



Representing genetic sequence data for pharmacogenomics: an evolutionary approach using ontological and relational models

Daniel L. Rubin, Farhad Shafa, Diane E. Oliver, Micheal Hewett and Russ B. Altman

Department of Genetics, Stanford Medical Informatics, MSOB X-215, Stanford, CA 94305-5479, USA

Received on January 24, 2002; revised and accepted on March 31, 2002

ABSTRACT

Motivation: The information model chosen to store biological data affects the types of queries possible, database performance, and difficulty in updating that information model. Genetic sequence data for pharmacogenetics studies can be complex, and the best information model to use may change over time. As experimental and analytical methods change, and as biological knowledge advances, the data storage requirements and types of queries needed may also change.

Results: We developed a model for genetic sequence and polymorphism data, and used XML Schema to specify the elements and attributes required for this model. We implemented this model as an ontology in a frame-based representation and as a relational model in a database system. We collected genetic data from two pharmacogenetics resequencing studies, and formulated queries useful for analysing these data. We compared the ontology and relational models in terms of query complexity, performance, and difficulty in changing the information model. Our results demonstrate benefits of evolving the schema for storing pharmacogenetics data: ontologies perform well in early design stages as the information model changes rapidly and simplify query formulation, while relational models offer improved query speed once the information model and types of queries needed stabilize.

Availability: Our ontology and relational models are available at <http://smi-web.stanford.edu/projects/helix/pubs/ismb02/>.

Contact: rubin@smi.stanford.edu;
russ.altman@stanford.edu; help@pharmgkb.org

Keywords: ontologies; relational databases; schema; data models; pharmacogenomics.

INTRODUCTION

Characterizing the variation in genome sequences is essential for understanding the genetic basis for disease, variation in drug response, and the virulence of pathogens. In particular, an increasing number of studies are being undertaken in pharmacogenetics—a discipline that seeks to understand how inherited differences in genetic sequences among people influence their response to drugs (Krynetski and Evans, 1999; Roses, 2000; Weinshilboum *et al.*, 1999). Since genome polymorphism detection efforts are producing an explosion of biological sequence information, effective storage and retrieval of genetic sequence data are critical for recognizing important variations in these sequences.

Many databases storing sequence data have been developed (Bairoch and Apweiler, 2000; Benson *et al.*, 2000; Boguski *et al.*, 1993; Letovsky *et al.*, 1998; Sherry *et al.*, 2001; Skupski *et al.*, 1999). The goal of most of these databases is to provide archival storage and retrieval of the data, with predefined analytical and query capabilities (e.g. BLAST searches) that were anticipated when the schema for the database was designed. Because the queries and analytical functionality these databases support has changed little, the database schema has remained relatively stable. Schema changes are therefore usually infrequent, and minor.

We are building the PharmGKB (<http://www.pharmgkb.org/>), a storage and retrieval system of pharmacogenetic data designed to contain genetic sequences, cellular and molecular phenotype, and clinical data (Hewett *et al.*, 2002; Klein *et al.*, 2001). One of the goals of this resource is to provide analytical functionality to connect genotype with phenotype. To accomplish this, we must provide high resolution queries that interconnect and inter-relate particular fields or parts of fields in the database. Such queries could be very difficult to accomplish unless the database schema provides sufficient detail with respect to

*To whom correspondence should be addressed.

the granularity of the data as well as all the relationships among the data entities.

Storing genetic and polymorphism data for pharmacogenomics studies is challenging because the information model needs to capture many distinctions, and new attributes or new concepts may arise that require the information model to be changed. Unlike a database such as Genbank (Benson *et al.*, 2000), where the central information object is the sequence, in a pharmacogenomics database, there are many information objects, such as genes, sequences, regions of interest within sequences, polymorphisms, primers, and sequencing assays. The range of queries and types of analyses that a user may desire are difficult to anticipate, and they may change over time. Therefore, the pharmacogenetics database schema needs to change as the database and its user community develops.

Information models differ in their suitability for rapid change and in terms of database performance. Ontologies in frame-based systems (also called knowledge representation systems) may be advantageous for rapidly evolving information models (Karp and Paley, 1995, 1996; Musen, 1999), since they map closely to an object oriented view of the world, where data about individual objects are stored in a manner that can be termed 'local.' Ontologies are also useful for supporting intelligent retrieval and summary of the data (Altman *et al.*, 1999; Hafner *et al.*, 1994) and for inferring new information (Noy and Hafner, 2000). Unfortunately, existing knowledge representation systems do not provide high speed access to large, complex knowledge bases (Karp and Paley, 1995).

Relational database systems[†] have been optimized for performance, and have been widely accepted in many production systems that work with very large data repositories. However, in complex domains such as pharmacogenomics, the database schema can be very complex, especially if there are many entities and relationships in the information model (and thus many tables containing information about single concepts, leading to more 'distributed' storage). It can be difficult to maintain the database schema if the information model needs frequent or extensive revisions.

Given these differences between ontologies and relational databases, it is not clear which is preferred for storing pharmacogenetics information. Pharmacogenetics data can be voluminous, and fast database performance afforded by a relational database is important for serving the needs of many simultaneous users searching across large amounts of genetic data. But pharmacogenetics data are also complex and changing, so an ontology model may be preferable. A study comparing both approaches is needed.

[†] When referring to 'relational databases' we exclude object relational and object oriented databases because they are not in common use.

Our objective was to develop a high resolution information model for genetic sequence and polymorphism data for pharmacogenetics studies, and to evaluate implementations in a frame-based system and in a relational database system. The goal of this study was to understand the suitability and limitations of each implementation in the context of building a database for pharmacogenomics. Such information could be helpful for deciding between frame-based and relational representations for other database implementations or for considering an evolution between these approaches.

SYSTEM AND METHODS

Information model

We needed a model for genetic sequence data suitable to represent the features in genetic data pertaining to pharmacogenomics and capable of storing data submitted by multiple study centers. Since each study center stores their data differently, and they have their own terminologies, it was important to develop a common model (the 'information model'). The information model for genetic sequence data details the information objects ('entities'), information specific to them ('attributes'), and relationships among the entities ('relationships'). An example entity is *Reference Sequence*, having an attribute 'sequence' containing the string of letters making up the sequence, and a relationship to *Gene*, representing the gene from which the sequence was derived. We developed our information model in consultation with three different study centers that perform resequencing studies to identify the commonalities in their data. The goal was to be able to represent genomic DNA sequences, as well as transcribed and translated sequences along with arbitrary annotations on these sequences, including all common modes of variation (and most uncommon ones too).

Our model for genetic sequence data for pharmacogenomics includes the following entities (Figure 1): genes, reference sequences, sequence coordinate systems, regions of interest, PCR assays, methods, variant discoveries, variants in individuals, populations, and subject variants. A *Gene* is a fundamental concept that describes information about genes, such as HGNC name, symbol, Locus Link ID, Genbank and RefSeq accessions, cytogenetic location, and gene product. The *Reference Sequence* is the sequence that is designated to be compared with other sequences. It has a relationship to *Gene*, indicating the gene for the sequence.

A *Sequence Coordinate System* is an entity used to map from one counting system to another for labelling base positions in sequence. Not all laboratories label their sequences using a coordinate system where the first base in the sequence is called '1.' The *Sequence Coordinate System* has one attribute giving a number

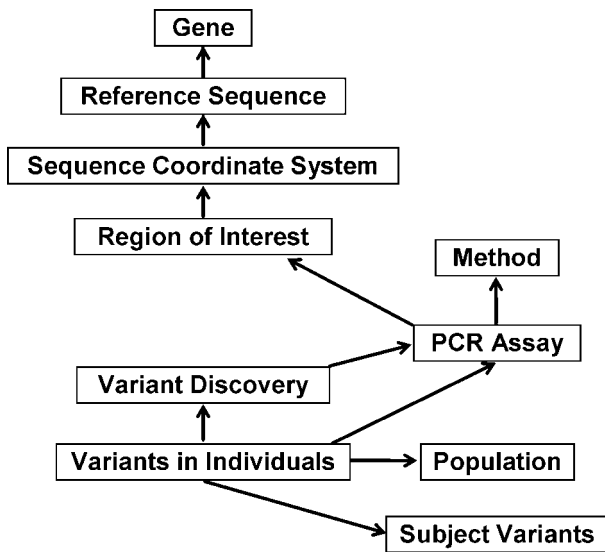


Fig. 1. Information model for genetic sequence data showing the information entities (in boxes) and relationships among those entities (arrows). The data-containing attributes for each entity are not shown (a full list of attributes is at <http://smi-web.stanford.edu/projects/helix/pubs/ismb02/schemas/SequenceData.xsd>).

in one counting system where the first base is '1,' and a second attribute storing the corresponding number in the other counting system. For example, if a laboratory wishes to start counting their sequences from '+100,' the *Sequence Coordinate System* entity would contain '1' and '+100.' Since a *Sequence Coordinate System* applies to a sequence, it has a relationship with *Reference Sequence*.

A *Region of Interest* is a substring of a sequence that is meaningful to the investigator. For example, it could be an exon, a promoter region, or an amplicon. The *Region of Interest* has two attributes storing the start and end positions in the sequence demarcating the region, counted in the coordinate system used for a sequence; thus, *Region of Interest* refers to a *Reference Sequence* through a *Sequence Coordinate System*.

A *PCR Assay* describes a PCR sequencing reaction. It has a *Method* that describes details of the assay method. It has additional attributes for the primers and it relates to a particular *Region of Interest* that is being sequenced. There are also attributes specifying the range of bases that were read with confidence in the assay (the 'interrogated range'), which is necessary since bases near the ends of the primers are often difficult to read accurately.

Laboratories often sequence regions that were previously known to have single nucleotide polymorphisms (SNP), also called 'variants.' Thus, we have an entity, *Variant Discovery* that represents these previously reported variants. This entity has attributes describing the

variant, including the *PCR Assay* used, the variant discovered, number of chromosomes used, and the literature citation.

The results of resequencing a group of study subjects is stored by the entity, *Variants in Individuals*. This entity refers to a *Population* that these individuals are from (e.g. the Coriell Anonymous 450 population), the *PCR Assay* that was used for sequencing these individuals, a *Variant Discovery* (if this study is resequencing a previously-reported SNP), and a set of *Subject Variants* that represent the variants observed in each individual. *Subject Variants* has attributes recording the subject identifier, the position of the SNP, and the variant observed at that position.

Representing entities and attributes in XML schema

We wanted to separate our information model for genetic sequence and polymorphism data from the actual database schema used to implement that model. This would allow us, for example, to switch from a frame-based schema to a relational schema without requiring users who interact with PharmGKB to change their software. Thus, we needed a representation of our information model of genetic data that was independent of a particular database schema.

Extensible Markup Language (XML) (Bray *et al.*, 1998) is useful for exchanging data between resources (Achard *et al.*, 2001; Tarczy-Hornoch *et al.*, 2000; Xie *et al.*, 2000) because it is extensible, readable by humans, unambiguously parsed by computers, and can be formally defined using XML schema. XML schema (W3C, 2001) is a language that specifies the structure and the data type of each element and attribute in an XML document. XML schemas are written in XML, and thus are self-describing. XML schemas are also extensible, permitting authors to develop customized constraints.

We used XML schema to represent our information model: to describe the entities and attributes in the model for genetic sequence data. The XML schema also served to specify a common XML file format for receiving data submitted to PharmGKB: it describes the XML elements and value types that are necessary for making a valid submission (Figure 2), and it serves as an interface between the internal storage format for data in PharmGKB and the formats used by various study centers.

The XML schema for a portion of the information model shown in Figure 1, *Variants in Individuals*, is shown in Figure 2 (the complete XML schema is available at <http://smi-web.stanford.edu/projects/helix/pubs/ismb02/schemas>). The *type* attribute specifies the data type of the XML element. For example, the *Population* element has 'type = xsd:string' meaning that this element contains a string. The cardinality of elements is specified by *minOccurs* and *maxOccurs*; for example, *Display-*

```

<xsd:element name="Variants_In_Individuals">
  <xsd:complexType>
    <xsd:sequence>
      <!-- A unique name for submitting this individual variant. -->
      <xsd:element name="Display Name" type="NonblankString" minOccurs="1" maxOccurs="1"/>
      <!-- A literature citation reporting this individual variant. -->
      <xsd:element name="Citation" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
      <!-- Refers to a patient population -->
      <xsd:element name="Population" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <!-- The previously-discovered variant that is being sequenced in this re-sequencing study. -->
      <xsd:element name="VariantDiscovery" type="NonblankString" minOccurs="1" maxOccurs="1"/>
      <!-- Refers to the assay used for detecting the polymorphisms in this individual. -->
      <xsd:element name="PCRAssay" type="NonblankString" minOccurs="1" maxOccurs="1"/>
      <!-- The variants found in this subject. -->
      <xsd:element ref="SubjectVariants" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

Fig. 2. XML schema for one class of information (*Variants in Individuals*) from the information model in Figure 1. The sub-elements specify attributes in the corresponding class; some of these attributes contain references to other data entities (bold), while others contain simple data values (italics). For example, filling in a value for the ‘Population’ and ‘PCRAssay’ attributes associates that individual with a population and specifies the assay used on that individual.

Name is single-cardinality and required, *Population* is single-cardinality and optional, and *SubjectVariants* is multiple-cardinality and optional.

We designed the XML schema to match our information model in terms of entities and attributes. Thus, the information model shown in Figure 1 has a nearly one-to-correspondence with the XML schema. The outer-most XML elements correspond to entities in our information model (*Variants in Individuals* and *Subject Variants* in Figure 1, for example), while the first level of nested XML elements correspond to attributes for the corresponding entities. This allowed investigators submitting data to PharmGKB to map their data to our model, and it provided a specification for us to translate the information model into an ontology and relational schema.

Implementing the information model in ontology and relational schemas

The information model of genetic data in Figure 1 was implemented in a frame-based system and in a relational database system. We used the XML schema version of the information model as a specification of the entities, attributes, attribute cardinality, and data types in the implementing schema. In order to have a similar starting point for comparing frame-based and relational systems, we designed the schema for the frame-based and relational system to closely match the XML schema.

To create the frame-based implementation, we used the Protégé suite of tools (Musen *et al.*, 2001; Noy *et al.*, 2000) to build an ontology (classes and slots) based on our information model. In general, entities in the

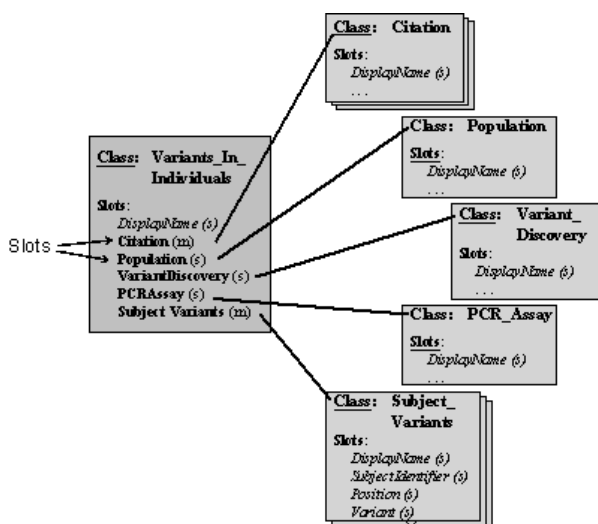


Fig. 3. This figure shows a portion of the ontology model for genetic sequence information (dealing with variants in individuals) and its relationships with other classes in the ontology. Values associated with instances of each class are stored in slots; some slots contain simple types such as strings (italics), while other contain references to other instances in the ontology (bold). Each slot can be single (s) or multiple cardinality (m). Not all slots on the related classes are shown. Notice the similarity between this figure and Figure 1.

information model were represented as classes in the ontology, while attributes were represented as slots in the ontology (Figure 3). The value type and cardinality for each slot in the ontology were specified as facets on the slots. Relationships among the entities were modelled

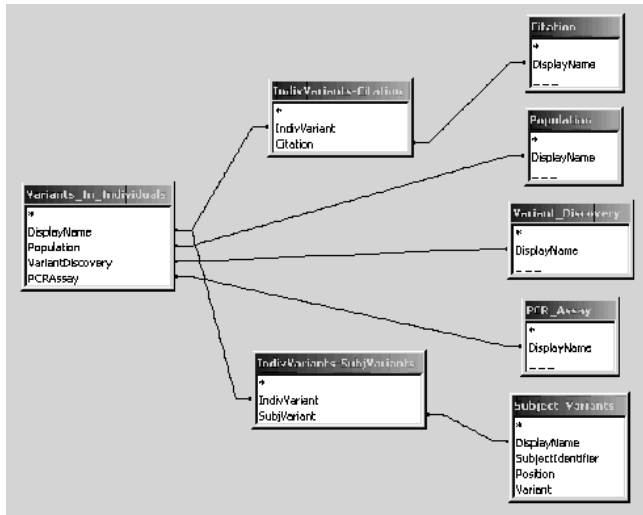


Fig. 4. Relational model for the same portion of the information model shown in Figure 3. The foreign key links between tables are shown as lines connecting the respective tables. In those cases where an attribute has single cardinality, there is a single reference to another table (e.g. Variants_In_Individuals and Population). When an attribute is multiple cardinality, an additional associative table intervenes between the related tables (e.g. Variants_In_Individuals and Citation).

as slots whose values are pointers to instances of the appropriate class. For example, in Figure 1, *Variants in Individuals* relate to a set of *Subject Variants*. In the ontology (Figure 3), there is a slot, ‘Subject Variants’ that contains references to zero or more instances of the class ‘Subject_Variants.’ Because of the object-oriented nature of the information model, the ontology design could closely parallel the information model (compare Figure 1 and Figure 3).

As with the frame-based implementation, we developed a relational schema directly from the XML schema of our information model (Figure 4). Entities in the information model were translated into tables, and single-cardinality attributes became columns in the respective tables. The data type for each column was determined by the data type specified by the XML schema for the corresponding attribute. For multiple-cardinality attributes, an additional table was needed to store the multiple values. For example, *Variants in Individuals* can have multiple *Subject Variants* (Figures 1 and 2). The latter multiple-cardinality attribute is represented in the relational schema as the table ‘IndivVariants-SubjVariants’ that associates *Variants in Individuals* with multiple *Subject Variants*. Similarly, *Variants in Individuals* is associated with multiple *Citations* through the IndivVariants-Citation table. In general, the relational schema has more complexity than the in-

Table 1. Execution time (in seconds) of test queries issued to frame based (‘ontology’) and relational database (‘relational’) implementations of the genetic sequence information model shown in Figure 1. Both implementations were tested on the same machine with the same data, and the queries were done sequentially. For the frame based system, 17 668 total variants needed to be processed in order to answer queries 3 and 4. For the relational system, 8 different tables needed to be joined to answer queries 4 and 5

Execution Times of Test Queries		Timing for Query	
		(seconds)	
Query		Ontology	Relational
		1	How many regions of interest are in the MDR1 gene?
2	List all regions of interest and start/stop positions relative to the reference sequence	0.8	0.6
3	For each variant, what is the base at that same position in the reference sequence?	180	8.4
4	Which subject has the most variants?	75	4.1
5	For each subject, find all the variants	0.6	0.6

formation model (compare Figures 1 and 4). Note that it would be possible to reduce the number of tables in the relational schema and improve its storage efficiency, but this would require making it deviate from the XML schema; our design choice in this study was to make the implementing schema similar to the XML schema.

The frame-based and relational systems were implemented on the same machine (Sun SPARC running Solaris 5.6). Protégé provides a database back end to save the ontology and instances in a relational database. We used Oracle 8.1.7 as the data repository for both the frame-based system and for the relational implementation.

Performance evaluation

We obtained two complete data sets containing results from resequencing genes in study subjects from one of the study centers (MDR1 and OCT2 from investigators at UCSF). We created a set of queries potentially useful for analysing the genetic sequence data and polymorphisms (Table 1). These queries were implemented as a set of SQL queries for the relational implementation of the database (Table 2). To query the frame-based implementation, we wrote procedures in Java (Table 2) using Protégé’s programming API (Noy *et al.*, 2000), which provides a view into the ontology conforming to the Open Knowledge-Base Connectivity (OKBC) protocol (Chaudhri *et al.*, 1998). The API provides methods such as ‘getCls’, ‘getInstances’ and ‘getOwnSlotValue’ for retrieving a class in the ontology, instances of a class, or the slot value of an instance, respectively. The time taken to execute these queries in the frame-based and relational systems was measured.

Table 2. Queries to relational and frame-based systems for query 5 in Table 1 (find all variants for each subject). The query for the frame-based system is in pseudo-code (actual java code implementing OKBC calls is available at <http://smi-web.stanford.edu/projects/helix/pubs/ismb02/>)

Queries To Relational And Frame-Based Systems

Query to Relational Database: (SQL)

```
SELECT t0.displayname, t7.precedingvarpos+1,t7.variant, substr(t1.sequence,t7.precedingvarpos+1,1),
t7.subjectident
FROM genesubmission t0,refseqsubmission t1, seqcoordsubmission t2, expregionsubmission t3,
pcrassaysubmission t4, indivsndsubmission t5, indivsndvariant t6, subjectvariant t7
WHERE t0.displayname = t1.gene AND t1.displayname = t2.refseq AND t2.displayname = t3.seqcoord
AND t3.displayname=t4.expreion AND t4.displayname = t5.sndassay AND t5.displayname = t6.indivsnd
AND t6.subvariant = t7.displayname AND NOT (substr(t7.variant,1,1) =
substr(t1.sequence,t7.precedingvarpos+1,1) AND substr(t7.variant,3,1) =
substr(t1.sequence,t7.precedingvarpos+1,1))
```

Query to Frame-Based System (pseudocode of OKBC calls in Java)

```
Get all instances of Subject Variants class
Create a hash table
for each instance of Subject Variants:
    subjID = value of the SubjectIdentifier slot
    variant = value of the variant slot
    if subjID is in hash table, add variant to list of variants
    else create new list for subjIDe
print hash table
```

We qualitatively evaluated the schemas of the frame-based system and the relational database in terms of complexity and noted potential difficulties maintaining each if the XML schema changed, requiring the database schema to be updated.

RESULTS

Evaluation of the information model

Our information model was reviewed by five different study centers that collect pharmacogenetics data. We found that our model captures the data at a level of detail necessary to store their data. We have tested the information model in terms of its ability to represent data from pharmacogenetics sequencing studies performed at three different study centers. These study centers mapped the XML schema of our information model to their internal database schemas and successfully produced valid XML submissions of their data. They are currently creating programs to create these files automatically as they collect new data.

One of the study centers submitted complete data sets from two genes, resulting in the following data that was stored in both the frame-based and relational systems: 39 *Reference Sequences*, 39 *Sequence Coordinate Systems*, 2 *Regions of Interest*, 72 *Variant Discoveries*, 39 *PCR Assays*, 72 *Variants in Individuals*, and 17 668 *Subject Variants*. We manually reviewed the data stored in both systems to confirm that there were no obvious errors in the storage of these data.

Comparison of frame-based and relational systems

The ontology representation was nearly identical to the information model of genetic sequence data (compare Figures 1 and 3). This is because entities in the information model could be mapped directly to classes in the ontology, and each attribute mapped to a slot. Multiple-cardinality attributes in the information model could also be represented as multi-valued slots (Figure 3). Furthermore, relationships among classes could be represented with slots whose values are instances of other classes. For example, all the variants seen in an individual are listed as multiple values of the ‘Subject Variants’ slot in the class *Variants in Individuals* (Figure 3). Because the ontology closely matched the information model, it was not difficult to keep them synchronized: changes or updates to the latter could be directly applied to the ontology.

The relational representation differed from the information model in many ways, though there were some similarities (compare Figures 1 and 4). While entities mapped to tables, additional tables were needed in the relational schema to represent multi-valued attributes. Consequently, each table could not be viewed as representing an entity in the information model; some tables represented relationships among entities. Generally, attributes for an entity in the information model were columns in a table in the relational schema. But for attributes that had multiple-cardinality, a separate table was needed, and no single column represented the attribute. For example, in the relational schema, there is no column ‘Subject Variants’ in

the *Variants in Individuals* table representing the variants that may found in subjects (Figure 4); instead, there is a table, 'IndivVariants-SubjVariants' representing this attribute. Since the relational model does not closely match the information model, changes in the information model could require complex changes in the relational schema to keep them synchronized.

The time required to process the test queries is shown in Table 1. The queries were processed sequentially, and they required 0.6–8.4 seconds to execute on the relational system, and 0.6–180 seconds on the frame-based system. All queries ran in the same time or faster on the relational system than on the frame-based system, but the differences were not very large except for two of the five queries. Queries 3 and 4 ran approximately 20 times faster on the relational system than on the frame-based system. However, queries on the frame-based system ran significantly faster after previous queries had brought instances into memory; for example, queries 3 and 4 took 3.8 and 1.4 seconds to complete, respectively, on the frame-based system when they were repeated.

DISCUSSION

All biomedical databases are not alike. Most use a relational database schema (Bairoch and Apweiler, 2000; Benson *et al.*, 2000; Harger *et al.*, 2000; Letovsky *et al.*, 1998), while a few are frame-based systems (Chen *et al.*, 1997; Conley, 1999; Karp *et al.*, 1998). Choosing a database schema for a system depends on the requirements of that system. For databases such as Genbank, a relational or even flat file schema is appropriate because it is a large data repository, the types of queries are relatively fixed, and query speed is paramount because the large number of queries handled per day. For databases that store information about complex relationships among entities, permit high-resolution queries of the data, and permit direct programmatic access to the database, modelling the data using an ontology in a frame-based system may be preferable (Altman *et al.*, 1999; Karp, 2000). We evaluated frame-based and relational approaches in this study to evaluate which is appropriate for pharmacogenomics data.

The first step in creating any database is to develop an information model. Many entities and relationships were needed for the information model of pharmacogenetics sequence and polymorphism data (Figure 1). The information model is important because it represents the domain, and it corresponds to the way study investigators think about the data. Furthermore, as biological understanding evolves, new entities or attributes of interest may arise, requiring us to change the information model. If there is a close correspondence between the information model and the database schema that implements that model, it will be easier to update and maintain that database schema when

the information model changes. In addition, a database schema that is similar to the information model is preferred if users write queries directly to the database: users think in terms of the information model, but need to think in terms of the database schema when writing queries. For PharmGKB, we need to provide users direct access to the database because the field of pharmacogenomics is too young for us to be able to predefine a set of standard queries that all users will be interested in. There are too many possible queries based on the complex relationships that can be derived from our information model (Figure 1).

We found that the ontology in the frame-based implementation nearly matched the information model, while the relational schema differed more from the information model. Since users think in terms of the information model when writing queries, we believe that the frame-based system is preferable for querying our pharmacogenetics database (compare queries in Table 2). Of course, this assumes there is a user-friendly query syntax or query interface that makes it possible to formulate queries in terms of the information model. The Protégé API is useful for creating queries in the frame-based system because queries can be written in terms of objects in the ontology.

The relational database implementation performed better than the frame-based implementation in our query tests. In particular, Query 3 took 3 minutes to complete in the frame based system, but only 8.4 seconds in the relational database (Table 1). The difference is likely due to the way queries were executed in the two systems. The frame-based system needed to load 17 668 instances of *Subject Variants* into memory, whereas for the relational database system, the *Subject_Variants* table was joined with the other tables needed to complete the query. Relational joins are executed much faster than iterative loading of instances, particularly when there are many instances. However, query performance on the fame-based system was much faster after instances were loaded into memory (queries 3 and 4 executed in approximately one-fiftieth of the time when they were repeated). In addition, our frame-based system was implemented using a simple relational database back end, and performance may be improved with an optimized database back end for the frame-based system.

Despite the poor performance on two of the queries, the frame-based system performed comparably with the relational system on three of the queries. This difference is likely related to the database schema and nature of the query. The queries where the frame-based system performed well are those where the query does not require iterating across many instances (traversing linked instances did not degrade speed). For example, to perform Query 1, it was necessary to iterate through 2 instances and traverse 3 linked instances; while for the same query in the relational implementation, it was necessary to join

multiple tables. These differences likely offset each other, resulting in comparable query performance.

To our knowledge, relational and ontological representations of biomedical data have not been directly compared to determine the magnitude of the tradeoffs for particular biomedical applications. While the differences in modelling flexibility or performance for frame-based and relational systems may have been anticipated, the magnitude of the differences in a particular domain such as pharmacogenetics could not be predicted. Furthermore, our results suggest that for pharmacogenetics data, an evolutionary approach may be desirable: during the initial phases of building PharmGKB, the ontology was desirable, and as the information model stabilizes and performance becomes more important, the information model can be implemented in a relational schema.

We did not intend the test queries used in this study to represent an exhaustive or thorough test of the performance of the two database systems. The purpose of these queries was to sample the performance of frame-based and relational implementations of our information model with realistic analytical tasks, and to identify major differences. Small differences in timings are probably not meaningful. The main result we wish to emphasize is that neither system had uniformly excellent or poor performance, and that queries where performance differed highlight the differences in the two systems.

We recognize that the relational database schema is not optimized. Certainly, a relational schema could have been created that used fewer tables. But our design choice in this study was to make the implementing schema similar to the XML schema for both systems; changing the relational schema would require making it deviate from the XML schema. Even if the number of tables had been reduced, it is unlikely that the relational schema would match our information model as closely as the ontology model in the frame-based implementation.

In summary, our results show benefits and disadvantages for both frame-based and relational implementations. The frame-based system was preferred with respect to having an ontology that nearly matched our information model for sequence data, but query performance may be slow as the database grows in size. The relational system had better query performance, but a more complex database schema that did not match our information model as well as the ontology. During the early phases of building PharmGKB, it appears the frame-based system is preferable, but as amount of data increases and the types of queries stabilize, it may be desirable to evolve to a relational implementation.

CONCLUSION

Choosing a database schema depends on the requirements of that system. Because pharmacogenetics data is rapidly

evolving and the types of queries and analyses users may desire will probably change, we need a database schema that can accommodate user needs. For genetic sequence data from pharmacogenetics studies, a frame-based system appears advantageous by providing an implementing schema that is close to the information model of pharmacogenetics sequence data. Users may find it easier to formulate queries, and the process of maintaining and updating the database schema will likely be easier as the information model changes. But query performance may become problematic as the database grows. If particular queries become important, or the information model stabilizes, it may be desirable to evolve from a frame-based to a relational implementation.

ACKNOWLEDGEMENTS

The PharmGKB is financially supported by grants from the National Institute of General Medical Sciences (NIGMS), Human Genome Research Institute (NHGRI) and National Library of Medicine (NLM) within the National Institutes of Health (NIH) and the Pharmacogenetics Research Network and Stanford University's Children's Health Initiative. This work is supported by the NIH/NIGMS Pharmacogenetics Research Network and Database (U01GM61374).

REFERENCES

- Achard,F., Vaysseix,G. and Barillot,E. (2001) XML, bioinformatics and data integration. *Bioinformatics*, **17**, 115–125.
- Altman,R.B., Bada,M., Chai,X.Q.J., Carillo,M.W., Chen,R.O. and Abernethy,N.F. (1999) RiboWeb: An ontology-based system for collaborative molecular biology. *IEEE Intelligent Systems and Their Applications*, **14**, 68–76.
- Bairoch,A. and Apweiler,R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, **28**, 45–48.
- Benson,D.A., Karsch-Mizrachi,I., Lipman,D.J., Ostell,J., Rapp,B.A. and Wheeler,D.L. (2000) GenBank. *Nucleic Acids Res.*, **28**, 15–18.
- Boguski,M.S., Lowe,T.M. and Tolstoshev,C.M. (1993) dbEST—database for 'expressed sequence tags'. *Nature Genet.*, **4**, 332–333.
- Bray,T., Paoli,J. and Sperberg-McQueen,C.M. (1998) Extensible Markup Language (XML) 1.0.
- Chaudhri,V.K., Farquhar,A., Fikes,R. and Karp,P.D. (1998) OKBC: A Programmatic Foundation for Knowledge Base Interoperability. *Artificial intelligence*, 600–607.
- Chen,R.O., Felciano,R. and Altman,R.B. (1997) RIBOWEB: linking structural computations to a knowledge base of published experimental data. *ISMB*, **5**, 84–87.
- Conley,E.C. (1999) Internet information on ion channels: issues of access and organization. *Meth. Enzymol.*, **294**, 704–731.
- Hafner,C.D., Baclawski,K., Futrelle,R.P., Fridman,N. and Sampath,S. (1994) Creating a knowledge base of biological research papers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **2**, 147–155.

- Harger,C., Chen,G., Farmer,A., Huang,W., Inman,J., Kiphart,D. *et al.*, (2000) The Genome Sequence DataBase. *Nucleic Acids Res.*, **28**, 31–32.
- Hewett,M., Oliver,D.E., Rubin,D.L., Easton,K.L., Stuart,J.M., Altman,R.B. *et al.*, (2002) PharmGKB: the Pharmacogenetics Knowledge Base. *Nucleic Acids Res.*, **30**, 163–165.
- Karp,P. and Paley,S. (1995) Knowledge representation in the large. *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*.
- Karp,P.D. (2000) An ontology for biological function based on molecular interactions. *Bioinformatics*, **16**, 269–285.
- Karp,P.D. and Paley,S. (1996) Integrated access to metabolic and genomic data. *J. Comput. Biol.*, **3**, 191–212.
- Karp,P.D., Riley,M., Paley,S.M., Pellegrini-Toole,A. and Krummenacker,M. (1998) EcoCyc: encyclopedia of *Escherichia coli* genes and metabolism. *Nucleic Acids Res.*, **26**, 50–53.
- Klein,T.E., Chang,J.T., Cho,M.K., Easton,K.L., Ferguson,R.W., Hewett,M. *et al.*, (2001) Integrating genotype and phenotype information: an overview of the PharmGKB project. *The Pharmacogenomics Journal*, **1**, 167–170.
- Krynetski,E.Y. and Evans,W.E. (1999) Pharmacogenetics as a molecular basis for individualized drug therapy: the thiopurine S-methyltransferase paradigm. *Pharm. Res.*, **16**, 342–349.
- Letovsky,S.L., Cottingham,R.W., Porter,C.J. and Li,P.W. (1998) GDB: the Human Genome Database. *Nucleic Acids Res.*, **26**, 94–99.
- Musen,M.A. (1999) Scalable software architectures for decision support. *Methods of Information in Medicine*, **38**, 229–238.
- Musen,M.A., Ferguson,R.W., Noy,N.F. and Crubezy,M. (2001) Protege-2000: A plug-in architecture to support knowledge acquisition, knowledge visualization, and the semantic Web. *J. Am. Med. Inform. Assoc.*, 1079–1079.
- Noy,N.F., Ferguson,R.W. and Musen,M.A. (2000) The knowledge model of Protege-2000: Combining interoperability and flexibility. *2th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000)*.
- Noy,N.F. and Hafner,C.D. (2000) Ontological foundations for experimental science knowledge bases. *Applied Artificial Intelligence*, **14**, 565–618.
- Roses,A.D. (2000) Pharmacogenetics and the practice of medicine. *Nature*, **405**, 857–865.
- Sherry,S.T., Ward,M.H., Kholodov,M., Baker,J., Phan,L., Smigielski,E.M. *et al.*, (2001) dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res.*, **29**, 308–311.
- Skupski,M.P., Booker,M., Farmer,A., Harpold,M., Huang,W., Inman,J. *et al.*, (1999) The Genome Sequence DataBase: towards an integrated functional genomics resource. *Nucleic Acids Res.*, **27**, 35–38.
- Tarczy-Hornoch,P., Shannon,P., Baskin,P., Espeseth,M. and Pagon,R.A. (2000) GeneClinics: a hybrid text/data electronic publishing model using XML applied to clinical genetic testing. *J. Am. Med. Inform. Assoc.*, **7**, 267–276.
- W3C (2001) XML Schema part 1: Structures, and XML Schema part 2: DatatypesXML Schema Working Group).
- Weinshilboum,R.M., Otterness,D.M. and Szumlanski,C.L. (1999) Methylation pharmacogenetics: catechol O-methyltransferase, thiopurine methyltransferase, and histamine N-methyltransferase. *Annu. Rev. Pharmacol. Toxicol.*, **39**, 19–52.
- Xie,G.C., DeMarco,R., Blevins,R. and Wang,Y.H. (2000) Storing biological sequence databases in relational form. *Bioinformatics*, **16**, 288–289.