

## context

The identification of gene variants expressed in a given condition is essential to the understanding of physiological and pathological states. This can be achieved by creating a DNA array representing all possible splicing events. An associated critical requirement is to maintain comprehensive functional information on the various RNA transcripts.

## objectives

The objective is to build a process that will:

- create a database of continuously updated functional information related to each of the array spots
- monitor splice variant specificity of each set of spots
- generate a human readable report containing an overview of the new or modified information of interest for each spot

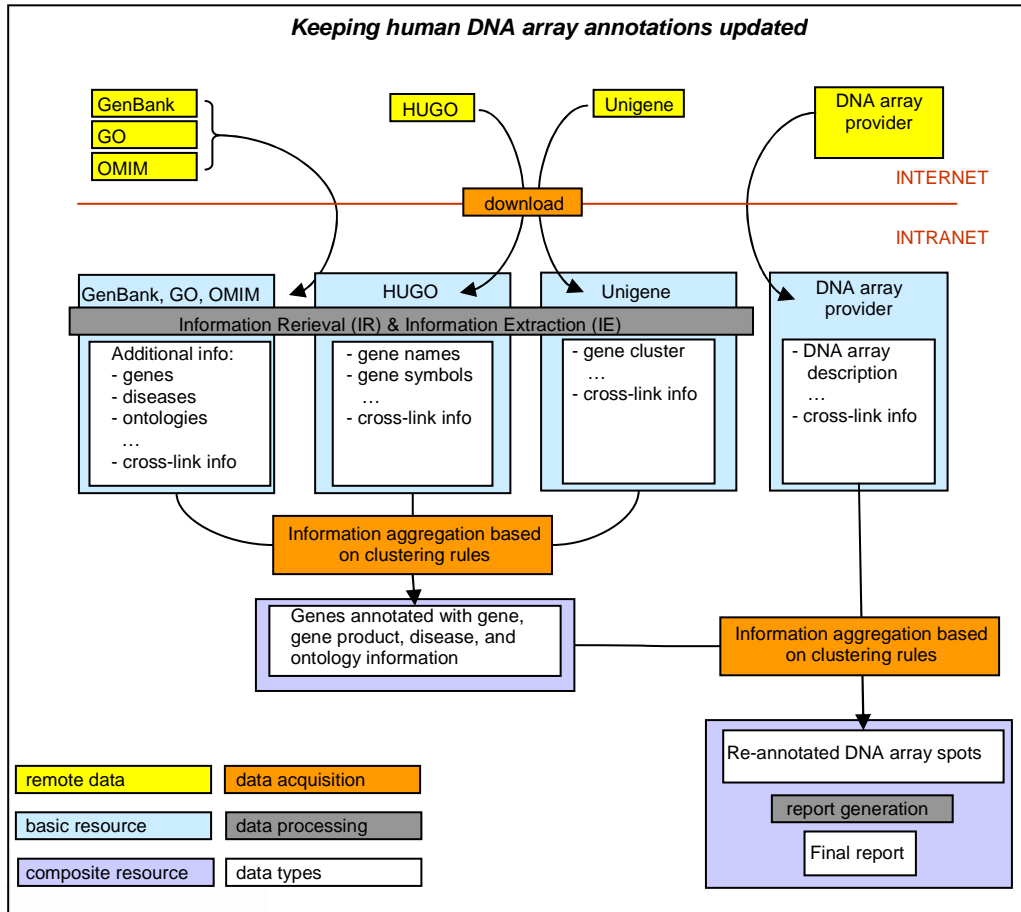
## GetDB™ solution

Our approach is to use GetDB™ to leverage a number of data sources by combining them with data provided by the DNA chip designer. The new annotations enable the re-interpretation of previous experimental results as well as a better understanding of new results. Furthermore, it improves the accuracy of the overall analysis by disregarding results from non specific probes.

### data sources

Selected data sources include:

- gene oriented databases: HUGO (official gene nomenclature), Unigene, RefSeq and H-Invitational
- protein oriented database: UniProt
- primary sequence database: GenBank
- disease related database: OMIM
- functional annotations relying on ontologies such as GO



Another source of data is provided by the DNA array designer, namely, probe sequences associated to gene names, exon numbers, and other information relevant to the probe.

- data processing

1 – After data has been internalized as *basic resources*, GetDB™ *plugins* implementing information retrieval (IR) and information extraction (IE) procedures are used to extract relevant information, including:

- official gene names and symbols (from HUGO)
- gene sequences (from Unigene, GenBank and H-invitational)
- functional annotations (from UniProt)
- disease related information (from OMIM)

Importantly, cross-references are extracted to generate links between heterogeneous data items.

2 - A *composite resource* containing annotated gene entries is created by a GetDB™ *construction method*, which aggregates the information items extracted at the previous step. This aggregation is based on clustering rules that use the above mentioned cross-references to decide whether two pieces of information provided by a particular resource or by two different resources can be grouped.

3 – A new *composite resource* is then built by adding DNA array related information using another GetDB™ *construction method*. The clustering rules invoked to establish mapping between spots and genes are either based on sequence similarities or gene identifiers.

## result

The end result is a *composite resource* of up-to-date annotated DNA array spots, which can further be processed using GetDB™ *plugins* to, for example, feed a relational database or generate human readable summaries about new information.

## GENOMINING IN SHORT

Genomining is a bioinformatics company, created in 2001, which provides real time solutions for the discovery and interpretation of data in biological research. It provides the life science industry with products ranging from data integration and manipulation frameworks, virtual screening management and optimization tools, to supercomputing solutions.

### CONTACTS

Katja Schuerer  
Bioinformatics Development Manager  
[Katja.schuerer@genomining.com](mailto:Katja.schuerer@genomining.com)  
Tel: +33 (0)1 42 31 08 01

Helene Chao  
Business Development  
[helene.chao@genomining.com](mailto:helene.chao@genomining.com)  
Tel: +33 (0)1 42 31 08 01

OUR WEBSITE : [www.genomining.com](http://www.genomining.com)

## Glossary

*Resource*: The basic unit on which GetDB™ applies data processing methods (plugin sequences). Resources represent data sets which have internal identifiers, names, versions and states. There are two main types of resources, namely, basic resources and composite resources.

*Basic Resource*: A GetDB™ resource which is acquired from another server either on the internal information system, or accessible through the internet. The resource consists of a set of files whose names match a filter expressed as a regular expression, which are locally updated by acquiring only new or modified data. Once acquired, normal plugin execution occurs on the basic resource.

*Composite Resource*: A GetDB™ resource which is created based on one or more other GetDB™ resources (the parent resources). The actual "building" or "construction" of the composite resource is handed off to a construction method. Once created, normal plugin execution occurs on the composite resource.

*Construction Method*: An arbitrarily complex program used to build a composite resource. The construction method mimics the download process of basic resources.

*Plugin*: A program which makes up the basic unit of processing in GetDB™. Plugins range from simple wrapper scripts to full fledged data processing programs. Plugins are run by GetDB™ whenever new data is incorporated into a resource. Dependencies between plugins can be expressed, thereby, creating plugin sequences.

