Design of a Genetics Database for Medical Research

by

William Chuang

S.B. in Biology, Massachusetts Institute of Technology (1991)
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering
and
Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2000

© William Chuang, MM. All rights reserved.

The author hereby grants to MIT permission to reproduce and
distribute publicly paper and electronic copies of this thesis document
in whole or in part.

Author ............................................ May 22, 2000

Department of Electrical Engineering and Computer Science

Certified by .................................. 22 May, 2000

C. Forbes Dewey, Jr.
Professor
Thesis Supervisor

Accepted by .................................................. 22 May, 2000

Arthur C. Smith
Chairman, Department Committee on Graduate Students
Design of a Genetics Database for Medical Research

by

William Chuang

Submitted to the Department of Electrical Engineering and Computer Science
on May 5, 2000, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science and Engineering
and
Masters of Engineering in Computer Science and Engineering

Abstract

Human medical research has traditionally been limited to the analysis of disease symptoms. While this research has resulted in many great medical advances, it has been encumbered by the lack of human genetic sequence data. Advances in sequencing technology and the advent of the Human Genome Project are rapidly changing this situation. By designing a robust schema for patient genetic data in the context of an electronic medical record, we hope to leverage this wealth of information for medical research purposes. This thesis describes the first global database designed to combine clinical (e.g. patient-specific) gene chip information with the complete Human Genome Database so that sophisticated queries can be made across both databases in an object-oriented manner.

Title: Professor
Acknowledgments

The scientific research and work comprising this thesis could not have been carried out without the help and guidance of my advisors and colleagues.

I would like to take this opportunity to thank Professor C. Forbes Dewey for his insight and guidance; Ngon Dao, for patiently working out the approach and database details; Orion Richardson, for his assistance and collaboration on numerous side projects; and Donna Wilker, for her administrative support, kind ear, and great stories.

Great thanks goes to Informix Software, Inc. for providing database software and technical support, and to Object Oriented Concepts, Inc. for making available the ORBacus CORBA development environment.

This project was supported by a National Institutes of Health Training Grant in Genomic Sciences.

I want to thank my family, for pushing me to get a higher degree and for supporting me in this endeavor. Thank you Shirley for listening to my worries, and thank you Mom and Dad for your drive to see me succeed. Finally, I want to thank my girlfriend Navaz, for making this experience a wonderful and happy one.
# Contents

1 Introduction  

2 Background  

2.1 Database Design  

2.1.1 The Hierarchical Data Model  

2.1.2 The Relational Data Model  

2.1.3 The Object-Oriented Data Model  

2.1.4 The Object-Relational Data Model  

2.2 Data Transport and Exchange  

2.2.1 XML for Data Exchange  

2.3 Privacy Considerations  

2.3.1 Patient Privacy and Anonymity  

2.3.2 Social and Ethical Issues  

3 Motivations  

3.1 Background  

3.1.1 Existing DB Designs: The Object Protocol Model (OPM)  

3.2 Initial Design Goals  

3.2.1 Object-Relational Affymetrix GATC Database  

3.2.2 Object-Relational Human Genome Database  

4 Database and Architecture Design  

4.1 Background
4.1.1 GATC Schema Creation ........................................... 24
4.1.2 HGDB Schema Creation ........................................ 25
4.2 Architectural Decisions ........................................... 26
  4.2.1 Database Connectivity ...................................... 26
  4.2.2 CORBA as a Communications Layer ..................... 26
4.3 Interface & Transport Design ................................... 28
  4.3.1 XML ............................................................ 28
  4.3.2 XLE ............................................................ 28
4.4 Design Conclusions ................................................ 29

5 Implementation ......................................................... 31
  5.1 Background ......................................................... 31
  5.2 Implementing the GATC Schema ................................ 32
  5.3 Implementing the HGDB Schema ................................ 32
    5.3.1 Schema Documentation .................................... 32
    5.3.2 Table Ordering ............................................. 33
    5.3.3 Field, Row Type, and Table Name Limitations .......... 34
  5.4 Merging the GATC and HGDB schemas ......................... 34
  5.5 Conclusions ....................................................... 35
    5.5.1 Implementation difficulties and results ................ 35
    5.5.2 Deploying the databases .................................. 36
    5.5.3 Current status .............................................. 36

6 Conclusions .......................................................... 38
  6.1 Software Design .................................................. 38
    6.1.1 Industry standards ....................................... 38
    6.1.2 General Utility ............................................ 38
  6.2 Future Directions ................................................ 39

A Network and System Architecture .................................. 40
  A.1 Network Architecture ........................................... 40
A.1.1 Network Topology .................................................. 40
A.1.2 HP ProCurve 8000M switch .......................................... 41
A.2 Servers and Services .................................................. 42
   A.2.1 IMAP Mail Server ............................................... 42
   A.2.2 Mail Aliases ...................................................... 44
   A.2.3 Apache Web Server ............................................. 44
   A.2.4 NCFTPD FTP Server ............................................. 48
A.3 Printer Administration .............................................. 48
A.4 On-Line Documentation ............................................. 49
   A.4.1 ORBacus .......................................................... 49
   A.4.2 Informix JDBC ................................................... 50
   A.4.3 Java API .......................................................... 50
A.5 Software .............................................................. 50
   A.5.1 Digital UNIX Administration ................................... 51
   A.5.2 Solaris Administration .......................................... 51

B FML network and website administration .......................... 52
   B.1 FML locker and member lists ..................................... 52
   B.2 FML IP address/hostname list .................................... 53
   B.3 Website Administration .......................................... 53
List of Figures

2-1  The relationship between queries in different database schemes . . . . 14

3-1  The Object-Protocol Model (OPM) architecture . . . . . . . . . . 20
3-2  The Object-Relational Database architecture for the Human Genome
     Database . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

4-1  GATC Expression Database schema . . . . . . . . . . . . . . . . . . . . 30

A-1  The Quantitative Spectroscopy and Image Analysis network . . . . 41
List of Tables

A.1 Comparison of AT&T and BSD-style printing systems . . . . . . . . . . . . 49
Chapter 1

Introduction

The Human Genome project has expanded the horizons of both the biological and medical communities, the latter of which is the ultimate consumer of these advances. The rates at which data have been generated within the Human Genome project are truly staggering. These data currently include sequence libraries, personal genetic information, derived data such as computed molecular images, traditional medical images associated with the genetic information and specific diseases, and annotations and overlays of such image data. A key issue is the development of new multimedia information tools that will make such data fully queryable and just as accessible as data that are currently stored in text and numeric form. Development of the right information systems will allow this project to take full advantage of both the Human Genome Project as well as the physiological data and engineering models that will be generated in this next “grand challenge” project.

Medical research into human diseases has been mostly based on the analysis of symptoms, and more recently, the use of genetic sequences. Until several years ago sequencing was a prohibitively expensive endeavor. With current technological advances and the huge push of the Human Genome Project it appears that the relevant sections will be sequenced within the next year. This wealth of data can be used for medical research, but the raw data must be organized into a coherent schema – one which links it to relevant information.

The sheer volume of the human genomic data is staggering. Effectively arrang-
ing it and the associated metadata for quick searches is an important goal. Patient genetic information must be searchable individually and as a group, against the reference human genome. Existing databases are geared toward the collection of genetic information, rather than for proposed medical research.
Chapter 2

Background

2.1 Database Design

A database is a collection of data which is specially organized for rapid search and retrieval. The earliest databases were flat files containing a set of records, each of which consisted of one or more fields. These fields are the basic units of data and each field corresponded to a particular attribute of the data stored in the database.

As computer processing and storage power has increased over the years, so has the need to store larger quantities and widely varying types of data. These driving forces have resulted in an ongoing evolution in the types of databases available, which can be divided into four main groups. To appreciate the difference between the different types of databases, the concept of a data model must be understood. Fundamentally, data is an undifferentiated set of bits. An abstract model is needed to organize how that data is viewed and to optimize access for both reading and writing. Unfortunately, fitting the data to a specific abstract model makes it more difficult to analyze it with other models. This loss of flexibility can be countered by creating much more complex data models. Fortunately, advances in processing power, storage capabilities, and information theory have made more complex data models feasible.
2.1.1 The Hierarchical Data Model

The next data model to emerge was the hierarchical data model. Here the data as well as the record types are linked in a treelike structure. For example, student records might be grouped under a record describing their departments of study. The corresponding data structure will contain student nodes groups under the department nodes, with the relevant fields associated with each type of node. Unfortunately, in this type of design, the child nodes can only be accessed through their parent nodes, which limits how the data can be retrieved. One well-known medical database system at the Massachusetts General Hospital (MGH) called the “MGH Utility Multi-Programming System” (aka MUMPS), used the hierarchical data model[5].

2.1.2 The Relational Data Model

The relational data model was first proposed in 1970, but it is still the prevailing model in use today[8]. In the relational model, the description of a particular entity is provided by the set of its attribute values, which are stored as a single row (a tuple) of a table (a relation). Each column in a table has one of a limited set of primitive data types. The relational approach supports queries involving multiple tables by means of a “join” operation that combines records with identical values of common attributes.

While the relational data model is mature and reliable, it encounters its limits when the data are complex[42]. Traditional relational databases in use today do not have the native capacity to deal with compound multimedia objects or to flexibly handle genetic information. Instead of true support, they use alphanumeric pointers to refer to other database structures that hold the objects. The few existing relational genetic databases do not incorporate medical image data and provide limited extensibility. This jeopardizes the data by requiring independent databases that must be kept in harmony, and also increases the complexity of all the applications that access and query the data, or modify it. Furthermore, they do not support the development of information objects with inheritance, a key strategy in using advanced
object-oriented languages such as C++ and Java to develop the next generation of applications. Figure 2-1 illustrates the differences in the relational and object-relational database approaches.

### 2.1.3 The Object-Oriented Data Model

Another development has been the incorporation of the object concept that has become significant in programming languages. In object-oriented databases (OODBs), all data are objects. Objects may be linked together by an "is-part-of" relationship to represent larger, composite objects. Classes of objects may form a hierarchy in which individual objects may inherit properties from objects farther up in the hierarchy.

The object-oriented data model has many advantages over the relational data model, but most commercial OODBs remain unstandardized, resulting in several different proprietary data models and interfaces. In addition, they also lag behind relational databases in transaction management, replication, and development environments. This has negatively affected their acceptance into the business and scientific markets.

### 2.1.4 The Object-Relational Data Model

The object-relational data model (ORDBM) was developed by both Stonebraker[37] and Kim[27], independently in the 1990’s. The ORDBM builds upon the strengths of both the relational data model and object-oriented data model.

The ORDBM is able to use relational data model technologies, since it is based upon relations between objects. Thus from its relational roots the ORDBM gains transaction management, concurrency control, query optimization, and other relational features. In addition, it is able to leverage the Standard Query Language (SQL), as well as the Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) interfaces and programming APIs.

The object-oriented facet allows ORDBM columns to contain many different types of data, entire rows, and even user-defined types and structures. The ODBC and
JDBC interfaces have also been extended to deal with the results that object-relational SQL commands can return. Unfortunately, while object-oriented databases focus on the ability to manipulate persistent objects, they have traditionally done so through SQL interfaces, which results in complex objects being viewed as simple tables; the resulting massaging of objects to and from tables requires extensive programmer modifications. The new SQL-3 standard[29] should allow better mappings between row types to user-defined classes.

Systems containing multimedia biological data require the object-relational paradigm for efficient operation. Figure 2-1 illustrates the differences between relational, object, and object-relational systems in their support of queries. Because the ORDBMS solution has the large object storage and inheritance capability of the object world combined with the key advantages of the relational world, it is an ideal solution for the genomic and physiologic data.

Figure 2-1: The relationship between queries in different database schemes. After M. Stonebraker, Object-Relational Database Management Systems, Morgan Kaufmann, San Francisco, 1996
2.2 Data Transport and Exchange

A major concern in modern database systems is the question of how to convey the results of a data query. While in many system specifications the input and output adhere to strict formats, there are instances where a given data type is incapable of capturing the semantics of the information being exchanged. As an example, as interfaces and schemas become more complicated, there will be a need to query the data models for information about themselves; this information is known as metadata, and allows a data model’s user to learn the context and assumptions upon which the data model is built.

Many database operations, especially those in medical research, return complex data which cannot be interpreted unambiguously without its associated contextual information.

2.2.1 XML for Data Exchange

The basic means of exchanging data is to use structured text. In order to achieve this goal there must be a standard framework for describing the structure of the text. The Extensible Markup Language (XML) standard has solidly emerged as the technology to solve this problem[47].

XML allows data to be structured in a hierarchical format. The “extensibility” of XML allows user-defined markup tags and data, to fit a specific domain. The associated XML Document Type Definition (DTD) is used to define the structure for a given domain; all documents in the domain can be verified against its DTD. XML is easier to use than its forefather, the Structured General Markup Language (SGML).

It has been shown that object-oriented structures can be mapped directly to the structure of XML[23], which makes it eminently suited for transporting and exchanging data from object-oriented and object-relational databases. XML’s flexibility and portability have gained it industry-wide acceptance.
2.3 Privacy Considerations

2.3.1 Patient Privacy and Anonymity

As used in hospitals and other care facilities, electronic medical records (EMRs) contain a multitude of patient information; images, doctors’ reviews/comments, CAT scans, blood type, address, sex, age, SSN, etc. While this information is relevant and needed in those locations, it can also be used to breach patient privacy if it falls into the wrong hands in a more open research environment.

This information could be used to discover the patients’ identity – for insurance, government, harassment, or other purposes. To assure patients of their privacy, their data must be “scrubbed” to remove any identifying characteristics. This is not as easy as it first appears, since the combination of multiple non-identifying data can often be just as specific as patients’ names. The difficulty lies in removing enough information to protect patient privacy without also reducing its medical research use.

LaTanya Sweeney (Assistant Professor of Computer Science and of Public Policy at CMU) has authored several papers dealing with this subject [40, 38, 39].

While the topic of anonymity is not directly relevant to this thesis, it must be definitively addressed before this system is brought into production.

2.3.2 Social and Ethical Issues

A veritable Pandora’s box of social and ethical issues will arise from the ability to perform genetic sequence-level research. Some of these issues include:

- selecting of traits for future offspring
- intelligence boosting (as in the case of the genetically “enhanced” mice, which seemed to be smarter after treatment)
- educating people about genetics and the probability of hereditary diseases
- viewing people as the composition of their genes

These issues must all be dealt with as human genetic research progresses. It is possible that these topics may eventually fall under government supervision, whether
directly or through new laws.
Chapter 3

Motivations

3.1 Background

There currently exist many genetics databases, most of which serve as catalogues or repositories for ongoing sequencing projects. These databases include GenBank, EMBL, Swiss-PROT, TRRD, DDBJ, to name a few. In addition there exist "meta-databases" which collate and attempt to annotate data from multiple sources. One major ongoing work is the Human Genome Project, whose goal is to map and sequence the entire human genome.

Limitations in relational data models reduce the ability of researchers to correlate genetic profile information to medical image and patient data. Since traditional relational databases do not fully support the concept of objects, the multimedia and genetic information must be arbitrarily linked to each other and their derived data in order to preserve data relationships. Such gratuitous object-to-object connections are merely "band-aids" that cover up faults in the underlying storage model and information schema. These patches become more prevalent as researchers access data spread across multiple databases. These cross-schema relationships are difficult to manage because their domains are often disparate and as the number of objects and databases grows, the number of relationships grows exponentially.
3.1.1 Existing DB Designs: The Object Protocol Model (OPM)

In 1996 a research group at the Lawrence Berkeley National Laboratory realized that treating data as objects would be the best method for handling large quantities of information for which there were fixed associations (for example, the classification of fruits).

At the time relational databases were the norm, and object-oriented databases had only a small following. Object-relational databases were just beginning to be developed and didn’t seem to be scalable enough for large-scale research uses. In this situation the research group created the Object-Protocol Model (OPM), a protocol for communicating object requests and results to and from normal relational databases.

The Object Broker

They created an Object Broker (OB) which is an application server that functions as the middle layer between the actual physical database server and the front-end tools. It is an example of the “three-tier” design for enterprise database software. Specifically, it performs the following tasks:

- user authentication
- user and group access security
- translation of object queries to SQL
- packaging of row results into an object structure
- multi-server/database connections
- logging of user transactions

The OB server communicates with the underlying database via SQL; it contains two translators: an OPM Query Language Translator that converts object queries into SQL batches, and a special translator for instance insert and updates that converts object updates into SQL batches. Object Broker clients are end-user programs that can perform queries and updates to a database via the Object Broker server (OB).
These actions are made through an OB client API library that handles the low-level details of communication with the OB server. Client libraries are currently provided in C, C++ and PERL.

Common Transport Language

OB client data transport is based upon the Common Transport Language (CTL), a generic data encoding language. CTL is integrated into the OB clients and servers.

![Diagram of the Object-Protocol Model (OPM) architecture](image)

**Figure 3-1:** The Object-Protocol Model (OPM) architecture

Their architecture essentially places object abstraction outside of the database. The database schema is automatically created from the higher-level “object schema” and so the underlying relational tables do not necessarily correspond to the actual objects. Modifications to the “object schema” require overhauls to both the object descriptions and the underlying relational tables. However, this was an excellent database architecture at the time and was chosen for use as the core of the Human Genome Project and for several other biological databases.
3.2 Initial Design Goals

Using the information gathered through extensive reading and talks with medical researchers and my advisor, I developed a list of initial goals, of which the main thrust was to design a database for medical research purposes, which would allow rapid lookups of patient genetic data within the context of an electronic medical record.

Over the course of several meetings the vision for my thesis solidified into the following goals:

- designing a database schema which extends the Affymetrix GATC DB to include metadata and bring in generic genomic data
  - metadata
    * physical substrate and ingredients (batch information)
    * experiment author/use information
    * experiment qualitative information
  - genomic data
    * including/caching/fetching Human Genome Database (HGDB) data to allow data-mining across the two domains
- creating a Informix DataBlade module which can perform fast searches on DNA/amino acid sequences
  - exact matches for a large search space (hundreds of kbps per gene)
  - non-exact (fuzzy) matches for base-pair and amino acid sequences
- exporting the search results with references to related data, in a flexible format
  - using XML to preserve information linkage
  - using Java as programming language
  - using CORBA as transport mechanism
  - using XLE to combine the first two abilities - requires schema in order to derive the DTD

This resulted in research into building Informix DataBlade modules, designing fast "fuzzy" search algorithms, using CORBA as a lightweight transport mechanism and XML as a data encapsulation mechanism. However, the database schema at the core of the thesis remained amorphous. Over the course of a few months, this grand vision
narrowed to focus on the schema. Professor Dewey and Ngon Dao suggested that I concentrate on probe and micro-array databases (such as those from Affymetrix) and link in Human Genome DataBase information:

- develop a object-relational Affymetrix / micro-array database
- develop a object-relational genomic database
- show that we can bring together HGDB and micro-array information
- Use global transport mechanisms and advanced data representations
  - CORBA for the transport mechanism
  - XML (IBM’s XLE) for representation of object-oriented data linkage
  - Java for global access GUIs

3.2.1 Object-Relational Affymetrix GATC Database

The Affymetrix GATC database schema is solely oriented toward the storage of experiment data from individual Affymetrix GeneChips. In this regard, it contains tables whose purpose is to correlate specific locations on the GeneChip with the parameters, measurements, and analysis of the experiment results. The main reference point for researchers is a “string” containing a gene name or an accession identifier, which is used to identify which gene a particular spot on the GeneChip was testing for.

Since GeneChips are commonly used in medical research to quickly identify which of thousands of genes are activated, the Affymetrix GATC databases are excellent ways to store individual patient genetic data. In order to use this data in an object-relational manner, a compatible database schema must be created and the data imported into the new database.

3.2.2 Object-Relational Human Genome Database

The Human Genome Database (HGDB) is based on its Object Protocol Model. This data model removes the object paradigm from within the database and provides it in a separate abstraction layer “above” the database. With this design the underlying relational database schema usually no longer matches its object model. In the event
of additions to the object model – an oft-occurring event as research progresses – the mappings between the object model and its underlying relational schema can become extremely complicated.

The goal here is to replicate the entire object model of the HGDB in an object-relational database. This approach provides several advantages, namely it would:

- remove the intervening abstraction layer and associated complexities
- provide an easily extensible and inheritable object model
- allow much faster cross-database searches

The resulting architecture can be described below, in contrast to the current HGDB architecture pictured in Figure 3-1.

Figure 3-2: The Object-Relational Database architecture for the Human Genome Database
Chapter 4

Database and Architecture Design

The fundamental part of this project is the design and implementation of the genetics databases. Without a logical design and schema, it would be impossible to phrase coherent queries and leverage the information contained in either database, much less perform cross-database queries.

4.1 Background

4.1.1 GATC Schema Creation

The first core of the project was the GATC database. Affymetrix teamed up with Molecular Dynamics to form the Genetic Analysis Technology Consortium (GATC)\[10\], in an attempt to provide a platform to design, process, read, and analyze DNA-chip arrays. The database architecture that the GATC Consortium created is a basic relational one, geared toward the storage of experimental data.

A major design goal was to have the object-relational implementation be compatible with the original GATC design. This is necessary to allow researchers the ability to easily import their experimental data directly into the new ORDBMs without requiring additional massaging. Given this, it was decided to treat nearly all the existing GATC relational tables show in Figure 4-1 as objects.
4.1.2 HGDB Schema Creation

An initial question was whether or not to store a reference copy (or subset thereof) of the human genome data within the database, as opposed to storing the entirety of the data, or requesting the data as needed. There are advantages and disadvantages to both approaches:

- **Storing a reference copy (or using replication servers):**
  
  **Advantages**
  1. Collation and integration of data from multiple sources must be done prior to any usage, and can be very time-consuming.
  2. Extensive storage space is required for the 3+ billion base-pairs, 100,000 genes, and all the annotation information.

  **Disadvantages**
  1. Can define our own request and data return formats.
  2. Searches would be sped up immensely.

- **Using off-site databases:**
  
  **Advantages**
  1. Local storage space is required only for the search results, which can be cached.

  **Disadvantages**
  1. Collation and integration of data from multiple sources must be done on the fly for each request. This may be slow and may require user intervention.
  2. Knowledge of the search request and return formats must be hard-coded into our genetics module and must also be kept up-to-date.
  3. Search speed is slow due to web, ftp, and/or e-mail based request latencies.

If the human genomic data were to be stored locally, its database should have two primary functions – the storage of (some) genetic data and metadata, and the ability to conduct fast searches on these contents. Due to the nature of the database, the addition of new data, whether for the reference genome or for individual patients, was expected to be infrequent compared to the occurrence of exploratory searches.

As the design phase progressed, it became obvious that this matter would need to be dealt with after the schema and database were created. Rather than create only
a small segment of the overall HGDB schema, it seemed best to incorporate it in its entirety into the design.

The Human Genome Database initially appeared to be very promising in regards to database design. The HGDB team had designed the database using the Object Protocol Model (OPM)[7], and so the top-level descriptions were entirely in terms of objects and their relationships to one another. Their full schema is too large to be included here but can be accessed on-line at http://www.gdb.org/gdb/schema.html.

4.2 Architectural Decisions

4.2.1 Database Connectivity

JDBC is an acronym for Java Database Connectivity, and ODBC similarly is an acronym for Open Database Connectivity. They provide methods for connecting to databases in Java and C (or C++) programs, respectively.

While the selection of database drivers ultimately boils down to a choice between programming languages, for the present most database vendors provide JDBC, JDBC ↔ ODBC, and ODBC drivers. If the client/server architecture were entirely in C or C++ it would be best to use ODBC drivers; if they were written in Java either JDBC or the JDBC ↔ ODBC bridge drivers would be appropriate.

JDBC allows connections to the database to be negotiated entirely in Java, without going through a JDBC↔ODBC conversion mechanism. This is both faster and requires fewer inter-operating parts, the latter of which strongly corresponds to greater ease-of-use and simpler maintenance requirements. Also, Informix strongly suggests that Java clients use the native JDBC drivers.

4.2.2 CORBA as a Communications Layer

Most database implementations act as servers, and the database companies provide client applications to access (query and write) the database. In this model, a custom client application would need to directly access the database server. This usually
involves transfer of a username and password across the network, which could be a strongly negative security breach, but with the use of 128-bit SSL connections is no longer an issue.

CORBA (Common Object Request Broker Architecture) provides an easy mechanism to avoid this failure. By using IDL (interface definition language), it is possible to define a set of functions which allow access to the server. Then using "jidl" or "idl2cpp" will create client and server-side stubs for later fleshing out.

Once this process is completed, the CORBA client talks to the CORBA server, which in turn talks to the database. In this setup only the CORBA server needs to know the database username and password; in addition, it can reside on the same physical host as the database to minimize network traffic and exposure.

In addition to security, CORBA also provides flexibility. CORBA client applications do not need to know the hostname and port number for the CORBA server at compile time. This eliminates the need for recompilation and redistribution of the CORBA clients if the CORBA server should ever move or be renamed.

CORBA is available for C, C++, and Java. The Java implementation was chosen since it is mature and allows a complete Java solution from CORBA client ↔ CORBA server ↔ database. Several free implementations are available, including ORBacus’ which is currently being used.

**CORBA alternatives – RPC, Java RMI, EJB**

RPC is a Sun Microsystems mechanism for performing “remote procedure callbacks”, essentially allowing execution of code on remote servers. Unfortunately, it isn’t standardized for all platforms and free implementations may not exist for all platforms. It requires C or C++ which is a limitation due to platform-specific compilation issues.

Java RMI stands for “Java Remote Method Invocation” and is essentially a Java implementation of RPC. While RMI has its benefits, it remains tied to whatever specific server it is coded for.

EJB stands for “Enterprise JavaBeans”, which are small Java modules that can be hooked together to create a larger application. This scheme allows for modularity,
code reusability, and quick bugfixes. However, its main goal is to provide a powerful server side engine instead of the client-server architecture needed for this project.

4.3 Interface & Transport Design

The interface layer between the genetics database and other models should be as portable as possible. We will be using CORBA for this communication medium, to allow interoperation with other modules in the Physiome project such as those of Orion Richardson and Ngon Dao.

4.3.1 XML

There are two opposing approaches to defining the interface. One choice is to create a richly detailed interface which specifies all possible interactions with our database. The other option is to limit the interface to a standard set of methods common across all the models; the data transferred would thus be more complex in structure. Since we wish our database to interface with other models, using a minimal interface with structured data is a more appealing. XML (eXtensible Markup Language) is admirably well-suited to the task of transferring data with the object-oriented structure required. In addition most databases can export directly to XML, and the results can be converted to objects in other languages.

A major factor in choosing this “strong XML with a minimal interface” approach is the standardization of related models in the Human Physiome Project, specifically Orion Richardson’s project[35], along these lines.

4.3.2 XLE

IBM’s Alphaworks has provided alpha versions of their XML Lightweight Extractor (XLE)[30] for free usage. Given an XML DTD, XLE allows users to annotate the DTD to associate various components with underlying data sources (databases). When requested, XLE will extract data from the data sources and assembles the data into
XML documents conforming to the DTD.

The mapping mechanisms in the annotated DTD allow XLE to assemble an XML document from relational tables related by foreign-key relationships or more advanced relationships.

This technology is especially well-suited to our design, as it melds the power of object-relational databases with the flexibility of XML as a data exchange medium.

4.4 Design Conclusions

The major design decision was the choice between imposing a rich CORBA interface with minimal XML markup, or the opposing choice of a minimal CORBA interface allowing complex XML markup. This resulted in extensive discussions with Prof Forbes Dewey, Ngon Dao, Orion Richardson, and other members of the research group.

Since the genetic database would need to interoperate with other modules in the Human Physiome Project, the best interface seemed to be a minimal one which would be implementable for all the modules. The extensibility of this design is thus placed squarely on the structured data used as the exchange medium, which produces a more flexible system. In addition, minimal reliance on CORBA allows its gradual phasing out in favor of newer technologies as they appear.
Figure 4.1: GATC Expression Database schema
Chapter 5

Implementation

The schema design and architecture decisions were only the first part of the total project. While in theory once the design was set, the implementation of the databases should have been straightforward, this was not the case. This chapter deals mainly with my experiences developing and implementing the databases and the transport mechanisms.

5.1 Background

The following software was used for this project:

- Informix JDBC Driver version 2.10C
- Sun Java Development Kit version 2.0
- Sun Solaris version 2.5.1
- Object-Oriented Concepts’ ORBacus for Java
- IBM XLE

The main database development was done on an Informix Universal Server, version 9.14, based on a Sun Ultrasparc workstation running Solaris 2.5.1. The database server is fully object-relational, and since it had been used for previous related work I decided to develop on the same platform.
Java was used as the development language because it has a short development cycle and a set of full-fledged graphical user interface widgets built in. Informix also graciously provided us with Java Database Connection (JDBC) drivers which were needed to use the object-relational features of the Informix Universal Server. The JDBC drivers rely on Java version 1.2 or higher.

### 5.2 Implementing the GATC Schema

Converting the GATC relational schema to an object-relational schema was relatively painless. The tables in the original GATC database can stand alone as single entities, and there is no inheritance required in the design. This relational schema can be directly mapped to an object-relational schema, and then individual objects can be extended to include additional data or metadata.

I talked with several graduate and post-doctoral students to see what information they felt was currently lacking in the Affymetrix GATC schema, and the main suggestions were to add:

- image metrics – to assess the quality of the image data
- metadata – to incorporate information about the experiment

### 5.3 Implementing the HGDB Schema

#### 5.3.1 Schema Documentation

The structure for the Human Genome Database was provided by the Human Genome Project in several forms:

- PostScript schema document
- relational specifications
- OPM schema definition
The PostScript schema documentation provided a very high-level overview of the object and their relationships to one another. The relational specifications consisted of a database dump of the tables, except that the tables only contained primitive types and ignored cross-references. Meanwhile, the OPM schema definition contained the OPM commands used in creating the database, and this was only somewhat useful. Luckily, the CTL schema file contained much more data, including the references to other objects. Given these sources, it was possible to piece together the actual schema by:

1. creating the object “skeleton” from the object descriptions in the PostScript schema documentation
2. fleshing out the objects with the database primitive types listed in the relational specifications
3. adding in the cross-references listed in the CTL schema file and verifying the results against the PostScript schema and relational specifications

5.3.2 Table Ordering

A key feature of relational databases is their ability to enforce constraints on the cross-references (foreign keys) that link tables. Relational database are very unyielding in that they disallow “forward” references to tables; hence a table must exist before it can be referenced. In object-relational databases, the same situation holds true, and is extended to row types as well. While this feature provides strong referential integrity, it becomes a hindrance to implementing objects, as these may have some type of circular references.

Ordering the myriad row types and tables in the HGDB required an extensive investment of time and energy. In addition, unfortunately, a moderate number of the cross-references in the HGDB were circular, and situations would occur where:

- objects would reference themselves
- pairs of objects would reference each other
- children of objects would reference their parents
Link Tables

Since references must be to already predefined objects, this becomes a classic “chicken and egg” problem. The only way out of these sets of circular references was to create what are known as “link tables”, in which the keys and values refer to rows in the original tables; the cross-reference constraints are then placed on the table after the original tables are created.

5.3.3 Field, Row Type, and Table Name Limitations

The version of the Informix Universal Server used also had a limitation of 18 characters for field names, row type names, and table names. This conflicted with the HGDB object schema which used long, descriptive names. To handle this discrepancy I was forced to iteratively map all field, row, and table names and their references, to abbreviated spellings.

5.4 Merging the GATC and HGDB schemas

Searches of the GATC database schema are oriented around a text string. Most researchers use a gene name or an accession identifier (AccessionID) to associate particular locations on the GeneChips with a specific gene.

Most HGDB schema objects are descended from AccessionIDs, which are unique identifiers for the hundreds of thousands of database objects in the Human Genome Database.

Thus the AccessionID provides us with a unique and logical center around which queries can be performed across both the Affymetrix GeneChip domain and the Human Genome Database domain. A medical researcher will be able to:

1. load into the database a set of GeneChips corresponding to individual patients
2. perform a database search over the set of GeneChip data for locations which are activated in all the patients
3. retrieve the AccessionID corresponding to each of those locations

4. search in the Human Genome Database for information about that particular AccessionID, including but not limited to:
   - the gene in question
   - what alleles the gene is associated with
   - what gene product(s) it is associated with
   - references for that gene and gene product in the medical literature
   - other researchers involved with related experiments

This combined database will provide the coalescence of the information required to perform complex queries in a short amount of time.

5.5 Conclusions

5.5.1 Implementation difficulties and results

Implementation of the HGDB object model was impaired by the several different forms it was provided in, all of which were incomplete individually. Piecing together these sections into a coherent whole consumed more time than I had allocated for this section of the project.

The Informix database server had two limitations which required additional work and slowed my progress:
   - an 18 character limit in field, row, and table names
   - an 32767 byte limit in row sizes

The first case resulted in a renaming of all the fields, row types, and table names and references to them in the SQL files. This “mapping” file is included in the appendices. The second limitation was due to a hard-coded maximum in the size of a row; since the Informix Universal Server set aside a large amount of space for each SET and LIST object, the size of several row types rapidly exceeded the 32767 byte limit. This forced the commenting out of several columns in two tables; once the IUS is upgraded to a newer version the problem will be fixed and the two tables will be fully restored.
5.5.2 **Deploying the databases**

The GATC and HGDB databases were created on the Informix Universal Server in the lab. Currently there are several mature object-relational database products on the market, providing essentially similar functionality. The development machine was a Sun Ultra-1 workstation running the Solaris 2.5.1 operating system, which represents a commonly workstation computing solution.

The current implementation of the database was loaded onto the Informix Universal Server (IUS) in a single file named “HGDB.sql”. The sequence of steps is as follows:

1. Start the Informix Universal Server from /etc/rc3.d/S99oninit (on boot)
2. Run the database client program “dbaccess” and connect to the IUS
3. Create an empty “hgdb” database
4. Load the “HGDB.sql” into the “dbaccess” query editor
5. Run the commands in “HGDB.sql”

The database client then executes the SQL commands in “HGDB.sql” sequentially, creating all of the row types and tables it specifies.

5.5.3 **Current status**

The GATC and HGDB databases are currently active and running in an Informix Universal Server (version 9.14) on the Sun Sparc-1 server “miro.mit.edu”. The source code for the two schemas exists on that server in the files /home/wchuang/src/Database/hgdb/HGDB (113.5 kbytes, 4840 lines of SQL) and /home/wchuang/src/Database/gatc/GATC.sql (9 kbytes, 368 lines of SQL).

Unfortunately, the GATC database is unpopulated, since I was unable to find any medical or biological research groups who were willing to volunteer their data for testing purposes. Privacy and proprietary information appear to be why the data is unavailable.
A newer version of the Informix Universal Server (version 9.20) reportedly fixes the 32767 byte row-size limitation and the 18 character field name length limitation mentioned earlier. Once the IUS is upgraded, the relevant lines in the SQL files can be uncommented and loaded into the database server.

The HGDB now contains the dictionaries, but not the genomic data, contacts, nor citations. Currently at least 40 MB of RAM are required for the initial creation of the HGDB; the Informix Universal Server doesn’t seem to use virtual memory at this stage, though eventually databases are “swapped out” when not in use.
Chapter 6

Conclusions

6.1 Software Design

6.1.1 Industry standards

This project would not have been able to achieve its goals without the use of many industry standards. Industry standards, both for protocol, data exchange, and semantics are crucial for forward progress. XML, Java, JDBC, and ORDBMs are excellent examples of how industry standards have moved the technology forward and benefitted everyone.

However, standards must be nurtured. Without cooperation with the various standards-making bodies and compliant implementations, the software industry will become fragmented, offering competing and uninteroperable products. Situations such as these only hurt the end-user, and severely hamper the ability to provide useful software.

6.1.2 General Utility

A common thread in modern software engineering is to allow extensibility while maintaining functionality; many projects deliver the functionality that users desire, but lack the ability to be extended.
6.2 Future Directions

In the near future this project should be moved to an upgraded database server such as Informix Universal Server v9.20, which can handle extended field names and larger row sizes. The physical workstation will also need more RAM and extensive disk space to store data from many individual Affymetrix GeneChips and either the entirety or a large part of the data from the Human Genome Project.

Early work extending this project should include automated caching of sections of the HGDB and/or periodic updates of the database with new changes from the HGDB; this is a slight variant of database replication, but the one-to-one object mapping from the HGDB to this database should simplify the work.

Another side project should be the addition of μ-array slide database schemas; this work is on the same scope as the creation of the GATC database, and will extend the reach of the project. Another important path to explore is the integration of protein information to expand the breadth and scope of the database.

It is hoped that this project will result in data-mining across the genetic patient data with human genomic data, to perhaps identify latent traits and/or grant new insights into the functionality of various genes.
Appendix A

Network and System Architecture

A.1 Network Architecture

The Quantitative Spectroscopy and Image Analysis (QSIA) network provides the infrastructure for our NSF-sponsored imaging center. This high-speed network is based on standard fiber optic technology and supports Gigabit (1 Gb/sec), Fast Ethernet (100 Mb/sec), and standard Ethernet (10 Mb/sec) connections.

A.1.1 Network Topology

The current network is based on a "star" topology. In this configuration there is a central switch (or hub) and all other machines are connected to it.

An HP ProCurve 8000M Switch located in m3-253 is the "heart" of the network. Connected to the switch are optical fiber connections to m3-154, m3-315, m16-324, and m56-353. The switch itself is connected to the MIT campus network, which provides our connectivity to the outside world.

The local bandwidth between any two machines directly connected to the switch is only limited by the speed of their individual connections to the switch. All connections to the "outside" world are limited by the bandwidth of the campus network, which is currently 10 Mb/sec.

A fiber network has a maximum run-length of 412 meters (1250 feet) in standard
half-duplex mode. The diameter of our current network is well within that limit.

Figure A-1: The Quantitative Spectroscopy and Image Analysis network

A.1.2 HP ProCurve 8000M switch

The HP ProCurve 8000M Switch contains a backplane that can handle 10 module cards. Module cards for this particular switch can be one of:

- **HP 8-Port 10/100Base-T Module (J4111A):** This module has 8 RJ45 ethernet ports; each port is auto-sensing for either 10 or 100MB/s connections. The RJ45 connector-type is the most common type of network connection in use today and is copper-based. Because the HP ProCurve switch is physically in 3-253, these ports are used for workstations within that lab. The switch contains two of these modules, for a total of 16 possible connections.

- **HP 4-Port 100Base-FX Module (J4112A):** This module has 4 SC-style 100MB/s fiber ports. Each can be directly connected to a network card which is physically near the switch. However, practical use requires a connection to a fiber "drop box" serving as one end of a long fiber run. A fiber-based network card in a workstation is then connected to a pair of fiber strands in the remote drop box (a pair of strands is receive/transmit [RX/TX] pair). The switch currently contains 3 of these modules for a total of 12 possible connections.
- **HP 1-Port Gigabit-SX Module (J4113A):** This module has a single Gigabit fiber port. It can be connected up to the same fiber connections described above. The switch contains 3 of these modules. One of the connections runs to the ICMIT server "icmit.mit.edu" in the lab, and the other two are scheduled to be connected to machines in Peter So’s lab (m3-315) and the remote electron microscopy lab (m56-353).

### A.2 Servers and Services

#### A.2.1 IMAP Mail Server

Lipchitz.mit.edu (aliased to “icmit.mit.edu”) is running the CMU version of the IMAP mail server (known as “Cyrus IMAP”) to provide fast, secure mail service for local and remote use.

**Mail Server Design**

The original UNIX mail server consists of a sendmail daemon which listens for both incoming and outgoing mail requests, and hands the former off to a local “mail” handler. This mail program adds the mail message to a file in /var/spool/mail/$user, where $user is the username of the local recipient. The standard IMAP mail server (from the University of Washington) uses the same format, mail file structure, and server directory structure.

The Cyrus IMAP server takes a different approach to the mail file and server directory structure:

- Instead of placing each user’s mail messages into a single monolithic file, it creates a mail directory for each user and places each mail message into separate files.
- It creates a database tracking system for each user, and notes the number and headers of new messages, old messages, and unread messages.
- It allows each user to create mail “sub-folders” within the user’s “root” folder.
By sorting the mail messages into separate files, the Cyrus IMAP server avoids having to read a user's entire mail file into memory for processing, and can instead work on smaller "chunks" at a time. The database tracking system is log-based, which ensures that modifications are atomic and resistant to corruption. The addition of "sub-folders" is a nice touch for users who previously had problems sorting large quantities of e-mail. Finally, the header tracking system allows users to leave their e-mail on the Cyrus IMAP server, and to view the headers without needing to download the entire bodies for all the e-mail messages.

Cyrus IMAP Configuration

The main configuration file resides in /etc/imapd.conf and consists of three lines:

```
configdirectory: /usr/local/imap
partition-default: /ext/spool/imap
admins: wchuang pat orionr
```

The "configdirectory" specifies the location of the IMAP server configuration files, which includes the list of users, quotas, global mailboxes, permissions on mailboxes, and more. The "partition-default" dictates where the IMAP server should setup the actual mailboxes, and the "admins" lists the users who are allowed to administer the IMAP server.

Cyrus IMAP Administration

Cyrus IMAP server administrators must first login to the server machine (lipchitz.mit.edu) and then run the program "/usr/local/bin/cyradm". The program will ask for the administrator's username and password (thus the /etc/passwd file must be readable by the Cyrus user or group) and once that is verified, will provide a "lipchitz.mit.edu>" prompt\(^1\). The "help" command will list the following commands:

\(^1\)If the username or password are incorrect, the user will be returned to a shell prompt – but not the original one. Type "exit" to get back to the original shell
lipchitz.mit.edu> help
createmailbox, cm create a mailbox
deleteaclmailbox, dam delete an ACL on a mailbox
deletemailbox, dm delete a mailbox
help get help on commands
listaclmailbox, lam list the ACL on a mailbox
listmailbox, lm list mailboxes
listquota, lq list quota on root
listquotaroot, lqr, lqm list quota roots on mailbox
quit exit program
renamemailbox, renm rename a mailbox
setaclmailbox, sam set an ACL on a mailbox
setquota, sq set quota limits

The commands are mostly self-explanatory. To create a mailbox for a new user, an administrator would use “cm user.$username”, where $username is the new user’s username. It is important to note that Cyrus IMAP users are allowed to create sub-folders within their own mail “root” folder.

A.2.2 Mail Aliases

The mail aliases exist in /var/adm/sendmail/aliases on lipchitz.mit.edu. Read the actual file to see how the normal and majordomo aliases are set up. There also is an extensive manual page on aliases available via “man 4 aliases”.

The /var/adm/sendmail/aliases file can only be modified by the system administrator. Once it has been changed, the system administrator must run “newaliases” to build the aliases database (aliases.db).

A.2.3 Apache Web Server

Lipchitz.mit.edu is also running the Apache web server, which allows the ICMIT to provide a website devoted solely to medical imaging and computing issues. The server has several standard module additions, including SSL (Secure Socket Layer), mod.php (Personal Home Pages), and mod_perl (to allow PERL scripting).
Files and Directories

The binaries, log files, and configuration live in /usr/local/apache. Of those, the configuration files are the most important and they are in the etc subdirectory.

The important files in the etc subdirectory are:
- magic: contains mappings of “magic numbers” to file types
- mime.types: contains mappings of mime types to file extensions
- access.conf: Apache configuration file - access permissions
- httpd.conf: Apache configuration file - server setup
- srm.conf: Apache configuration file - directory mappings
- jserv.conf: Apache JServ (java servlet) configuration file
- jserv.properties: (used by above)

The important files in the bin subdirectory are:
- dbmmanage: to manage user authentication DBM files
- htdigest: to create/update user authentication files
- htpasswd: to add users and update passwords in user auth files

The important files in the sbin subdirectory are:
- ab: to stress-test the Apache web server
- apachectl: to manage the Apache web server
- apxs: to load/unload Apache extension modules
- httpd: the Apache web server
- logresolve: to resolve hostnames for IP addresses in logfiles
- rotatelogs: to rotate logfiles after a certain time or size
- ssl_gcache: to cache SSL certificates

The libexec subdirectory contains Apache extension modules, which can be dynamically loaded and unloaded from the server for extra functionality.

The var subdirectory contains logfiles and process id files.

The include subdirectory contains header files for programmers wishing to write modules using the Apache extension module interface.

The share subdirectory contains servlets, CGI programs, and icons. Its “htdocs” subdirectory is not actually used by our Apache web server (see the srm.conf in /usr/local/apache/etc file to see what the DocumentRoot is actually set to).
Managing the Server

The Apache web server is invoked when lipchitz.mit.edu boots up. The Digital Unix initialization scripts live in /sbin/init.d; each specific runlevels has a directory in /sbin, such as /sbin/rc0.d, /sbin/rc1.d, /sbin/rc2.d, and /sbin/rc3.d. The contents of these directories are symbolic links to the actual scripts in /sbin/init.d, of which “httpd” is one.

These scripts take “start”, “stop”, or “restart” arguments to modify the run status of a particular service:

```
lipchitz# /sbin/init.d/httpd restart
```

The same job can be done by explicitly running the Apache control manager with:

```
lipchitz# /usr/local/apache/sbin/apachectl restart
```

Updating the SSL Certificate

The Apache web server has an site certificate which allows SSL (secure socket layer) connections to it. SSL uses a protocol that exchanges a randomly derived one-time key to encrypt each user’s SSL sessions (each key is never reused).

Our web server has a site certificate which is “signed” by the MIT Certificate Authority and has a lifetime of exactly one year. This assures users that they are actually securely connecting to our web server, and not to a website which is posing as https://lipchitz.mit.edu/. A new site certificate must be requested every year.

The SSL software and certificates are in the /usr/local/ssl directory. To request a new certificate you should follow these steps:

1. Create a reasonably large amount of some random data (usually
   lipchitz# ssleay md5 * > rand.dat or
   lipchitz# ps algwwx > rand.dat will do)

2. Generate a private key – keep this somewhere safe and private
   • Without a passphrase:
     lipchitz# ssleay genrsa -rand rand.dat > key.pem
   • With a passphrase:
     lipchitz# ssleay genrsa -rand rand.dat -des3 1024 > key.pem
3. Generate a certificate signing request (CSR)

- lipchitz# ssleay req -key key.pem -new > req.pem
- When prompted for input, use these answers:
  (a) Country Name = US
  (b) State or Province Name = Massachusetts (type it out in full)
  (c) Locality Name = Cambridge
  (d) Organization Name = Massachusetts Institute of Technology
  (e) Organizational Unit Name = Information Systems
  (f) Common Name = icmit.mit.edu (this is the machine’s hostname)
  (g) Email Address = (none)
  (h) A challenge password = (anything you want – remember it in case your request is challenged)

4. Find the file /var/ssl/certs/req.pem and send it to jis@mit.edu including the BEGIN and END lines.

5. You will receive a certificate (between the BEGIN and END lines). It should be saved in /usr/local/ssl/certs/lipchitz.mit.edu.pem

   The Apache web server must be fully restarted before the new site certificate will take effect. If there is an error message it will show up in the SSL error logs in /usr/local/apache/var/log/ssl_error_log.*

**htdig search engine**

The icmit.mit.edu web server is also using the “htdig” search engine software to index the documents it serves. This allows quick searches within the icmit.mit.edu web document space.

The software source files are located in /usr/local/htdig, while the actual installation lives in /data4/projects/htdig. The relevant programs are located in the bin subdirectory and the configuration file is conf/htdig.conf.

Once the search engine is configured the only maintenance it requires is the weekly (or more often) indexing of the web server documents. This is done by the “rundig” script in the bin/rundig, which can be executed by the cron service at specific dates and times. These commands are listed in the file /var/spool/cron/crontabs/root.
A.2.4 NCFTPDP FTP Server

Another server we run on lipchitz.mit.edu is an FTP server. This particular FTP server is the ncftpd and not the standard ftplib nor the UWashington ftplib. It is faster and has more features than the other two.

The ncftpd is automatically started during lipchitz.mit.edu boot sequence, from the script /sbin/init.d/ncftpd. The configuration files for the FTP server are in /usr/local/etc/ncftpd, and the actual FTP site is in /data3/ftp.

Please keep in mind that the FTP protocol sends usernames and passwords in the clear, so a passive network sniffer on any subnet between you and lipchitz.mit.edu can get your password. File transfers can instead be done anonymously into the /data3/ftp/anon/incoming directory; this means that the FTP username will be “anonymous” and the password will be your e-mail address.

Another method of transferring files is to use the “scp” program, which stands for “secure copy”. This is one of the “ssh” suite of tools and can be used to copy files to and from remote locations:

lipchitz% scp -r -1 userdhost.mit.edu:/mit/user/thesis ./thesis

A.3 Printer Administration

The ICMIT group has an HP LaserJet 4000N printer named “akagi” which handles standard text and PostScript input. It does not have a duplexing module. The printer is not accessible from the Athena Computing Environment.

The set of PCs running Microsoft Windows NT 4.0 print through the primary domain controller, brancusi.mit.edu. The configuration is set there through the Windows NT printer administration tools.

The DEC Alpha machines lipchitz.mit.edu and klimt.mit.edu access “akagi” through the HP print server “yeager.mit.edu” using a BSD (Berkeley) style printing system. The relevant information for maintaining the printer queue is in the /etc/printcap file on lipchitz.

The Solaris workstation (miro.mit.edu) accesses akagi through an HP print server
known as “yeager.mit.edu”. Solaris printer tools are derivative of the AT&T system, as opposed to the more common and familiar BSD system. The set of commands to enable and configure printing on miro.mit.edu are:

miro# /usr/bin/disable akagi
miro# /usr/sbin/lpshut
miro# /usr/sbin/lpsystem -t bsd yeager.mit.edu
miro# /usr/sbin/lpadmin -p akagi -D "HP LaserJet 4000N in 3-253" -P letter -s yeager!akagi
miro# /usr/lib/lp/lpsched
miro# /usr/bin/enable akagi

<table>
<thead>
<tr>
<th>AT&amp;T</th>
<th>BSD</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>lp</td>
<td>lpr</td>
<td>Submits jobs for printing</td>
</tr>
<tr>
<td>lpsched</td>
<td>lpd</td>
<td>The printing daemon</td>
</tr>
<tr>
<td>lpshut</td>
<td>-</td>
<td>Stops the printing daemon</td>
</tr>
<tr>
<td>lpstat</td>
<td>lpq</td>
<td>Checks the status of a queue</td>
</tr>
<tr>
<td>cancel</td>
<td>lprm</td>
<td>Removes jobs from a queue</td>
</tr>
<tr>
<td>lpmove</td>
<td>-</td>
<td>Moves jobs among queues</td>
</tr>
<tr>
<td>lpadmin</td>
<td>/etc/printcap</td>
<td>Configures the printing system</td>
</tr>
<tr>
<td>accept</td>
<td>lpc enable</td>
<td>Enables queueing</td>
</tr>
<tr>
<td>reject</td>
<td>lpc disable</td>
<td>Disables queueing</td>
</tr>
<tr>
<td>enable</td>
<td>lpc start</td>
<td>Starts printing to a device</td>
</tr>
<tr>
<td>disable</td>
<td>lpc stop</td>
<td>Stops printing to a device</td>
</tr>
<tr>
<td></td>
<td>lpc topq</td>
<td>Re-orders jobs in the queue</td>
</tr>
</tbody>
</table>

Table A.1: Comparison of AT&T and BSD-style printing systems

A.4 On-Line Documentation

The ICMIT group is often engaged in extensive programming projects, and to facilitate the work we have a large physical library of manuals as well as copious on-line documentation.

A.4.1 ORBacus

Object Oriented Computing (http://www.ooc.com/) provides ORBacus – a full CORBA implementation distributed free of charge for non-commercial use. ORBacus is installed on giacometti.mit.edu (Linux), miro.mit.edu (Solaris), and our main server lipchitz.mit.edu (Digital Unix). It provides both Java and C++ implementations for
breadth of use.

The ORBacus Java .jar files are available with the same path for all three platforms, specifically /usr/local/lib/OB.jar. This should be added to the CLASSPATH environment variable of all ORBacus users. Documentation for ORBacus is available in Adobe Acrobat form at http://icmit.mit.edu/docs/ORBacus.

A.4.2 Informix JDBC

The Sun Ultrasparc server “miro.mit.edu” is running an Informix Universal Server database to handle our bioinformatics needs. Java Database Connectivity (JDBC) drivers are available on giacometti.mit.edu, miro.mit.edu, and lipchitz.mit.edu in /usr/local/informix/lib/ifxjdbc.jar on all three platforms. Documentation for the drivers is available at http://icmit.mit.edu/docs/ifxjdbc.

A.4.3 Java API

The Java 1.1.5 and 1.2 API specifications are available at http://icmit.mit.edu/docs/java.

A.5 Software

For remote X service you will probably want to use VNC, which provides a persistent client-server X architecture. The VNC server sends X updates to the VNC client; if the VNC client exits then the VNC server buffers the X requests until the client reconnects. The VNC client is available for all major platforms in native compiled format, as well as in Java form for additional portability. To start the server, run:

```
lipchitz% vncserver -geometry 1152x864 -depth 8
```

which will return to you a port number. You can then start up the VNC client on another machine and connect to ”lipchitz.mit.edu:<port number>” with your password.
A.5.1 Digital UNIX Administration

Use "setld -i" to list all the installed software packages. "setld" is used for almost all third-party software (official) installation.

A.5.2 Solaris Administration

The main Solaris software administration tools deal with adding and removing software packages. They are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkgadd</td>
<td>to add new software packages</td>
</tr>
<tr>
<td>pkgls</td>
<td>to list new software packages</td>
</tr>
<tr>
<td>pkgrm</td>
<td>to remove new software packages</td>
</tr>
</tbody>
</table>

Optional (non-system) software is usually installed in the /opt directory tree. See the website http://www.sunfreeware.com/ for some free Solaris software.
Appendix B

FML network and website administration

The Fluid Mechanics Laboratory (FML) as a whole has an internal network and a website for its students and staff. FML network and website administrators are required to managed the allocated IP addresses, keep the student and staff list up to date, and maintain the “fluids” locker and website.

B.1 FML locker and member lists

The “fluids” locker is available from an Athena Computing Environment workstation. The webpages are served from within its “/mit/fluids/www” subdirectory and contain general information such as lab duties and the FML seminar schedule in addition to the member lists and the allocated IP address.

When new lab members (graduate students, post-doctoral staff, and visiting scientists) join the FML, they will sign up via the “new member registration” page which will send the “fluids-www” mailing list the relevant information. The administrator will need to update the file /mit/fluids/www/students.html (for Master’s and PhD students) or /mit/fluids/www/faculty.html (for postdocs and visiting scientists) to include that information. In addition, students should be added to the “fml-students” mailing list with the command:
athena% blanche fml-students -a username

which will return immediately, but take effect overnight. If a student is to be deleted from the list, use:

athena% blanche fml-students -d username

If at all possible, grab a digital camera and take a picture of the new member to add to the web page.

B.2 FML IP address/hostname list

If a new workstation or server is brought into the FML, its users will usually want it to have network connectivity. The FML network administrator manages this task. The FML “purchases” a group of IP addresses from MIT Network Services, and allocates them internally because of high lab member and workstation turnover.

Lab IP addresses are allocated from the list in /mit/fluids/www/mit-only addresses.html, and once they run out we have no more. The FML network administrator must check every few months to see if workstations or servers have been deactivated or moved, so the IP addresses can be “freed up” on the list. Therefore the addresses.html file must always be kept up to date.

Physically the FML network is on the 18.80 subnet and is connected to the MITnet 18.80.0.1 router. Within the FML “loft” area we have 3 repeaters, with a fourth located in the main lab area. The repeaters are chained to each other, and don’t do any specific filtering. While machines may be connected to any of the repeaters, it is most efficient if those on the same subnet are connected to the same repeater.

B.3 Website Administration

Seminar announcements should be added to the FML web pages as appropriate. The web pages should also be kept up to date.
Bibliography


   http://www.who.int/hlt/historical/English/nomenclatures.html.

   G. Vaysseix, C. Helgesen, and P. Rodriguez-Tom’e. A proposal for a standard

[4] Bruce Birren, Eric Green, Phil Hieter, Sue Klapholz, and Rick Myers. Genome

[5] M. S. Bloise and E. H. Shortliffe. The computer meets medicine: Emergence of
   a discipline. In E. H. Shortliffe and L. E. Perreault, editors, Medical
   Informatics: Computer Applications in Health Care. Addison Wesley, Reading,

   modeling of integrated metabolic processes. In German Conference on

   (OPM) and OPM Data Management Tools.


database strategy for the microcirculation physiome project. paper, February
1999.

http://www.vex.net/ ben/corba/orbmatrix.html.


[17] National Center for Biotechnology Information. Genbank overview,

[18] Nathan Goodman, Steve Rozen, and Lincoln Stein. The case for componentry
in genome information systems. Technical report, Meeting on Interconnection
of Molecular Biology Databases, Stanford University, 1994.

P. Toussaint, and O. W. Weier. A CORBA-based integration of distributed


GATC.sql

-- Recreating the Affymetrix GATC DB on miro

-- General Subschema: probe array design
-- The following tables with their constituent fields are detailed
-- for Probe Array Design:
-- - chip_design (also called "probe array_design")
-- - analysis_scheme
-- - unit_type
-- - scheme_unit
-- - scheme_block
-- - scheme_atom
-- - scheme_cell
-- - biological_item

CREATE TABLE chip_design -- Also called "probe array_design"
(id INT PRIMARY KEY, -- key
 name VARCHAR (32) UNIQUE,
 number_x INT,
 number_y INT);

CREATE TABLE analysis_scheme
(id INT PRIMARY KEY, -- key
 chip_design_id INT REFERENCES chip_design (id),
 name VARCHAR (32) UNIQUE);

CREATE TABLE unit_type
(id INT PRIMARY KEY, -- key
 name VARCHAR (32) UNIQUE);

CREATE TABLE scheme_unit
(scheme_id INT REFERENCES analysis_scheme (id), -- key
 unit_idx INT, -- key
 type_id INT REFERENCES unit_type (id), -- key
 name VARCHAR (32),
 direction INT,
 mutation_id INT, -- key
 PRIMARY KEY (scheme_id, unit_idx));

CREATE TABLE biological_item
(id INT PRIMARY KEY, -- key
 item_name VARCHAR (32) UNIQUE);

CREATE TABLE scheme_block
(scheme_id INT, -- key
 unit_idx INT, -- key
 block_idx INT, -- key
 item_id INT REFERENCES biological_item (id),
 item_id INT REFERENCES scheme_unit (id),
 FOREIGN KEY (scheme_id, unit_idx, block_idx),
 PRIMARY KEY (scheme_id, unit_idx, block_idx, atom_idx),
 FOREIGN KEY (scheme_id, unit_idx, block_idx) REFERENCES scheme_block (id));

CREATE TABLE scheme_atom
(scheme_id INT, -- key
 unit_idx INT, -- key
 block_idx INT, -- key
 atom_idx INT, -- key
 cell_idx INT, -- key
 location_x INT, -- key
 location_y INT, -- key
 pbase CHAR,
 feature VARCHAR (32),
 qualifier VARCHAR (32),
 probe_length INT,
 flag INT,
 PRIMARY KEY (scheme_id, unit_idx, block_idx, atom_idx, cell_idx),
 FOREIGN KEY (scheme_id, unit_idx, block_idx, atom_idx) REFERENCES scheme_atom (id),
 FOREIGN KEY (scheme_id, unit_idx, block_idx) REFERENCES scheme_block (id));

-- General Subschema: protocol parameters
-- The following tables with their constituent fields are detailed
-- for Protocol Parameters:
-- - protocol
-- - parameter
-- - protocol_template
-- - parameter_template
-- - template_type
-- - parameter_units
-- - parameter_type

CREATE TABLE template_type
(id INT PRIMARY KEY, -- key
 name VARCHAR (32) UNIQUE);

CREATE TABLE protocol_tmpl
(id INT PRIMARY KEY, -- key
 type_id INT REFERENCES template_type (id), -- key
 name VARCHAR (32) UNIQUE);

CREATE TABLE protocol
(id INT PRIMARY KEY, -- key
 unit_idx INT, -- key
 block_idx INT, -- key
 atom_idx INT, -- key
 position INT, -- key
 unit_no INT,
 PRIMARY KEY (scheme_id, unit_idx, block_idx, atom_idx),
 FOREIGN KEY (scheme_id, unit_idx, block_idx) REFERENCES protocol_tmpl (id));

CREATE TABLE parameter
(id INT PRIMARY KEY, -- key
 parameter CLICK, -- key
 parameter_name VARCHAR (32), -- key
 parameter_type VARCHAR (32),
 foreign_key茆 parameter_tmpl (id),
 FOREIGN KEY (scheme_id, unit_idx, block_idx, atom_idx) REFERENCES parameter (id),
 FOREIGN KEY (scheme_id, unit_idx, block_idx) REFERENCES parameter_tmpl (id));

CREATE TABLE parameter_tmpl
(id INT PRIMARY KEY, -- key
 parameter_id INT REFERENCES parameter (id), -- key
 name VARCHAR (32) UNIQUE)
CREATE TABLE protocol
{
  id INT PRIMARY KEY,
  protocol_tmpl_id INT REFERENCES protocol_tmpl (id)
};

CREATE TABLE parameter
{
  protocol_id INT REFERENCES protocol (id),
  parameter_idx INT, -- key
  string_value VARCHAR (255), -- key
  PRIMARY KEY (protocol_id, parameter_idx)
};

CREATE TABLE parameter_units
{
  id INT PRIMARY KEY,
  name VARCHAR (32) UNIQUE
};

CREATE TABLE parameter_type
{
  id INT PRIMARY KEY,
  name VARCHAR (32) UNIQUE
};

CREATE TABLE parameter_tmpl
{
  protocol_tmpl_id INT REFERENCES protocol_tmpl (id), -- key
  protocol_id INT REFERENCES protocol (id), -- key
  parameter_idx INT, -- key
  string_value VARCHAR (255), -- key
  PRIMARY KEY (protocol_tmpl_id, parameter_idx)
};

-- General Subschema: experiment setup
-- The following tables with their constituent fields are detailed
-- for Experiment Setup:
-- - experiment
-- - target
-- - target_type
-- - physical_chip (also called "physical_probe_array")

CREATE TABLE target
{
  id INT PRIMARY KEY,
  protocol_id INT, -- key
  target_type_id INT, -- key
  concentration FLOAT, -- key
  date_prepared DATETIME YEAR TO DAY, -- key
  prepared_by VARCHAR (64) -- key
};

CREATE TABLE target_type
{
  id INT PRIMARY KEY,
  name VARCHAR (32) UNIQUE
};

CREATE TABLE physical_chip
{
  id INT PRIMARY KEY,
  design_id INT REFERENCES chip_design (id), -- key
  expiration_date DATETIME YEAR TO DAY, -- key
};

CREATE TABLE experiment
{
  id INT PRIMARY KEY,
  protocol_id INT REFERENCES protocol (id), -- key
  target_id INT REFERENCES target (id), -- key
  physical_chip_id INT REFERENCES physical_chip (id), -- key
  name VARCHAR (32) UNIQUE, -- key
  dat_file_name VARCHAR (255)
};

-- General Subschema: analysis results
-- The following tables with their constituent fields are detailed
-- for Analysis Results:
-- - analysis
-- - analysis_data_set_collection <-> analysis_data_coll
-- - analysis_data_set
-- - analysis_data_set_type <-> analysis_data_type
-- - analysis_algorithm
-- - algorithm_type
-- - measurement_element_result <-> measure_elem_rslt
-- - abs_gene_expr_result <-> abs_gene_result
-- - abs_gene_expr_result_type <-> abs_gene_result_t
-- - abs_gene_expr_atom_result <-> abs_gene_atom_rslt
-- - rel_gene_expr_result <-> rel_gene_result
-- - rel_gene_expr_result_type <-> rel_gene_result_t

CREATE TABLE analysis
{
  id INT PRIMARY KEY,
  protocol_id INT, -- key
  scheme_id INT, -- key
  algorithm_id INT, -- key
  data_set_coll_id INT, -- key
  analyst_id VARCHAR (32)
};

CREATE TABLE analysis_algorithm
{
  id INT PRIMARY KEY,
  type_id INT REFERENCES algorithm_type (id), -- key
  name VARCHAR (32) UNIQUE
};

CREATE TABLE analysis_data_coll
{
  id INT PRIMARY KEY
};

CREATE TABLE algorithm_type
{
  id INT PRIMARY KEY,
  name VARCHAR (32) UNIQUE
};

CREATE TABLE analysis_data_set
{
  id INT PRIMARY KEY
};

CREATE TABLE analysis_data_set_collection
{
  id INT PRIMARY KEY
};

CREATE TABLE analysis_data_type
{
  id INT PRIMARY KEY
};

CREATE TABLE analysis_scheme
{
  id INT PRIMARY KEY
};

CREATE TABLE analysis_result
{
  id INT PRIMARY KEY
};

CREATE TABLE abs_gene_expr_result
{
  id INT PRIMARY KEY
};

CREATE TABLE abs_gene_expr_result_type
{
  id INT PRIMARY KEY
};

CREATE TABLE abs_gene_expr_atom_result
{
  id INT PRIMARY KEY
};

CREATE TABLE rel_gene_expr_result
{
  id INT PRIMARY KEY
};

CREATE TABLE rel_gene_expr_result_type
{
  id INT PRIMARY KEY
};

CREATE TABLE rel_gene_expr_atom_result
{
  id INT PRIMARY KEY
};
CREATE TABLE analysis_data_type
(
    id INT PRIMARY KEY, -- key
    name VARCHAR (32) UNIQUE
);

CREATE TABLE analysis_data_set
(
    id INT PRIMARY KEY, -- key
    collection_id INT REFERENCES analysis_data_coll (id),
    analysis_id INT REFERENCES analysis (id),
    expt_id INT REFERENCES experiment (id),
    type_id INT REFERENCES analysis_data_type (id)
);

CREATE TABLE measure_elem_rslt
(
    analysis_id INT REFERENCES analysis (id), -- key
    location_x INT, -- key
    location_y INT, -- key
    intensity FLOAT, -- key
    statistic FLOAT, -- key
    pixels INT, -- key
    flag INT, -- key
    intensity_orig FLOAT, -- key
    PRIMARY KEY (analysis_id, location_x, location_y)
);

CREATE TABLE abs_gene_result
(
    id INT PRIMARY KEY, -- key
    name VARCHAR (32) UNIQUE
);

CREATE TABLE abs_gene_result_t
(
    id INT PRIMARY KEY, -- key
    name VARCHAR (32) UNIQUE
);

CREATE TABLE abs_gene_result
(
    analysis_id INT REFERENCES analysis (id), -- key
    item_id INT REFERENCES biological_item (id), -- key
    type_id INT REFERENCES abs_gene_result_t (id),
    number_positive INT,
    number_negative INT,
    number_used INT,
    number_all INT,
    avg_log_ratio FLOAT,
    pm_excess INT,
    number_in_avg INT,
    nm_excess INT,
    avg_diff_intens FLOAT,
    PRIMARY KEY (analysis_id, item_id)
);

CREATE TABLE abs_gene_atom_rslt
(
    analysis_id INT REFERENCES analysis (id), -- key
    unit_idx INT, -- key
    block_idx INT, -- key
    atom_idx INT, -- key
    background_intens FLOAT,
    flag INT,
    PRIMARY KEY (analysis_id, unit_idx, block_idx, atom_idx)
);
<xml version="1.0" >
<!DOCTYPE datadefs PUBLIC
"-//ICMIT//SPL DataServer Configuration::19990417//EN"
"datadefs.dtd">
<datadefs>
<classdef id='analysis_scheme' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='chip_design_id' desc=''>
    <dbColumn column='chip_design_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>
    </element>
  </attrib>
</classdef>
<classdef id='unit_type' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>
    </element>
  </attrib>
</classdef>
<classdef id='scheme_unit' desc=''>
  <attrib id='direction' desc=''>
    <dbColumn column='direction'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='scheme_id' desc=''>
    <dbColumn column='scheme_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='unit_idx' desc=''>
    <dbColumn column='unit_idx'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='type_id' desc=''>
    <dbColumn column='type_id'/>
  </attrib>
  <attrib id='mutation_id' desc=''>
    <dbColumn column='mutation_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>
    </element>
  </attrib>
</classdef>
<classdef id='biological_item' desc=''>
  <attrib id='item_name' desc=''>
    <dbColumn column='item_name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>
    </element>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
</classdef>
<classdef id='scheme_block' desc=''>
  <attrib id='itemid' desc=''>
    <dbColumn column='itemjid'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='schemejid' desc=''>
    <dbColumn column='scheme-id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='unitjidx' desc=''>
    <dbColumn column='unit-idx'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
  <attrib id='block_idx' desc=''>
    <dbColumn column='block-idx'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
</classdef>
<classdef id='scheme_atom' desc=''>
  <attrib id='tbase' desc=''>
    <element type='short'>
      <dbType distinct='no' type='integer'/>
    </element>
  </attrib>
</classdef>
</datadefs>
<classdef id='parameter_units' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>

<classdef id='parameter_type' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>

<classdef id='parameter_tmpl' desc=''>
  <attrib id='units_id' desc=''>
    <dbColumn column='units_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='string_value' desc=''>
    <dbColumn column='string_value'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
  <attrib id='protocol_tmpl_id' desc=''>
    <dbColumn column='protocol_tmpl_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='type_id' desc=''>
    <dbColumn column='type_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='parameter_idx' desc=''>
    <dbColumn column='parameter_idx'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>

<classdef id='protocol' desc=''>
  <attrib id='template_id' desc=''>
    <dbColumn column='template_id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
</classdef>

<classdef id='protocol_tmpl' desc=''>
  <attrib id='typeid' desc=''>
    <dbColumn column='type-id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>

<classdef id='parameter' desc=''>
  <attrib id='protocolid' desc=''>
    <dbColumn column='protocol-id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='string-value' desc=''>
    <dbColumn column='string-value'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
  <attrib id='parameter-idx' desc=''>
    <dbColumn column='arameter-idx'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
</classdef>

<classdef id='parametertype' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>

<classdef id='parameter-units' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
      <dbType distinct='no' type='integer'/>  
    </element>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
      <dbType distinct='no' type='varchar'/>  
    </element>
  </attrib>
</classdef>
<classdef id='target_type' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'/>
  </attrib>
</classdef>

<classdef id='target' desc=''>
  <attrib id='protocolid' desc=''>
    <dbColumn column='protocol-id'/>
    <element type='short'/>
  </attrib>
  <attrib id='target-type-id' desc=''>
    <dbColumn column='target-typeid'/>
    <element type='short'/>
  </attrib>
  <attrib id='preparedby' desc=''>
    <dbColumn column='prepared-by'/>
    <element type='string'/>
  </attrib>
  <attrib id='concentration' desc=''>
    <dbColumn column='concentration'/>
    <element type='float'/>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
  <attrib id='dateprepared' desc=''>
    <dbColumn column='dateprepared'/>
    <element type='datetime'/>
  </attrib>
</classdef>

<classdef id='physical-chip' desc=''>
  <attrib id='expiration date' desc=''>
    <dbColumn column='expirationdate'/>
    <element type='datetime'/>
  </attrib>
  <attrib id='design-id' desc=''>
    <dbColumn column='designid'/>
    <element type='short'/>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
</classdef>

<classdef id='experiment' desc=''>
  <attrib id='protocolid' desc=''>
    <dbColumn column='protocol-id'/>
    <element type='short'/>
  </attrib>
  <attrib id='physical-chip-id' desc=''>
    <dbColumn column='physical-chip-id'/>
    <element type='short'/>
  </attrib>
  <attrib id='dat_filemname' desc=''>
    <dbColumn column='dat-file-name'/>
    <element type='string'/>
  </attrib>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
  <attrib id='targetid' desc=''>
    <dbColumn column='target-id'/>
    <element type='short'/>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'/>
  </attrib>
</classdef>

<classdef id='algorithm_type' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
</classdef>
<element type='short'>
    <dbType distinct='no' type='integer'/>
</element>
</attrib>
<attrib id='analyst_id' desc=''>
    <dbColumn column='analyst_id'/>
    <element type='string'>
        <dbType distinct='no' type='varchar'/>
    </element>
</attrib>
</attrib>
<attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'>
        <dbType distinct='no' type='integer'/>
    </element>
</attrib>
</attrib>
<attrib id='analysis_date' desc=''>
    <dbColumn column='analysis_date'/>
    <element type='datetime'>
        <dbType distinct='no' type='datetime'/>
    </element>
</attrib>
</attrib>
<attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'>
        <dbType distinct='no' type='varchar'/>
    </element>
</attrib>
</attrib>
</classdef>
<classdef id='analysis-data-type' desc=''>
    <attrib id='id' desc=''>
        <dbColumn column='id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='name' desc=''>
        <dbColumn column='name'/>
        <element type='string'>
            <dbType distinct='no' type='varchar'/>
        </element>
    </attrib>
</classdef>
<classdef id='analysis-data-set' desc=''>
    <attrib id='collectionid' desc=''>
        <dbColumn column='collectionid'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='analysis-id' desc=''>
        <dbColumn column='analysis-id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='exptjid' desc=''>
        <dbColumn column='expt-jid'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
</classdef>
</classdef>
<classdef id='analysis-algorithm' desc=''>
    <attrib id='typeid' desc=''>
        <dbColumn column='type-id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='id' desc=''>
        <dbColumn column='id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='name' desc=''>
        <dbColumn column='name'/>
        <element type='string'>
            <dbType distinct='no' type='varchar'/>
        </element>
    </attrib>
</classdef>
<classdef id='analysis-data-coll' desc=''>
    <attrib id='id' desc=''>
        <dbColumn column='id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
</classdef>
</classdef>
<classdef id='analysis' desc=''>
    <attrib id='data-set-colljid' desc=''>
        <dbColumn column='data-set-coll-id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='algorithmid' desc=''>
        <dbColumn column='algorithmid'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='protocol-id' desc=''>
        <dbColumn column='protocol-id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='schemeid' desc=''>
        <dbColumn column='scheme-id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='analystid' desc=''>
        <dbColumn column='analyst-id'/>
        <element type='string'>
            <dbType distinct='no' type='varchar'/>
        </element>
    </attrib>
    <attrib id='id' desc=''>
        <dbColumn column='id'/>
        <element type='short'>
            <dbType distinct='no' type='integer'/>
        </element>
    </attrib>
    <attrib id='analysis-date' desc=''>
        <dbColumn column='analysis-date'/>
        <element type='datetime'>
            <dbType distinct='no' type='datetime'/>
        </element>
    </attrib>
    <attrib id='name' desc=''>
        <dbColumn column='name'/>
        <element type='string'>
            <dbType distinct='no' type='varchar'/>
        </element>
    </attrib>
</classdef>
<classdef id='measure_element_result' desc=''>
  <attrib id='intensity_orig' desc=''>
    <dbColumn column='intensity_orig'/>
    <element type='float'/>
  </attrib>
  <attrib id='intensity' desc=''>
    <dbColumn column='intensity'/>
    <element type='float'/>
  </attrib>
  <attrib id='pixels' desc=''>
    <dbColumn column='pixels'/>
    <element type='short'/>
  </attrib>
  <attrib id='analysis_id' desc=''>
    <dbColumn column='analysis_id'/>
    <element type='short'/>
  </attrib>
  <attrib id='statistic' desc=''>
    <dbColumn column='statistic'/>
    <element type='float'/>
  </attrib>
  <attrib id='flag' desc=''>
    <dbColumn column='flag'/>
    <element type='short'/>
  </attrib>
  <attrib id='location_y' desc=''>
    <dbColumn column='location_y'/>
    <element type='short'/>
  </attrib>
  <attrib id='location_x' desc=''>
    <dbColumn column='location_x'/>
    <element type='short'/>
  </attrib>
</classdef>

<classdef id='abs_gene_result_t' desc=''>
  <attrib id='id' desc=''>
    <dbColumn column='id'/>
    <element type='short'/>
  </attrib>
  <attrib id='name' desc=''>
    <dbColumn column='name'/>
    <element type='string'/>
  </attrib>
</classdef>

<classdef id='abs_gene_result' desc=''>
  <attrib id='avg_diff_intens' desc=''>
    <dbColumn column='avg_diff_intens'/>
    <element type='float'/>
  </attrib>
  <attrib id='mm_excess' desc=''>
    <dbColumn column='mm_excess'/>
    <element type='short'/>
  </attrib>
  <attrib id='number_in_avg' desc=''>
    <dbColumn column='number_in_avg'/>
    <element type='short'/>
  </attrib>
  <attrib id='avg_log_ratio' desc=''>
    <dbColumn column='avg_log_ratio'/>
    <element type='float'/>
  </attrib>
  <attrib id='pm_excess' desc=''>
    <dbColumn column='pm_excess'/>
    <element type='short'/>
  </attrib>
  <attrib id='number_all' desc=''>
    <dbColumn column='number_all'/>
    <element type='short'/>
  </attrib>
  <attrib id='analysis_id' desc=''>
    <dbColumn column='analysis_id'/>
    <element type='short'/>
  </attrib>
</classdef>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='number_negative' desc=''>
<dbColumn column='number_negative'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='typeid' desc=''>
<dbColumn column='type-id'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='number_positive' desc=''>
<dbColumn column='number-positive'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='numberused' desc=''>
<dbColumn column='number_used'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

</classdef>

<classdef id='abs-gene-atomrslt' desc=''>
<attrib id='atom_idx' desc=''>
<dbColumn column='atom-idx'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='analysis-id' desc=''>
<dbColumn column='analysis-id'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='background-intens' desc=''>
<dbColumn column='background_intens'/>
<element type='float'>
<dbType distinct='no' type='float'/>
</element>
</attrib>

<attrib id='unit-idx' desc=''>
<dbColumn column='unit-idx'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='flag' desc=''>
<dbColumn column='flag'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='block-idx' desc=''>
<dbColumn column='block-idx'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

</classdef>

<classdef id='rel-gene-result-t' desc=''>
<attrib id='id' desc=''>
<dbColumn column='id'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='name' desc=''>
<dbColumn column='name'/>
<element type='string'>
<dbType distinct='no' type='varchar'/>
</element>
</attrib>

</classdef>

<classdef id='relgene-result' desc=''>
<attrib id='decrease-ratio' desc=''>
<dbColumn column='decrease-ratio'/>
<element type='float'>
<dbType distinct='no' type='float'/>
</element>
</attrib>

<attrib id='number-increase' desc=''>
<dbColumn column='number_increase'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='itemid' desc=''>
<dbColumn column='itemid'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='pos-delta' desc=''>
<dbColumn column='pos-delta'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='avg-diff-rounding' desc=''>
<dbColumn column='avgdiff_rounding'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='logavgratio' desc=''>
<dbColumn column='log-avg-ratio'/>
<element type='float'>
<dbType distinct='no' type='float'/>
</element>
</attrib>

<attrib id='analysis-id' desc=''>
<dbColumn column='analysis-id'/>
<element type='short'>
<dbType distinct='no' type='integer'/>
</element>
</attrib>

<attrib id='dpos-dneg-ratio' desc=''>
<dbColumn column='dpos_dneg_ratio'/>
<element type='float'>
<dbType distinct='no' type='float'/>
</element>
</attrib>

</classdef>
..//gatc-XML-DTD

<class id='abs_gene_result' classdef='abs_gene_result' primaryKey='analysis_id' desc=''
  <dbTable table='abs_gene_result'/>
</class>

<class id='abs_gene_atom_rslt' classdef='abs_gene_atom_rslt' primaryKey='unit_idx' desc=''
  <dbTable table='abs_gene_atom_rslt'/>
</class>

<class id='rel_gene_result_t' classdef='rel_gene_result_t' primaryKey='id' desc=''
  <dbTable table='rel_gene_result_t'/>
  <relationship class='rel_gene_result' attrib='type_id' type='1:1' desc=''
</relationship>
</class>

<class id='rel_gene_result' classdef='rel_gene_result' primaryKey='analysis_id' desc=''
  <dbTable table='rel_gene_result'/>
</class>

</datadefs>
-- Class: Contact
create row type Contact_t
{
  _oid    numeric (10,0) not null,
  accessionID  varchar (50) not null,
  displayName varchar (200) not null,
  modDate     datetime year to day not null,
  objectClass varchar (30) not null,
  version     int not null,
  searchName  varchar (200) not null,
};
create table Contact of type Contact_t
( primary key (_oid) );

-- Class: AccessionedObject -> AccessObject
create row type AccessObject_t
{
  _aid    varchar (255) not null,
  submitter numeric (10,0) not null,
  name     varchar (255) not null,
};
create table AccessObject of type AccessObject_t
( primary key (_aid),
  foreign key (submitter) references Contact
 );

-- Class: ASOTDict -> ASOTDict
create row type ASOTDict_t
{
  _code varchar (20) not null,
  _defn varchar (255),
  _value varchar (255) not null,
};
create table ASOTDict of type ASOTDict_t
( primary key (_code) );
insert into ASOTDict (_code, _value) values
( 'WildType', 'Wild Type' ),
insert into ASOTDict (_code, _value) values
( 'Mutant', 'Mutant' ),
insert into ASOTDict (_code, _value) values
( 'Unknown', 'Unknown' );

-- Class: AnnotationTypeDict -> AnnotationTypeDict
create row type AnnoTDict_t
{
  _code varchar (20) not null,
  _defn varchar (255),
  _value varchar (255) not null,
};
create table AnnoTDict of type AnnoTDict_t
( primary key (_code) );
insert into AnnoTDict (_code, _value) values
( 'Comment', 'Comment' ),
insert into AnnoTDict (_code, _value) values
( 'ChangeRequest', 'ChangeRequest' ),
insert into AnnoTDict (_code, _value) values
( 'GDB 5.x', 'GDB 5.x history' ),
insert into AnnoTDict (_code, _value) values
( 'DeleteRequest', 'DeleteRequest' ),
insert into AnnoTDict (_code, _value) values
( 'CascadeDeleted', 'CascadeDeleted' ),
insert into AnnoTDict (_code, _value) values
( 'ChangeOwner', 'ChangeOwner' );

-- Class: AnnotationTypeDict
create row type AdminAnnoTDict_t
{
  _code varchar (20) not null,
  _defn varchar (255),
  _value varchar (255) not null,
};
create table AdminAnnoTDict of type AdminAnnoTDict_t
( primary key (_code) );
insert into AdminAnnoTDict (_code, _value) values
( 'IncludeInReport', 'Include in Annual Report' ),
insert into AdminAnnoTDict (_code, _value) values
( 'checkedByCC', 'Checked by CC' );

-- Class: AminoAcidDict
create row type AminoAcidDict_t
{
  _code varchar (3) not null,
  _defn varchar (255),
  _value varchar (255) not null,
};
create table AminoAcidDict of type AminoAcidDict_t
( primary key (_code) );
insert into AminoAcidDict (_code, _value) values
( 'ala', 'Alanine' ),
insert into AminoAcidDict (_code, _value) values
( 'arg', 'Arginine' ),
insert into AminoAcidDict (_code, _value) values
( 'asn', 'Asparagine' ),
insert into AminoAcidDict (_code, _value) values
( 'asp', 'Aspartic acid' ),
insert into AminoAcidDict (_code, _value) values
( 'cys', 'Cysteine' ),
insert into AminoAcidDict (_code, _value) values
( 'gln', 'Glutamine' ),
insert into AminoAcidDict (_code, _value) values
( 'glu', 'Glutamic acid' ),
insert into AminoAcidDict (_code, _value) values
( 'gly', 'Glycine' ),
insert into AminoAcidDict (_code, _value) values
( 'his', 'Histidine' ),
insert into AminoAcidDict (_code, _value) values
( 'ile', 'Isoleucine' ),
insert into AminoAcidDict (_code, _value) values
( 'leu', 'Leucine' ),
insert into AminoAcidDict (_code, _value) values
( 'lys', 'Lysine' ),
insert into AminoAcidDict (_code, _value) values
( 'met', 'Methionine' ),
insert into AminoAcidDict (_code, _value) values
( 'phe', 'Phenylalanine' ),
insert into AminoAcidDict (_code, _value) values
( 'pro', 'Proline' ),
insert into AminoAcidDict (_code, _value) values
( 'phe', 'Phe' ),
insert into AminoAcidDict (_code, _value) values
( 'ser', 'Serine' ),
insert into AminoAcidDict (_code, _value) values
( 'thr', 'Threonine' ),
insert into AminoAcidDict (_code, _value) values
( 'tre', 'Threonine' ),
insert into AminoAcidDict (_code, _value) values
( 'trp', 'Tryptophan' ),
insert into AminoAcidDict (_code, _value) values
( 'tyr', 'Tyrosine' ),
insert into AminoAcidDict (_code, _value) values
( 'val', 'Valine' );
## AminoAcidDict

- **gln**: `Glutamic acid`
- **glu**: `Glycine`
- **his**: `Histidine`
- **ile**: `Isoleucine`
- **leu**: `Leucine`
- **lys**: `Lysine`
- **met**: `Methionine`
- **phe**: `Phenylalanine`
- **pro**: `Proline`
- **ser**: `Serine`
- **thr**: `Threonine`
- **trp**: `Tryptophan`
- **tyr**: `Tyrosine`
- **val**: `Valine`

## ApprovalStatusDict

- **Unreviewed**
- **Approved**
- **Rejected**
- **RejectLeft**
- **RejectRight**
- **Superseded**
- **UnderReview**
- **InConflict**

## AvailabilityDict

- **Unknown**
- **FreelyAvailable**
- **Collaborators**
- **Future**
- **Unavailable**
- **With Restrictions**

## BreakpointCauseDict

- **Unknown**
- **Natural**
- **Experimental**

## CellLineTypeDict

- **Unknown**
- **SCH** (Somatic Cell Hybrid)

---

```
create table AminoAcidDict ( _code varchar (20) not null, _defn varchar (255), _value varchar (255) not null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'gln', 'Glutamic acid', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'glu', 'Glycine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'his', 'Histidine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'ile', 'Isoleucine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'leu', 'Leucine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'lys', 'Lysine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'met', 'Methionine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'phe', 'Phenylalanine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'pro', 'Proline', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'ser', 'Serine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'thr', 'Threonine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'trp', 'Tryptophan', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'tyr', 'Tyrosine', null );
insert into AminoAcidDict ( _code, _defn, _value ) values ( 'val', 'Valine', null );
```
insert into CellLine_TDict (_code, _value) values (
    'Lymphoblast', 'Lymphoblast' 
    );
insert into CellLine_TDict (_code, _value) values (
    'Lymphocyte', 'Lymphocyte' 
    );
insert into CellLine_TDict (_code, _value) values (
    'Fibroblast', 'Fibroblast' 
    );
insert into CellLine_TDict (_code, _value) values (
    'Tumor', 'Tumor' 
    );
insert into CellLine_TDict (_code, _value) values (
    'RH', 'Radiation Hybrid' 
    );
insert into CellLine_TDict (_code, _value) values (
    'Other', 'Other' 
    );

-- Class: CompartmentDict -> CmprtDict
-- create row type CmprtDict_t
| _code varchar (20) not null, |
| _defn varchar (255), |
| _value varchar (255) not null |
create table CmprtDict of type CmprtDict-t 
( primary key (_code) );
insert into CmprtDict (_code, _value) values (
    'Nucleus', 'Nucleus' 
    );
insert into CmprtDict (_code, _value) values (
    'Mitochondrion', 'Mitochondrion' 
    );

-- Class: CytogeneticBandDict -> CytoGBDict
-- create row type CytoGBDict_t
| _code varchar (20) not null, |
| _defn varchar (255), |
| _value varchar (255) not null |
create table CytoGBDict of type CytoGBDict-t 
( primary key (_code) );
insert into CytoGBDict (_code, _value) values (
    'RBand', 'R-Band' 
    );
insert into CytoGBDict (_code, _value) values (
    'GBand', 'G-Band' 
    );
insert into CytoGBDict (_code, _value) values (
    'Satellite', 'Satellite' 
    );
insert into CytoGBDict (_code, _value) values (
    'Telomere', 'Telomere' 
    );
insert into CytoGBDict (_code, _value) values (
    'Centromere', 'Centromere' 
    );

-- Class: CytogeneticResolutionsDict -> CytoGRDict
-- create row type CytoGRDict_t
| _code varchar (20) not null, |
| _defn varchar (255), |
| _value varchar (255) not null |
create table CytoGRDict of type CytoGRDict-t 
( primary key (_code) );
insert into CytoGRDict (_code, _value) values (
    '400', '400' 
    );
insert into CytoGRDict (_code, _value) values (
    '550', '550' 
    );
insert into CytoGRDict (_code, _value) values (
    '850', '850' 
    );

-- Class: DBObjectStatusDict -> DB_OSDict
-- create row type DB_OSDict_t
| _code varchar (20) not null, |
| _defn varchar (255), |
| _value varchar (255) not null |
create table DB_OSDict of type DB_OSDict-t 
( primary key (_code) );
insert into DB_OSDict (_code, _value) values (
    'Active', 'Active' 
    );
insert into DB_OSDict (_code, _value) values (
    'Merged', 'Merged' 
    );
insert into DB_OSDict (_code, _value) values (
    'Split', 'Split' 
    );
insert into DB_OSDict (_code, _value) values (
    'Conflict', 'Conflict' 
    );
insert into DB_OSDict (_code, _value) values (
    'Deactivated', 'Deactivated' 
    );

-- Class: DNTTypeDict -> DNA_TDict
-- create row type DNA_TDict_t
| _code varchar (20) not null, |
| _defn varchar (255), |
| _value varchar (255) not null |
create table DNA_TDict of type DNA_TDict-t 
( primary key (_code) );
insert into DNA_TDict (_code, _value) values (
    'Unknown', 'Unknown' 
    );
insert into DNA_TDict (_code, _value) values (
    'Genomic', 'Genomic DNA' 
    );
insert into DNATDict (_code, _value) values ('Synthetic', 'Synthetic DNA');
insert into DNATDict (_code, _value) values ('Viral', 'Viral DNA');
insert into DNATDict (_code, _value) values ('cDNA', 'cDNA');

-- Class: EndpointDict
-- create row type EndpointDict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table EndpointDict of type EndpointDict_t 
| primary key (_code) |
insert into EndpointDict (_code, _value) values ('Entire', 'Entire');
insert into EndpointDict (_code, _value) values ('Start', 'Start');
insert into EndpointDict (_code, _value) values ('End', 'End');

-- Class: EnzymeUseDict -> EnzUseDict
-- create row type EnzUseDict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table EnzUseDict of type EnzUseDict_t 
| primary key (_code) |
insert into EnzUseDict (_code, _value) values ('SingleDigest', 'Single Digest');
insert into EnzUseDict (_code, _value) values ('DoubleDigest', 'Double Digest');
insert into EnzUseDict (_code, _value) values ('AlternateDigest', 'Alternate Digest');

-- Class: ErrorTypeDict -> Error_TDict
-- create row type Error_TDict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table Error_TDict of type Error_TDict_t 
| primary key (_code) |
insert into Error_TDict (_code, _value) values ('StdErr', 'Standard Error');
insert into Error_TDict (_code, _value) values ('LOD', 'LOD');

-- Class: FontStyleDict -> FontSDict
-- create row type FontSDict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table FontSDict of type FontSDict_t 
| primary key (_code) |
insert into FontSDict (_code, _value) values ('Normal', 'Normal');
insert into FontSDict (_code, _value) values ('Bold', 'Bold');
insert into FontSDict (_code, _value) values ('Italic', 'Italic');
insert into FontSDict (_code, _value) values ('Underlined', 'Underlined');

-- Class: FragileSiteAgentDict -> FragileSADict
-- create row type FragileSADict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table FragileSADict of type FragileSADict_t 
| primary key (_code) |
insert into FragileSADict (_code, _value) values ('Unclassified', 'Unclassified');
insert into FragileSADict (_code, _value) values ('Azacytidine5', '5-Azacytidine type');
insert into FragileSADict (_code, _value) values ('BrdU', 'BrdU type');
insert into FragileSADict (_code, _value) values ('Aphidicolin', 'Aphidicolin type');
insert into FragileSADict (_code, _value) values ('DistamycinA', 'Distamycin A type');
insert into FragileSADict (_code, _value) values ('FolicAcid', 'Folic acid type');

-- Class: GeneElementTypeDict -> GeneElem_TDict
-- create row type GeneElem_TDict_t
| _code     | varchar (20) not null,  
| _defn     | varchar (255),  
| _value    | varchar (255) not null |
create table GeneElem_TDict of type GeneElem_TDict_t 
| primary key (_code) |
create table GeneElemTDict of type GeneElemTDict_t
( primary key (_code) );

insert into GeneElemTDict (_code, _value) values 'Other', 'Other'
insert into GeneElemTDict (_code, _value) values 'Intron', 'Intron'
insert into GeneElemTDict (_code, _value) values 'Exon', 'Exon'
insert into GeneElemTDict (_code, _value) values 'TranscriptionStart', 'Transcription Start Site'
insert into GeneElemTDict (_code, _value) values 'Polyadenylation', 'Polyadenylation site'

create table GrowthSDict of type GrowthSDict-t
( primary key (_code) );

insert into GrowthSDict (_code, _value) values 'Unknown', 'Unknown'
insert into GrowthSDict (_code, _value) values 'Fetal', 'Fetal'
insert into GrowthSDict (_code, _value) values 'Adult', 'Adult'
insert into GrowthSDict (_code, _value) values 'Infant', 'Infant'
insert into GrowthSDict (_code, _value) values 'Child', 'Child'

create table IMethodDict of type IMethodDict-t
( primary key (_code) );

insert into IMethodDict (_code, _value) values 'Flowsorted', 'Flow Sorted'
insert into IMethodDict (_code, _value) values 'Other', 'Other'

create table LibLocTDict of type LibLocTDict-t
( primary key (_code) );

insert into LibLocTDict (_code, _value) values 'Original', 'Original'
insert into LibLocTDict (_code, _value) values 'Replated', 'Replated'

create table LineSDict of type LineSDict-t
( primary key (_code) );

insert into LineSDict (_code, _value) values 'Solid', 'Solid'
insert into LineSDict (_code, _value) values 'Dashed', 'Dashed'
insert into LineSDict (_code, _value) values 'Dotted', 'Dotted'

create table MapSexDict of type MapSexDict-t
( primary key (_code) );

insert into MapSexDict (_code, _value) values 'Unknown', 'Unknown'
insert into MapSexDict (_code, _value) values 'Male', 'Male'
insert into MapSexDict (_code, _value) values 'Female', 'Female'
```sql
create table MapSexDict (_code, _value) values
( 'SexAveraged', 'Sex-averaged' );

-- Class: NameStatusDict -> NameStatDict
--
class row type NameStatDict_t
{
    _code smallint not null,
    _defn varchar (255),
    _value varchar (255) not null
}

create table NameStatDict of type NameStatDict_t
( primary key (_code) );

insert into NameStatDict (_code, _value) values
( 'l', 'Alias' );
insert into NameStatDict (_code, _value) values
( '0', 'Primary' );
insert into NameStatDict (_code, _value) values
( '2', 'Obsolete' );
insert into NameStatDict (_code, _value) values
( '3', 'Merge' );
insert into NameStatDict (_code, _value) values
( 4 , 'Split' );

-- Class: NameTypeDict
--
class row type NameTypeDict_t
{
    _code smallint not null,
    _defn varchar (255),
    _value varchar (255) not null
}

create table NameTypeDict of type NameTypeDict_t
( primary key (_code) );

insert into NameTypeDict (_code, _value) values
( '2', 'Name' );
insert into NameTypeDict (_code, _value) values
( '0', 'Symbol' );
insert into NameTypeDict (_code, _value) values
( '1', 'DNA Segment Number' );
insert into NameTypeDict (_code, _value) values
( '3', 'Accession' );
insert into NameTypeDict (_code, _value) values
( '4', 'ORF Number' );

-- Class: ObservationTypeDict
--
class row type ObsTDict_t
{
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
}

create table Obs_TDict of type ObsTDict_t
( primary key (_code) );

insert into Obs_TDict (_code, _value) values
( 'Unknown', 'Unknown' );
insert into Obs_TDict (_code, _value) values
( 'Amplifies', 'Amplifies from' );
insert into Obs_TDict (_code, _value) values
( 'TemplateFor', 'Is template for' );
insert into Obs_TDict (_code, _value) values
( 'Hybridizes', 'Hybridizes with' );
insert into Obs_TDict (_code, _value) values
( 'EndClone', 'Subcloned from end of' );
insert into Obs_TDict (_code, _value) values
( 'HasEndClone', 'Has end clone' );
insert into Obs_TDict (_code, _value) values
( 'Subclone', 'Subcloned from' );
insert into Obs_TDict (_code, _value) values
( 'HasSubclone', 'Has subclone' );
insert into Obs_TDict (_code, _value) values
( 'EndSeq', 'Sequenced from end of' );
insert into Obs_TDict (_code, _value) values
( 'SourceEndSeq', 'Source of end sequence for' );
insert into Obs_TDict (_code, _value) values
( 'Seq', 'Sequenced from' );
insert into Obs_TDict (_code, _value) values
( 'SourceSeq', 'Source of sequence for' );
insert into Obs_TDict (_code, _value) values
( 'SeqIdent', 'Shares sequence identity with' );
insert into Obs_TDict (_code, _value) values
( 'Copy', 'Copied from' );
insert into Obs_TDict (_code, _value) values
( 'HasCopy', 'Has copy' );
insert into Obs_TDict (_code, _value) values
( 'Fingerprint', 'Shares Fingerprint with' );

-- Class: ObservationTypePlusDict
--
class row type ObsTPlusDict_t
{
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
}

create table Obs_TPlusDict of type ObsTPlusDict_t
( primary key (_code) );

insert into Obs_TPlusDict (_code, _value) values
( 'Unknown', 'Unknown' );
insert into Obs_TPlusDict (_code, _value) values
( 'Amplifies', 'Amplifies from' );
insert into Obs_TPlusDict (_code, _value) values
( 'TemplateFor', 'Is template for' );
insert into Obs_TPlusDict (_code, _value) values
( 'Hybridizes', 'Hybridizes with' );
insert into Obs_TPlusDict (_code, _value) values
( 'EndClone', 'Subcloned from end of' );
insert into Obs_TPlusDict (_code, _value) values
( 'HasEndClone', 'Has end clone' );
```

insert into ObsTPlusDict (_code, _value) values
('Subclone', 'Subcloned from');
insert into ObsTPlusDict (_code, _value) values
('HasSubclone', 'Has subclone');
insert into ObsTPlusDict (_code, _value) values
('EndSeq', 'Sequenced from end of');
insert into ObsTPlusDict (_code, _value) values
('SourceEndSeq', 'Source of end sequence for');
insert into ObsTPlusDict (_code, _value) values
('Seq', 'Sequenced from');
insert into ObsTPlusDict (_code, _value) values
('SourceSeq', 'Source of sequence for');
insert into ObsTPlusDict (_code, _value) values
('Copy', 'Copied from');
insert into ObsTPlusDict (_code, _value) values
('HasCopy', 'Has copy');
insert into ObsTPlusDict (_code, _value) values
('Fingerprint', 'Shares Fingerprint with');

create row type OrderRelDict_t
(_code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null);

create table OrderRelDict of type OrderRelDict_t
(primary key (_code));

insert into OrderRelDict (_code, _value) values
('Overlaps', 'Overlaps');
insert into OrderRelDict (_code, _value) values
('Gap', 'No Overlap');
insert into OrderRelDict (_code, _value) values
('Inconsistent', 'Inconsistent');
insert into OrderRelDict (_code, _value) values
('OverlapLeft', 'First Overhangs Left');
insert into OrderRelDict (_code, _value) values
('OverlapBoth', 'Both Overhang');
insert into OrderRelDict (_code, _value) values
('Before', 'Before');
insert into OrderRelDict (_code, _value) values
('After', 'After');

insert into OrderRelPDict (_code, _value) values
('Overlaps', 'Overlaps');
insert into OrderRelPDict (_code, _value) values
('Gap', 'No Overlap');
insert into OrderRelPDict (_code, _value) values
('Inconsistent', 'Inconsistent');
insert into OrderRelPDict (_code, _value) values
('OverlapLeft', 'First Overhangs Left');
insert into OrderRelPDict (_code, _value) values
('OverlapBoth', 'Both Overhang');
insert into OrderRelPDict (_code, _value) values
('Before', 'Before');
insert into OrderRelPDict (_code, _value) values
('After', 'After');
( 'Abuts', 'Abuts' );
insert into OrderRelPDict (_code, _value) values
( 'ContainedIn', 'Contained In' );
insert into OrderRelPDict (_code, _value) values
( 'Contains', 'Contains' );
insert into OrderRelPDict (_code, _value) values
( 'ContainsLeft', 'Contains Left' );
insert into OrderRelPDict (_code, _value) values
( 'ContainsRight', 'Contains Right' );
insert into OrderRelPDict (_code, _value) values
( 'Identical', 'Identical' );
insert into OrderRelPDict (_code, _value) values
( 'PutativeMatch', 'Putative Match' );

-- Class: OrientationDict -> OrientDict
create row type OrientDict_t
{ _code varchar (10) not null,
  _defn varchar (255),
  _value varchar (255) not null
};
create table OrientDict of type OrientDict_t
( primary key (_code) );
insert into OrientDict (_code, _value) values
( '5''-End', '5''-End' );
insert into OrientDict (_code, _value) values
( '3''-End', '3''-End' );
insert into OrientDict (_code, _value) values
( 'Unknown', 'Unknown' );

-- Class: PloidyDict
create row type PloidyDict_t
{ _code varchar (20) not null,
  _defn varchar (255),
  _value varchar (255) not null
};
create table PloidyDict of type PloidyDict_t
( primary key (_code) );
insert into PloidyDict (_code, _value) values
( 'Haploid', 'Haploid' );
insert into PloidyDict (_code, _value) values
( 'Diploid', 'Diploid' );
insert into PloidyDict (_code, _value) values
( 'Unknown', 'Unknown' );

-- Class: PopulationSpecDict -> PopSpecDict
create row type PopSpecDict_t
{ _code int not null,
  _defn varchar (255),
  _value varchar (255) not null
};
create table PopSpecDict of type PopSpecDict_t
( primary key (_code) );
insert into PopSpecDict (_code, _value) values
( 0, 'Race' );
insert into PopSpecDict (_code, _value) values
( 1, 'Subgeographic region' );
insert into PopSpecDict (_code, _value) values
( 2, 'Directional' );
insert into PopSpecDict (_code, _value) values
( 3, 'Relatedness' );
insert into PopSpecDict (_code, _value) values
( 4, 'Disease' );
insert into PopSpecDict (_code, _value) values
( 5, 'Sex' );
insert into PopSpecDict (_code, _value) values
( 6, 'Study Group' );
insert into PopSpecDict (_code, _value) values
( 7, 'Ethnic isolate' );
insert into PopSpecDict (_code, _value) values
( 8, 'Continental' );
insert into PopSpecDict (_code, _value) values
( 9, 'Country' );
insert into PopSpecDict (_code, _value) values
( 10, 'State' );
insert into PopSpecDict (_code, _value) values
( 11, 'City' );

-- Class: QualitativeLevelDict -> QualLDict
create row type QualLDict_t
{ _code smallint not null,
  _defn varchar (255),
  _value varchar (255) not null
};
create table QualLDict of type QualLDict_t
( primary key (_code) );
insert into QualLDict (_code, _value) values
( 0, 'Medium' );
insert into QualLDict (_code, _value) values
( 1, 'High' );
insert into QualLDict (_code, _value) values
( 2, 'Low' );

-- Class: RNATypeDict -> RNA_TDict
create row type RNA_TDict_t
{
create table RNATDict of type RNA_TDict_t
( _code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null );

create table Rearr_TDict_t
( _code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null );

create table RegEffDict of type RegEffDict_t
( _code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null );

create table RelOrDict of type RelOrDict_t
( _code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null );

create table Repeat_TDict_t
( _code varchar (20) not null,
_defn varchar (255),
_value varchar (255) not null );
{ 'MINISAT', 'MINISAT - minisatellite repeats' );
insert into RepeatTDict (_code, _value) values ('ALPHA', 'ALPHA - alpha satellite repeats');
insert into RepeatTDict (_code, _value) values ('OTHER', 'Other');

-- Class: ResolutionStatusDict -> RStatusDict

create row type RStatusDict_t
{
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
};
create table RStatusDict of type RStatusDict_t
( primary key (_code) );
insert into RStatusDict (_code, _value) values ('Unreviewed', 'Unreviewed');
insert into RStatusDict (_code, _value) values ('Approved', 'Approved');
insert into RStatusDict (_code, _value) values ('Rejected', 'Rejected');
insert into RStatusDict (_code, _value) values ('RejectLeft', 'RejectLeft');
insert into RStatusDict (_code, _value) values ('RejectRight', 'RejectRight');
insert into RStatusDict (_code, _value) values ('InConflict', 'In Conflict');

-- Class: SequencingStatusDict -> SeqStatDict

create row type SeqStatDict_t
{
    _code smallint not null,
    _defn varchar (255),
    _value varchar (255) not null
};
create table SeqStatDict of type SeqStatDict_t
( primary key (_code) );
insert into SeqStatDict (_code, _value) values (0, 'Proposed');
insert into SeqStatDict (_code, _value) values (1, 'Assigned');
insert into SeqStatDict (_code, _value) values (2, 'In Progress');
insert into SeqStatDict (_code, _value) values (3, 'First Pass');
insert into SeqStatDict (_code, _value) values (4, 'Finished');

-- Class: ShapeDict

create row type ShapeDict_t
{
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
};
create table ShapeDict of type ShapeDict_t
( primary key (_code) );
insert into ShapeDict (_code, _value) values ('Rectangle', 'Rectangle');
insert into ShapeDict (_code, _value) values ('RoundStartRect', 'Round start rectangle');
insert into ShapeDict (_code, _value) values ('RoundEndRect', 'Round end rectangle');
insert into ShapeDict (_code, _value) values ('Square', 'Square');
insert into ShapeDict (_code, _value) values ('Triangle', 'Triangle');

-- Class: UnitsDict

create row type UnitsDict_t
{
    _code varchar (5) not null,
    _defn varchar (255),
    _value varchar (255) not null
};
create table UnitsDict of type UnitsDict_t
insert into UnitsDict (_code, _value) values ('HcM', 'Haldane centiMorgans');
insert into UnitsDict (_code, _value) values ('KcM', 'Kosambi centiMorgans');
insert into UnitsDict (_code, _value) values ('cR', 'CentiRays');
insert into UnitsDict (_code, _value) values ('ln', 'Fractional length from pter');
insert into UnitsDict (_code, _value) values ('rf', 'Recombination fraction');
insert into UnitsDict (_code, _value) values ('ov', 'Percent overlap');
insert into UnitsDict (_code, _value) values ('ord', 'Ordinal');
insert into UnitsDict (_code, _value) values ('du', 'Dustin units');
insert into UnitsDict (_code, _value) values ('kb', 'Kilobases');

-- Class: VectorTypeDict -> VectorTDict
create row type VectorTDict-t
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
create table VectorTDict of type VectorTDict_t
    primary key (_code);
insert into VectorTDict (_code, _value) values ('UNKNOWN', 'Unknown');
insert into VectorTDict (_code, _value) values ('YAC', 'YAC');
insert into VectorTDict (_code, _value) values ('BAC', 'BAC');
insert into VectorTDict (_code, _value) values ('PAC', 'PAC');
insert into VectorTDict (_code, _value) values ('Cosmid', 'Cosmid');
insert into VectorTDict (_code, _value) values ('Plasmid', 'Plasmid');
insert into VectorTDict (_code, _value) values ('Phagemid', 'Phagemid');
insert into VectorTDict (_code, _value) values ('M13', 'M13');
insert into VectorTDict (_code, _value) values ('P1', 'P1');
insert into VectorTDict (_code, _value) values ('Phage', 'Phage');

-- Class: VisualMethodDict -> VisMetDict
create row type VisMetDict_t
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
create table VisMetDict of type VisMetDict_t
    primary key (_code);
insert into VisMetDict (_code, _value) values ('Unknown', 'Unknown');
insert into VisMetDict (_code, _value) values ('Radiolabelling', 'Radioactive Labelling');
insert into VisMetDict (_code, _value) values ('DNAStaining', 'DNA Staining');
insert into VisMetDict (_code, _value) values ('ASO', 'Allele Specific Oligonucleotide');
insert into VisMetDict (_code, _value) values ('LSP', 'Locus Specific Probe');
insert into VisMetDict (_code, _value) values ('RSP', 'Repeat Specific Probe');

-- Class: YesNoUnknown_Und dict -> YesNoUnkUDict
create row type YesNoUnkUDict-t
    _code varchar (30) not null,
    _defn varchar (255),
    _value varchar (255) not null
create table YesNoUnkUDict of type YesNoUnkUDict_t
    primary key (_code);
insert into YesNoUnkUDict (_code, _value) values ('Unknown', 'Unknown');
insert into YesNoUnkUDict (_code, _value) values ('No', 'No');
insert into YesNoUnkUDict (_code, _value) values ('Yes', 'Yes');

-- Class: YesNoUnknown_Yes Dict -> YesNoUnkYDict
create row type YesNoUnkYDict-t
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
create table YesNoUnkYDict of type YesNoUnkYDict_t
    primary key (_code);
insert into YesNoUnkYDict (_code, _value) values ('Yes', 'Yes');
insert into YesNoUnkYDict (_code, _value) values ('No', 'No');
insert into YesNoUnkYDict (_code, _value) values ('Yes', 'Yes');

-- Class: YesNoNoDict
create row type YesNoNoDict_t
(
    _code varchar (5) not null,
    _defn varchar (255),
    _value varchar (255) not null
);
create table YesNoNoDict of type YesNoNoDict_t
( primary key (_code) )
insert into YesNoNoDict (_code, _value) values ('No', 'No');
insert into YesNoNoDict (_code, _value) values ('Yes', 'Yes');

-- Class: YesNoYesDict
create row type YesNoYesDict_t
(
    _code varchar (5) not null,
    _defn varchar (255),
    _value varchar (255) not null
);
create table YesNoYesDict of type YesNoYesDict_t
( primary key (_code) )
insert into YesNoYesDict (_code, _value) values ('Yes', 'Yes');
insert into YesNoYesDict (_code, _value) values ('No', 'No');

-- Class: YesUnknownUnkDict
create row type YesUnkUDict_t
(
    _code varchar (20) not null,
    _defn varchar (255),
    _value varchar (255) not null
);
create table YesUnkUDict of type YesUnkUDict_t
( primary key (_code) )
insert into YesUnkUDict (_code, _value) values ('Unknown', 'Unknown');
insert into YesUnkUDict (_code, _value) values ('Yes', 'Yes');

-- Class: ObjectHistory --> OHistory
class ObjectHistory
{
    _changes
    ObjectHistory_changes --> OHistory_chgs
    }
create row type OHistory_chgs_t
(
    _id numeric (10,0) not null,  -- identity
    _oid numeric (10,0) not null,
    attributeName varchar (255) not null,
    changedFrom varchar (255),
    changedTo varchar (255),
    elementID int
);
create table OHistory_chgs of type OHistory_chgs_t
( primary key (_oid) )

create row type OHistory_t
(
    _id numeric (10,0) not null,
    lastModifiedBy numeric (10,0) not null,
    modDate datetime year to day not null,
    object int not null,
    objectClass varchar (30) not null,
    version int not null,
    changes numeric (10,0) not null
);
create table OHistory of type OHistory_t
( primary key (_oid),
  foreign key (lastModifiedBy) references Contact,
  foreign key (changes) references OHistory_chgs
);

-- Class: ObjectName --> OName

-- ObjectName_history --> OName_hist

-- ObjectName_submissions --> OName_submit

create row type OName_hist_t
(
    OName_id numeric (10,0) not null,
    OHist_id numeric (10,0) not null
);
create row type OName_submit_t
(
    OName_id numeric (10,0) not null,
    Submit_id numeric (10,0) not null
);
create row type OName_t
(
    _oid numeric (10,0) not null,
    dBobject numeric (10,0) not null,
    displayName varchar (200) not null,
    history SET (OName_hist_t not null),
CREATE TABLE AObject AS

CREATE TYPE AO_history_t AS

CREATE TYPE AOQsubmit T AS

CREATE TYPE AObject AS

CREATE TABLE AdminAnnotation AS

CREATE TABLE Annotation AS

CREATE TYPE AdminAnnotation anno AS

CREATE TYPE Annotation anno AS

CREATE TYPE AdminAnnotation annoTypes AS

CREATE TYPE AdminAnnotation dBObjects AS

CREATE TABLE AdmAnno AS

CREATE TABLE Anno anno AS

CREATE TABLE Anno dBObjects AS
create table Anno of type Anno_t {
    foreign key (annoType) references Anno_TDict (_code)
} under AObject;

create table ExternalDB of type ExternalDB_t under AObject;

create table ExtLink of type ExtLink-t primary key (_oid),
    foreign key (externalDB) references ExternalDB,
    foreign key (isPutative) references YesNoNoDict (_code),
    foreign key (owner) references Contact
under ExtLink;

create table CitLink of type CitLink-t under ExtLink;

create table GenLink of type GenLink-t under ExtLink;

create table DBO_objects of type DBO_names_t
    foreign key (_oid) references DBO,
    foreign key (isPutative) references YesNoNoDict (_code),
    foreign key (owner) references Contact
under DBO;
CREATE TABLE DBO_names OF TYPE DBO_names_t
( PRIMARY KEY (_oid),
FOREIGN KEY (source_) REFERENCES Contact,
FOREIGN KEY (nameType) REFERENCES NameTypeDict (_code),
FOREIGN KEY (nameStatus) REFERENCES NameStatDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

create row type DBO_hist_t
{
  DObject_id numeric (10,0) not null,
  OHistory_id numeric (10,0) not null
};

create row type DBO_repBy_t
{
  DObject1_id numeric (10,0) not null,
  DObject2_id numeric (10,0) not null
};

create row type DBO_submit_t
{
  DObject_id numeric (10,0) not null,
  Submission_id numeric (10,0) not null
};

CREATE ROW TYPE DObject_t
{
  _oid numeric (10,0) not null,
  accessionID VARCHAR (50) NOT NULL,
  accessionNumber numeric (18,0) NOT NULL, -- hidden
  addDate DATETIME NOT NULL,
  annotations SET (Anno_t NOT NULL),
  approvalStatus VARCHAR (20) NOT NULL,
  citations SET (CtLink_t NOT NULL),
  comment VARCHAR (255),
  displayName VARCHAR (200) NOT NULL,
  externalLinks SET (GmLink_t NOT NULL),
  history SET (DBO_hist_t NOT NULL),
  isReleased SMALLINT NOT NULL,
  lastModifiedBy numeric (10,0) NOT NULL,
  lastModifiedDate DATETIME NOT NULL,
  modifyDate DATETIME NOT NULL,
  name numeric (10,0) NOT NULL,
  ObjectClass VARCHAR (30) NOT NULL,
  owner numeric (10,0) NOT NULL,
  releaseDate DATETIME NOT NULL,
  replacedBy SET (DBO_repBy_t NOT NULL),
  status VARCHAR (20) NOT NULL,
  submissions SET (DBO_submit_t NOT NULL),
  version INT NOT NULL
};

-- NOTE: Normally the DObject would be a derivative (child)
-- of the AccessionedObject, but the Informix Universal
-- Server runs out of memory while creating the DB with
-- this modification. When we get Informix v9.20, make
-- DObject_t a child of AccessObject_t and try it again.
-- -- William Chuang (wchuang@mit.edu)

CREATE TABLE DObject OF TYPE DObject_t

-- LINK TABLE between DObject -> DObject
create table DBO_repBy of type DBO_repBy_t
{
  foreign key (DBObject1_id) references DObject,
  foreign key (DBObject2_id) references DObject
};

-- LINK TABLE between DObject -> ObjectHistory
create table DBO_hist of type DBO_hist_t
{
  foreign key (DBObject_id) references DObject,
  foreign key (OHistory_id) references OHistory
};

-- Class: Submission
CREATE ROW TYPE Submissions_t UNDER DObject_t;
CREATE TABLE Submission OF TYPE Submissions_t UNDER DObject;

-- LINK TABLE between AnnotationObject -> ObjectHistory
create table AO_history of type AO_history_t
{
  foreign key (AObject_id) references AnnotationObject,
  foreign key (OHistory_id) references OHistory
};

-- LINK TABLE between AnnotationObject -> Submission
create table AO_submit of type AO_submit_t
{
  foreign key (AObject_id) references AnnotationObject,
  foreign key (Submission_id) references Submission
};

-- LINK TABLE between AdminAnnotationTypeDict
-- LINK TABLE between ObjectName -> Submission
--
cREATE table OName_submit of type OName-submit_t
(
foreign key (OName_id) references OName,
foreign key (Submit_id) references Submission
);

-- Class: AlleleSet -> AllSet
----
CREATE ROW TYPE AllSet_allfrag_t
(
_aid numeric (10,0) NOT NULL, -- identity
_oid numeric (10,0) not null,
_order INT NOT NULL,
allele numeric (10,0) NOT NULL,
fragment VARCHAR (255) NOT NULL,
isReleased SMALLINT NOT NULL,
owner numeric (10,0) NOT NULL
);
CREATE ROW TYPE AllSet_confrag_t
(
_aid numeric (10,0) NOT NULL, -- identity
_oid numeric (10,0) not null,
constantFragments VARCHAR (255) NOT NULL,
isReleased SMALLINT NOT NULL,
owner numeric (10,0) NOT NULL
);
CREATE TABLE AllSet_confrag OF TYPE AllSet_confrag_t
( PRIMARY KEY (_oid),
FOREIGN KEY (owner) REFERENCES Contact
);
CREATE ROW TYPE AllSet_all_t
(
_oid numeric (10,0) not null,
population VARCHAR (80),
allnumber numeric (10,0) not null,
size VARCHAR (255) NOT NULL,
frequency FLOAT,
obsSet FLOAT,
calcSet FLOAT,
numberOfChrom SMALLINT,
allfreq numeric (10,0) NOT NULL,
owner numeric (10,0) NOT NULL,
isReleased SMALLINT NOT NULL
);
CREATE ROW TYPE AllSet_t
(

-- LINK TABLE between REnz -> ObjectHistory
--
cREATE table REnzhist of type REnz_hist_t
(
RE_id numeric (10,0) not null,
OHist_id numeric (10,0) not null
);
CREATE ROW TYPE REnz_t
(
_oid numeric (10,0) not null,
displayName varchar (200) not null,
history SET (REnz_hist_t not null),
methylationSite varchar (20),
modDate datetime year to day not null,
prototype varchar (20),
recogSeq varchar (20) not null,
searchName varchar (200) not null,
version INT
);
create table REnz of type REnz_t
( primary key (_oid)
);
CREATE ROW TYPE DetMetpro_t
{
  _aid numeric (10,0) not null, -- identity
  _oid numeric (10,0) not null,
  _order INT NOT NULL,
  isReleased SMALLINT NOT NULL,
  owner numeric (10,0) not null,
  probes numeric (10,0) NOT NULL,
}

----- Class: Allele -----
CREATE ROW TYPE Allele_t
{
  alleleSets SET (AllSet-t NOT NULL)
} UNDER DBOBJECT_t;
CREATE TABLE Allele OF TYPE Allele_t
( primary key (_oid)
) UNDER DBOBJECT;
CREATE TABLE AllSet-allfrag OF TYPE AllSet-allfrag-t
( PRIMARY KEY (_oid),
  FOREIGN KEY (allele) REFERENCES Allele,
  FOREIGN KEY (owner) REFERENCES Contact
);

----- Class: AlleleFrequency -> AllFreq
-----
CREATE ROW TYPE AllFreqofreq_t
{
  _aid numeric (10,0) not null, -- identity
  _oid numeric (10,0) not null,
  allele numeric (10,0) NOT NULL,
  frequency FLOAT,
  isReleased SMALLINT NOT NULL,
  owner numeric (10,0) not null,
  stdDev FLOAT
};
CREATE TABLE AllFreqofreq OF TYPE AllFreqofreq_t
( PRIMARY KEY (_oid),
  FOREIGN KEY (allele) REFERENCES Allele,
  FOREIGN KEY (owner) REFERENCES Contact
);
CREATE row type AllFreq_pop_t
{
  AllFreqofreq_id numeric (10,0) not null,
  Population_id numeric (10,0) not null
};

----- Class: VariationType -> VarType
-----
CREATE ROW TYPE VarTypeTC-t
{
  _aid numeric (10,0) not null, -- identity
  _oid numeric (10,0) not null,
  sub numeric (10,0) not null,
  super varchar (200) not null,
  calculatedHet FLOAT,
  frequency SET (AllFreqofreq_t NOT NULL),
  populations SET (AllFreq_pop_t NOT NULL),
  numberOfChrom SMALLINT,
  observedHet FLOAT
} UNDER DBOBJECT_t;

CREATE ROW TYPE AllFreq_t
{
  alleleSet numeric (10,0) NOT NULL,
  calculatedHet FLOAT,
  frequencies SET (AllFreqofreq_t NOT NULL),
  populations SET (AllFreq_pop_t NOT NULL),
  numberofChrom SMALLINT,
  observedHet FLOAT
} UNDER DBOBJECT_t;

CREATE ROW TYPE VarTypeVarType_t
{
  VT1_id numeric (10,0) not null,
  VT2_id numeric (10,0) not null
};
CREATE row type VarTypeVarType_of type VarTypeVarType_t
{
  VT1_id numeric (10,0) not null,
  VT2_id numeric (10,0) not null
};

CREATE TABLE VarType of type VarType-t
primary key (_oid),
foreign key (broaderTerm) references VarType
create table VarTypeVarType of type VarTypeVarType_t
foreign key (VT1_id) references VarType,
foreign key (VT2_id) references VarType
create table VarType_TC of type VarType_TC_t
-- LINK TABLE between VariationType -> ObjectHistory
create table VarTypeHist of type VarTypeHist_t
{ foreign key (VType_id) references VarType,
foreign key (OHist_id) references OHistory
};

-- Class: Variation -> Var
CREATE ROW TYPE Var_typeQ_t
{ typeQuery VARCHAR (200) NOT NULL,
type numeric (10,0) not null
};
CREATE TABLE Var_typeQ OF TYPE Var_typeQ_t
( FOREIGN KEY (type) REFERENCES VarType )
;
create row type Var_ASO_t
{ Var_id numeric (10,0) not null,
ASO_id numeric (10,0) not null
};
create row type Var_detMet_t
{ Var_id numeric (10,0) not null,
DM_id numeric (10,0) not null
};
create row type Var_genSegs_t
{ Var_id numeric (10,0) not null,
GS_id numeric (10,0) not null
};
CREATE ROW TYPE Var_t
{ ASO SET (Var_ASO_t NOT NULL),
DNAType VARCHAR (20) NOT NULL,
detectMethods SET (Var_detMet_t NOT NULL),
genomicSegments SET (Var_genSegs_t NOT NULL),
mutantAminoAcid VARCHAR (3),
mutantNuclSeq VARCHAR (255),
nullASeq SET (nullASeqLink_t NOT NULL),
typeQueries SET (Var_typeQ_t NOT NULL),
variationType numeric (10,0) NOT NULL,
wildTnlminoAcid VARCHAR (3),
wildNuclSeq VARCHAR (255)
) UNDER DBObject_t;

--- Class: AlleleSpecificOligomer -> ASO
---- Class: Expression -> Expr
---- Expression_pattern -> Expr_pat
CREATE ROW TYPE Expr_pat_t
{ _aid numeric (10,0) NOT NULL, -- identity
 oid numeric (10,0) not null,
_order INT NOT NULL,
isReleased SMALLINT NOT NULL,
owner numeric (10,0) not null,
pattern VARCHAR (255) NOT NULL
};
CREATE TABLE Expr_pat OF TYPE Expr_pat_t
( PRIMARY KEY (_aid),
FOREIGN KEY (owner) REFERENCES Contact )
;
CREATE ROW TYPE Expr_t
{ pattern LIST (Expr_pat_t NOT NULL),
gene numeric (10,0) NOT NULL
) UNDER DBObject_t;

--- Class: GeneFamily -> GenFam
CREATE ROW TYPE GenFam_genes_t  
(  
  GF_id           numeric (10,0) not null,  
  Gene_id         numeric (10,0) not null  
);  

CREATE ROW TYPE GenFam_subF_t  
(  
  GF_parent_id    numeric (10,0) not null,  
  GF_child_id     numeric (10,0) not null  
);  

CREATE ROW TYPE GenFam_t  
(  
  definition      LIST (VARCHAR (255) NOT NULL),  
  genes           SET (GenFam_genes_t NOT NULL),  
  subFamilies     SET (GenFam_subF_t NOT NULL),  
  superFamilies   SET (GenFam_subF_t NOT NULL)  
) UNDER DBObject_t;  

CREATE TABLE GenFam OF TYPE GenFam_t  
UNDER DBObject;  

----  
Class: GeneProduct -> GenPro  
----  
CREATE ROW TYPE GenPro_genes_t  
(  
  _aid            numeric (10,0) NOT NULL,  
  _oid            numeric (10,0) not null,  
  gene            numeric (10,0) NOT NULL,  
  isReleased      SMALLINT NOT NULL,  
  nCopies         SMALLINT NOT NULL,  
  owner           numeric (10,0) not null,  
  subunit         VARCHAR (20) NOT NULL  
);  

CREATE ROW TYPE GenPro_t  
(  
  nuclASeq         SET (NuclASeqLink_t NOT NULL),  
  structures       SET (StructureLink_t NOT NULL),  
  genes            SET (GenPro_genes_t NOT NULL)  
) UNDER DBObject_t;  

---  
Class: Protein  
---  
CREATE ROW TYPE Protein_proSeq_t  
(  
  gene            numeric (10,0) NOT NULL,  
  sequence        numeric (10,0) NOT NULL  
);  

CREATE ROW TYPE Protein_t  
(  
  proteinSequences SET (Protein_proSeq_t NOT NULL),  
  enzymes         SET (EnzymeLink_t NOT NULL)  
) UNDER GenPro_t;  

----  
Class: RNA  
----  
CREATE ROW TYPE RNA_t  
(  
  RNAtype          VARCHAR (20) NOT NULL  
) UNDER GenPro_t;  

----  
Class: TaxonomicLink  
----  
create row type TaxonomicLink_t  
(  
  organism        numeric (10,0) not null  
) under ExtLink_t;  

----  
Class: Organism  
----  
CREATE ROW TYPE Organism_t  
(  
  taxonomyLinks   SET (TaxonomicLink_t NOT NULL),  
  url             VARCHAR (255)  
) UNDER DBObject_t;  

CREATE TABLE Organism OF TYPE Organism_t  
UNDER DBObject;  

create table TaxonomicLink of type TaxonomicLink_t  
(  
  foreign key (organism) references Organism  
) under ExtLink;  

----  
Class: Clone  
----  
CREATE ROW TYPE Clone_libAddr_t  
(  
  _aid            numeric (10,0) NOT NULL,  
  _oid            numeric (10,0) not null,  
  columnLoc       VARCHAR (20),  
  isReleased      SMALLINT NOT NULL,  
  library         numeric (10,0) NOT NULL,  
  locationtype    VARCHAR (20) NOT NULL,  
  owner           numeric (10,0) not null,  
  plateLoc        VARCHAR (20),  
  rowLoc          VARCHAR (20)  
);  

CREATE ROW TYPE Clone_reFrag_t  
(  
  _aid            numeric (10,0) NOT NULL,  
  _oid            numeric (10,0) not null,  
  enzyme          numeric (10,0) NOT NULL,  
  isReleased      SMALLINT NOT NULL,  
  owner           numeric (10,0) not null,  
  size            FLOAT NOT NULL  
);  

CREATE TABLE Clone_reFrag OF TYPE Clone_reFrag_t  
(  
  PRIMARY KEY (_oid),  
  FOREIGN KEY (enzyme) REFERENCES REns,  
  FOREIGN KEY (owner) REFERENCES Contact  
);
create row type Clone_gExcEnz_t {
  Clone_id numeric (10,0) not null,
  RE_id numeric (10,0) not null
};

create row type Clone_vExcEnz_t {
  Clone_id numeric (10,0) not null,
  RE_id numeric (10,0) not null
};

create row type Clone_vInsEnz_t {
  Clone_id numeric (10,0) not null,
  RE_id numeric (10,0) not null
};

----
Class: GenomicSegment -> GenSeg
----
CREATE ROW TYPE GenSeg_clones_t {
  _oid numeric (10,0) not null,
  clone_c numeric (10,0) NOT NULL,
  DNAType_c VARCHAR (20) NOT NULL,
  insertSize_c FLOAT,
  vectorType_c VARCHAR (20) NOT NULL,
  position_c VARCHAR (20),
  relType_c VARCHAR (20) NOT NULL,
  rel_c numeric (10,0) NOT NULL,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_dData_t {
  _oid numeric (10,0) not null,
  segment_1 numeric (10,0) NOT NULL,
  endPoint_1 VARCHAR (20) NOT NULL,
  segment_2 numeric (10,0) NOT NULL,
  endPoint_2 VARCHAR (20) NOT NULL,
  distance FLOAT NOT NULL,
  units VARCHAR (5) NOT NULL,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_allLoc_t {
  _aid numeric (10,0) NOT NULL,  -- identity
  _oid numeric (10,0) not null,
  LFM VARCHAR (200) NOT NULL,
  RFM VARCHAR (200) NOT NULL,
  chromosome_3 numeric (10,0) NOT NULL,
  isReleased SMALLINT NOT NULL,
  owner numeric (10,0) not null
};

CREATE ROW TYPE GenSeg_mSC_t {
  _aid numeric (10,0) NOT NULL,  -- identity
  _oid numeric (10,0) not null,
  chromosome numeric (18,0),
  isReleased SMALLINT NOT NULL,
  maxMaySearch FLOAT,
  minMaySearch FLOAT,
  owner numeric (10,0) not null
};

CREATE TABLE GenSeg_mSC OF TYPE GenSeg_mSC_t 
( PRIMARY KEY (_aid),
  FOREIGN KEY (owner) REFERENCES Contact )
;

CREATE ROW TYPE GenSeg_var_t {
  _oid numeric (10,0) not null,
  polymorphism numeric (10,0) NOT NULL,
  vType numeric (10,0) NOT NULL,
  position VARCHAR (255),
  maxHet FLOAT,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_varQ_t {
  _oid numeric (10,0) not null,
  polymorphism numeric (10,0) NOT NULL,
  vTypeQuery VARCHAR (200) NOT NULL,
  maxHet FLOAT,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_AMP_t {
  _oid numeric (10,0) not null,
  amplimer numeric (10,0) NOT NULL,
  DNAType_a VARCHAR (20) NOT NULL,
  ampLength FLOAT,
  sourceEST_a VARCHAR (255),
  position_a VARCHAR (255),
  relType_a VARCHAR (20) NOT NULL,
  rel_a numeric (10,0) NOT NULL,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_cytoL_t {
  _oid numeric (10,0) not null,
  chromosome numeric (10,0) NOT NULL,
  LFM numeric (10,0) NOT NULL,
  RFM numeric (10,0) NOT NULL,
  cytoband VARCHAR (255) NOT NULL,
  mapElement_ VARCHAR (255),
  cytoMap numeric (10,0) NOT NULL,
  approvalStatus_ VARCHAR (20) NOT NULL,
  estMB FLOAT,
  nMrange FLOAT,
  owner numeric (10,0) not null,
  isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_otherL_t
(...
CREATE ROW TYPE GenSeg-relL-t
{
 oid numeric (10,0) not null,
 chromosome numeric (10,0) NOT NULL,
 mapElement numeric (10,0) NOT NULL,
 coordinate FLOAT,
 units VARCHAR (5) NOT NULL,
 estMB FLOAT,
 MBRange numeric (10,0) NOT NULL,
 owner numeric (10,0) not null,
 isReleased SMALLINT NOT NULL
}

CREATE ROW TYPE GenSeg__mC_t
{
 _aid numeric (10,0) NOT NULL,
 oid numeric (10,0) not null,
 chromosome numeric (18,0),
 isReleased SMALLINT NOT NULL,
 maxMay FLOAT,
 minMay FLOAT,
 owner numeric (10,0) not null
}

CREATE TABLE GenSegmC OF TYPE GenSegmC_t
( PRIMARY KEY (_oid),
 FOREIGN KEY (owner) REFERENCES Contact )

CREATE ROW TYPE GenSeg_genes_t
{
 _oid numeric (10,0) NOT NULL, -- identity
 _aid numeric (10,0) not null,
 chromosome numeric (18,0),
 isReleased SMALLINT NOT NULL,
 position VARCHAR (255),
 reltype VARCHAR (20) NOT NULL,
 owner numeric (10,0) not null,
 isReleased SMALLINT NOT NULL
}

CREATE ROW TYPE GenSeg-oData_t
{
 _oid numeric (10,0) not null,
 relationship VARCHAR (20) NOT NULL,
 marker numeric (10,0) NOT NULL,
 observationType VARCHAR (20) NOT NULL,
 position VARCHAR (255) NOT NULL,
 owner numeric (10,0) not null,
 isReleased SMALLINT NOT NULL
}

CREATE ROW TYPE GenSeg-oMark_t
{
 _oid numeric (10,0) not null,
 marker VARCHAR (255),
 position VARCHAR (255),
 reltype VARCHAR (20) NOT NULL,
 owner numeric (10,0) not null,
 isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_ESTs_t
{
 _oid numeric (10,0) not null,
 ESTe numeric (10,0) NOT NULL,
 end VARCHAR (10) NOT NULL,
 clone numeric (10,0) NOT NULL,
 position VARCHAR (255),
 rel numeric (10,0) NOT NULL,
 owner numeric (10,0) not null,
 isReleased SMALLINT NOT NULL
};

CREATE ROW TYPE GenSeg_seqStat_t
{
 completionDate DATETIME YEAR TO DAY NOT NULL,
 seqStat numeric (10,0) NOT NULL,
 sequencingStatus SMALLINT NOT NULL
};

create row type HomologyLink_t
( segment numeric (10,0) not null ) under ExtLink_t;

create row type PhenotypeLink_t
( segment numeric (10,0) not null ) under ExtLink_t;

-- Class: LocEvidenceType -> LocEvidT
--

create row type LocEvidT_TC_t
{
 _aid numeric (10,0) not null, -- identity
 _oid numeric (10,0) not null,
 sub numeric (10,0) not null,
 super varchar (20) not null
};

create row type LocEvidT_hist_t
{
 LET_id numeric (10,0) not null,
 OHist_id numeric (10,0) not null
};

create row type LocEvidT_narrowT_t
{
 narrowT_id numeric (10,0) not null,
 LocEvidT_id numeric (10,0) not null
}
create row type LocEvidT_t
{
    _oid    numeric (10,0) not null,
    TC      SRT (LocEvidT_TC_t not null),
    broaderTerm numeric (10,0) not null,
    displayName varchar (200) not null,
    history SRT (LocEvidT_hist_t not null),
    modDate  datetime year to day not null,
    narrowerTerms SRT (LocEvidT_narrowT_t not null),
    searchName varchar (200) not null,
    version  int not null
};
create table LocEvidT of type LocEvidT_t
( primary key (_oid),
  foreign key (broaderTerm) references LocEvidT );
create table LocEvidT_TC of type LocEvidT_TC_t
( primary key (_oid),
  foreign key (sub) references LocEvidT );
create table LocEvidT_hist of type LocEvidT_hist_t
( foreign key (LET-id) references LocEvidT,
  foreign key (OHist_id) references OHistory );
create table LocEvidT_narrowT of type LocEvidT_narrowT_t
( foreign key (narrowT-id) references LocEvidT,
  foreign key (LocEvidTid) references LocEvidT )

----- Class: SequencingStatus -> SeqStatus
-----
create row type SeqStatus_t
{
    completionDate  datetime year to day,
    segment          numeric (10,0) not null,
    sequencingStatus smallint not null
};
create row type genseg_map_t
{
    genseg_id     numeric (10,0) not null,
    map_id        numeric (10,0) not null
};

----- Class: Amplimer
-----
CREATE ROW TYPE Amp_primers_t
{
    _aid         numeric (10,0) NOT NULL, -- identity
    _oid         numeric (10,0) not null,
    isReleased   SMALLINT NOT NULL,
    owner        numeric (10,0) not null,
    primerName   VARCHAR (200) NOT NULL,
    primerSequence VARCHAR (200) NOT NULL
};
CREATE TABLE Amp_primers OF TYPE Amp_primers_t
( PRIMARY KEY (_oid),
  FOREIGN KEY (owner) REFERENCES Contact )

----- Class: PCRCondition
-----
CREATE ROW TYPE PCRCond_prot_t
{
    _aid         numeric (10,0) NOT NULL, -- identity
    _oid         numeric (10,0) not null,
    _order       INT NOT NULL,
    isReleased   SMALLINT NOT NULL,
    owner        numeric (10,0) not null,
    sequence     VARCHAR (255) NOT NULL
};
CREATE TABLE Amp_seq OF TYPE Amp_seq_t
( PRIMARY KEY (_oid),
  FOREIGN KEY (owner) REFERENCES Contact )

----- Class: PCRCondition
-----
CREATE ROW TYPE PCRCond_prot_t
{
    _aid         numeric (10,0) NOT NULL, -- identity
    _oid         numeric (10,0) not null,
    _order       INT NOT NULL,
    isReleased   SMALLINT NOT NULL,
CREATE ROW TYPE Dist-seg_t
{
    _aid numeric (10,0) NOT NULL, -- identity
    _old numeric (10,0) not null,
    _order INT NOT NULL,
    endPoint VARCHAR (20) NOT NULL,
    segment numeric (10,0) NOT NULL
};

CREATE TABLE Dist-seg OF TYPE Dist-seg_t
(PRIMARY KEY (_old),
FOREIGN KEY (endPoint) REFERENCES EndpointDict (_code),
FOREIGN KEY (segment) REFERENCES GenSeg);

CREATE ROW TYPE Dist_t
{
    distance FLOAT NOT NULL,
    error FLOAT,
    errorType VARCHAR (20),
    likelihood FLOAT,
    likelihoodType VARCHAR (20),
    segments LIST (Dist-seg_t NOT NULL),
    units VARCHAR (5) NOT NULL
} UNDER Observation_t;

CREATE TABLE Dist OF TYPE Dist_t
(FOREIGN KEY (errorType) REFERENCES ErrorTDict (_code),
FOREIGN KEY (likelihoodType) REFERENCES OrderL-TDict (_code),
FOREIGN KEY (units) REFERENCES UnitsDict (_code)
) UNDER Observation;

------ Class: Observation ------

CREATE ROW TYPE Order-segments_t
{
    _aid numeric (10,0) NOT NULL, -- identity
    _old numeric (10,0) not null,
    _order INT NOT NULL,
    segments numeric (10,0) NOT NULL
};

CREATE TABLE Order-segments OF TYPE Order-segments_t
(PRIMARY KEY (_old),
FOREIGN KEY (segments) REFERENCES GenSeg);

CREATE ROW TYPE Order_t
{
    likelihood FLOAT,
    likelihoodType VARCHAR (20),
    observationType VARCHAR (20) NOT NULL,
    overlap FLOAT,
    position VARCHAR (255),
    relationship VARCHAR (20) NOT NULL,
    segments LIST (Order-segments_t NOT NULL),
    units VARCHAR (5)
} UNDER Observation_t;

CREATE TABLE Order_OF TYPE Order_t
(FOREIGN KEY (likelihoodType) REFERENCES OrderLTDict (_code),
FOREIGN KEY (observationType) REFERENCES Obs_TDict (_code),
FOREIGN KEY (relationship) REFERENCES OrderRelDict (_code),
FOREIGN KEY (units) REFERENCES UnitsDict (_code))
) UNDER Observation;

CREATE TABLE GenSeg_amp OF TYPE GenSeg-amp_t
( PRIMARY KEY (_oid),
FOREIGN KEY (DNAType-a) REFERENCES DNA_TDict (_code),
FOREIGN KEY (amplimer) REFERENCES Amplimer,
FOREIGN KEY (rel-a) REFERENCES Order,
FOREIGN KEY (reltypea) REFERENCES OrderRelDict (_code),
FOREIGN KEY (sourceESTa) REFERENCES EST--
)
);-- Class: Map
-- create row type Map_alignC-t
{
    _aid numeric (10,0) not null, -- identity
    _oid numeric (10,0) not null,
    a_ float,
    b_ float,
    maxCoord float not null,
    minCoord float not null,
    isReleased smallint not null,
    owner numeric (10,0) not null
};

CREATE TABLE Map-alignC of type Map-alignC-t
| primary key _oid,
foreignKey (owner) references Contact |
);

create row type Map_inclM-t
{
    _aid numeric (10,0) not null, -- identity
    _oid numeric (10,0) not null,
    includedFrom varchar (20) not null,
    map Orientation varChar (20) not null,
    owner numeric (10,0) not null
};

create table Map_inclM of type Map-inclM_t
| primary key _oid,
foreignKey (owner) references Contact |
);

create row type Map_tiers-t
{
    tier int not null,
    title varchar (50)
};

create table Map_tiers of type Map-tiers-t;

create row type Map-copiedFrom-t
{
    copiedFromId numeric (10,0) not null,
    copiedFrom varchar (20) not null
};

create table Map-copiedFrom of type Map-copiedFrom_t
| primary key _oid,
foreignKey (owner) references Contact |
);

create row type Map-t
{
    a float,
b
    alignCoeff SET (Map_alignmentC_t not null),
    chromosome numeric (10,0) not null,
    copiedFrom SET (Map_copiedFrom_t not null),
    includesMap SET (Map_inclM_t not null),
    likelihoodType varchar (20),
    mapOf numeric (10,0) not null,
    maxCoord float not null,
    maxUcoord float,
    minCoord float not null,
    minUcoord float,
    orderLikelihood float,
    publiclyEditable varchar (5) not null,
    tiers numeric (10,0) not null,
    units SET (Map_tiers_t not null),--
    } UNDER GenSeg_t;

---- Class: Chromosome

CREATE ROW TYPE Chromosome-t
{
    cellCompartment VARCHAR (20) NOT NULL,
    maxCoord FLOAT,
    minCoord FLOAT,
    maps SET (Map_t NOT NULL)
} UNDER GenSeg_t;

CREATE TABLE Chromosome OF TYPE Chromosome_t
| primary key _id,
foreignKey (cellCompartment) references CmprtDict (_code) |
);

CREATE ROW TYPE Style-t
{
    colorBlue char,
    colorGreen char,
    colorRed char,
    fontStyle VARCHAR (20),
    label VARCHAR (40),
    lineStyle VARCHAR (20),
    shape VARCHAR (20),
    useRGBForText VARCHAR (5) NOT NULL
} UNDER DBObject_t;

CREATE TABLE Style OF TYPE Style-t
| primary key _id,
foreignKey (fontStyle) references FontSDict (_code),
foreignKey (lineStyle) references LineSDict (_code),
foreignKey (shape) references ShapeDict (_code),
foreignKey (useRGBForText) references YesNoYesDict (_code) |
);

create table Map of type Map-t
| primary key _id,
foreignKey (chromosome) references Chromosome,
foreignKey (likelihoodType) references OrderLTDict (_code),
foreignKey (relationship) references OrderRelDict (_code),
foreignKey (units) references UnitsDict (_code))
) UNDER DBObject;
create table MapElement of type MapElement_t
( primary key (_oid),
foreign key (LFM) references MapElement,
foreign key (RFM) references MapElement,
foreign key (ambiguous) references YesNoNoDict (_code),
foreign key (arbitraryCoord) references YesNoNoDict (_code),
foreign key (draw) references YesNoNoDict (_code),
foreign key (map) references Map,
foreign key (point) references EndpointDict (_code),
foreign key (segment) references GenSeg,
foreign key (style) references Style
) under DBObject;

-- LINK TABLE between MapElement -> LocEvidenceType

create table MapElemLocalE of type MapElemLocalE_t
( foreign key (MElemId) references MapElement,
foreign key (LocEvId) references LocEvidence
);

CREATE TABLE GenSeg_cytol OF TYPE GenSeg_cytol_t
( PRIMARY KEY (_oid),
FOREIGN KEY (LFM) REFERENCES GenSeg,
FOREIGN KEY (RFM) REFERENCES GenSeg,
FOREIGN KEY (approvalStatus_) REFERENCES ApprStatDict (_code),
FOREIGN KEY (chromosome_) REFERENCES Chromosome,
FOREIGN KEY (cytoMap) REFERENCES Map,
FOREIGN KEY (mapElement_) REFERENCES MapElement,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_otherL OF TYPE GenSeg_otherL_t
( PRIMARY KEY (_oid),
FOREIGN KEY (chromosome_2) REFERENCES Chromosome,
FOREIGN KEY (mapElement_2) REFERENCES MapElement,
FOREIGN KEY (map_2) REFERENCES Map,
FOREIGN KEY (units_2) REFERENCES UnitsDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_ESTs OF TYPE GenSeg_ESTs_t
( PRIMARY KEY (_oid),
FOREIGN KEY (EST_e) REFERENCES EST,
FOREIGN KEY (clone_e) REFERENCES Clone,
FOREIGN KEY (end_e) REFERENCES OrientDict (_code),
FOREIGN KEY (rel_e) REFERENCES Order_,
FOREIGN KEY (reltype_e) REFERENCES OrderRelDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

--- Class: Library

create row type Lib_gExcEnz_t
( Library_id numeric (10,0) not null,
RS_id numeric (10,0) not null
);

create row type Lib_vExcEnz_t
( Library_id numeric (10,0) not null,
CREATE ROW TYPE Lib_vInsEnz_t
(
Library_id numeric(10,0) not null,
RE_id numeric(10,0) not null
);

CREATE ROW TYPE Library_t
(
DNAType VARCHAR (20) NOT NULL,
genExcEnzymes SET (Lib_gExcEnz_t NOT NULL),
hostOrganism numeric(10,0) not null,
source numeric(10,0) not null,
vecExcEnzymes SET (Lib_vExcEnz_t NOT NULL),
vecInsEnzymes SET (Lib_vInsEnz_t NOT NULL),
vectorName VARCHAR (255),
vectorType VARCHAR (20) NOT NULL
) UNDER GenSeg_t;

create table ReagentSource of type ReagentSource_t
(
foreign key (reagent) references GenSeg,
foreign key (availability) references AvailDict (_code),
foreign key (supplier) references Contact
) under AObject;

CREATE TABLE GenSeg_clones OF TYPE GenSeg_clones_t
( PRIMARY KEY (_oid),
FOREIGN KEY (DNAType_c) REFERENCES DNA_TDict (_code),
FOREIGN KEY (clone_c) REFERENCES Clone,
FOREIGN KEY (rel_c) REFERENCES Order_c,
FOREIGN KEY (reltype_c) REFERENCES OrderRelDict (_code),
FOREIGN KEY (vectorType_c) REFERENCES Vector_TDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

create table SeqStatus of type SeqStatus_t
( foreign key (segment) references GenSeg,
foreign key (sequencingStatus) references SeqStatDict (_code)
) under AObject;

CREATE TABLE Library OF TYPE Library_t
( FOREIGN KEY (DNAType) REFERENCES DNA_TDict (_code),
FOREIGN KEY (hostOrganism) REFERENCES Organism,
FOREIGN KEY (source) REFERENCES GenSeg,
FOREIGN KEY (vectorType) REFERENCES Vector_TDict (_code)
) UNDER GenSeg;

CREATE TABLE Clone libAddr OF TYPE Clone_libAddr_t
( PRIMARY KEY (_oid),
FOREIGN KEY (library) REFERENCES Library,
FOREIGN KEY (locationType) REFERENCES LibLocTDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE DetMet_pro OF TYPE DetMet_proSt
( PRIMARY KEY (_oid),
FOREIGN KEY (probes) REFERENCES GenSeg,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE DetMet_t
(
enzymes LIST (DetMet_enz_t NOT NULL),
gelPercent FLOAT,
probes LIST (DetMet_pro_t NOT NULL),
sepMethod VARCHAR (20) NOT NULL,
variations SET (Var_t NOT NULL),
visualMethod VARCHAR (20) NOT NULL
) UNDER DBOBJECT_t;

CREATE TABLE DetMet OF TYPE DetMet_t
( FOREIGN KEY (sepMethod) REFERENCES SepMDict (_code),
FOREIGN KEY (visualMethod) REFERENCES VisMetDict (_code)
) UNDER DBOBJECT;

-- -- Class: HomologyLink
--
create table HomologyLink of type HomologyLink_t
( foreign key (segment) references GenSeg
) under ExtLink;

-- -- Class: PhenotypeLink
--
create table PhenotypeLink of type PhenotypeLink_t
( foreign key (segment) references GenSeg
) under ExtLink;

---- -- Class: Bin
----
CREATE TABLE Bin OF TYPE Bin_t
( primary key (_oid)
) UNDER GenSeg;
--- Class: CellLine -> CL

CREATE ROW TYPE CL_cytoR_t
  (_aid numeric (10,0) NOT NULL, -- identity
   _oid numeric (10,0) NOT NULL,
   from_ int8 NOT NULL,
   to_ int8 NOT NULL,
   isReleased SMALLINT,
   owner numeric (10,0) NOT NULL);

CREATE TABLE CL_cytoR OF TYPE CL_cytoR_t
  ( PRIMARY KEY (_oid),
    FOREIGN KEY (owner) REFERENCES Contact )
);

create row type CL_rearr_t
  ( CellLine_id numeric (10,0) NOT NULL,
    Rearrange_id numeric (10,0) NOT NULL );

CREATE ROW TYPE CL_t
  ( cellLineType VARCHAR (20) NOT NULL,
    growthStage VARCHAR (20) NOT NULL,
    karyotype VARCHAR (255),
    numberOfPassages int,
    pathology VARCHAR (255),
    cytoid Regions SET (CL_cytoR_t NOT NULL),
    libraries SET (Library_t NOT NULL),
    rearrangements SET (CL_rearr_t NOT NULL)
  ) UNDER GenSeg;

CREATE TABLE CL OF TYPE CL_t
  ( primary key (_oid),
    FOREIGN KEY (cellLineType) REFERENCES CellLine_TDict (_code),
    FOREIGN KEY (growthStage) REFERENCES GrowthSDict (_code)
  ) UNDER GenSeg;

--- Class: Breakpoint

CREATE ROW TYPE Breakpoint_t
  ( rearrangements SET (Rearr_t NOT NULL)
  ) UNDER GenSeg;

CREATE TABLE Breakpoint OF TYPE Breakpoint_t
  UNDER GenSeg;

--- LINK TABLE between Rearr -> Breakpoint

create table Rearr_breakps of type Rearr_breakps_t
  ( foreign key (Rearr_id) references Rearr,
    foreign key (Break_id) references Breakpoint )
);

--- LINK TABLE between CellLine -> Rearrangements

create table CL_rearr of type CL_rearr_t
  ( foreign key (CellLine_id) references CL,
    foreign key (Rearrange_id) references Rearr )
);

--- Class: CytogeneticMarker -> CytoGM

CREATE ROW TYPE CytoGM_res_t
  ( CMarker_id numeric (10,0) NOT NULL,
    Dict_id varchar (10) NOT NULL )
);

CREATE ROW TYPE cytogm_cytogm_t
  ( cytoid numeric (10,0) NOT NULL,
    cytoid2 numeric (10,0) NOT NULL )
);

CREATE ROW TYPE CytoGM-t
  ( bandType VARCHAR (20) NOT NULL,
    childBands SET (CytoGM_cytogm_t NOT NULL),
    celllines SET (CL_t NOT NULL),
    chromosome numeric (10,0) NOT NULL,
    parent numeric (10,0) NOT NULL,
    resolutions SET (CytoGM_res_t NOT NULL)
  ) UNDER GenSeg_t;

CREATE TABLE CytoGM OF TYPE CytoGM-t
  ( primary key (_oid),
    FOREIGN KEY (bandType) REFERENCES CytogMDict (_code),
    FOREIGN KEY (chromosome) REFERENCES Chromosome,
    FOREIGN KEY (parent) REFERENCES CytoGM )
CREATE TABLE CytoGMCytoGM of type CytoGMCytoGM_t
(
    foreign key (cytol_id) references CytoGM,
    foreign key (cyto2_id) references CytoGM
);

CREATE TABLE CytoGM_res of type CytoGM_res_t
(
    foreign key (CMarker_id) references CytoGM,
    foreign key (Dict_id) references CytoGRDict
);

CREATE ROW TYPE ChromR_cytoR_t
(
    _aid numeric (10,0) NOT NULL,  -- identity
    _oid numeric (10,0) not null,  
    from_ numeric (10,0) NOT NULL,
    isReleased SMALLINT NOT NULL,
    owner numeric (10,0) not null,
    to_ numeric (10,0) NOT NULL
);

CREATE TABLE ChromR_cytoR of TYPE ChromR_cytoR_t
(  
    PRIMARY KEY (_oid),
    FOREIGN KEY (from_) REFERENCES CytoGM,
    FOREIGN KEY (to_) REFERENCES CytoGM,
    FOREIGN KEY (owner) REFERENCES Contact
);

CREATE ROW TYPE ChromR of TYPE ChromR_t
(
    isolationMethod VARCHAR (20) NOT NULL,
    cytoRegions SET (ChromR_cytoR_t NOT NULL)
)
UNDER GenSeg_t;

CREATE TABLE Chrom OF TYPE Chrom_t
(  
    FOREIGN KEY (isolationMethod) REFERENCES IMethodDict (_code)
)
UNDER GenSeg;

CREATE ROW TYPE DSegment of TYPE DSegment_t
(
    agent VARCHAR (20),
    frequency VARCHAR (20) NOT NULL
)
UNDER GenSeg_t;

CREATE TABLE DSegment of TYPE DSegment_t UNDER GenSeg;

CREATE ROW TYPE RegR_transF_t
(
    _aid numeric (10,0) NOT NULL,  -- identity
    _oid numeric (10,0) not null,
    effect VARCHAR (20),
    isReleased SMALLINT NOT NULL,
    owner numeric (10,0) not null,
    protein numeric (10,0) not null,
    regulates numeric (10,0) not null
);

CREATE TABLE GenPro OF TYPE GenPro_t
UNDER DBOBJECT;

CREATE TABLE RNA OF TYPE RNA_t
(  
    FOREIGN KEY (RNAtype) REFERENCES RNATDict (_code)
)
UNDER GenPro;

create table StructureLink of type StructureLink_t
(  
    foreign key (geneProduct) references GenPro
)
under ExtLink;

CREATE TABLE Protein OF TYPE Protein_t
create table EnzymeLink of type EnzymeLink_t
{
  foreign key (protein) references Protein
} under ExtLink;

CREATE ROW TYPE RegR_t
{
  transFactors SET (RegR_transF_t NOT NULL)
} UNDER GenSeg_t;

CREATE TABLE RegR OF TYPE RegR_t
UNDER GenSeg;

----
Class: Gene
----
CREATE ROW TYPE GeneElement-t
{
  elementNumber VARCHAR (5),
  elementType VARCHAR (20) NOT NULL,
  gene numeric (10,0) NOT NULL
} UNDER GenSeg_t;

-- NOTE: Commented out some of the columns in order to get
-- around a 32767 byte row-size limitation in Informix 9.14
-- -- William Chuang (wchuang@mit.edu)

CREATE table Gene of type Gene_t
{
  asos SET (ASO_t NOT NULL),
  sub
  numeric (10,0) not null,
  super numeric (10,0) not null,
  evidence SET (GeneEvid_t NOT NULL),
  evidenceQuery SET (Expr_t NOT NULL),
  isPseudogene numeric (10,0) NOT NULL,
  isPutative VARCHAR (5) NULL,
  mutations SET (Mutation_t NOT NULL),
  products SET (GenePro_t NOT NULL),
  protSeqLink SET (ProtSeqLink_t NOT NULL)
} UNDER GenSeg_t;

-- NOTE: Commented out some of the columns in order to get
-- around a 32767 byte row-size limitation in Informix 9.14
-- -- William Chuang (wchuang@mit.edu)

CREATE table GenSeggenes of type GenSeggenes-t
{ PRIMARY KEY (_oid),
  geneg
  references Gene,
  rel-g
  references Order,
  reltype-g
  references OrderRelDict (_code),
  protSeqLink
  references ProtSeqLink
} UNDER GenSeg;

create table GeneEvidenceType -> GeneEvidT

create row type GeneEvidT_TC_t
{
  _aid numeric (10,0) not null, -- identity
  _old numeric (10,0) not null,
  sub
  numeric (10,0) not null,
  super varchar (255) not null,
}

create row type GeneEvidT_hist_t
{
  GET1_id numeric (10,0) not null,
  GET2_id numeric (10,0) not null,
}

create row type GeneEvidT_GET-t
{
  GET1_id numeric (10,0) not null,
  GET2_id numeric (10,0) not null
}

create table GeneEvidT of type GeneEvidT_t
{
  _oid numeric (10,0) not null,
  TC
  SET (GeneEvidT_TC_t not null),
  broaderTerm numeric (10,0) not null,
  displayName varchar (200) not null,
  history SET (GeneEvidT_hist_t not null),
  modDate datetime year to day not null,
  narrowerTerms SET (GeneEvidT_GET_t not null),
  searchName varchar (200) not null,
  version int not null
}

create table GeneEvidT_TC of type GeneEvidT_TC_t
{ primary key (_oid),
  foreign key (broaderTerm) references GeneEvidT
}

create table GeneEvidT_GET of type GeneEvidT_GET_t
{ primary key (_oid),
"FOREIGN KEY (owner) REFERENCES Contact"
}

create table ProtSeqLink of type ProtSeqLink_t
{
  foreign key (gene) references Gene
} under ExtLink;

CREATE TABLE Protein_proSeq OF TYPE Protein_proSeqct
FOREIGN KEY (gene-) REFERENCES Gene,
FOREIGN KEY (sequence) REFERENCES ProtSeqLink

CREATE TABLE RegR_transF OF TYPE RegR_transF-t
{ PRIMARY KEY (_oid),
  FOREIGN KEY (effect) REFERENCES RegEffDict (_code),
  FOREIGN KEY (protein) REFERENCES Protein,
  FOREIGN KEY (regulates) REFERENCES Gene,
  FOREIGN KEY (owner) REFERENCES Contact
};

-- -- Class: GeneEvidenceType -> GeneEvidT
--
create row type GeneEvidT_TC_t
{
  _aid numeric (10,0) not null, -- identity
  _old numeric (10,0) not null,
  sub
  numeric (10,0) not null,
  super varchar (255) not null,
}

create row type GeneEvidT_hist_t
{
  GET1_id numeric (10,0) not null,
  GET2_id numeric (10,0) not null,
}

create row type GeneEvidT_GET-t
{
  GET1_id numeric (10,0) not null,
  GET2_id numeric (10,0) not null
}

create table GeneEvidT of type GeneEvidT_t
{
  _oid numeric (10,0) not null,
  TC
  SET (GeneEvidT_TC_t not null),
  broaderTerm numeric (10,0) not null,
  displayName varchar (200) not null,
  history SET (GeneEvidT_hist_t not null),
  modDate datetime year to day not null,
  narrowerTerms SET (GeneEvidT_GET_t not null),
  searchName varchar (200) not null,
  version int not null
}

create table GeneEvidT_TC of type GeneEvidT_TC_t
{ primary key (_oid),
"FOREIGN KEY (owner) REFERENCES Contact"
};
create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT GET of type GeneEvidT_GET_t
{
foreign key (GET1_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
foreign key (confidence) references QualLDict (_code),
foreign key (otherSpecies) references Organism
} under AObject;

create table GeneEvidT of type GeneEvidT-t
{
foreign key (sub) references GeneEvidT
};

create table GeneEvidT of type GeneEvidT-t
{
foreign key (GETl_id) references GeneEvidT,
foreign key (GET2_id) references GeneEvidT
};

create table GeneEvidT hist of type GeneEvidT_hist_t
{
foreign key (GET_id) references GeneEvidT,
foreign key (OHist_id) references OHistory
};

create table GeneEvid of type GeneEvid-t
{
foreign key (gene) references Gene,
foreign key (type) references GeneEvidT,
create table Lib_vExcEnz of type Lib_vExcEnz_t
{
foreign key (Library_id) references Library,
foreign key (RE_id) references REnz
};

-- LINK TABLE between Library -> RestrictionEnzyme

create table Lib_vInsEnz of type Lib_vInsEnz_t
{
foreign key (Library_id) references Library,
foreign key (RE_id) references REnz
};

-- Class: ContigMap

create row type ContigMap_hits_t
{
CMap_id numeric (10,0) not null,
Order_id numeric (10,0) not null
};

create row type ContigMap_t
{
hits SET (ContigMap_hits_t not null)
} under Map_t;

create table ContigMap of type ContigMap_t under Map;

-- LINK TABLE between ContigMap -> Order_

create table ContigMap_hits of type ContigMap_hits_t
{
foreign key (CMap_id) references ContigMap,
foreign key (Order_id) references Order_
};

-- Class: CytogeneticMap -> CytoGMap

create row type CytoGMap_t under Map_t;

create table CytoGMap of type CytoGMap_t under Map;

-- Class: IntegratedMap -> IntegMap

create row type IntegMap_t under Map_t;

create table IntegMap of type IntegMap_t under Map;

-- Class: LinkageMap

create row type LinkageMap_t
{
   gender varchar (20) not null
} under Map_t;

create table LinkageMap of type LinkageMap_t
{
foreign key (gender) references MapSexDict (_code)
} under Map;

-- Class: RadiationHybridMap -> RadHybMap

create row type RadHybMap_t
{
mappingPanel numeric (10,0) not null
} under Map_t;

-- Class: SequenceFeatureMap -> SeqFeatMap

create row type SeqFeatMap_t under Map_t;

create table SeqFeatMap of type SeqFeatMap_t under Map;

-- Class: SyntenyMap

create row type SyntenyMap_t
{
humanMap numeric (10,0) not null
} under Map_t;

create table SyntenyMap of type SyntenyMap_t under Map;

-----

-- Class: MappingPanel -> MP

create row type MP_members_t
{
   MP_id numeric (10,0) not null,
   GS_id numeric (10,0) not null
};

CREATE ROW TYPE MP_t
{
members SET (MP_members_t NOT NULL),
panelSize int,
radDose int,
rhMaps SET (RadHybMap_t NOT NULL),}
CREATE TABLE Repeat OF TYPE Repeat_t
  |
  FOREIGN KEY (repeatType) REFERENCES RepeatTDict (_code)
) UNDER GenSeg;

-----

 Class: SequencingRegion
-----

 CREATE ROW TYPE SequencingRegion_t UNDER GenSeg_t;
 CREATE TABLE SequencingRegion OF TYPE SequencingRegion_t UNDER GenSeg;

-----

 Class: SyntenicRegion
-----

 CREATE ROW TYPE SyntenicRegion_t
  |
  leftInnerMarker numeric (10,0) not null,
  leftOuterMarker numeric (10,0) not null,
  rightInnerMarker numeric (10,0) not null,
  rightOuterMarker numeric (10,0) not null
) UNDER GenSeg_t;

 CREATE TABLE SyntenicRegion OF TYPE SyntenicRegion_t
  |
  FOREIGN KEY (leftInnerMarker) REFERENCES GenSeg,
  FOREIGN KEY (leftOuterMarker) REFERENCES GenSeg,
  FOREIGN KEY (rightInnerMarker) REFERENCES GenSeg,
  FOREIGN KEY (rightOuterMarker) REFERENCES GenSeg
) UNDER GenSeg;

-----

 Class: Population -> Pop
-----

 CREATE ROW TYPE Pop_specs_t
  |
  _aid numeric (10,0) NOT NULL, -- identity
  _oid numeric (10,0) not null,
  _order INT NOT NULL,
  isReleased SMALLINT NOT NULL,
  owner numeric (10,0) not null,
  specName VARCHAR (255) NOT NULL,
  specType INT NOT NULL
)

 CREATE TABLE Pop_specs OF TYPE Pop_specs_t
  |
  PRIMARY KEY (_oid),
  FOREIGN KEY (specType) REFERENCES PopSpecDict (_code),
  FOREIGN KEY (owner) REFERENCES Contact
)

 CREATE ROW TYPE Pop_t
  |
  specifications LIST (Pop_specs_t NOT NULL)
) UNDERDBObject_t;

---

CREATE TABLE Repeat OF TYPE Repeat_t
  |
  FOREIGN KEY (repeatType) REFERENCES RepeatTDict (_code)
) UNDER GenSeg;

-----

 Class: SequencingRegion
-----

 CREATE ROW TYPE SequencingRegion_t UNDER GenSeg_t;
 CREATE TABLE SequencingRegion OF TYPE SequencingRegion_t UNDER GenSeg;

-----

 Class: SyntenicRegion
-----

 CREATE ROW TYPE SyntenicRegion_t
  |
  leftInnerMarker numeric (10,0) not null,
  leftOuterMarker numeric (10,0) not null,
  rightInnerMarker numeric (10,0) not null,
  rightOuterMarker numeric (10,0) not null
) UNDER GenSeg_t;

 CREATE TABLE SyntenicRegion OF TYPE SyntenicRegion_t
  |
  FOREIGN KEY (leftInnerMarker) REFERENCES GenSeg,
  FOREIGN KEY (leftOuterMarker) REFERENCES GenSeg,
  FOREIGN KEY (rightInnerMarker) REFERENCES GenSeg,
  FOREIGN KEY (rightOuterMarker) REFERENCES GenSeg
) UNDER GenSeg;

-----

 Class: Population -> Pop
-----

 CREATE ROW TYPE Pop_specs_t
  |
  _aid numeric (10,0) NOT NULL, -- identity
  _oid numeric (10,0) not null,
  _order INT NOT NULL,
  isReleased SMALLINT NOT NULL,
  owner numeric (10,0) not null,
  specName VARCHAR (255) NOT NULL,
  specType INT NOT NULL
)

 CREATE TABLE Pop_specs OF TYPE Pop_specs_t
  |
  PRIMARY KEY (_oid),
  FOREIGN KEY (specType) REFERENCES PopSpecDict (_code),
  FOREIGN KEY (owner) REFERENCES Contact
)

 CREATE ROW TYPE Pop_t
  |
  specifications LIST (Pop_specs_t NOT NULL)
) UNDERDBObject_t;
CREATE TABLE Pop OF TYPE Pop_t
UNDERDBObject;

CREATE TABLE Var OF TYPE Var_t
(
FOREIGN KEY (mutantAminoAcid) REFERENCES AminoAcidDict (_code),
FOREIGN KEY (variationType) REFERENCES VarType,
FOREIGN KEY (wildTAminoAcid) REFERENCES AminoAcidDict (_code)
) UNDERDBObject;

----
Class: Polymorphism -> Polym
----
CREATE ROW TYPE Polym_locs_t
(_aid numeric (10,0) NOT NULL, -- identity
_oid numeric (10,0) not null,
isReleased SMALLINT NOT NULL,
owner numeric (10,0) not null,
position VARCHAR (255) NOT NULL,
segment numeric (10,0) NOT NULL);

CREATE TABLE Polym_locs OF TYPE Polym_locs_t
( PRIMARY KEY (_oid),
FOREIGN KEY (segment) REFERENCES GenSeg,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE ROW TYPE Polym_t
(allele SET (AllSet_t NOT NULL),
maxHet FLOAT,
locations SET (Polym_locs_t NOT NULL)
) UNDER Var_t;

CREATE TABLE Polym_OF TYPE Polym_t
UNDER Var;

CREATE TABLE GenSeg_var OF TYPE GenSeg_var_t
( PRIMARY KEY (_oid),
FOREIGN KEY (polymorphism) REFERENCES Polym,
FOREIGN KEY (vType) REFERENCES VarType,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_varQ OF TYPE GenSeg_varQ_t
( PRIMARY KEY (_oid),
FOREIGN KEY (polymorphism) REFERENCES Polym,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_segStat OF TYPE GenSeg_segStat_t
( FOREIGN KEY (segStat) REFERENCES SegStatus,
 FOREIGN KEY (sequencedBy) REFERENCES Contact,
 FOREIGN KEY (sequencingStatus) REFERENCES SeqStatDict (_code)
);

CREATE TABLE GenSeg_dData OF TYPE GenSeg_dData_t
( PRIMARY KEY (_oid),
FOREIGN KEY (endPoint_1) REFERENCES EndpointDict (_code),
FOREIGN KEY (endPoint_2) REFERENCES EndpointDict (_code),
FOREIGN KEY (segment_1) REFERENCES GenSeg,
FOREIGN KEY (segment_2) REFERENCES GenSeg,
FOREIGN KEY (units) REFERENCES UnitsDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_allLoc OF TYPE GenSeg_allLoc_t
( PRIMARY KEY (_oid),
FOREIGN KEY (chromosome_3) REFERENCES Chromosome,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_rell OF TYPE GenSeg_rell_t
( PRIMARY KEY (_oid),
FOREIGN KEY (mapElement_4) REFERENCES MapElement,
FOREIGN KEY (map_4) REFERENCES Map,
FOREIGN KEY (probe_4) REFERENCES GenSeg,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_oData OF TYPE GenSeg_oData_t
( PRIMARY KEY (_oid),
FOREIGN KEY (marker_2) REFERENCES GenSeg,
FOREIGN KEY (observationType) REFERENCES Obs_TPlusDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE GenSeg_oMark OF TYPE GenSeg_oMark_t
( PRIMARY KEY (_oid),
FOREIGN KEY (marker-o) REFERENCES GenSeg,
FOREIGN KEY (rel-o) REFERENCES Order_,
FOREIGN KEY (reltypeo) REFERENCES OrderRelDict (_code),
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE AllSet OF TYPE AllSetSt
( FOREIGN KEY (detectMethod) REFERENCES DetMet,
FOREIGN KEY (polymorphism) REFERENCES Polym,
FOREIGN KEY (owner) REFERENCES Contact
) UNDERDBObject;

CREATE TABLE AllFreq OF TYPE AllFreqct
( PRIMARY KEY (_oid),
FOREIGN KEY (alleleSet) REFERENCES AllSet,
check (numberOfChrom >= 0)
) UNDERisObject;

CREATE TABLE AllSet all OF TYPE AllSet allt
( PRIMARY KEY (_oid),
FOREIGN KEY (allnumber) REFERENCES Allele,
FOREIGN KEY (owner) REFERENCES Contact
);

CREATE TABLE AllFreq all OF TYPE AllFreq allt
( PRIMARY KEY (_oid),
FOREIGN KEY (alleleSet) REFERENCES AllSet,
check (numberOfChrom >= 0)
) UNDERisObject;
create table Cft_desc of type Cft_desc_t
( primary key (_oid),
 foreign key (owner) references Contact
);

create row type Cft_elts_t
(
 _aid numeric (10,0) not null,
 _oid numeric (10,0) not null,
 comment varchar (255),
 currentStatus varchar (20) not null,
 isReleased smallint not null,
 item numeric (10,0) not null,
 resolutionStatus varchar (20) not null
);

create table Cft_elts of type Cft_elts_t
( primary key (_oid),
 foreign key (item) references DBObject,
 foreign key (currentStatus) references ApprStatDict (_code),
 foreign key (resolutionStatus) references RStatusDict (_code)
);

-- Not sure about this one
--
create row type Cft_ocflts_t
( _oid numeric (10,0) not null,
 otherConflicts numeric (10,0) not null,
 owner numeric (10,0) not null,
 isReleased smallint not null
);

create table Cft_ocflts of type Cft_ocflts_t
( primary key (_oid),
 foreign key (owner) references Contact
);

create row type Cft_hist_t
( Conflict_id numeric (10,0) not null,
 OHistory_id numeric (10,0) not null
);

create row type Conflict_t
( _oid numeric (10,0) not null,
 displayName varchar (200) not null,
 elements SET (Cft_elts_t not null),
 history SET (Cft_hist_t not null),
 isReleased smallint not null,
 modDate datetime year to day not null,
 otherConflicts numeric (10,0) not null,
 owner numeric (10,0) not null,
 resolutionComment varchar (255),
 reviewed varchar (5) not null,
 url varchar (200),
 version int not null,
 description LIST (Cft_desc_t not null)
);

create table Conflict of type Conflict_t
( primary key (_oid),
 foreign key (reviewed) references YesNoNoDict (_code),
 foreign key (otherConflicts) references Conflict,
 foreign key (owner) references Contact
);

create table Cft_hist of type Cft_hist_t
( foreign key (Conflict_id) references Conflict,
 foreign key (OHistory_id) references OHistory
);

-- Class: ObservationPlus -> ObsPlus
--
create row type ObsPlus_obs_t
( _aid numeric (10,0) not null, -- identity
 _oid numeric (10,0) not null,
 _order int not null,
 observation varchar (20) not null,
 observationInverse varchar (20) not null
);

create table ObsPlus_obs of type ObsPlus_obs_t
( primary key (_oid),
 foreign key (observation) references Obs_TDict (_code),
 foreign key (observationInverse) references Obs_TPlusDict (_code)
);

create row type ObsPlus_t
( _aid numeric (10,0) not null,
 modDate datetime year to day not null,
 observations LIST (ObsPlus_obs_t not null),
 version int not null
);

create table ObsPlus of type ObsPlus_t
( primary key (_oid)
);

-- Class: RelationPlus -> RelPlus
--
create row type RelPlus_rel_t
( _aid numeric (10,0) not null, -- identity
 _old numeric (10,0) not null,
 _order int not null,
 relation varchar (20) not null,
 relationInverse varchar (20) not null
);

create table RelPlus_rel of type RelPlus_rel_t
( primary key (_oid),
 foreign key (relation) references OrderRelDict (_code),
 foreign key (relationInverse) references OrderRelPDict (_code)
);

create row type RelPlus_t
( primary key (_oid),
 foreign key (reviewed) references YesNoNoDict (_code),
 foreign key (otherConflicts) references Conflict,
 foreign key (owner) references Contact
);
create table RelPlus of type RelPlus_t
(
  primary key (_oid)
);

-- Class: SchemaVersion
--
create row type SchemaVersion_t
(
  _oid numeric (10,0) not null,
  majorVersionNumber int not null,
  minorVersionNumber int not null,
  note varchar (255),
  releaseDate datetime year to day not null
);

create table SchemaVersion of type SchemaVersion_t
(
  primary key (_oid)
);
-- newAnnotationObject.sql

-- newCascaded.sql

-- newConflict.sql

-- newDicts.sql

../Informix/HGDB/mappings

ApprovalStatusDict ApprovalStatusDict_t
AvailabilityDict AvailabilityDict_t
BreakpointCauseDict BreakpointCauseDict_t
CellLineTypeDict CellLineTypeDict_t
CompartmentDict CompartmentDict_t
CytoGeneticBandDict CytoGeneticBandDict_t
CytoGeneticResolutionsDict CytoGeneticResolutionsDict_t
DBObjectStatusDict DBObjectStatusDict_t
DNATypeDict DNATypeDict_t
EnzymeUseDict EnzymeUseDict_t
ErrorTypeDict ErrorTypeDict_t
FontStyleDict FontStyleDict_t
FragileSiteAgentDict FragileSiteAgentDict_t
GeneElementTypeDict GeneElementTypeDict_t
GrowthStageDict GrowthStageDict_t
IsolationMethodDict IsolationMethodDict_t
LibLocationTypeDict LibLocationTypeDict_t
NameStatusDict NameStatusDict_t
ObservationTypeDict ObservationTypeDict_t
OccurrenceFreqDict OccurrenceFreqDict_t
<table>
<thead>
<tr>
<th>OrderLikelihoodTypeDict</th>
<th>OrderLikelihoodTypeDict_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderRelationshipDict</td>
<td>OrderRelationshipDict_t</td>
</tr>
<tr>
<td>OrderRelationshipPlusDict</td>
<td>OrderRelationshipPlusDict_t</td>
</tr>
<tr>
<td>OrientationDict</td>
<td>OrientationDict_t</td>
</tr>
<tr>
<td>PopulationSpecDict</td>
<td>PopulationSpecDict_t</td>
</tr>
<tr>
<td>QualitativeLevelDict</td>
<td>QualitativeLevelDict_t</td>
</tr>
<tr>
<td>RNASpecifiedDict</td>
<td>RNASpecifiedDict_t</td>
</tr>
<tr>
<td>RearrangementTypeDict</td>
<td>RearrangementTypeDict_t</td>
</tr>
<tr>
<td>RegulatoryEffectDict</td>
<td>RegulatoryEffectDict_T</td>
</tr>
<tr>
<td>RelativeOrientationDict</td>
<td>RelativeOrientationDict_t</td>
</tr>
<tr>
<td>RepeatTypeDict</td>
<td>Repeat_TypeDict_t</td>
</tr>
<tr>
<td>ResolutionStatusDict</td>
<td>ResolutionStatusDict_t</td>
</tr>
<tr>
<td>SepMethodDict</td>
<td>SepMethodDict_t</td>
</tr>
<tr>
<td>SequencingStatusDict</td>
<td>SequencingStatusDict_t</td>
</tr>
<tr>
<td>VectorTypeDict</td>
<td>Vector_TypeDict_t</td>
</tr>
<tr>
<td>VisualMethodDict</td>
<td>VisualMethodDict_t</td>
</tr>
<tr>
<td>YesNoUnknown_UnkDict</td>
<td>YesNoUnknown_UnkDict_t</td>
</tr>
<tr>
<td>YesNoUnknown_YesDict</td>
<td>YesNoUnknown_YesDict_t</td>
</tr>
<tr>
<td>YesUnknown_UnkDict</td>
<td>YesUnknown_UnkDict_t</td>
</tr>
</tbody>
</table>

```
-- newExternalLink.sql
--
ExternalLink  ExtLink
ExternalLink_t ExtLink_t
ExternalLink_submissions  ExtLink_submission
ExternalLink_submissions_t ExtLink_submission_t
CitationLink  CitLink
CitationLink_t CitLink_t
CitationLink_DBObjects  CitLink_DBObjects
CitationLink_DBObjects_t CitLink_DBObjects_t
GenericLink  GenLink
GenericLink_t GenLink_t
GenericLink_DBObjects  GenLink_DBObjects
GenericLink_DBObjects_t GenLink_DBObjects_t
NucleicAcidSequenceLink  NuclASeqLink
NucleicAcidSequenceLink_t NuclASeqLink_t
ProteinSequenceLink  ProtSeqLink
ProteinSequenceLink_t ProtSeqLink_t

-- newObjects.sql
--
ObjectHistory  OHistory
ObjectHistory_t OHistory_t
ObjectHistory_changes  OHistory_changes
ObjectHistory_changes_t OHistory_changes_t
ObjectName  OName
ObjectName_t OName_t
ObjectName_history  ONameHistory
ObjectName_history_t ONameHistory_t
ObservationPlus  ObsPlus
ObservationPlus_t ObsPlus_t
ObservationPlus_observations  ObsPlus_observations
ObservationPlus_observations_t ObsPlus_observations_t
RelationPlus  RelPlus
RelationPlus_t RelPlus_t
RelationPlus_relations  RelPlus_relations
RelationPlus_relations_t RelPlus_relations_t
RestrictionEnzyme  ResEnzyme
RestrictionEnzyme_t ResEnzyme_t
RestrictionEnzyme_history  ResEnzymeHistory
RestrictionEnzyme_history_t ResEnzymeHistory_t
```
<table>
<thead>
<tr>
<th>VariationType</th>
<th>VariationType_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>VariationType_TC</td>
<td>VariationType_TC_t</td>
</tr>
<tr>
<td>VariationType_history</td>
<td>VariationType_history_t</td>
</tr>
</tbody>
</table>

-- newDBObject.sql --

<table>
<thead>
<tr>
<th>DetectMethod_probes</th>
<th>DetMet_pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>DetectMethod_probes_t</td>
<td>DetMet_pro_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression_t</td>
<td>Expr_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression_pattern</th>
<th>Expr_pat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression_pattern_t</td>
<td>Expr_pat_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GeneFamily</th>
<th>GenFam</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneFamily_t</td>
<td>GenFam_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GeneFamily_genes</th>
<th>GenFam_genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneFamily_genes_t</td>
<td>GenFam_genes_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GeneFamily_subFamilies</th>
<th>GenFam_subF</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneFamily_subFamilies_t</td>
<td>GenFam_subF_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GeneProduct</th>
<th>GenPro</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneProduct_t</td>
<td>GenPro_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GeneProduct_genomes</th>
<th>GenPro_genomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneProduct_genomes_t</td>
<td>GenPro_genomes_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protein_proteinSequences</th>
<th>Protein_proSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protein_proteinSequences_t</td>
<td>Protein_proSeq_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment</th>
<th>GenSeg</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_t</td>
<td>GenSeg_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment clones</th>
<th>GenSeg_clones</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_clones_t</td>
<td>GenSeg_clones_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_distanceData</th>
<th>GenSeg_dData</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_distanceData_t</td>
<td>GenSeg_dData_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_allLocations</th>
<th>GenSeg_allLoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_allLocations_t</td>
<td>GenSeg_allLoc_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_maySearchCoords</th>
<th>GenSeg_mSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_maySearchCoords_t</td>
<td>GenSeg_mSC_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment variants</th>
<th>GenSeg_var</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_variants_t</td>
<td>GenSeg_var_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_variantsQ</th>
<th>GenSeg_varQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_variantsQ_t</td>
<td>GenSeg_varQ_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_amplimers</th>
<th>GenSeg_amp</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_amplimers_t</td>
<td>GenSeg_amp_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_cytoLocations</th>
<th>GenSeg_cytoL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_cytoLocations_t</td>
<td>GenSeg_cytoL_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_otherLocations</th>
<th>GenSeg_otherL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_otherLocations_t</td>
<td>GenSeg_otherL_t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GenomicSegment_relatedLocations</th>
<th>GenSeg_reL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenomicSegment_relatedLocations_t</td>
<td>GenSeg_reL_t</td>
</tr>
<tr>
<td>GenomicSegment_mayCoords</td>
<td>GenSeg_mC</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>GenomicSegment_genes</td>
<td>GenSeg_genes</td>
</tr>
<tr>
<td>GenomicSegment_orderData</td>
<td>GenSeg_oData</td>
</tr>
<tr>
<td>GenomicSegment_otherMarkers</td>
<td>GenSeg_oMark</td>
</tr>
<tr>
<td>GenomicSegment_ESTs</td>
<td>GenSeg_ESTs</td>
</tr>
<tr>
<td>GenomicSegment_sequencingStatus</td>
<td>GenSeg_seqStat</td>
</tr>
<tr>
<td>Amplimer_primers</td>
<td>Amp_primers</td>
</tr>
<tr>
<td>Amplimer_sequence</td>
<td>Amp_seq</td>
</tr>
<tr>
<td>CellLine</td>
<td>CL</td>
</tr>
<tr>
<td>CellLine_cytoRegions</td>
<td>CL_cytoR</td>
</tr>
<tr>
<td>CellLine_rearrangements</td>
<td>CL_rearr</td>
</tr>
<tr>
<td>ChromosomeReagent</td>
<td>ChromR</td>
</tr>
<tr>
<td>ChromosomeReagent_cytoRegions</td>
<td>ChromR_cytoR</td>
</tr>
<tr>
<td>Clone_libraryAddresses</td>
<td>Clone_libAddr</td>
</tr>
<tr>
<td>Clone_restrictionFragments</td>
<td>Clone_reFrag</td>
</tr>
<tr>
<td>Clone_genomeExcisionEnzymes</td>
<td>Clone_gExcEnz</td>
</tr>
<tr>
<td>Clone_vectorExcisionEnzymes</td>
<td>Clone_vExcEnz</td>
</tr>
<tr>
<td>Clone_vectorInsertionEnzymes</td>
<td>Clone_vInsEnz</td>
</tr>
<tr>
<td>CytogeneticMarker</td>
<td>CytoGM</td>
</tr>
<tr>
<td>CytogeneticMarker_resolutions</td>
<td>CytoGM_res</td>
</tr>
<tr>
<td>Library_genomeExcisionEnzymes</td>
<td>Lib_gExcEnz</td>
</tr>
<tr>
<td>Library_vectorExcisionEnzymes</td>
<td>Lib_vExcEnz</td>
</tr>
<tr>
<td>Library_vectorInsertionEnzymes</td>
<td>Lib_vInsEnz</td>
</tr>
<tr>
<td>MappingPanel</td>
<td>MP</td>
</tr>
<tr>
<td>MappingPanel_members</td>
<td>MP_members</td>
</tr>
<tr>
<td>RegulatoryRegion</td>
<td>RegR</td>
</tr>
<tr>
<td>RegulatoryRegion_transFactors</td>
<td>RegR_transF</td>
</tr>
<tr>
<td>Distance</td>
<td>Dist</td>
</tr>
<tr>
<td>Distance_segments</td>
<td>Dist_seg</td>
</tr>
<tr>
<td>PCRCondition</td>
<td>PCRCond</td>
</tr>
<tr>
<td>PCRCondition_protocol</td>
<td>PCRCond_prot</td>
</tr>
<tr>
<td>PCRCondition_amplimer</td>
<td>PCRCond_amp</td>
</tr>
<tr>
<td>Population</td>
<td>Pop</td>
</tr>
<tr>
<td>Population_specifications</td>
<td>Pop_specs</td>
</tr>
<tr>
<td>Rearrangement</td>
<td>Rearr</td>
</tr>
<tr>
<td>Rearrangement_breakpoints</td>
<td>Rearr_breakp</td>
</tr>
<tr>
<td>Variation</td>
<td>Var</td>
</tr>
<tr>
<td>Variation_ASO</td>
<td>Var_ASO</td>
</tr>
<tr>
<td>Variation_typeQueries</td>
<td>Var_typeQ</td>
</tr>
</tbody>
</table>