

A CORBA server for the Radiation Hybrid DataBase

P. Rodriguez-Tomé, C. Helgesen, P. Lijnzaad and K. Jungfer

EMBL Outstation, European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton, Cambridge CB10 1SD, UK
rhdb@embl-ebi.ac.uk

Abstract

Modern biology depends on a wide range of software interacting with a large number of data sources, varying both in size, complexity and structure. The range of important databases in molecular biology and genetics makes it crucial to overcome the problems which this multiplicity presents. At EMBL-EBI we have started to use CORBA technology to support interoperability between a variety of databases, as well as to facilitate the integration of tools that access these databases. Within the Radiation Hybrid DataBase project we are confronted daily with the interoperation and linking issues. In this paper we present a CORBA infrastructure implemented to access the Radiation Hybrid DataBase.

Introduction

The diversity and structure of biological data as well as the complexity of the databases that store them, complicate their use. These databases are maintained on heterogeneous computer systems using heterogeneous database management systems which include Relational DBMS, Object-Oriented DBMS, flat-files as well as home-made systems, which often do not follow international standards. This is further compounded by the variety of forms in which applications require the data to be provided. The huge amount of data generated by the various genome sequencing projects opens the road to genome comparison at the sequence level. By comparing genome maps we will be able to study genome organization and evolution in great detail, identify disease genes etc. The data are becoming more and more complex, and the need for expert curation of the databases increases. It is not feasible anymore to collect all the data in one central place, especially if the expertise to curate the data is not available at this site. However it is crucial for researchers to be able to access all these data in an efficient way.

It is thus becoming increasingly important to devise a strategy which will facilitate the smooth integration of data and software, or, in today's jargon, achieve interoperability. One could argue that the existing state of affairs, i.e. databases providing unique identifiers and

referring to other database's own identifiers, allows the users to access all the relevant data by following these links. Even with the use of software like SRS (Etzold and Argos 1993) that make use of these links, this can be an extremely tedious task. True interoperability should aim at:

- providing a description of the data independent of the database implementation,
- facilitating the development and evolution of software,
- being accessible through the Internet from heterogeneous architectures,
- leaving a large degree of autonomy to each database at the implementation level.

Our view is that this will not be achieved by imposing a common schema or a common DBMS system to all the biological databases, but by following a standard to support communication and interoperability.

One of the significant developments in this area derives from the work of the Object Management Group (OMG) (<http://www.omg.org/>), a collaboration including the major commercial software vendors, information technology vendors and the academic establishment. Over the years the OMG has defined an open standard, the Common Object Request Broker Architecture (CORBA), which allows interoperability between distributed applications (Siegel 1996).

The Radiation Hybrid DataBase

Radiation hybrids are a powerful means to generate linkage maps of the genome, and form the basis for many of the current sequence ready maps. Radiation hybrids are produced by fusing irradiated donor cells with recipient rodent cells. These hybrids cells lines, grouped in 'panels' of identical radiation doses, each contain many large chromosome fragments produced by radiation breakage. Each panel is screened by PCR amplification, producing 'scoring data', to identify those hybrids that have retained a given locus. Nearby loci will tend to show similar retention patterns thus allowing proximity to be inferred (Walter et al. 1994)

An international collaborative project is producing a large number of radiation hybrid data (PCR hybridization results) for the human genome, allowing the generation of a very precise STS map (Schuler et al. 1996), and

integrating human cDNA with meiotically ordered polymorphic markers. These maps are indispensable in positional cloning of candidate genes to study multifactorial diseases.

The DataBase

Since July 1995, the EBI has maintained RHdb (Rodriguez-Tomé and Lijnzaad 1997) a public database for radiation hybrid data (<http://www.embl-ebi.ac.uk/RHdb>). The database contains both the raw scoring data on radiation hybrid panels as well as resulting maps. All entries are systematically cross-referenced to other databases when appropriate.

Data is submitted in a specific format, and WWW based tools allow the user to access the data via Internet. The traditional 'flat-file' format is used to periodically produce a complete human-readable dump the database. The flat-files are also indexed using the SRS system to provide query access.

Our first objective was to generate a data model independent of the actual database implementation. For that we have used the Object Modeling Technique (OMT) (Rumbaugh et al. 1991) to produce a data model that reflects our biological understanding of these data.

This data model can be mapped to different DBMSs, both relational and object oriented ones. We have mapped our object model to a relational database using Oracle™, which is the RDBMS used at the EBI.

Support of user needs

With the present implementation, users cannot directly access the database and build their own queries. It does not support complex queries or batch processing. We could say that in this form, the database does not meet the full needs of the user community:

1. provide a set of standard interfaces that permit the easy retrieval of answers to common questions,
2. support for extracting large amounts of data for specialised mapping software,
3. support for complex *ad hoc* queries,
4. allow the client to run on the platform of choice for the user,
5. ability to respond to the changing needs of the community.

CORBA and distributed objects

CORBA, the Common Object Broker Architecture, develops on the notion that in order to facilitate interoperability, it is necessary to make as many implementation details (location, network protocols, operating system, implementation language) as transparent as possible. This goal is achieved by a strict separation of interface and implementation which allows the developer to concentrate on the functionality of an application.

Servers are represented in CORBA by objects whose *interface* is specified by using the Interface Definition Language (IDL). The services that can be delivered to a client by the server are described in the *interface*. An Object Request Broker (ORB) serving as a central mediator, relays the requests from client to server, and the result back to the client. The interface written in IDL is compiled into language specific code (e.g. C, C++, SmallTalk, Java) at either end of the ORB into an interface *stub* at the client end and an interface *skeleton* at the server end. It enables the development of distributed objects : objects that are resident somewhere on a network.

To be CORBA-compliant, a vendor must implement the CORBA core: the Object Model, the Architecture, the IDL syntax and semantics, the ORB and at least one target language mapping. In addition the vendor is free to implement a number of services that will follow separate compliance points.

The ORB stores the IDL definition of each object in the Interface Repository. The ORB and the interface definitions make clients and servers independent of each other and allow to combine distributed objects in a flexible manner.

Every CORBA object has to be registered with the ORB. After registration, the ORB knows what operations and object supports and how to access it. A client can then access a CORBA object with no prior knowledge of its implementation, by obtaining a reference to this object from the ORB, thus giving the basis for interoperability.

A CORBA Application

As a test case we have applied the CORBA technology to support external access to the Radiation Hybrid Database. This has proven valuable experience for subsequently building CORBA servers for the other databases at EBI.

The Interface Definition

We can define the main object in the database as *the scoring results of a specific PCR hybridization on a given panel using a specific STS (or marker) for which the primer sequences, and type are given*. This object is named an experiment or assay, and is given a unique identifier (id) in the database. STSs are the skeleton of the physical maps, genetic maps and RH maps, shared by all genomic databases. They are of interest to any researcher working on a particular genomic region.

The second most important objects are the resulting maps. As a first step we have only implemented these two objects in the interface definition.

In this server implementation the data is represented by two different components :

1. Assay and AssayExtent (a collection of database objects of the type Assay)
2. Map and MapManager.

The structure and services of the assay Extent: The AssayExtent is represented by a single CORBA object

which - given a specific id - returns a structure containing the experimental data :

- the panel and score vector,
- the primer sequences,
- the type of Marker,
- the chromosome localisation,
- the cross-references to other databases.

The full IDL for the AssayExtent is shown below.

```
module RH {
  struct Assay {
    string id;
    string chromosome;
    string panel;
    string score;
    string primer_a;
    string primer_m;
    string crossrefs;
    string types;
  };
  interface AssayExtent {
    typedef sequence<string> IDs;
    typedef sequence<string> Scores;
    typedef sequence<string> Crossrefs;
    typedef sequence<string> Types;
    boolean exists(in string id);
    Assay get(in string id);
    Scores getScores(in IDs ids);
    Crossrefs getCrossrefs(in IDs ids);
    Types getTypes(in IDs ids);
  };
};
```

The structure and services of the Map object: The maps are, in contrast to the assays, represented by individual CORBA objects (see table below). A client can retrieve references to the maps from the map manager. Having a reference to a map, the client will communicate directly with the map object, bypassing the manager.

```
struct Marker {
  string id;
  float pos;
  boolean fw;
};
typedef sequence<Marker> Markers;
interface Map {
  string id();
  string chromosome();
  long size();
  float top();
  float bottom();
  float pos(in string id);
  Markers getAllMarkers();
  string first();
  string prev(in string id);
  string next(in string id);
  string last();
};
```

```
typedef sequence<Map> Maps;
interface MapManager {
  Map get(in string mapid);
  Maps getAll();
  Map find(in string id);
};
typedef sequence<boolean> Bools;
interface QueryEvaluator {
  Bools eval(in Map map, in string
query);
};
};
```

The server implementation

The main steps of the server implementation and execution are :

1. Initialise the ORB
CORBA.ORB orb = CORBA.ORB.init();
CORBA.BOA boa = orb.BOA_init();
2. Instantiate the distributed objects (here the MapManager)
MapManager m = new
MapManager("EBIMapManager");
3. Register the distributed objects with the ORB
boa.obj_is_ready(m);
4. Register the server program and enter the event loop, waiting for client requests
boa.impl_is_ready();

The implementation also includes the implementation of the methods as database queries. The code for the server has been written and compiled on a Sun™ workstation using the Visibroker™ environment from Visigenic (<http://www.visigenic.com/>).

Two different implementations of the server have been developed, using the Java™ or C++ languages (Jungfer et al. forthcoming), respectively.

A Java server

The IDL-to-Java™ compiler generates a skeleton, used to implement the server side. The developer has to provide the code to access the database. At the time of this work we did not have any Java™ interface to the Oracle™ database. We accessed it through OraPerl5 scripts called from the Java™ code.

A C++ server

The same IDL is now compiled using the IDL-to-C++ compiler. Our aim in this case is to directly access the relational database. The RDBMS we use, Oracle™, does not yet provide a mapping between CORBA Objects or C++ classes and the relational schema.

We have used Persistence™ (from Persistence Inc., <http://www.persistence.com>) to map the C++ classes of our skeleton to the relational schema. Persistence™ generates C++ classes and methods reflecting the tables and

relationships in the relational schema as well as standard methods to access the relational tables : insert, delete, select, update. Only the more domain specific queries need to be implemented. The Persistence™ generated classes are then used to implement the C++ classes and methods defined in the IDL to C++ mapping (Fig. 1).

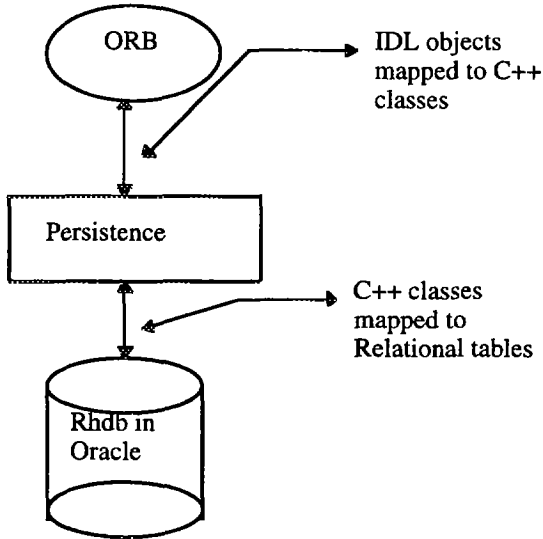


Fig. 1 : The C++ server implementation

Client implementation:

On the client side, the IDL is compiled to generate the stubs. All the methods for querying the database being implemented on the server side, the developer of the client need only to write the application specific routines. The client program needs to connect to the ORB, get the object references and invoke the methods applied to the objects :

1. Initialise ORB
`ORBA.ORB orb = CORBA.ORB.init(this);`
2. Get object reference to map manager
`RH.MapManager mm =
RH.MapManager_var.bind("EBIMapManager");`
3. Invoke methods as they were specified in the IDL definition of the map manager. Here we ask the question: "Get object reference to the map with the id 'FW1'"
`Map map = mm.get("FW1");`

Client examples

As an example of using the server, we have implemented different clients in Java™. The following client was developed using Visibroker. More information can be found at http://www.ebi.ac.uk/RHdb/RHdb_corba.html.

The Map viewer applet

The purpose of this application is to visualize the maps

contained by the database, and extract information concerning markers included in specific areas of a map.

The Java™ client currently displays six maps side by side. Clicking on a map position will retrieve the corresponding information by accessing the AssayExtent object. The user can highlight all markers on the maps which correspond to a specific type, or satisfy an arbitrary score vector specification. This query is resolved by using the query evaluator component which calculates the markers to be highlighted using both the AssayExtent and Map objects.

This map viewer is a Java™ applet that can be accessed with a Java™ compatible WWW browser.

Conclusion

CORBA is already an industry standard. Application of this standard to the EBI databases and services will provide us with a very flexible and dynamic approach that will be able to reflect our understanding of biology, and the structure of biological information as it evolves. We have initiated a project involving specialist databases in Europe that have a long-term collaboration with the EBI to include them in the CORBA approach. This will initially allow a better exchange of data between the EBI and its collaborators, and subsequently offer access to these databases to the research community.

References

- Etzold, T. and Argos, P. 1993. SRS - an indexing and retrieval tool for flat file data libraries. *Comput. Appl. Biosci.* 9:49-57.
- Siegel, J., 1996. *CORBA, Fundamentals and Programming*. Wiley Computer Publishing Group.
- Walter, M.A., Spillet, D.J., Thomas, P., Weissenbach, J. and Goodefellow, P.N. 1994. A method for constructing radiation hybrid maps of whole genomes. *Nature Genet.*, 7:22-28.
- Schuler, G.D. et al., 1996. A Gene Map of the Human Genome *Science*, 274(5287):540-546.
- Rodriguez-Tomé, P. and Lijnzaad, P. 1997. The Radiation Hybrid DataBase. *Nucleic Acid Res*, 25(1)81-84.
- Rumbaugh, J. et al, 1991. *Object Oriented modeling and Design*, Prentice Hall.
- Jungfer, K. et al, Forthcoming.