Modeling spatial data has grown in importance over the years, but it is still not easily achieved. Spatial data usually does not fit within the constructs of traditional databases and the users of such data have been in narrow fields that have escaped the interest of computer scientists. As such, two broad views of handling spatial data have arisen: the field model and the object model. The field model uses functions to map every point in space into one element from the range of elements. These functions may be piecewise if there are clear boundaries between items, or they may be continuous for an element like elevation which has a smooth gradient. The object model uses geometrical objects like lines, curves and polygons to draw boundaries around items. Attributes within the object are split into spatial and non-spatial. Generally, amorphous items that have no clear boundaries are field modeled (like a spreading fire), while the object model is used for elements with clear boundaries that can be represented with stationary geometry.

The field model requires a spatial framework, field functions and field operations in order to be appropriately defined. The spatial framework is the methodology used to locate and place elements. Commonly latitude and longitude are used. The field functions define how elements are to be placed within the framework. The field operations define how spatial objects are related, of which there can be three. Local operations define the value of a new field based on the input from that same location. Focal operations work on a small neighborhood of items around the element. For example, an infinitesimal used when calculating slopes. Zonal operations work on larger areas and perform aggregation type functions.

In order to represent the spatial data, a common set of geometric types need to be used that can construct more complicated shapes. These types also interact with other types in various ways. Set-oriented interactions are simple union, intersection, membership type operations. Topological operations deal with transformed data. One example is the within function. Even if a shape is deformed, it still contains the same items. Directional operations handle the orientation of the type in relation to others, in an absolute sense and in terms of the viewer’s perspective. Metrics and Euclidean operations are based on measurements and traditional notions of addition and multiplication. The aforementioned operations all operate in a static fashion, such that they do not change the dataset. Dynamic operations, like merge and split will actually change the data. Both types need to be supported in spatial databases.

In order to describe databases, the ER Model is used which captures information regarding tables and their attributes as well as the kind of relationships it may have to other tables. The ER model easily maps to the relational model used in creating relational databases. To extend this model to spatial data, extra information needs to be encoded, which can be done pictographically. Adding certain diagrams to the entities can clarify its spatial relationship to other entities implicitly while also describing the data type held within.
Database System Concepts: Chapter 1

The basic relational model is described in detail in Database System Concepts. A database consists of a set of tables, called relations. Each relation is comprised of attributes, the values of which are taken from a specific domain for that attribute. A relation is defined as a subset of the Cartesian product of items from different domains. Each row in this list is called a tuple, the ordering of which is not important. Each attribute is considered to be atomic, so that it is a scalar value.

In order to describe a relation, a schema is created which acts as the data type for the relation. It consists of a list of attributes and their possible domains. Instantiating a schema results in a relation.

Keys are required in relations to uniquely identify tuples. A superkey is a set of one or more items which can be used to uniquely identify a tuple. There may be different possible combinations of attributes within a relation which can be used as a superkey. These are all termed candidate keys. When one is chosen, it is called the primary key. Such a key should rarely change once created for a tuple and it should never contain data that can exist in another tuple in the same form. Keys are used to create relationships between relations. When a key in a relation originates in another relation, it is called a foreign key and allows for referencing of the other table.