Genome Analysis, Vol.4: Strategies for Physical Mapping*, pages 1–38. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1992.