DATA BASE ARCHITECTURE

IVAN FLORES
Computer Consultant
Professor of Computer Methodology
City University of New York

Contents

Van Nostrand Reinhold Company Regional Offices:
New York Cincinnati Atlanta Dallas San Francisco

Van Nostrand Reinhold Company International Offices:
London Toronto Melbourne

Library of Congress Catalog Card Number: 80-29564
ISBN: 0-442-22729-9

Manufactured in the United States of America

Published by Van Nostrand Reinhold Company
135 West 50th Street, New York, N.Y. 10020

Published simultaneously in Canada by Van Nostrand Reinhold Ltd.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging in Publication Data

Flores, Ivan.
Data base architecture.

Includes index.
1. Data base management. I. Title.
QA76.9.D3F62 001.64'2 80-29564.
ISBN 0-442-22729-9

I
FUNDAMENTALS
1
Introduction

This book describes the structure and use of information when it is organized into a data base. The approach is basic, but the reader should have some background in the structure of data. Along the way some topics are quickly reviewed and a uniform terminology is presented.

This chapter is an introduction to the Data Base Management System (DBMS), which reviews data and sets forth the concept of, and need for, the data base. Section 1.1 reviews data and its properties. Section 1.2 defines the application, the application program, a group of similarly oriented application programs called the application family and the files they need, an application file group, all in the nonDBMS setting.

Section 1.3 makes clear how the DBMS can facilitate storage and access for the typical application family. Section 1.4 puts the DBMS into perspective for the overall computer system. Section 1.5 describes what is meant by architecture in several contexts. The need for a schema and subschema approach is discussed in light of the application family in Section 1.6.

Finally, some of the factors affecting DBMS design are examined. The activities required to use and maintain the data base are discussed in Section 1.7. The aims and benefits which accrue from the use of a DBMS are examined in Section 1.8.

1.1 DATA

The overall objective of the Data Base Management System is to control and use data effectively. Therefore, an important purpose of this introduction is to review what data is and how it may be used.

What is Data

Data is a representation of the real world. We can't expect it to represent the entire world. Only the portion of particular interest to one or more applications is represented. One portion of the real world containing individuals of a like type is called a **population**. The population consists of individuals. An **individual**

may be one of a group of people, as in a personnel application where the individuals of the population consist of employees.

Individuals may be entities or objects rather than people. Thus, for an accounts receivable application, individuals are companies, the account, the client. For an inventory application, the individuals are parts for which we wish to assure that there is a minimum number in stock.

Each individual has properties, some of which are of interest to us and some of which are not. A personnel file might record an employee's height, weight, sex, hair color and so forth, but not his car type nor his bank account number.

Then data is a representation in the computer of a portion of the real world called the *population* and describes certain properties of the individual members of this population.

Quanta of Data

Let us review terms which describe units of data. For this introduction, the discussion is oversimplified to the situation which prevailed before the advent of data bases but is put into perspective shortly. The units of data are presented pictorially in Figure 1.1.1, where we see an individual in his population. The individual has characteristics called **attributes.** An example is an attribute named *HEIGHT.* An **attribute name** is the *name* of an attribute which is described in the data base, or file–in this case, *HEIGHT.* In the figure the individual has a height described as 5'10". We say that 5'10" is the **attribute value.**

Corresponding to each attribute of interest there is a quantity of data called the **field.** For this field, there is a field name and a field value. The **field name,** often the same as the attribute name (but it need not be), helps us refer to this field. The **field value** is some representation, appearing in the field, which might be binary, decimal, hexdecimal, etc., of the original attribute value for this individual.

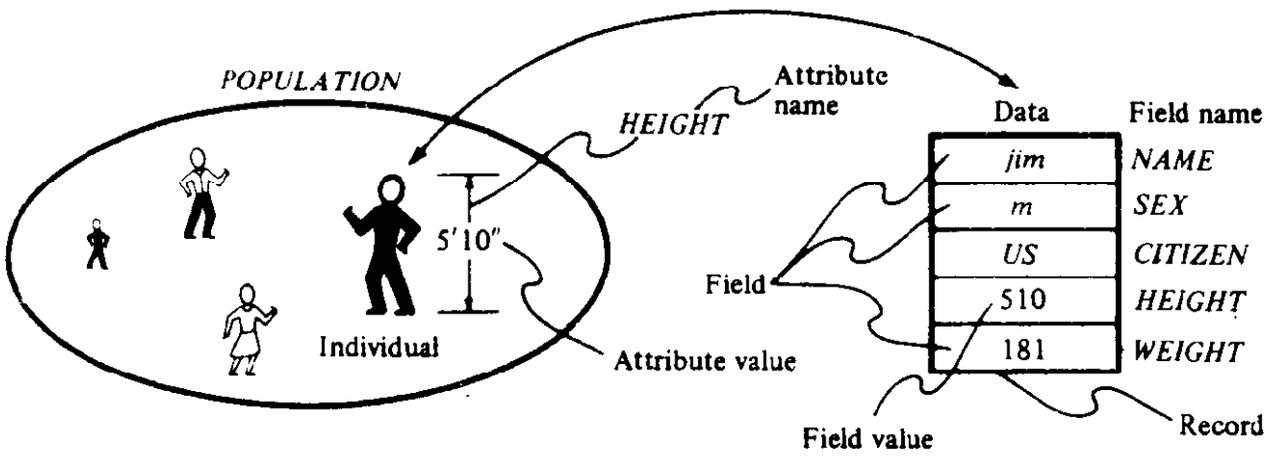

Figure 1.1.1. Data quanta and their relation to the population.

Figure 1.1.1 shows a collection of all applicable fields for one individual, called a record. The **record**, therefore, is a collection of field values for a single individual. Field names are not usually contained in the record; we use them to identify and refer to fields.

How is Data Used?

Before the advent of data bases each user or user group "owned" and was responsible for his own data and for keeping it up to date. The important thing to recognize about a population is that it changes in several ways which we shall examine. There is hardly a population of importance which does not change in some way; one which does not change at all is called an **archive**. If we use an obsolete representation of today's population, processing this data will produce erroneous results. Therefore, it is important that the representation be as current as possible.

We have called a collection of individuals a *population*. For each individual there is a record and there should be as many records as individuals. This collection of records is called the **file** to distinguish it from the collection of individuals.

The user may elect to use this file in one of three ways, which are illustrated in Figure 1.1.2. These are:

- **retrieve**–examine one or more records to extract information but not otherwise affect the record of the file;
- **post**–alter a *record* because the individual has changed and the record should now reflect the altered attributes of the individual;
- **maintain**–not only do individuals change but so does the population, by the addition and deletion of individuals–so maintenance alters the *file* to reflect the alteration in the population.

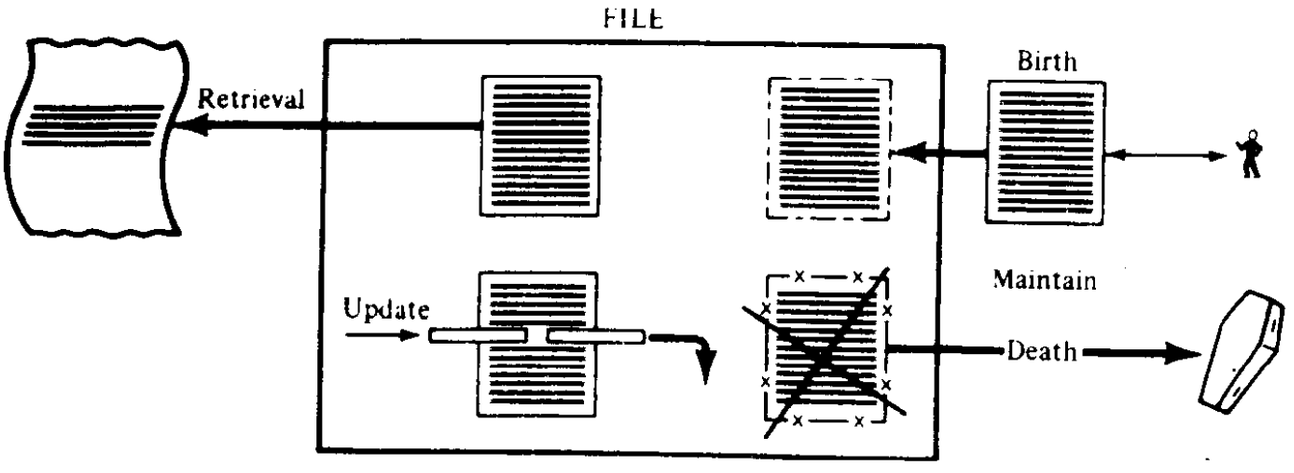

Figure 1.1.2. Functions performed on the file.

Retrieve

For retrieval a user needs information about one or more records. For instance, an account holder may wish to know the current balance in his account. This access to the account holder's record does not alter the record in any way. Information is copied from the record, formated and printed or displayed to the user.

Post

To post or modify a file, the system is informed of which individuals have incurred a change, which attributes have changed and what the new values are for these attributes. Posting then consists of finding a record for an individual for which a change has occurred in one of his attribute values and replacing old field values with new field values, representations of the new attribute values. This is shown in Figure 1.1.2 in the center. Posting, according to this definition, changes *records, not files.*

Maintain

Maintenance means adding or deleting records from the file rather than simply altering some of the records which comprise the file. Consider the employees on our staff. These employees come and go as they are hired and fired. We need new records for hired employees; we can (sometimes) dispose of records for employees that have departed. Maintenance consists of adding (birth) or deleting (death) these records to or from our file to perceive an altered population.

Other Features

File design depends not only on the kind of activities needed but also upon the point in time when each needs to be done. When a number of requests for access to a file are saved and grouped together to be handled at one time, this is called **batch processing**. The file is processed at certain times only, and may be unavailable at other times.

In contrast there is **real time processing**. When a savings account holder requests his balance, we give him an immediate answer–the file is always available. Some activities may use batch processing while others use real time processing; whenever a request appears it is immediately scheduled to be acted upon–it is not saved.

Thus for a banking application, retrieval may use real time processing, while updating and maintaining may be done daily on a batch basis.

A second feature of importance is the **frequency of use**. This is applicable in batch processing where a file need be available only when it is in use.

Finally, there is the **activity ratio** or **turnover** which describes the percentage of the file involved in batch processing each time the file is put into use. For example, in accounts receivable we may find that 30% of our clients are billable during any billing period, whereas, for payroll, almost all our employees receive checks during a given pay period.

1.2 APPLICATIONS

The Application

An **application** is some instance when one or more than one file is used. The use may be retrieval, posting, maintenance or creation. An application requires the presence of a program which controls the computer; it provides instructions for whatever actions place the data in the new required state. This program is generally written by a programmer in a **procedure oriented language (POL)**, sometimes called a compiler language. Before the program can run, it must be translated into machine language and debugged.

When a data base management system (DBMS) is in use, again, there is a need to access files. However, many DBMS's have a built-in facility so that unsophisticated users can "talk" directly to them without writing a program. This facility is called a **query processing facility**. Here DBMS accepts the user's request directly and provides answers directly to him.

Hereafter, when we refer to an application, and we will do that often, we refer to the use of one or more files which require the writing of a program. This program is called the **application program.**

Terminal Application Program

A **terminal application program** accesses, posts, modifies or creates files through the data base and has the additional properties that it can communicate with an on-line user by means of one or more terminals. Thus the data retrieved by the terminal application program may be sent to a viewer via the terminal. The operator may also enter data directly at the terminal. The terminal program is written by an application programmer. It is separate and apart from any query processing facility that the DBMS may provide. This facility is discussed later.

We now continue to examine files, discussing the file group and the file family initially in the nonDBMS environment. In this way we clarify the structures before viewing them in the DBMS environment.

section/time
section/room
department/professor
professor/room (office)
department/student (major)

Relations within One Population

We have examined the prerequisite relation which should be included in the XEDB. Another such relation is that of administration. In the academic world professors administer; even the university president is a professor in some department.

11.2 XEDB GRAPH

The first order of business is to develop a graph to convey visually the relations which prevail within and between populations. To do this:

1. first partition each population into its parts;
2. then superimpose the relations between populations;
3. finally graph the relations within populations.

Vertical Graph

First develop a graph which shows how each population is divided into smaller and smaller parts, Figure 11.2.1. On the left we see *XEDB* pointing to an entry

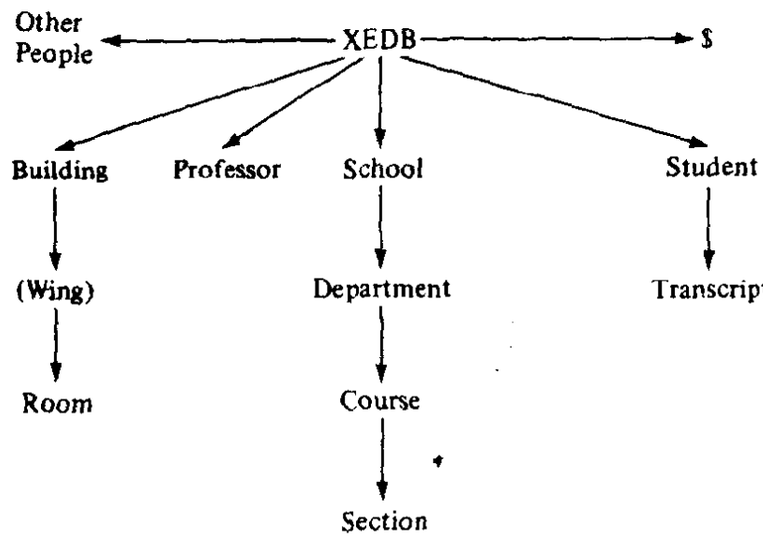

Figure 11.2.1. Graph of the extended educational data base, XEDB, showing only the population partitions.

the file represents. These three circles are also encircled to indicate that they form a **population group**. In our inventory example, the populations are the stock reports, the vendor information and the purchase orders.

We shall see later how the DBMS gives the application program a unified view of all the pictured files. This view is called the **subschema** or **external model**. The program works with this *single* representation of the data instead of three or more files as formerly.

Application Family

It is rare that a commercial data processing problem can be handled completely and properly by a single application program. Most commercial applications have many files, representing many populations, and several application programs are provided to do the whole job.

Figure 1.2.2 illustrates this. A tangible example is forthcoming subsequently. In the figure we see three application programs, as conveyed by the hexagons. These programs comprise the **application family**.

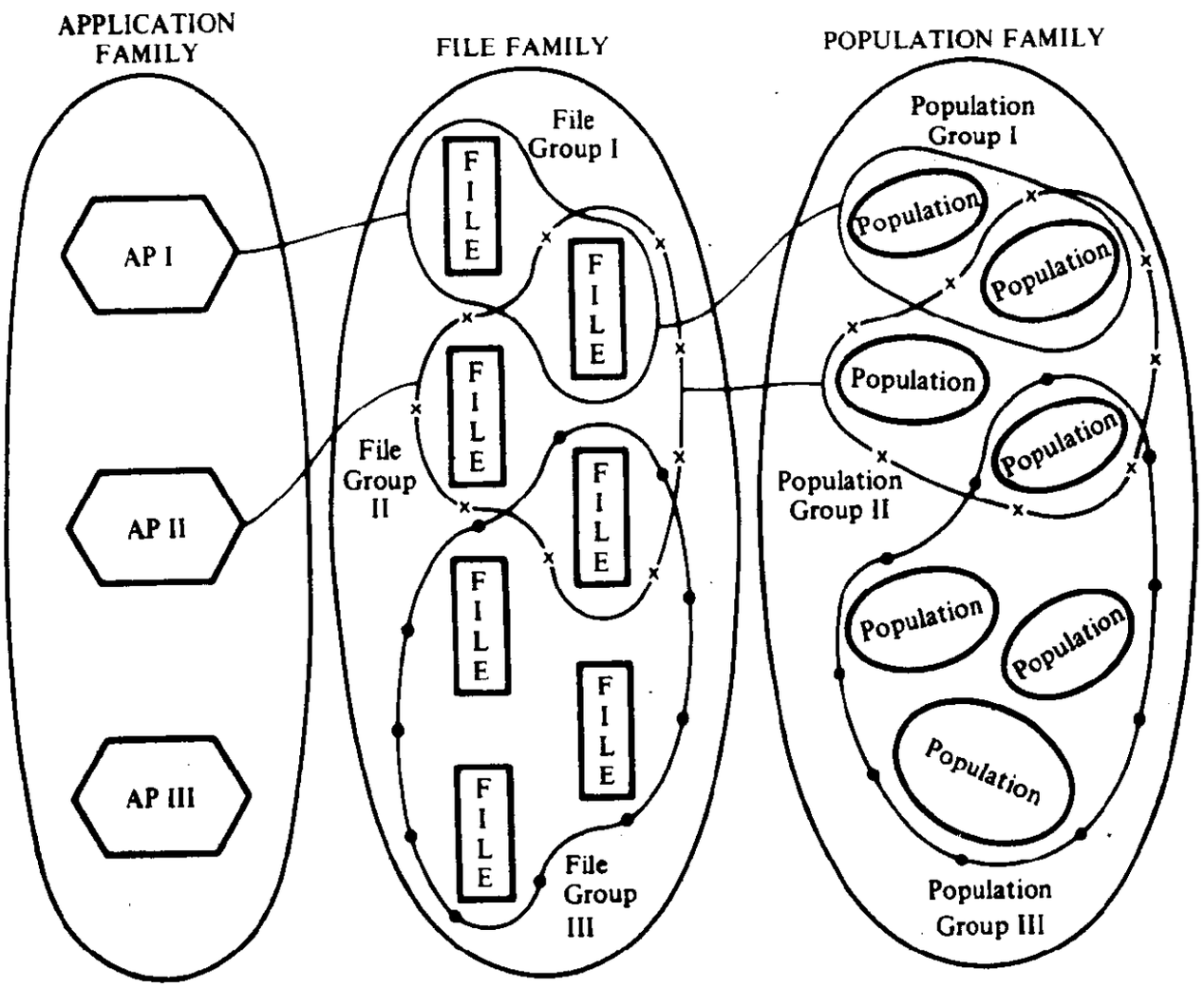

Figure 1.2.2. Several applications form an application family, for which a file family and population family is defined.

Each application program has a set of files that it works with, its file group. These file groups may overlap, as shown in the figure; one file may be associated with two or more application groups.

The collection of these files is called the **file family**. The file family is then defined as containing a set of files where each file is used by some application program in the application family.

Each file corresponds to and represents a population. Then, clearly, there is a group of populations called the **population family** which corresponds to the file family.

Inventory Example

Consider the inventory example. We have examined earlier the stock posting application program. This is only one of the activities required of the inventory family. A simplified version of the inventory family is presented as Figure 1.2.3.

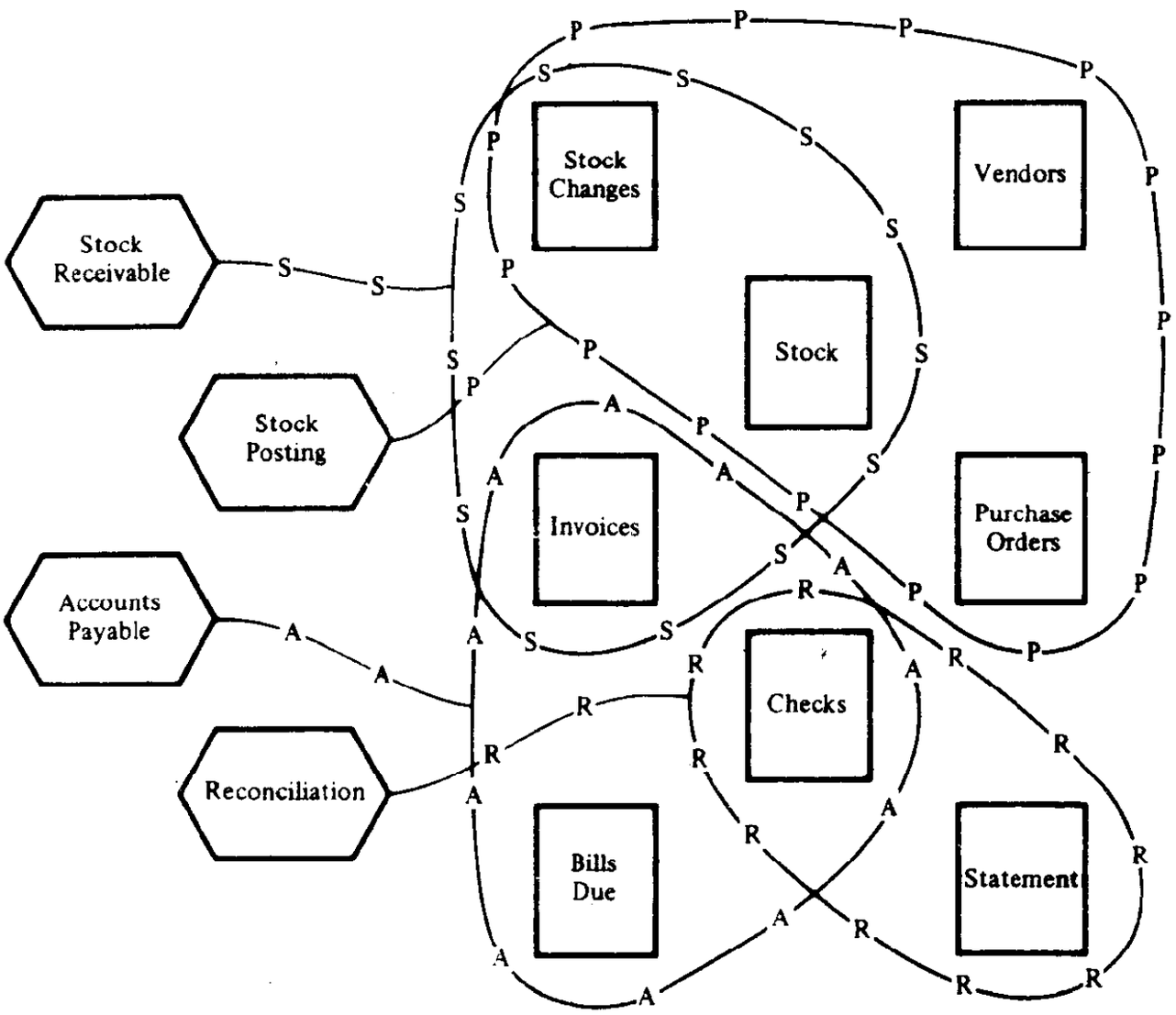

Figure 1.2.3. An inventory application family.

For the stock posting activity, one input is a stock change file. It contains a description of how many items of each kind were physically removed from stock during the posting period. Deliveries were also made to the plant during this period. They are recorded and entered into the system. The stock posting program posts each part record according to the number of parts withdrawn from stock or put on the shelf. When the quantity on hand for a given part reaches the reorder point, the stock posting program finds or calculates a reorder quantity. It then finds the vendor and notes the need for a purchase order.

The purchase order is not issued immediately. Other parts may need to be ordered from the same vendor and it would be wasteful to issue one purchase order for each depleted part. At the end of the first phase of operation all the parts to be ordered from a single vendor are combined into a single purchase order. Then this purchase order is printed by the computer system to be sent to the vendor.

Another application program for the inventory family is run when shipments are received. The items are put into stock and the stock receivable application program goes to work. It correlates the shipping invoice with the purchase order and the report from the stock clerk who has placed the item on the shelf.

Eventually the company is billed for the stock received. During this third phase the bill is correlated with the invoice for the merchandise received and the purchase order originally sent to the vendor. It should be remembered that we don't always get what we order. Some of the merchandise is back-ordered; some merchandise is shipped in quantities less than or greater than that in the original order. As an example of the latter, if we order 100 units, we may be delivered a gross of the item because they are packed in one gross containers.

The accounts payable application program reviews all these data and issues a check to the vendor for the merchandise received, if everything matches up.

Finally, shown at the bottom of the figure, is a reconciliation program which matches up the checks issued against the bank statement, to make sure that the company's bank balance has been correctly figured.

Relations

The application family deals with many populations. Individuals in different populations or, sometimes, within the same population relate to each other with respect to one or more of these applications. What is a "relation"?

This concept is so important that all of Chapter 2 is devoted to it. A relation is a quality which binds together two or more individuals. Let us itemize a few of the relations which prevail among individuals in the inventory example:

1. A bin holds parts for storage.
2. A vendor sells parts to us.

3. A purchase order requests sale and shipment of several different parts
4. An invoice heralds the arrival of a shipment and perhaps a nonshipment (back order).
5. A bill requests payment for one or more shipments.
6. A check provides payment for one or more bills.
7. A statement confirms payment of one or more checks.

The *relation* should be clear in each of these statements for it binds together: *bin* and *part* for (1); *vendor* and *part* for (2); etc.

1.3 NEED FOR THE DATA BASE SYSTEM

Coordination within an Application Family

First consider the problem of file handling. For the files of the application family where no DBMS is present, each application program is separate and distinct. There is no coordination between application programs in the same family. Usually, though not always, application programs are run at different times. Even when two applications are running concurrently under multiprogramming, no coordination is possible.

We have seen how two application programs may have recourse to the same file. In the inventory example of the last section, the purchase order file is needed by three different programs:

- The purchase order is first created by the stock posting program and a record describing it is placed in the purchase order file.
- The stock receivable program is informed when some or all of the merchandise ordered is received and notes this on the purchase order file.
- The accounts receivable program determines that merchandise ordered and received has been paid for and is thus on the purchase order file.

Not only may a single file have multiple users, but each user may have different needs with respect to a file record. That is, different attributes of the same individual are needed for different application program.

Duplicate File Solution

One solution to the problem that a single file has multiple users is to provide a physically different file for each application program. The difficulty is that they not be identical: one may be updated by its application program while the other is not. In some situations this is viable. Consider a personnel file and a payroll file. Both describe individuals employed by our firm. Each contains a record for the individual. However, the characteristics of the individual of interest