COMBINING APPLICATIONS AND REMOTE DATABASES VIEW IN A COMMON SQL DISTRIBUTED GENOMIC DATABASE

Pierre-Emmanuel Gros, Joan Hérisson, Nicolas Ferey and Rachid Gherbi

Bioinformatics team, LIMSI-CNRS, Université Paris-Sud,
BP 133, F-91403 Orsay Cedex, France
http://www.limsi.fr
E-mail: genoteam@limsi.fr
Phone: +33 1 69 85 81 64
Fax: +33 1 69 85 80 88

Abstract

Huge volumes of bioinformatics data are emerging from sequencing efforts, gene expression assays, X-ray crystallography of proteins, and many other activities. High-throughput experimental methods produce masses of data, so that the whole of biology has changed from a data-light science into a data-driven science. Currently there are a lot of databases and software tools dealing with these genomic data. In general, each tool and database uses a different type of data in exchange protocols, and usually they offer specific services. These Databases are design with different languages and run on different operating systems. Therefore biologists are in a difficult situation where they have to use, process and store heterogeneous data when using heterogeneous software tools and databases. Our framework, GenoMEDIA provides two main middleware to help for this integration, Lydia and Antje. On the one hand, the Lydia middleware offers us facilities for working simultaneously with a variety of Services and Databases. On the other hand, the Antje one ,with the concept of remote view, is designed to allow users to manage multiple heterogeneous remote databases in a uniform vision. The aim of this paper is to present GenoMEDIA and how heterogeneous databases and remote services are integrated, in particular how Antje was designed, implemented and tested with various genomic

Keywords: Bioinformatics, Distributed Databases, Service and Database Integration, Distributed Genomic Platform.

1 INTRODUCTION

The sequencing of the human genome and that of other organisms is just one element of an emerging trend in the life sciences: while the knowledge, experience, and insight of researchers remain indispensable elements, the understanding of life processes is increasingly a data driven enterprise. Huge volumes of bioinformatics data are emerging from sequencing efforts, gene expression assays, X-ray crystallography of proteins, and many other activities. High-throughput experimental methods produce masses of data, so that the whole of biology has changed from a data-light science into a data-driven science. The field of Bioinformatics relates to the collection, organization and analysis of this large amount of biological data using networks of computers and databases (usually with reference to the genome project and DNA sequence information). To give an impression of the data's size, consider the numbers below:

- DNA sequences: { 16.000.000.000 bases (= 16 Gbp (Giga base pairs)) { Human genome = 3.2 Gbp (equivalent size to 6 complete years of the New York Times)
- Literature: { PubMed 14.000.000 abstracts
- Protein sequences: { SWISSPROT: 130.000 annotated protein sequences{ TrEMBL: 850.000 protein sequences
- Protein structures: { PDB: > 25.000 protein structures with an average of ca. 400 residues

A large number of systems are now available that provide the access to various biological databanks. Some of them offer the possibility to access to multiple distributed databanks in order to gather and integrate the maximum of necessary resources. One of the most used integration approach is the "linkdriven" one. This "linkdriven" federation is used by many websites, which provide an interface to multiple biological resources. In such websites interfaces, the integration

systems are generally built with an architecture based on data-mediator and wrapper. The system provides its users with a single cohesive view with seamless integrated information. This integration cannot be easily accomplished by a single information provider. The well Known *Genbank* databases hosted by the National Center of Biotechnology Information (NCBI) of the USA can be considered as the largest information integration system with more than ten databases interlinking with each other via hyperlinks. The information integration systems allow the users to formulate their queries in domain relation terms *de_ned* in advance to describe a target application domain. A mediator will then construct a query plan to decompose the user query (Burger, Link & Ritter, 1997) into sub queries to external data sources, and determine the execution and data flow orders of the sub-queries. The mediators (Risch, 2004) (Risch, Josifovski, & Katchaounov, 2003) (Burger, Link et al, 1997) (Mork, Halevy & Tarczy-Hornoch, 2001) rely on wrappers to provide some transparent access to the data sources. The wrappers serve as translators between the mediator and the data sources.
In many cases, these Integration Systems also allow users to invoke sequence alignment algorithms such as BLAST as the result of a query. Usually, the users need to explicitly state which resources should be used for retrieving the answers, requiring good knowledge of the underlying sources. The data source systems are often implemented using specialised retrieval packages. Most of the integration is linkdriven based and is achieved by the creation of cross-reference indexes. For instance, the *SRS* language performs a search in databanks indexes (including string search, regular expressions, numerical ranges and dates), with combinations of queries using 'and', 'or' and 'not'. The link construct is introduced according to the combination of databanks. This allows queries of both forms `and' all entries in databank *dbA* that are referenced in databank *dbB*, and `and all entries in databank *dbA* that reference entries in databank *dbB*. The two main advantages of such systems is the fact that the queries relating to knowledge in different databanks can be invoked, and the query processing could be preformed in a fast way. However, although this is a first step in integrating data sources, such solution does not handle the differences in terminology used in the underlying sources, it is syntactic based and it only allows limited query functionality over multiple databanks. Moreover, adding a new resource requires cross-referencing with the other resources, which is not easy to do.
Existing systems like BioKleisli, K2 (Davidson, Crabtree, Brunk, Schug, Tannen, Overton et al, 2001), TINet (Eckman, Kosky & Laroco, 2001), P/FDM (Kemp, Angelopoulos & Gray, 2000;Embury, 1991), TAMBIS and BioTRIFU, use an approach based on view integration. The next step of integration is performed by broker pattern upon a query language. In this approach, the underlying schemas are integrated to form a global schema. The global schema is queried in a high-level language such as CPL (e.g. BioKleisli) or OQL (e.g. K2). The languages in TAMBIS (Stevens, Goble, Paton, Bechhofer, Baker & Brass, 2003) (Goble, Stevens, Bechhofer, Paton, Baker, Peim et al, 2001) and BioTRIFU (Lambrix & Jakoniene, 2003) have been inspired by the study of the use of current biological databanks. Usually, the federated language is broken in several short fragments, and every fragment has to access each data source (Finin, Fritzson, Mckay.& Mcentire, 1994).
In TAMBIS project, the queries written over the GRAIL description logic (DL) were mapped into execution plans in CPL/Kleisli (Hart & Wong, 1994).

The most impressive broker is the IBM's DiscoveryLink system (Haas, Schwarz, Kodali, Kotlar.& Swope, 2001). In such system, the language used is *SQL* and a wrapper is used in order to access database. The general schema of using DiscoveryLink is as foolowing:

1. Mapping the information stored by the data source into DiscoveryLink's relational data model,
2. Informing DiscoveryLink about the query processing capabilities of the data sources,
3. Mapping the query fragments submitted to the wrapper into requests that can be processed using the native query language or programming interface of the data source,
4. Issuing such requests and, following their execution, returning the results.

Our framework *GenoMEDIA(*Gros, Ferey, Hérisson, & Gherbi, 2004*)* is close in term of goals to DiscoveryLink approach, but it achieves its aims in a different manner. Like DiscoveryLink, we used *SQL* as a federator language (Agha,1986; Williams,1997). In our approach, and in particular in the Antje concept, SQL Table takes the role of "volatile" wrapper. Indeed, for Antje, the wrapper is "volatile" insofar as the data format is extracted from the structure of the table.

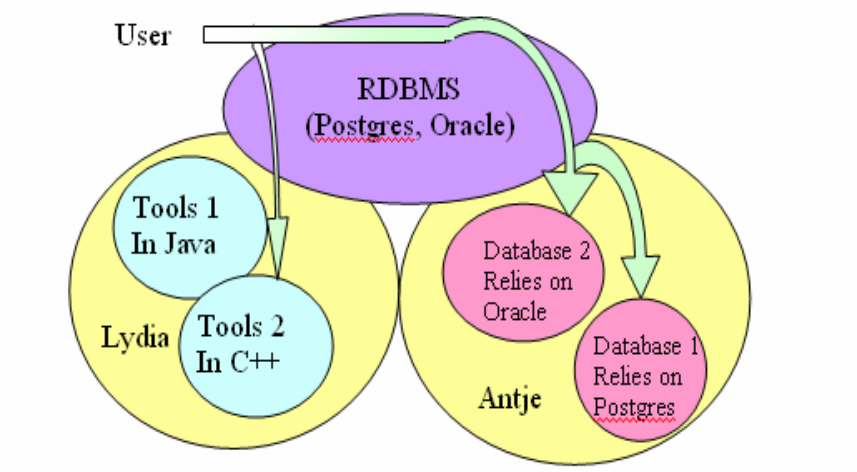

Figure 1. GenoMEDIA Architecture

The *GenoMEDIA* architecture (figure 1), allows a user to send an augmented unified query through standard *SQL* that involves remote heterogeneous databases and services. Unlike *GeneWeaver* (Bryson, Luck, Joy & Jones, 2000), *GenoMEDIA* keep a fully *SQL* approach for managing existing *DBMS* technologies (PHP, ODBC,..). Like in Oracle's (Oracle, n.d) distributed database architecture, *Antje*'s goal is to facilitate access to databases on multiple computers by making them appear as if they were a single file available through a user-friendly interface. *Antje* use the notion of remote view instead of the notion of database link. Users do not have to use the database link into the *SQL* statements. This paper will discuss how *Antje* was designed, implemented and tested in efforts to unify remote biological databases. Finally, we will present our integration graphic user interface for *Antje*; *AntjeViewer*. The main goal of *AntjeViewer* is to allow users to easily integrate databases without any lines of *SQL*.

2 INTEGRATION OF REMOTE DATABASE INTO GENOMEDIA:THE ANTJE APPROACH

2.1 Why a database is not a simple web service?

The goal of *Lydia* middleware is to map a function or tools to a *SQL* function. However, the way of integrating multiple databases within a unified scheme is different for at least two reasons: A tool is an object/agent, in the sense of using parameters and results for its computation. A database, to the contrary, is a bulk of data with no notion of parameters or result attached to it. The *SQL* query serves for prescribing the parameters and the results that can be used with a database.

gene name	protein name	translation
A1	P1	AATG
A2	P2	ATTG

For the query: "select protein name from table1" we consider no parameter and two results (P1 and P2). But for the query " `select gene_name from table1 where translation='AATG'` ", the parameter is the translation and the result is the "gene_name". A bad solution for integrating databases is to enumerate the all requests and to attempt to map this enumeration on the services. In the example, this would be performed as follows:

- Select gene_name from table1
- Select protein_name from table1
- Select translation from table1

…

We obviously cannot enumerate all these queries, which mean that we cannot integrate databases in the same way as tools.

2.2 Distributed Database System Approach

We can find and access a large set of biological databases over the network. One of our goals in this work was to make this set appear to users as a single source. With *Antje*, we aim to "encapsulate" heterogeneous distributed databases in order to make them appear as a single and local database. So, the main role of *Antje*, a backend process behind a standard *DBMS*, is to hide the distribution and heterogeneity of the data.

We consider that a database is a network-based application component with data-oriented architecture using standard interface languages like *SQL* for an entity relationship model. In order to transform a database into a web service, we have to find a uniform database connection.

3 THE ANTJE CONCEPT

Before describing the *Antje* concept, we present in the following some goals that motivated our design of *Antje*:

- Perform the integration in an easy and transparent manner.
- Avoid writing from outset a new distributed SQL server. Indeed, several technologies (JDBC (Sun, n.d) (ObjectWeb, n.d), ODBC (Microsoft, n.d), PHP) currently exist for DBMS, and we do not want to have a high development cost for the client application.
- Allow for a variety of database integration configurations because, for various reasons, biologists might not want to integrate all the data.

The *SQL* language defines the notion of a table. A table is a set of rows, and a row is an ordered list of values. Similarly, *SQL* define the notion of a view table. A view table never exists physically. Instead, it is formed from the rows of the underlying base table(s) when this view table is specified in a *SQL* statement. The syntax of a view definition in *SQL* is usually: `CREATE VIEW <table_name> AS <query_expression>`
For example, the query "create view rich_employee as select name from employee where salary>300000", defines a view formed with the employee's name whose salary is greater than 300000.
View table is usually used in order:

- To restrict data access
- To make easier some complex queries
- To allow data independence
- To present different views of the same data

Our approach extends the notion of view to the notion of "remote view". In the view table, we allow user to enter his view with a remote *SQL* query. To achieve this aim, we extend the "create view" syntax to:
`Create view <table_name> as 'remote query'`, where the syntax of 'remote query' is:
execute*Antje* ('remote database connection' , ' network localization of the database ', 'remote *SQL* query').
The parameter 'remote database connection' is a database connection like ODBC or JDBC. Therefore, when the *DBMS* accesses to our view, the method execute*Antje* is called in order to fill the view with the remote query. The use of a standard connection like ODBC allows us to resolve the problem of connecting heterogeneous *DBMS*.
But there are many more problems associated with heterogenous *DBMS* than just the connection problem. For example, not all *DBMS* accepts "cursor". Therefore, *Antje* give to the user a wrapper for ODBC/JDBC driver in order to assure some kind of services:

- The capabilities to count row of a query
- The capabilities to use cursor, in order to allow a memory management upon the retrieval data from a remote query.

With this approach of wrapped driver, we are able to integrate data which come from:

- Entity relationship databases like PostgreSQL, MySQL or Oracle
- Excel spreadsheet
- Comma-separated or tab-separated flat file (for SGD (SGD, n.d) database)

Therefore, *Antje* allows performing *SQL* statements upon a large amount of heterogeneous data.
We give here below an illustration with two scenarios of our mechanism for the view creation and integration.

3.1 Two scenarios of integration

In this section, we will present two scenarios dealing with the integration process. We integrate a part of *ADNArch* (entity relationship of *GenBank* (NCBI, n.d) database) and a part of PROSE (entity relationship of *Swissprot* (SIB/EBI, n.d) (GUMC, n.d) database). From *GenBank*, we will map the CDS (coding domain sequence) of Aeropyrum pernix organism.
This CDS table contains few fields: transl_except, codon_start, db_xref, translation, note, gene, product, protein_id. We decide to import the gene and note field.

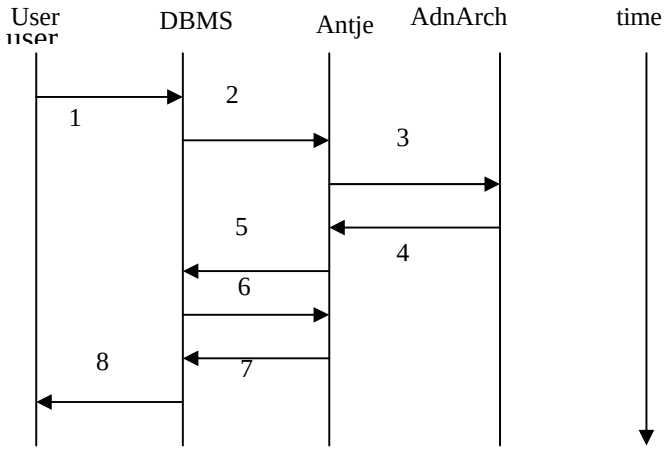

Figure 2. Scenario of the table integration with a remote view into Antje

The goal of this first scenario (Figure 2) is to show how a table is mapped into *Antje*. In this scenario, *Antje* will map a remote table upon a *SQL* view.

1. The user asks, through a special function, the *DBMS* to integrate the table CDS into *Antje*. He must give to *Antje*:
   - The network location of the database that corresponds to the requested table.
   - The field to map into Antje.
2. The DBMS creates a JVM (GBorg, n.d)(stands for Java Virtual Machine) with a standard value, and calls Antje with the user parameter.
3. Antje runs upon JVM. The middleware creates a JDBC connection to the database. JDBC technology is an API that provides cross-DBMS connectivity to a wide range of SQL databases and access to other tabular data

sources, such as spreadsheets or flat files. The “send” request retrieves information like: The type of the requested field (in our example, the gene field has the SQL type VARCHAR, and the note field is a TEXT type) and an evaluation of the maximum memory needed for a request upon this table.

4. The requested database sends all needed information to Antje.
   Antje records into the DBMS, a view like: View "CDS Aeropyrum pernix "

```
Column |       Type         |
-------+--------------------+
gene   | character varying |
note   | TEXT               |
```

Where the view definition is:

```
 SELECT Aeropyrum_pernix.gene,
 Aeropyrum_pernix.note FROM
executeAntje('org.PostgreSQL.Driver','jdbc:PostgreSQL://doro.limsi.fr:5432/ADNArc
h, 'select gene,note from CDS_Aquifex_ pernix ');
```

where “execute*Antje*”, is the function that retrieves the data in order to fill the view. We give here below a scenario (Figure 3) where this function is called.

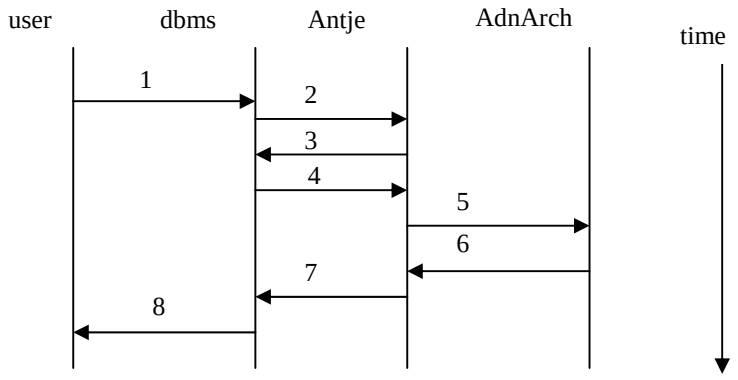

Figure 3. Scenario of the use of a remote view table through Antje

1. The user sends a query to the DBMS, like ‘select gene from CDS Aeropyrum pernix’.
2. The DBMS tries to retrieve information from the view, and it calls the function executeAntje with the query as a parameter.
3. and 4. Antje retrieves from the DBMS the maximum memory consumed by this query (in order to match the memory available by the JVM and the memory needed to answer the query).
5. and 6. Antje opens a JDBC connection to the database and it opens a cursor upon the asked query. Then it asks, via a fetch cursor instruction, some amount of row, in order to avoid JVM a “Out of memory” problem.
7. Antje sends back to the “CDS View” the result of the query, and the DBMS sends back the full filled view to the user.

These two previous scenarios illustrate how a user can easily integrate a remote database into Antje as a collection of “remote view”.
In addition to this advantage, we can enumerate some other benefits in the following:

- Through JDBC, the user can easily integrate databases using common and widely used database systems like Oracle, Postgres, MySQL
- We do not change the user part of our DBMS. Antje runs as a backend of the server. Therefore, the user can use existing technologies (PHP, Perl ...) to built standard user graphic interfaces.

Moreover, the time of life of this view could be choosing in function of the kind of storage of the remote view. Indeed if we store the remote view in:

- A DBMS view , the time of life of the view tables is the time of the query and the view will reside in memory
- A DBMS temporary table, the view will stand until the end of the user session.
- A DBMS table, the view will stand until the user drop the table.

The main question is the cost of an Antje's update when the remote database is updated. There are two main possibilities. As data are coming back from database and wrapped on the fly through in a SQL format, when a database like SWISSProt is updated:

- Either the update is made only upon the data (additional data, new proteinic, sequence…), and there is no update to make in the middleware. Indeed, Antje just transform a remote format into a SQL format.
- Either the update of the remote database is made upon the format of this database. In this case the concerned part of Antje must be update in order to map the new format.

4 ANTJEVIEWER

In order for *Antje* to fully assist users in integrating heterogeneous databases, a graphic user interface was designed for two reasons: a GUI is more pleasant to use than *SQL*; Biologists do not need to integrate all the data available but select those that will be of particular use to them. Moreover, some genomic databases can contain some non-validated data (annotations for example) that the user does not want to consider. Therefore, we implemented an integration graphical tool called *AntjeViewer*. This tool allows users to select/modify/add/delete the parts of a database they want to integrate. Some snapshots presenting *AntjeViewer* are given below.

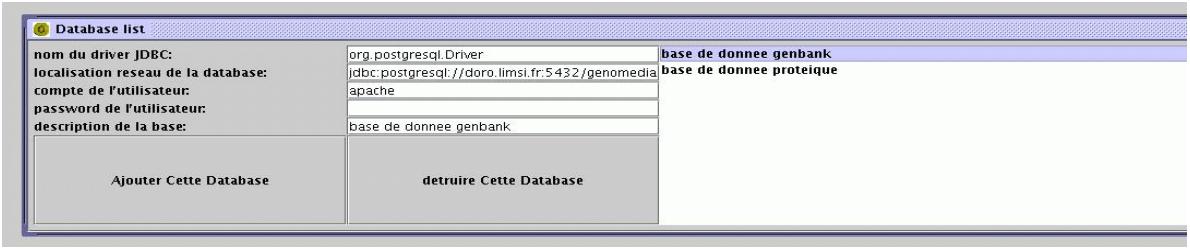

Figure 4. Database description and incorporation

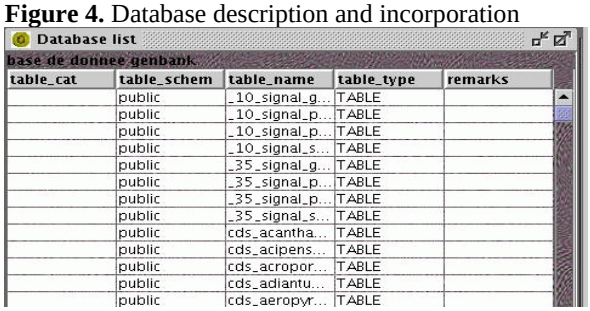

Figure 5. Tables from the database exploration

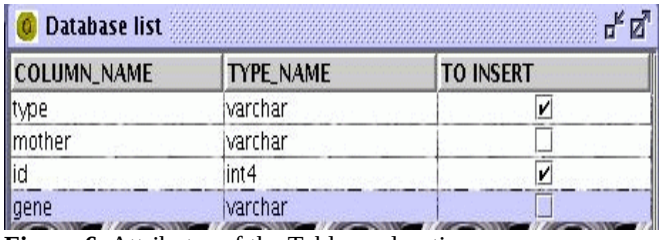

Figure 6. Attributes of the Table exploration

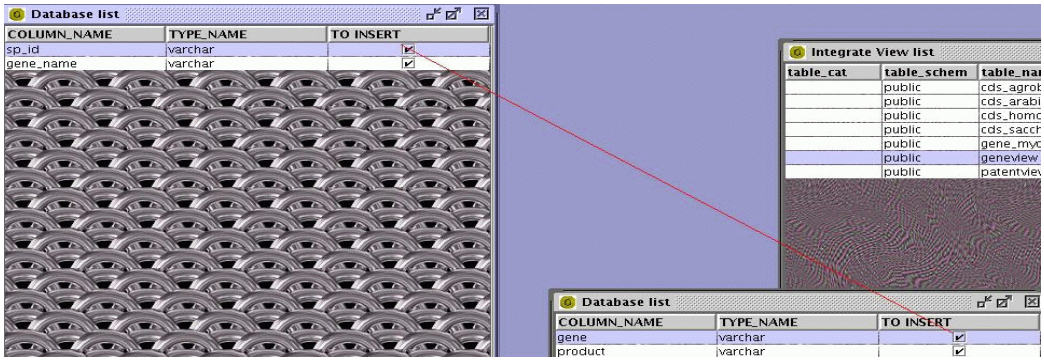

Figure 7.a user make a syntactic link between two views

Figure 4: On the first snapshot, we see that two well-known databases (GenBank and Swissprot) are ready for integration. In order to integrate any database, the user must give to Antje the network location of this database, a JDBC driver, and a user identity. AntjeViewer uses the JDBC driver in order to specify the nature of the underlining Database Management System. Indeed, a real DMBS like PostgreSQL supports a great many more features than a simple CSV file.

Figure 5: With this information, Antje allows the user to explore all the tables of the database (presently GenBank database). The structure of a table is computed from the Meta-information given by the JDBC driver of the data source. The main information retrieved from the driver is a description of the tables available in the given catalogue, and in particular the name of the tables. ADNArch relies upon a Postgreql, so AntjeViewer is able to retrieve the name and the style of the tables

Figure 6: AntjeViewer retrieves the description of table columns. With this description, AntjeViewer is able to show the column name and the "type "name of the column.

After selecting a table, the user can check the attribute he wants to integrate as a "remote view" (here he selects the "type" and the "id" attributes).

Figure 7: Now, the user can establish linkages between previously created views. As seen in this snapshot, Antje created internally for the user a new view using a link between the "sp_id" and "gene" attributes.

This capability of *AntjeViewer* to create relationship between views can be massively used in bioinformatics. For example, there are two major databases of proteins:

- *Swissprot*; a protein sequence database which strives to provide a high level of annotation
- *PIR*-PSD(GUMC, n.d); which is an integrated public bioinformatics resource that supports genomic and proteomic research and scientific studies.

After integrating these two databases into *Antje*, users are able to create new views based upon the both databases at the same time. In our example, a new protein database would be created to gather annotation from *Swissprot* and scientific studies from *PIR-PSD*.

After integrating different databases, a user might want to visualize the result of his new database. For that, *Antje* is able to create a package of multiple html files representing the database. These files are usually used with a server web as shown in Figures 8 and 9.

Using our GUI *AntjeViewer*, users can integrate in an easy manner heterogeneous data. This integration may be used through a simple *SQL* client or published through a Web server, like in Figure 8.

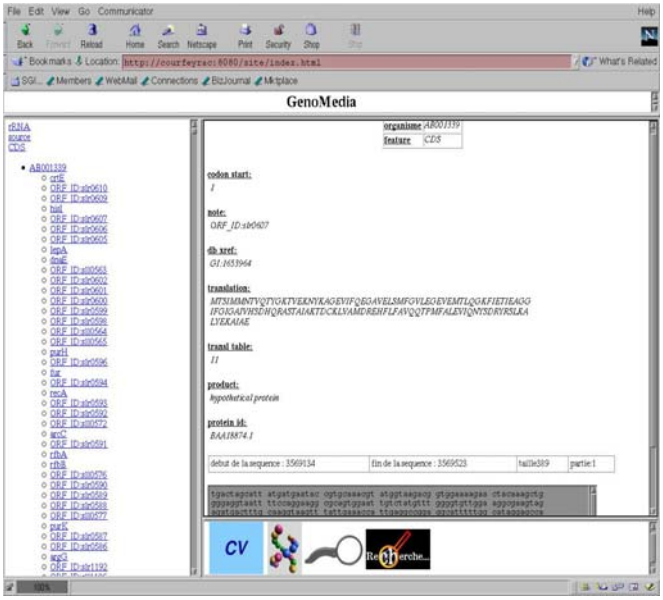

Figure 8: Antje is able to generate from integrated databases a complete web portal

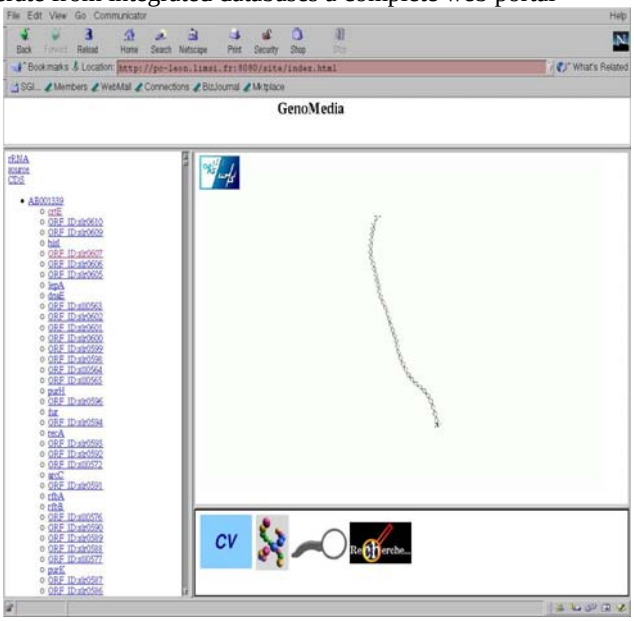

Figure 9: AND-Viewer
This Figure 9 represents the DNA structure of a gene. The tool ADN-Viewer (Hérisson & Gherbi, 01)provides such image.For *Lydia*, it is just a SQL function providing a Blob (standing for Binary Large Object)

5 CONCLUSIONS

In this paper, we presented a new approach that addresses the problem of integrating heterogeneous databases and software utilities within a generic distributed platform, *GenoMEDIA*. This software platform achieves two goals. The

first is to allow clients (users or tools) to simultaneously use several applications in order to produce query results in a transparent and user-friendly manner. The user does not have to learn a new interface, because he can use his favorite Internet browser or any external application connected to the platform. The second goal is to provide two integration middle wares both for database and service handling, through the Antje and Lydia concepts. This allows users to perform a considerable amount of non-systematic studies and analysis on huge amounts of data in a reasonable timeframe.
However, in order to analyze the extracted information, the main problem facing us is the user interface (UI). How can the biologist manage to manipulate and explore the information in a usable manner?
Multiple user interfaces provide different views of the same information and coordinate services available to users. To design and develop different views for complex systems, we can apply UI patterns that encapsulate best design practices for interactive systems. UI design patterns are prescriptive, and provide software designers with solutions for different UI facets, such as for navigation-specific problems. Multiple views, and insuring best UI practices for their design, can be particularly useful for bioinformatics systems – which contain vast amounts of information, and where navigation and exploration of data are important user issues. We are investigating new solutions in order to virtually explore both textual and factual genomic data. This approach is mainly based on the definition of a genomic data federator language, answering the requirements and specificities of genomic databases. This language takes into account the heterogeneity of both textual and factual data. New representation methods to view these data within an immersive environment are now being developed and first results have been obtained. Interfacing GenoMEDIA with a virtual reality system requires producing high quality graphical representation structures for real time visualization of biological information.

5 REFERENCES

Agha, G (1986) Actors: A Model of Concurrent Computation in Distributed Systems. *MIT Press*,Cambridge, MA,USA
GBorg (n.d.) PL/java. Retrieved 1 January, 2005 from the GBorg website: http://gborg.PostgreSQL.org/project/pljava/projdisplay.php

Burger, E., Link J. & Ritter, O. (1997) A Multi-Agent Architecture for the Integration of Genomic Information, *1st Int. Workshop on Intelligent Information Integration, in conjunction with KI'97*. Freiburg, Germany

Bryson, K, Luck, M, Joy, M, & Jones, D.T. (2000), Applying Agents to Bioinformatics in GeneWeaver, in Cooperative Information Agents IV, *Lecture Notes in Artificial Intelligence*, 1860, 60-71, Springer-Verlag

Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., & Stoeckert, C. (2001) K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources. *IBM Systems Journal*, *40*(2), pp. 512-531

Eckman, B., Kosky, A. & Laroco, L. (2001). Extending traditional query-based integration approaches for functional characterization of post-genomic data. *Bioinformatics, 17*(7), 579-670.

Embury, S.M. (1991), The Implementation of an Interface to Metadata in P/FDM, *University of Aberdeen Technical Report* (AUCS/TR9114),UK.

Goble,C.A., Stevens,R., Bechhofer, G. Ng, S., Paton, N.W., Baker, P.G., Peim, M.& Brass, A. (2001). Transparent Access to Multiple Bioinformatics Information Sources. *IBM Systems Journal, 40*(2), 532 - 552.

Gros, P-E, Ferey, N., Hérisson, J. & Gherbi R. (2004) GenoMEDIA, a Middleware Plate-form for Distributed genomic Information. *IEEE International Conference on Information and Communication Technologies: From Theory to Application*, Damascus, Syria

GUMC (n.d.) PIR-PSD database. Retrieved 1 January, 2005 from the GUMC website: http://pir.georgetown.edu/pirwww/aboutpir/aboutpir.html

Hart, K & Wong, L. (1994),CPL as a Query Language for Genetic Databases, Retrieved 1 January, 2005 from the Citeseer website: http://citeseer.ist.psu.edu/hart94cpl.html

Haas, L.M., Schwarz, P.M., Kodali, P., Kotlar, E. & Swope, J.E.R.W.C (2001) Discoverylink: A system for integrated access to life sciences data sources. *IBM Systems Journal 40*, 489-511

Hérisson, J. & Gherbi, R. (2001) Model-based Prediction of the 3D Trajectory of Huge DNA Sequences., *IEEE International Symposium on Bio-Informatics and Biomedical Engineering*. Washington DC,USA
Finin ,T., Fritzson, R., Mckay, D.& Mcentire, R. (1994) KQML as an Agent Communication, *3rd Int Conference on Information and Knowledge Management*. Gaithersburg, MD, USA

Kemp, G., Angelopoulos, N. & Gray, P. (2000). A schema-based approach to building a bioinformatics database federation. *In Proceedings of the IEEE International Symposium on Bioinformatics and Biomedical Engineering*. Washington, DC, USA

Lambrix, P. & Jakoniene, V. (2003). Towards transparent access to multiple biological databanks. *In Proc. 1st Asia-Pacific Bioinformatics Conference*. Adelaide, Australia.

Microsoft (n.d) Microsoft® Open Database Connectivity. Retrieved 1 January, 2005 from MSDN website: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/dasdkodbcoverview.asp

Mork, P., Halevy, A. & Tarczy-Hornoch , P. (2001) A Model for Data Integration Systems of Biomedical Data Applied to Online Genetic Databases. *Proceedings of the Symposium of the American Medical Informatics Association*. Seattle, WA, USA

NCBI (n.d.) GenBank Database. Retrieved 1 January, 2005 from the NCBI website: http://www.ncbi.nlm.nih.gov
ObjectWeb (n.d.) C-JDBC: Clustered JDBC. Retrieved 1 January, 2005 from the ObjectWeb website: http://c-jdbc.objectweb.org/

Oracle (2004) Oracle® Database Administrator's Guide 10g Release 1 (10.1) Distributed Database Concepts. Retrieved 1 January, 2005 from the Oracle website: http://otn.oracle.com/pls/db10g/portal.portal_demo3?selected=1

Risch,T. (2004*) Mediators for Querying Heterogeneous Data, The Practical Handbook of Internet Computing*. Baton Rouge, US: Chapman & Hall/CRC

Risch,T., Josifovski,,V & Katchaounov, T (2003): Functional Data Integration in a Distributed Mediator System, in: Functional *Approach to Data Management - Modeling, Analyzing and Integrating Heterogeneous Data*. Berlin, Germany: Springer.

SGD (n.d.) Homepage of the Saccharomyces Genome Database. Available from: http://www.yeastgenome.org/

SIB/EBI (n.d.) Homepage of Swissprot Protein knowledgebase. Available from: http://www.expasy.org/sprot/

Stevens, R., Goble, C., Paton, N. W., Bechhofer, S., Ng,G.,Baker,P. & Brass. A.(2003). Complex Query Formulation over Diverse Information Sources in TAMBIS. In Lacroix, Z. & Critchlow, T. (Eds.), *Bioinformatics:* Managing Scientific Data. San Francisco, US: Morgan Kaufmann.

Sun (n.d.) J2EE JDBC Technology. Retrieved 1 January, 2005 from the Sun website: http://java.sun.com/products/jdbc/

Williams, N. (1997) Bioinformatics: How to Get Databases Talking the Same Language. *Science 275*(5298), 301-302.