Desktop Experiment Management

Y. Ioannidis[*] M. Livny E. Haber R. Miller O. Tsatalos J. Wiener

{yannis,miron,haber,rmiller,odysseas,wiener}@cs.wisc.edu
Computer Sciences Dept., 1210 W. Dayton St., Univ. of Wisconsin, Madison, WI 53706

1 Motivation

Traditionally, the scale and scope of an experimental study was determined by the ability of the research team to generate data. Over the past few years, we have been experiencing an unprecedented increase in the ability of small teams of experimental scientists to generate data. This has led to a shift in the balance between the different components of an experimental study. Today, it is not uncommon to find a study whose scale and scope have been determined by the ability of the team to manage the data rather than to generate it. Whether the discipline is experimental computer science [4], genetics, earth and space sciences, soil sciences, or high-energy physics, scientists are faced in their daily work with an experiment and data management bottleneck. Unfortunately, an experimental scientist can not find ready off-the-shelf management tools that offer both the functionality required by a scientific environment and an interface that feels natural and intuitive to the non-expert. While no special expertise is needed to manage a collection of images stored as files on a PC or as pictures in a paper notebook, existing database systems (DBMSs) that offer the desired functionality require expertise that most teams of experimental scientists do not have and can not afford to pay for. This poses a challenge to the database community to develop management tools that can be tailored by a typical scientific team to effectively manage their unique experimental environment.

To address this challenge, we have undertaken an effort to develop a desktop *Experiment Management System* (*EMS*) [2]. We view the desktop EMS as the sole interface between the experimental scientist and the data. Throughout the life cycle of a study, the system will support the scientist in a range activities; it will be used to design the study, to order experiments, to manage the data, and to analyze the results. Before such a system can be implemented and placed on the desk of typical experimental scientists, answers to a range of traditional and non-traditional database management problems must be obtained. In this paper, we give an overview of the activities performed by scientists throughout the course of an experimental study and present the overall architecture of the EMS under development. We then discuss the issues that we have addressed so far and outline some of the solutions that we plan to incorporate in the system.

2 Experiment Life Cycle

To achieve its goals, an EMS will use conceptual schemas for various activities that are important throughout the course of an experimental study. From discussions with scientists from different disciplines, we have concluded that these activities are common to most experimental studies. We use the term *experiment life-cycle* to denote the entire set of these activities together with the way scientists iterate over them during such a study. A pictorial abstraction of the experiment life-cycle is shown in Figure 1. It essentially consists of multiple loops traversed by the researcher multiple times in the

[*]Partially supported by the National Science Foundation under Grants IRI-9113736 and IRI-9157368 (PYI Award) and by grants from DEC, IBM, HP, and AT&T.

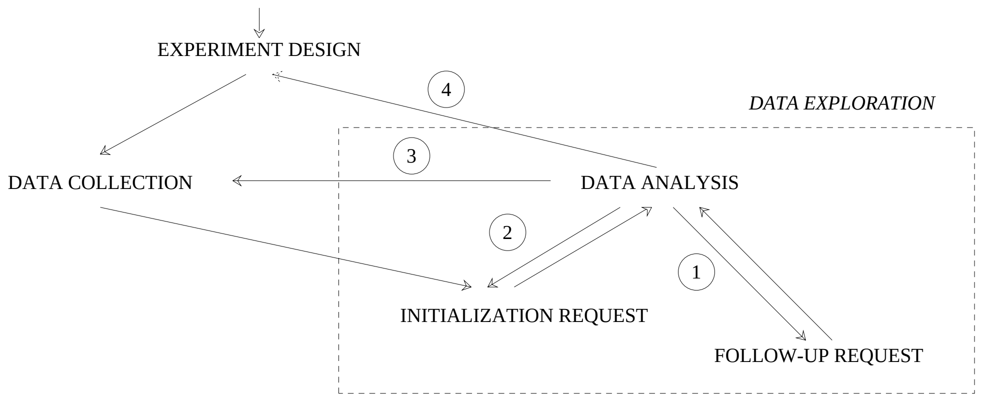

Figure 1: Life cycle of an experimental study.

course of a study. In the figure, the following stages can be identified. **Experiment Design:** The structure of each experiment is defined determining the input and the output of the experiments. **Data Collection:** Experiments are actually conducted. The researcher specifies the experiment set-up and the precise values of all the input parameters to the experiment, and the relevant output data is then collected. **Data Exploration:** The researcher studies the collected data to draw conclusions about the subject of the experiment. As shown in Figure 1, there are three types of actions that the scientist may perform on the data, which are described separately. *Initialization requests:* Whenever scientists start to explore a new vain of thought in an experimental study, their first request on the collected data must reference all properties of the phenomenon or system under study that are expected to remain unchanged throughout the exploration of the new idea. Thus, such a request needs to deal with a large portion of the experiment schema. *Data Analysis:* After receiving the requested data, scientists analyze it based on domain-specific knowledge that is relevant to the studied phenomenon. *Follow-up Requests:* Based on the results of the analysis of some obtained data, quite often scientists pose new requests that are very similar to the previous ones, having the answers of the latter as a reference point. This is due to the predominantly exploratory nature of experimental science. Follow-up requests represent the most common form of interaction during a study.

3 An EMS Architecture

A fundamental premise of our effort to develop a desktop EMS has been to provide the typical scientist with a single interface for all stages of the experiment life-cycle. Figure 2 presents the architecture of the EMS under development at the Univ. of Wisconsin, which is based on this premise. On the front end, the user interacts with the system via intuitive language and graphical interfaces (*User Interfaces*). At the heart of the system lies a database system (*Core DBMS*), which provides the traditional query and data storage services. On the back end, the EMS is coupled via *Data Translators* to a variety of data sources (*experimentation environments* where experiments can be run, other EMSs, and DBMSs). Finally, the EMS has an active component, which coordinates the interaction between the system and the external data sources (*Experimentation Manager*).

In the following sections we outline the main properties of these components. Before we do so, we would like to point out that the *User Interfaces* component is based on a 'schema-centric' approach. Since the most important piece of information that is necessary throughout the life-cycle of the experiment is the conceptual schema of the scientific data, we use it as the common foundation for all interactions between the scientist and the EMS. Whereas in a conventional DBMS, the schema captures the structure of the data in a database, in an EMS environment, the schema also captures the

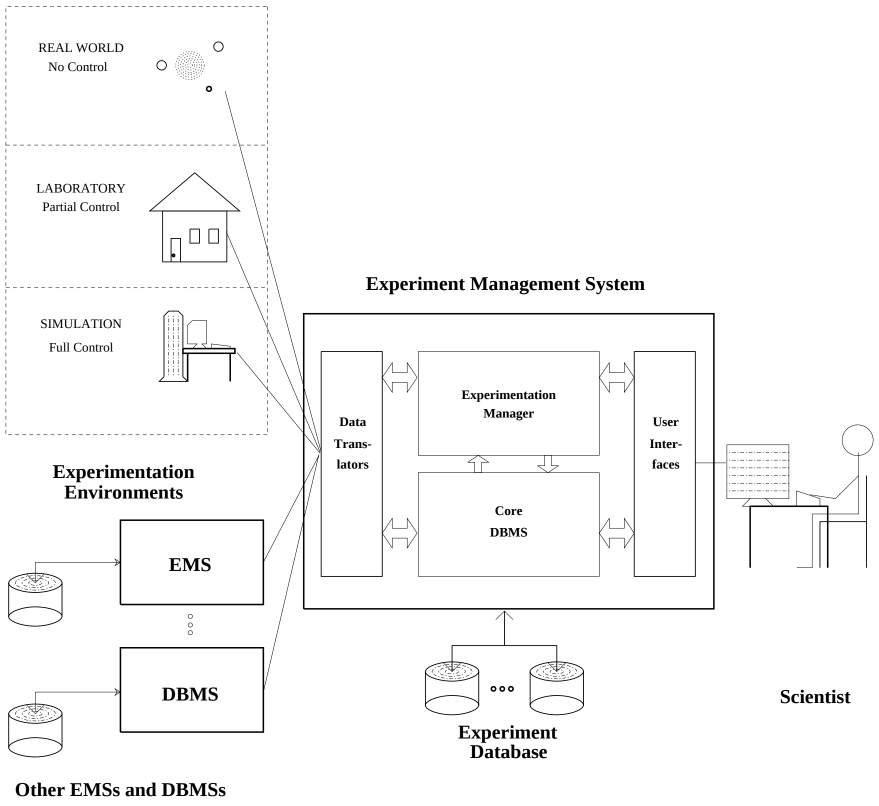

Figure 2: Architecture of an Experiment Management System.

structure of the experiment itself. This is a side-effect of the effort to describe the structure of the data: in order to organize the data in a meaningful way, the design of the experiment is essentially represented as well. Therefore, the schema of an experiment is called to play two new major roles: (1) To serve as a formal document describing the experiment; (2) To serve as the template for specifying data and experiments. Conceptual schemas undertake these new roles not only within the EMS, but also in interactions between collaborating scientists.

While in a traditional DBMS environment, schemas are manipulated by professional database administrators, in a desktop EMS environment, the scientists themselves are the ones who define and modify schemas. Also, by the nature of scientific studies, ad-hoc queries are the main form of interaction with the system. This requires that scientists can easily obtain the necessary details of schemas. Given the complexity and size of typical schemas of experimental studies, the above implies that the EMS itself should be constantly prompting scientists with the appropriate schemas.

4 The Core DBMS

The Core DBMS of the EMS under development is based on object-oriented (OO) technology. Due to the special needs of many experimental sciences, we have developed our own OO data model, Moose (Modeling Objects Of Scientific Experiments) and query language, Fox (Finding Objects of eXperiments). Detailed descriptions of these may be found elsewhere [1, 2, 9].

The development of Moose and Fox was based on several characteristics of the data found in scientific experiments and of the way scientists are expected to interact with an EMS. The main features of Moose are: (1) Treatment of collections (e.g., sets) as first-class objects, which may be associated with other information possibly independent of the objects in the collection. (2) Distinction between the structural components of a complex object (*part-of* relationships) from any other objects with which it may be associated. (3) Explicit representation of collections indexed by some other, arbitrary, collection (i.e., generalized forms of arrays). (4) Support for *virtual* (i.e., *derived*) relationships (resp. classes), whose values (resp. membership) is defined through rules that are based on the Fox query language. In addition, any Moose schema has a straightforward graph representation, where classes are represented as nodes and relationships as edges.

The main features of Fox are: (1) Access to individual elements of collections indexed by arbitrary sets. (2) Query path expressions traversing arbitrary types of relationships in arbitrary ways. (3) Periodic data, e.g., time series or spatial domains, concisely described and virtually defined using the virtual relationship mechanism of Moose.

Efficient processing of Fox queries requires extensive index support on the part of the DBMS due to the expected complexity of Moose schemas for experiments. Feature (2) of Fox above renders all existing path indexing schemes inapplicable. We have designed a novel indexing scheme, *generalized multi-level access paths*, that removes all such restrictions [8]. In addition, it allows partial indexing (e.g., indexing intentionally-specified collections of objects) and indexing that associates two groups of objects instead of the traditional pair of individual objects.

5 The Graphical User Interface

The design of the User Interfaces has been shaped by two goals: to provide an integrated tool to be used in all stages of the experiment life cycle, and to allow scientists to use the system in a manner that is natural to them [3]. For schema (i.e., experiment) design, the interface provides a *schema editor*, which uses the services of a *graph editor* due to the graph representation of Moose schemas. This component is already implemented and has been used in the context of real experiments in soil sciences [7]. Work is underway on examining if the graph representation is the right metaphor for Moose schemas at all times, what other alternatives exist, and how they can be supported simultaneously.

For querying (and possibly data display), the interface will use the graphical representation of a schema as a template. Because of the large size of experiment schemas, path expressions in queries will tend to be very long. We are currently working on developing techniques to allow the specification of incomplete paths (e.g., specifying only the two end-points) that will be completed internally by the system based on the schema structure and the semantics of the relationships involved in the candidate complete paths. In addition, based on the exploration state of the experiment life-cycle, follow-up queries are expected to be most common. Our efforts focus on allowing scientists to use the results of previous requests as the basis for future ones.

6 The Experimentation Manager

An important feature of an EMS is that it will be capable of hiding the distinction between the data collection and data exploration stages of the experiment life-cycle. The scientist will be given the freedom to request data without any knowledge of whether it has already been measured and recorded or not. The EMS will decide whether to simply retrieve the data from its database, or initiate some action outside of the system. The Experimentation Manager is the module responsible for making this decision. In the case where experiments need to be invoked, the Experimentation Manager must identify the appropriate experimentation environment, collect all the information necessary to run that experiment, and feed it to the translator. Much of that information will not be part of the scientist's request. We have been investigating techniques that can be used to infer the necessary information by using the virtual relationships mechanism of Moose.

7 The Translators

By the nature of experimental studies, an EMS should provide a cohesive interface to a range of experimentation environments, which may have been independently developed, and should be able to communicate with other EMSs and DBMSs that manage data of interest already collected as part of other studies, so that duplication of effort is avoided (Figure 2). Since these data sources will most likely not use Moose as their data model, Data Translators should be incorporated to the EMS to translate between the various schemas.

We have focused on the problem of translating schemas and their instances from different data models into Moose. In addition, for the case where multiple experimentation environments are generating data for the same experiment, special emphasis is given on integrating multiple schemas. We have developed a formal correctness criterion (based on information capacity of schemas) for schema translation and integration, which is motivated by the practical requirements of the schema translation task [6, 5]. We are currently using this work in developing prototype Data Translators.

8 Status

Our work on the EMS has benefited immensely from our collaboration with scientists from soil sciences and molecular biology. The schema editor that has been developed is already being used to capture experiments in these disciplines. Part of the Core DBMS has been implemented, supporting a portion of the Fox language. We expect to have a first prototype of the EMS by the summer of 1994.

References

[1] Y. Ioannidis and M. Livny. MOOSE: Modeling Objects in a Simulation Environment. In G. X. Ritter, editor, *Information Processing 89*, pages 821–826. North Holland, August 1989.

[2] Y. Ioannidis and M. Livny. Conceptual Schemas: Multi-Faceted Tools for Desktop Scientific Experiment Management. *Journal of Intelligent and Cooperative Information Systems*, 1(3), December 1992.

[3] Y. Ioannidis, M. Livny, and E. Haber. Graphical User Interfaces for the Management of Scientific Experiments and Data. *ACM-Sigmod Record*, 20(1):47–53, March 1992.

[4] M. Livny. DELAB - A Simulation Laboratory. In *Proc. of the 1987 Winter Simulation Conference*, Atlanta, GA, December 1987.

[5] R. Miller, Y. Ioannidis, and R. Ramakrishnan. The Use of Information Capacity in Schema Integration and Translation. Submitted for publication.

[6] R. Miller, Y. Ioannidis, and R. Ramakrishnan. Understanding Schemas. In *International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, Vienna, Austria, April 1993.

[7] L. M. Murdock. Developing an Object-Oriented Experiment Data Management System for the Cupid Plant-Environment Model. Master's thesis, University of Wisconsin, Madison, June 1992.

[8] O. Tsatalos and Y. Ioannidis. A Unifying Scheme for Multi-Level Access Paths and Materialized Views, February 1993. Submitted for publication.

[9] J. Wiener and Y. Ioannidis. A Moose and a Fox Can Aid Scientists with Data Management Problems, 1993. Submitted for publication.