Accounting Transactions I: Debits & Credits

Lincoln Stoller, Ph.D.

In this series of articles I provide an overview of how data structure, entry, and processing are affected by accounting transactions. After explaining the theory behind transactions I'll consider three simplified examples:

- point-of-sale entries,
- cash withdrawals,
- invoices.

Each of these examples will be treated in detail in subsequent articles. There are many other possible types of transactions, but these are sufficient to develop the following general techniques:

- batch processing to speed record entry,
- using different input layouts for different records in the same file,
- using a related file to store information that pertains to specific records.

In this first article I want to dispel the common misconception that there is no simple rule for understanding debits and credits. This belief recently led a friend to say in frustration, "Accounting makes no sense! Sometimes credits are credits, and sometimes credits are debits!" Not the clearest complaint, but it gets the point across. This is not the kind of confusion you want to carry with you as you program accounting systems.

If hope I've found a way to clarify the theory behind debits and credits. I hope this explanation enables you to see the simplicity of accounting — something that most people involved with accounting don't appreciate.

Where Accounting Fits In To DB Design

When you add accounting functionality to a business database you add new rules that affect all aspects of database design. I've illustrated this in Figure1, where each row represents one of the basic aspects of database design, and each column represents an independent set of rules. Each cell represents a particular set of rules applied to a particular task. The check marks in the cells of this figure represent those cases where rules affect the corresponding aspects of database design. I've omitted a check at the bottom of the first column since internal DB rules don't say anything about what reports are needed.

RULES DATA	Internal DB Design Rules	Business Rules	Accounting Rules
Data Structure			
Data Entry			
Data Processing			
Reports, Queries	-		

Figure 1: A matrix representing the rules that effect the data.

The Meaning of Double Entry Transactions

Transactions are one of an accounting system's two most basic elements, the other being accounts, which I discussed in my article "The Structure of Accounting Systems" in Dimensions volume 3, number 5. If you think of the account structure as the skeleton of the system, providing the framework in which everything takes place, then transactions act as the blood carrying information to all parts of the system.

There is nothing special about a double entry transaction — it's a simple concept. Let me illustrate. If you have two pockets of change and you move two dollars from the left pocket to the right pocket, then the balance of funds (i.e. the total amount) on your left goes down while the balance on your right goes up.

You can record this event as a double entry transaction if you represent these pockets using a Left and a Right Pocket Cash Account. Say that debiting means you're adding cash, while crediting means you're taking cash away (that's what debiting and crediting do mean in this case, but more on that later). With this language you can write the following double entry transaction:

"Debit $2 from the Left Pocket, Credit $2 to the Right Pocket."

But things are not as simple as they seem. In accounting systems there is another dimension to transactions that makes them subtle and, unfortunately, mysterious.
Accounting has a principle that I'll call "completeness," which means:
1- all transfers of stuff of value are represented by transactions,
2- all transactions involve your accounts and ***only*** your accounts.

The first part is easy enough — it's the second part that confuses people. When we think about recording the transfer of valuables we usually consider our gain or loss in terms of the gain or loss of others. But accounting tells us to ignore the gain or loss of others — it tells us to record gains and losses only as they affect us. In a double entry system all accounts refer to things you own or things you owe.

For example, if you finally pay off that $50 fine levied against you for running naked through the Lansing State Capitol, you might think you'd record a transaction lowering your cash balance and raising the cash balance of the State of Michigan. But that's wrong. Instead you record a decrease in your cash balance against a decrease in an account that records your debts. The debt account, otherwise know as a liability account, could either be specific to fines due to the State of Michigan, or it could be a general Miscellaneous Debts account.

As another example, say you sell your friend Nina a puppy for ten dollars and she pays you in cash. In this case you *do not* balance your increase in cash by a decrease in her cash account. Instead, you balance your increase by decreasing some account *of your own*. The "sales" account is invented for this purpose and it allows you to balance your receipt of funds by decreasing the sales account balance, as shown in Figure 2.

Sales Transaction: Puppy to Nina.

	Credit	Debit
Sales Account		$10
Cash Account	$10	

Figure 2: A cash sale transaction.

In the previous example we saw how a sale increases the cash balance while decreasing the sales balance. You might think that the balances should increase in both accounts. After all, what sensible system would report increasing sales with a decreasing total?

But think of this: if you credited both accounts, that would violate the rule that there must be an origin and a destination for all assets. This is the meaning of the "debits equal credits" rule. Consequently, if you're working with two accounts and you want to credit one, then you *must* debit the other.

This paradox is resolved by dividing accounts into credit accounts and debit accounts. In credit accounts an increasing credit balance corresponds to a growth in assets. In debit accounts an increasing debit balance corresponds to a growth in assets. This means that if you debit an account of one type and credit an account of the other type, the net effect is to move the balance in the same direction, up or down, in both accounts.

Furthermore, credit accounts usually maintain a credit balance while debit accounts usually maintain a debit balance. (Expense-type accounts are credit accounts that maintain a debit balance, but then their balance represents a loss of resources.)

As we have seen, the sales and the cash accounts are of opposite type: the sales account is a credit account, and the cash account is a debit account. When we credit the sales account and debit the cash account we are raising the balance in both accounts. We obey the rule that all assets have an origin and a destination (debits = credits) and we make sense by saying that when we sell something our cash balance grows.

For some undoubtedly good reason the established convention holds that cash accounts, as well as all other accounts that represent actual assets, are of debit type. This is the convention, so you must accept it. It means than when you deposit cash, buy land, or acquire a machine, you debit the corresponding account. Debit-type accounts are fairly easy to identify since they usually represent tangible or valuable resources.

Credit accounts are all those accounts that correspond to business ownership and to repayment obligations. It's easy enough to identify repayment obligations which are debts you owe, but ownership is harder to define. Ownership accounts, also called equity accounts, are used to track stocks or shares in the company. Equity accounts are also used to track all sales, revenues, purchases and expenses.

As a general rule of thumb, accounts that track actual resources are debit accounts. For these accounts a debit entry means an increase, and a debit balance means there are resources in the account. For all other accounts a credit means an increase, and a credit balance represents a positive net value.

The Basic Transaction Record

As I mentioned in the previously cited article, every account has a unique ID, a number, and a name. Each account also stores the net total of all items debited or credited to the account. This is an ultra-simple model for an account, but this is enough for our purposes.

The basic transaction consists of a header and a list of components. The components refer to the accounts and each component stores the amount debited or credited to the corresponding account. The header stores information that applies to the transaction as a whole:

- date,
- ID number,
- type of transaction (if you're distinguishing different types),
- comment.

To ensure that the data contained in each transaction are complete and correct, we impose integrity constraints. These represent both database design and accounting conditions, and must hold for both new and modified transactions. Specifications and constraints are shown in Figure 3.

Basic Data Specifications

- Transaction header:
  - ID (longint),
  - entry date,
  - transaction type (alphanumeric code),
  - comment (text).
- Components (any number):
  - account ID (longint),
  - debit amount (real).

Basic Integrity Constraints

- Transaction:
  - transaction ID must be unique,
  - entry date must lie within some reasonable range,
  - transaction type must be one of several predefined values.
- Components:
  - referenced accounts must exist,
  - debit total must equal credit total,

Figure 3: Transaction data specifications and integrity constraints.

The basic transaction can be stored in the many-to-one file structure shown in Figure 4. In this structure the components are not assigned unique record IDs, which greatly simplifies record management. This would be inappropriate in the case where the components themselves were referred to by other records. This structure assumes that the structure will not be extended in this manner.

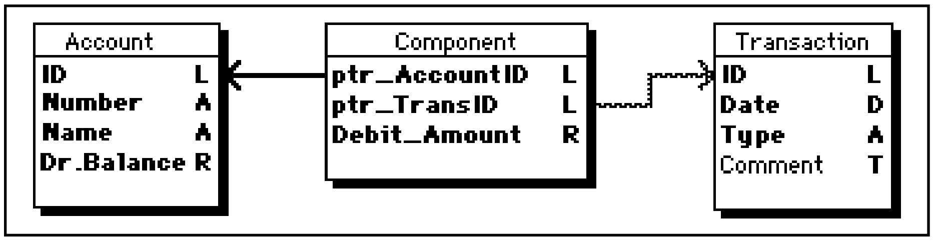

Figure 4: The basic transaction file structure.

Complex Transactions and a Preview

Most transactions contain additional information, are subject to additional rules, or both. A sales transaction, for example, would have to have a sales account among its components; a credit purchase would have to have a due date. Other transaction types involve a great deal more than the basic header and component structure. You can be sure that businesses requiring custom software will need to process complex transactions.

We can classify transactions according to the amount of complexity that arises from each of three sources:
• the number and nature of their component accounts,
• their entry methods,
• the related data that they handle.

In each of the subequent articles in this series I'll consider a type of transaction that exemplifies one of these factors and I'll develop a programming technique particular to that case. This classification system and the techniques I will use are summarized below.

1- Component accounts
Some transactions need to be treated differently because they involve specific accounts. A cash withdrawal is an example. Its particular constraints include sequential numbering, withdrawal approval; its particular processes include check printing and deferred check printing.

In my next article I'll handle this case by storing cash transactions along with other types of transactions in a common transactions file. I'll identify them by assigning a unique tag, and I'll process them with special input and output layouts.

2- Entry methods
Some transactions, such as point-of-sale (POS) entries, require the fastest possible data entry. While POS entries require speed, they do not usually require the immediate updating of account balances. You can achieve immediate speed gains, at the expense of temporary inconsistencies in the data, by accepting the entry without fully processing it. All partially processed entries are completed at a later, more convenient time. This describes a method known as batch processing. I'll describe batch processing in detail in the third article of this series.

3- Related data
Transactions tracking special types of events will store specialized information and require special processing. In these cases accounting information may only be part of the transaction's information. Accounting rules are still in effect but a large share of the data processing will be dictated by business rules.

Here I'm using the notion of business rules broadly. When I say "business rules" I mean any specification that is neither in the sphere of accounting nor internal to database design. These auxiliary specs could just as well be rooted in medicine, science, or politics.

An invoice is an example of a transaction that records and is processed according to both accounting and business rules. An invoice records both the movement of assets and also tracks the fulfillment of, and payment for an order, the terms of payment, as well as shipping, tax, and commission information. This is not to mention goods and services, which are also usually tracked through the invoice.

In the fourth article of this series I handle the invoice transaction using an extended file structure. There I'll employ a related-one file that's linked to the generic transaction record described above. A single invoice record will be associated with a generic transaction record whenever invoice transaction information is entered. In this structure the generic transaction record will store accounting information while the invoice record will store the extra, non-accounting information.

Debits and Credits Are Dead

All transactions boil down to debits and credits. The total amount being debited always equals the total amount being credited. A transaction involving any set of accounts is acceptable, from an accounting view point, if the debits equal the credits.

All of the debiting and crediting affects our own accounts, and each account is either of debit-type or credit-type. The value associated with an account that represent the accumulation of actual resources (asset accounts) is measured by their debit balance. The value associated with accounts that represent either debts or business ownership (liability and equity accounts) is measured by their credit balance.

Before you make any attempt to construct or interpret a transaction, you must distinguish the debit accounts from the credit accounts. Understanding account types is the key to understanding a transaction's meaning. Until you know the difference, accounting remains impenetrable. Once you can identify the accounts, the rules of accounting become incredibly simple.