Coping with Data Modeling Diversity *

Victor M. Markowitz
Information and Computing Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA
Email: VMMarkowitz@lbl.gov

July 13th, 1995

Abstract

Scientific data are often scattered among heterogenous data repositories. Exploring data across multiple data repositories requires the ability of understanding and correlating their structures (schemas). Such correlations need to address the diversity of views of the scientific domain represented by different data repositories as well as the diversity of data modeling languages used for expressing these views. We briefly examine several data models that are currently employed for specifying molecular biology databases in the context of a reference data model and discuss the problem of converting schemas between these data models.

1 Background

Scientific data repositories are often isolated and characterized by various degrees of heterogeneity: they usually represent different views (schemas) of the scientific domain, where the schemas are specified using different languages (data models), and are implemented using different data management systems, such as file management systems or database management systems (DBMSs). Interacting with multiple data repositories requires the ability of understanding and correlating schemas specified using different data models.

Data models currently used for describing molecular biology data repositories include (but are not limited to): the relational model (underlying all relational DBMSs), various versions of the Entity-Relationship Model (ERM) [3] and Extended Entity-Relationship Model (EERM) [13], the data model of Genera [11], the data model of ACEDB (underlying the ACEDB data management system) [5], and the Object-Protocol Model (OPM) [4].

*This work is supported by the Office of Health and Environmental Research Program of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

For example, the relational model is used for specifying (directly or via tools) the databases implemented with relational DBMSs, such as the Genome Sequence Data Base (GSDB) and Genome Data Base (GDB), EER is used for specifying the object schema Flybase database at Harvard University and several databases at the Chromosome 22 Center at University of Pennsylvania;[1] Genera is used for specifying the object schema of the Enzyme database[2] and the XLocus database of cross-species gene-based linkages;[3] ACEDB is used for developing numerous databases[4], including the Integrated Genomic Database (IGD);[5] OPM is used for specifying the object schema of databases such as the Citations database,[6] of the new GDB multidatabase system, and the Protein Data Bank (PDB).[7]

The data models of Genera, ACEDB, EERM, and OPM share similar constructs. We briefly examine these data models using a reference data model and discuss the problem of converting schemas between these data models.

2 A Reference Data Model

A *data model* [18] provides constructs for defining: (i) the structure of data (*schema*) using a *data definition language* (DDL); and the operations allowed on data using a *data manipulation language* (DML). In this section we briefly describe the DDL of a reference object data model that is similar to several semantic [6, 8] and object data models [2].

2.1 Basic Constructs

Database schemas are specified in this reference data model in terms of objects that are qualified by attributes and are classified in classes. A subset of the attributes associated with a class is usually specified as the *identifier* for the objects in the class. A class can be specified as the specialization (subclass) of other classes; a specialization class inherits the attributes of all its (direct and transitive) superclasses. Subclass-superclass relationships can be used for organizing classes in *class hierarchies*. Examples of classes are shown in figure 1.

Attributes take values from classes of values, and can be simple or consist of a tuple of simple attributes. An attribute can be single-valued, set-valued, or list-valued. A *simple* attribute is associated with either a single class of values or a union of such classes. A *tuple* attribute (e.g., `(left_fragment, right_fragment)` of `Connection` in figure 1) consists of

[1]http://www.cis.upenn.edu/ clib/chr22db/chr22dbhome.html

[2]http://www.agron.missouri.edu/enzyme.html

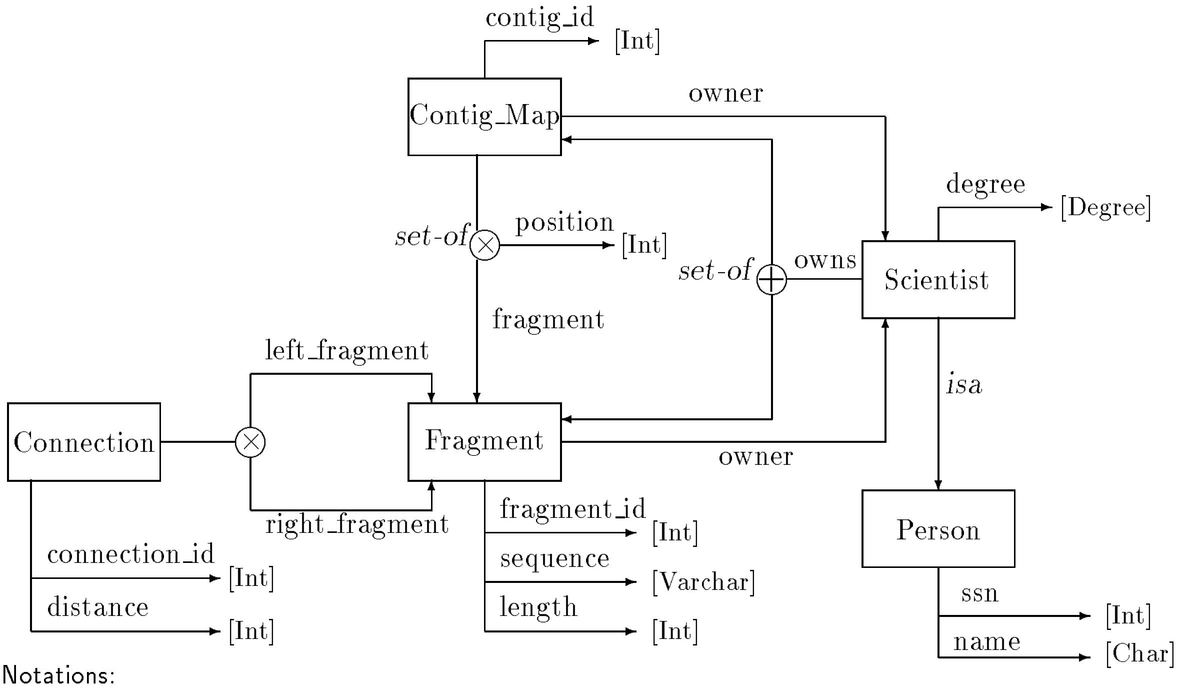
[3]http://cgsc.biology.yale.edu/xlocus.html

[4]A list is available at http://probe.nalusda.gov:8000/acedocs/acedbfaq.html

[5]http://genome.dkfz-heidelberg.de/igd

[6]http://gdbdoc.gdb.org/citdb/citdb.html

[7]http://www.pdb.bnl.gov/3DBase/3DBase.html



Notations:

- Labeled rectangles represent classes.
- Arrows represent attributes, where the arrows point to the associated classes of values .
- Circles containing a "+" denote unions of classes.
- Circles containing a "×" denote tuple attributes.
- `Degree` is a set of enumerated values: {`"PhD"`, `"MSc"`, `"DSc"`, `"BA"`, `"BSc"`}.

Figure 1: Diagrammatic Representation of Classes of Objects

several *component* simple attributes, where each component attribute is associated with a single class of values or a union of classes. Depending on the type of its class of values, an attribute can be primitive or abstract: a *primitive* attribute take values from a *controlled* class of enumerated atomic values or ranges of values, or a class of atomic values of predefined data types; an *abstract* attribute takes values from a class or union of classes of objects (e.g., `owns` of `Person` in figure 1).

Attributes are characterized by null and inverse constraints. *Null constraints* refer to whether an attribute can have null (empty set) values. *Inverse constraints* regard pairs of abstract non-derived attributes and express object cross referencing: if an abstract attribute A of class O_i (e.g., `owner` of `Contig_Map` in figure 1) is defined as the *inverse* of abstract attribute B of class O_j (e.g., `owns` of `Person` in figure 1), then for every instance x of O_i and every instance y of O_j, if y is a value of $A(x)$ then x is also a value of $B(y)$.

2.2 Derived Constructs

Derived attributes have values that are derived from the values of other attributes using derivation expressions , such as arithmetic expressions, aggregate functions (`min`, `max`, `sum`, avg, count), and attribute composition. Arithmetic expression and aggregate function derivations are straightforward and are not further discussed here.

An attribute A of a class O_i, can be defined as a *composition* of other attributes by associating it with (a composition derivation consisting of) one path or a union of paths of the following form: B_1 $[O_{i_1}]$ B_2 $[O_{i_2}]$ ... B_n $[O_{i_n}]$, where each O_{i_k} $(1 \leq k \leq n)$ denotes a class, and each B_k $(1 \leq k \leq n)$ denotes an attribute associated with $O_{i_{(k-1)}}$ $(O_{i_0} = O_i)$ and takes values from value class that either includes or is a superclass of O_{i_k}. For example, the following derived attribute can be associated with class Connection shown in figure 1: right_fragment [Fragment] owner.

We consider only two types of derived (virtual) classes: derived subclasses of one or several object classes and derived superclasses of several classes (e.g., see [1]).

A *derived subclass*, O_s, of one or (intersection of) several object classes, O_1, ..., O_m ($m \geq 1$), and/or associated with a condition, consists of the subset of objects that belong to the intersection of classes O_i, $1 \leq i \leq m$, and satisfy the associated condition. For example, a class of fragments longer than 100 base pairs, Long_Fragment, can be specified as a derived subclass of class Fragment shown in figure 1, associated with condition: length > 100.

A *derived superclass*, O_g, of object classes O_1, ..., O_m ($m \geq 2$), consists of the union of objects belonging to these classes. For example, a class Contig_or_Fragment can be defined as a derived superclass of classes Contig_Map and Fragment shown in figure 1.

2.3 Additional Constructs

Some data models support data versioning (e.g., see [17]) by partitioning the attributes of an object class into *non-versioned* and *versioned* attributes. Non-versioned attributes describe the stable (non evolving) part of an object (such as ssn of class Person in figure 1), while versioned attributes describe the changing (evolving) part of an object (such as degree of class Scientist in figure 1). A versioned abstract attribute can refer to either object versions that are specified as *current* (default) versions or to *specific* object versions.

Molecular biology data are often explored using a hierarchical (directed tree) structure, where the root of the tree represents a class of objects of interest, T, and the other nodes of the tree represent objects and attribute values related to the objects of T. Consider for example the object classes shown in figure 1. Contig maps and their related owners and fragments can be organized in a directed tree structure whose root node represents (and is labeled with the name of) class Contig_Map, with classes Fragment and Scientist and attribute position of Contig_Map represented as nodes connected from the root, attribute name of class Person represented as a node connected from the node labeled Scientist, and attributes fragment_id and sequence of class Fragment represented as nodes connected from the node labeled Fragment. Hierarchical object structures are not usually supported in object data models, but can be expressed indirectly via nested object queries.

3 Data Models for Molecular Biology Databases

In this section we examine the main data modeling constructs of the relational model, EERM, Genera, ACEDB, and OPM in the context of the reference object data described in section 2. EERM, Genera, ACEDB, and OPM support the concept of object, classification of objects in classes and qualifying objects using attributes.

3.1 Relational Model

Unlike EERM, Genera, ACEDB, and OPM, the relational model allows specifying databases in terms of relations (tables) consisting of tuples (rows), where each tuple element represents the value of a single-valued primitive attribute. Tuples are uniquely identified by *primary-key* attribute values and can refer to (primary-key values of) tuples in other relations via *foreign-key* attribute values. Objects can be represented by tuples of one or several relations, where more than one relation may be required in order to represent constructs that are not supported in relational DBMSs, such as set-valued attributes. Abstract attributes and subclass-superclass relationships are represented using foreign-key attributes, controlled value classes are represented using rules and (possibly) additional relations, while inherited attributes, derived attributes and classes are represented using SQL procedures. Object retrieval and update are also represented using SQL procedures.

3.2 Extended Entity-Relationship (EER) Model

An EER object is called an *entity* and EER object classes are called *entity-sets*. The EER model supports simple, primitive or abstract attributes; class hierarchies; and controlled classes of enumerated atomic values or ranges of values. In EER, abstract attributes are specified as *relationship-sets* between (and external to) entity-sets, rather than local to entity-sets. The EER model does not support set-valued attributes, tuple attributes, union value classes, and derived classes and attributes. However, set-valued attributes can be modeled using additional (auxiliary) entity-sets, tuple attributes can be specified using auxiliary relationship-sets, and a union value class can be specified using an auxiliary entity-set specified as a generalization of the entity-sets that are involved in a union value class.

3.3 Genera

The Genera data model supports attributes that can be single or set valued, primitive or abstract, simple or tuples of simple attributes; controlled classes of enumerated atomic values; derived attribute composition involving single path derivations. The Genera data model does not support class hierarchies, union value classes, and derived classes.

3.4 ACEDB

The ACEDB data model supports attributes that can be single or set valued, primitive or abstract, simple or consisting of tuples of simple attributes; controlled classes of enumerated atomic values; derived attributes. The ACEDB data model does not support directly class hierarchies and union value classes. However, a subclass can be modeled using a derived subclass construct and a union value class can be specified using an auxiliary class specified as a superclass of the classes involved in the union value class. The semantics of null constraints and uniqueness of values in ACEDB is different from the analogous constraints in the reference data model. In addition, ACEDB supports directly the specification of hierarchically structured classes of objects and provides a *constructed type* for defining recursive object structures.

3.5 Object-Protocol Model (OPM)

OPM supports all the basic and derived constructs of the reference data model presented above. In addition, OPM supports versioned attributes, and has a protocol class construct that allows modeling molecular biology laboratory protocols (e.g., specifying protocols in terms of their subprotocol steps, input-output connections of protocol steps).

3.6 Comments

The Genera data model is a strict subset of OPM, the EER model is less expressive than OPM and ACEDB, and OPM and ACEDB support different flavors of similar constructs, but also have some constructs that do not have a direct correspondent in the other model. The relational model can be used for expressing object structures but with substantially larger, and therefore less comprehensible, schemas than the object data models.

We have not considered in the discussion above the ASN.1 notation used at NCBI for specifying structured data files [16], since ASN.1 is not a data model per se. However, except class hierarchies, the constructs of the reference data model can be expressed using ASN.1.

EERM, Genera, ACEDB, and OPM are affected by implementation considerations. For example, since the EER, Genera, and OPM tools target primarily relational DBMSs, the EERM, Genera, and OPM data types reflect the data types supported by these DBMSs; the null constraints in the ACEDB data model have different semantics than in other data models because they reflect ACEDB's implementation of tree structures.

A data model can be judged for the conciseness and clarity of the schemas expressed with its constructs. Some data models are easier to use for modeling certain structures than others, but all data models impose certain restrictions. A data model cannot be considered as well defined if it has ambiguous constructs. Data model well-definedness is very important from a computer science point of view, but seems to be of little concern to developers and users of molecular biology databases.

Developers of molecular biology databases are somewhat concerned about the size of database schemas since large schemas are hard to maintain, but very few (if any) seem to care about the clarity of the data model used for specifying these schemas. Scientists using molecular biology databases are not interested in the size, clarity, or even lack of ambiguity of database schemas, and are concerned only about the ease of interaction with these databases. Thus, ACEDB is popular because of its graphical user interfaces, tolerance of incomplete data, and data exchange and reorganization facilities; Genera is used mainly for its ability of providing high-level Web-based interfaces on top of (existing or new) Sybase databases; EER and OPM are known primarily because of their associated tools that provide object-based interfaces on top of relational DBMSs.

The report on the first Meeting on Interconnection of Molecular Biology Databases [9] indicates that there is agreement on the need of using object data models for modeling molecular biology databases (MBDs), but there is no agreement on what data model or individual constructs are preferable. Developing a standard data model for MBDs has been promoted by some for facilitating MBD interoperation. Standardization, however, could have a negative effect on MBD modeling. Thus, object constructs are not always appropriate for modeling certain molecular biology data. For example, object constructs are not sufficient for accurately modeling molecular biology laboratory experiments [4]. Additional constructs are needed for better modeling sequences and maps. Such extensions should not be denied via standardization as long they are justified and rigorously defined. Furthermore, standardization is not need for interconnecting MBDs which can be achieved by developing converters between the data models underlying MBDs. Most of these data models are already closely related. Translating from one data model to another depends only on providing clear and complete semantic definitions for the constructs of these data models. Unfortunately, such definitions are not always available.

4 Converting Schemas

In this section we briefly discuss the problem of converting schemas between different data models.

4.1 Schema Conversion Criteria

Schema conversion needs a measure of the *information capacity* preserved by the conversion. Thus, converting a schema S_1 in data model M_1 to schema S_2 in data model M_2 (M_1 and M_2 can be the same or different data models), involves defining a data mapping (converter) that can transform instances of S_1 into instances of S_2. Informally, S_2 is said to preserve the information capacity of S_1 if the data mapping associated with the conversion of S_1 into S_2 transforms instances of S_1 into instances of S_2 without loss of information.

Information Capacity Preserving Schema Conversion.[8] Let S_1 be a schema in data model M_1 and let S_2 be a schema in data model M_2. S_2 *preserves the information capacity* of S_1 if

1. there exists a *total* function ρ such that ρ converts (maps) a consistent state of S_1 into a consistent state of S_2;
2. there exists a function ρ' such that ρ' converts a consistent state of S_2 into a consistent states of S_1;
3. the composition of ρ followed by ρ' is the identity on the set of all consistent states of S_1.

If ρ' is also total and the composition of ρ' followed by ρ is the identity on the set of all consistent states of S_2, then S_1 and S_2 have *equivalent information capacity*. □

Information capacity has been used extensively to characterize the correctness of schema conversions within the same data model (e.g., relational database schema normalization transformations) or between different data models. The degree of information capacity preservation depends on the goal of schema conversion [15]:

1. if schema S_2 is used only as a view for querying a database specified using schema S_1 then the conversion of S_1 into S_2 does not need to preserve the information capacity of S_1; however, if S_1 is also used for querying a database specified using schema S_2 then S_1 and S_2 must have equivalent information capacity;
2. if schema S_2 is used for viewing (e.g., browsing) an entire database specified using schema S_1 then S_2 must have at least the information capacity of S_1;
3. if schema S_2 is used for updating a database specified using schema S_1 then S_1 and S_2 must have equivalent information capacity.

4.2 Schema Conversion for Molecular Biology Databases

Schema conversion is a key component of heterogenous database systems where it underlies the construction of *global views* of multiple databases expressed in some common data model or *local views* of multiple databases expressed in the local data model of each component database [12]. For example, the IGD system involves converting relational schemas into ACEDB schemas and then integrating these schemas into a global ACEDB schema.

Data management tools used for developing molecular biology databases also involve schema conversions. Thus, the EER, Genera, and OPM tools include translators that convert EER, Genera, and OPM schemas, respectively, into relational schemas. These tools also support reverse conversions from relational schemas to EER, Genera, and OPM schemas.

[8] This definition is based on the definition of information capacity in [7].

Other schema converters include a Genera to ACEDB converter that allows using ACEDB for viewing data stored in relational databases,[9] and a prototype schema converter from an early version of OPM to ACEDB [10].

Since the EER, Genera, and OPM tools provide object interfaces for browsing and querying entire relational databases, their schema translators must generate relational schemas that are information capacity equivalent with the object schemas they represent. Sometimes (e.g., when unnormalized relations are used for representing classes of objects) additional constraints are required for ensuring information capacity equivalence (e.g., see [14]).

Schema converters that do not preserve information capacity can lead to distorted (e.g., query) results [15]. However, the information capacity equivalence criterion can sometimes be too stringent [10]. For example, the reverse engineering or retrofitting object schemas on top of existing relational schemas sometimes adds information missing in (and thus semantically enriching) the converted relational schema, or ignores information that is not relevant for constructing the object schemas. This suggests that schema conversion needs to consider preserving information capacity regarding subsets of the schemas involved in the conversion. Further work is required for developing criteria for determining subsets that are 'relevant' to schema conversion. Furthermore, reliable schema converters cannot be developed without precise syntactic and semantic definitions for the data models targeted for conversion. Such definitions are not always available.

Acknowledgements. The author wants to thank Stan Letovsky for providing details and examples regarding the Genera data model, and Otto Ritter and Detlef Wolf for sharing their experience with ACEDB.

References

[1] Abiteboul, S., and Bonner, A., Objects and Views, Proc. of the 1991 *ACM SIGMOD* International Conference on Management of Data, SIGMOD Record 20, 2, 1991.

[2] Bancilhon, F., Delobel. C., and Kanellakis, P., Building an Object-Oriented Database System: The Story of O_2, Morgan Kaufmann Publishers, Inc., 1992.

[3] Chen, P.P., The Entity-Relationship Model- Towards a Unified View of Data, *ACM Transactions on Database Systems*, **1**, 1, (March 1976), pp. 9-36.

[4] Chen, I.A., and Markowitz, V.M., An Overview of the Object-Protocol Model (OPM) and OPM Data Management Tools, *Information Systems*, Pergamon Press, 1995; available together with additional documentation at `http://gizmo.lbl.gov/opm.html`.

[9] Developed by Stan Letovsky at GDB, Baltimore.

[10] Developed by Detlef Wolf the German Cancer Research Institute (DKFZ) in Heidelberg, Germany. This prototype was developed in 1993 and has not been updated to subsequent versions of OPM and ACEDB.

[5] Durbin, R., and Thierry-Mieg, J., Syntactic Definitions for the ACEDB Data Base Manager, 1992; available together with additional documentation at `http://probe.nalusda.gov:8000/acedocs/`.

[6] Hammer, M., and McLeod, D., Database Description with SDM: A Semantic Database Model, *ACM Transactions on Database Systems*, **6**, 3, (September 1981).

[7] Hull, R., Relative Information Capacity of Simple Relational Database Schemata, *SIMA Journal of Computing*, **15**, 3 (August 1986), pp. 856-886.

[8] Hull, R., and King, R., Semantic Database Modeling: Survey, Applications, and Research Issues, *Computing Surveys* **19**, 3 (September 1987).

[9] Karp, P., Report of the First Meeting on Interconnection of Molecular Biology Databases (MIMBD 94), Stanford, California, August 1994; `http://www.sri.ai.com/people/pkarp/mimbd/mimbd-94.html`.

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

[11] Letovsky, S., Genera: A Specification-Driven Web/Database Gateway Tool, unpublished manuscript; see also `http://gdbdoc.gdb.org/letovsky/genera/genera.html`.

[12] Markowitz, V.M., and Ritter, O., There's Never Time to Do It Right, But Always Time to Do It Over, 2nd Meeting on the Interconnection of Molecular Biology Databases (MIMBD), July 1995, Cambridge, United Kingdom; `http://gizmo.lbl.gov/DM_TOOLS/PAPERS/TNT/TNT.html`.

[13] Markowitz, V.M., and Shoshani, A., An Overview of the Lawrence Berkeley Laboratory Extended Entity-Relationship Database Tools, *Proc. of the 13th International Conference on the Entity-Relationship Approach*, 1994, available together with additional documentation at `http://gizmo.lbl.gov/DM_TOOLS/EER/eer.html`.

[14] Markowitz, V.M., and Shoshani, A., Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach, *ACM Transactions on Database Systems*, **17**, 3 (September 1992).

[15] 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.

[16] Programmer's Reference. National Center for Biotechnology Information, Bethesda, Maryland, November 1991.

[17] Sciore, E., Versioning and Configuration Management in an Object-Oriented Data Model, *VLDB Journal*, **3**, 1994, pp. 77-106.

[18] Tsichritzis, D.C., and Lochovsky, F.H., *Data Models*, Prentice-Hall, 1982.