# Chapter 3
# Desk-Top Schema Management

## 3.1 Introduction

By definition, a DBMS manages data. By necessity, such a system must also manage the database schemas that describe the organization of the data. Several trends are increasing the need for sophisticated schema management. First, more people are interacting with schemas than in the past. Previously, schemas were mainly used by database administrators (DBAs) as tools for database design. Average users would only occasionally interact with schemas, viewing them only as a reference to look up names of attributes and relations when forming queries. As computers became cheaper, more novice users have gained access to databases. At the same time, DBAs have become more expensive, thus novice users often must manage their databases (including designing their schemas) by themselves. In addition, the data being managed is becoming more complex, resulting in more intricate schemas. As shown in the previous chapter, many systems provide visual schema representations that are editable via direct-manipulation to better support schema design for database novices, and for the design of complex schemas. This type of interface does make schemas easier to work with, though it also increases the role of schemas; these visual schema representations are good templates for querying and browsing, reducing the amount of information about database structure that the user must remember.

The increased user base, usage, and complexity of schemas demands tools to permit their visualization and editing. The term *Schema Manager* is used to describe such a tool, supporting interaction with schemas for all their various users and roles. This chapter discusses the types of interaction between users and schemas, and the functionality required to support such interaction. The term *Desk-Top* Schema Management is used because this work is oriented towards tools for computers on the desk-tops of average users as well as database specialists.

## 3.2 Motivating Application

The need for a DTSM was originally seen in the context of the ZOO project (Figure 3.1), an ongoing effort to develop a Desk-Top Experiment Management System. The goal of ZOO is to enable scientists to manage the entire life-cycle of their experimental studies via a single uniform desk-top interface. This is achieved by placing the conceptual/logical schema of the experiment data at the center of ZOO. Whether designing a study, invoking an experiment, querying the data, or analyzing query results, a graphical presentation of the schema is used to perform the activity; in essence, the schema captures the experimental study itself. Schemas in ZOO are based on the MOOSE object-oriented data model [WI93], which is similar to many other object-oriented or semantic models. (MOOSE schemas are used in many of the examples in this thesis.) The needs of several experimental scientists collaborating in the development of ZOO have led us to understand the types of interaction and required functionality described below.

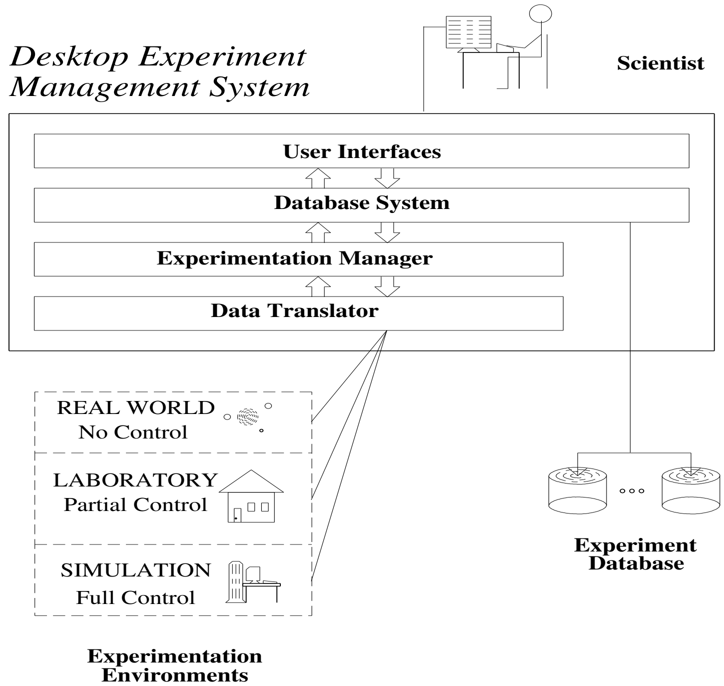


Figure 3.1. An overview of the ZOO Desk-Top Experiment Management System.

## 3.3 Modes of User Interaction

In order to explain the interaction between users and schemas, it is first necessary to explain some of the terminology: *schema visualization* describes a representation of the information of a schema that a person can look at, or the process of creating such a representation from a schema. *Visual style* refers to the general form of a schema visualization. For example, Figure 2.12 itself is a schema visualization. Figure 2.12 uses a graph-like visual style to represent a MOOSE schema, with nodes representing object classes and arcs representing relationships.

A user of a Desk-Top Schema Manager may interact visually with the system in one of the following three modes: *creation/modification* of visual styles, *creation/modification* of schema visualizations (with analogous effect on the underlying schemas), and *exploration* of schemas. The last two do not necessarily occur independently, but their separation helps in identifying their properties.

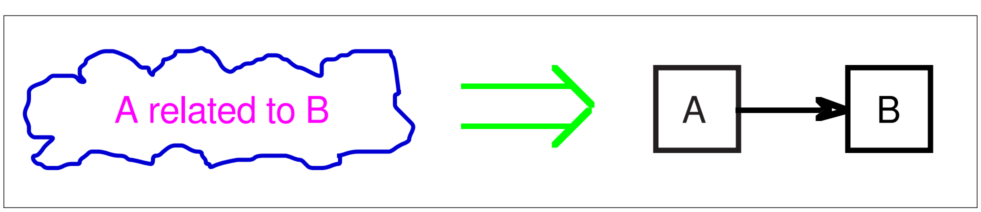


Figure 3.2. An example of visual style creation.

In visual style creation and modification, a user describes new ways to visualize schemas, being able to follow any personal ‘look and feel’ desired. These are then used during the other two modes of interaction for visual representation of schemas. Figure 3.2 gives an example of visual style creation, where the abstract idea of something called A, another thing called B, and a relationship between, is turned into a visual style of A and B as boxes, and the relationship as an arrow between.

In schema creation and modification, a user builds a visualization that conveys the structure of the underlying data schema, as well as any personal information and aesthetic preferences, by visual manipulations that, based on their effects, may be classified as:

- *Editing,* which affects information in the underlying data schema,
- *Enhancing,* which affects personal annotations but not the data schema, and
- *Embellishing,* which affects only the aesthetics of the visualization.

Examples of these three categories of schema modification are shown below. Figure 3.3 shows an Editing change, where a relationship is added to the schema between A and B, changing the schema with respect to the database. Figure 3.4 shows an Enhancement, where the user decides that B is more important, and changes the color to reflect that; this change has no effect on the schema seen by the database. Figure 3.5 shows a modification that is Embellishment. The change in locations of A and B improves the appearance of the schema for the user, but it does not affect the schema seen by the database. The meaning of the change in layout of A and B may be difficult to express in terms of simple attributes such as importance, but it is nonetheless important and should be preserved. The distinction between Enhancing and Embellishing is subtle; section 4.6 examines this distinction in more detail and describes the difference between the two.

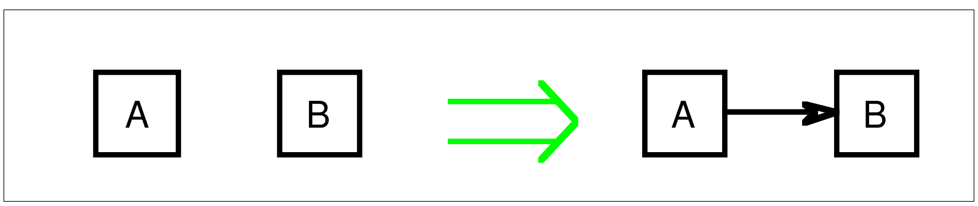


Figure 3.3. An example of Editing.

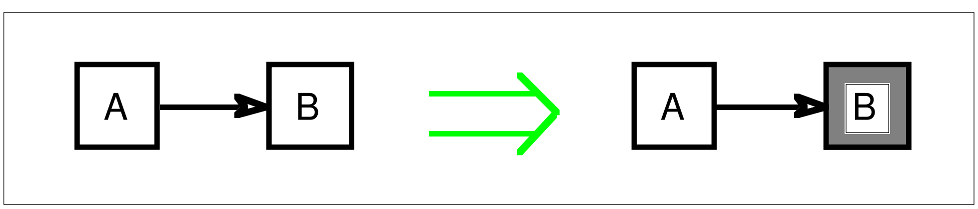


Figure 3.4. An example of Enhancing.

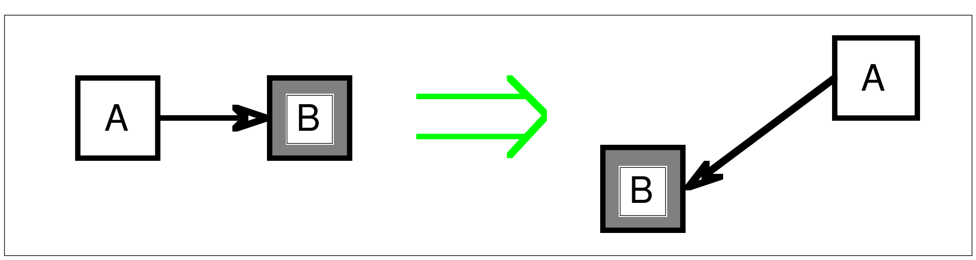


Figure 3.5. An example of Embellishing.

In schema exploration, a user simply navigates through the schema. The most powerful tool to aid this task is dynamic, fine-grained flexibility of visual representation, allowing a schema to be visualized using arbitrary styles. Other useful tools include hard copy output (paper provides portability, higher resolution, and the ability to piece together visualizations larger than any computer screen), as well as traditional operations such as zooming and panning.

## 3.4 Functionality

Given the schema management needs discussed in the introduction to this chapter, and the modes of interaction described in the previous section, the user of a DTSM should be offered functionality and flexibility that is not found in current database GUIs. A DTSM must, like existing database interfaces, support creation of schemas through direct manipulation of schema visualizations, as well as four additional capabilities:

1. It must support evolution in the fundamental visual styles. Over time, users may find existing visual styles unsatisfactory in bringing out the important aspects of their schemas. They should be allowed to change the styles to reflect this, so the visual styles should not be hard-wired into the system.
2. It must allow users to add personal information to schema visualizations beyond what the database needs. Each (group of) user(s) may have a different view of the high-level organization of the schema and may need to express different concepts on top of the data model. A DTSM must appear to allow a different data model for each user.
3. It must accommodate varying senses of aesthetics among different users. Users should be permitted to change the appearance of schema visualizations, and the system should maintain it to the greatest extent possible.
4. It must provide choice in visual representation. There are many ways to visualize schema information, and users may differ over which they find most intuitive. In addition, visual styles that work well for small schemas may not scale up, and different styles may highlight different aspects of a schema. A DTSM should enable a user to select different representations for the same database schema. This choice should be available in any granularity, so that it can be made independently for individual pieces of the schema, and also dynamically, so that a user can easily flip between different representations during schema exploration.

In providing the above capabilities, a DTSM must fulfill several requirements with respect to usability, scalability, and performance. First, it must be usable by non-experts and should hide database technicalities that are irrelevant to the user's goals, e.g., syntax of database languages. Second, it must be able to manage large and complex schemas and offer abstraction tools that hide details and reveal higher-level structure. Third, it must offer reasonable performance. Although some of these requirements are difficult to measure, feedback from real users should be used to guide design and implementation toward these goals.

## 3.5 Capability Example

The last three desirable capabilities mentioned above are illustrated in the following example. Consider an object-oriented database storing information on various Companies, which consist of Departments, Employees, and Buildings, where all four of these entities are captured as classes. For the purposes of this example, Departments have as attributes their name, the Employees with which they are associated, and one or more floors on which they are located. Employees have a name, a salary, a home address, a Department with which they are associated, and a floor on which they work. Buildings have a name, an address, and one or more floors. Floors have a number and a boolean attribute indicating whether or not smoking is permitted on the floor. Addresses have a street and a city. Figure 3.6 shows a visualization of this database schema in the form of a graph, where classes and relationships appear as nodes and edges, respectively.[1]

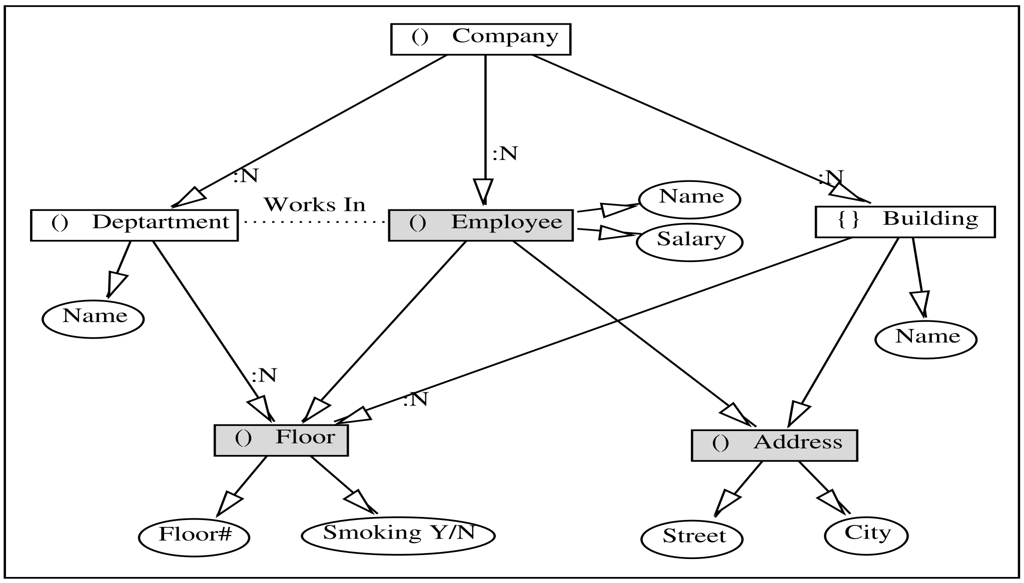


Figure 3.6: A simple schema using a graph representation.

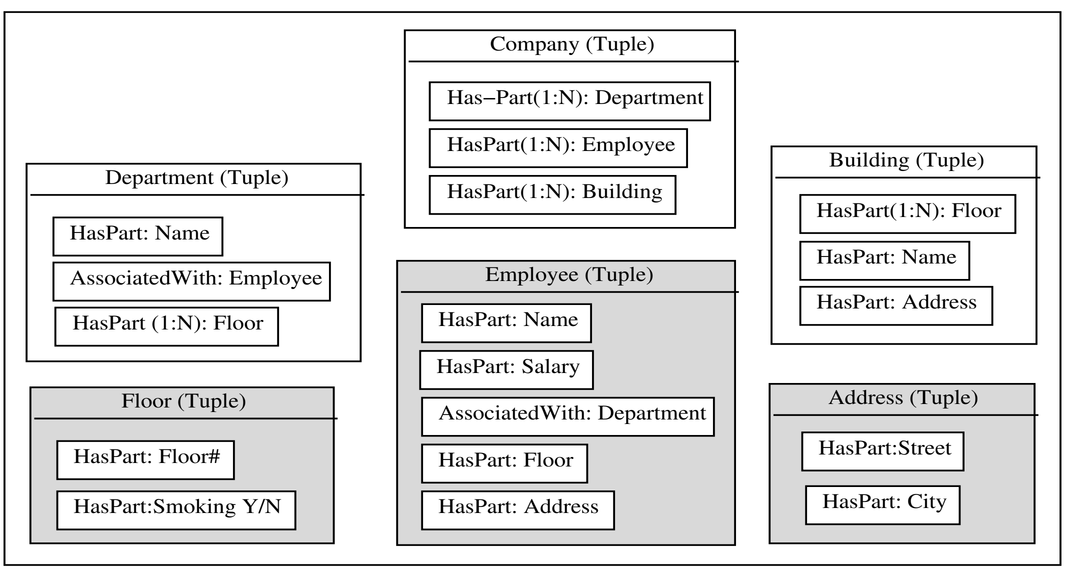


Figure 3.7. The same schema using a containment representation.

[1]All figures in this thesis showing example schemas, except those in Chapters 2 and 4, have been created as PostScript output from OPOSSUM Desk-Top Schema Manager (described in Chapter 5).

Assume that a user wishes to assign to each object class a level of importance. For example, Employees, Addresses, and Floors are of primary importance to the user, while the other classes are secondary. A DTSM should allow users to express such information, as for example in Figure 3.6, where the important classes are given a different background color.

Figure 3.7 shows a different visualization of the above schema. This time a containment representation is used, where each class appears as a box containing a set of smaller boxes, one for each relationship it has with another class. Clearly, both visualizations are equivalent in capturing the database schema and the class importance personal information. Nevertheless, it is also clear that there are trade-offs in the quality of the two visual representations. Figure 3.6 is better at showing global structure and transitive relationships. On the other hand, it is worse at showing immediate relationships because, especially in a complex schema, there may be very long edges and many edge crossings. Figure 3.7 brings out local structure better in such cases.

Note that the user may also affect the aesthetics of a visualization: in Figure 3.6, the shared classes, Floor and Address, are moved below the others to make it easier to identify them and to reduce spatial density of edge crossings. The specific placement reflects no schema or personal information, and is therefore not captured in Figure 3.7 where such layout is not necessary in this case.

## 3.6 Summary

This chapter has described the idea of Desk-Top Schema Management, allowing more people to access schemas in more ways. It discussed the interactions that users will have with a Desk-Top Schema Manager, and the functionality a DTSM must have to support those interactions. The next chapter describes a formal foundation on which a DTSM can be built so that it provides this functionality.