EXPLORING HETEROGENEOUS MOLECULAR BIOLOGY DATABASES IN THE CONTEXT OF THE OBJECT-PROTOCOL MODEL

Victor M. Markowitz,* I-Min A. Chen, and Anthony S. Kosky

Information and Computing Sciences Division
Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Abstract

Solutions currently promoted for exploring heterogeneous molecular biology databases (MBDs) include providing Web links between MBDs or constructing MBDs consisting of links, physically integrating MBDs into data warehouses, and accessing MBDs using multidatabase query systems. Arguably the most difficult tasks in exploring heterogeneous MBDs are understanding the semantics of component MBDs and their connections, and specifying and interpreting queries expressed over MBDs. However, most existing solutions address only superficially these problems.

We propose a tool-based strategy for exploring heterogeneous MBDs in the context of the Object-Protocol Model (OPM). Our strategy involves developing tools that provide facilities for examining the semantics of MBDs; constructing and maintaining OPM views for MBDs; assembling MBDs into an OPM-based multidatabase system, while documenting MBD schemas and known schema links between MBDs; supporting multidatabase queries via uniform OPM interfaces; and assisting scientists in specifying and interpreting multidatabase queries. Each of these tools can be used independently and therefore represents a valuable resource in its own right. We discuss the status of implementing our strategy and our plans in pursuing further this strategy.

1 Introduction

Data of interest to molecular biologists are distributed over numerous heterogeneous *molecular biology databases* (MBDs). These MBDs display heterogeneity at various levels: they are implemented using different systems, such as structured files or database management systems (DBMSs), are based on different views of the molecular biology domain, and contain different and possibly conflicting data. Furthermore, each MBD represents some part of the

*To whom all correspondence should be sent: VMMarkowitz@lbl.gov, (510)486-4004 (Fax)

molecular biology domain and is often designed to address certain queries or applications. The data in an MBD are structured according to a *schema* specified in a *data definition language* (DDL) and are manipulated using operations specified in a *data manipulation language* (DML), where these languages are based on a *data model* that defines the semantics of their constructs and operations. Exploring multiple MBDs entails coping with the distribution of data among MBDs, the heterogeneity of the systems underlying these MBDs, and the semantic (schema representation) heterogeneity of these MBDs.

Strategies for managing heterogeneous MBDs can be grouped into two main categories(see [2, 14, 20] for related classifications of heterogeneous database systems):

1. *Consolidation* strategies that entail replacing heterogeneous MBDs with a single homogeneous MBD formed by physically integrating the component MBDs, or by requiring MBDs to be reorganized using a common DDL or DBMS.

2. *Federation* strategies that allow access to multiple heterogeneous MBDs, while the component MBDs preserve their autonomy, that is, their local definitions, applications, and policy of exchanging data with other MBDs. Federation strategies include:

   (a) incorporating in MBDs references (links) to elements in other MBDs, or constructing MBDs consisting of such links;

   (b) organizing MBDs into loosely-coupled multidatabase systems; and

   (c) constructing data warehouses.

Heterogeneous MBDs can be connected via hypertext links on the Web at the level of individual data items. Data retrieval in such systems is limited to selecting a starting data item within one MBD and then following hyperlinks between data items within or across MBDs. Note that data item links (e.g., hypertext links) between MBDs do not require or comply with schema correlations across MBDs. Numerous MBDs are currently providing such links. However, missing and inconsistent links between MBDs prompted some archival MBDs to propose coordinating the management of links between their MBDs [1]. Systems such as SRS [10, 11] and LinkDB [13] extract existing link information from (usually flat file) MBDs, and construct indexes for both direct and reverse links allowing fast access to these MBDs. These systems resolve heterogeneity issues such as duplicate or incompatible identifiers and provide only simple index and key match retrieval, but lack the ability of supporting full query facilities.

Multidatabase systems are collections of loosely coupled MBDs which are not integrated using a global schema. Querying multidatabase systems involves constructing queries over component MBDs, where a query explicitly refers to the elements of each MBD involved. Component MBDs of multidatabase systems can be queried using a common query language, as is done in Kleisli [3], or can be both described and queried using a common data model, as entailed by the Object-Protocol Model tool-based strategy described in this paper. The common query language approach does not require the component MBDs to be represented using a common DDL or data model; however, the users are required to have some knowledge regarding the structure of the MBDs they query. On the other hand, the common data model approach requires all participating MBDs to have a view defined in a common DDL so that users can examine and query component MBDs in the context of the same data model. Unlike link-based MBD systems, multidatabase systems support query languages that allow specifying complex query conditions across MBDs. A query translator is needed for translating queries expressed in the multidatabase query language to subqueries targeting component MBDs, and for optimizing these queries.

Data warehouses entail developing a *global* schema (view) of the component MBDs, where definitions of these MBDs are expressed in a common DDL and discrepancies between these definitions are resolved before they are integrated into the global schema. Data from component MBDs are transformed in order to comply with this global schema, and loaded into a central data repository. The Integrated Genomic Database (IGD) [18], Genome Topographer (GT) [9], and Entrez [21] are examples of data warehouses, where IGD is developed with the ACeDB database system, GT is developed with the Gemstone commercial object-oriented DBMS, and Entrez is based on ASN.1 structured files. The query facilities of data warehouses are provided by the underlying system (e.g., ACeDB), and query processing is local to the warehouse. However, constructing data warehouses requires costly initial integration of component MBDs, followed by frequent synchronization with these MBDs in order to capture the evolution of their schemas. Moreover, data warehouses need to be updated on a regular basis in order to reflect updates of component MBDs.

In this paper, we propose a tool-based strategy for exploring heterogeneous MBDs in the context of the Object-Protocol Model (OPM). We will argue that possibly the most difficult tasks in exploring heterogeneous MBDs is understanding the semantics of component MBDs and their connections, and specifying and interpreting queries expressed over multiple MBDs. Our strategy involves developing tools that provide facilities for constructing and maintaining

OPM views for MBDs implemented using a variety of DBMSs, assembling MBDs into an OPM-based multidatabase system, while documenting MBD schemas and known schema links between MBDs, examining the semantics of MBDs, supporting multidatabase queries via uniform OPM interfaces, and assisting scientists in specifying and interpreting queries. Each of these tools can be used independently and therefore represents a valuable resource in its own right.

The rest of this document is organized as follows. The main semantic problems of exploring heterogeneous MBDs are discussed in section 2. Our tool-based strategy is described in section 3. An example of applying our strategy for querying GDB 6.0 and GSDB 2.0 is described in section 4. In section 5, we review the status of implementing our strategy and our plans for continuing this work.

2 Semantic Problems of Exploring Molecular Biology Databases

2.1 Semantics of Global Schemas and Views

Examining data within and across MBDs is currently hampered by the lack of information on MBDs and their semantics. The need for comprehensive documentation of MBD schemas was discussed extensively at the last two Meetings on Interconnection of Molecular Biology Databases [15, 19]. It was observed at these meetings that MBD schemas capture domain knowledge about biology and therefore the goal of schema design is not only achieving an efficient implementation but also supporting biological exploration.

Existing systems for exploring heterogeneous MBDs do not address the problem of understanding the semantics of component MBDs. For example, systems that support links between MBDs do not provide any information regarding the structure or semantics of the linked MBDs. Some multidatabase systems require users to know the structure (schemas) of component MBDs, without providing them with any support for this purpose. The highest expectations are promoted by data warehouses which present a single unified view while insulating users from the component MBDs. Unfortunately, systems such as GT, IGD, and Entrez are no better documented than their component MBDs, if at all. IGD, GT, and Entrez are based on global schemas (views) of their component MBDs, expressed in ACeDB DDL for IGD, Gemstone DDL for GT, and ASN.1 for Entrez. As of March 1996, no documentation was available on the structure of the GT data warehouse; IGD's global schema specified in ACeDB DDL is scarcely discussed (see [18]) and questions, such as the relationship between the relatively simple IGD schema and often more complex schemas of its components, are left unanswered; the structure of the ASN.1 files underlying Entrez's is also scarcely documented (see [21]).

The global schemas of systems such as GT, IGD, and Entrez are not based on schema integration techniques, but are the result of independent schema design processes based on the domain knowledge underlying the component MBDs. The global schemas for GT, IGD, and Entrez were developed locally by small groups, and therefore do not represent 'consensus' schemas. In order to reduce the complexity of schema design and to make global schemas general enough, developers usually design these schemas using 'generic' classes and/or attributes, which may not be applicable to individual component MBDs and may not fully capture the semantics of the data. Little or no information regarding the relationships between global schemas and participating MBD schemas is provided. Furthermore, the design of the global schemas of such integrated databases are expressed in system-dependent DDLs (e.g., ACeDB's DDL), may be affected by system considerations and therefore contain features which do not reflect domain modeling requirements, or may not capture all the information in each component MBD. For example, certain features of the IGD schema are governed by ACeDB's limitations for modeling large (approaching one gigabyte) databases, rather than by any semantic considerations.

Constructing global schemas or local views for exploring heterogeneous MBDs usually requires detecting semantic conflicts between schemas of component MBDs, ranging from naming conflicts and inconsistencies to detecting identical entities of interest that are represented differently. The same concept can be represented in different schemas by using synonyms, alternative terminology, or different data structures. For example one database could use the term *primer* to represent a class of primer sequences, while another could use the term *oligo*, and yet another could simply represent primers directly using their sequence data. Homonyms can cause naming conflicts in a heterogeneous MBD environment. Domain conflicts can be caused by storing similar values using different units or formats in different MBDs, or from conflicting data arising from different experiments or experimental techniques. Entities of interest can be represented using various data structures in different MBDs, where the diversity of representations stems from different views of the data (e.g., an `Author` can be represented only as an attribute of a `Citation` or as an independent object) and on the underlying DDL (e.g., a `Citation` can be represented as an object of a class within an object data model, but needs to be represented with one or several tables using a relational DDL). Other causes of conflicts include different ways of representing incomplete information (e.g., the meaning of nulls), and different ways of identifying objects in MBDs.

Resolving schema conflicts is a very complex task and may involve various methods ranging from simple renamings in order to resolve naming conflicts to schema restructurings in order to resolve structural dissimilarities. Systems such as IGD and SRS detect and resolve only simple schema and data conflicts, such as some name and object identification conflicts. Alternatively, a heterogeneous MBD system could leave conflict resolution to users. If users are responsible for conflict resolution, such a system could provide a mechanism for recording such resolutions and making them available to other users.

2.2 Semantics of Data Exploration

MBDs are usually explored via specially constructed schemas or views. These views may not necessarily preserve the *information capacity* [16, 17] of component MBDs. The tool-based strategy described in the next section, for example, involves constructing OPM views of component MBDs, where an OPM view may entail constraints (e.g., referential integrity constraints) that are not enforced in the underlying MBD. Consequently access to an underlying MBD through an OPM view is restricted to those data which comply with these constraints, while other data are discarded. Discrepancies between the information capacities of the views employed for exploring heterogeneous MBDs and the underlying component MBDs can be a source of confusion if not properly documented and explained. This is especially critical for data warehouses where data are converted from the format of component MBDs into that of the data warehouse and are subsequently physically loaded into the data warehouse. Anecdotal evidence suggests that information loss occurs during some data conversion processes underlying IGD, but the causes and extent of this problem information loss have not been examined or documented.

Often the power of the facilities provided for exploring heterogeneous MBDs is not properly characterized. Query capabilities provided by MBDs vary between two extremes. Systems such as SRS [10] support only queries with limited keyword matching capabilities. Entrez [21] provides two query interfaces, NetEntrez and WebEntrez, both supporting the expression of form-based queries. The query language of IGD consists of, and is limited to, the ACeDB query language. On the other hand, systems such as Kleisli allow users to query databases using powerful programming languages such as CPL. Although users of these systems can submit very complex queries, it is difficult to imagine a biologist mastering such languages.

Often interfaces for exploring heterogeneous MBDs do not provide any help in clarifying the semantics of the queries users specify or in interpreting the semantics of the query results. For example the Entrez interface provides a number of query forms and various choices of attributes on which to search, but does not offer any description of the extents over which these queries search, or the semantics of the individual attributes. These problems are compounded by the extremely large and complex molecular biology nomenclature and by various differences in interpretations of this nomenclature within the molecular biology community.

Users of many MBD systems, such as Entrez and ENQUire (see http://csl.ncsa.uiuc.edu:80/ENQUire/), interact directly with a Web query interface, and may not be aware of the existence of a global schema. It is difficult or even impossible for users of these systems to detect whether there are conflicts in component MBDs and to realize how the conflicts are resolved. Therefore, when a user receives uninterpreted answers to a query involving conflicting MBDs, information regarding conflicts and how the conflicts are resolved are all hidden from the user.

3 Exploring Heterogeneous Molecular Biology Databases in the Context of the Object-Protocol Model

In this section we briefly describe our tool-based strategy for exploring heterogeneous MBDs, and the tools we are developing in order to pursue this strategy. We will start by presenting the Object-Protocol Model (OPM) and the existing suite of OPM tools which form the backbone for this strategy. Then we will describe how our strategy addresses the semantic problems mentioned in the previous section. An overview of OPM and OPM tools can be found in [4].

3.1 The Object-Protocol Model and Tools

Objects in OPM are uniquely identified by object identifiers, are qualified by attributes, and are classified into classes. Classes can be organized in subclass-superclass hierarchies, where a subclass *inherits* the attributes of its superclasses.

Attributes can be *simple* or consist of a *tuple* of simple attributes. A simple attribute can have a single value, a set of values, or a list of values, and can be *primitive*, if it is associated with a system-provided data type or a controlled value class with controlled values or ranges, or *abstract*, if it takes values from other classes. The attributes of an object class can be partitioned into *non-versioned* and *versioned* attributes, where the former represent stable properties, while the latter represent evolving properties.

OPM provides protocol classes for modeling laboratory experiments. Protocols can be recursively expanded, where a protocol can be specified (expanded) in terms of alternative subprotocols, sequences of subprotocols, and optional protocols. A protocol class can be associated with regular as well as input and output attributes that are used for specifying input and output connections between protocols.

OPM supports the specification of derived attributes using derivation rules involving arithmetic expressions, aggregate functions, and attribute compositions (or path expressions). OPM also supports derived subclasses and derived superclasses. A derived subclass is defined as a subclass of another derived or non-derived class with an optional derivation condition. A derived superclass is defined as a union of two or more derived or non-derived classes.

An OPM query consists of a select, insert, delete, or update statement; only select queries are considered in this paper. These statements can involve conditions consisting of and-or compositions of atomic comparisons. An OPM select query on target class O_i can involve local, inherited, and derived attributes associated with O_i, as well as path expressions starting with these attributes. Although each OPM query is associated with a single target class, this limitation can be offset in part by using abstract attributes (referencing other object or protocol classes) in select and condition statements. Furthermore, multi-target class queries can be constructed using derived classes and derived attributes.

OPM data management tools provide facilities for developing databases using commercial relational DBMSs. OPM schemas can be specified using an OPM Schema Editor or a regular text editor, and can be published in various formats, such as LaTeX and Html. An OPM Schema Translator can be used for mapping OPM schemas into DBMS-specific relational schema definitions and SQL stored procedures [5]. The OPM Schema Translator also generates a *mapping dictionary* with information regarding the mapping of OPM elements into DBMS elements. The OPM Query Translator processes OPM queries and, using the mapping dictionary mentioned above, translates them into SQL queries [7].

3.2 The OPM Tool-Based Strategy

Our strategy for exploring heterogeneous MBDs involves additional OPM tools that provide facilities for (1) constructing OPM views for MBDs developed with or without OPM; (2) assembling MBDs within an OPM-based multidatabase system, while documenting MBD schemas and known schema links between MBDs; and (3) expressing, processing, and interpreting multidatabase queries in this multidatabase system.

3.2.1 Constructing OPM Views for MBDs

OPM views for MBDs can be constructed using an OPM Retrofitting tool [6]. This tool allows constructing one or more OPM views for existing MBDs developed without the OPM data management tools, or constructing multiple OPM views for MBDs developed using the OPM tools.

The OPM Retrofitting tool follows an iterative strategy of constructing OPM views for MBDs. First, a canonical (default) OPM view is generated automatically from the underlying MBD schema. Then this canonical OPM view can be refined using schema restructuring operations, such as renaming and/or removing classes and attributes, merging and splitting classes, adding or removing subclass relationships, defining derived classes and attributes, and so on.

A *mapping dictionary* contains information on the DBMS representations of the view (OPM) constructs. This mapping dictionary is used for generating appropriate retrieval and update methods for the view attributes and classes, and underlies browsing and querying MBD via OPM views.

3.2.2 The Multidatabase Directory

Incorporating an MBD into an OPM multidatabase system involves constructing one or more OPM views of the MBD, and entering information about the MBD and its views into a *Multidatabase Directory*. The multidatabase directory stores information necessary for accessing and formulating queries over the component MBDs, including:

1. *General information* describing each MBD accessible in the system, and the information necessary in order to access that MBD. In particular, for each MBD, the MBD Directory will contain: (i) the *MBD name* and a *brief description* of the purpose of the MBD; (ii) the *physical location* and *history* of the MBD, including the original

charter of the MBD, pointers to related MBDs, etc; (iii) information on the *DDL* and *implementation details* for the MBD, including either precise definitions and examples for the DDL or references to where such information can be found, information on the underlying DBMSs, and implementation strategy for the MBD; (iv) *contact information* for the MBD; (v) *access information*, such as the type (e.g., browsing, querying, update) of data manipulation supported, internet addresses, URLs, and subscription information, as well as references and/or links to more detailed documentation; and (vi) *keywords* (e.g., `sequence` and `human genome`) for high level searches of the schema library described below.

2. An *MBD Schema Library* containing the schemas and related information for component MBDs. Each component MBD can have multiple schemas, including OPM schemas, schemas in the native DDL of the MBD and schemas in other DDLs of interest. The MBD Schema Library contains information on each major schema component (class, attribute), including: (i) *semantic descriptions* describing the physical or real world concepts represented by the schema component, and possibly constraints on the values of instances of the component that cannot be expressed in the underlying DDL; (ii) *design motivation* for the use of a particular construct in representing application data; (iii) *synonyms* and *keywords* for identifying differences in terminology and for establishing potential correspondences between the components of MBD schemas; and (iv) *sample data* providing examples of how the schema constructs are used and how typical data may appear. In addition general information on each schema is stored, including explanations and motivation for the particular view of the MBD provided by the schema. For OPM schemas, the mapping dictionary containing the OPM–DBMS correspondence is also recorded.

3. An *MBD Link Library* containing information about known links between classes in different MBDs. Information on each link includes a description of its semantics, the nature of the correspondence (one-to-one, surjective and so on), and any data-manipulations, such as reformatting of accession numbers, that need to be performed in order to traverse the link.

The Multidatabase Directory is maintained and can be examined using various tools, such as tools for keyword searches that allow identifying MBDs and schemas relevant to a particular query. The Multidatabase Directory is an essential part of processing multidatabase queries.

3.2.3 Supporting Multidatabase Queries

Queries in an OPM-based multidatabase system are expressed in the OPM multidatabase query language (OPM*QL) [8]. OPM*QL extends the single-database OPM query language, OPM-QL, with constructs needed for querying multiple databases. These extensions include the ability to query multiple classes, possibly from distinct databases; constructs that allow navigation between the classes of multiple databases following inter-database links; and the ability to rename fields of a query in order to resolve potential naming conflicts between multiple databases.

An example of a simple OPM*QL query over databases `GSDB` and `GDB` is:

```
SELECT Name = GSDB:Gene.name, Annotation = GDB:Gene.annotation
FROM   GSDB:Gene, GDB:Gene
WHERE  GDB:Gene.accessionID = GSDB:Gene.gdb_xref   AND  GSDB:Gene.name = "ACHE"
```

In the query above, the term "`GSDB:Gene`" refers to class `Gene` of database schema `GSDB` (which must be recorded in the MBD directory) while term "`GSDB:Gene.name`" refers to attribute `name` of class `Gene`. If a class name is unique among all the classes listed in the multidatabase directory, the database name can be omitted from a term, for example "`Gene`" could be used instead of "`GSDB:Gene`".

The `WHERE` statement consists of and-or compositions of atomic comparisons. Conditions can involve multiple classes, possibly from different databases.

Processing OPM multidatabase queries involves generating OPM-QL queries over individual databases in the multidatabase system, and combining the results of these queries using a local query processor. The stages of generating OPM-QL queries and manipulating data locally may be interleaved depending on the particular query evaluation strategy being pursued.

3.2.4 Formulating and Interpreting Multidatabase Queries

The most difficult problems of querying multiple heterogeneous MBDs are (1) formulating a query, which involves determining the MBDs contain relevant data, understanding how data are represented in each of these MBDs, and how data in these MBDs relate to one another; and (2) interpreting the result of a query. Addressing these problems requires comprehensive information on the MBDs that are explored, and unfortunately such information is seldom available. While it cannot fill existing gaps in the documentation of MBDs, the

Multidatabase Directory can help by making existing documentation available through a single resource and in a uniform representation. Browsing and keyword-search tools can be used to identify MBDs potentially of interest. Documentation on MBDs and their schemas can be then examined in order to determine whether they do indeed contain relevent data. When the MBD schema and documentation are not sufficient to clarify certain semantic issues, sample data can provide additional insight by allowing comparisons of data representations for the same or similar data in different MBDs. The MBD Link Library can be consulted in order to determine known correspondences between relevent data in heterogeneous MBDs. Furthermore, using inter-database links in multidatabase queries simplifies their formulation by resolving representational incompatibilities, such as different formats for accession numbers.

The information in the Multidatabase Directory together with the semantics of the operations underlying multidatabase query processing can be used for interpreting query results. For example, information on the semantics of objects in a given class can be used for annotating query results, information about inconsistent inter-database links can be used for explaining null query results, and so on. Consider, for instance, class `Citation` in database Map_X containing only citations published between April 1990 and March 1996; the result of a query requesting all the citations in `Citation` can then state that the results refer to citations published in this time range.

4 An Example

In this section we illustrate how our strategy can be used for exploring heterogeneous MBDs, by describing an application involving the Genome Data Base (GDB) and the Genome Sequence Database (GSDB).

4.1 The GDB-GSDB Multidatabase System

The Genome Data Base (GDB) is an archival MBD of genomic mapping data maintained at Johns Hopkins School of Medicine, Baltimore [12]. The new version of GDB, GDB 6.0 (see http://wwwtest.gdb.org/gdb/), was developed with the Sybase DBMS using the OPM toolkit [4]. GDB contains objects identified by *accession numbers* and are classified in classes organized in a class hierarchy. The main classes of this class hierarchy contain objects representing genomic data, literature references, and information on people and organizations.

The Genome Sequence Database (GSDB) is an archival MBD of genome sequence data maintained at the National Center for Genome Resources, Santa Fe. The current version of GSDB, GSDB 2.0 (see http://www.ncgr.org/gsdb/gsdb.html), has also been developed with Sybase DBMS but without using the OPM toolkit. For GSDB 2.0, an OPM view (see http://gizmo.lbl.gov/DM_TOOLS/OPM/opm_4.html) has been constructed using the OPM Retrofitting tool; this view allows GSDB to be accessed using the OPM query tools [7]. GSDB 2.0 is structured around one main class of objects, `Entry`, whose objects represent DNA sequences identified by accession numbers; the actual sequences (strings) are represented by objects of another class, `Sequence`. GSDB 2.0 also contains objects representing various entities, including genes, products, sources, and references.

Both GDB and GSDB have a `Gene` class. In GSDB 2.0, genes are considered to be a kind of `Feature`, and are characterized by gene names and references to external MBDs, such as GDB, that contain additional information on genes. In GDB, genes are represented by objects of class `Gene` and are characterized by information that includes the reason a genomic region is considered a gene, links to gene families the gene belongs to, mapping information, and references to derived sequences.

Sequences are represented in GSDB by objects of class `Sequence`. Sequence data include the actual sequence, sequence length, and information on the source of the sequence. Sequence information in GDB is represented by objects of class `SequenceLink`. These objects contain annotations linking primary GDB objects to external sequence MBDs such as GSDB, as well as information regarding the beginning and end points of sequences.

Both GDB and GSDB contain classes representing products. In GDB, products are limited to gene products, while in GSDB a product can be associated with any feature. In both GDB and GSDB, these classes seem primarily to serve as a way of referencing external MBDs, such as protein MBDs.

Both GSDB and GDB contain data representing references and/or citations. In GSDB, a `Reference` object is considered as a kind of (i.e., a specialization of) `Feature` object. References in GSDB are characterized by titles, publication status, lists of authors and editors, and external references to the Medline bibliographic database. In GDB, citations are represented by objects of class `Citation` and are further classified in subclasses of `Citation` representing books, journals, articles and so on.

4.2 Examples of Queries Expressed over GDB and GSDB

The following OPM multidatabase queries are examples of typical queries expressed over GDB 6.0 and GSDB 2.0. These queries were suggested by Chris Fields of the National Center for Genome Resources, Santa Fe, and were specified with help provided by Ken Fasman and Stan Letovsky of the Johns Hopkins School of Medicine, Baltimore and Carol Harger of the National Center for Genome Resources.

Query 1: Find the protein kinase genes on chromosome 4. To identify protein kinase genes in GSDB it is necessary to first find protein kinase products in the GSDB `Product` class, and then find `Genes` associated with the same `Feature` as the `Product`. The corresponding `Gene` in GDB can then be accessed by following the `gdb_xref` attribute from the GSDB `Gene`, if present, and equating it with the GDB `accessionID` attribute. Some string reformatting was needed in this query in order to resolve discrepancies between the representations of accession numbers in GDB and GSDB; this reformatting was implemented using functions built into the OPM multidatabase query language, but are ignored in the queries shown below. In order to test whether a `Gene` occurs on chromosome 4, one can then follow the path in GDB from `Gene` to `MapElement` to `Map` to `Chromosome`.

```
SELECT GDB:Gene.displayName, GDB:Gene.accessionID, Feature.products.name
FROM   GSDB:Feature, GDB:Gene
WHERE  Feature.products.name MATCH "%protein kinase%"
  AND  Feature.genes.gdb_xref = GDB:Gene.accessionID
  AND  GDB:Gene.mapElements.map.chromosome.displayName = "4";
```

Query 2: Find sequenced regions on chromosome 17 with length greater than 100,000. Map elements on chromosome 17 are selected from the GDB class `MapElement` using the path from class `MapElement` to class `Map` to class `Chromosome`. Links from `MapElement` objects to GSDB `Entry` objects are found using the GDB `SequenceLink` class. From the GSDB `Entry` the corresponding sequence can be found and tested to see if its length is greater than 100,000.

```
SELECT Entry.accession_number, Entry.sequence.length
FROM   GDB:MapElement, GDB:SequenceLink, GSDB:Entry
WHERE  MapElement.map.chromosome = "17"
  AND  SequenceLink.dBObject = MapElement.segment
```

```
  AND  SequenceLink.externalDB.displayName = "GSDB"
  AND  SequenceLink.accessionID = Entry.accession_number
  AND  Entry.sequences.length > 100000;
```

Query 3: Find the sequences of ESTs mapped between 4q21.1 - 21.2. Currently this query requires two sub-queries: the first sub-query finds the coordinate range and the second sub-query finds ESTs with coordinates in that range and their sequences. Planned extensions to the multidatabase query system will allow this query to be expressed as a single OPM query.

The first part of the query finds the coordinates of the points q21.1 and q21.2 in the `Cytogenetic Map` of chromosome 4:

```
SELECT MapElement.coordinate, MapElement.point, MapElement.segment.displayName
FROM   GDB:MapElement
WHERE  MapElement.map.objectClass = "CytogeneticMap"
  AND  MapElement.map.chromosome.displayName = "4"
  AND  MapElement.segment.displayName IN {"q21.1", "q21.2"};
```

Next, one can retrieve the expressed `Amplimers` occurring between these coordinates and lookup the corresponding sequence in GSDB.

```
SELECT Amplimer.displayName, Entry.accession_number,
       Entry.sequences.length, Entry.sequences.sequence
FROM   GDB:Amplimer, GDB:SequenceLink, GSDB:Entry
WHERE  Amplimer.isExpressed = "Yes"
  AND  Amplimer.mapElements.map.chromosome.displayName = "4"
  AND  Amplimer.mapElements.sortCoord >= START_COORD
  AND  Amplimer.mapElements.sortCoord <= END_COORD
  AND  SequenceLink.dbObject = Amplimer
  AND  SequenceLink.externalDB.displayName = "GSDB"
  AND  SequenceLink.accessionID = Entry.accession_number;
```

where `START_COORD` and `END_COORD` are the values from the previous query.

5 Pursuing the OPM Tool Based Strategy

We have developed some of the tools required for implementing our strategy for exploring heterogeneous MBDs in the context of OPM. In this section we will describe the current state of our tools and discuss our plans for pursuing the implementation of this strategy.

5.1 Constructing OPM Views

The current version of the OPM Retrofitting tool can be applied to MBDs developed with Sybase and can be adapted straightforwardly to MBDs developed with other commercial relational DBMSs, such as Oracle and Informix. We are in the process of extending the OPM Retrofitting tool to MBDs developed using non-relational DBMSs, such as ACeDB, and/or defined using non-relational DDLs, such as ASN.1. These extensions will broaden the range of MBDs for which an OPM view can be constructed, and that can thus be included in an OPM multidatabase system.

5.2 The Multidatabase Directory

As mentioned in section 3, central to our strategy of assembling MBDs into a multidatabase system is a Multidatabase Directory that includes general information on MBDs, MBD links, and MBD schemas. In our current implementation, the MBD Schema Library consists only of the OPM schema, associated schema documentation, and mapping information for each MBD. OPM supports extensive schema documentation capabilities: each class or attribute in an OPM schema can be associated with description and user-specified properties; for a controlled value class, each controlled value can also be associated with its description. Therefore, detailed schema descriptions can be embedded in an OPM schema definition. We are not aware of any other data models that support such documentation capabilities. Nevertheless, this is still not adequate for assisting users in examining and understanding the semantics of MBDs, nor in specifying and interpreting multidatabase queries.

We plan to develop an extended Multidatabase Directory as an independent resource that will provide support for examining and understanding MBDs as well as help scientists in specifying queries across multiple MBDs. The MBD Schema Library part of this Directory will contain schemas for MBDs expressed not only in OPM but in a variety of different DDLs as well, including each MBD's native DDL and several DDLs (e.g., ASN.1 and ACeDB), which are widely used within the molecular biology community. Consequently, scientists interested in a particular MBD will be able to view the MBD schema in a DDL with which they are familiar. The versions of an MBD schema represented in different DDLs will be generated using *schema conversion* tools that will follow the iterative schema conversion methodology underlying the OPM Retrofitting tool. The MBD Schema Library will also contain abstract *overview* schemas, in which related schema components will be grouped together into higher-level components in order to provide a more concise and comprehensible high-level view of the MBDs.

MBD schema documentation will contain *sample data* that will help to reveal schema nuances that are not evident in the schema definition. Further observing how the same or similar data are represented in different MBDs will help to give insights into how to exchange data between MBDs. Sample data will be annotated in order to explain the significance of its various components.

As a development and maintenance resource, the Multidatabase Directory will provide facilities for constructing, extending, and maintaining (revising, updating) information on MBDs. These facilities will include tools for constructing abstract overview schemas and schema and data converters for transforming schemas expressed in an MBD's native DDL into schemas expressed in alternative DDLs. Since MBD schemas evolve over time, the MBD Schema Library will support *schema versioning* and will include tools for keeping track of MBD schema changes. *Schema annotation* facilities will allow scientists to share their understanding and/or view of MBD schemas and thus contribute to enhancing the comprehensibility and value of MBD schema documentation. Search engines will be provided for identifying MBDs relevant to a particular topic, and for quickly determining the relevant parts of a particular MBD.

Certain MBDs provide additional tools such as sequence analysis programs for analyzing a DNA sequence. Such data analysis tools can also be employed in a multidatabase system, so the Multidatabase Directory needs to be extended in order to include information regarding software support.

5.3 Supporting Multidatabase Queries

The current (first) version of the OPM multidatabase query translator [8] has been developed between October 1995 and January 1996. This version of the translator supports the expression of queries that combine (join) and manipulate data from multiple MBDs, and relies on information on the OPM schemas and remote access facilities of these MBDs contained in the Multidatabase Directory.

The multidatabase query processing strategy currently pursued, involves two stages:

1. OPM multidatabase queries are decomposed into component OPM queries for each component database involved in the query, where single-database OPM queries are

evaluated using the existing OPM query translator [7].

2. Data retrieved from each single-database OPM query are assembled locally into the result of the multidatabase query, where the local query processor is capable of performing joins and evaluating conditions over complex nested data-structures.

Although this query processing strategy is very simple and general, it can be inefficient for certain types of queries. For example, for evaluating a query that selects a small number of genes from the GDB class `Gene` and then finds the related genes in the GSDB `Gene` class, it would be inefficient to retrieve all the GDB and GSDB genes separately and then compare their accession numbers, rather than just looking up the GSDB genes using the accession numbers of the genes retrieved from GDB. A more efficient query strategy could find an order for the subqueries and evaluate them in sequence, so that the results of each subquery would be used to restrict the next subquery in the sequence. However such a strategy would be considerably more difficult to implement, since it would require statistics on sizes of individual classes and the selectivity of constraints in order to determine an optimal evaluation order. Although we consider pursuing such strategies in the future, in the short term we plan to increase the efficiency of multidatabase query processing by using *inter-database links*.

Inter-database links are known connections between heterogeneous databases that are recorded in the Multidatabase Directory together with the metadata on component databases. An example of an interdatabase link is the link between the `Gene` class in GSDB and the `Gene` class in GDB, represented by attribute `gdb_xref` of class `Gene` in GSDB; this attribute contains GDB accession numbers and thus indirectly points to GDB `Gene` objects. Following such a link allows retrieving from a component database only the objects that are involved in specific links, instead of retrieving all the objects in a class, where following inter-database links predetermines a query evaluation order.

From the perspective of a user constructing OPM multidatabase queries, inter-database links look like regular OPM abstract attributes (which represent intra-database links), except that the result of following such a link will be an object in another database rather than an object in a different class of the same database. Thus the Multidatabase Directory will associate an attribute name with each inter-database link, thus augmenting the list of attribute names that are already associated with an OPM class. These attributes can then be used for including the inter-database links in attribute paths in a query.

It should be noted that inter-database links do not subsume the general multidatabase joins already implemented, but rather complement them: multiple MBDs can be queried using multi-database joins (as done in our current implementation), inter-database links, or a combination of the two. This means that users are not confined to using the links already determined and included in the MBD Link Library, but can determine their own correspondences between databases as well. Using a combination of multidatabase joins, inter-database links, and other locally performed data manipulations, it should be possible to express very general and efficient multidatabase queries.

In order to assist users in understanding the semantics of multidatabase queries, the OPM multidatabase query processor will also provide support for interpreting queries in terms of the semantics of both the target MBDs and the query processing operations.

5.4 Technological Alternatives

There are commercial distributed-join software tools, such as the Sybase *Enterprise CONNECT* family of products, that allow querying multiple relational databases. It should be noted that such tools do not help constructing multidatabase systems: one still needs to understand the component databases in their relational representation, their semantics and links. The OPM tools support higher level representations of databases, using abstract constructs that are better suited for representing biological data. In addition, the Multidatabase Directory simplifies substantially the task of exploring and understanding multiple databases.

A distributed-join tool could underly the processing of OPM multidatabase queries, where an OPM multidatabase query would first be translated into multidatabase SQL queries. The multidatabase SQL queries could be then processed by the distributed-join tool and the query results could be then converted into OPM data format. This query processing strategy is different from our current strategy of translating an OPM multidatabase query into OPM queries over individual databases.

Although we plan to examine this alternative query processing strategy in terms of cost and performance, we are aware of several problems inherent to this alternative. First, a distributed-join tool can be used only for a set of data sources supported by the tool: usually major commercial DBMSs or widely used standards, but not the more specialized data-sources frequently used for molecular biology databases (e.g., ASN.1, ACeDB). For data sources that are not supported by such a tool, additional programming would be still required. Moreover, such a tool is restricted to the constructs of a standard relational query language (e.g., SQL). Such query languages have been found to be overly restrictive and difficult to use for querying complex MBDs: for example, an SQL query over the relational schema for GSDB 2.0 will in general involve substantially more tables, and will be considerably more complex, than an equivalent OPM query expressed over the OPM view of GSDB. Furthermore, the OPM multidatabase query translator is based on a more powerful nested relational algebra which supports directly operations on nested sets and complex data structures. Finally, using a distributed-join tool for processing OPM multidatabase queries will make the performance of the OPM multidatabase query translator dependent on this tool. With our current query processing approach, we have the flexibility of experimenting with any query optimization strategy and hopefully achieve better query performance.

References

[1] Blake, J., et al. Inter-Connection of Biological Databases: Exploring Different Levels of Molecular Biology Database Federation. In [19].

[2] Bright, M.W., Hurson, A.R., and Pakzad, H. A Taxonomy and Current Issues in Multidatabase Systems. *IEEE Computer*, 25(3), pp. 50-59, 1992.

[3] Buneman, P., Davidson, S., Hart, K., Overton, C., and Wong, L. A Data Transformation System for Biological Data Sources. In *Proc. of the 21st Int. Conference on Very Large Data Bases*, pp. 158-169, 1995.

[4] Chen, I.A., and Markowitz, V.M. An Overview of the Object-Protocol Model (OPM) and OPM Data Management Tools. *Information Systems*, 20(5), pp. 393-418, 1995.

[5] Chen, I.A., and Markowitz, V.M., OPM Schema Translator 4.0, Reference Manual, Technical Report LBL-35582 (revised), 1995.

[6] Chen, I.A., and Markowitz, V.M., Constructing and Maintaining Scientific Database Views, Technical Report LBL-38359, 1996.

[7] Chen, I.A., Markowitz, V.M., and Szeto, E., The OPM Query Translator, Technical Report LBL-33706, 1995. Available at http://gizmo.lbl.gov/opm.html.

[8] Chen, I.A., Kosky, A., Markowitz, V.M., and Szeto, E., OPM*QS: The Object-Protocol Model Multidatabase Query System, Technical Report LBL-38181, 1995.

[9] Cozza, S., Reed, E. C., Salit, J., Chang, W., Marr, T. Genome Topographer: A Next Generation Genome Database System (Abstract), presented at the meeting on Genome Mapping and Sequencing, Cold Spring Harbor Laboratory, Cold Spring Harbor, 1994.

[10] Etzold, T., and Argos, P. SRS, An Indexing and Retrieval Tools for Flat File Data Libraries. *Computer Applications of Biosciences*, **9**, 1, pp. 49-57, 1993. See also http://www.embl-heidelberg.de/srs/srsc.

[11] Etzold, T., and Argos, P. Transforming a Set of Biological Flat File Libraries to a Fast Access Network. *Computer Applications of Biosciences*, **9**, 1, pp. 58-64, 1993.

[12] Fasman, K.H., Letovsky, S.I., Cottingham, R.W., and Kingsbury, D. T. Improvements to the GDB Human Genome Data Base. Nucleic Acids Research, Vol. 24, No. 1, pp. 57-63, 1996. See also http://wwwtest.gdb.org/gdb/about.html.

[13] Goto, S., Akiyama, Y., and Kanehisa, M. LinkDB: A Database of Cross Links Between Molecular Biology Databases. In [19].

[14] Heimbigner, D., and McLeod, D. A Federated Architecture for Information Management, *ACM Transactions on Office Information Systems* 3(3), pp. 253-278, 1983.

[15] Karp, P., Report of the 1st Meeting on Interconnection of Molecular Biology Databases, Stanford, California, 1994; http://www.sri.ai.com/people/pkarp/mimbd/mimbd-94.html.

[16] Kosky, A., Davidson, S., and Buneman, P. Semantics of Database Transformations. Technical Report MS-CIS-95-25, University of Pennsylvania, 1995.

[17] Miller, R.J., Ioannidis, Y.E., and Ramakrishnan, R., The Use of Information Capacity in Schema Integration and Translation, *Proc. of the 19th International Conference on Very Large Databases*, 1993, pp. 120-133.

[18] Ritter, O. The Integrated Genomic Database. In *Computational Methods in Genome Research* (S. Suhai, ed.), pp. 57-73, Plenum, 1994. See also http://genome.dkfz-heidelberg.de:80/igd/start_igd_doc.html.

[19] Second Meeting on Interconnection of Molecular Biology Databases, Cambridge, United Kingdom, 1995, http://www-genome.wi.mit.edu/informatics/abstracts.html.

[20] Sheth, A.P., and Larson, J.A. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3), pp. 183-236, 1990.

[21] Shuler, G.D., Epstein, J.A., Ohkawa, H., Kans, J.A. Entrez. In *Methods in Enzymology*, (R. Doolittle, ed.). Academic Press, Inc. In press. See also http://www3.ncbi.nlm.nih.gov/Entrez/.