FlyMine

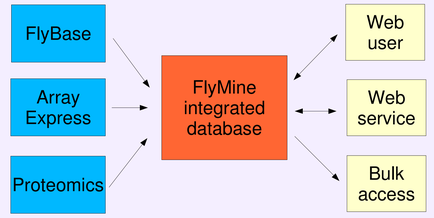
An integrated database for *Drosophila* and *Anopheles* genomics

www.flymine.org

Introduction

Currently, biological data is stored in a wide variety of formats in numerous different places, making complex queries across these data sources very difficult, if not impossible. FlyMine is an open-source data warehouse style system for integrating the data from many sources into one object-based database, allowing complex queries to be performed on the integrated data. Initially, the focus is on genomic, proteomic and microarray data for Drosophila and Anopheles, however the system is designed to be completely generic and is not tied to any particular data model – in fact all the model-specific parts of the system are automatically generated from a single UML diagram.

FlyMine has been designed from the outset as a powerful query tool, giving biologists the ability to perform arbitrary and complex queries on the data, and not be constrained by "fill-in-the-blanks" query templates. Access to the database will be via a website and SOAP-based webservice, and the system is available for local use under a LGPL licence.



Further information and download

Further information and documentation can be found on the FlyMine website (www.flymine.org), by joining one of the FlyMine electronic mailing lists (details on the website), or by email to info@flymine.org. The code is available for download under the open-source LGPL licence. We welcome co-developers.

The FlyMine team: Andrew Varley, Richard Smith, Matthew Wakeling, Mark Woodbridge, François Guillier, Rachel Lyne, Rajasekhar Paidi and Gos Micklem.

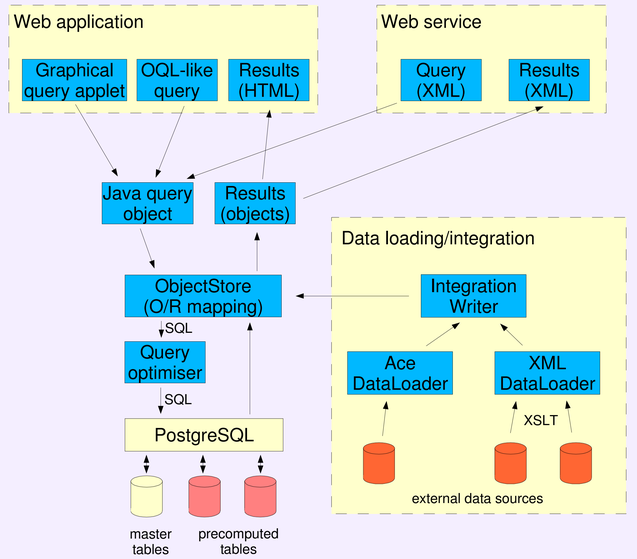
ObjectStore

At the heart of the FlyMine system is an ObjectStore. This is a Java class that accepts a FlyMine Java Query object and returns a table of Java business objects representing the results of the query. The ObjectStore deals with all the details of mapping Java business objects to tables in a relational database and is currently based on the open-source Apache OJB system, which we have heavily modified.

Even though the underlying queries to the database are executed in SQL, the ObjectStore hides all details of the underlying relational schema from users of the system; all queries are specified in terms of business objects and their relationships. This is a lot more intuitive than using SQL.

Results

Results of a query are returned as a table of Java business objects. References and collections in the returned objects are loaded automatically when first accessed. The results object intelligently handles paging of results sets, anticipating requests to the database for further rows according to use and buffering accordingly. This prevents unnecessary load on the database and avoids transferring large amounts of data.



Generic SQL query optimiser

*** Just released ***

The SQL query optimiser can be used independently of the rest of the FlyMine system to improve query response times for any read-only SQL database.

A single SQL query involving access to more than two or three tables of data will tend to be quite slow. Once loaded with data, FlyMine is able to create a large collection of *precomputed tables* - tables that are materialised views of one or more master tables. All incoming queries are automatically analysed to see if any combinations of these precomputed tables can be used to shorten the response time.

The precomputed tables are defined by a standard SQL query, so can include WHERE and GROUP BY/HAVING clauses if needed. The query optimiser builds a list of all possible rewritten queries containing all possible combinations of master and precomputed tables and asks the database to estimate the time each one will take. The estimated fastest is the query that actually gets run in the database (see diagram below), although a heuristic is applied to prevent over analysis of simple queries.

By asking the database for estimates of query execution times before attempting to execute a query, FlyMine is not vunerable to the database becoming overloaded if a badly specified query is submitted.

Queries

FlyMine provides a number of query interfaces. They are illustrated here with a trivial sample query: *"Show gene expression data for genes which have GO term GO:0000278 applied"*.

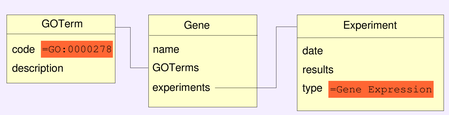
a) OQL-like query language

```
SELECT gene, exp
FROM Gene AS gene, GOTerm AS term, Experiment AS exp
WHERE gene.GOTerms CONTAINS term
AND term.code = "GO:0000278"
AND gene.experiments CONTAINS exp
AND exp.type = "Gene Expression"
```

b) FlyMine Java query classes

```
Query q = new Query();
QueryClass qcGene = new QueryClass(Gene.class);
QueryClass qcTerm = new QueryClass(GOTerm.class);
QueryField qfGeneTerm = new QueryField(qcGene, "GOTerms");
ConstraintSet cs = new ConstraintSet();
Constraint con1 = new ContainsConstraint(qfGeneTerm,
                  ContainsConstraint.CONTAINS, qcTerm);
cs.add(con1);
.
.
q.setConstraint(cs);
```

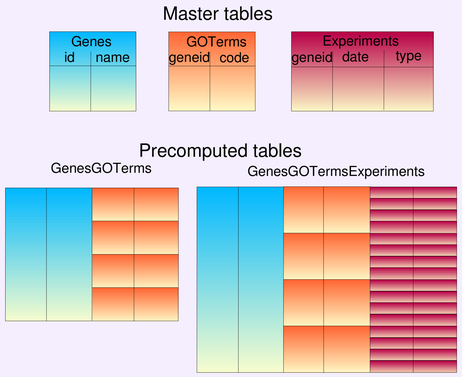
c) Graphical query interface (under development)
Users drag and drop boxes representing objects in the data model and place constraints on the fields of the objects and their relationships to other objects. This will be accessible as a web application.



Data loading and integration

Reading and translation of data from different sources is handled by a selection of DataLoader classes. These convert data in an external format (eg. foreign XML, AceDB format, flat-file format) into Java business objects conforming to a FlyMine data model. These objects are then passed on to an IntegrationWriter to handle the storage of the objects in the database.

The IntegrationWriter has to decide various things about the objects it receives from a DataLoader, since, for example, some information about the objects may already exist in the database, and the existing information may conflict with the incoming data. The IntegrationWriter keeps track of the origin of each piece of information for every object in the database and consults various configuration parameters to decide whether existing information should be added to or replaced in the light of new data from a different source. Therefore, the order in which data sources are loaded when building the integrated database is unimportant.



Incoming SQL query

```
SELECT genes.name, experiments.date
FROM genes, goterms, experiments
WHERE genes.id = goterms.geneid
AND genes.id = experiments.geneid
AND goterms.code = "GO:0000278"
AND experiments.type = "Gene Expression"
```

how long? 5 seconds

Possible rewritten queries using precomputed tables

```
SELECT genes_name AS name,
       experiments.date
FROM genesgoterms, experiments
WHERE genes_id = experiments.geneid
AND goterms_code = "GO:0000278"
AND experiments.type = "Gene Expression"
```

how long? 1 second

```
SELECT genes_name AS name,
       experiments_date AS date
FROM genesgotermsexperiments
AND goterms_code = "GO:0000278"
AND experiments_type = "Gene Expression"
```

how long? 200ms

Department of Genetics, University of Cambridge,
Downing Street, Cambridge CB2 3EH, UK
Tel: +44 1223 333377 Email: info@flymine.org

FlyMine is funded by the Wellcome Trust (grant no. 067205), awarded to M. Ashburner, G. Micklem, S. Russell, K. Lilley, and K. Mizuguchi.