Design of an Object-Oriented Database for Reverse Genetics

Krys J. Kochut [1,3] Jonathan Arnold [2] John A. Miller [1,3] Walter D. Potter [1,3]

[1]Department of Computer Science
University of Georgia
Athens, Georgia

[2]Department of Genetics
University of Georgia
Athens, Georgia

[3]Artificial Intelligence Programs
University of Georgia
Athens, Georgia

kochut@cs.uga.edu

Abstract

We present the design of an object-oriented database system for reverse genetics applications. Such a database will encapsulate not only the data in the genetic and physical maps, but also the methods used to create the maps as well as methods to link them to other databases, such as GenBank, PIR, and MedLine. The purpose of this database is to provide the fungal genetics community with an electronic tool for identifying the biochemical function of any DNA fragment in the database -- electronic reverse genetics. Such a tool for reverse genetics will enable researchers to identify the biochemical functions associated with genes encoding proteins in fungal development pathways, purine metabolism, the heat shock response, and molecular chromosome mechanics and evolution. Our initial goal is to apply the database for the genome mapping of the filamentous fungi, *Aspergillus nidulans* and *Neurospora crassa*, at the University of Georgia and the University of Leeds in England.

Introduction

Recently, a number of technical advances in molecular biology, such as cloning and sequencing DNA fragments, have resulted in a new approach to genetics. Where traditionally genetics has proceeded from a phenotype to a DNA fragment (gene), the new genetics with its molecular tools often proceeds in reverse: from an anonymous DNA fragment to its biochemical function (phenotype). The major tool for facilitating this process of reverse genetics is a physical map of a chromosome or chromosome fragment.

A physical map is a partial ordering of DNA fragments along a chromosome. The oldest kind of physical map is a cytological map provided by visible banding patterns along a chromosome, like in man. The most detailed physical map is the complete DNA sequence of an organism's genome. At this time a variety of technologies are being developed to map and sequence bacterial, fungal, plant, and animal genomes. One chromosome in yeast, for example, has been recently completely sequenced. A major unsolved problem in these initiatives is what to do with all the genome data and formulating how to make it accessible and usable by scientists and medical practitioners. Physical maps of chromosomes or chromosome fragments are one of the most important ways to organize molecular data and make it available in a useful format. Creating these maps involves the concerted efforts of an entire community of scientists studying a particular organism like man, and such a project must be carefully planned, orchestrated, and be highly automated because of the sheer size of the problem.

Let us assume that we have a collection (or library) of DNA fragments, which is essentially derived from several 'identical' copies of the human genome. Many of the fragments in this library will overlap. The experimental problem is to develop methods that will recognize these overlaps to assemble a map. The simplest kind of data that can be collected on each DNA fragment (or clone) is scoring for the presence or absence of a particular DNA marker. With a panel of such markers each DNA fragment can then be assigned a binary "call number" to link it to other DNA fragments and other genetic databases, like GenBank (the collection of all published DNA sequences), in the electronic search for a biochemical function. This is also the kind of mapping data which scales easily to large automated mapping projects.

We have elected to create a model information system for the genome mapping of the fungus *A. nidulans*, initially. It is a well developed model genetic system with over 400 known genes with an average interlocus spacing of less than 6 map units and with over 190 kilobases of DNA sequence already available (Timberlake 1991). It is more convenient and efficient to develop both new physical mapping methods and new database software tools in simpler systems, like *A. nidulans*, with small genomes (31 megabases (Brody et al. 1991)) and little repetitive DNA (Timberlake 1978). By providing a physical map for *A. nidulans* and *N. crassa* and an electronic tool for reverse genetics, this project will enable researchers to identify the biochemical functions associated with genes encoding proteins.

Related Research

Relational and other traditional record-oriented database systems are designed for the management of regular, for-

matted data, and provide excellent support for commercial data processing applications. However, the data models of these systems are not suited for many biological applications, such as restriction mapping, sequencing, or physical mapping. This is for the following reasons. First, the objects in the database, like DNA sequences in GenBank, can vary tremendously in size from less than 1 kilobase to over 170 kilobases (Epstein-Barr Virus) and an object-oriented database (OODB) structure can be much more compact and practical (Shin et al. 1992, Figure 8), although this has not precluded a relational database system being used to manage GenBank sequences (Cinkosky et al. 1991). Second, molecular mapping data contains complex objects (e.g. clonal restriction maps) involving hierarchical (is-a) relationships among DNA fragments. Third, the objects have behaviors that need to be recorded in the database and an OODB gives a coherent means to manage these methods and heuristics (rather than as an add on). Fourth, there are some simple and reasonable queries that simply cannot be constructed reliably under the relational model (Shin et al. 1992, p. 236). Fifth, an OODB allows the use of references to objects in the database so that when the object (e.g. clone) is renamed, there is no need to update the database, thereby lowering transaction costs.

Genome mapping initiatives will provide a new application domain for these new object-oriented database systems. However, very few Object-Oriented Databases for maintaining genetic information have been developed or are currently under development. Even less has been published in the form of technical papers, not to mention journal papers. Surveys of ongoing genome database projects appear in (Lander, Langridge, & Saccocio 1991, Frenkel 1991, Pearson et al. 1991, Erickson 1992). What follows is largely based on private communication (e-mail, letters).

ACeDB is a database for *C. elegans*, designed by Richard Durbin at MRC Laboratory for Molecular Biology, Cambridge, UK and Jean Thierry-Mieg at CNRS--CRBM, Montpellier, France. The ACeDB system is used by a number of laboratories around the world (not only for *C. elegans* -- there are sites using the system on *A. thaliana*). The authors of ACeDB do not classify their system as an OODBMS (ACeDB technical documentation), but some features of an object-oriented DBMS are supported. However, schema design/modification, concurrency control, and recovery mechanisms are not supported on a level required by a large database system. From our (limited) use of the system, we were impressed with its nice, user-friendly graphical interface.

The Worm Community System (WCS) is under development at the University of Arizona (Bruce Schatz et al.). The current version of the system has been in use at approximately 20 sites for about two years.

A Genetic Mapping Database for the mouse is under development at the Whitehead Institute at MIT (Nathan Goodman, Eric Lander et al.). Although there are no published papers on the design of the database, we obtained technical documentation (*Data Model* and *Reference Manual*) from Dr. Nathan Goodman. The Data Model suggests that this database is oriented more towards genetic mapping, as opposed to our model, which is more oriented towards physical mapping. The database is implemented in ObjectStore (by Object Design). ObjectStore, which is essentially a persistent object-oriented language based on C++, provides good efficiency, but schema specification is, we feel, more difficult (in comparison to, e.g. UniSQL/X (Kim 1992)) due to its low-level nature. ObjectStore is also used as the database engine in the SIGMA system (System for Integrated Genome Map Assembly), produced by the Human Genome Information Resource at Los Alamos National Laboratory (SIGMA. 1992).

Shin et al. (Shin et al. 1992) developed an object-oriented genome database for *E. coli*. Their database is implemented in the object-oriented database management system ONTOS (Andrews, Harris, & Sinkel 1989). Other related examples include databases for the nematode (Rowley & Rockland 1991), *E. coli* (Kazic et al. 1990), and protein hydrophobic cores (Kemp & Gray 1990).

Cuticchia et al. developed a Contig Mapping and Analysis Package (CMAP) (Cuticchia et al. 1992), which provides a foundation for computer-aided reverse genetics by organizing information about DNA fragments derived from an organism's genome into a physical map. The user can store a variety of information about a particular segment of DNA in a relational database (VAX/VMS *Rdb*). This information can be both descriptive, such as any genes contained in a particular DNA fragment, or experimental, such as the digital call numbers assigned to a particular clone (by one of a variety of experimental protocols).

However, since our goals call for the storage and retrieval of complex objects (genetic and physical maps, and images of gels and restriction maps) and for the integration of other computational methods (e.g., FASTA (Pearson & Lipman 1988), MAPMAKER (Lander & Green 1987), MAPSEARCH (Rudd et al. 1990, Rudd et al. 1991), and programs to identify coding regions, as well as simulated annealing), an object-oriented database is necessary for us to achieve our goals fully. In the subsequent sections we present design of a more integrated, wider-scope CMAP which is based on an object-oriented database.

Computer-Aided Genetic Engineering

The relational data model is not suited for many biological applications (such as restriction mapping, sequencing, or physical mapping) because it is more difficult and costly to represent these data as a collection of homogeneous tables in one integrated system (Shin et al. 1992). More importantly, a relational database cannot capture adequately the rules and heuristics in genetics,

relating the objects in the database, such as database searches, the physical map, and the methods to generate the physical map. Moreover, the physical and genetic maps are dynamic quantities, which evolve with the addition of clones, DNA sequences, and genes to the database. Lastly, the DNA fragments have spatial interrelationships, depending on which map is being considered. In this respect, there is an ongoing effort (with the development of ACeDB, OODB for the mouse, and the OODB for *E. coli* (Shin et al. 1992)) to demonstrate that object-oriented data modeling is conceptually better suited for biomatrix applications, such as creating a physical mapping database.

The physical mapping database is made up of spatially related clones along a chromosome as well as possibly their position along the genetic map. In addition, the clones may have their own restriction map and partial sequences, like an STS. These three maps and the associated mapping methods become organizing principles for the other information stored in STS, GenBank, and PIR databases. An aspect of the proposed physical mapping database that makes it more complex than a traditional database application, is the necessity of using special functions, like MAPMAKER, simulated annealing, or FASTA, on the attributes in the database. These functions perform integrated analyses of different types of data. The result of an analysis can be the production of a physical or genetic map or a list of homologous sequences.

Active KDL (Knowledge/Data Language) (Miller et al. 1990, Potter et al. 1990, Miller et al. 1991, Miller et al. 1991, Kochut et al. 1991, Kochut, Miller, & Potter 1991, Potter et al. 1993b, Potter et al. 1993a) is an object-oriented database system, which evolved from earlier work on KDL which has been ongoing since 1986 (Potter & Kerschberg 1986). The foundations of Active KDL are threefold: (i) object-oriented programming, (ii) functional programming, and (iii) hyper-semantic data modeling. These areas strongly influenced the design of Active KDL's three sublanguages: the schema definition language (SDL), the query language (QL), and the database programming language (DBPL). Because of the capabilities and elegance of these sublanguages, Active KDL is capable of supporting demanding information intensive applications, such as physical mapping. We now illustrate the application of the methodology to the development of a physical mapping database system.

Features of Active KDL

The following goals underline the design decisions that guided the development of Active KDL: (i) Active KDL consists of a *Schema Definition Language* (SDL), a *Query Language* (QL), and a *Database Programming Language* (DBPL). The database programming language is a superset of the query language. (ii) Active KDL supports object-oriented data modeling and is based on the following abstraction mechanisms: Classification/Instantiation, Specialization/Generalization, Aggregation, Association (or Membership), Knowledge (in the form of Constraints and Heuristics), and Concurrent/Temporal capabilities. (iii) The language is functional at the schema design level (SDL), the query level (QL), and at the application level (DBPL). The violation of strict referential transparency resulting from combining the functional and the object-oriented paradigms is precisely identified and kept to a minimum. (iv) The language is simple, yet powerful. Active KDL schema specification, queries, and application routines are compact and easy to understand and design. (v) When used as a query language, Active KDL is a *closed language*, that is, it is possible to use the result of a query as an argument to another query. This important closure property, which is a cornerstone of the relational model, is missing from many object-oriented database systems (Alashqur, Su, & Lam 1989). (vi) Active KDL supports active objects, that is, objects performing some task. (These facilities, not used here, are designed to support high concurrency (Miller & Griffeth 1991).)

```
object-type ::=
  OBJECT_TYPE class-name HAS
    [ SUPERTYPES: class-name { , class-name };]
    [ SUBTYPES:   class-name { , class-name }
                        [ HIDING function-list ];]
    [ ATTRIBUTES: { attribute-name: type-name
                        [ WITH CONSTRAINT: constraint ];}]
    [ MEMBERS:    { member-name:
                        [ SET OF | LIST OF ] class-name
                        [ INVERSE OF
                            member-name [(class-name)]]
                        [ WITH CONSTRAINT: constraint ];}]
    [ CONSTRAINTS:          { constraint; }]
    [ HEURISTICS: { rule; }]
    [ METHODS:    { method; }]
  END class-name;
```

Figure 1: Syntax of Object-Type Definition.

Object-types (or *classes*) form the foundation of Active KDL. An Active KDL database consists of a collection of objects, i.e. instances of object-types. Similar objects are grouped into object-types. An example of an object-type is Clone, shown as the first object type definition in Figure 2. This object-type consists of the set of all gene objects which are currently stored in GenBank. In general, an object (as a value) is an entity composed of other values whose types may be different. The syntax of an object-type definition is shown in Figure 1.

The optional characteristics of an object-type correspond to the KDM modeling primitives while the class-name uniquely identifies the object-type. Any object-type may be defined as a specialized (derived) form of one or more other object-types, called *supertypes*. As an example, in Figure 2, we present object type

definitions for *Clone, Contig, Chromosome,* and *Gene*, a fragment of the actual schema expressed in Active KDL (EER diagram is shown in Figure 3).

An object-type (or class) may be viewed as an encapsulation of functions. In Active KDL, there are four flavors of functions: (i) ATTRIBUTES, which are stored functions (if the attribute refers to an independently existing object(s) it is called a member and identified as such), (ii) CONSTRAINTS, which are Boolean functions, (iii) HEURISTICS, which are functions or rules expressed using the query language, and (iv) METHODS, which are quasi-functions (may have limited side-effects) expressed using the database programming language.

Physical Mapping Database

A physical mapping information system is composed of a database of DNA fragments stored as clones, sequence-tagged sites or STSs (Olson et al. 1989), DNA sequences, and genes. In addition, the system maintains connections to other standard databases, such as GenBank or the Genetic Map (Clutterbuck 1990). The purpose of providing these linkages is to provide a biochemical function for the DNA fragment, which may occur either by placing it on the genetic map or by identifying similar sequences in PIR, or by identifying a coding region, for example.

The knowledge about the genetic system is captured in KDL and reflects: (i) allowable values for the data (CONSTRAINTS); (ii) DEFINITIONS AND RULES about DNA fragments (such as genes are DNA fragments; genes may be covered by certain clones; clones make up contigs; contigs are contained on one chromosome); and (iii) METHODS for physical mapping (KDL can access the method of simulated annealing to update the physical map of a chromosome or place a new clone on the genetic map in order to identify its function). The database system can be directed to place a DNA fragment automatically on the physical and genetic maps as well as initiate database searches of PIR and GenBank using FASTA to gain insights on function. A strategy for identifying biochemical function is embedded in the KDL schema. Similar intelligent tools have been developed for shotgun sequencing, for example GENESIS (Friedland et al. 1982) and restriction maps (Shin et al. 1992).

Schema Design

The schema design was guided by the overall goal of providing a mapping workstation facilitating reverse genetics. The system we develop will be able to display genetic and physical maps of any organism stored in the database (our exemplar is *Aspergillus nidulans*). For example, a user, through simple query formulation or more typically through menu selection, will be able to display an overall genetic map of an organism's genome, zoom in on a chromosome and see the estimated positions of the known genes, zoom further to establish a clearer indication of relative distances and increase the labeling information shown on the known genes. Also the user will be able to superimpose RFLP (Restriction Fragment Length Polymorphism) and STS information on such displays.

```
OBJECT_TYPE Clone HAS
  // Instances of this class are clonable DNA fragments.
  SUPERTYPES:      DNA_Fragment;
  ATTRIBUTES:
      Index: STRING;                  // Clone Identifier
      Location: Storage_Library;
      Active: BOOLEAN;                // Contains viable clone?
      Date: STRING;                   // Date Clone was isolated
      Clone_Comment: Comment;         // Misc. clone information
      Repetitive: BOOLEAN;            // Has repetitive DNA?
      rRNA: BOOLEAN;                  // rRNA is coded for in clone?
      Centromeric: BOOLEAN;           // In the centromere?
      Telomeric: BOOLEAN;             // In a telomere?
      Nucleolar: BOOLEAN;             // In nucleolar organizer?
      Order_Num: INTEGER;             // Location on contig
      Clone_Size: INTEGER;            // Size of clone in kb
      Num_Probes: INTEGER;            // Number of oligo probes used

      // Hybridization profiles: if the clone hybridizes with the ith
      // (i = 1..30) oligonucleotide probe (DNA seq. of 9 to 12 bases)
      Oligo:  LIST OF BOOLEAN WITH CONSTRAINT
                Oligo_Size (c: Clone): BOOLEAN =
                  Size (Oligo (c)) = Num_Probes (c);

      // Hybridization profiles: if the clone hybridizes with the ith
      // (i = 1..8) chromosome, rated as High, Medium, or Low
      Chromo_Specific: LIST OF (H, M, L) WITH CONSTRAINT
                CS_Size (c: Clone): BOOLEAN =
                  Size(Chromo_Specific (c)) =
                    Num_Of_Chromosomes (From_Organism (c);
  MEMBERS:
      Known_Genes: SET OF Gene;  // Known genes within clone
      On_Contig: Contig;         // Localized within Contig
      From_Organism: Organism;   // Source Organism
  CONSTRAINT:
      Index_Check (c: Clone): BOOLEAN =
        Index (c) = SubLibrary (Location (c)) + Plate (Location (c))
              + Row (Location (c)) + Column (Location (c));
  METHODS:
      Match_Gene (c: Clone): Gene;
      // Return the gene (if there is one) whose STS matches the clone's.

END Clone;
```

Figure 2: Fragment of the Active KDL Schema.

Eventually, having exhausted the resolution capabilities of the genetic map, the user may delve deeper by switching to a physical map. A switch to the corresponding physical map is achieved by correlating the end points of the genetic map shown in the display with the corresponding positions in the physical map. Once this is done, an overlapping sequence of clones will be displayed in the appropriate window. Depending on the level of resolution at which the switch occurs, details such as contained genes, STSs, etc. may be displayed with the clones.

```
OBJECT_TYPE Contig HAS
  // Instances of this class are contiguous sections of chromosome which
  // result from piecing the clones back together.
  ATTRIBUTES:
      Contig_Name: STRING;
      Contig_Size: INTEGER;        // Size of contig in kb
      Gap_Size: INTEGER;           // Number of kb to next contig
  MEMBERS:
      Clones: LIST OF Clone INVERSE OF On_Contig;
      On_Chromosome: Chromosome;
  HEURISTICS:
      Current_Clone (k: Contig): Clone = Current(Clones (k));
      Next_Clone (k: Contig): Clone = Next(Clones (k));
      Prior_Clone (k: Contig): Clone = Prior(Clones (k));
  METHODS:
      Set_Current (k: Contig; c: Clone);
END Contig;

OBJECT_TYPE Chromosome HAS
  // Instances of this class are chromosomes.
  ATTRIBUTES:
      Chromosome_Num: INTEGER WITH CONSTRAINT
      Number_Check (c: Chromosome): BOOLEAN =
              Chromosome_Num (c) IN
                {1 .. Num_Of_Chromosomes (From_Organism (c))};
  MEMBERS:
      From_Organism: Organism INVERSE OF Genome;
      Genes: LIST OF Gene INVERSE OF On_Chromosome (Gene);
      Contigs: LIST OF Contig INVERSE OF
              On_Chromosome (Contig);
      RFLPs: LIST OF RFLP;
  HEURISTICS:  Clones (c: Chromosome): LIST OF Clone =
                      Clones (Contigs (c));
END Chromosome;

OBJECT_TYPE Gene HAS          // Instances of this class are genes.
  SUPERTYPES: DNA_Fragment;
  ATTRIBUTES:
      Name: STRING;
      Gap_Size: INTEGER;          // Number of kb to next gene
  MEMBERS:
      On_Chromosome: Chromosome;
      Affected_Trait: Trait;
      Generated_Polypeptides: SET OF Polypeptide;
END Gene;
```

Figure 2 (cont.): Fragment of the Active KDL Schema.

At this point the user may get a bigger picture by unzooming (e.g., to see what clones are in the neighborhood or to see how the clones form a contig or how contigs and gaps represent a chromosome), or the user may look at finer and finer detail by zooming (e.g., to display the internal structure of a single clone along with any textual information about the clone, or possibly even to display the nucleotides in some subsequence of the clone if the clone has been sequenced). Each display will have a default annotation or labeling (e.g., name of a clone), but additional information is easily obtained by moving to the border of the window where clicking a mouse button will cause a menu to pop-up. The menu will allow the relevant facts about the object (or objects) in the window to be accessed (e.g., the date on which the clone was isolated).

To provide such mapping capabilities, a substantial amount of information about a variety of different kinds of objects need to be stored. The object-types of greatest interest are:

1. DNA_Fragment -- a segment of DNA;
2. Clone -- a DNA fragment stored in a cloning vector, e.g. a phage;
3. Contig -- a contiguous sequence of overlapping clones;
4. Chromosome -- the object of in vitro reconstruction;
5. Gene -- a fragment of DNA with specific functions;
6. GB_Gene -- a DNA fragment with its sequence known and characterized sufficiently well to be included in GenBank;
7. Genetic_Map -- an ordered sequence of genetic loci along chromosomes with map distances between nearest neighbors (in *A. nidulans*, all map distances are uncorrected and they are the product of two- and three-point crosses between nearest neighbors (Clutterbuck 1990));
8. Physical_Map -- for example, a sequence of clones ordered by their position along the chromosome.

A diagrammatic representation of the schema is given in terms of an Enhanced Entity-Relationship (EER) Diagram, see Figure 3. Figure 3 displays the object-types (or entity-types) as rectangles, relationships between object-types as diamonds, and specialization inheritance hierarchies with bold is-a arcs. The labels on the lines extending from the diamonds indicate the cardinality constraints (e.g., many-to-many, many-to-one, or one-to-one). This high level EER design can be elaborated in the form of an Active KDL schema specification. The EER diagram is shown in Figure 3, and a fragment of the corresponding Active KDL schema is shown in Figure 2.

Queries and Updates

The objects in the genetics database can be retrieved by using the query language (presented to the scientist through the graphical user interface). Several of the more common queries can be captured as heuristics or methods, so that only the name and a list of parameters need to be given. For example, to display a physical map of chromosome number VI of *Aspergillus nidulans* in a window, using Active KDL syntax one could simply enter the following query (the query could be accomplished by pointing and clicking in the graphical user interface or by a scrolling menu bar and menu boxes as in CMAP (Cuticchia et al. 1992)):

```
FOR ALL p IN Physical_Map
  WHERE Genus (p) = "Aspergillus" AND
        Species (p) = "nidulans"
```

```
    EVAL Display_Chromo (p, 6)
END;
```

Most of the work done by this query is accomplished by the Display_Chromo method. This method will display the clones as very short line segments that go together to form contigs. A group of contigs and gaps constitute the chromosome. The display produced by this Display_Chromo method would contain several hundred clones displayed as very short overlapping line segments pieced together to form contigs.

Using X-Windows, we will be able to (in a very portable way) produce multiple simultaneous displays, for example, a genetic map of *A. nidulans* in one window, its corresponding physical map in another, and a zoomed section of the physical map in still another. As an additional enhancement to ease-of-use many of the most common queries may be initiated by clicking a mouse button to select an option from a menu.

A typical scenario for a geneticist using the CMAP system might proceed as follows (Bates et al. 1991): The system will come up with an initial help window. Moving the mouse to the border and clicking a button will cause the main menu to pop-up. Option one is to display a genetic map. The user is then asked to chose between the entire genome or a particular chromosome. If the entire genome is chosen, an overall display of all (or initial few) chromosomes is produced. On the other hand, if a particular chromosome is chosen, then a more detailed genetic map will be displayed that shows the estimated positions of genes. This display can also be produced by selecting a chromosome from an overall display and chosing the zoom option in this window's pop-menu.

Let us suppose at this point that the user wishes to see a physical map of the first contig of chromosome number VI. Assuming the user knows the contig's approximate extent within this chromosome, he/she can select this part of the display by clicking the mouse button near both ends of the contig (the selected portion is now highlighted). Next, pressing the display physical map option in the pop-up menu will produce a detailed display of this contig. To see more detail within the contig, i.e., the clones and their associated identifiers, this display will need to be zoomed. If part of the display was not selected, then the entire display is zoomed. Since this is too much information to fit within the window, only the first portion of the contig is actually shown. The rest may be seen by dragging the position maker at the bottom of the window.

Finally, let us suppose the user is interested in seeing a single clone and learning information about it such as contained genes. From a window showing a single chromosome he/she may select one of the clones (e.g., L67A07) which will then be highlighted, and then chose the zoom option in this window's pop-up menu. This will produce a display of clone L67A07 along with some of its associated information. Besides the graphical display oriented queries, textual information may be retrieved either in association with a graphical display or independently. Suppose that a physical map is currently being displayed in a window, and that this map has been zoomed to the point of containing just a few clones. One of the options in the pop-up menu for this window will be to list the known genes contained within the displayed clones. An example of a query that is independent of graphical displays is the following:

```
FOR ALL c IN Clone
    WHERE Oligo (c) >= {1, 1, 1, 0 .. 0}
    APPLY Index (c)
END;
```

This query will retrieve the index (clone identifier) of all clones which at least hybridize with the first three oligonucleotide probes.

A large portion of our database maintenance will be devoted to updating the data. A common update that will need to be performed is the following: "Add new clones to the physical map". Once sufficient information on a clone (or clones) has been obtained (i.e., values for some of its attribute values are known), then it may be entered into the database. Our user interface will facilitate entry of this information by popping up a blank template for the user to fill in attribute values. Of vital importance, are the relationships (or associations) that this new clones objects will have with other objects in the database. Using the Oligo (representing oligonucleotide) attribute information, the Reconstruct method within the Physical_Map object-type, can be executed to automatically reform the map, given the new clones. We reconstruct the map by applying simulated annealing. Note that even when adding a single clone, there is no efficient procedure to guarantee an optimal solution, since if this were possible, it would be easy to construct a polynomial algorithm for the entire problem which is known to be NP-hard. In (Cuticchia, Arnold, & Timberlake 1992), various heuristic techniques are compared, and our simulated annealing approach is found to provide excellent map reconstruction. Additional types of updates and integrity constraints on updates may be found in (Cuticchia et al. 1992) and will be added to the object-oriented database to check for inconsistencies in the data. Another important aspect to these relationships between objects is in cross-validation of data of heterogeneous quality and sources. For example, a contig map can be cross-validated by a chromosome walk (Cuticchia et al. 1992). Queries will be constructed exploiting these relationships to assist in the maintenance of a self-consistent genome database so that the database manager and user can routinely detect problematic data. In addition, we are already developing procedures (Fu, Timberlake, & Arnold 1992), which will provide, for example, confidence levels on the links in a physical map, and this kind of inference information will

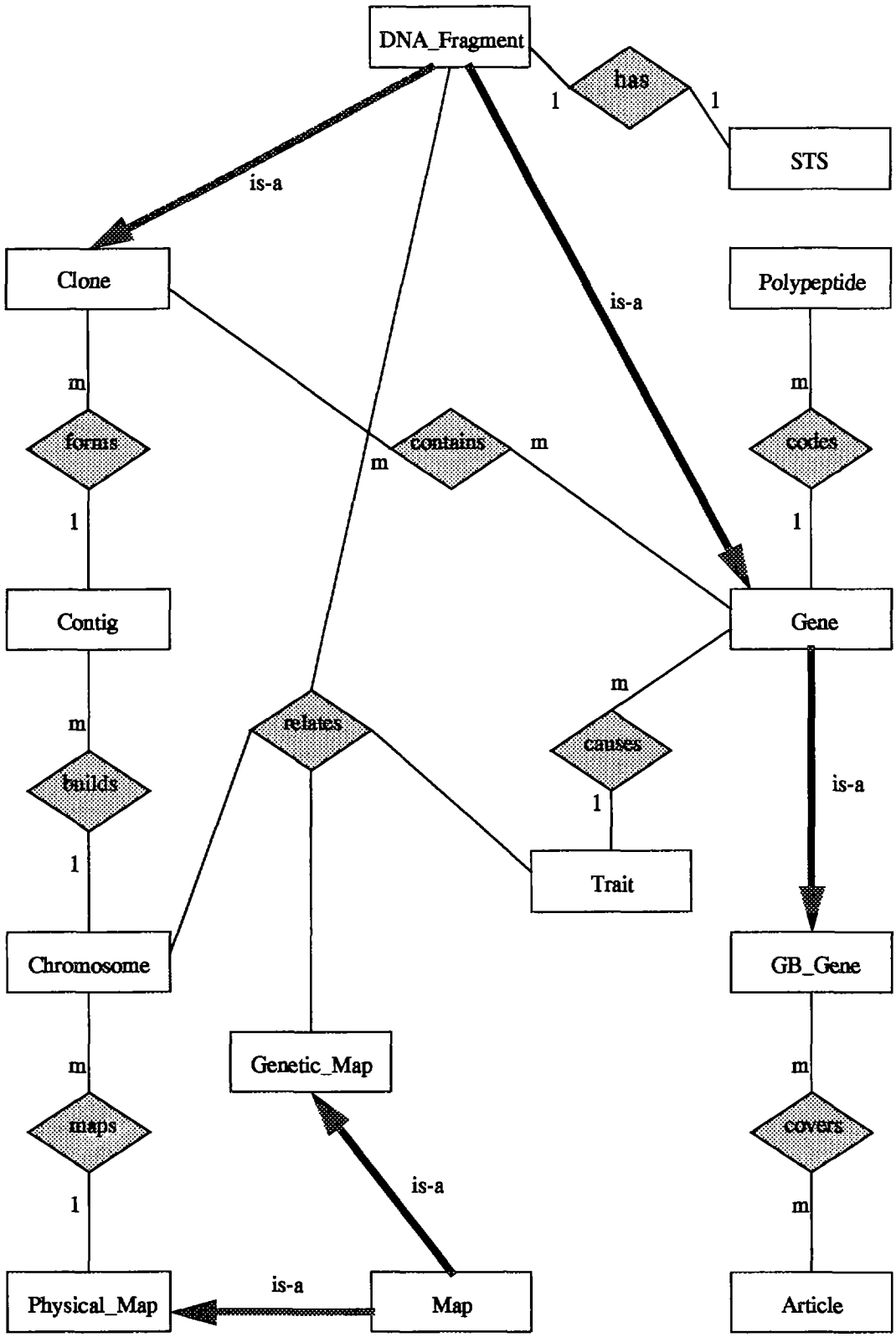

Figure 3: EER Diagram.

be a part of what is presented to the scientist.

Conclusions

We have presented the design of an object-oriented database system for reverse genetics. Our database encapsulates not only the data in the genetic and physical maps, but also the methods used to create the maps as well as methods to link them to other databases, such as GenBank, PIR, and MedLine. The database will provide the fungal genetics community with an electronic tool for identifying the biochemical function of any DNA fragment in the database -- electronic reverse genetics.

In the design we have used Active KDL as a tool for designing an object-oriented database. Active KDL provides an array of modeling primitives and it proved to be particularly good for designing demanding database applications, such as *genome mapping*. Our design can be regarded as an argument supporting the strengths of Object-Oriented Database Systems in modeling complex, non-business applications.

Our initial goal is to apply the database for the genome mapping of the filamentous fungi, *Aspergillus nidulans* and *Neurospora crassa*. Likely benefits of this project include providing a simple model for the more complex initiative of mapping the small flowering plant, *Arabidopsis thaliana* and the initiative of mapping *Homo sapiens*.

References

Alashqur, A.M., S. Su, & H. Lam 1989. OQL: A Query Language for Manipulating Object-Oriented Databases, *Proc. Fifteenth Int. Conf. on Very Large Databases*, Amsterdam .

Andrews, T., C. Harris, & K. Sinkel 1989. The ONTOS Object Database, *Ontologic Technical Report*, Burlington, MA .

Bates, G.P., M.E. MacDonald, S. Baxendale, S. Youngman, C. Lin, W.L. Whaley, J.J. Wasmuth, J.F. Gusella, & H. Lehrach 1991. Defined physical limits of the Huntington Disease Gene Candidate Region,, *American J. of Human Genetics 49* , 7-16.

Brody, H., J. Griffith, A.J. Cuticchia, J. Arnold, & W.E. Timberlake 1991. Chromosome-specific recombinant DNA libraries from the fungus *Aspergillus nidulans*, *Nucleic Acids Res. 19*(11) , 3105-3109.

Cinkosky, M.J., J.W. Fickett, P. Gilna, & C. Burks 1991. Electronic Data Publishing and GenBank, *Science 252*(5010) , 1273-1277.

Clutterbuck, A.J. 1990. *Aspergillus nidulans. Genetic Maps: Locus Maps of Complex Genomes, Fifth Edition*, Cold Spring Harbor Laboratory Press, Plainview, N.Y.

Cuticchia, A.J., J. Arnold, & W.E. Timberlake 1992. Use of simulated annealing for chromosome reconstruction experiments based on binary scoring, *Genetics 132* , 591-601.

Cuticchia, A.J., J. Arnold, H. Brody, & W.E. Timberlake 1992. CMAP: Contig mapping and analysis package: a relational database for chromosome reconstruction, *CABIOS 8* , 467-474.

Erickson, D. 1992. Hacking the Genome, *Scientific American April 1992* , 128-137.

Frenkel, K.A. 1991. The Human Genome Project and Informatics, *Communication of the ACM 34*(11) .

Friedland, P., L. Kedes, D. Brutlag, Y. Iwasaki, & R. Bach 1982. GENESIS: a knowledge-based genetic engineering simulation system for representation of genetic data and experiment planning, *Nucleic Acids Res. 10*(1) , 323-340.

Fu, Y.X., W.E. Timberlake, & J. Arnold 1992. On the design of genome mapping experiments using short synthetic oligonucleotides, *Biometrics 48* , 337-359.

Kazic, T., E. Lusk, R. Olson, R. Overbeck, & S. Tuecko 1990.Prototyping databases in Prolog, in *The Practice of Prolog*, Sterling, L. ed., MIT Press, Cambridge, MA., .

Kemp, G.J.L. & P.M.D. Gray 1990. Finding hydrophobic microdomains using an object-oriented database, *CABIOS 6*(4) , 357-363.

Kim, W. 1992. On Unifying Relational and Object-Oriented Database Systems, *European Conference on Object-Oriented Programming ECOOP '92*, Utrecht, The Netherlands , 1-18.

Kochut, K.J., J.A. Miller, & W.D. Potter 1991. Design of a CLOS Version of Active KDL: A Knowledge/Data Base System Capable of Query Driven Simulation, *Proc. 1991 AI and Simulation Conf.*, New Orleans, LA .

Kochut, K.J., J.A. Miller, W.D. Potter, & A.D. Wright 1991. h-KDL: A Historically Extended Functional Object-Oriented Database System, *Proc. Tools '91 Int. Conf.*, Santa Barbara, CA .

Lander, E.S. & P. Green 1987. Construction of multilocus genetic linkage maps in humans, *Proc. National Academy Sci., USA 84*, , 2363-2367.

Lander, E.S., R. Langridge, & D.M. Saccocio 1991. Mapping and Interpreting Biological Information, *Communication of the ACM 34*(11) .

Miller, J.A., W.D. Potter, K.J. Kochut, & O.R. Weyrich

1990. Model Instantiation for Query Driven Simulation in Active KDL, *Proc. 23rd Annual Simulation Symp.*, Nashville, TN .

Miller, J.A., W.D. Potter, K.J. Kochut, A.A. Keskin, & E. Ucar 1991. The Active KDL Object-Oriented Database System and Its Application to Simulation Support, *J. of Object-Oriented Programming (Special Issue on Databases) 4*(4) .

Miller, J.A. & N.D. Griffeth 1991. Performance Modeling of Database and Simulation Protocols: Design Choices for Query Driven Simulation, *Proc. 24th Annual Simulation Symp.*, New Orleans, LA .

Miller, J.A., K.J. Kochut, W.D. Potter, E. Ucar, & A.A. Keskin 1991. Query Driven Simulation Using Active KDL: A Functional Object-Oriented Database System, *Int. J. in Computer Simulation 1*(1) .

Olson, M.V., L. Hood, C. Cantor, & D. Botstein 1989. A common language for mapping of the human genome, *Science 245* , 1434-1435.

Pearson, W.R. & D.J. Lipman 1988. Improved tools for biological sequence analysis, *Proc. National Academy of Sci., USA 85*, , 2444-2448.

Pearson, P.L., B. Maidak, M. Chipperfield, & R. Robbins 1991. The Human Genome Initiative -- Do databases reflect current progress, *Science 254* , 200-225.

Potter, W.D. & L. Kerschberg 1986. A Unified Approach to Modeling Knowledge and Data, *Proc. IFIP TC2 Conf. on Knowledge and Data (DS-2)*, Algarve, Portugal (Published by North-Holland as *Data and Knowledge DS-2* (1988).) .

Potter, W.D., J.A. Miller, K.J. Kochut, & S.W. Wood 1990. Supporting an Intelligent Simulation/Modeling Environment Using the Active KDL Object-Oriented Database Programming Language, *Proc. 21st Annual Pitt. Conf. on Simulation and Modeling*, Pittsburgh, PA .

Potter, W.D., T.A. Byrd, J.A. Miller, & K.J. Kochut 1993. Extending Decision Support Systems: The Integration of Data, Knowledge, and Model Management, *Annals of Operations Research 38* , 501-527.

Potter, W.D., K.J. Kochut, J.A. Miller, V.P. Gandham, & R.V. Polamraju 1993. The Evolution of the Knowledge/Data Model, *Advances in Databases and Artificial Intelligence* (to appear), Petry & Delcambre (eds.), .

Rowley, S. & C. Rockland 1991. The design of simulation languages for systems with multiple modularities, *SIMULATION 56*(3) , 153-163.

Rudd, K.E., W. Miller, J. Ostell, & D.A. Benson 1990. Alignment of *Escherichia coli* K12 DNA sequences to a genomic restriction map, *Nucleic Acids Res. 18*(2) , 313-321.

Rudd, K.E., W. Miller, C. Werner, J. Ostell, C. Tolstoshev,, & S.G. Satterfield 1991. Mapping sequenced *E. coli* genes by computer: software, strategies and examples, *Nucleic Acids Res. 19*(3) , 637-647.

SIGMA. 1992. *System for Integrated Genome Map Assembly. User Manual, Version 0.70*, Human Genome Information Resource, Center for Human Studies, Los Alamos National Laboratory.

Shin, D.-G., C. Lee, J. Zhang, K.E. Rudd, & C.M. Berg 1992. Redesigning, implementing and integrating *Escherichia coli* genome software tools with an object-oriented database system, *CABIOS 8*(3) , 227-238.

Timberlake, W.E. 1978. Low repetitive DNA content in *Aspergillus nidulans*, *Science 202* , 973-975.

Timberlake, W.E. 1991. Molecular genetics of *Aspergillus* development, *Ann. Rev. Genetics 24* , 5-36.