The MLPQ System

MLPQ, short for Management of Linear Programming Queries, is a constraint-relational database system that is continuously built and updated at the University of Nebraska-Lincoln since 1995. The initial goal of the MLPQ system was to seamlessly extend traditional relational database systems. For example, MLPQ extends the minimum and maximum aggregation operators, which relational database systems apply only to a finite set of constant values, to operators that find the minimum or the maximum of a linear function over \( k \) attribute variables, which range over a \( k \)-dimensional polyhedral space represented by rational linear inequality constraints in the database. Of course, that is exactly linear programming, something that goes way beyond the normal capabilities of relational databases and was thought before to be a “separate issue.” This separation was unfortunate because raw operations research data needs to be often reformulated before it becomes suitable as input to a linear program solver. MLPQ allows such reformulations to be done by simple database queries.

The MLPQ system was later found to be also excellent as an extension of geographic information systems (GISs). Although GISs normally use the vector representation of geographic data, the vector representation is easily translated into a linear constraint representation. The vector representation becomes increasingly hard to maintain for higher-dimensional data. For instance, a \( k \)-dimensional hypercube can be represented only by \( 2^n \) vertices in the vector data model. However, they can be represented by only \( 2n \) linear constraints. This example only hints at the difficulties that GISs encounter when they try to provide more than 2-dimensional cartographic data management. In fact, to make advances GISs are increasingly forced to translate internally the vector representation into a constraint representation to effi-
ciently evaluate the queries. After the query evaluation is completed, GISs routinely translate back the result into the “normal” vector representation. The MLPQ system can be viewed simply as an advanced GIS that omits this “normal” vector representation and keeps instead internally the constraint representation at all times. The greater generality of the constraint representation provides MLPQ the flexibility to handle three dimensional spatial data and three or four-dimensional spatio-temporal data.

Another application area where MLPQ proves advantageous is the management of moving objects databases. It is becoming increasingly accepted to represent the trajectory of moving objects by linear constraint approximations. These approximation also can be conveniently represented in constraint databases, which are then easily queried by the MLPQ constraint-relational system.

Although MLPQ has wide-ranging capabilities, it integrates all of its parts in a natural manner, using the same graphical user interface and query languages. Even a user who is only familiar with standard SQL can solve some difficult problems as shown by the examples in this chapter.

Section 13.1 gives a brief overview of the MLPQ system architecture and describes the functions of its six main modules. Section 13.2 describes how the MLPQ system can be used as a relational database system. Section 13.3 describes how the MLPQ system can be used as a geographic information system. Section 13.4 describes how the MLPQ system can be used as a moving objects database system. Section 13.5 discusses some topological queries in MLPQ. Section 13.7 considers linear programming. Section 13.6 discusses some applications of recursive SQL and Datalog queries in MLPQ. Finally, Section 13.8 describes how to make MLPQ databases webaccessible.

13.1 The MLPQ System Architecture

An overview of the architecture of the MLPQ system is shown in Figure 13.1. The architecture consists of six main modules, which can be described as follows:

**Representation:** The representation module is responsible for the internal representation of constraint databases and queries. This module communicates with the external constraint database storage structures. If it cannot find some needed constraint relation in the constraint database store, then it calls the approximation module that searches its relational database store and provides a converted constraint relation. The representation module provides output to the query evaluation module. The constraint relations of the input files are like those described in Chapter 4.