Exp-DB: Fast Development of Information Systems for Experiment Tracking

Nomair A. Naeem[1], Stéphane Raymond[2], Anne Poupon[3], Miroslaw Cygler[2], and Bettina Kemme[1]

[1] School of Computer Science, McGill University, Montreal, Canada
[2] Macromol., Structure Group, NRC Biotechnology Research Inst., Montreal, Canada
[3] Lab. d'Enzymologie et Biochimie Structurales, CNRS, Gif-sur-Yvette, France

Abstract. Bioinformatics research groups require information systems keeping track of experiments and their results. However, current solutions are often ad-hoc, difficult to maintain, extend, or use in different context. This paper presents Exp-DB, an infrastructure for the fast development of information systems for experiment tracking. Exp-DB provides a basic, extensible database design, components for accessing the data, and a web-based interface. It allows for fast implementation of an initial system even by non IT-experts which can then be extended step-by-step.

1 Introduction

Bioinformatics research is driven by experiments. Typically, experiments are executed in from of a workflow: related experiments are executed in a predefined order and might depend on each other. We refer to a set of such related experiments as *experiment workflow*. With the introduction of new technology, more and more experiments are automated leading to an explosion in the number of experiments and a need to closer analyze the settings of previous experiments in order to focus on promising experiment paths.

As a result, many research groups have started to develop web-based information systems that keep track of their experiments, e.g., [2, 5, 6]. The startup cost, that is, the initial investment until a first system is running, can be quite high. Since these systems are often built by biologists or other scientists, the team members first have to learn the essentials of the technology if they do not know it yet. Since at the beginning it is not yet clear how the system is going to evolve, and due to time pressure and lack of experience, the systems often lack modularity and extensibility, and are only usable in very specific context. As a result, each research group has to develop its own system completely from scratch in order to reflect their particular way of performing experiments.

This paper addresses the problems and issues mentioned above. It presents Exp-DB, a multi-tier infrastructure intended to help small scientific research groups to develop their own information system for experiment tracking. The design and implementation of Exp-DB was motivated by two goals. Firstly, the design, architecture, and infrastructure must be simple enough to be understood in reasonably short time by people without extensive computer skills. This is

needed to have a first running system in short time. Secondly, the system must provide sufficient modularity and flexibility such that it can be extended, adjusted and optimized in many different ways (although this might require more computer science knowledge than the first step).

The architecture of Exp-DB follows a strict separation of tasks: data storage, application logic, and presentation logic are implemented in separate components. Within each of the components, different concepts are implemented separately from each other in different modules. The architecture is based on Java Technology and uses Apache/Tomcat as its execution environment.

A research group develops its own system based on Exp-DB in the following steps. Exp-DB provides an initial, quite simple relational data model. Relying on the basic structure of the Exp-DB data model, attributes have to be added to existing tables, and new tables have to be created reflecting the particular research conducted. For instance, for each type of experiment, typically one additional table has to be added to the database plus some indication how this experiment type is related to other experiment types. For each newly created table, one small application logic program has to be generated. No further work is required to get a first running system. Web-pages are automatically generated from the entered information. The tasks to be performed require moderate knowledge in relational database design, and basics in Java programming.

There exist many databases storing keeping track of scientific experiments, e.g., [5, 6, **?**]. General laboratory systems are e.g., [4, 7, 1, 8]. However, none of these system provides the general functionality and simplicity of our system.

2 Experiments and Workflows

Our system has been designed while cooperating with the Macromolecular Structure Group at the Biotechnology Research Center, National Research Council, Canada (http://www.bri.nrc.ca/pha_bio/msg/msg_en.htm), and we would like to take their experimental setup as an example throughout the paper. This group is engaged in finding the structure of proteins mainly through x-ray crystallography. The workflow starts with (1) selecting a set of genes or subsequences of genes as promising targets. For each selected target the further steps include (2) cloning, (3) protein production, (4) purification, (5) crystallization, (6) X-ray crystallography, and (7) structure analysis. (8) An alternative method after step following step (5) is NMR (nuclear magnetic resonance) spectrometry. (9) Optional quality testing occurs after the purification step (4). (10) Also, if steps (5) to (8) are unsuccessful, fragmentation splits the proteins retrieved in step (4) into smaller fragments that might be easier to crystallize. Steps (2) to (4) might be skipped if the proteins are provided by other labs.

From this example workflow, we can introduce the following definitions. An **experiment type** describes a specific form of experiment, e.g., "cloning", "purification", etc. An **experiment** is an instance of an experiment type. An **experiment workflow** is a directed graph. Each node depicts an experiment. An edge from experiment $E1$ to experiment $E2$ indicates that $E1$ was performed

before $E2$ and its output served as input for $E2$. Experiments are not randomly combined to experiment workflows but usually follow standard workflow patterns. That is, an experiment of type T1 might typically follow an experiment of type T2 but, e.g., not vice versa. Some patterns, however, are impossible (e.g., *Crystallization* → *Cloning*). This means, we have to restrict experiment workflows to only allow meaningful transitions between experiment types. A *workflow model* a directed graph where nodes are experiment types. An arrow from experiment type T1 to experiment type T2 indicates that the output of an experiment of type T1 can (but need not) be input of an experiment of type T2. From there, a **legal workflow** is an experiment workflow such that for each arrow from experiment E1 of type T1 to E2 of type T2 in the workflow graph, the workflow model graph has an arrow from T1 to T2. A workflow can have one-to-many and many-to-one relationships. That is, experiment E1 of type T1 can have edges to experiments E21 and E22 both of type T2 as long as there is an arrow from T1 to T2 in the model (one purification experiment is input for many crystallization experiments). In a similar way, many experiments of the same type can be input to one further experiment (e.g. the results of many x-ray crystallography experiments can be input for a single structure analysis).

3 Database Design

Exp-DB distinguishes different types of information. textitAccess control information is needed to restrict access to the system. *Special information* is information that is very specific to a given research environment, e.g., information about genes and proteins. Finally, *meta information* helps the interface layer to retrieve information about the structure of the tables and their attributes. This is needed for presenting data correctly and for type checking.

Experimental information is the backbone of the schema and describes experiment types, experiments, workflow models, and experiment workflows. The `Experiment` table contains general information about each experiment independent of the experiment type. Type dependent information is stored in separate tables, one for each type. The table `Workflow` is used to express workflows by indicating parent and child experiments. The workflow model graph is encoded into the `Workflow Model` table. Whenever a user wants to enter a record into the `Workflow` table, the system checks in the `Workflow Model` table whether such an experiment flow is legal.

4 Architecture and Implementation

The system has a three-tier architecture and follows standard software development guidelines. Users (client tier) interact with the system through a standard browser. The middle tier is responsible for the presentation logic. It uses the MVC (Model View Controller) architecture. This helps to provide a clean separation between the presentation and the processing of data. Our runtime environment is an Apache Tomcat 4 web server that provides support for JSP 1.2

and Servlets 2.3 specifications. The backend tier can be any relational database system. The presentation logic generates the webpages using JavaServer Pages (JSP). The JSPs represents the "View" that the users receive from the system. The application logic is implemented as a combination of Servlets and JavaBeans, and fulfills different needs. The JavaBeans encapsulate the data access to the database system and represent the data in an abstract interface to JSPs and Servlets. They implement the "Model". Finally, the Servlets "control" how the user interacts with the system, and how the different components should be called. As backend tier, we currently use PostgreSQL.

The system consists of several modules. Each module consists of a set of JSPs and JavaBeans that implement a specific functionality: users can enter new information (e.g., new experiments), search the database, or access specific existing entries in the database. All requests are filtered through the controller servlet `dispatcherServlet` that forwards it to the corresponding module.

The *non-experiment* addition module is responsible for entering individual entries to non experimental tables. It has a single JSP that is valid for all tables, and a JavaBean for each table. It is completely hidden from the JSP how the data is internally stored since all data access is passed through the JavaBean.

The *experiment* addition module is more complex since the addition of an experiment requires entries into several tables and the experiment must be linked to a specific workflow. The user is guided through a sequence of web-pages (JSP) to enter all relevant information. Correctness checks are performed to guarantee that only legal workflows are allows.

The *query module* provides a set of predefined queries that are typical for experimental databases. Furthermore, the user can ask more complex queries in form of SQL queries or through a query wizard.

The *record module* provides the functionality to access (view, modify, delete) an individual existing record. For each of these methods, there exist one single JSP (independently of the table). This JSP calls the corresponding JavaBean of the table to retrieve or store the data, and then dynamically generates the webpage. Furthermore, this module provides navigation through an experiment workflow. Starting with the view of an initial experiment, the user can navigate to the information about the child experiments or parent experiments.

Exp-DB only provides the basic framework. Before it will be operational, it has to be extended and adjusted. We expect the most typical extension to be the addition of new experiment types and the specification of the workflow model. Such extension can be done very easily. Special extensions are possible, but will require more knowledge about the underlying technology.

References

1. G. Alonso, C. Hagen: Geo-Opera: Workflow Concepts for Spatial Processes, Int. Conference on Scientific and Statistical Database Management, 1997.
2. J. Frew, R.Bose: Earth System Science Workbench: A Data Management Infrastructure of Earth Science Products, Int. Conference on Scientific and Statistical Database Management, 2001.

3. G. L. Gilliland, M. Tung, J. Ladner: The Biological Macromolecule Crystallization Database and NASA Protein Crystal Growth Archive, Journal of Research of the National Institute of Standards and Technology, 101(3), 1996.
4. N. Goodman, S. Rozen, L. D. Stein, A.G. Smith: The LabBase system for data management in large scale biology research laboratories. Bioinformatics, 14, 562-574.
5. P. W. Haebel, V. L. Arcus, E. N. Baker and P. Metcalf: LISA: an intranet-based flexible database for protein crystallography project management, Acta Crystallographica D57, 1341-1343, 2001.
6. M. Harris and T. Alwyin: Xtrack – a web-based crystallographic notebook, Acta Crystallographica D58, 1889-1891, 2002.
7. Y. Ioannidis, M. Livny, S. Gupta, N. Ponnekanti: ZOO: A Desktop Experiment Management Environment. Int. Conference on Very Large Databases, 1996.
8. J. Meidanis, G. Vossen, M. Weske: Using Workflow Management in DNA Sequencing. Int. Conference on Cooperative Information Systems, 1996.