InterMine

Object integration and warehousing software

www.intermine.org

Introduction

InterMine is a general purpose object-oriented data warehouse system developed as part of the FlyMine project. InterMine enables configurable integration of complex data from multiple sources into a single query-optimised object database. Users are able to create powerful, arbitrary queries using an intuitive web application or an object based query language.

InterMine is model-independent: any model specific components can be automatically generated. The system enables integration of models in various formats (UML, XML-Schema, DAG or OWL) and can incorporate data from SQL databases, XML or flat files.

InterMine is designed to be a powerful query tool, giving users the ability to design complex queries on the data and not be constrained by "fill-in-the-blanks" query templates. A web based query builder application and a SOAP-based webservice are also provided to operate on any InterMine model. The system is available for download under the open-source LGPL licence.

Current InterMine Implementations

1. FlyMine - an online database of integrated *Drosophila* and *Anopheles* genomic and proteomic data. FlyMine aims to integrate standard formats where possible, the core of the object model is the Sequence Ontology merged with ensembl annotation, MAGE microarray data, PSI protein interaction and GO annotation. Though FlyMine primarily focuses on fly data the same system could be used for any organism.

2. Type I diabetes - with funding from the Cambridge-MIT Institute, a collaboration has been established with groups at the CIMR (Smink, Gottgens, Todd, Caldas). InterMine is being used to integrate genomics data from human, mouse and other vertebrates as part of efforts to study this complex disease and elucidate gene regulatory networks.

3. InterMine will be the central informatics component of a blood stem cell systems biology consortium which has recently been established.

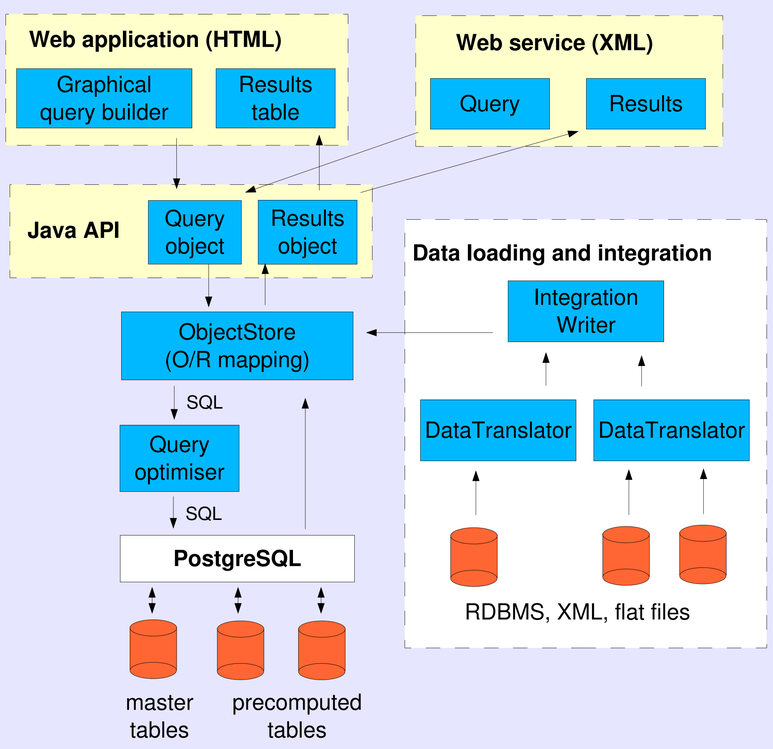
ObjectStore

At the heart of the InterMine system is the ObjectStore. It provides a powerful mechanism for querying and retrieving objects and is heavily optimised for read performance. The ObjectStore Java class accepts an InterMine query and returns a table of Java business objects representing the results.

Features:
- automatic, transparent mapping of objects to an underlying relational database (PostgreSQL)
- support for object models with multiple inheritance
- caching of retrieved objects
- results paging with a pre-fetch system capable of anticipating future requests and buffering accordingly
- object/relational mapping strategy designed to boost retrieval performance at the cost of disk space
- lazy materialisation of references and collections from retrieved objects
- optional pre-computation of common queries for huge performance gains

The ObjectStore is capable of estimating the resources required to run an arbitrary query. The ensures that demanding or poorly defined queries do not have the potential to overload the server and detrimentally affect other users.

InterMine architecture



Queries

InterMine has a number of query interfaces. They are illustrated here with a trivial example: "*Show details of employees of the finance department of age greater than 45*".

a) InterMine Java query classes – locally or via a web service

```
Query q = new Query();
QueryClass employee = new QueryClass(Employee.class);
QueryClass department = new QueryClass(Department.class);
q.addToSelect(employee);
q.addFrom(employee);
q.addFrom(department);
Constraint cs = new ConstraintSet();
cs.add(new SimpleConstraint(new QueryField(employee, "age"),
    ConstraintOp.GREATER_THAN, new QueryValue(new Integer(45))));
cs.add(new SimpleConstraint(new QueryField(department, "name"),
    ConstraintOp.EQUALS, new QueryValue("finance")));
cs.add(new ContainsConstraint(new QueryObjectReference(
    employee, "department"), ConstraintOp.CONTAINS,
    department)));
q.setConstraint(cs);
```

b) IQL – InterMine implementation of OQL

```
select Employee from Employee, Department
   where Employee.age > 45
      and Department.name = 'finance'
      and Employee.department CONTAINS Department;
```

c) In the web based query builder (screenshot below)

Generic SQL query optimiser

Rapid query performance is essential for any data warehouse. To retain model independence InterMine employs a generic SQL query optimiser. Once loaded with data InterMine is able to create a large collection of pre-computed tables – tables that are materialised views of one or more master tables. All incoming queries are analysed to see if any combinations of these pre-computed tables can be used to shorten response time.

Pre-computed tables are defined by standard SQL statements so can include "where" constraints if needed. The query optimiser builds a list of all possible rewritten queries and asks the database server to estimate the time that each would take to run. The query estimated to be the fastest is executed. A heuristic is applied to prevent over-analysis of simple queries.

It is planned that in future InterMine will log incoming queries and adopt a machine learning approach to define the most useful pre-computed tables.

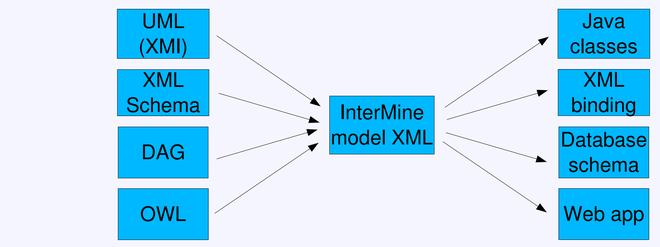
The SQL query optimiser can be used independently of InterMine to improve query response times for any read-only SQL database.

Data loading pipeline

Model creation
InterMine requires a model definition as an input. This is specified in XML and can be created automatically from a UML diagram, XML-Schema, OWL and DAG formats.

An InterMine object warehouse may be created from a single data source but is designed to support integration of multiple sources. A model is created from a user defined merge specification which describes intersecting mapping between models.

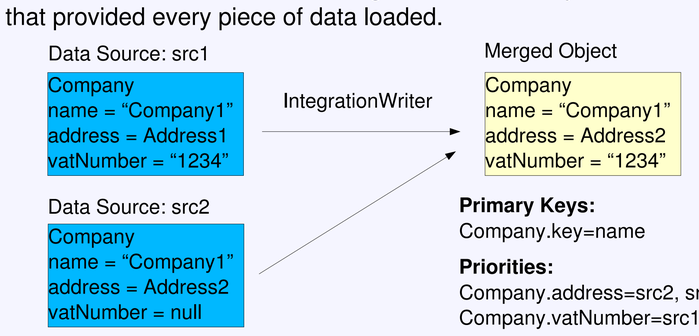
Automatic code generation
All InterMine code is data model independent. The InterMine model XML is used for automatic generation of Java business objects (bound to XML), a database schema and Object/Relational mapping specification.



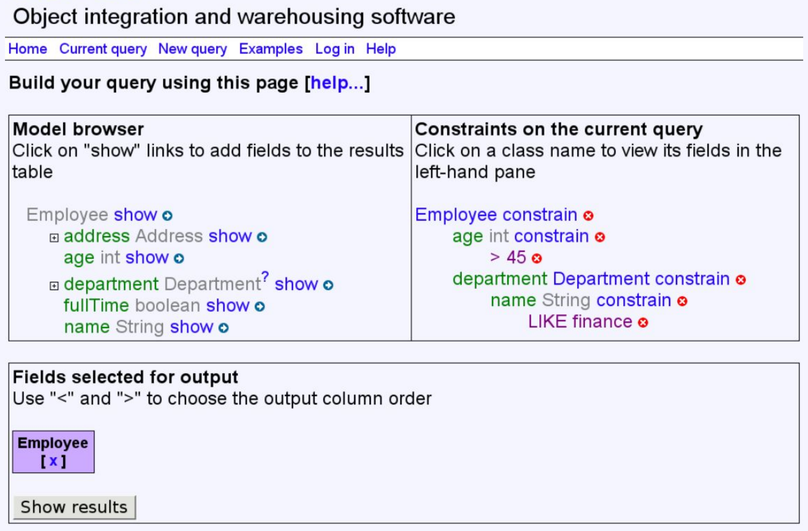
Data retrieval and translation
Classes are provided to convert data from various formats (RDBMS, XML or flat-files) into Java business objects conforming to an InterMine model. Where the target model is different to that of the source a translation step is required. Simple name changing and class/attribute removal can be defined in the model merge specification, more complex transformations require custom code to be written using an expressive translation API. Future releases of InterMine will see the merge specification become more sophisticated.

Integration
The IntegrationWriter loads business objects into the target ObjectStore. In some cases objects from two different sources represent the same data and should be merged. It is possible that some fields of the merged object may originate from two different sources (see example). Multiple sets of primary keys can be defined for each class in the model, individual sources can be defined to use one or more of these depending on available fields.

Where the same data is available in more than one source a user defined priority configuration is used to choose which values to accept, defined on a class or attribute level. The IntegrationWriter always tracks the sources that provided every piece of data loaded.



Web Application - Query Builder



The InterMine package includes a model independent Java web application for querying and browsing an ObjectStore. It allows web users to construct and execute queries then view and save the results without using code or knowledge of a query language.

Features of the web application:

- model independence, but configurable on a per-model basis
- model navigation (via references between classes) using the query builder
- any class or field in the model can be included in the output
- the query output can be filtered/constrained using any class or field in the model
- query result sets (bags) can be saved by name for use in other queries and logical operations
- bags can be created from files (for example, from a file of identifiers)
- queries and bags can be saved between sessions
- users can login to access their queries and saved bags (history)
- query results can be exported in simple text formats suitable for reading into spreadsheets and other programs
- facility to create templates for commonly used queries

Further information and download

Further information, documentation and a demo query builder can be found on the InterMine website (www.intermine.org). Email to info@intermine.org. InterMine has been in active development for over 18 months. The code is open-source and available under LGPL. We welcome co-developers.

The FlyMine database and user documentation can be accessed at www.flymine.org. FlyMine specific code is also available under LGPL.

The FlyMine team: Richard Smith, Matthew Wakeling, Mark Woodbridge, François Guillier, Rachel Lyne, Kim Rutherford, Wenyan Ji, Tom Riley and Gos Micklem.

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.