Hybrid Integration of Molecular-biological Annotation Data

Toralf Kirsten†, Hong-Hai Do†, Christine Körner*, Erhard Rahm†‡

†Interdisciplinary Centre for Bioinformatics, University of Leipzig
http://www.izbi.de, {kirsten, do}@izbi.uni-leipzig.de

‡Dept. of Computer Science, University of Leipzig
http://dbs.uni-leipzig.de, rahm@informatik.uni-leipzig.de

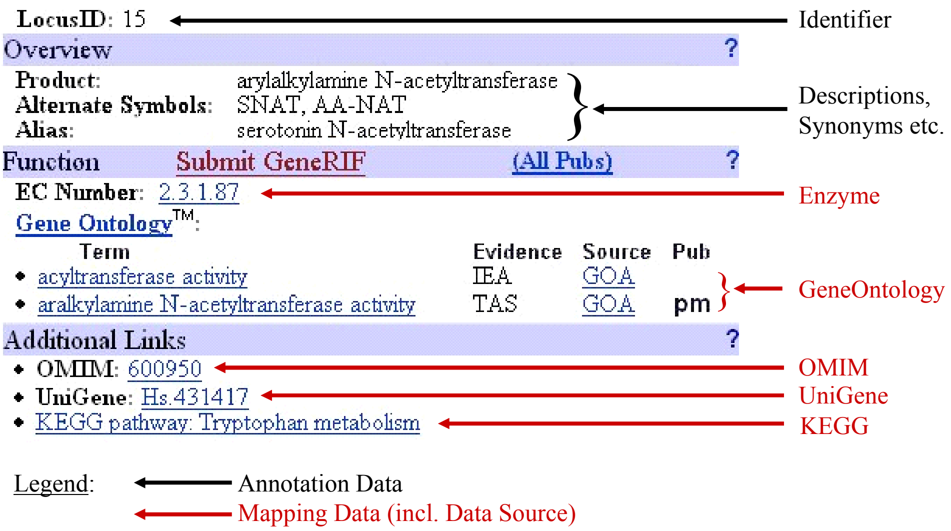
*Fraunhofer Institute, St. Augustin
http://www.ais.fraunshofer.de, christine.koerner@ais.fraunhofer.de

Abstract: We present a new approach to integrate annotation data from public sources for the expression analysis of genes and proteins. Expression data is materialized in a data warehouse supporting high performance for data-intensive analysis tasks. On the other hand, annotation data is integrated virtually according to analysis needs. Our virtual integration utilizes the commercial product SRS (Sequence Retrieval System) of LION bioscience. To couple the data warehouse and SRS, we implemented a query mediator exploiting correspondences between molecular-biological objects explicitly captured from public data sources. This hybrid integration approach has been implemented for a large gene expression warehouse and supports functional analysis using annotation data from GeneOntology, Locuslink and Ensembl. The paper motivates the chosen approach, details the integration concept and implementation, and provides results of preliminary performance tests.

1 Introduction

After the complete genomes of various organisms have been sequenced, the focus of genomic research has shifted to studying and comparing the functions of genes and their products. The knowledge about molecular-biological objects, such as genes, proteins, pathways etc., is continuously collected, curated and made available in hundreds of publicly accessible data sources [Ga04]. The high number of the data sources and their heterogeneity renders the integration of molecular-biological annotation data for functional analysis a major challenge in the bioinformatics domain.

To illustrate the data we have to deal with, Figure 1 shows a sample annotation for a gene uniquely identified by accession number 15 in the public source Locuslink [PM02]. The entry comprises different descriptions, which we group into *annotation* and *mapping data*. Annotation data consists of source-specific attributes, such as *Product* and *Alternate Symbols*. In contrast, mapping data refers to inter-related objects in other sources and is typically represented by web links. The objects are identified by their source-specific accession ids, for example, gene locus 15 in LocusLink, gene cluster Hs.431417 in UniGene [Wh03], or enzyme 2.3.1.87 in Enzyme [Ba00]. We denote the set of correspondences between objects of two data sources as a *mapping*. Inter-relating objects by mappings allows combining the annotation knowledge from multiple sources for analy-



LocusID: 15

Overview ?

Product: arylalkylamine N-acetyltransferase
Alternate Symbols: SNAT, AA-NAT
Alias: serotonin N-acetyltransferase

Function Submit GeneRIF (All Pubs) ?

EC Number: 2.3.1.87

Gene Ontology™:

Term	Evidence	Source	Pub
acyltransferase activity	IEA	GOA	
aralkylamine N-acetyltransferase activity	TAS	GOA	pm

Additional Links ?

- OMIM: 600950
- UniGene: Hs.431417
- KEGG pathway: Tryptophan metabolism

Figure 1: Annotation and Mapping Data in Locuslink

sis. In the example, the analysis of LocusLink genes can be enriched by annotations of the referenced GeneOntology [As00] or UniGene objects.

Establishing and browsing web links represent a first step to integrating different sources, which, due to its simplicity, is widely used. Unfortunately, web links only support interactive analysis for single objects at a time, but not automatic analysis for large sets of objects. Such a set-oriented analysis capability is especially needed for high-throughput expression analysis. In this paper we present a new approach to integrate annotation data from public sources for expression analysis. Large amounts of expression data generated by microarray experiments are physically stored together with experimental descriptions in a data warehouse to support performance-critical analysis tasks. Annotation data, on the other hand, is virtually integrated by a query mediator which utilizes the commercial product SRS (Sequence Retrieval System) to access annotation data of public data sources.

The key aspects of our approach are:

- We combine a materialized and a virtual data integration to exploit their advantages in a new hybrid approach. On the one hand, the data warehouse offers high performance for complex analysis tasks on large amounts of expression data. On the other hand, up-to-date annotation data can be retrieved for analysis when needed.
- Public data sources are uniformly integrated and accessed through the widely accepted SRS tool, which offers wrapper interfaces to a large number of molecular-biological data sources, including flat files and relational databases. Hence, we avoid the re-implementation of import functions and can easily add sources supported by SRS.
- We explicitly extract mapping data from the data sources and store them in a separate database, the so-called *mapping database*. This separation allows us to determine dif-

ferent join paths between two sources to relate their objects with each other and to pre-compute them for good query performance.

- The approach has been implemented as an extension to the GeWare platform (gene expression warehouse) [KDR03, KDR04] and integrates several public data sources to support the expression analysis. The web interface is accessible under http://www.izbi.de/GEWARE. Performance tests have shown the practicability of our approach.

The rest of the paper is organized as follows. Section 2 discusses related integration approaches. Section 3 describes two main analysis scenarios and their integration requirements. Section 4 gives an overview of our integration concept. Section 5 and 6 describe the central components and their function in more detail, namely the mapping database and the query mediator, respectively. Section 7 presents the results of selected performance tests. Section 8 concludes the paper.

2 Related Work

An overview of representative approaches used for data integration in the bioinformatics domain is given by [LC03], [HK04] and [St03]. Previous solutions mostly follow either a materialized or a virtual integration approach. The former approach physically stores the data in a central database or data warehouse, which can offer high performance for data-intensive analysis tasks. The latter approach typically uses a mediator to perform data access at run time and provide the most current data. In the following we discuss some of these approaches and how they differ from our approach.

Similar to our approach the mediator-based systems DiscoveryLink [Ha01], Kleisli [CCW03, Wo98] and SRS [EHB03, ZD02] do not pursue a (laborious) semantic integration of all data sources by constructing an application-specific global schema. They use a simple schema comprising of the sources and their attributes, which makes it relatively easy to add new data sources. Currently, Kleisli offers interfaces to more than 60 public sources and SRS provides wrappers to more than 700 data sources. Typically, complete copies of data sources are maintained locally and periodically updated for availability and performance reasons. As the price for flexibility, DiscoveryLink and Kleisli leave the task of semantically integration to the responsibility of the user. In particular, the user has to explicitly specify join conditions in queries to relate objects/data from different sources with each other.

SRS and our approach address this problem by capturing and utilizing existing mappings, i.e. correspondences at instance level. SRS maintains indices on these mappings and thus can achieve high query performance. However, SRS only uses the shortest path between two sources as the join path to relate their objects with each other. This represents a restriction as the user may have another preference. Moreover, alternate paths may yield better results than the shortest path [La04]. Therefore, our approach aims at a more flexible and efficient computation of join operations by a) supporting multiple alternative paths and b) pre-computing joins between the sources to a previously determined central source so that join paths with a maximal length of 2 are possible via the central source.

COLUMBA [Ro04] physically integrates protein annotations from several sources into a local database. Source data is imported mainly in its original schema to reduce the effort required for schema integration and data import as much as possible. The source schemas are connected using a mapping to a previously selected central source, the Protein Data Bank (PDB). We also use this technique to construct our mapping database. In contrast to COLUMBA, which only allows a single mapping between a source and the central source, we support multiple mappings, each of which may be pre-computed using a different join path.

ALADIN (Almost Automatic Data Integration) [LN05] generalizes the COLUMBA approach for integrating different kinds of annotation data. A main extension is in the automatic analysis of instance data to detect object associations and duplications. This work is orthogonal to ours and helps to establish new mappings. We are currently focusing on utilizing existing mappings and their compositions.

Our GenMapper [DR04] tool also follows a physical integration of annotation data by using a generic schema called GAM (Generic Annotation Model). The GAM stores both (intra-) associations between objects of the same source and (inter-) association between objects of different sources. High-level operators are used to generate annotation views for different analysis purposes. However, GAM focuses specifically on mapping data and cannot handle data with complex structures, such as geometric data of protein folding structures and genomic sequences. Our hybrid approach uses GenMapper to precompute mappings for different join paths, which are then imported into the mapping database.

3 Integration Requirements for Analysis

Figure 2 shows two common analysis scenarios in the bioinformatics domain, namely expression and annotation analysis. Expression analysis detects and compares the gene and protein activity under different circumstances, such as in normal and diseased tissues. The main goal is to identify groups of genes or proteins, showing consistently similar or different expression patterns. For example, genes, which are highly active in tumor cells but not in normal cells, could be responsible for the uncontrolled proliferation of the tumor cells. Analyzing the annotations of those genes can reveal the similarities and differences in their currently known functions and infer new gene functions. On the other side, searching in annotation data allows to generally identify genes or proteins with similar functions. This gene / protein groups can be used as input for expression analysis to get insights about their expression behavior.

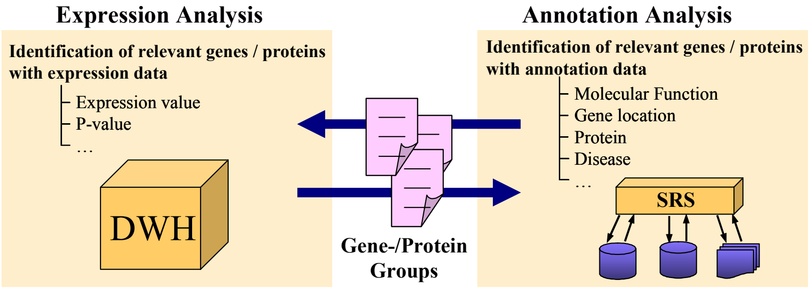


Figure 2: Different Analysis Scenarios

The usage of annotation data in these scenarios, leads to the following requirements for the integration task.

- *Flexibility and adaptability:* Public data sources are constantly extended and modified at the instance level and also underlie changes at the schema level. Hence, it is important to access current data. Furthermore, the high number of relevant sources presupposes a flexible solution to easily "plug in" a new source when needed. Both observations motivate a virtual integration of annotation data and favor the utilization of a powerful infrastructure such as SRS.
- *Inter-source mappings:* Depending on the research focus of the user, different kinds of annotations may be required for different types of objects. This presupposes the ability to flexibly associate annotations with objects from different sources. For instance, it should be possible to determine functions, e.g. as expressed in GeneOntology terms, for genes in Locuslink, UniGene, NetAffx etc. In addition, filters, such as exact and pattern matching, and their combinations are necessary to identify interesting objects. Finally, alternative join paths should be supported due to the high degree of interconnectivity between sources.
- *Data quality:* Annotations from different sources may largely vary in data quality, e.g. due to different update frequencies and algorithms to calculate object homology. To support user acceptance it is necessary to document how the data has been integrated, e.g. from which source and using which join paths, so that the user can judge its quality.
- *Performance:* Query performance is obviously of key importance for the user acceptance in interactive analysis. Therefore, the physical integration using a data warehouse is recommended for large amounts of expression data. Mediator-based query processing should also be optimized, especially the execution of resource-intensive join operations to relate objects from different public sources. Hence, advanced techniques, such as indexing or pre-computation and materialization of common join paths should be applied to improve query time.

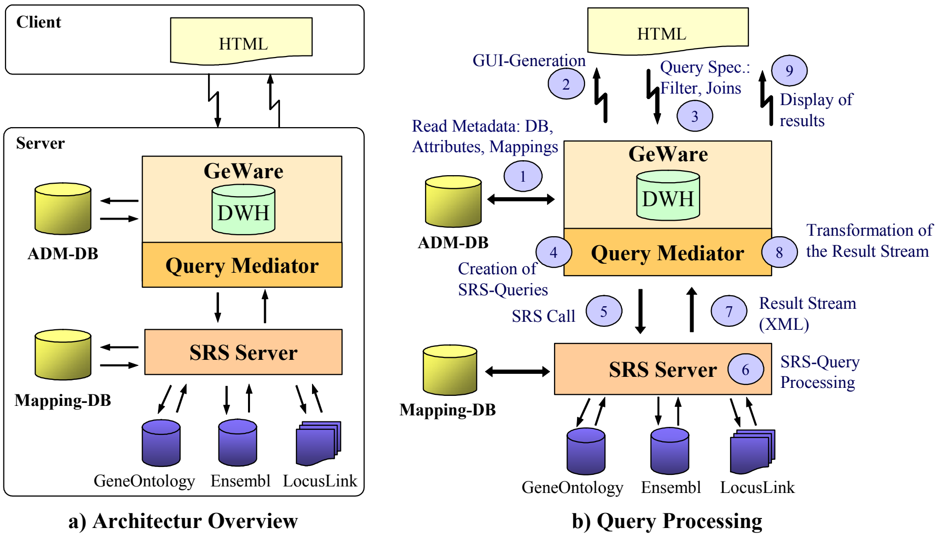
4 Integration Architecture

4.1 Overview

According to the integration requirements described in the last section, we have designed and implemented a hybrid integration system. Its architecture is illustrated in Figure 3a comprising the following components:

- *GeWare*, a data warehouse supporting expression analysis, is used as integration and test platform for our approach.
- *SRS* is used to query and retrieve annotations from the relevant public sources. Currently the following sources are integrated: the widely used GeneOntology as well as the gene sources Locuslink, Ensembl [Bi04, Po04], UniGene, and the vendor-based source NetAffx [Ch04] providing annotations for the genes of Affymetrix microarrays.

- Our *query mediator* acts as the interface between GeWare and SRS. It transforms user-specified queries into SRS-specific queries which are then forwarded to SRS for execution. Finally, the query mediator combines the results delivered by SRS, performs necessary transformations, and visualizes them on the user web interface.
- The *mapping database* stores pre-computed mappings between the sources. For each source, the mapping database maintains a mapping table storing all correspondences between the source and a pre-selected central source. This star-like schema makes it possible to efficiently perform join operations through the central source.
- The *ADM database* serves administration purposes and stores metadata about the integrated sources, such as their names, attributes and the information about the available mappings (mapping names, and join paths used to compute them). We utilize this metadata to automatically generate the web interface for query formulation.



a) Architectur Overview **b) Query Processing**

Figure 3: Integration approach and corresponding components

The next two subsections describe the interaction between the components in two main processes, the integration of data sources and query processing, respectively. In Section 5 and Section 6, we focus on the issues of metadata management within the mapping and ADM database, and of query processing in the query mediator, respectively.

4.2 Data Source Integration

The comprehensive wrapper library provided by SRS supports numerous data sources available in the bioinformatics domain and allows us to easily add new sources. In particular, we use these wrappers to integrate the flat file-based source Locuslink and two relational databases, Ensembl and GeneOntology. To achieve good performance for interactive queries, we maintain local copies of these data sources for integration in SRS. The ADM database holds metadata about the sources, especially the names of the sources and their attributes.

In our approach, the data sources are organized in a star-like schema supporting efficient join queries. For each object type, one of the sources is chosen as the central source, to which mappings from all other sources of this type are pre-computed. For example, Locuslink is a reference data source for gene annotations. Its identifier, the Locuslink accession, is linked in many other sources and often used for citations in scientific publications. Hence, we choose Locuslink as the central gene source in our current implementation to support gene expression analysis. To construct the mapping database, we import the mappings from Locuslink to all other sources, in particular to UniGene, Ensembl, NetAffx and GeneOntology, which are pre-computed and provided by GenMapper [DR04]. To link a source with the central source, alternative mappings can be computed using different join paths and imported. Each mapping is then registered in the mapping database with the path employed to compute them (see Section 5).

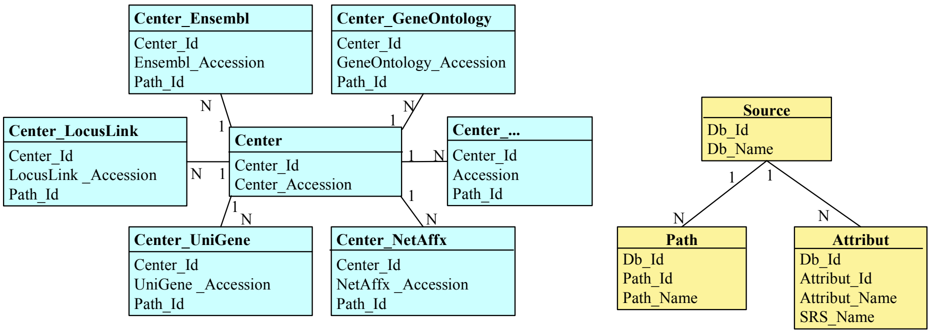
4.3 Query Processing

Figure 3b shows the general workflow of query processing in our system (see Section 6 for more details). The workflow starts with querying metadata about the available sources, attributes and mappings from the ADM database (Step 1). Using this metadata, the web interface is automatically generated (Step 2). Then, the user can formulate the query by selecting the data sources and relevant attributes, and specifying filter conditions and join paths (Step 3). The query mediator interprets the user query and generates a query plan, which consists of one or multiple SRS-specific queries (Step 4). The query plan is passed to the SRS server for execution (Step 5 and 6). While subqueries for selection and projection are performed within the corresponding sources, SRS uses the mapping database to perform join operations. The query result is then returned as one or multiple XML stream (Step 7). The query mediator parses the streams to extract the relevant data (Step 8), which is then prepared in different formats, e.g. HTML for displaying on web browser, and CSV for download (Step 9).

5 Metadata Management

5.1 The Mapping Database

Previous integration systems, such as SRS and GenMapper, determine corresponding objects between two sources using a multi-way join operation along the shortest, automatically determined path connecting them with each other. This approach leads to several problems. First, the shortest path may not always be the best one for joining two particular sources. Other (probably longer) paths may deliver better data, e.g., if the involved sources are updated more frequently than those in the shortest path. Second, the composition of many mappings can lead to performance problems, even for the shortest paths, if they are to be evaluated at run time. One solution to improve query time is to pre-compute and materialize all possible paths in the database. However, this would lead to an enormous amount of mappings and object correspondences (complexity $O(n^2)$ with n sources) which is fairly impractical to manage and update. We address these problems on the one hand by supporting several alternative paths, which can be selected by the users according to their preference or analysis needs. On the other hand, instead of pre-computing join paths between all sources, we identify a central source and pre-compute only the join paths between the remaining sources to the central source, through which the join operations are performed at run-time.



a) Schema of the Mapping-Database **b) Schema of the ADM-Database**

Figure 4: Metadata Management in ADM and Mapping Database

Similarly to COLUMBA, the data sources are connected in a star-like (multidimensional) schema in our approach. In contrast to COLUMBA, we maintain the mappings in a separate database for optimized join processing and support alternative mappings between a source and the central source. Figure 4a shows the database schema of the mapping database. There is a center table for the central source and a mapping table for each additional data source. All objects of the central source are uniquely identified by the key *Center_ID.* These ids are used as foreign keys in the mapping tables to represent the object relationships at the instance level. Note that a mapping table is used to maintain all mappings of different paths between the respective source and the central source. Each path is identified by a *Path_Id* identifier referring to a specific path which has been used to pre-compute the mappings. Every supported path is described in the ADM database including metadata such as its name and the involved intermediate sources (see Subsection 5.2).

For example, assume we want to relate genes from UniGene with annotations from Ensembl. Neither UniGene nor Ensembl maintain a direct mapping to each other. Hence it is necessary to relate their objects through common objects in other sources. By analyzing the set of available mappings, we could identify *UniGene-Locuslink-NetAffx-Ensembl* as a possible join path. Without pre-computation, three mappings, each between two neighbor sources in the path, have to be retrieved and successively composed. In our implementation, the mapping table *Center_UniGene* provides a direct mapping *Locuslink-UniGene*. The mapping table *Center_Ensembl* contains the mapping *Locuslink-Ensembl*, which has been previously pre-computed using the path *Locuslink-NetAffx-Ensembl*. Hence, we need only to join these two mappings.

The number of the mappings to be pre-computed and materialized in the mapping database is linear with the number of the sources to be integrated. The support for alternative join paths does not affect the linear complexity (k*n mappings with n sources with k alternative mappings on average per source). New annotation sources can easily be added by creating new mapping tables to hold the corresponding mapping data. This does not affect the run-time complexity because the join operations within the mapping database never involve more than 2 mappings (*source-center-source*). Mapping tables for sources that are no longer required can be removed. Storing mapping data for each source in separate mapping tables simplifies the data update task. In particular, a mapping can be easily updated by deleting it and inserting the new one. The local copies of annotation sources can be independently replaced in SRS by a new version.

The prerequisite to integrate a new source is that there is at least one mapping path between it and the central source or that such a path can be constructed by joining existing paths. Therefore, the selection of the central source plays an important role in this integration approach. Quality criteria, such as update frequency and acceptance by the users, should be considered. Furthermore, if the source already provides direct mappings to many other sources, these mappings can be taken to quickly construct the mapping database. For example, Locuslink and SwissProt represent reference sources for gene and protein annotations, respectively, and maintain a large number of mappings to other (smaller) sources. Hence, they are good candidates for the central source to integrate annotations for gene and protein analysis.

5.2 The ADM Database

Figure 4b shows a portion of the ADM database schema holding metadata about the integrated data sources. The *Source* table records a unique source identifier (*Db_Id*) and the source names. The available attributes of a source are stored in the table *Attribute*, which also contains their SRS-specific names used to translate the user query into a SRS-specific query. All join paths, for which a mapping is materialized in the mapping database, are stored in the *Path* table. The path name concatenates all names of sources that have participated on the join path. Hence, the user can easily differentiate between alternative mappings and identify one for her need. Currently, we import this data partly manually and partly automatically by means of specific database scripts, which extract metadata from the corresponding sources. Subsection 6.2 discusses the process of using this metadata to automatically generate web interfaces for query specification.

6 Query Processing within the Query Mediator

6.1 Query Types

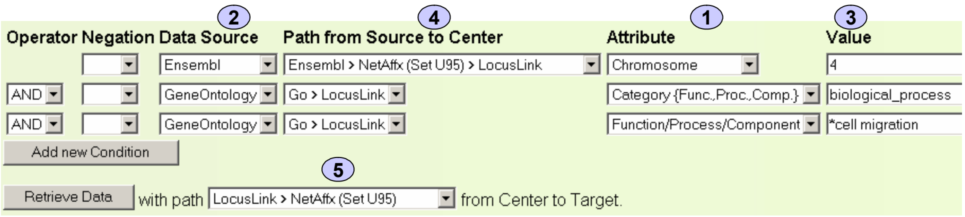
The query mediator supports two kinds of queries, projection and selection queries, according to the specific requirements of expression and annotation analysis, respectively (see Subsection 3.1):

- *Projection queries* support expression analysis and return a uniform view with user-specified annotation attributes for a given gene group. In a query, the attributes may stem from different sources.
- The goal of *selection queries* is to identify sets of genes showing some common properties. This can be done by applying filter conditions on the corresponding annotation attributes. The gene sets can then be used in expression analysis to compare their expression behavior.

These two query types differ from each other in their input and output data. Projection queries need a gene group as input while selection queries produce a gene group as output. However, they are processed in the same way by associating genes with annotation attributes from the selected sources.

6.2 Query Formulation

The query web interface is generated automatically using the source-specific metadata stored in the ADM database. The user formulates queries on the web interface by selecting relevant attributes (projection queries) and specifying filter conditions (selection queries). Figure 5 shows an example of a selection query to identify all genes, which are located on chromosome *4* and are associated with the biological process *cell migration.*



Operator	Negation	Data Source	Path from Source to Center	Attribute	Value
		Ensembl	Ensembl > NetAffx (Set U95) > LocusLink	Chromosome	4
AND		GeneOntology	Go > LocusLink	Category {Func.,Proc.,Comp.}	biological_process
AND		GeneOntology	Go > LocusLink	Function/Process/Component	*cell migration

Retrieve Data with path LocusLink > NetAffx (Set U95) from Center to Target.

Figure 5: Query Formulation on the automatically generated web interface

A query may consider attributes (1) stemming from different sources (2). For each attribute, a filter condition (3) can be specified allowing for exact or pattern matching queries. In our example of Figure 5, the asterisk in front of the filter value *"cell migration*" characterizes a similarity search; the other two values are used for exact search. Furthermore, the user has to specify the mapping to connect the source of the selected attribute to the central source by selecting a join path (4). Multiple conditions can be added and combined using the logical operators OR, AND and NOT whereby OR has the lowest and NOT the highest priority in the query evaluation process. Finally, according to the type of genes to be returned, a mapping between the central source and the target source is to be selected (5).

While SRS only supports to filter attributes of the source from which the data is to be retrieved, our implementation supports the combination of attributes from different sources within a selection query. Moreover, our implementation provides the possibility to combine attributes of different sources (projection) within the same query, which is also currently not directly supported by SRS.

6.3 Generation of Query Plans

From the user specifications on the web interface (see Figure 5), the query mediator generates a SRS-specific query for later execution by the SRS server. This process is performed in three steps, *Block formation* to split the queries into blocks according to the logical operators, *Grouping of source-specific attributes* to determine and group subqueries on attributes belong to the same source to be executed together, and *Assembling SRS query* to generate the final query in SRS-specific syntax and terms. Figure 6 illustrates these steps using the example query from Section 6.2. We discuss the single steps in the following:

1. *Block formation:* First, the filter conditions of a selection query are divided by the logical operator OR into single blocks. Each block contains either one or multiple filter conditions connected with each other by the AND operator. Our query example from Section 6.1 does not contain the OR operator. Hence, there is only one block constructed (see Figure 6, Step 1) holding all three filter conditions. This step is not

necessary for projection queries, which do not require filter conditions and build a view for all specified attributes.

2. *Grouping of source-specific attributes:* Within each block obtained from Step 1, the attributes and filter conditions are grouped according to their data source and the mappings to the central source. Each group of attributes and filter conditions concerning the same source and mapping will be valuated together in a subquery. Figure 6, Step 2, shows two identified groups *a* and *b* for the attributes *Category* and *Process* of GeneOntology, and the attribute *Chromosome of Ensembl*, respectively.

3. *Assembling SRS query:* The source and attribute names are replaced by SRS-internal names, which are previously captured and stored in the ADM database. The names of the selected mappings, i.e. the paths, are substituted by their identifiers in the mapping database. For example, Figure 6, Step 3, shows in Line 3 the second and third filter conditions specified on the web interface. The source *GeneOntology* and the attributes *Category* and *Process* are replaced by the internal names *GoTerm*, *typ* and *tna*, respectively. SRS is then invoked by calling its interpreter "*getz*" (Line 1).

1. Step: Block formation

Block	*Path*	*Sourcee*	*Attribute*	*Filter value*
1	Ensembl>NetAffx(Set U95)>LocusLink	Ensembl	Chromosome	4
1	GeneOntology>LocusLink	GeneOntology	Category	biological_process
1	GeneOntology>LocusLink	GeneOntology	Process	*cell migration

2. Step: Grouping of source-specific attributes

Block	*Group*	*Path*	*Source*	*Attribute*	*Filter value*
1	a	Ensembl>NetAffx(Set U95)>LocusLink	Ensembl	Chromosome	4
1	b	GeneOntology>LocusLink	GeneOntology	Category	biological_process
1	b	GeneOntology>LocusLink	GeneOntology	Process	*cell migration

3. Step: SRS-Query assembling

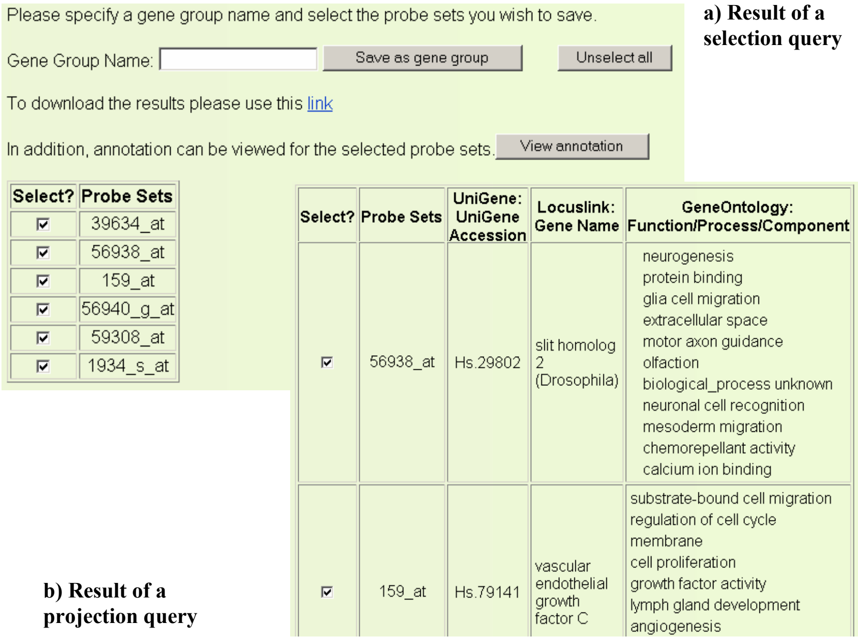
```
1 getz -vf "accession" "([Mapping-pid:5]
2   < (Center < ([Mapping-pid:2]<([EnsemblGene-cnm:4]))
3   < ([Mapping-pid:1]<([GoTerm-typ: biological_process] & [GoTerm-tna:*cell migration]))))
```

Figure 6: Steps for creating the Query Plan

From the SRS-specific query in Figure 6, Step 3, we can see, that the objects of *EnsemblGene* and *GoTerm* are first identified by applying the corresponding filters (Lines 2 and 3) and then uniformly mapped to the central identifier *Center_Id* (Line 2) using the mapping ids 1 and 2, respectively. The resulting central identifiers are in turn mapped to the target data source NetAffx using the mapping with id 5, (Line 1). The result of the query consists in a set of NetAffx accessions indicating the corresponding genes.

6.4 Extraction and Result Transformation

According to the complexity of the user query specified on the web interface, one or multiple SRS-specific queries are generated and executed. For each such query (e.g. shown in Figure 6, Step 3), SRS returns the result as a XML stream. The stream is then parsed by the query mediator to extract the relevant data. The query mediator then assembles the extracted data of all streams into an internal data structure for later visualization or export. It is also able to perform compensation routines for those functions, which are not yet supported in some DBMS, such as intersection in MySQL, and has not been considered in SRS. A gene group as the result of a selection query can be used as input for a projection query to obtain other annotations for the genes of interest. On the other side, from the result of a projection query, the user can also identify the relevant genes and save them as a new gene group for further queries. The exchange of gene groups between the queries allows us to perform successive refinement for an initially large set of genes.



Please specify a gene group name and select the probe sets you wish to save.

Gene Group Name:

To download the results please use this link

In addition, annotation can be viewed for the selected probe sets.

Select?	Probe Sets
☑	39634_at
☑	56938_at
☑	159_at
☑	56940_g_at
☑	59308_at
☑	1934_s_at

a) Result of a selection query

Select?	Probe Sets	UniGene: UniGene Accession	Locuslink: Gene Name	GeneOntology: Function/Process/Component
☑	56938_at	Hs.29802	slit homolog 2 (Drosophila)	neurogenesis protein binding glia cell migration extracellular space motor axon guidance olfaction biological_process unknown neuronal cell recognition mesoderm migration chemorepellant activity calcium ion binding
☑	159_at	Hs.79141	vascular endothelial growth factor C	substrate-bound cell migration regulation of cell cycle membrane cell proliferation growth factor activity lymph gland development angiogenesis

b) Result of a projection query

Figure 7: Results of Projection and Selection Queries

Figure 7a shows a portion of the result for the example query in Section 6.2. In particular, it contains a set of NetAffx genes which are localized on chromosome *4* and known to have a function in the biological process *cell migration*. The genes are stored in a gene group, for which a projection query is performed to obtain an annotation view as shown in Figure 7b. In particular, the UniGene accession, the Locuslink gene name, and the all functional annotations of GeneOntology are included in the view, based on which the user can further judge the relevance of the genes.

7 Performance Analysis

For testing the integration approach and measuring the performance we used an Intel-based platform with the following hard- and software configuration.

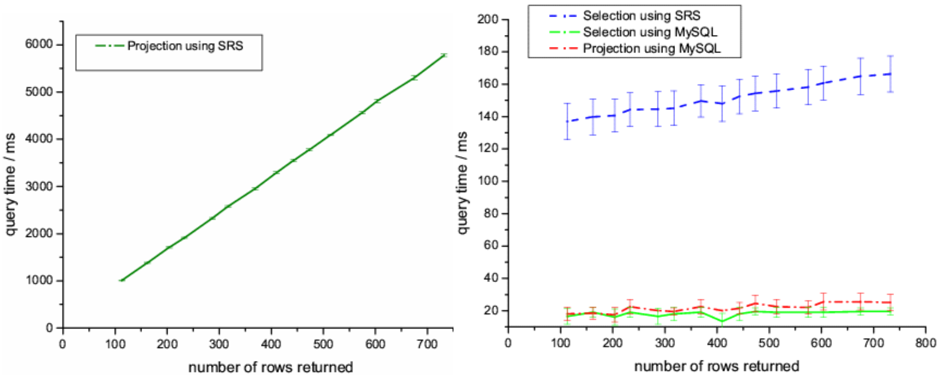
Hardware:		**Software:**	
CPU:	4 x Intel Xeon 2.5 GHz	**OS:**	Linux, Fedora 2.4.22
RAM:	8 GB	**DBMS:**	IBM DB2 8.1.0
			MySQL, Version 4.0.17-max
		SRS-Server:	SRS Relational 7.3.1 for Linux
		Java:	Java 2 SUN Platform,
			Standard Edition, Version 1.4.2

The data warehouse GeWare and the ADM and mapping databases are managed by the relational database system DB2 of IBM. The query mediator and all GeWare functions are written in Java. SRS was installed on the same machine together with the locally replicated sources Locuslink (file-based), Ensembl (MySQL) and GeneOntology (MySQL).

We focus on two performance tests investigating the query execution times for different result set sizes. To determine the time overhead induced by SRS, we examine the difference in query time between using SRS to query a relational database and accessing the database directly, i.e. without SRS[1]. We measure the elapsed time of 15 different queries only involving the Ensembl database in MySQL. Each query uses a different filter condition for the attribute *des* (gene description) to return result sets of different size. The queries are repeated 20 times in order to determine the average and standard deviation (shown as error bar) of the elapsed time.

Due to the large difference in query times, we first show the result for projection queries using SRS in Figure 8a and the remaining results, i.e. for selection queries using SRS and for both selection and projection queries directly accessing MySQL in Figure 8b. Please recall that selection queries only return the accessions of the identified objects, while projection queries return the objects together with the retrieved annotations.

W.r.t to the increasing size of the result set, we observe a significant linear increase in query time for projection queries in SRS (Figure 8a). For selection queries, SRS also requires linear time w.r.t. to the size of the result set (Figure 8b). However, selection queries can be performed much faster than projection queries in SRS. On the other side, we observe almost negligible query time when directly accessing MySQL. For larger result sets, the query time remains almost constant. This leads to the conclusion that SRS produces much time overhead in processing the data obtained from a relational source.



a) Query Time of SRS Projection Queries **b) Query Time SRS/MySQL Queries**

Figure 8: Performance of projection and selection queries using SRS and MySQL

[1] To execute a query, SRS in turn creates a query plan consisting of SQL statements to access the corresponding relational database. We use these SQL statements to perform the test in the latter case, i.e. accessing the database directly.

The second test determines the query time for the single steps in the execution of a query involving SRS. For this purpose, we define 11 different queries uniformly involving Ensemble, NetAffx, and the center source Locuslink. They all employ the mappings *Ensembl-Locuslink* and *Locuslink-NetAffx* in order to identify NetAffx genes having a particular pattern in the attribute *des* (gene description) of Ensembl. Figure 9 shows the result of this test. Each query is again repeated 20 times in order to determine the average and standard deviation (shown as error bar) of the elapsed time.

The measured values for each step subsume the elapsed time of all its previous steps. For example, Step 2 performing a mapping between Ensembl and LocusLink subsumes Step 1 to select relevant data from Ensembl. The time of the last step, i.e. Step 4 mapping of the identified Locuslink genes to the required NetAffx genes, represent the entire elapsed time of the query. Overall, the query time increase linear with the amount of the data to be retrieved and is acceptable for even large amount of result data. The first step, selection from Ensembl, performs fastest and the elapsed time remains relatively constant for different size of the result set. As we can see in Figure 9, querying the mapping database (Step 2-4) to evaluate the specified mappings is more expensive than accessing other sources (Step 1) and thus exhibits high potential for performance optimization. Currently, the mapping database is completely managed and accessed by SRS like other sources. As an alternative, the query mediator may be implemented to directly access the mapping database, so that we obtain more opportunities for tuning.

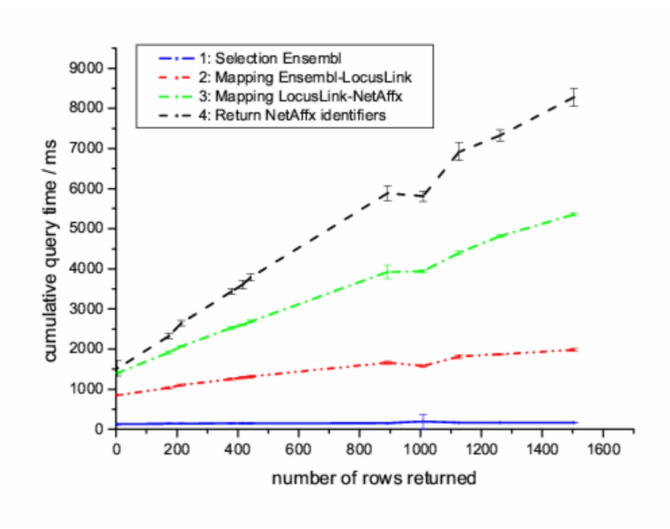


Figure 9: Query Time of Portions of Selection Queries to Ensemble

8 Conclusions

We presented a hybrid approach for the integration of annotation data from public data sources to support expression analysis of genes and proteins. Expression data is physically stored together with diverse experimental descriptions in a data warehouse supporting high performance expression analysis. Up-to-date annotation data is virtually integrated using a mediator and is retrieved on demand according to the analysis needs. The data warehouse and mediator are coupled by means of a query mediator, which exploits existing mappings between the integrated sources for join processing. The mappings are explicitly computed to involve a common central source, through which join operations can be efficiently performed at run time. The use of the a powerful commercial product, SRS of LION bioscience, for the mediator and the generic schema of the database to store the pre-computed mappings allows us to easily integrate a new source or update an existing source. The integration approach has been implemented as an enhancement of our gene expression data warehouse, but is also applicable to other domains, e.g. for protein analysis. The performance evaluation has shown the practicability of our integration approach.

9 References

[As00] Ashburner, M. et al.: Gene Ontology: tool for the unification of biology. Nature Genetics 25: 25-29, 2000. http://www.geneontology.org

[Ba00] Bairoch A.: The ENZYME database in 2000 Nucleic Acids Research 28:304-305 (2000). http://www.expasy.org/enzyme

[Bi04] Birney, E. et al.: An Overview of Ensembl. Genome Research 14: 925-928, 2004.

[CCW03] Chen, J.; Chung, S.Y.; Wong, L.: The Kleisli Query System as a Backbone for Bioinformatics Data Integration and Analysis. In [LC03]: 147-187.

[Ch04] Cheng, J. et al.: NetAffx gene ontology mining tool: a visual approach for microarray data analysis. Bioinformatics 20(9), 1462-3, 2004.

[DR04] Do, H.-H.; Rahm, E.: Flexible Integration of Molecular-biological Annotation Data: The GenMapper Approach. Proc. the 9th Int. Conf. on Extending Database Technology, Heraklion (Greece) 2004. Springer LNCS, 2004.

[EHB03] Etzold, T.; Harris, H.; Beaulah, S.: SRS: An Integration Platform for Databanks and Analysis Tools in Bioinformatics. In [LC03]: 109-145.

[Ga04] Galperin, M.Y.: The Molecular Biology Database Collection - 2004 update. Nucleic Acids Research 32, Database issue, 2004.

[Ha01] Haas, L. et al.: DiscoveryLink – A System for Integrated Access to Life Sciences Data Sources. IBM System Journal 40 (2), 2001.

[HK04] Hernandez, T.; Kambhampati, S.: Integration of Biological Sources: Current Systems and Challenges Ahead. SIGMOD Record 33(3), 2004.

[KDR03] Kirsten, T.; Do, H.-H.; Rahm, E.: A Multidimensional Data Warehouse for Gene Expression Analysis. In: Proc. German Conference on Bioinformatics, Munich 2003.

[KDR04] Kirsten, T.; Do, H.-H.; Rahm, E.: A Data Warehouse for Multidimensional Gene Expression Analysis. Technical Report, IZBI, University of Leipzig, 2004.

[La04] Lacroix, Z. et al.: Links and Paths through Life Science Data Sources. In [Ra04]: 203 – 211.

[LC03] Lacroix, Z.; Critchlow T. (Hrsg.): Bioinformatics: Managing Scientific Data. Morgan Kaufmann, 2003.

[LN05] Leser, U., Naumann, F: (Almost) Hands-Off Information Integration for the Life Sciences. Proc. 2nd Conference on Innovative Data Systems Research (CIDR), 2005

[PM02] Pruitt, K.D.; Maglott, D.R.: RefSeq and LocusLink: NCBI Gene-centered Resources. Nucleic Acids Research 29 (1), 2001. http://www.ncbi.nlm.nih.gov/projects/LocusLink/

[Po04] Potter, S.C. et al.: The Ensembl Analysis Pipeline. Genome Research 14: 934-941, 2004.

[Ra04] Rahm, E. (Ed.): Proceedings 1st Intl. Workshop Data Integration in the Life Sciences (DILS) 2004. LNBI 2994, Springer-Verlag, 2004.

[Ro04] Rother, K. et al.: COLUMBA: Multidimensional Data Integration of Protein Annotations. In [Ra04]: 156-171.

[St03] Stein, L.: Integrating Biological Databases. Nature Review Genetics 4(5): 337-345, 2003.

[Wh03] Wheeler D.L. et al.: Database Resources of the National Center for Biotechnology. Nucleic Acids Research 31: 28-33, 2003.
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=unigene

[Wo98] Wong, L.: Kleisli, a Functional Query System. Journal of Functional Programming, 1 (1): 1-000, 1998.

[Zd02] Zdobnov, E.M. et al.: The EBI SRS server – recent developments. Bioinformatics 18: 368-373, 2002.