

## 5 The MADS Query and Manipulation Languages

Many conceptual paradigms limit their scope to data description, thus assisting users in the database design phase only. In these approaches, users have to turn to another paradigm, usually at the logical level, for querying and data manipulation. This burden can be taken off from users by extending the scope of the conceptual paradigm to include a data manipulation language, for loading and modifying the content of the database, and a query language, for retrieving information from the database. With such a broader set of facilities, users can reason about their world of interest using a single set of concepts all over the lifespan of their applications. In the design phase, users can build a description of their information structures (the conceptual schema) without the need to take into account the multiple implementation restrictions of available GISs or DBMSs. In the operational phase, the languages defined for specifying operations at the conceptual level enable users to insert, update, delete, and query the objects and relationships they have defined in the schema. Similarly, the result to user queries will show objects according to the format defined in the conceptual schema. The user interface thus becomes a homogeneous world governed by a conceptual perception. Although a natural and straightforward idea for improving usability, this is rarely adopted in database management approaches that use CASE tools in the design phase. Indeed, very few conceptual query languages for traditional databases have been proposed, e.g., [Bloesch 95] [Siau 98]. Also, while several visual query languages for spatial databases have been proposed, e.g., [Laurini 04] [Ferri 05], a main issue to be addressed by such languages concerns solving the ambiguity of queries. However, none of these proposals achieve overall consistency between the data definition and the query and data manipulation languages. We regard the targeted consistency as a major contribution of the MADS model for advancing the state of art and practice in information management.

This chapter discusses, defines, and illustrates the basic operators that can support the specification of MADS-compliant query and manipulation languages. The focus of the presentation is on highlighting what are the basic functionalities that must be supported in order to be able to manipulate and query multi-faceted data, i.e., data that may be spatial, temporal, spatio-temporal, and multi-represented. This all-encompassing vision of data characterizes the MADS approach, while existing languages tend to deal with only a subset of the potential facets of data. For example, the spatial facet is addressed by several query languages, including, at the standards level, SQL/MM [ISO 03d], which extends SQL for querying and manipulating spatial data. However, SQL/MM only considers the discrete view of space, which is a strong limitation for many applications. Many languages have

been proposed for temporal databases, but none of them has got widely acceptance. No existing language covers the three dimensions: space (with both views, discrete and continuous), time, and multi-representation.

The goal of this chapter is to get the reader share our understanding of the functionality that is needed to support multi-faceted data. This is why we choose to present the principles and rules that characterize a correct level of support, and then present the basic operators that act as building blocks for the languages to be defined. Relational database systems have perfectly illustrated the fact that two kinds of languages are essential in making a database approach successful. In a temporal sequence, first come formal languages (e.g., the relational algebra), whose role is to precisely (i.e., with no ambiguity) define a sound foundation on top of which other languages can be built. Formal query languages typically split into algebraic approaches or calculus-based approaches. Algebraic approaches, as the one adopted in this chapter, define a set of operators that can be combined into expressions of any complexity to meet user requirements. They reflect a procedural view of data querying, in the sense that algebraic expressions instruct the system on how to compute the desired result. They allow implementing a query execution component for a DBMS kernel via the implementation of the operators included in the language. Calculus-based approaches aim at asserting what is the desired result without having to state how to compute it. They rely on predicate calculus theory and logic languages. Although more appealing from a scientific perspective, because they better conform to the separation of concerns between the conceptual level and the logical level, their practical use is mostly as supporting background theory for visual languages (e.g., Query By Example or QBE).

Once a formal language has been defined for a given data model, user-oriented languages may be built on top of the formal language to provide friendlier user interfaces. This is how, for instance, SQL was designed, aiming at hiding relational algebra. Work on MADS user interfaces has focused on the specification of a visual query language allowing formulating a query by direct manipulation on the screen, mainly through point and click interactions. The intent is to hide the algebraic specifications and free users from the burden of learning the algebraic syntax and rules or any equivalent textual language. This is in line with usability studies showing the superiority (from a user-oriented perspective) of visual query languages with respect to textual SQL-like languages. In the framework of the MurMur project [Parent 06], two visual interfacing tools were developed: a schema editor, providing facilities for visual schema definition, and a query editor, supporting the visual query language. In both cases user actions are automatically translated into expressions of the underlying textual language (the MADS data definition language and the MADS algebra). Readers interested in these tools are referred to [Parent 06] for more details.

In presenting the MADS operators, we deliberately avoid giving a formal description as usually found in research papers. Despite the informal style, for each operator we provide a rigorous definition of its syntax and semantics. Unambiguous answers are given to questions such as what are the operands, what is their use, which predicates can be specified, how the result is defined, and how the in-