MetaFam: a unified classification of protein families. II. Schema and query capabilities

Elizabeth Shoop*, Kevin A. T. Silverstein, James E. Johnson and Ernest F. Retzel*

Computational Biology Centers, Academic Health Center, University of Minnesota, Mayo Mail Code 43, 420 Delaware St SE, Minneapolis, MN 55455-0312, USA

Received on May 31, 2000; revised on September 22, 2000; accepted on November 23, 2000

ABSTRACT

Motivation: Protein sequence and family data is accumulating at such a rapid rate that state-of-the-art databases and interface tools are required to aid curators with their classifications. We have designed such a system, MetaFam, to facilitate the comparison and integration of public protein sequence and family data. This paper presents the global schema, integration issues, and query capabilities of MetaFam.

Results: MetaFam is an integrated data warehouse of information about protein families and their sequences. This data has been collected into a consistent global schema, and stored in an Oracle relational database. The warehouse implementation allows for quick removal of outdated data sets. In addition to the relational implementation of the primary schema, we have developed several derived tables that enable efficient access from data visualization and exploration tools. Through a series of straightforward SQL queries, we demonstrate the usefulness of this data warehouse for comparing protein family classifications and for functional assignment of new sequences.

Availability: Access to the MetaFam database is provided through a Java applet called MetaFamView, which can be run from the MetaFam web site at http://www.metafam.ahc.umn.edu/. Access to the relational data via named Oracle accounts can be arranged with the authors. Arrangements can also be made to obtain the data in Oracle 'export dump' format.

Contact: metafam@ahc.umn.edu

Supplementary information: The complete relational schema, integration scripts, and analysis queries are available from the authors.

INTRODUCTION

A very large amount of biological data exists in many separate databases and web sites throughout the world. In 1993, the DOE convened a panel of experts who recommended that integrating databases together was vital to the success of the Human Genome project (Robbins, 1993). Since that time, the data in separate biological databases has increased dramatically, and several efforts at integration have been investigated. The reported solutions fit into the following five categories.

Linked, indexed data. These systems connect flat file databases using World Wide Web (WWW) links and indexes. Examples include SRS (Etzold and Argos, 1993; Etzold and Verde, 1996; Etzold et al., 1996), DBGET/LinkDB (Fujibuchi et al., 1997), Entrez (Schuler et al., 1996; McEntyre, 1998) and the Molecular Information Agent (Gribskov, 1999). While not truly integrated systems, the advantage of these linked solutions is that they provide a single entry point of access for users, and they are relatively easy to implement. In practice, these systems have been extremely useful and popular with the user community, thus demonstrating the need for integrated data sources. The disadvantages of these systems are: (1) the indexes and WWW links require maintenance and are prone to errors; (2) browsing or keyword searches on indexes are the only available data analysis methods for users (no ad-hoc queries); and (3) they require a large amount of manual work by the user to integrate the data for further analysis (Davidson et al., 1995).

Loose integration with views on each data source. This approach organizes heterogeneous databases into a multidatabase system without a common schema, but with a common query mechanism. Examples include the CPL/Kleisli system (Davidson et al., 1997; Chung and Wong, 1999; Buneman et al., 1999) and TAMMIS (Baker et al., 1998, 1999). The advantages of these systems are: (1) they provide users with a single query interface; (2) they transparently and automatically access the underlying heterogeneous data sources; (3) the data sources are updated by each of their curators and the updates are obtained when queries are executed; and (4) they provide a single integrated result to users. The

To whom correspondence should be addressed.
disadvantage of these systems is that response times for executing distributed queries across the internet are slow and this doesn’t permit interactive use. Recently, it has been suggested that CPL/Kleisli could be used to materialize an integrated dataset (Chung and Wong, 1999). This would potentially alleviate this problem, but would add the need to keep the materialized data up to date.

**Tight integration with views.** This approach organizes heterogeneous databases into a database federation with a common schema and a central query mechanism. An example of this is OPM*QS (Chen and Markowitz, 1995; Markowitz et al., 1996; Chen et al., 1998). These systems share the advantages of loose integration systems. In addition, users have a single integrated schema representation of the data sources. The disadvantages of this type of system are the effort needed to create and maintain the global schema and the response time to execute queries.

**Loose integration with materialized data.** This approach organizes heterogeneous databases into a data warehouse without a common schema and requires all data to be periodically loaded into a central location. An example of this is GenoBase (Overbeek and Taylor, 1995). The advantages of this type of system are that little effort is required to integrate the schemas of the underlying databases, and query response time is faster with materialized data. The disadvantages are: (1) the only way to connect the heterogeneous sources are exact matches on identifiers shared by each data source; and (2) there must be a mechanism to keep the materialized data up to date. A disadvantage of GenoBase in particular was that it was built as a database of prolog clauses, which does not scale very well (Davidson et al., 1995).

**Data warehouse: tight integration with materialized data.** Construct a data warehouse with a common schema and periodical load all data into a central location. An example of this is Integrated Genomics Database, IGD (Ritter et al., 1994), which was implemented using AceDB. The advantages of this type of tight integration are: (1) a single global schema is used to combine data based on semantics, thus providing the richest integration possible; (2) a single interface is provided; and (3) data access to a single materialized database is faster than to several distant individual databases. The disadvantages of this type of system are: (1) there is extra work involved with creating and maintaining a global schema; and (2) the materialized data must be kept up to date. IGD in particular suffered from additional problems because of certain design decisions (Davidson et al., 1995). The very large datasets integrated into IGD led to significant performance problems, because of the database size limitations of the AceDB system. In addition, the original prototype for IGD did not contain clearly defined mechanisms for keeping the integrated data up to date (Ritter et al., 1994).

In this paper we present how we have used the data warehouse approach to construct an integrated database of protein family information. We favor this approach because of the added richness of queries that can be executed against this dataset, and because we can optimize the data warehouse for fast access, which is required by interactive data visualization tools. These two data exploration mechanisms, ad-hoc queries over integrated data and interactive visualization tools, will enhance the scientific discovery process (Friszman et al., 1998). Other added benefits of fully integrating multiple databases are that: (1) it provides a means of detecting anomalies and omissions in each of the underlying constituent databases; and (2) it provides a more complete set of information than can be found in each individual database.

Macaulay and colleagues suggested that a model system approach should be taken to investigate biological database integration (Macaulay et al., 1998). We have attempted such an approach to produce the MetaFam database, which is a data warehouse containing data from a tractable subset of all biological data, namely protein family databases and their related protein sequence information. MetaFam contains sequence data from SWISS-PROT and TrEMBL (Bairoch and Apweiler, 2000), PIR (Barker et al., 2000), GenPept (Benson et al., 2000), and NRL3D (Barker et al., 2000), and family data from Blocks+ (Henikoff et al., 2000), DOMO (Gracy and Argos, 1998a,b), Pfam (Bateman et al., 2000), PIR-ALN (Srinivasarao et al., 1999; Barker et al., 2000) (3 classifications), PRINTS (Attwood et al., 2000), PROSITE (Hofmann et al., 1999), ProDom (Corpet et al., 2000), PROTOMAP (Yona et al., 2000), SBASE (Murvai et al., 2000), and SYSTERS (Krause et al., 2000). By using a set of databases that contain information of similar content, we can more easily attempt a full integration solution than if we had tried to integrate biological databases of differing content. However, our integration of 12 protein family databases from ten separate sources and five protein databases from three different sources provides us with enough heterogeneity in data sources to encounter and solve several key issues in data integration.

MetaFam has many potential uses; the following are a few examples that we will discuss. First, the integrated data enables us to compare family constituency between family databases and detect anomalies in families of each of the databases. This combined resource thus enables curators of each database to improve their family classifications. Second, the combined family information can be used to match a set of hit protein sequences from a BLAST (Altschul et al., 1997) report with the protein family members from each different family database. This

MetaFam: schema and query capabilities
enables us to use family assignment information from multiple sources as a basis for assigning function to an unknown sequence that has been run through BLAST or some other similarity search program. Third, the relational implementation enables us to derive new data tables from the base information. This new data can be accessed quickly by an interactive data exploration tool that enables users to visually explore the correspondences between family classifications.

In this paper, we present the global schema for the MetaFam data warehouse and discuss its implementation in an Oracle relational database management system (DBMS). We then provide some example uses of the database using SQL queries. Finally, some key integration issues for data warehouses and our solutions to them will be discussed.

DATA WAREHOUSE SCHEMA AND DATA POPULATION

The global conceptual data schema for MetaFam is depicted in Figure 1. It can be logically divided into two major sections: entities associated with (1) protein sequences, shaded and shown on the right portion of the diagram, and (2) protein families, on the left. This conceptual schema was converted into a relational schema in an Oracle DBMS. In almost all cases, there is a one-to-one mapping between each entity and a table in the DBMS. The exceptions are the cases where an entity
contains an attribute for a protein sequence; these were placed in tables of their own for efficiency reasons. The next two subsections of this manuscript contain a description of some of the entities and attributes and what is populated in the underlying tables. Entity names are in bold face font in the text, and attribute names are in italics.

Non-redundant set of protein sequences

To compare the classifications of numerous protein family databases, where each has classified a different sequence source, it is necessary to create a new non-redundant set of sequences that includes all sequence sources used. In practice, this is difficult because not only are the protein sequence sources different, but two classifications may also include different releases from the same sequence source. Our solution is to periodically compile a non-redundant set of protein sequences, and to translate all sequences, old or new, used by the family databases into our own set of non-redundant keys.

Whenever any of the public sequence database sources (PubSeqDBSource) releases a new version of their sequence database (PubSeqDB), we compile a BLAST database (Blastdb) that includes all of these sources. This is used later for similarity searches using BLAST (Altschul et al., 1997). We then attach a version number (Blastdbversion) to this agglomerated set of sequences. Currently, we use the sources SWISS-PROT and TrEMBL (Bairoch and Apweiler, 2000), PIR (Barker et al., 2000), GenPept (Benson et al., 2000), and NRL3D (Barker et al., 2000). There may be repeats among the individual protein sequences (ProtSequence) from each of these sources. To remove this redundancy, we apply the program nrdb (http://www.blast.wustl.edu/pub/nrdb/) to identify exact duplicates, and a single non-redundant sequence key (NRSeqKey of NRSeq) is assigned for each set of identical sequences. We are able to keep track of sequences listed in old releases of each sequence database through the historical list of accession numbers (HistoricalAccession).

Collection of protein family databases

Periodically, we collect the latest releases (ProtFamilyDBSource) from a variety of protein family database sources (ProtFamilyDBSource) into a new set (ProtFamilyDBSet), and assign this collective set of family databases a unique identifier (ProtFamilyDBSetId). We did this so that we can keep information from more than one set of protein family data sources in our relational database at any one time.

Each family database will have many protein families (ProtFamily). Each of these families, in turn, will have many sequence/domain members, as originally defined (OriginalProtFamilyDomain). In order to compare the family sets among the various databases, we must translate the identifiers of their original members into our set of non-redundant keys. This is done internally using the ProtSeq-NRSeqPair table if a match is found, or by joining with the HistoricalAccession table when necessary. When all original identifiers are matched up to an NRSeq, the translated Domain members are stored in a separate table (Domain). One additional entity for each NRSeq in the ProtFamilyDBSet is the reference domain (ReferenceDomain), defined in the accompanying paper (Silverstein et al., 2001). The construction of the reference domain is dependent on how the sequence was divided into domains among all the family databases. This construct allows us to refer to a particular region on a sequence, and determine if that region has been classified by various databases. The Domain-ReferenceDomainPair constructs an occurrence of a defined Domain on an NRSeq within a ReferenceDomain. This can be used to determine how many databases have classified a domain on a particular region of a sequence.

The MetaFam algorithm, also described in the accompanying paper (Silverstein et al., 2001), collects together related families from each database into family supersets (SupersetCluster). Within each of these superset clusters, numerous pairwise correspondences (SupersetCorrespondence) between families may exist. The weight of each correspondence (SupersetClusterWeight) is simply the fraction of members in the intersection of the two families. These correspondences between family members enable us to derive every family (ProtFamilyId from ProtFamily) in each SupersetCluster and the description of that family (ProtFamilyDescription, which came from the original sources). From this information, we developed a method for defining a SupersetDescription for each SupersetCluster. This method, written as an SQL script, chooses a description from one of the constituent families. If a description exists for one or more curated families (i.e. a family from Blocks+, Pfam, PIR-ALN, PRINTS, PROSITE, or SBASE) in the superset, then the description is taken from the curated family with the largest number of members in the superset (i.e. is most representative). If only families from an automated database (e.g. DOMO, ProDom, PROTOMAP, or SYSTERS) are present in the superset, then the description is taken from the most representative family in that set. When a null description is found, the next-most representative family is used (giving preference to curated ones) until all possibilities are exhausted.

Data population

We have created automated scripts (written in Perl) to parse the data from each of the sequence and family sources, and to load this data into an Oracle database. The script which collects the family data also calls a C++ program to cluster related families together into supersets; see accompanying paper (Silverstein et al., 2001) for...
details. All of this data is used to populate the tables outlined in the primary data schema of Figure 1.

Since the data is updated frequently, proper versioning and data removal are important. Thus, we have designed our implementation of the schema to facilitate the removal of subsets of data. All tables that primarily pertain to protein sequences are tagged with a Blastdbversion identifier. Similarly all tables whose scope includes aspects of protein families are tagged with the appropriate ProtFamDBSetId. An SQL script has been written to remove all table entries with a specified tag. If resources permit, we will store each version on a separate database partition in the future.

ADDITIONAL DERIVED DATA IN WAREHOUSE

Although the primary schema encompasses the full range of data integrated from the underlying sources, in practice more tables are needed. Specifically, derived tables must be created for efficient real-time exploration of the data, and for complex statistical summaries. Without these derived tables, numerous tables would have to be dynamically joined together, causing significant delays for the end-user.

We have designed a set of derived tables, shown in Figure 2a, with the primary goal of minimizing access time for the user of our graphical visualization tool, called MetaFamView (available for public use at http://www.metafam.ahc.umn.edu/). Indexes are created on each table to speed data retrieval. Entities which are coordinated in a view are joined in advance and packaged together in a single table. Precisely which entities (and attributes of each) are combined depends on the logic of our interface. We have tried to restrict the scope of each query to include only those attributes required in a particular view. In certain situations, we collect information in advance that is likely to be viewed next. (For example, when viewing a superset we get the descriptions of all families in that superset, in case the user brushes over a family to view its description.) These general principles are further exemplified as we describe each of the view tables.

The first table in Figure 2a, FamilyNRDom, serves the vital purpose of coordinating nearly all related objects in MetaFam. This table links together each domain of every sequence with the families and supersets in which it is found. The MetaFam reference domain that corresponds to each domain is also linked by that table. This broad table allows the user of our visualization tool to simultaneously view a diverse set of data on a domain of interest quickly and efficiently.

Separate tables have been created to coordinate data for specific purposes. SupersetNRDom links together vital information about all of the domains in each superset. In this case, three items derived from different entities are grouped together: (1) the corresponding reference domain; (2) the number of families in that superset to which the domain belongs; and (3) the length of the protein on which that domain is defined. Similarly, SupersetFamily coordinates data on the families of each superset (i.e. the number of domains in each family and their descriptions). In some cases, it is desirable to get a smaller set of information on a specific family or non-redundant protein sequence. For example, FamilyDescription contains descriptions and some curation information for families, while NRSeqOfSupersetDesc and NRSeqOfSupersetAlias separately provide information complementary to that provided by SupersetNRDom about sequences in a superset (i.e. descriptions and cross-references among the public sequence databases, respectively).

In addition to tables designed explicitly for graphical navigation, a few tables have been created for comparative statistics. Two tables designed for this purpose are shown in Figure 2b. Each of these tables is the result of performing data analysis using reusable SQL scripts on the base data from the schema in Figure 1. The first,
SupClusRefDomScore, defines a net score composed of 'in' and 'out' scores for each domain in a superset. The scores reflect the number of families that have placed the domain in and out of the superset in question, and is weighted by the curation level of each family. The second table in Figure 2b assigns numerous quality measures to each superset. Individually, these assessment measures quantify attributes of a superset such as connectedness, consistency, and size. Both these assessment measures and scores are discussed in more detail in the accompanying paper (Silverstein et al., 2001).

**EXAMPLE USES**

Using a DBMS provides benefits over flat files for analysis of stored sequence and family data. The standard querying capabilities of the database allow us to generate new, valuable information without having to write additional programs. We can also combine data in very diverse ways to meet a variety of objectives. These general facts are illustrated with examples in this section.

In some situations, it is useful to compare the data only among two family databases. For example, in an attempt to expand their set of curated families, Pfam has been identifying families in ProDom with no counterpart in their own database (Bateman et al., 2000). To further expand their database, we recently provided researchers at Pfam with a full list of supersets that had DOMO families with no Pfam counterparts. This was accomplished by running the following straightforward SQL commands.

First, get each DOMO family that is in any superset:

```sql
create table tmpSCwithDomoFam as
select distinct SupersetClusterId, ProtFamDBId, ProtFamilyId
from SupersetFamily
where (ProtFamDBSetId = 5) and (ProtFamDBId = 'domo');
```

Then find each DOMO family that corresponds to a Pfam family in a superset:

```sql
create table tmpSCDomoPfam as
select *
from SupersetCorrespondence
where (ProtFamDBSetId = 5) and ((ProtFamilyDB1Id = 'domo' and ProtFamilyDB2Id = 'pfam')
or (ProtFamilyDB1Id = 'pfam' and ProtFamilyDB2Id = 'domo')) ;
```

Then use the minus operator to find each DOMO family that is in a superset but does not have a corresponding Pfam family.

```sql
create table tmpSCwithDomobutnotPfam as
select SupersetClusterId
from tmpSCwithDomoFam
minus
select SupersetClusterId
from tmpSCDomoPfam ;
```

Another useful comparison might be between same-source, different-version database pairs. When an automated database makes changes to their clustering procedure, we are able to identify the set of families which remained unchanged, those that were split up, and those that were re-combined. This would aid these researchers in evaluating their changes.

Comparisons need not be pairwise. We have devised additional queries capable of providing tailored lists aimed at each family database curator. These lists identify classified families that their database doesn’t have. Additionally the family supersets can be rank-ordered in terms of the qualitative features of the supersets (e.g. number of domain members, number of automated or curated databases, connectedness, or consistency of the superset). Given this data, a curator could focus on adding families that have been identified consistently by automated databases, but remain completely uncharacterized by curation efforts. An example of this is the following SQL code, which returns a set of families from the automated ProtoMap database that have no counterpart in other databases:

```sql
select SupersetClusterId, ProtFamilyId, DomainCount
from SupersetFamily
where ProtFamDBId = 'protomap' and ProtFamDBSetId = 6
and SupersetClusterId >
(select max(SupersetClusterId)
from SupersetCorrespondence
where ProtFamDBSetId = 6)
order by DomainCount desc ;
```

In this example, we take advantage of the fact that the **SupersetCorrespondence** table contains only supersets with multiple corresponding families. We also know that additional single families that had no correspondence to other families were added to the **SupersetFamily** table and **SupersetClusterId** values were assigned starting with a number one higher than the last superset with multiple corresponding families in it.

The value of the data we provide is not restricted to the curators of existing family databases. We use MetaFam to match hits from BLAST (Altschul et al., 1997) similarity searches against the non-redundant protein dataset to protein domains in particular supersets. An essential part of this analysis was to develop an additional Oracle function called MAXCOVERAGE, which returns the...
maximum coverage value between two regions on a sequence (see accompanying paper for a definition of maximum coverage; Silverstein et al., 2001). This function is as follows, using Oracle’s PL/SQL language, which is designed to enable developers to add extensions beyond what standard SQL can provide:

```sql
Function MAXCOVERAGE
  (begin1 IN number,
   end1 in number,
   begin2 in number,
   end2 in number)
RETURN number IS
  minRange number;
  len1 number;
  len2 number;
  unspecified number := -1;
BEGIN
  if ((begin1 = unspecified) OR
      (end1 = unspecified) OR
      (begin2 = unspecified) OR
      (end2 = unspecified)) then
    RETURN 1.0;
  end if;

  minRange := LEAST(end1, end2) -
             GREATEST(begin1, begin2) + 1;
  if (minRange < 0) then
    RETURN 0.0;
  else
    len1 := end1 - begin1 + 1;
    len2 := end2 - begin2 + 1;
    RETURN (minRange/(LEAST(len1, len2)));
  end if;
END; -- Function MAXCOVERAGE
```

We use this function when we want to check whether a hit region on a sequence from a BLAST report sufficiently covers a reference domain on a protein in a superset (see accompanying paper for an explanation of reference domains; Silverstein et al., 2001). Here is a small portion of the SQL code from a larger analysis script that accomplishes this task:

```sql
select A.PROJECTSEQID,
       A.HITID,
       A.NRSeqKey,
       A.ProtSequenceDescription,
       A.HIT_EVALUE,
       A.HITSEQSTARTPOS,
       A.HITSEQENDPOS,
       B.RefDomainId,
       B.RefDomainStart,
       B.RefDomainEnd,
       B.SupersetClusterId,
from BLASTALIGNMENT A, ReferenceDomain B
where A.NRSeqKey = B.NRSeqKey
  and MAXCOVERAGE(RefDomainStart,
                  RefDomainEnd,
                  HITSEQSTARTPOS,
                  HITSEQENDPOS) >= 0.48;
```

In this example, the BLASTALIGNMENT table contains several columns of information from a BLAST report, including the start and end of the region in the protein to which the input sequence was aligned. These are called HITSEQSTARTPOS and HITSEQENDPOS above.

### CONCLUSIONS

The key data integration issues that we have addressed with this work are the following.

**Global schema generation.** The data warehouse approach requires the development of a global schema for the stored data. This schema must be developed and kept updated over the lifespan of the warehouse data. We have presented the complete schema and discussed some of our design decisions. The time spent to model the tight integration, when done diligently, can mean that fewer changes are needed over the lifetime of the database. Over the course of generating six versions of MetaFam, we have yet to resort to a complete schema redesign, despite having added new family databases and removed a protein sequence database. While there was data modeling work involved in creating this schema, the payoff is that we have obtained a semantically richer integration than the loose integration approaches. For example, in addition to keeping information about each protein family from each separate source, we combine the information and devise supersets which contain correspondences between pairs of families from the underlying sources. A second example is the reference domain construct, which is derived by combining domain information from each of the constituent databases.

**Regular updates to underlying databases.** The curators of the protein sequence and protein family databases regularly update their collections. Any useful integration procedure must be able to keep up with these updates and perform re-integration at regular intervals. We have addressed this by marking every row in each table with either a protein family database set identifier or a protein sequence database set identifier. Each time the underlying databases are updated, we have a procedure for generating a new version and removing older versions.

**Heterogeneous file formats.** Each underlying database to be integrated has a separate file format. Data parsers that produce a consistent format that matches the global schema and that can be used for storing data in the warehouse must be developed for each database to be integrated. We have developed parsers and incorporated them into an automated procedure for generating the MetaFam database.
Differing scope of derived databases and redundancy in underlying data sources. The protein family classification databases are derived databases, in that their developers have attempted to structure the primary protein sequence data into groups of proteins with the same or similar function. One difficulty of integrating these derived databases is that each is based on different underlying sets of different releases of the protein databases. These underlying protein databases contain duplicated sequences with different identifiers assigned by each database. We have handled this issue by generating a non-redundant protein sequence dataset with a single key that has cross-references between same-sequence members of each primary protein database, including historical accession numbers. We map each protein in each derived family database to this non-redundant key, so that families can be compared between the derived databases.

Missing or conflicting textual descriptions. Each protein family database that is curated contains textual descriptions for their families of proteins. The content of these descriptions varies from database to database. The family databases which automatically produce families of proteins often have no descriptions at all. We developed a method for using the most appropriate of the available family descriptions to describe a MetaFam superset. This can then be assigned to a missing description from any of the superset’s families, thus increasing the information content of the automated databases.

Generating warehouse tables for visualization tool access. Building data warehouses suitable for visual exploration tools and knowledge discovery requires additional data preparation (Fayyad et al., 1996). This is a task that needs to be automated so that it can be executed each time a new re-integration is needed. We have devised a series of SQL scripts that automatically generate the prepared data for the MetaFamView tool each time the underlying data is updated.

We have provided some examples that illustrate the power of tight integration, using a relational database management system. With standard SQL queries and additional Oracle PL/SQL functions, we are capable of deriving data to meet most objectives involving protein sequence classification. The data warehouse framework allows us to bridge a broader set of data. The MetaFam data is now being used as part of an even larger data warehouse that includes sequence processing, contig building, and similarity results (DNA-to-protein and protein-to-protein). Thus, we have begun to expand the scope of our queries to include potential functional assignments. The intersection between the MetaFam portion of our data warehouse and the additional portions that contain similarity results is limited to one relationship between the NRSeq and a ‘hit’ from a similarity result. For this reason, the integration of these two portions has been quite easy. Thus, in practice we have found integration of other disparate sources of data to be a tractable exercise.

We have additional plans for MetaFam. We are currently participating in a structural genomics grant proposal to identify targets of novel folds. A first step might be to identify supersets which are consistent, but include no sequences with an alias to the structural Protein Data Bank (PDB). Since some distantly-related families share a common structural scaffold (Babbitt and Gerlt, 1997), this simplistic approach will incorrectly predict novel folds in some cases. To address this caveat, we intend to expand MetaFam to include structural classifications (Hubbard et al., 1999; Orengo et al., 1999; Holm and Sander, 1999; Sowdhamini et al., 1996) and a new automated approach which uses features of multiple-sequence alignments to identify potential distant structural relationships between families (Babbitt, 2000).

For those researchers who are interested, it is possible to set up a copy of MetaFam on a server of your own. The hardware and software requirements for an installation are as follows: a unix server with minimum 1 GB of memory (we use an E450 from Sun Microsystems); 5.5 GB of disk space for tables and indexes; Oracle 8.0.5 or higher.

ACKNOWLEDGEMENTS

The authors wish to thank the reviewers of this manuscript for very thoughtful attention and helpful suggestions for its improvement. The authors also thank Ihab Awad for suggestions which led to the improvement of the schema, Tim Kunau for extensive database administrative support, and Margaret Mayer for some work toward implementing the relational schema. This work was supported by an SCA with the USDA-ARS. Additionally, the authors are grateful for funding from several sources: The National Science Foundation (NSF) Functional Genomics Program for Soybean #9872565 and Genomics of Wood Formation in Loblolly Pine #9975806; the Department of Energy (DOE) Pine Gene Discovery Project, #DE-FC07 97ID13550; the United Soybean Board and the United States Department of Agriculture (USDA) Public EST Project for Soybean, #58-3625-8-117; and institutional support from the University of Minnesota Academic Health Center. ES was partly supported by funds from NSF award #DBI-9753283.

REFERENCES


Attwood,T.K., Croning,M.D.R., Flower,D.R., Lewis,A.P.,


E.Shoop et al.


