CHAPTER 1

Introduction

Exercises

1.2 Answer: Two disadvantages associated with database systems are listed below.

a. Setup of the database system requires more knowledge, money, skills, and time.

b. The complexity of the database may result in poor performance.

1.6 Answer: Programming language classification:

- Procedural: C, C++, Java, Basic, Fortran, Cobol, Pascal
- Non-procedural: Lisp and Prolog

Note: Lisp and Prolog support some procedural constructs, but the core of both these languages is non-procedural.

In theory, non-procedural languages are easier to learn, because they let the programmer concentrate on *what* needs to be done, rather than *how* to do it. This is not always true in practice, especially if procedural languages are learned first.

1.7 Answer: Six major steps in setting up a database for a particular enterprise are:

- Define the high level requirements of the enterprise (this step generates a document known as the system requirements specification.)
- Define a model containing all appropriate types of data and data relationships.
- Define the integrity constraints on the data.
- Define the physical level.
- For each known problem to be solved on a regular basis (e.g., tasks to be carried out by clerks or Web users) define a user interface to carry out the task, and write the necessary application programs to implement the user interface.

- Create/initialize the database.

1.8 Answer: Let *tgrid* be a two-dimensional integer array of size $n \times m$.

a.
- The physical level would simply be $m \times n$ (probably consecutive) storage locations of whatever size is specified by the implementation (e.g., 32 bits each).
- The conceptual level is a grid of boxes, each possibly containing an integer, which is *n* boxes high by *m* boxes wide.
- There are $2^{m \times n}$ possible views. For example, a view might be the entire array, or particular row of the array, or all *n* rows but only columns 1 through *i*.

b.
- Consider the following Pascal declarations:

  type *tgrid* = **array**[1..*n*, 1..*m*] **of integer**;
  var *vgrid1*, *vgrid2* : *tgrid*

  Then *tgrid* is a schema, whereas the value of variables *vgrid1* and *vgrid2* are instances.
- To illustrate further, consider the schema **array**[1..2, 1..2] **of integer**. Two instances of this scheme are:

1	16
7	89

17	90
412	8