LBL-31493
UC-605

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

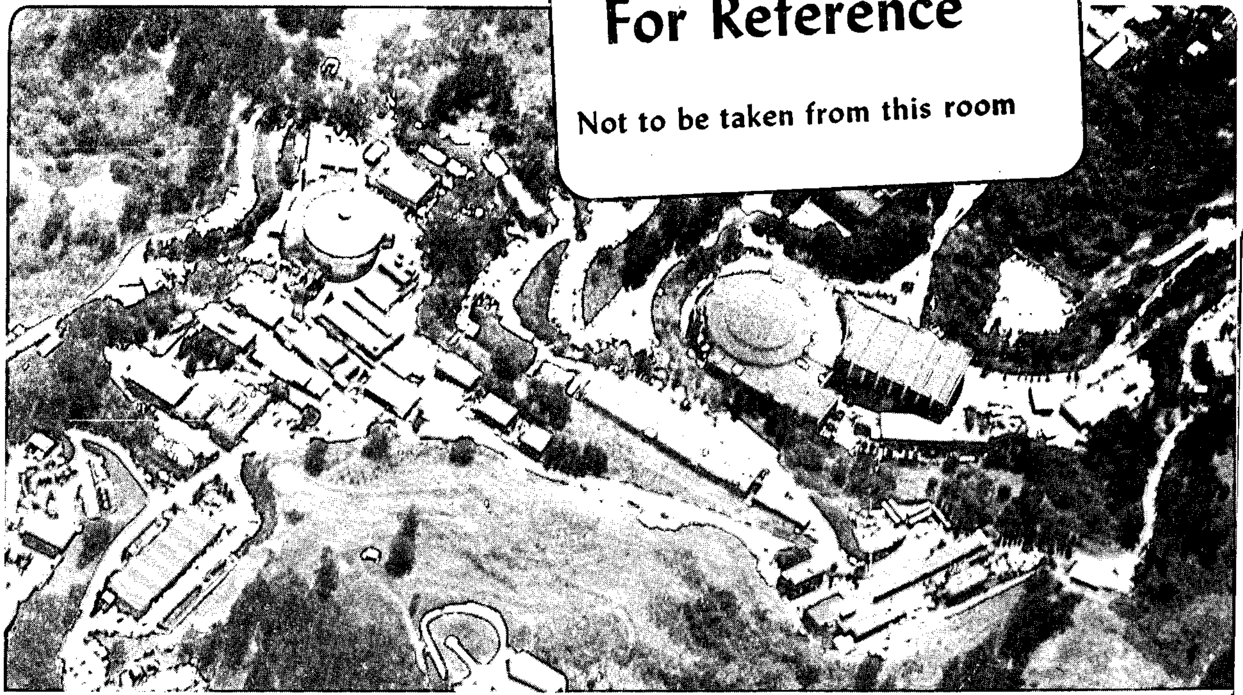
Information and Computing Sciences Division

Presented at the Sixth International Conference on Scientific Statistical Database Management, Ascona, Switzerland, June 9-12, 1992, and to be published in the Proceedings

Data Management for Genomic Mapping Applications: A Case Study

V.M. Markowitz, S. Lewis, J. McCarthy, F. Olken, and M. Zorn

May 1992



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

LBL-31493

DATA MANAGEMENT FOR GENOMIC MAPPING APPLICATIONS : A CASE SUDY*

Victor M. Markowitz[1]
Suzanna Lewis[2]
John McCarthy[3]
Frank Olken[4]
Manfred Zorn[5]

Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

May 1992

To be published in the Proceedings of the
6th International Conference on Scientific and Statistical Database Management
hold at Ascona, Switzerland, June 9-12, 1992

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

[1] Author's E-mail address: *VMMarkowitz@lbl.gov* Office phone number:(510) 486-6835
[2] Author's E-mail address: *SELewis@lbl.gov* Office phone number:(510) 486-7370
[3] Author's E-mail address: *JLMcCarthy@lbl.gov* Office phone number:(510) 486-5307
[4] Author's E-mail address: *F_Olken@lbl.gov* Office phone number:(510) 486-5891
[5] Author's E-mail address: *MDZorn@lbl.gov* Office phone number:(510) 486-5041

Data Management for Genomic Mapping Applications: A Case Study [*]

Victor M. Markowitz, Suzanna Lewis, John McCarthy, Frank Olken, and Manfred Zorn

Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road, Berkeley, CA 94720

Abstract

In this paper we describe a new approach to the construction of data management systems for genomic mapping applications in molecular biology, genetics, and plant breeding. We discuss the architecture of such systems and propose an incremental approach to the development of such systems. We illustrate the proposed approach and architecture with a case study of a prototype data management system for genomic maps.

1. Introduction

1.1 Background

The information controlling the development of biological organisms (bacteria, yeast, plant, human) is encoded in the *genome* of the organism. This information is encoded in polymeric molecules known as DNA, comprising two complementary strands. In each strand, information is encoded as a sequence of *nucleotides* which constitute a four letter alphabet, denoted as A,C,T,G. The complementarity of the two opposing strands (A paired with T, C paired with G) permits replication of the DNA molecule during cell division.

Regions of the DNA, known as *genes* specify the information for protein molecules. Some proteins are structural components of the cell, while others catalyze chemical reactions. Other DNA sequences control the expression of the genes, code for other cell products, constitute structural components of the DNA, or have no known functions.

In higher organisms (yeast, plants, animals, humans) the DNA is organized into several linear chromosomes. Humans have 22 pairs of homologous chromosomes (one from each parent), plus an (X, X)

[*] Issued as technical report LBL-31493. This work was supported by the Office of Health and Environmental Research Program and the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.

pair for women and an (X, Y) pair for men. Gametes (sperm and eggs) have only half the number of chromosomes (one of each pair). During gamete formation, *recombination* of pairs of chromosomes sometimes occurs, so that the gamete may inherit chromosomes which are spliced from pieces from each pair. Genes which are located close to each other on chromosomes tend to be inherited together.

It is possible to determine the chromosome number and ordinal position of a gene on the chromosome by cytological studies and by *genetic mapping*. In genetic mapping, patterns of co-inheritance of genes are analyzed in order to estimate the frequency of recombination events and thus infer the distances between and order of the genes (*markers*) on the chromosome. This analysis permits determining the approximate location (to within 1 or 2 million nucleotides) of disease genes without knowledge of the gene sequence. Genetic maps are also used to facilitate plant breeding (corn, wheat, soybeans, pine trees, tomatoes).

There are several projects currently underway that attempt to determine the complete DNA sequence of various organisms (E. coli, C. elegans, yeast, mice, humans, etc.). In humans, chromosomes range from 50 million to 250 million nucleotides. Since existing technology permits sequencing only fragments of a few hundred nucleotides, chromosomal DNA is cut into smaller pieces, the pieces are propagated as clones, and the order of the pieces on the chromosome is reconstructed by *physical mapping*. Other landmarks, such as chromosome banding patterns, known genes, and the sites at which restriction enzymes cut (such enzymes recognize specific subsequences) can also be placed on the physical map.

In this paper we use the term *genomic maps* to denote genetic, physical, and other types of chromosomal maps. Examples of genetic and physical maps are shown in figure 1.1. The remainder of this paper is devoted to the development of systems for managing genomic maps and related data.

1.2 Data Management for Genomic Mapping Applications

Information regarding DNA sequences, markers, and maps must be maintained and made available to molecular biologists for analysis and various manipulations. The growing amount and complexity of genomic map and sequence data requires database management systems (DBMSs) for effective storage, access, control, and manipulation of the data. However, DBMSs do not directly support the data view expected by molecular biologists. Consequently, a data management system for genomic mapping applications has a two-level architecture consisting of an *application* level for defining the application-specific data structure, and supporting application-specific operations, and a *database* level for storing the application data and providing various data management facilities (see figure 2.2). The application and database levels are related by mappings between the application-specific data structure and operations and the database structure and manipulations, respectively.

In this paper we present a modular and incremental approach to the development of data management systems for genomic mapping applications. We claim that the development of mappings between the application-specific and database levels can be accelerated by adding a third, intermediate, level for decomposing mappings into simpler mappings. We propose using the *Extended Entity-Relationship* (EER) model for this intermediate level. This choice allows immediately using existing EER to DBMS mapping tools.

We illustrate our approach to the development of data management systems for genomic mapping applications by describing a prototype *Chromosome Information System* (*CIS*) developed at Lawrence Berkeley Laboratory [2]. *CIS* follows the architecture outlined above. The current version of *CIS* includes data on various types of genomic maps (e.g. cytogenetic, genetic, physical, radiation), on genomic markers, such as probes and loci, and on related reference information, such as citations. Molecular biologists interact with *CIS* through a graphical user interface that makes the underlying database transparent. Queries for retrieving maps, for example, can be formulated by selecting a region on a

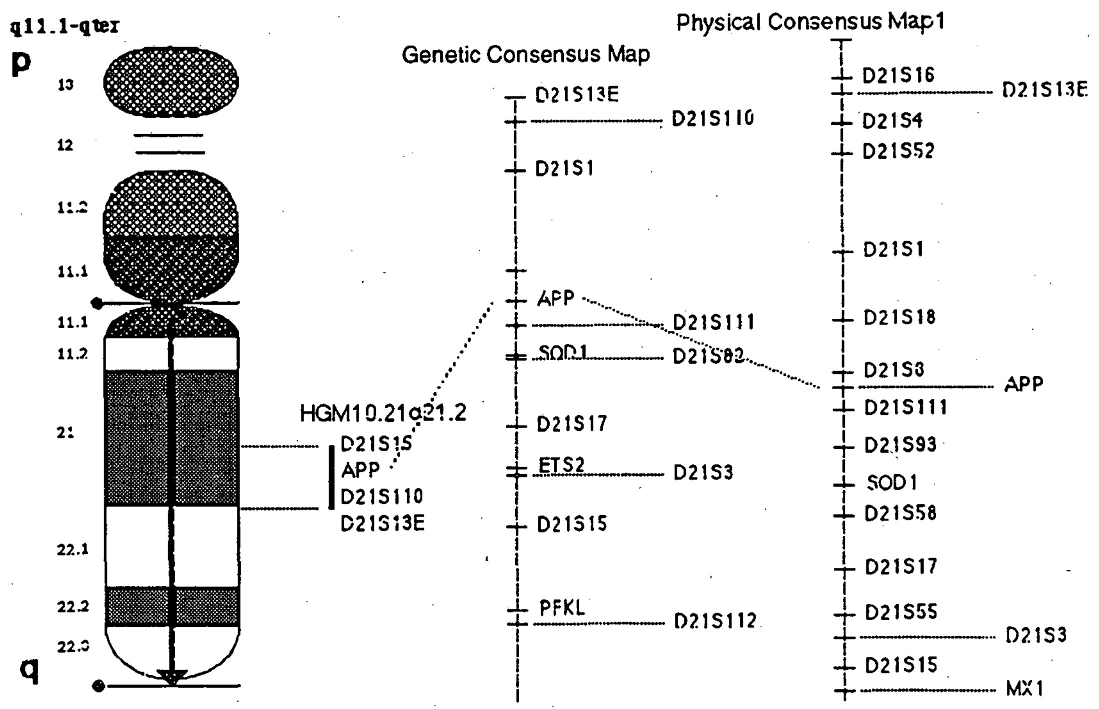


Figure 1.1 Examples of Genetic and Physical Maps.

graphical representation of a chromosome shown on the screen or by choosing the desired map attributes from a list of available attributes. The current version of the ***CIS*** database is implemented on Sun workstations using the Sybase DBMS [10].

The paper is organized as follows. In section 2 we present the basic architecture of a data management system for genomic mapping applications, and discuss the main aspects of specifying the structure of genomic map data. In section 3 we propose extending the basic architecture of a data management system for genomic mapping applications in order to facilitate its implementation, and discuss using the EER model for describing the structure of genomic mapping applications. The implementation of ***CIS*** is described in section 4. We conclude the paper by discussing future extensions of the ***CIS*** prototype.

2. Data Management for Genomic Mapping Applications

In this section we briefly discuss the structure of *genomic maps*, and examine the main components of a data management system for genomic mapping applications.

2.1 Genomic Maps

Genomic maps and related data can be viewed as consisting of *objects* characterized by (having) *attributes* (see figure 2.1). Objects sharing common attributes are considered to have the same nature (type), and are *classified* in *object classes* (homogeneous sets of objects). For example, *genomic maps* are objects that share common attributes, such as *Name*, *Type* (e.g. cytogenetic, physical), etc, and therefore can be classified in a class of objects called *Map*, as shown in figure 2.1. Attributes can be

(a) *atomic* (e.g. *Type*, *Name*, *Title*), or *composite*, that is, consisting of an aggregation of atomic attributes (e.g. (*Locus* , *Location*);

Object	***Map***	***Locus***	***Citations***
Attributes	*Type*	*Type*	*Title*
	Name	*Name*	*Publication*
	{ (*Locus* , *Location*) }	{ *Probe* }	*Publisher*
	{ *Citation* }	{ *Citation* }	{ *Authors* }
	. . .	. . .	. . .

Figure 2.1 Examples of Genomic Map Objects and Attributes.

(b) *single-valued* (e.g. *Title*), or *set-valued* (e.g. { *Citation* }, { (*Locus* , *Location*) });

(c) *local* (e.g. *Name*, *Title*), or *referential*, that is, attributes that can be used as references to other objects (e.g. attribute *Locus* of *Map* can be used as a reference to *Locus* objects).

Relationships between classes of objects are implied as follows: a class of objects is related to another class of objects, if one class of objects is associated with a *referential* attribute referencing objects in the other class of objects.

Regarding the relationship between objects and their attributes, attributes can be *optional* (i.e. null values/empty sets are allowed) or *mandatory* (i.e. null values/empty sets are not allowed). Attribute values are defined over *domains*, have certain *formats*, and may be required to satisfy certain restrictions (e.g. must fall within specific ranges).

Object classes and attributes are associated with (i) a collection of *binary comparison operators*, which includes at least an operation for testing equality of two object instances; and (ii) a collection of *constructive operations* (such as arithmetic operation for numeric attribute values), which can be empty. Examples of more complex comparison operators and constructive operations for genomic maps are *map alignment* and *map construction*, respectively.

2.2 Basic Architecture of a Data Management System for Genomic Mapping Applications

Developing data management systems for genomic mapping applications requires various mechanisms, such as languages for specifying data structures and manipulations, mechanisms for efficient data access, facilities allowing concurrent data access by multiple users, and mechanisms for enforcing the validity and integrity of data. Such mechanisms are usually provided by database management systems (DBMSs). However, commercially available DBMSs do not directly support the data structure and operations specific to genomic mapping applications.

In order to resolve the discrepancy between the constructs of a DBMS and the application-specific data structure and operations, a data management system for genomic mapping applications should have the basic two-level architecture shown in figure 2.2:

1. an *application* level contains the application-specific data definition (schema), and supports application-specific operations;

2. a *database* level contains the application data structured according to a database definition (schema) using constructs supported by the underlying DBMS, and supports DBMS-specific data manipulations.

The application and database levels are related by a *schema translation* mapping the application-specific schema into a database schema, an *operation translation* mapping application-specific operations into database operations, and a *data conversion* mapping data structured according to the database definition into data structured according to the application-specific schema.

Users interact with the data management system via a *user interface* that provides a *schema specification* facility for defining the application-specific data structure, a *data presentation* facility for displaying application data in a suitable (e.g. graphical) format for users, and an *operation specification* facility for specifying application-specific operations, such as selection, browsing, and editing of genomic maps.

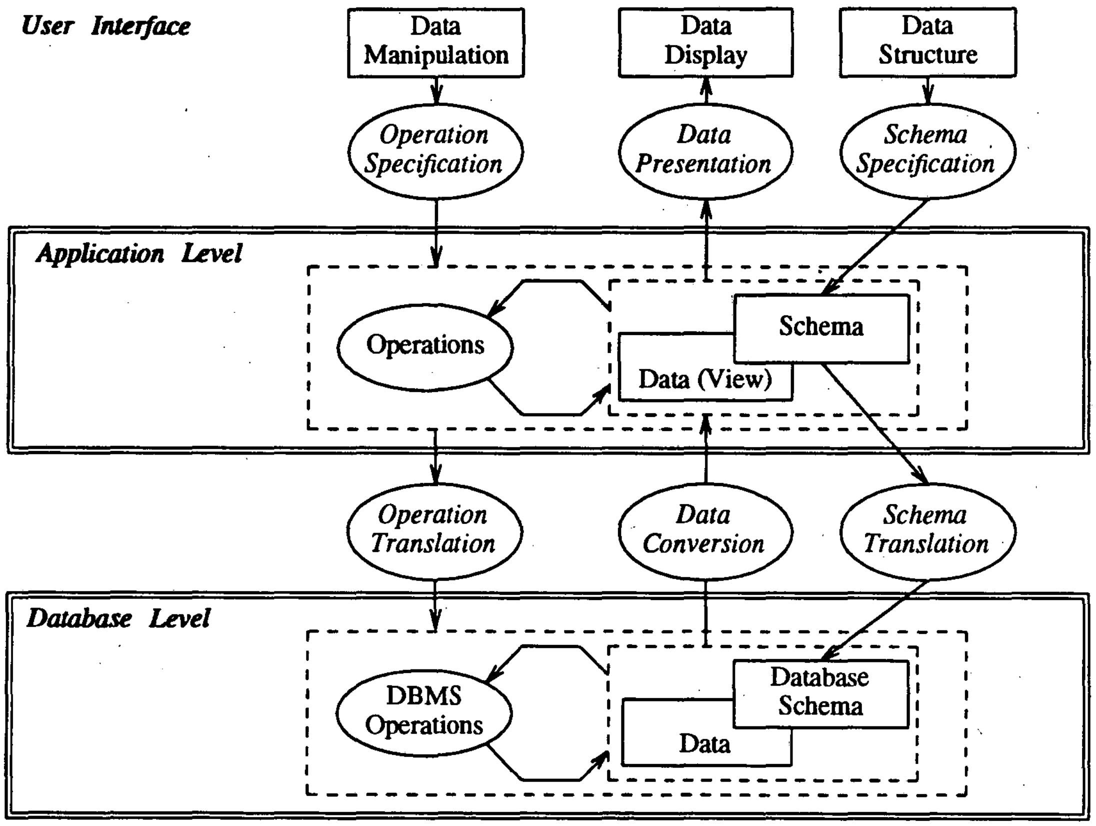


Figure 2.2 Basic Architecture of a Data Management System for Genomic Mapping Applications.

3. Developing a Data Management System for Genomic Mapping Applications

Databases developed with commercial DBMSs have very large definitions (schemas). For the Sybase DBMS, for example, a database consisting of tens of relations (tables) requires thousands of lines of code for defining tables, indexes, and procedures for maintaining the integrity of data. Moreover, the customary database design process for a DBMS is very confusing to non-expert users mainly because DBMS constructs obscure the semantics of an application with low-level technical details. Similarly, manipulating (i.e. retrieving and updating) data in databases involves the development of numerous procedures. Developing and maintaining database definitions and procedures is a tedious, error prone, and time consuming process. These problems lead to the development of tools that facilitate the database definition process by supporting design methodologies based on data models that have more semantic intuition, such as the *Entity–Relationship* (ER) and *Extended Entity–Relationship* (EER) models.

3.1 Extended Architecture of a Data Management System for Genomic Mapping Applications

We propose to extend the basic architecture of a data management system for genomic mapping applications presented in section 2, by introducing an intermediate level between the application and database levels, as shown in figure 3.1; we propose using for this intermediate level an *Extended Entity–Relationship* (EER) model. The EER level allows decomposing the operation and schema translations between the application and database levels into simpler translations between the application and EER levels, and the EER and database levels, respectively. Decomposing the operation and schema translations has several advantages. First, EER schemas, which are described in terms of objects and object connections, are inherently more concise, simpler to specify, and simpler to comprehend than DBMS schemas. Accordingly, expressing (translating) application-specific data structures and operations using EER constructs and operations is significantly simpler than using DBMS constructs and manipulations. Molecular biologists can learn to understand and critique a database design expressed in EER terms, whereas few can do so for detailed DBMS specifications. Second, EER schemas can be readily implemented over commercial DBMSs using tools that can automatically carry out the EER to DBMS schema translation. Automatic translation from concise EER specifications to more verbose DBMS definitions is faster and less error-prone than developing DBMS definitions directly, and allows quick schema changes during design. Moreover, developing an EER schema is independent of a specific DBMS, therefore it is possible to transfer EER schemas developed for a specific (e.g. relational) DBMS to future (e.g. object-oriented), DBMSs.

3.2 Describing Genomic Map Structures Using the Extended Entity–Relationship Model

The EER model is a version of the popular ER model proposed in [1]. Compared to the ER model, the EER model has two additional constructs, generalization and full aggregation [4]. For the sake of brevity, we only sketch the definitions of the constructs of the EER model; detailed definitions can be found in [4] and in reviews such as [8].

In the EER model, the structure of an application is described using an EER schema, and can be

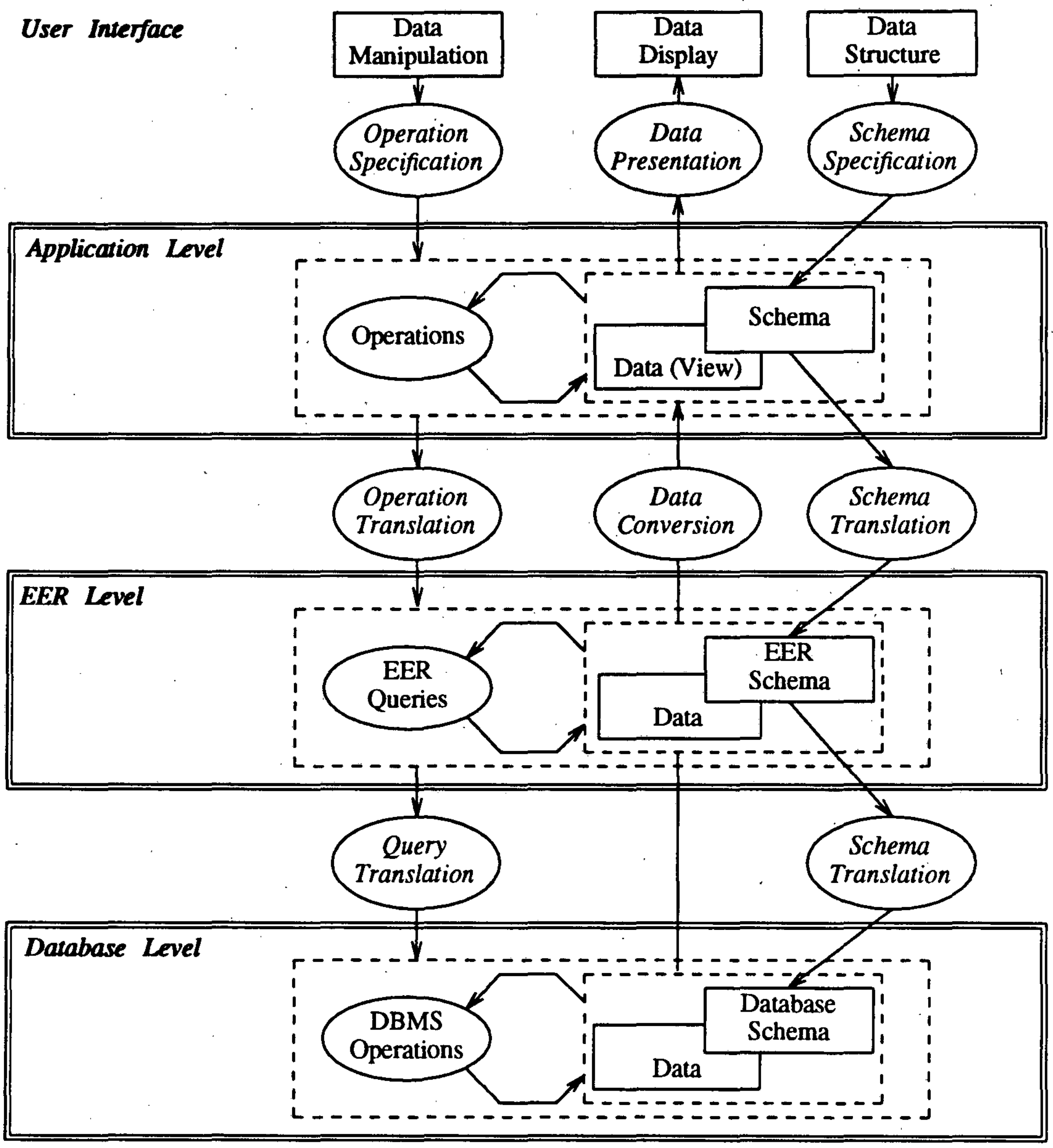


Figure 3.1 Extended Architecture of a Data Management System for Genomic Mapping Applications.

represented graphically in a diagrammatic form such as that shown in figure 3.2. An EER schema consists of specifications of atomic types of objects called entity-sets (e.g. MAP, PROBE, LOCUS, CITATION, AUTHOR), and associations of entity-sets called relationship-sets (e.g. CONSISTS OF, WRITTEN BY). Individual entity-set and relationship-set instances (i.e. individual entities and relationships) are qualified by *attributes* (e.g. NAME, TYPE, TITLE). Entity-sets, relationship-sets, and attributes are represented diagrammatically in an EER schema by rectangle, diamond, and ellipse shaped vertices; relationship-sets are connected by arcs to the object-sets they associate, and entity-sets and relationship-sets are connected by arcs to their attributes. Generalization is an abstraction mechanism that allows viewing a set of specialization entity-sets (e.g. PROBE, LOCUS) as a single *generic* entity-set (e.g. MAP OBJECT). The attributes and relationships which are common to the entity-sets that are generalized (e.g. NAME) are then associated only with the generic entity-set. A specialization entity-set (e.g. PROBE) inherits the attributes of all its generic entity-sets (e.g. PROBES inherits NAME from REFERENCED OBJECT). Generalization is represented diagrammatically in an EER schema by arcs labeled *ISA*, connecting the specialization entity-sets to their generic entity-sets.

The object and attribute concepts of the EER model are closer to the concepts naturally employed by molecular biologists to describe the structure of their applications. However, the EER object and attribute concepts are too restricted for directly describing application-specific (e.g. genomic map) objects and attributes. For example, the EER model does not directly support set-valued and composite attributes As discussed in [4], such restrictions are essential for an accurate mapping of EER schemas into DBMS schemas, and they can be overcome by using *auxiliary* EER objects. Auxiliary EER objects do not represent application-specific objects, and are used for specifying structures that are not directly supported in the

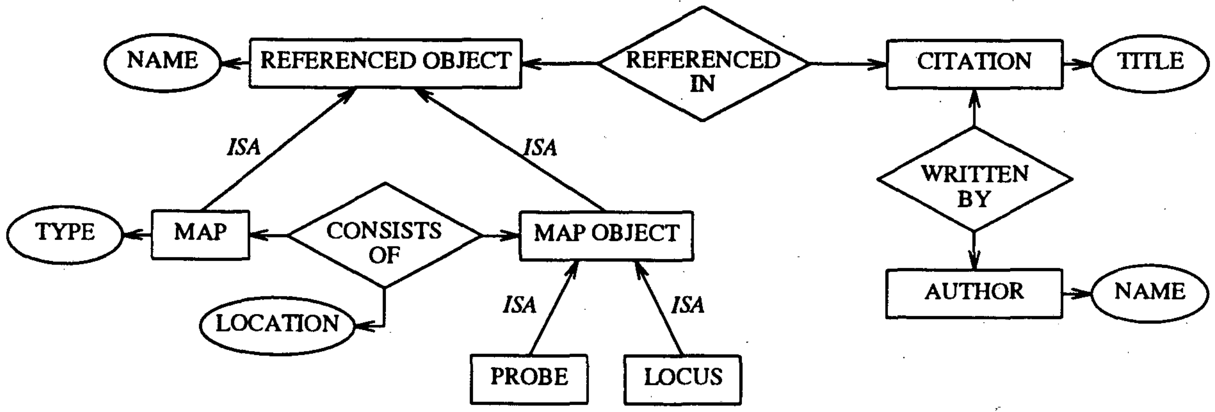


Figure 3.2 An EER Schema Example.

EER model. For example, consider class of objects *Map* shown in figure 2.1. *Map* can be represented by an EER entity-set, MAP, as shown in the EER schema of figure 3.2. However, in order to represent that a *Map* object involves sets of *Probe* objects, as well as *Locus* objects (each *Locus* at a certain *Location*), an auxiliary relationship-set, CONSISTS OF, and an auxiliary generic entity-set, MAP OBJECT, are needed, where MAP OBJECT generalizes the entity-sets representing *Probes* and *Locus*, respectively. Similarly, in order to represent that *Map*, *Probe*, and *Locus* objects are associated with *Citation* objects, an auxiliary generic entity-set, REFERENCED OBJECT, and an auxiliary relationship-set, REFERENCED IN, are needed, where REFERENCED OBJECT generalizes the entity-sets representing *Map*, *Probe* and *Locus*, respectively.

Using multiple (including *auxiliary*) EER object-sets for representing a single application-specific object implies that data regarding an (instance of an) application-specific object is usually scattered among multiple EER object-sets. Consequently, the EER representation of an application-specific structure must be complemented with operational definitions specifying the construction of application-specific objects from multiple instances of EER object-sets. A language that can be used for such definitions is the *Concise Object Query Language* (COQL) defined in [5] and [7]. For example, consider class of objects *Map* shown in figure 2.1, and suppose that *Map* is represented as shown in figure 3.2. Then the EER structural representation of *Map* can be complemented with the following COQL query:

```
OUTPUT          MAP: NAME, TYPE, NAME OF LOCUS, LOCATION OF CONSISTS_OF, TITLE OF
                CITATION;

CONNECTIONS     MAP CONSISTS_OF LOCUS; MAP REFERENCED_IN CITATION;
```

This query associates every MAP entity with the value of its (local) attribute TYPE, the value of the inherited attribute NAME, with sets of pairs of LOCUS NAME and LOCATION values, and with sets of CITATION TITLE values.

4. The *Chromosome Information System*

In this section we describe a prototype data management system for a genomic mapping application developed at Lawrence Berkeley Laboratory, the *Chromosome Information System* (*CIS*).

4.1 The Architecture of the *Chromosome Information System*

The architecture of ***CIS*** follows the three-level architecture described in section 3:

1. in ***CIS*** the *user interface* and its underlying application-specific level are embedded in a module called ***GenomeGuide***;
2. the ***CIS*** *database* is called ***GenomeBase***, and is currently implemented with the Sybase DBMS [10];
3. the intermediate level is based on the EER model.

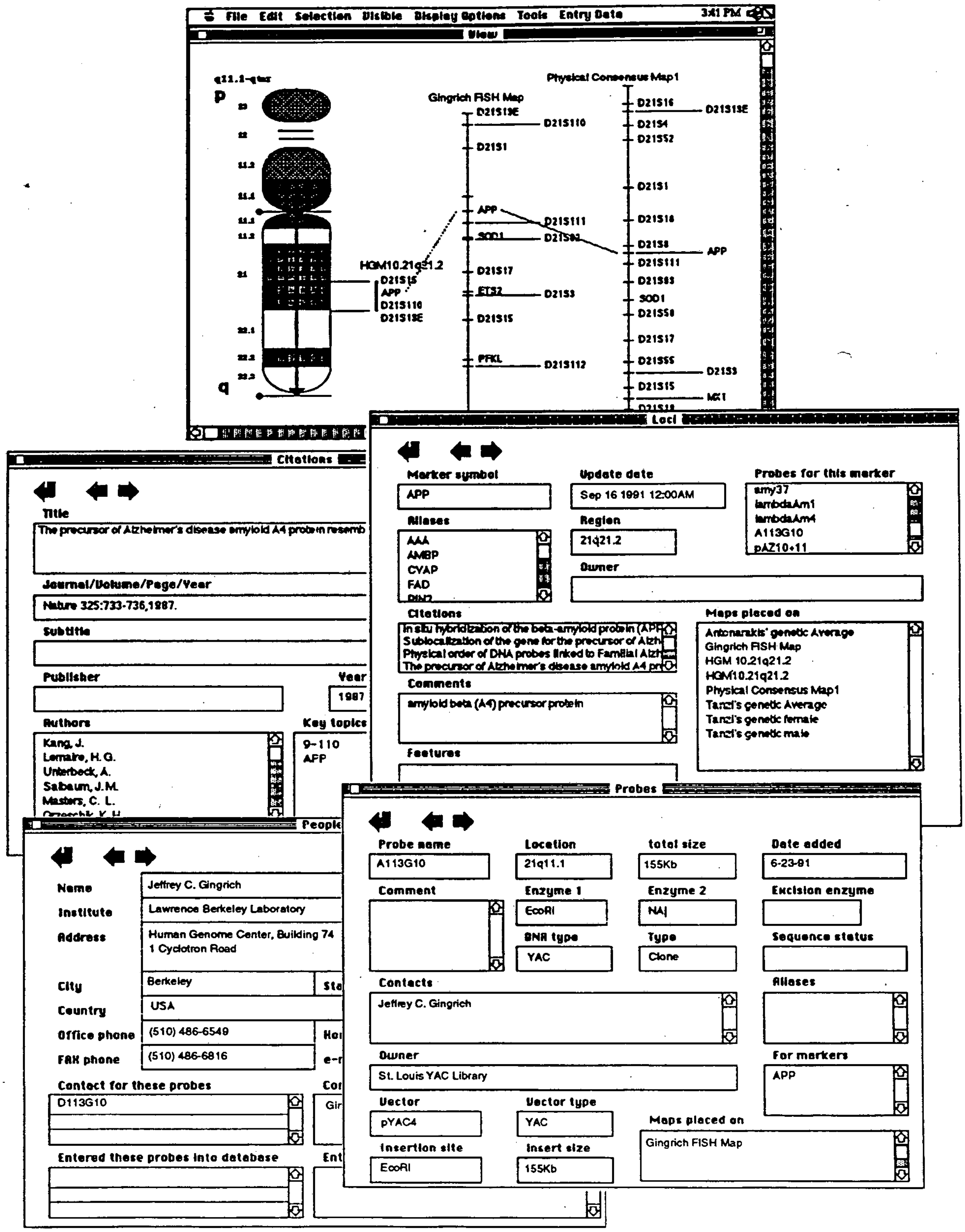
CIS involves mappings for translating the EER schema into the ***GenomeBase*** schema, for translating EER queries into Sybase (*Transact-SQL*) procedures for ***GenomeBase***, and for converting data in ***GenomeBase*** into data structured for ***GenomeGuide***. Currently, only the EER to Sybase schema translation is fully automated, while the generation of the SQL procedures is manual, and the data conversion is hard-wired in the ***GenomeGuide*** module.

4.2 The User Interface and Application Level : *GenomeGuide*

GenomeGuide supports a direct manipulation graphical user interface. Information is displayed via map diagrams, symbols, on-screen textual forms, and pull-down or pop-up menus, in multiple *windows* on a workstation screen. Color, shading, and typography are used to highlight different types of information. Users can move a *mouse* or another pointing device to different points on the screen, use the mouse to select items of interest or menu options, and reposition or resize windows.

Figure 4.1 illustrates how *GenomeGuide* currently accesses and presents diverse types of ***CIS*** information on a monochrome monitor. There are currently five major types of windows (see the top bar labels): *View*, *Citations*, *People*, *Loci*, and *Probes*. The initial *View* window presented to a user (top) includes a meta-phase cytogenetic map (left) for reference.

Users retrieve maps of interest by choosing the desired map type(s) (e.g genetic, physical), by scrolling through lists of map names (within each type), or by specifying criteria such as specific probes, loci, or regions of interest. This example shows three types of maps for Chromosome 21: a regional map, a


File Edit Selection Visible Display Options Tools Entry Data 3:41 PM
View
q11.1-qter
p
q
Gingrich FISH Map
Physical Consensus Map1
HGM10.21q21.2
D21S13E D21S110 D21S1 APP D21S111 SOD1 D21S93 D21S17 ETS2 D21S3 D21S15 PFKL D21S112
D21S16 D21S4 D21S52 D21S1 D21S18 D21S8 D21S111 D21S93 SOD1 D21S58 D21S17 D21S55 D21S3 D21S15 MX1
D21S15 APP D21S110 D21S13E
Loci
Citations
People
Probes

Loci

Marker symbol	APP
Update date	Sep 16 1991 12:00AM
Probes for this marker	amy37; lambdaAm1; lambdaAm4; A113G10; pAZ10+11
Aliases	AAA; AMBP; CVAP; FAD; DIN2
Region	21q21.2
Owner	
Citations	In situ hybridization of the beta-amyloid protein (APP; Sublocalization of the gene for the precursor of Alzh; Physical order of DNA probes linked to Familial Alzh; The precursor of Alzheimer's disease amyloid A4 pr
Comments	amyloid beta (A4) precursor protein
Features	
Maps placed on	Antonarakis' genetic Average; Gingrich FISH Map; HGM 10.21q21.2; HGM10.21q21.2; Physical Consensus Map1; Tanzi's genetic Average; Tanzi's genetic female; Tanzi's genetic male

Citations

Title	The precursor of Alzheimer's disease amyloid A4 protein resemb
Journal/Volume/Page/Year	Nature 325:733-736,1987.
Subtitle	
Publisher	
Year	1987
Authors	Kang, J.; Lemaire, H. G.; Unterbeck, A.; Salbaum, J. M.; Masters, C. L.; Grzeschik, K. H.
Key topics	9-110; APP

People

Name	Jeffrey C. Gingrich
Institute	Lawrence Berkeley Laboratory
Address	Human Genome Center, Building 74; 1 Cyclotron Road
City	Berkeley
Country	USA
Office phone	(510) 486-6549
FAX phone	(510) 486-6816
Contact for these probes	D113G10
Entered these probes into database	

Probes

Probe name	A113G10
Location	21q11.1
total size	155Kb
Date added	6-23-91
Comment	
Enzyme 1	EcoRI
Enzyme 2	NAI
Excision enzyme	
DNA type	YAC
Type	Clone
Sequence status	
Contacts	Jeffrey C. Gingrich
Aliases	
Owner	St. Louis YAC Library
For markers	APP
Vector	pYAC4
Vector type	YAC
Maps placed on	Gingrich FISH Map
Insertion site	EcoRI
Insert size	155Kb

Figure 4.1 Major Types of Windows of the *CIS GenomeGuide* User Interface.

genetic map, and a physical map.

Users can open additional windows for more detailed information for an item simply by clicking on that item. In figure 4.1, for example, the user *clicked* on the locus name *APP* shown on the maps to open the *Loci* window (middle right), for more detailed information about the *APP* locus. Similarly, *clicking* on the fourth line of the *citations* list for the *APP* locus opens the *Citations* window (middle left) for the article entitled "The precursor ..." . The *Probes* and *People* windows (lower right and left) were opened successively by *clicking* on the probe name *A113G10* in the *APP Loci* window and then the contact name *Jeffrey C. Gingrich* in the Probe window for *A113G10*. In this way users navigate through the information, moving easily by clicking from one piece of information to another.

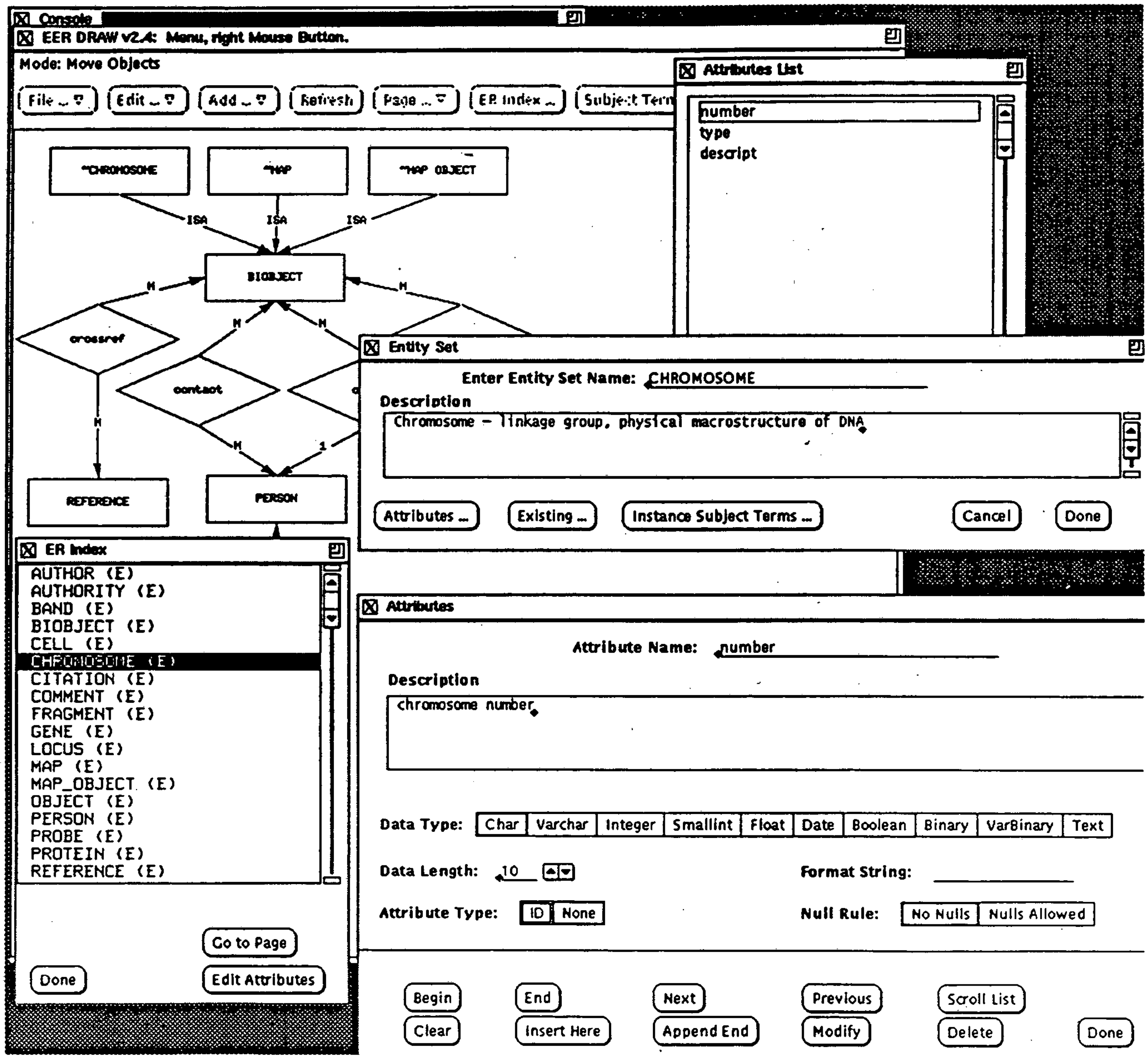
Regarding the editing capabilities of *GenomeGuide*, map objects are currently defined by entering their component attributes into a window form. Once map objects have been created, maps are built up by editing the scrolling list of objects that comprise the map. A user edits such lists by entering map object selections and then using *insert*, *append* and *delete* buttons to change the list. Once the map list is satisfactory, the user can require to draw the map by clicking on a *draw map* button. All entered or edited information is subsequently stored in the ***CIS*** database.

4.3 The EER Level : *ERDRAW*

The structure of the ***CIS*** application has been specified using a graphical EER schema editor called ***ERDRAW*** [9]. Different types of genomic map objects, as well as their interrelationships, have been analyzed and described using an EER schema. This schema contains different types of entities (e.g., maps, loci, probes, persons, and bibliographic citations), relationships between entities (e.g., citations can pertain to one or more loci), and attributes that describe each type of entity (such as author and title for citations). The present EER representation of maps accommodates any kind of linear map (e.g., cytogenetic, genetic, and physical maps). Naming conflicts are avoided by allowing synonyms for objects. References (e.g, a literature citation or a pointer to another database), and persons (e.g., owners or contact persons), can be associated with any genomic map object. The EER schema for ***CIS*** is described in [11].

Figure 4.2 shows several windows illustrating a typical ***ERDRAW*** session. After loading an existing EER diagram for editing, the user browses objects shown in the *ER Index* window. Once a required object-set is found (e.g., CHROMOSOME), the page of the EER diagram where this object appears is brought onto the screen. Next, the user enters a description for the CHROMOSOME object-set in the *Entity Set* window, and chooses the *Attributes* button to enter a description for the *number* attribute of CHROMOSOME. The *Attribute List* window displays other attributes of the CHROMOSOME object-set.

We are presently developing a *Query Specification and Resolution* (*QSR*) tool, the query counterpart of ***ERDRAW***. ***QSR*** is based on a *Concise Object Query Language* (COQL) that allows specifying concise queries in terms of EER objects and attributes [5]. As explained below, currently the operations supported by ***GenomeGuide*** are carried out by executing SQL procedures. ***QSR*** will allow developing these procedures at a higher (EER) level of abstraction. Using ***QSR*** will provide many of the same benefits outlined above for ***ERDRAW***, including simplicity, understandability, reduced coding and vendor independence.



Enter Entity Set Name: CHROMOSOME

Description

Chromosome - linkage group, physical macrostructure of DNA

AUTHOR (E)
AUTHORITY (E)
BAND (E)
BIOBJECT (E)
CELL (E)
CHROMOSOME (E)
CITATION (E)
COMMENT (E)
FRAGMENT (E)
GENE (E)
LOCUS (E)
MAP (E)
MAP_OBJECT (E)
OBJECT (E)
PERSON (E)
PROBE (E)
PROTEIN (E)
REFERENCE (E)

Attribute Name: number

Description

chromosome number

Data Type: Char | Varchar | Integer | Smallint | Float | Date | Boolean | Binary | VarBinary | Text

Data Length: 10

Format String:

Attribute Type: [✓ ID] None

Null Rule: No Nulls | Nulls Allowed

Figure 4.2 Major Types of Windows of the ***ERDRAW*** EER Schema Editor.

4.4 The Database Level : *GenomeBase*

The ***CIS*** database, ***GenomeBase***, was developed with the the Sybase DBMS using a *Schema Definition and Translation* (***SDT***) tool [6]. ***SDT*** automatically translates EER schemas into complete DBMS schema definitions for several relational DBMSs: Sybase, Informix, Oracle, DB2, and Ingres. Designing ***GenomeBase*** at the EER level and using ***SDT*** for automatically generating the Sybase definition for ***GenomeBase*** increased the productivity and accuracy of the database definition process, and allowed implementing quickly schema changes during design. In addition, ***SDT*** generates procedures (*triggers* in Sybase) for maintaining referential integrity constraints. Thus, while the EER specification for ***GenomeBase*** is about 7 pages of EER diagrams (150 lines of text specifications), its complete Sybase schema definition is about 3500 lines long.

Access to specific maps and even individual data items in ***GenomeBase*** can be restricted to the owner of the data only, to a specified group of users, or extended to the general public, thus facilitating data sharing, while ensuring privacy for certain kinds of information. Similarly, ***GenomeBase*** contains procedures that control which users can add, edit, and delete individual maps and data items.

4.5 Connecting *GenomeGuide* with *GenomeBase*

Operations supported by ***GenomeGuide*** are carried out by executing SQL procedures on ***GenomeBase***. As an expedient for the initial version of ***CIS***, a library of hand-coded procedures (implemented using Sybase's *Transact-SQL*) currently assembles complex genomic map objects from the corresponding tables, columns and rows representing them in the relational database. Implementing these procedures is time consuming and error-prone, especially as database schema changes inevitably occur.

We have currently completed the first version of a *Query Interpretation and Translation* (***QIT***) tool. ***QIT*** is the query translator counterpart of ***SDT***, and translates EER-based COQL queries into SQL queries [7]. We plan to use ***QIT*** to generate automatically the SQL procedures required for performing the ***GenomeGuide*** operations, and thus replace our current hand-coded procedures. This will increase the productivity and accuracy of procedure development, allow rapid adaptation of procedures to schema changes, and ensure DBMS independence.

In the current version of ***CIS***, the mappings between genomic map structures and operations in ***GenomeGuide***, and the database definition and manipulations of ***GenomeBase*** via the intermediate EER schema and queries, are specified manually. We plan to develop tools that will allow automatic mapping of genomic map structures and operations into EER schemas and queries. Coupled with ***SDT*** and ***QIT***, these tools will provide a complete mapping path between the application and database levels.

5. Summary and Future Development

We have proposed in this paper a three-level architecture for data management systems for genomic mapping applications. We have described a prototype data management system that follows this architecture, the ***Chromosome Information System*** (***CIS***). The initial ***CIS*** prototype incorporates information about chromosomes, maps, markers, probes, sequences, and bibliographic citations. The principal features of ***CIS*** include a direct manipulation, what-you-see-is-what-you-get user interface, ***GenomeGuide***, a separate but closely coupled, shared database, ***GenomeBase***, and high-level (EER) data modeling to represent genomic maps and related data. Database design at a high level of abstraction allows modeling genomic maps more accurately and concisely than with DBMS constructs, and insulates the application data structure definition from any particular DBMS.

Our first implementation of ***GenomeGuide*** has been done in Supercard (a Hypertalk clone) on a MacIntosh, chosen because of its support for rapid prototyping of graphical user interfaces. Communication with the Sybase-based ***GenomeBase*** on Sun is done via Sybase's HyperDB-LIB software on MacIntosh, using a TCP/IP protocol over Ethernet.

ERDRAW [9] and ***SDT*** [6] were developed at LBL; both are written in C and run on Sun workstations under Unix. ***ERDRAW*** uses the X-window Xlib and Xview toolkit. ***SDT*** currently generates relational schema definitions for Sybase 4.0, DB2, Ingres 6.3, Oracle 6.0, and Informix 4.0 DBMSs.

The initial version of ***GenomeGuide*** stimulated suggestions and detailed requirements from users, but was limited in flexibility, speed, and portability. To address these issues, we are re-implementing ***GenomeGuide*** on a Sun/Unix platform, using a standard windows toolkit (X-windows), a high-level graphical object management system (DataViews), and a graphical user interface builder (X-designer) to specify text windows and dialogs. The new version will also include enhancements for zooming in on specified regions for more detail, and ability to graphically manipulate map objects so that users can effectively arrange maps using a graphic representation provided for each map object. ***GenomeGuide*** will thus behave like CAD software, where changes in the graphic representation are immediately reflected in the corresponding numeric data representation and vice versa.

CIS already incorporates information that molecular biologists require from various external databases (e.g. *GenBank*, *GDB*, etc.). Although many of these databases are accessible via high speed computer networks, they often reside on different hardware, with different operating and database management systems, and different user interfaces. As of now, none are accessible from programs (i.e., they assume a human user interacting with the database interface). They also reflect very different conceptual views of genomic maps. Consequently, at present, most external information must be downloaded from external databases, reformatted, and reloaded into appropriate tables in ***GenomeBase***.

At present, ***GenomeBase*** (and thus ***GenomeGuide***) can only access local databases running on a Sybase database server at LBL. In the future, we plan to automate and streamline direct network access to external database systems by describing their content, structure, and network access procedures as part of a *metadatabase*. An EER description of each external database will allow us to map object definitions from ***GenomeGuide*** to the external database and to generate retrieval and update procedures to interpret the ***GenomeGuide*** operations. The metadatabase will also help integrate and reconcile naming and structural differences and inconsistencies that must be detected and resolved.

Acknowledgements. We gratefully acknowledge the sponsorship of Charles Cantor while Director of LBL's Human Genome Center. We also want to thank molecular biologists who have shared their data and suggestions with us, including Jan-Fang Cheng, Hong Fang, Jeff Gingrich, Rafael Oliva, Jesus Sainz, and Cassandra Smith. Special thanks are also due to Bill Johnson for his contribution to the development of the first version of ***CIS***, Ernest Szeto for implementing ***ERDRAW***, Weiping Fang for implementing ***SDT***, and Arie Shoshani for his contribution to the development of the database tools.

References

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

[2] W. Johnson, S. Lewis, V.M. Markowitz, J. McCarthy, F. Olken, and M. Zorn, "The Chromosome Information System: An Overview", TR LBL-29675, Lawrence Berkeley Laboratory, August 1990.

[3] E.S. Lander, R. Langridge, and D.M. Saccocio, "Computing in Molecular Biology: Mapping and Interpreting Biological Information", *IEEE Computer,* **16**, 11 (Nov. 1991), pp. 6-13.

[4] V.M. Markowitz and A. Shoshani, "Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach", to appear in *ACM Trans. on Database Systems.*

[5] V.M. Markowitz and A. Shoshani, "Object Queries Over Extended Entity-Relationship Oriented Databases: Language, Implementation, and Applications", TR LBL-32019, Lawrence Berkeley Laboratory, February 1992.

[6] V.M. Markowitz and W. Fang, "***SDT*** 4.1. Reference Manual", TR LBL-27843, Lawrence Berkeley Laboratory, May 1991.

[7] V.M. Markowitz and E. Szeto, "A Query Translation Tool. Design Document and Reference Manual", TR LBL-31451, Lawrence Berkeley Laboratory, March 1992.

[8] T.J. Teorey, D. Yang, and J.P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", *Computing Surveys* **18**,2 (June 1986), pp. 197-222.

[9] E. Szeto and V.M. Markowitz, "***ERDRAW*** 2.2. Reference Manual", TR LBL-PUB-3084, Lawrence Berkeley Laboratory, May 1991.

[10] Sybase, Inc., "Transact-SQL User's Guide", Release 4.0, Emeryville, California, Oct. 1989.

[11] M. Zorn, "GenomeBase: Schema for a Mapping database", TR LBL-30666, Lawrence Berkeley Laboratory, May 1991.