Accounting Transactions II: Cash Transactions

By Lincoln Stoller, Ph.D.

Accounting transactions are the “beans” of the proverbial bean counters, and they are at the heart of any business. The financial numbers they contribute are crucial to management decisions and play an important role in the ultimate success or failure of a business.

In this four-article series we use 4th Dimensions to explore different types of transactions. In the first article, “Accounting Transactions, I,” Dimensions v.3, n.6, September/October 1994, we considered the basic logical and relational structure of transactions. In this article we focus on cash transactions and techniques for managing them. In the final articles in the series we’ll consider point-of-sale transactions and invoice transactions, employing different techniques in each case.

Cash Transactions Defined:

Cash transactions occur when cash is deposited or withdrawn from a cash account. A cash account is any account to which you deposit cash and which must maintain a positive cash balance. This includes checking, CD, money market, savings, or mutual fund accounts. Cash accounts are debit accounts, which means that when there is money in the account, the account has a debit balance. (Credit and debit accounts were discussed in the first article of this series.) A transaction that leads to a credit balance would cause the account to be overdrawn.

A cash transaction records a debit to the cash account when money is deposited, and a credit when it is withdrawn. Since accounting requires all transactions to be balanced, a cash transaction must have one or more additional components indicating where the money comes from, or goes to.

For example, if you bought a $2,000 scanner, $450 of software, and paid $101 in tax, you would enter the following transaction:

Account	debit	credit
First National Bank of Kalamazoo		2,501.
Computer Hardware	2,000.	

Computer Software	450.	
Sales Taxes Paid	101.	
Total:	2,501.	2,501.

The First National account is the cash account, the $2501 credit records a withdrawal. The two computer accounts are debit accounts, which maintain debit balances. As asset accounts they keep track of "stuff of value."

Hardware and software are tracked by different accounts in order to know how much of each you have acquired. They are also treated differently for income tax purposes. The Sales Taxes Paid account is an expense account that maintains a debit balance. A debit balance indicates a loss incurred through operations, which in this case is the payment of a tax.

Separating the Analysis of Function from Implementation Requirements:

An important point in database design is to analyze functional specifications independently of any database file or field structure. Internal considerations have nothing to do with understanding external requirements. Once you understand the way the data should flow and how the system should be structured, then you can turn to implementing the design. To put it another way, your understanding of the user's specifications should not depend on requirements of your particular programming environment.

Although the separation of external requirements from internal considerations is a golden rule that eventually gets broken, it is best to adhere to it as long as possible. The design method used here, based on the Data Flow Diagram (DFD), the System Structure Chart (SSC), and the Data Dictionary, are powerful techniques that support this separation. These are described in greater detail in my article, "Working Toward Step One; Working Toward a 4D File Structure" in Dimensions v.1, n.5, May/June 1992.

Flow Diagram:

The data flow diagram shows the tasks that need to be performed by the system and illustrates the how they interrelate. It presents a map of information flow in the problem domain. This means that it uses terms drawn from the user's description of the problem and the user's description of processes and reference information. For more

on the advantages of designing your application in terms of the problem domain, see “Writing Maintainable Code, III: Stable Code” in Dimensions, v.2, n.6, September/October 1993.

The DFD does not show how the application is structured or how users access these functions. It is specifically a dynamic information map without reference to the interface or program structure.

The DFD in Figure 1 illustrates what needs to be done when a user enters, modifies, or deletes a transaction. The figure applies to cash transactions and excludes other aspects of cash account management, listed in the next section.

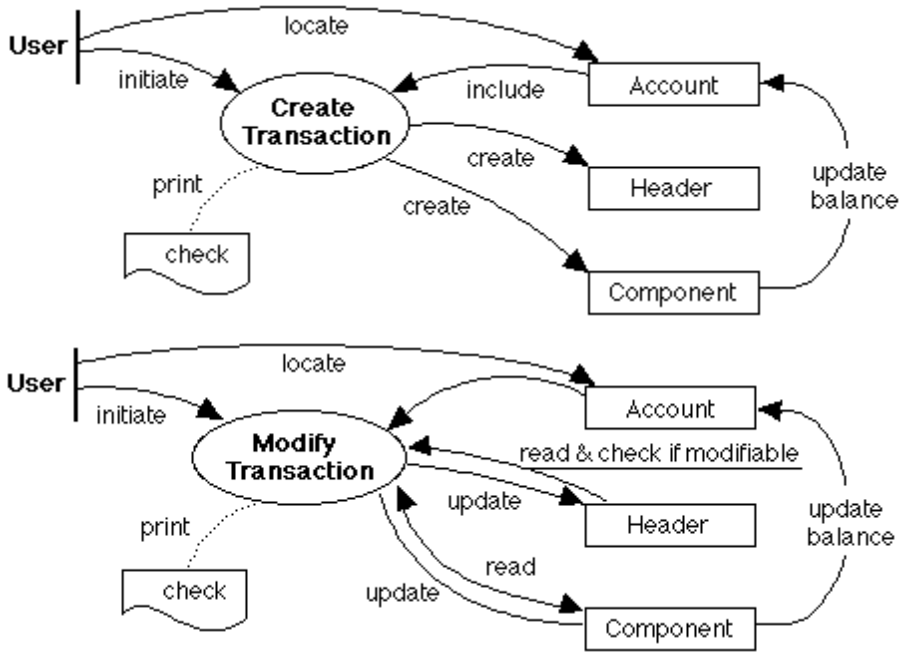

Figure 1: Cash Transaction Entry and Modification Data Flow Diagrams.

Each process is represented by a name in an oval. The user initiates each process. The lines coming from rectangles to the oval represent the user drawing information from the data stores. Lines from the oval to the rectangles represent the process of storing information. An oval connected to another oval with an arrow indicates one process that logically precedes and feeds into another process. The dotted line from the oval to the document marked “check” indicates an action that may or may not occur.

In the “modify transaction” diagram of Figure 1 the line labeled “check if modifiable” leads from “header” rectangle to the “modify

transaction" oval and represents the process of determining whether the transaction can be modified as determined by whether or not a check has been printed. The model is oversimplified since the system does not include a general ledger. If we did include a general ledger we would have to verify that the transaction had not been posted before we could allow it to be modified.

Structure Chart

The System Structure Chart (SSC) breaks the program down into areas and functions. The SSC provides a blueprint for the program's user interface. We can create an interface by associating the central areas in the chart to layouts, while letting the boxes that radiate from these areas correspond to functions accessed through the layouts.

The structure chart also describes the application in the user's terms. It displays the pathways the user follows to control the application. It does not refer to the storage or flow of information, as this is the role of the flow diagram.

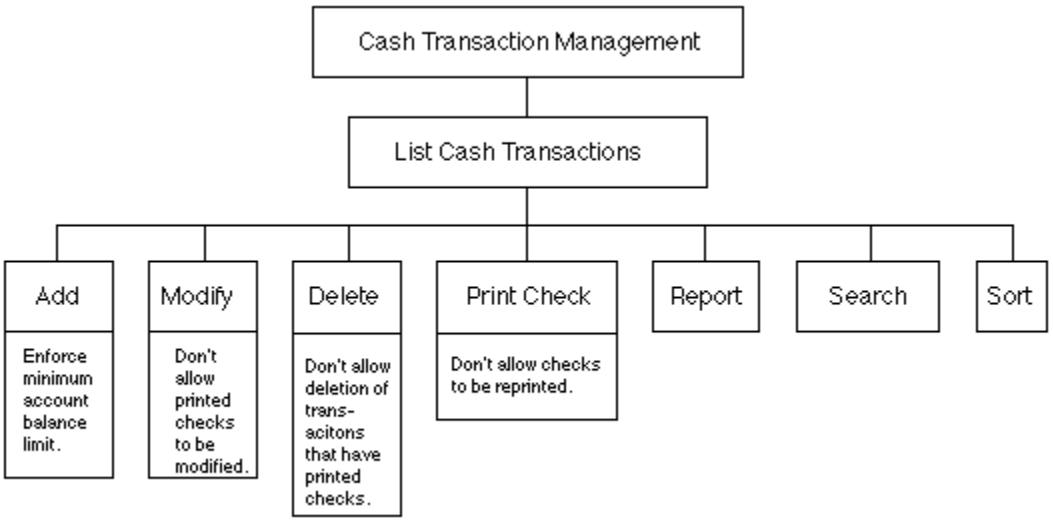

Figure 2. Cash Transaction Management Structure Chart

Each function is listed along with any conditions that apply to it. Only the business or accounting conditions are described. Conditions having to do with relational integrity or program design are considered in the Internal Data Dictionary of Table 2.

Information Requirements

Cash transactions are subject to requirements of general accounting and requirements specific to each business. For example, printed checks must have a sequentially assigned check number. Check numbering is a property of cash accounts in general. Cash accounts also have a minimum balance, and withdrawals should be prevented if they would cause the balance to drop below this limit. Minimum balance requirements are particular to each business.

In addition there are internal specifications required to support the relational structure or performance requirements. These include primary or foreign keys, derived and denormalized information, flag values and compound keys.

The initial application specifications only include the external requirements. This is enough to create the flow diagram, structure chart, and what I call the "external" data dictionary. In the design phase, where one considers performance, coding, and interface requirements, the data dictionary is extended to include the internal requirements.

External Data Dictionary:

A detailed list of the information handled by the database is expressed in the Generalized Data Dictionary, or GDD. The Generalized Data Dictionary, an extension of the usual data dictionary, includes both the items stored in the datafile and items derived from them for the purposes of display or reporting. The GDD also lists the rules that affect the data.

The GDD is composed of two parts, the external data dictionary and internal data dictionary. The former lists the items that appear in the user's specifications; the latter lists the internal format used to store or support these items. The external dictionary for cash transactions is shown in Table 1.

Table 1: External Data Dictionary

Item	Field Type	Enter-able	Description
Cash Account Number	real	yes	lookedup from account record
Cash Account Balance	real	no	lookedup from account record
Date of entry	date	yes	

Deposit or withdrawal	boolean	yes	
Amount deposited or withdrawn	real	yes	subject to minimum account balance requirements
Name of payee or payor	alpha	yes	If withdrawal, then this name appears on the check.
Check number, if applicable	longint	yes	If assigned by user, then check cannot printed. If check is marked as "to be printed," then check number is assigned when check is printed.
Description	text	yes	
To be printed	boolean	yes	Can be set only for withdrawals, and then only when the system assigns check number.
Has been printed	boolean	no	Set by system when check is printed.

Balancing account information:

This information is repeated as many times as there are balancing components.

Account name	alpha	yes	lookedup from account record.
Account number	alpha	yes	lookedup from account record.
Component amount	real	yes	
Debit or credit	boolean	yes	

The File Structure of Transaction Data:

The data model used in this series of articles stores transactions in a single one-to-many file structure. In this model each transaction has a single header record and multiple transaction component records. All transactions can be handled in this manner no matter how they are entered or what accounts they affect.

In the case of special types of transactions that need to store additional information, as in the case of invoices, another record will

be created in a separate file. This auxilliary record will be related to the transaction's header record in a one-to-one relationship.

Storing different types of transactions in a single file structure does not mean that all types must be entered through the same input layout. In fact, it is important that we enter and display different types of transactions in different layouts. In order to distinguish transactions we add a "transaction type" field to the transaction information. Cash transactions are assigned the code "CASH" to indicate that the cash transaction screen should be used to display and modify them.

The cash entry screen supports those items listed in the external data dictionary and it needs to enforce the cash entry requirements. We'll use the included subrecord technique for data entry and modification, as described in my article "Temporary Subrecord Method for Entering Related-Many Records," in Dimensions v.3, n.1, November/December 1993.

Using subrecords enables us to add and delete components without affecting the information stored on disk and without starting a transaction. (That is, without instructing 4D to cache information using the Start Transaction command). This is because subrecords are created and stored in memory — they are only saved to disk when their parent record is saved.

In the temporary subrecord method the subrecords are never saved to disk. They are created only to display existing related records. Before the parent record is saved, the subrecord information is copied to the related records file and the subrecords are deleted.

Using this method the transaction components, which have a related-many relationship to the transaction header record, are displayed as subrecords. These subrecords appear in an included layout area, displayed on the layout of the transaction header.

As mentioned in the above reference, the subrecords mirror the structure of the components file. However, they only need to contain information that is necessary for the entry and display of the components. Compound key information does not need to be stored with the subrecord, for example, because it can be created when the subrecord is copied to the component file.

On the other hand, the temporary subrecord must store information that will enable the system to locate preexisting components if they need to be updated. A subrecord cannot be deleted entirely in the case where it represents an existing related component record.

Enough information must be retained to enable the system to locate and delete the preexisting component. Two special subrecord fields are used to support these requirements, they are the “Save to Disk” field and the “Delete from Disk” field. These are shown in the Internal Data Dictionary.

File Structure and the Internal Data Dictionary

The file structure consists of a transaction header file (which stores information common to all components), the transaction components file, the data entry subfile (attached to the header file), and the accounts file. This is shown in Figure 3.

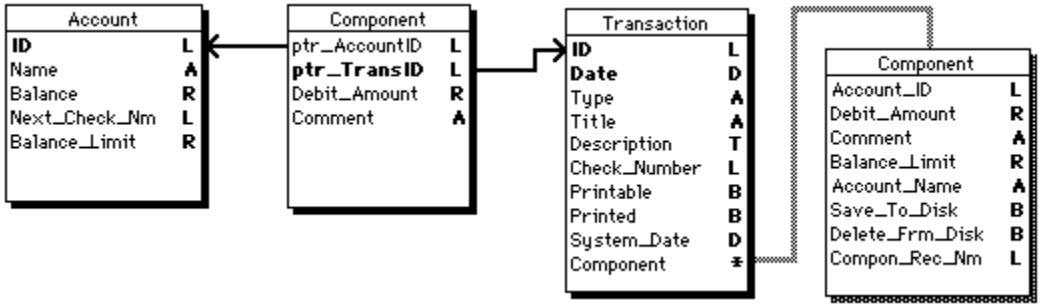

Figure 3: 4D file structure used to support cash transactions.

With the file structure of Figure 3 we can fill in the internal data dictionary. This lists the information required to support the storage, retrieval, and presentation of the information listed in Table 1. The internal data dictionary is given in Table 2. Combining the internal and the external data dictionaries describes the complete file and field structure of the database.

Table 2: Internal Data Dictionary

Item	Field Type	Enter-able	Description
Transaction ID	longint	no	Unique and required.
Transaction type	alpha	no	Set to “Cash” for entries created through this layout.
System date	date	no	Set according to the system clock.
Balance limit	real	no	Read from account.

User name	alpha	no	
Balancing account information:			
Account ID	longint	no	Assigned according to user selection.
Component record number	longint	no	Read from existing components when they exist. Set to -1 if they don't exist.
Balance limit	real	no	Read from account.
Save to disk	boolean	no	Set as TRUE when new component is created, or an existing component is modifed but not deleted.
Delete from disk	boolean	no	Set as TRUE when an existing component is deleted. When a new component is deleted its subrecord is simply deleted.

Interface:

General Transaction List

Since all transactions share a common file structure it is possible to display them using a generic layout. A general transaction list gives the user the most freedom in examining account and transaction history, but it only provides general functions. The general input layout is set as the default when viewing transactions through the general output layout.

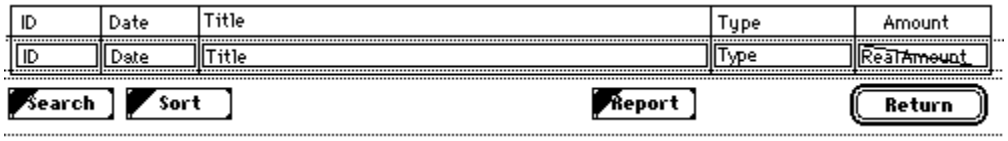

Figure 4: General transaction output layout displays transaction header records.

While this is a good way to provide common access to all transaction information, it is a poor way to handle varied, specific, and often complex information. So, even though all transactions share a basic form, we also need a way to associate specific types of transactions — cash transactions in particular — with specific input layouts. Let us examine this two-part technique.

Cash Transaction List

Cash transactions can be isolated from other transactions by displaying a list of the cash account components. Since each cash transaction will only have one cash component, we can display a list of cash components without having to worry about several components appearing for a single transaction. Even though this list displays records in the component file, rather than in the transaction header file, we can draw information from the parent header record to display the date and title of the transaction. This component output layout is shown in Figure 5.

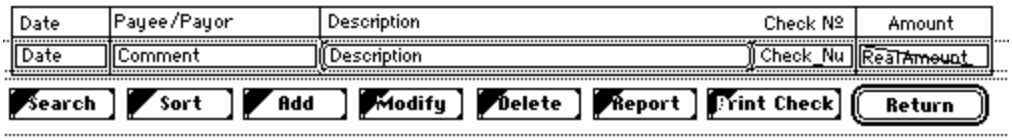

Figure 5: Output layout for cash components.

This is our specialized output layout that handles the specific cash transaction functions listed in the structure chart in Figure 2. These functions are activated by pressing buttons in the footer area.

While we do not wish to get involved with the details of 4th Dimension code, we should point out that whenever the user performs a search from this layout, the developer must intercept the resulting selection and screen out those components that are not related to the cash account. For example, if the user searches for all components entered in December 1994, the code triggered by the search button must also perform a Search in Selection for components related to the cash account. This will ensure that only cash components are displayed.

Cash Transaction Input Layout

Cash entry is done through a transaction input layout stored in the header file. This layout has an included layout displayed within it that lists noncash transaction components as subrecords. The subrecords are created when the user adds components. The subrecord information is copied to the components file, and the subrecords deleted, before the header record is saved. The entry layout is shown in Figure 6.

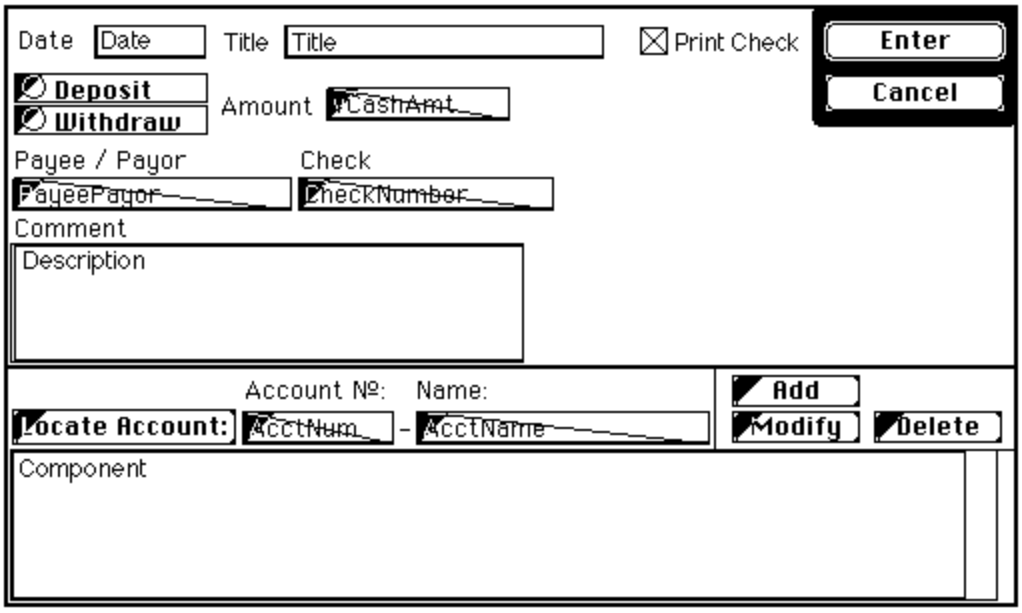

Figure 6: Cash entry layout.

Functional specifications for the input layout are drawn from the system structure chart of Figure 2 and the External Data Dictionary of Table 2. These specify entry fields, screening criteria, record modification criteria, and integrity conditions.

In particular, the specifications require,

Entry Screening:

• Impose conditions on the check number and the “print check” check box.

• Enforce minimum account balance limit.

• Prevent the cash account from being listed as one of the components.

Integrity Conditions:

• Assign ID numbers and increment counters.

• Ensure that the transaction is balanced, reject entry if it is not.

• Update account balances. Reject the entry if any of the affected accounts are locked.

• Update component records. Reject the entry if any of the affected components is locked.

• Assign “CASH” to the transaction type field.

We have not considered the details of how to program the multi-record access required to support the transaction integrity conditions. For example, transaction entry requires that we obtain read/write access to all accounts and components that are modified when the transaction is entered. There is a whole arsenal of ways to handle multi-record entries, none of which is specific to accounting. We considered a number of methods in our series entitled "Managing Multiple Record Entries," Dimensions v.2, n.2 and n.3, January/February and March/April 1993.

Overview

We have outlined how to handle cash transactions using a single file structure. This single file structure can hold both cash transactions and transactions of other types. This approach requires displaying transactions in particular layouts depending on their type. This is in contrast to some data bases where the records of each file are all of the same type and are all displayed through the same layouts.

The current series of articles has two objectives. The first is to explore different ways of handling transactions. Transactions are an important topic because they contribute critical information needed in every business. A good understanding of transactions is needed to ensure that data integrity conditions are always enforced.

Our second objective is to consider how different types of data can be handled within a single file structure. This involves using different input and output layouts for different types of records stored in a common file.

In the final two articles we will consider point-of-sale transactions and invoice transactions. In the process, we will explore other techniques for handling transactions including batch processing and extensions to the transaction file structure.