CORPORUS CHRISTI STATE UNIVERSITY

MICRO DATA BASE FOR BANKING

A Graduate Project Report
Submitted to the Faculty of the
College of Science and Technology of
Corpus Christy State University
In Partial Fulfillment of the Requirements
For the Degree
Master of Science in
Computer Science

by

Wijit Siripun
March, 1981
This graduate report was approved by Dr. Melvin H. Mckinney as director and by Dr. John W. Stevenson and Dr. Roy S. Ellzey as Advisory Committees.
ACKNOWLEDGMENTS

I wish to thank Dr. Melvin H. McKinney, Dr. John W. Stevenson, and Dr. Roy S. Ellezy for being my committees. I also thank Mr. Eroy W. Wilkins for his informations and guidance.
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................... I

ABSTRACT ......................................................... II

CHAPTER

I. INTRODUCTION ............................................... 1-1

II. FUNCTIONS OF COMMERCIAL BANKS ..................... 2-1

III. THE SCHEMA ................................................ 3-1

IV. THE APPLICATIONS ......................................... 4-1

V. THE SUBROUTINES ........................................... 5-1

REFERENCES .................................................... 6-1
ABSTRACT

Centralized Control and Data Independence are the two main answers to the question "Why data base?". In this project, Micro Data Base System is used to keep the records of saving and checking accounts of a bank. Only the data items needed in the process of saving and checking accounts are kept in the data base. The hardware used in the project is a Z80 based microcomputer with CP/M operating system. The softwares can be divided into two categories. The first one defines the structure of data base(schema) by using Data Description Language (DDL). The second category of software is the application programs used in creating, updating, deleting the records in the data base. The application programs are completely in themself in the process of saving and checking accounts. Another words, they can do everything as the bank in the real world can. The MDBS uses the network approach, so the terminologies used in this project are the Data Base Task Group (DBTG) terms. The host language is Basic and the Data Manipulation Language (DML) is called via assembly language subroutine calls.
CHAPTER 1  INTRODUCTION

In this chapter, the general ideas about this project are presented, and the outlines of the following chapters are mentioned.

Generally, there are three approaches in which records and their relationships, i.e., schema, can be defined:

1. the network approach
2. the hierarchical approach
3. the relational approach

In this project, Micro Data Base System (MDBS) is used as a database management system to develop a data base for banking. MDBS uses the concept of the network approach proposed by CODASYL Data Base Task Group (DBTG) in April, 1971. MDBS is developed by Micro Data Base Inc., Indiana.

Banking has a lot of services served to the customers. But in this project, only saving and checking accounts are the two services chosen from all the services of banking to be the model of the data base. The schema of the data base is shown in Fig. 1.1. The data items in each record type are discussed in chapter 3. According to the schema, there are three SYSTEM record types, and this implies that there are three singular sets, and the data base can be accessed directly by users through the three SYSTEM record types.
FIG 1.1

There are seven application programs used to accomplish the job in saving and checking accounts, and they are listed below:

1. OPEN AND CLOSE ACCOUNTS PROGRAM
2. ENTER TRANSACTIONS PROGRAM
3. UPDATE PROGRAM
4. CALCULATE INTEREST PROGRAM
5. PRINT SAVING STATEMENT PROGRAM
6. PRINT CHECKING STATEMENT PROGRAM
7. READ-ONLY THE DATA BASE PROGRAM
The details of the schema and the application programs are skipped at this point and will be discussed again in chapter 3 and chapter 4.

The rest of this chapter will discuss about the outlines of the following chapters. In chapter 2, the general functions of commercial banks are mentioned, and some facts about economics are discussed in the chapter.

Chapter 3 and chapter 4 are devoted to the details of this project. Chapter 3 concentrates on the schema while chapter 4 concentrates on the application programs. Both chapters sometimes refer to each other.

Chapter 5 discusses about the subroutines used in this project. There are total twelve subroutines in this project, but only the ones considered to be interesting are discussed in details.
CHAPTER 2  FUNCTIONS OF COMMERCIAL BANKS

The business of banking is very broad and far-reaching; with the recent introduction of the one-bank holding company and the possibility of relaxation of some of the restrictions imposed on banking, the number and variety of services provided by commercial banks will probably expand. Recent innovations in banking include the introduction of credit cards, accounting services for business firms, factoring, leasing, participating in the Eurodollar market, and lock box banking. Moreover, many banks are employing management science techniques to improve the many financial services provided. The importance of commercial banks can best be illustrated by a brief explanation of their major function.

1 CREATING MONEY

One of the major functions of commercial banks and a distinguishing feature that separates them from other financial institutions is the ability to create and to destroy money. This is accomplished by the lending and investing activities of commercial banks in cooperation with the Federal Reserve System. The power of the commercial banking systems to create money is of great economic significance. It results in the elastic credit system that is necessary for economic progress at a relatively steady rate of growth. If bank credit were not available, the expansion of our productive facilities and operation would be severely limited. Under such conditions productive units would
be to maintain larger working balances to meet the fluctuating requirements for funds. Such a practice would be uneconomic since large sums would be held idle for some periods, while during the seasonal peaks of business activity such sums might be insufficient.

We need an adequate but not excessive money supply in the economy. If the money supply increases at a more rapid rate than does the production of goods and services, inflation is the result, with all of its ill effects on the various economic processes. Conversely, if the money supply lags behind production, the economy may suffer from deflation with equally undesirable effects. The objective of Federal Reserve policy is to provide a money supply commensurate with the national objectives of stable prices, sound economic growth, and a high level of employment. Commercial banks play a very important role in the implementation of these policies. They serve as a conduit through which the money supply is increased or decreased in an effort to attain these worthy objectives of the economy.

2. PAYMENTS MECHANISM

Providing for a payments mechanism or the transfer of funds is one of the important functions performed by commercial banks, and it is increasing in importance as greater reliance is placed on the used of checks and credit cards. The increasing efficiency with which funds are managed is indicated by the gradual decline in money holdings relative to the gross national product.
over the years, despite an increase in the number of financial transactions. For several years many sources have assumed that approximately 90 percent of the transactions were settled by check, but a greater percentage is probably more realistic. The only kind of money that has increased relative to national income during the past decade and a half is coin, primarily because of the growth of metering and vending machines. Demand deposits are assuming a larger portion of the transaction load, and they are being used more efficiently. This is a result of improved technology, the desire of business firms and individuals to use their funds more efficiently, and the ability of depositors to synchronize their receipts and expenditures.

Most of the checks are cleared through the commercial banking system. Checks drawn on and deposited in the same bank merely transfer funds from one account to another. If only two banks are involved in the same town, there is a direct exchange of checks. When several banks are involved in the same city, a clearinghouse arrangement is usually employed. The process becomes a bit more complicated, time-consuming, and expensive when checks are cleared between banks located in different part of the country. Such clearings are often handled through the correspondent banking system. Checks may also be cleared through regional banks of the Federal Reserve System in a manner similar to clearing through the correspondent banking system. About a third of the checks were cleared through the Federal Reserve System, but in dollar amount these checks accounted for less than 20 percent of the total.
3. POOLING OF SAVINGS

Commercial banks perform a very important service to all sectors of the economy by providing facilities for the pooling of savings and making them available for economically and socially desirable purposes. The saver is rewarded by the payment of interest on his savings, which are safe and in a highly liquid form. These pooled funds are made available to businessmen who may use them for the expansion of their productive capacity and to consumer goods. Our society enjoys a high degree of assluence that enables us to save a substantial portion of income. A large part of these savings flows into the savings departments of commercial banks.

4. EXTENSION OF CREDIT

The primary function of commercial banks is the extension of credit to worthy borrowers. From the beginning of time, organizers of banks have been motivated by the opportunities presented by the lending function, and charters have been granted by governments primarily because there was a need for credit in a particular community. In making credit available, commercial banks are rendering a great social service; through their actions production is increased, capital investsments are expanded, and a higher standard of living is realized.

Bank lending is very important to the economy, for it makes possible the financial of agricultural, commercial, and industrial activities of a country. It makes possible what economists have
call indirect or roundabout production, as compared to direct production where consumable goods are secured by the direct application of labor to land or natural wealth. Bank loans also make possible production for inventory. The food industry provides us with an excellent example; all of the food that is harvested and processed cannot be consumed immediately. Loans to canners, however, enable them to purchase, process, can, and store the food which may at a later time be sold to retailers and ultimately to consumers. During this interval of time—from producer to canner, to wholesaler, to retailer, and finally to consumer—bank loans have made possible the economic handling of the food corp.

5. TRUST SERVICES

Increased incomes have made possible the accumulation of wealth which in turn has contributed to the growth of the trust services of commercial banks. Individuals who have accumulated an estate, even of moderate size, have an interest in providing for the distribution of the assets before death. Many of these individuals have made wills and have asked trust departments to act as executor. Moreover, many of these wills have provided for the creation of personal trusts under which trust departments have the responsibility of investing and caring for the funds and distributing the proceeds as established by the trust agreement.
6. SAFEKEEPING OF VALUABLES

The safekeeping of valuables is one of the oldest service provided by commercial banks. They have vaults that are difficult to enter even by the best of burglars and have established a record of proper custody. The protection of valuables falls into two areas of departments of a bank: safe deposit boxes and safekeeping. Safe deposit boxes are made available to customer on a rental basis. Under such an arrangement the customer has control of his valuables at all times. The bank merely provides the vault, the box, and the other facilities necessary for a proper safe deposit box operation. Finally, and most importantly, the bank controls the access to the vault; i.e., the bank guarantees that the customer who has rented the box or his authorized representative is the only one permitted access.
CHAPTER 3 THE SCHEMA

This chapter presents the details of the schema along with the examples of the schema. First the detail of the record types are discussed in Sec. 3.1 then following by the details of set types in Sec. 3.2. Fig 3.1 illustrates the schema, and Fig 3.2 illustrates the data description language of the schema.

FIG 3.1
0010 FILES SCHEMA.DBS 1 512
0020 DRIVE 1 50
0030 PASSWORDS
 0040 ANAME 255 255 APASSWORD
 0050 BNAME 255 255 BPASSWORD
 0060 CNAMExx 255 255 CPASSWORD
 0070 DNAME 255 255 DPASSWORD
0080 RECORD CUSTOMER
 0090 ITEM SOCIALNO CHAR 011
1000 RECORD SAVACCT
 1010 ITEM SAVINGNO CHAR 008
 1020 ITEM SNAME CHAR 020
 1030 ITEM SSTREET CHAR 025
 1040 ITEM SCITY CHAR 015
 1050 ITEM SSTATE CHAR 002
 1060 ITEM SZIP CHAR 005
 1070 ITEM SAVEPHON CHAR 012
 1080 ITEM SSTATUS CHAR 001
 1090 ITEM SOPENDAT CHAR 008
 1100 ITEM SOBALANC REAL 008
 1110 ITEM SNBALANC REAL 008
 1120 ITEM SACCHARGE REAL 004
 1130 ITEM SINTERES REAL 004
 1140 ITEM STINTERE REAL 004
 1150 ITEM SNWITH INT 002
 1160 ITEM SLASTREP CHAR 008
0270 RECORD CHEKACCT
 0280 ITEM CHEKN0 CHAR 008
 0290 ITEM CNAMExx CHAR 020
 0300 ITEM CSTREET CHAR 025
 0310 ITEM CCITY CHAR 015
 0320 ITEM CSTATE CHAR 002
 0330 ITEM CZIP CHAR 005
 0340 ITEM CHEKPHON CHAR 012
 0350 ITEM CHSTATUS CHAR 001
 0360 ITEM CHPENDAT CHAR 008
 0370 ITEM CBALANC REAL 008
 0380 ITEM CKBALANC REAL 008
 0390 ITEM CACHARGE REAL 004
 0400 ITEM CLASTREP CHAR 008
0410 RECORD SAUXTION
 0420 ITEM SXXDATE CHAR 008
 0430 ITEM SXXCODE CHAR 001
 0440 ITEM SXXAMOUNT REAL 008
0450 RECORD CHXTION
 0460 ITEM CXDATE CHAR 008
 0470 ITEM CXCODE CHAR 001
 0480 ITEM CXAMOUNT REAL 008
0490 SET SYSBUCUS AUTO 1 IN
0500 SORTED SOCIALNO
0510 OWNER SYSTEM
0520 MEMBER CUSTOMER
0530 SET SYSSAVE AUTO 1 IN
0540 SORTED SAVINGNO
FIG 3.2 IDL OF THE SCHEMA
3.1 DETAILS OF THE RECORD TYPES

From Fig 3.1, there are total 8 record types, and three of them are SYSTEM record types. The three SYSTEM record types imply that the data base can be accessed directly three ways through the three SYSTEM record types. The SYSTEM record types are not needed to be defined as record types in the DDL in Fig 3.2, but they are needed to be defined in the set types owned by the SYSTEM record types.

The CUSTOMER record type, from Fig 3.1, represents the customers of the bank. There is only one data item in the CUSTOMER record type, i.e., SOCIALNO (social security number). Because each person has only one social security number, each occurrence of the CUSTOMER record type represents a customer. Line 0080 and 0090, from Fig 3.2, are employed to define the CUSTOMER record type.

The SAVEACCT (saving account) and the CHEKACCT (checking account) record types are used to keep the records of saving accounts and checking accounts respectively. The data items of the SAVEACCT record type and their meanings are listed in Fig 3.3, and the data items of the CHEKACCT record type and their meanings are listed in Fig 3.4. The keys of SAVEACCT and CHEKACCT are SAVINGNO (saving account number) and CHEKNO (checking account number) respectively. Each occurrence of SAVEACCT and CHEKACCT represents an account number. Line 0110 to 0260, from Fig 3.2, are used to define the SAVEACCT, and line 0270 to 0040 are used to define CHEKACCT. In this project, the number from 000-0000 to 500-0002
<table>
<thead>
<tr>
<th>Line Number</th>
<th>Data Item Name</th>
<th>Meaning</th>
<th>Length(byte)/Format</th>
<th>Variable Name Used In BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110</td>
<td>SAVINGNO</td>
<td>Saving Account Number</td>
<td>8/xxx-xxxx</td>
<td>S0$</td>
</tr>
<tr>
<td>0120</td>
<td>SNAME</td>
<td>Name</td>
<td>20/none</td>
<td>S1$</td>
</tr>
<tr>
<td>0130</td>
<td>SSTREET</td>
<td>Street</td>
<td>25/none</td>
<td>S2.1$</td>
</tr>
<tr>
<td>0140</td>
<td>SCITY</td>
<td>City</td>
<td>15/none</td>
<td>S2.2$</td>
</tr>
<tr>
<td>0150</td>
<td>SSTATE</td>
<td>State</td>
<td>2/none</td>
<td>S2.3$</td>
</tr>
<tr>
<td>0160</td>
<td>SZIP</td>
<td>Zip Code</td>
<td>5/none</td>
<td>S2.4$</td>
</tr>
<tr>
<td>0170</td>
<td>SAVEPHON</td>
<td>Phone Number</td>
<td>12/xxx-xxx-xxxx</td>
<td>S3$</td>
</tr>
<tr>
<td>0180</td>
<td>SASTATUS</td>
<td>Saving Account Status</td>
<td>1/none</td>
<td>S4$</td>
</tr>
<tr>
<td>0190</td>
<td>SOPENDAT</td>
<td>Open Account Date</td>
<td>8/(YY/MM/DD)</td>
<td>S5$</td>
</tr>
<tr>
<td>0200</td>
<td>SOBALANC</td>
<td>Forward Balance</td>
<td>8/none</td>
<td>S6#</td>
</tr>
<tr>
<td>0210</td>
<td>SNBALANC</td>
<td>Up-To-Date Balance</td>
<td>8/none</td>
<td>S7#</td>
</tr>
<tr>
<td>0220</td>
<td>SACHARGE</td>
<td>Saving Activity Charge</td>
<td>4/none</td>
<td>S8#</td>
</tr>
<tr>
<td>0230</td>
<td>SINTERES</td>
<td>Interest Earned In This Quarter</td>
<td>4/none</td>
<td>S9#</td>
</tr>
<tr>
<td>0240</td>
<td>STINTERE</td>
<td>Interest Earned In This Year</td>
<td>4/none</td>
<td>S10#</td>
</tr>
<tr>
<td>0250</td>
<td>SNOWITH</td>
<td>Number Of Withdrawal</td>
<td>2/none</td>
<td>S11%</td>
</tr>
<tr>
<td>0260</td>
<td>SLASTREP</td>
<td>Date Of Last Statement</td>
<td>8/(YY/MM/DD)</td>
<td>S12$</td>
</tr>
</tbody>
</table>

FIG 3.3
<table>
<thead>
<tr>
<th>Line Number</th>
<th>Data Item Name</th>
<th>Meaning</th>
<th>Length(byte)/Format</th>
<th>Variable Name Used In BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0280</td>
<td>CHECKNO</td>
<td>Checking Account Number</td>
<td>8/xxxx-xxxx</td>
<td>HO$</td>
</tr>
<tr>
<td>0290</td>
<td>CNAME</td>
<td>Name</td>
<td>20/none</td>
<td>H1$</td>
</tr>
<tr>
<td>0300</td>
<td>CSTREET</td>
<td>Street</td>
<td>25/none</td>
<td>H2.1$</td>
</tr>
<tr>
<td>0310</td>
<td>C_CITY</td>
<td>City</td>
<td>15/none</td>
<td>H2.2$</td>
</tr>
<tr>
<td>0320</td>
<td>C_STATE</td>
<td>State</td>
<td>2/none</td>
<td>H2.3$</td>
</tr>
<tr>
<td>0330</td>
<td>CZIP</td>
<td>Zip Code</td>
<td>5/none</td>
<td>H2.4$</td>
</tr>
<tr>
<td>0340</td>
<td>CHEKPHON</td>
<td>Phone Number</td>
<td>12/xxx-xxx-xxxx</td>
<td>H3$</td>
</tr>
<tr>
<td>0350</td>
<td>CHSTATUS</td>
<td>Checking Account Status</td>
<td>1/none</td>
<td>H4$</td>
</tr>
<tr>
<td>0360</td>
<td>COPENDAT</td>
<td>Open Account Date</td>
<td>8/(YY/MM/DD)</td>
<td>H5$</td>
</tr>
<tr>
<td>0370</td>
<td>COBALANC</td>
<td>Forward Balance</td>
<td>8/none</td>
<td>H6#</td>
</tr>
<tr>
<td>0380</td>
<td>CNBALANC</td>
<td>Up-To-Date Balance</td>
<td>8/none</td>
<td>H7#</td>
</tr>
<tr>
<td>0390</td>
<td>CACHARGE</td>
<td>Checking Activity Charge</td>
<td>4/none</td>
<td>H8$</td>
</tr>
<tr>
<td>0400</td>
<td>CLASTREP</td>
<td>Date Of Last Statement</td>
<td>8/(YY/MM/DD)</td>
<td>H9$</td>
</tr>
</tbody>
</table>

FIG 3.4
are assigned to saving accounts, and the numbers from 500-0014 to 999-9991 are assigned to checking accounts. Note that there are gabs between 500-0002 and 500-0014, and the reason is that not all the numbers can be assigned to be account numbers. A number must satify a procedure called MOD 11 ARITHMETIC WEIGHTS (see Sec 5.1) before it can be assigned to be an account number. The procedure is employed to ensure about the integrity and the security problem.

From Fig 3.3, only the data items considered to be non-trival are discussed. SAVINGNO is used as the key of SAVEACCT record type. SAVINGNO is a eight byte data item, but there are only seven digits. Again, not all the numbers between 000-0000 and 500-0002 can be assigned to be saving account numbers. For example, 300-0000 can not be saving account number because it fails to pass the test.

SASTATUS (saving account status), a one byte field, represents the status of saving accounts. It may be 0, 1, 7, or 9. "0" means closed accounts. "9" also means closed accounts. The difference between "0" and "9" accounts is that the "0" accounts are the accounts closed in the previous quarter while the "9" accounts are the accounts closed in this quarter and are needed to be changed to "0" at the end of the quarter. "1" accounts are active accounts. "7" accounts are the accounts called "ranged account". In SAVEACCT record type, there are six ranged accounts. The ranged accounts are 000-0000, 100-0007, 200-0003, 300-0011, 400-0006, and 500-0002. The purpose of ranged accounts is used to determine the range of the accounts wanted to be
printed in each run of the PRINT SAVING STATEMENT PROGRAM (see Sec 4-2.5). For example, the active accounts between 000-0000 and 100-0007 can be the accounts specified to be printed on the first run of PRINT SAVING STATEMENT PROGRAM, and accounts between 100-0007 and 200-0003 can be specified to be printed on the second run. If there are no ranged accounts, all the active accounts in the data base, may be thousands, are printed, and this may consume too much time which may not available. The other advantage of ranged account is to reduce the load of mailing clerks. With ranged accounts, the saving statements are distributed on several runs (usually several nights), and this can simplify the mailing jobs.

SOBALANC (saving old balance) is the balance at the beginning of each quarter and stays unchanged until the end of the quarter. At the end of quarter, the true balance at that time is copied into SOBALANC. The SOBALANC gives the information about the amount of money that is forwarded from the last quarter. The information is needed in the saving statement sent to the customers at the end of every quarter.

SNBALANC (saving new balance) is the balance that is updated every night according to the transactions of that day. SNBALANC gives the information about the most up-to-date balance.
SINTERES (saving interest) is the sum of the interests in a quarter, while STINTERE (saving total interest) is the sum of the interests in a year. SINTERES is updated every night because the rate of interest is compound daily. At the end of quarter, SINTERES is added into the STINTERE (STINTERE ← STINTERE + SINTERES). Another words, SINTERES gives the information about the interests earned in a quarter, and STINTERE gives the information about the interests earned in a year. The customers use STINTERE for taxing purpose at the end of the year.

SNOWITH (number of withdrawal) is the number of withdrawals in a quarter. There is a two-dollar charge per withdrawal after three withdrawals in a quarter. The UPDATE PROGRAM (see Sec. 4-2.3), in each run, determines if the SNOWITH is greater than three. If SNOWITH is greater than three, the UPDATE PROGRAM puts the appropriate amount of money into the SACHARGE data item. Both SNOWITH and SACHARGE are set to zero at the beginning of every quarter by the PRINT SAVING STATEMENT PROGRAM.

SLASTREP (saving last report) contains the date of the last report sent to the customer. This data item is needed in the phase "DATE OF LAST STATEMENT" in the saving statements which are sent to the customers at the end of every quarter. At the end of quarter, SLASTREP is set by PRINT SAVING STATEMENT PROGRAM to be equal to the date the program is run. From Fig. 3.3, note that the format of SLASTREP is YY/MM/DD, and the reason will be given when the SAVXTION and CHXTION record types are discussed.
From Fig 3.4, the data items of CHEKACCT are shown. There are total 15 data items in CHEKACCT record type. In this project, the checking accounts do not give interest. The numbers from 500-0014 to 999-9991 are assigned to checking account numbers. Not all the numbers can be checking account numbers because some of them do not pass the MOD 11 ARITHMETIC WEIGHTS test. Fig 3.5 shows the analogous between the SAVEACCT and the CHEKACCT record types, so the details of the CHEKACCT will not be discussed again. Note that all the data items in Fig 3.5 are referred by the variable names used in the application programs (written in BASIC language) instead of the data item names defined in the DDL.

<table>
<thead>
<tr>
<th>SAVING ACCOUNT</th>
<th>CHECKING ACCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0$</td>
<td>H0$</td>
</tr>
<tr>
<td>S1$</td>
<td>H1$</td>
</tr>
<tr>
<td>S2.1$</td>
<td>H2.1$</td>
</tr>
<tr>
<td>S2.2$</td>
<td>H2.2$</td>
</tr>
<tr>
<td>S2.3$</td>
<td>H2.3$</td>
</tr>
<tr>
<td>S2.4$</td>
<td>H2.4$</td>
</tr>
<tr>
<td>S3$</td>
<td>H3$</td>
</tr>
<tr>
<td>S4$</td>
<td>H4$</td>
</tr>
<tr>
<td>S5$</td>
<td>H5$</td>
</tr>
<tr>
<td>S6#</td>
<td>H6#</td>
</tr>
<tr>
<td>S7#</td>
<td>H7#</td>
</tr>
<tr>
<td>S8!</td>
<td>H8!</td>
</tr>
<tr>
<td>S12$</td>
<td>H9$</td>
</tr>
</tbody>
</table>

FIG 3.5
There are two more record types left to be discussed, i.e., SAVXTION (saving account transaction) and CHXTION (checking account transaction). Both of them have three data items. From Fig 5.6, the data items of SAVXTION and their meaning are shown. SXDATE is the date of transaction. SXCODE is the code of transaction (D means deposit; W means withdrawal). SXAMOUNT is the amount of money in transaction. Fig 5.7 illustrates the data items of CHXTION.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Data Item Name</th>
<th>Meaning</th>
<th>Length(byte)/Format</th>
<th>Variable Name Used In BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0420</td>
<td>SXDATE</td>
<td>Saving Transaction Date</td>
<td>8/(YY/MM/DD)</td>
<td>T1$</td>
</tr>
<tr>
<td>0430</td>
<td>SXCODE</td>
<td>Transaction Code (D=Deposit; W=Withdrawal)</td>
<td>1/none</td>
<td>T2$</td>
</tr>
<tr>
<td>0440</td>
<td>SXAMOUNT</td>
<td>Saving Transaction Amount</td>
<td>8/none</td>
<td>T3$</td>
</tr>
</tbody>
</table>

FIG 5.6

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Data Item Name</th>
<th>Meaning</th>
<th>Length(byte)/Format</th>
<th>Variable Name Used In BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0460</td>
<td>CXDATE</td>
<td>Checking Transaction Date</td>
<td>8/(YY/MM/DD)</td>
<td>H1$</td>
</tr>
<tr>
<td>0470</td>
<td>CXCODE</td>
<td>Transaction Code (D=Deposit; W=Withdrawal)</td>
<td>1/none</td>
<td>H2$</td>
</tr>
<tr>
<td>0480</td>
<td>CXAMOUNT</td>
<td>Checking Transaction Amount</td>
<td>8/none</td>
<td>H3$</td>
</tr>
</tbody>
</table>

FIG 5.7
The data items needed to be discussed are the format of saving transaction date, i.e., SXDATE and checking transaction date, i.e., CXDATE. Both of them are defined as character type data items. Because the transactions needed to be sorted in ascending order according to the time the transactions occur, the YY/MM/DD format is more appropriate than the MM/DD/YY format. The MM/DD/YY format does not give any problems as long as all the transactions occur in the same year. However, if the transactions span over than one year, the MM/DD/YY can not be used. For example, if "11/02/80" (November 2, 1980) is compared to "01/10/81" (January 10, 1981), 01/10/81 will be logically found before 11/02/80. This is because when the computer compares the first characters -- 0 is compared to 1 --, the computer finds that 0 is less than 1 and decides that 01/10/81 is less than 11/02/80, and this is not true in the real world. If the YY/MM/DD format is used, the above problem will never occur. In this project, the YY/MM/DD format is choosen to be the format of all the dates which are kept in the data base. However all the dates submited to and received from the computer still have the MM/DD/YY format. There are subroutines in the application programs that change the MM/DD/YY format to the YY/MM/DD format, and vice versa.
3.2 DETAILS OF THE SET TYPES

From Fig 3.1, there are seven set types in the schema. All of them have 1:n relationships, and this implies that each occurrence of the owner record type has one occurrence of the set type owned by the record type. SYSCUS (system-customer), SYSSAVE (system-saving account), SYSCHECK (system-checking) set types have SYSTEM as their owner record types. Each SYSTEM record type has only one occurrence, so there is only one occurrence of the SYSCUS, SYSSAVE, and SYSCHECK set types. The order of SYSCUS, SYSSAVE, and SYSCHECK are sorted by SOCIALNO, SAVINGNO, and CHEKNO respectively. Lines 0490 to 0600, from Fig 3.2, are used to define the three set types.

Each customer may have either a saving account or a checking account. Some customers may even have several saving accounts and several checking accounts. These informations are conveyed by the CUSSAVE (customer-saving account) and CUSCHECK (customer-checking account) set types. From Fig 3.2, lines 0610 to 0640 are used to define CUSSAVE, and lines 0650 to 0680 are used to define CUSCHECK. The order of the CUSSAVE set type is sorted by SAVINGNO, and the order of CUSCHECK is sorted by CHEKNO.
Fig 3.8(a) shows that the person whose social security number is 123-45-6789 has 2 saving accounts, 111-1115 and 444-4449, and also a checking account, 666-6668. From Fig 3.8(b), the person whose social security number is 234-56-7890 has one saving account, 333-3334 and has no checking account.
The saving account transactions and the checking account transactions have the relationships with the saving accounts and the checking accounts via the SAVEXAC and the CHECKXAC set types respectively. Lines 0690 to 0760, from Fig 3.2, are used to define the SAVEXAC and the CHECKXAC set types. Note that no data item is specified as the sorted keys of both the SAVEXAC and the CHECKXAC set types. However "SORTED" is specified in line 0700 and 0740, and this implies that the order of SAVEXAC and the CHECKXAC set types are sorted by every data item in the SAVXTION and the CHKXTION record types respectively. For example, saving account number 111-1115, from Fig 3.9(a), has two transactions which occurred in the same date, and the "D" (deposit) transaction is logically found before the "W" (withdrawal) transaction because "D" is less than "W". From Fig 3.9(b), the transaction with 125 dollars is logically found before the transaction with 150 dollars because 125 is less than 150.

![Diagram](image-url)
CHAPTER 4  THE APPLICATION PROGRAMS

In this chapter, all the application programs accessed the data base are discussed. But before discussing them, section 4-1 is devoted to discuss about a problem which arises from the human errors at the data entry point. The problem is "the transaction" problem. After the problem is introduced, the rests of the chapter discuss about all the application programs. There are total seven application programs shown in Fig 4.1. The abbreviated names in the parentheses following the full names are used to refer to the programs.

1. OPEN AND CLOSE ACCOUNTS PROGRAM (B1)
2. ENTER TRANSACTIONS PROGRAM (B2)
3. UPDATE PROGRAM (UPDATE)
4. CALCULATE INTEREST PROGRAM (IN)
5. PRINT SAVING STATEMENT PROGRAM (P1)
6. PRINT CHECKING STATEMENT PROGRAM (P2)
7. READ-ONLY THE DATA BASE PROGRAM (LIST)

FIG 4.1

From Fig 4.1, B1 is used to open new accounts and close old accounts. B2 is used to enter all the transactions. UPDATE is used to find the most up-to-date balance of every active account and determine if there is any activity charge in the account. IN is used to calculate the amount of interests which are paid by a compound daily rate. Note that there is no interest paid to the checking accounts. P1 and P2 are used to print the saving
and checking statements respectively. Saving statements are printed at the end of every quarter, and checking statements are printed at the end of every month. LIST is used to retrieve all the details of a specified account.

4-1 THE TRANSACTION DATE PROBLEM

Recalled that the transactions of both saving and checking accounts have three data items, i.e., date of transaction, code of transaction (D means deposit; W means withdrawal), and amount of transaction; all the data items are used as the sorted keys of SAVEXAC and CHECKXAC set types. The format of transaction dates kept in the data base is YY/MM/DD, even though the data-entry clerk use the MM/DD/YY format. There is a subroutine change the YY/MM/DD format to the MM/DD/YY format, and vice versa.

If all the transactions, may be hundreds in each day, are entered into the data base correctly, there is no problem, but that is impossible. If the transaction date is entered incorrectly, the logical position of the transaction is not in the place where it is supposed to be. Fig 4.2(a) shows that saving account number 111-1115 have three transactions which are entered correctly. If the date of the last transaction is entered incorrectly as 02/12/81 instead of 02/21/81, the logical positions of the transactions will look like the one shown in Fig 4.2(b). Because there are many transactions in each day, the error can occur easily.
There are two methods to detect the above error. Both methods have the same idea but different ways to implement. The idea is the program used to accept the transactions (B2) should go to pick up "the correct date" from somewhere and compare the correct date with the transaction date. If both dates are the same, the program will accept the transaction, otherwise the program will reject the transaction. Here are the two methods that the correct date can be obtained:

1. Using a Real Time Card

2. Using Software to Simulate the Correct Date
The first method employs hardware, i.e., some microcomputers have the circuits similar to the digital clocks. The program accepted the transactions can go to pick up the date from the circuits and uses it to compare to the incoming date entered by the data-entry clerk. The result of comparing determines if the transaction will be accepted. All the real time cards have their own batteries, so even though the computers are turned off, the clocks on the cards still work.

The second method is implemented by using software. The correct date is entered only once and is kept in a file on a disk. The program used to accept the transactions reads the correct date from the file and uses it to compare to the incoming transaction date. If the dates are equal, the transaction will be accepted, otherwise the transaction will be rejected. There is one more problem with this method, i.e., the date needed to be changed. For example, the forth of January is needed to be changed to the fifth of January. So the last program that run on each night has the responsibility to change the date. In this project, IN is the last program that run on each night.

Comparing between both methods, the first method is easier than the second one, however, the microcomputer used in this project does not have the real time card, so the second method is employed.
As a matter of fact, not only B2 reads the correct date from the disk but most of the programs also read the date. For example, B2 reads the date to detect the error while UPDATE uses the date to find the transactions that occur in that business day.

In this project, the file that contains the correct date is named "DATE".
4-2 DETAILS OF THE APPLICATION PROGRAMS

In this section, all the application programs are discussed. The general functions of each of them are presented along with the HIPO chart.

4-2.1 OPEN AND CLOSE ACCOUNTS PROGRAM (B1)

B1 is used to open and close both the saving and checking accounts. Before an account can be opened or closed, all the data items entered from the terminal are edited, and after the account is opened or closed, the informations about the account are printed at the log file. The HOPO is shown in Fig 4.3.

FIG 4.3

Note: "O" means the decision of the upper level module determines which of the lower modules will be called.
4-2.2 ENTER TRANSACTIONS PROGRAM (B2)

B2 is used to enter every type of transaction into the database. The types of transactions may be saving deposit, saving withdrawal, checking deposit, or checking withdrawal. Similar to B1, all the transaction entered into the database must be edited before they are accepted, and after the transactions are accepted, the informations of the transactions are printed at the log file. The HIPO of B2 is shown in Fig 4.4

![Diagram](image)

**FIG 4.4**

Note: "△" means the decision of the upper level module determines which of the lower level modules will be called.
4-2.3 UPDATE PROGRAM (UPDATE)

UPDATE is run every night except holidays because there is no transaction on holidays. UPDATE is used to calculate the balance of all the active accounts. Active accounts mean the accounts which have saving account status (SASTATUS) or checking account status (CHSTATUS) equal to "1". In addition to calculating the balances, UPDATE is used to determine if there are any activity charges in the accounts. The rules used in charging the money from the accounts are shown below.

Saving Accounts: There is a two-dollar charge per withdrawal after three withdrawals in a quarter.

Checking Accounts: balance ≥ $500......................no charge
$400 ≤ balance < $500............... $3
$300 ≤ balance < $400............... $4
balance < $300................... $5

The HIPO of UPDATE is shown in Fig 4.5.

```
UPDATE PROGRAM

OPEN DATA BASE

UPDATE SAVING ACCOUNT

UPDATE CHECKING ACCOUNT

CLOSE DATA BASE

DETERMINE SAVING ACTIVITY CHARGE

DETERMINE CHECKING ACTIVITY CHARGE

FIG 4.5
```
CALCULATE INTEREST PROGRAM (IN)

IN is used to calculate the interests of saving accounts. The interest rate is 7% and compounded daily. There is no interest paid to the checking accounts. IN is the last program run on each night and has the responsibility to change the "correct date". The subroutine used to change the correct date is named "UPDATE THE CORRECT DATE". (Recalled from Sec 4-1 that the "correct date" is used to solve the transaction date problem.) The HIPO chart of IN is shown in Fig 4.6.

```
  CALCULATE
     INTEREST
    /
   /
  OPEN DATA BASE UPDATE THE CORRECT DATE CALCULATE INTEREST CLOSE DATA BASE
```

FIG 4.6

In this project, it is assumed that there is no transaction on Saturday and Sunday, and the interest is compounded daily. To improve the system efficiency, IN has a subroutine to determine if a date is Friday. If it is Friday, the interest will be compounded three days -Friday, Saturday, and Sunday- without requiring to run IN on Saturday and Sunday. With the subroutine,
IN is run only once instead of three times on the weekend. The subroutine used to determine day of the week is "DETERMINE DAY OF THE WEEK". The subroutine is discussed in Sec 5-2.
4-2.5 PRINT SAVING STATEMENT PROGRAM (P1)

P1 is used to print the saving statements sent to the customers at the end of every quarter and initialize some data items to be ready for the next quarter. The data items set to zero by P1 are the number of withdrawals (SNOWITH), saving activity charges (SACHARGE), interests earned in this quarter (SINTERES). Besides, the date of the last statement (SLASTREP) is set to be equal to the date that P1 is run. The SLASTREP data item is needed in the phase "DATE OF LAST STATEMENT" in the saving statements sent to the customers. The HIPO chart of P1 is shown in Fig 4.7.

FIG 4.7
4.2.6 PRINT CHECKING STATEMENT PROGRAM (P2)

P2 is similar to P1. P2 is used to print the checking statements sent to the customers. The statements are sent at the end of every month. After printing a statement of an account, P2, same as P1, initializes some data items of the account. The data item set to zero is the checking activity charge (CACHARGE). The date of last report is set to be equal to the date P2 is run, and checking old balance is set to be equal to checking new balance (COBALANC ← CNBALANC). Fig 4.8 shows the HIPO chart of P2.

```
PRINT CHECKING STATEMENT

OPEN DATABASE

PRINT CHECKING STATEMENT

CLOSE DATABASE

PRINT A STATEMENT

INITIALIZE (SOME) DATA ITEMS
```

FIG 4.8
4-2.7 READ-ONLY THE DATA BASE PROGRAM (LIST)

Given an account number, LIST can be used to retrieve all the informations of the account. For example, LIST can used to answer the customers about their saving or checking account balances. Fig 4.9 illustrates the HIPO chart of LIST.

![HIPO Chart]

Note: "○" means the decision of the upper level module determines which of the lower modules will be called.
CHAPTER 5  THE SUBROUTINES

There are twelve subroutines in this project, however, only three of them are discussed in this chapter. The first one named SELF CHECK DIGIT is discussed in Sec 5-1. The SELF CHECK DIGIT subroutine uses the method called MOD 11 ARITHMETIC WEIGHTS to detect the miscoding of account numbers entered from the terminal. In Sec 5-1, a number of "check digits" are presented, and the comparison between them are shown.

The second subroutine named CALCULATE NUMBER OF DAYS is used to, given two dates, calculate the number of days between the two dates. The third subroutine is named DETERMINE DAY OF THE WEEK. Given a date, the third subroutine can be used to determine the day of the week. The day of the week is represented by digits 0 through 6 where 0 is Sunday. The second and third subroutine are discussed together in Sec 5.2 because they relate to each other.

5-1 CHECK DIGITS

Although there are other ways of controlling the accuracy of manual encoding, the use of check digits is an accepted practice. This is reasonable insurance against miscoding, and it is to be encouraged. No check digit system yet developed will catch every conceivable error. The number of misposts that may "get by" an electronic system in which a check digit system is operative normally will be far less than the number that occur in posting from nonencoded media. But perfect posting through
a theoretically perfect check digit system is not an isolated goal. The system adopted should be evaluated in the light of its error detection capabilities, efficiency, compatibility with available hardware and, very likely, compatibility with neighboring banks that may eventually consider joint processing.

Any number may be manipulated by arithmetic means to produce a single digit known as a check digit. This new digit becomes part of the number, ordinarily occupying the unit position (520636). It is used primarily to verify that the number has been transcribed correctly. Verification is done by repeating the arithmetic used to create the check digit. For example, a check digit for an account number such as 52063 can be evolved by any one of the processes that follow. To use the check digit for verification of transcription, the same arithmetic process is repeated, and the result is compared with the original check digit. If the original check digit and the computed one do not correspond, an error is indicated. The following definitions will clarify the process descriptions:

Modulus or Mod - The number by which the result of a calculation is divided to arrive at a remainder. For example, 222/11 = 20 with a remainder of 2; 11 is the module, or divisor.

Basic account number - The number that includes all digits except the check digit.

Weights - The numbers that are multiplied times the digits of the basic account number in a check digit process.

Transcription error - An error involving a substitution for one digit in its own position. For example: a 3 for a 1, 2, 4,
5, 6, 7, 8, 9 or 0; a 6 for a 1, 2, 3, 4, 5, 7, 8, 9 or 0.

Transposition error — An error caused by reversing two digits in adjacent positions. For example: 25431 for 24531.

Double transposition error — An error caused by reversing digits in alternate positions. For example: 254351 for 234551.

Random error — a combination of any of these errors mentioned above or one not included above. For example: 254351 for 245531 or 362295 for 326265.

Sample Calculations of MOD 10
and MOD 11 Check Digit Systems

Sample A

MOD 10 — Double Add Double

Take a basic account number 5 2 0 6 3

Multiply the units position and

every alternate position 5 0 3

by two 2 2 2

Products 10 0 6

Bring down those digits

not multiplied by two 2 6

Cross-add the digits and these

nonmultiplied digits 1+0+2+0+6+6 = 15

Subtract this total from the next

highest number ending in zero 20-15

Check Digit 3

Self-Checking Account number 52-063-5
Sample B

MOD 10 - 1-3-7

Take a basic account number 5 2 0 6 3
multiply by the fixed weight 1,3,7,1,3,7,etc. 1 3 7 1 3

Products 5 6 0 6 9

Add these products 5+6+0+6+9

Result 26

Subtract this total from the next highest number ending in zero 30

-26

Check Digit 4

Self-Checking Account Number 52-063-4

Sample C

MOD 11 - Arithmetic Weights

Take a basic account number 5 2 0 6 3

Multiply each digit by the proper weight (i.e., starting with the units position the multiplier is 2, the tens position is 3, and so on) 6 5 4 3 2

Products 30 10 0 18 6

Add these products 30+10+0+18+6 = 64

Divide this sum by eleven 64/11 = 5

and a remainder of 9
Subtract this remainder from eleven

\[ 11 - 9 = 2 \]

Check Digit 2

Self-Checking Account Number 52-063-2

Sample D

MOD 11 = Geometric Weights

Take a basic account number 5 2 0 6 3

Multiply each position by the proper weight, (i.e., starting with the unit position of the account number the weight of each position is increased by the power of 2 or 2, 4, 8, 16, etc.) 32 16 8 4 2

\[
\begin{array}{c}
\text{Products} \\
160 32 0 24 6
\end{array}
\]

Cross-add the products 160 + 32 + 0 + 24 + 6 = 222

Divide this sum by eleven 222 / 11 = 20 and a remainder of 2

Subtract the remainder from eleven 11

\[ 11 - 2 = 9 \]

The difference is the Check Digit 9

Self-Checking Account Number 52-063-9
## Error Detection in Check Digit Systems

<table>
<thead>
<tr>
<th>Type of Check Digit Numbering System</th>
<th>Type of Error the System Detects</th>
<th>Type of Error That May Not Be Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulus 10—Double Add Double</td>
<td>100% of the transcription errors</td>
<td>Transpositions in adjacent positions between 09 and 90</td>
</tr>
<tr>
<td></td>
<td>97% of transposition errors</td>
<td>Alternate column transpositions of interchanged digits</td>
</tr>
<tr>
<td></td>
<td>90% of all types of random errors</td>
<td>Certain random errors caused from the depression of the wrong horizontal row on a 10-key adding keyboard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random errors caused by substituting a valid check digit for an otherwise erroneously recorded basic account number</td>
</tr>
<tr>
<td>Modulus 10—1-3-7</td>
<td>100% of transcription errors</td>
<td>Certain types of random errors</td>
</tr>
<tr>
<td></td>
<td>88.9% of transposition errors</td>
<td>A percentage of transposition errors</td>
</tr>
<tr>
<td></td>
<td>88.9% of double transposition errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>90% of all random type errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A high percentage of alternate column transpositions</td>
<td></td>
</tr>
<tr>
<td>Modulus 11—Arithmetic Weights</td>
<td>All transcription errors</td>
<td>Some types of random errors caused by erroneous recording</td>
</tr>
<tr>
<td></td>
<td>All single or double transposition errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>90% of random errors of various sorts</td>
<td></td>
</tr>
<tr>
<td>Modulus 11—Geometric Weights</td>
<td>100% of transcription errors</td>
<td>About 10% of random errors</td>
</tr>
<tr>
<td></td>
<td>100% of single and double transposition type errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>90% of the random errors</td>
<td></td>
</tr>
</tbody>
</table>
5-2 THE CALENDAR FUNCTIONS

For the period March 1, 1900 through February 28,2100, these equations interchangeably solve for dates and days. Given two dates, the number of days between them can be calculated. Given one date and a specified number of days, a second date can be found. The equations can also determine the day of the week given the date. The day of the week is represented by the digits 0 through 6 where 0 is Sunday. Given a date, its Julian Day number can be found. (The Julian number is an astronomical convention representing the number of days since January 1, 4713 B.C.)

1. To compute the day number from the date:

Julian Day number = INT(365.25 $y'$) + INT(30.6001 $m'$) + $d + 1720982$

where $y'$ = \[ \begin{cases} \text{year} - 1 & \text{if } m = 1 \text{ or } 2 \\ \text{year} & \text{if } m > 2 \end{cases} \]

$m'$ = \[ \begin{cases} \text{month} + 13 & \text{if } m = 1 \text{ or } 2 \\ \text{month} - 1 & \text{if } m > 2 \end{cases} \]

Then days between dates is found by

Days = Day number$_2$ - Day number$_1$

2. To compute the date from a day number:

Day # = Julian Day Number - 1720982

$y'$ = INT \[ \left[ \frac{\text{Day #} - 122.1}{365.25} \right] \]

$m'$ = INT \[ \left[ \frac{\text{Day #} - \text{INT}(365.25 y')}{30.6001} \right] \]
Day of the month = Day # - \( \text{INT} \left[ 365.25 y' \right] - \text{INT} \left[ 30.6001 m' \right] \)

Month = \( m = \begin{cases} m' - 13 & \text{if } m' = 14 \text{ or } 15 \\ m' - 1 & \text{if } m' < 14 \end{cases} \)

Year = \( \begin{cases} y' & \text{if } m > 2 \\ y' + 1 & \text{if } m = 1 \text{ or } 2 \end{cases} \)

3. To compute the day of the week:

Day of the week = \( 7 \times \text{FRAC} \left( \frac{\text{Day #} + 5}{7} \right) \)
<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cromemco Z1 Microcomputer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Micro Data Base System</td>
</tr>
</tbody>
</table>

**Texts**

1. C.J. Date "An Introduction To Data Base System"
   Addison-Wesley Publishing Company
2. Alfonso F. Cardense "Data Base Management System"
   Allyn and Bacon, Inc.
3. Charles D. Coen "Bank Administration Manual",
   Volume 1, Bank Administration Institute
   Park Ridge, Illinois
4. Edward W. Reed, Richard V. Cotter, Edward K. Gill,
   Richard K. Smith "Commercial Banking",
   Prentice Hall, Inc
5. HP-67 Calculator Manual, Hewlett-Packard Company