data on a variety of advanced database systems. Chapter 11 describes major data mining applications as well as typical commercial data mining systems. Criteria for choosing a data mining system are also provided.

1.7 Data Mining Task Primitives

Each user will have a **data mining task** in mind, that is, some form of data analysis that he or she would like to have performed. A data mining task can be specified in the form of a **data mining query**, which is input to the data mining system. A data mining query is defined in terms of **data mining task primitives**. These primitives allow the user to *interactively* communicate with the data mining system during discovery in order to direct the mining process, or examine the findings from different angles or depths. The data mining primitives specify the following, as illustrated in Figure 1.13.

- The set of *task-relevant data* to be mined: This specifies the portions of the database or the set of data in which the user is interested. This includes the database attributes or data warehouse dimensions of interest (referred to as the *relevant attributes or dimensions*).
- The *kind of knowledge* to be mined: This specifies the *data mining functions* to be performed, such as characterization, discrimination, association or correlation analysis, classification, prediction, clustering, outlier analysis, or evolution analysis.
- The *background knowledge* to be used in the discovery process: This knowledge about the domain to be mined is useful for guiding the knowledge discovery process and for evaluating the patterns found. *Concept hierarchies* are a popular form of background knowledge, which allow data to be mined at multiple levels of abstraction. An example of a concept hierarchy for the attribute (or dimension) *age* is shown in Figure 1.14. User beliefs regarding relationships in the data are another form of background knowledge.
- The *interestingness measures and thresholds* for pattern evaluation: They may be used to guide the mining process or, after discovery, to evaluate the discovered patterns. Different kinds of knowledge may have different interestingness measures. For example, interestingness measures for association rules include *support* and *confidence*. Rules whose support and confidence values are below user-specified thresholds are considered uninteresting.
- The expected *representation for visualizing* the discovered patterns: This refers to the form in which discovered patterns are to be displayed, which may include rules, tables, charts, graphs, decision trees, and cubes.

A **data mining query language** can be designed to incorporate these primitives, allowing users to flexibly interact with data mining systems. Having a data mining query language provides a foundation on which user-friendly graphical interfaces can be built.

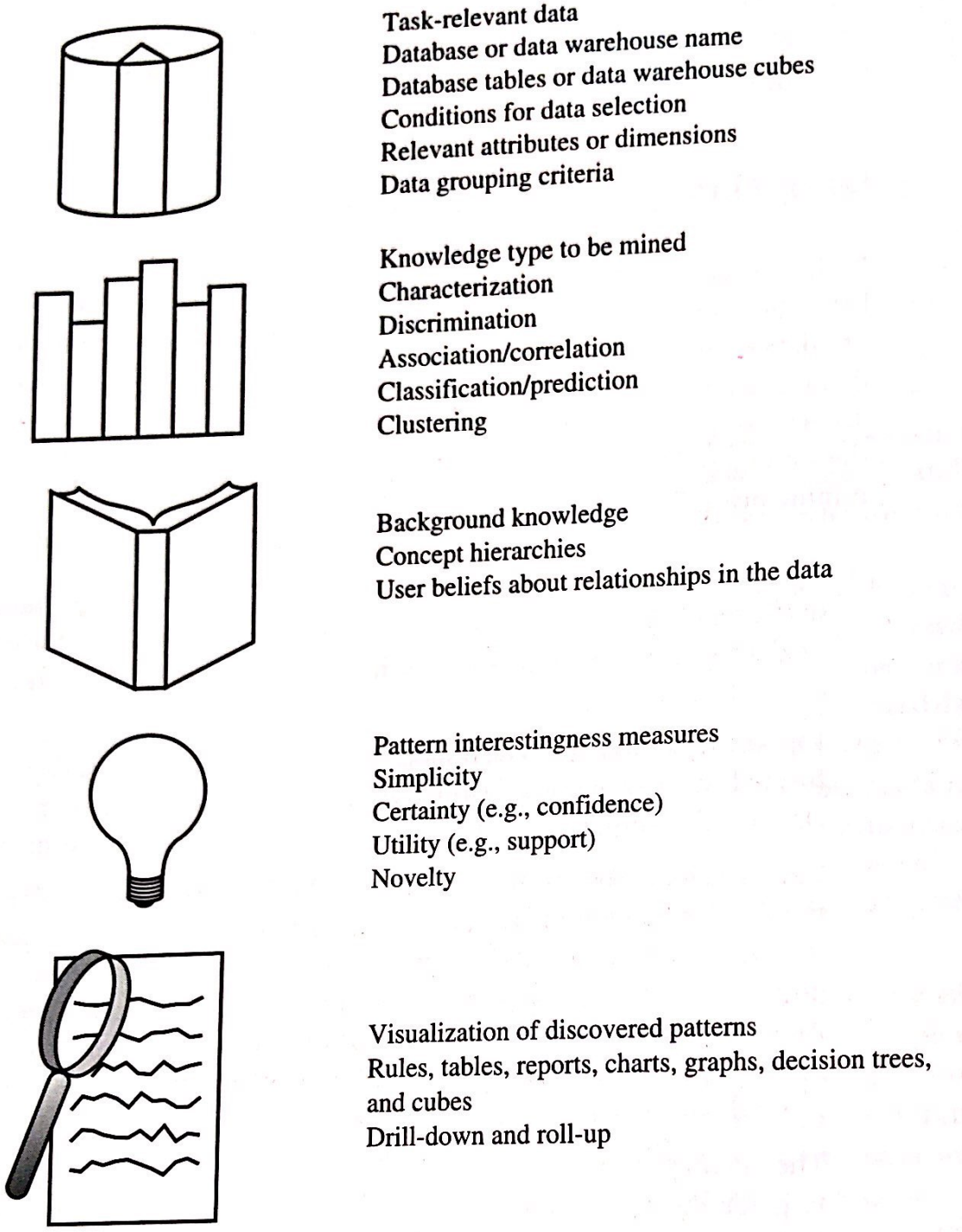

Figure 1.13 Primitives for specifying a data mining task.

This facilitates a data mining system's communication with other information systems and its integration with the overall information processing environment.

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks, from data characterization to evolution analysis. Each task has different requirements. The design of an effective data mining query language requires a deep understanding of the power, limitation, and underlying mechanisms of the various kinds of data mining tasks.

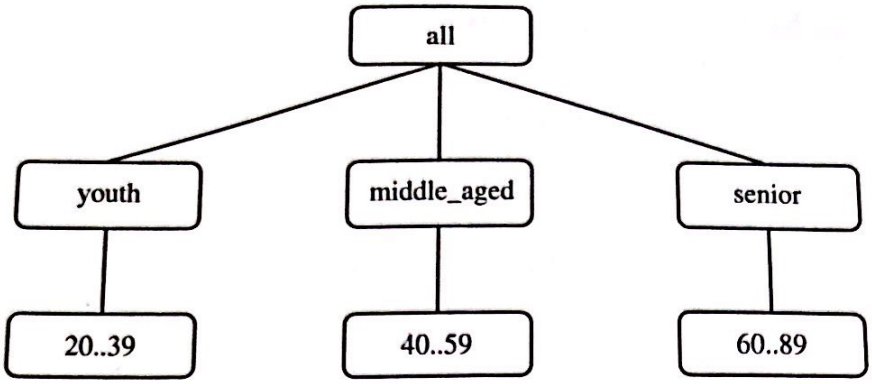

Figure 1.14 A concept hierarchy for the attribute (or dimension) *age*. The root node represents the most general abstraction level, denoted as all.

There are several proposals on data mining languages and standards. In this book, we use a data mining query language known as DMQL (Data Mining Query Language), which was designed as a teaching tool, based on the above primitives. Examples of its use to specify data mining queries appear throughout this book. The language adopts an SQL-like syntax, so that it can easily be integrated with the relational query language, SQL. Let's look at how it can be used to specify a data mining task.

Example 1.11 **Mining classification rules.** Suppose, as a marketing manager of *AllElectronics*, you would like to classify customers based on their buying patterns. You are especially interested in those customers whose salary is no less than $40,000, and who have bought more than $1,000 worth of items, each of which is priced at no less than $100. In particular, you are interested in the customer's age, income, the types of items purchased, the purchase location, and where the items were made. You would like to view the resulting classification in the form of rules. This data mining query is expressed in DMQL[3] as follows, where each line of the query has been enumerated to aid in our discussion.

```
(1) use database AllElectronics_db
(2) use hierarchy location_hierarchy for T.branch, age_hierarchy for C.age
(3) mine classification as promising_customers
(4) in relevance to C.age, C.income, I.type, I.place_made, T.branch
(5) from customer C, item I, transaction T
(6) where I.item_ID = T.item_ID and C.cust_ID = T.cust_ID
        and C.income ≥ 40,000 and I.price ≥ 100
(7) group by T.cust_ID
```

[3]Note that in this book, query language keywords are displayed in sans serif font.

(8) **having** sum(I.price) $\geq$ 1,000
(9) **display as** rules

The data mining query is parsed to form an SQL query that retrieves the set of task-relevant data specified by lines 1 and 4 to 8. That is, line 1 specifies the *All-Electronics* database, line 4 lists the relevant attributes (i.e., on which mining is to be performed) from the relations specified in line 5 for the conditions given in lines 6 and 7. Line 2 specifies that the concept hierarchies *location_hierarchy* and *age_hierarchy* be used as background knowledge to generalize branch locations and customer age values, respectively. Line 3 specifies that the kind of knowledge to be mined for this task is classification. Note that we want to generate a classification model for "promising_customers" versus "non_promising_customers." In classification, often, an attribute may be specified as the **class label attribute**, whose values *explicitly* represent the classes. However, in this example, the two classes are *implicit*. That is, the specified data are retrieved and considered examples of "promising_customers," whereas the remaining customers in the customer table are considered as "non-promising_customers." Classification is performed based on this training set. Line 9 specifies that the mining results are to be displayed as a set of rules. Several detailed classification methods are introduced in Chapter 6. ■

There is no standard data mining query language as of yet; however, researchers and industry have been making good progress in this direction. Microsoft's OLE DB for Data Mining (described in the appendix of this book) includes DMX, an XML-styled data mining language. Other standardization efforts include PMML (Programming data Model Markup Language) and CRISP-DM (CRoss-Industry Standard Process for Data Mining).

1.8 Integration of a Data Mining System with a Database or Data Warehouse System

Section 1.2 outlined the major components of the architecture for a typical data mining system (Figure 1.5). A good system architecture will facilitate the data mining system to make best use of the software environment, accomplish data mining tasks in an efficient and timely manner, interoperate and exchange information with other information systems, be adaptable to users' diverse requirements, and evolve with time.

A critical question in the design of a data mining (DM) system is how to integrate or *couple* the DM system with a database (DB) system and/or a data warehouse (DW) system. If a DM system works as a stand-alone system or is embedded in an application program, there are no DB or DW systems with which it has to communicate. This simple scheme is called *no coupling*, where the main focus of the DM design rests on developing effective and efficient algorithms for mining the available data sets. However, when a DM system works in an environment that requires it to communicate with other information system components, such as DB and DW systems, possible integration schemes include