HOW TO DESIGN A DATABASE

By Robert N. West, CPA, Ph.D.

As more and more information processing migrates from the MIS group out to the various user departments, database design is becoming an increasingly valued skill. The accounting department has as much data to analyze as any other user department. Yet although many management accountants and financial managers know how to use a database package to extract data and print reports, few know how to *design* a database. Therefore, this article focuses not on using a database package such as Microsoft Access, but rather on the more conceptual (and more important) issue of designing a database properly.

The intended audience is management accountants who want to use a database for analysis and ad hoc reporting purposes. For example, with the information from this article you could build local databases for project tracking, human resource analysis, tracking short-term investments for the cash management function, monitoring inventory, and so on. But the level of knowledge anyone gains from a short article is not sufficient to design a mission-critical database system—not even for a small business. There are many excellent database books and many excellent database consultants for larger projects.

Spreadsheet Skills vs. Database Skills

Most financial managers and management accountants possess strong spreadsheet skills, but database design differs from spreadsheet design. Spreadsheet design carries very few rules, and, as a result, spreadsheets are easy to create. But not all spreadsheets are clear and logical, and most lack data integrity controls. When spreadsheets lack clarity and logic, the data and formulas are difficult to modify, and errors typically result. Database design, on the other hand, is more formal and has rules of structure. Errors are less likely to occur with a properly designed database, and it is easier to extract data and obtain reports from a properly designed database. Unfortunately, many databases are improperly designed, which can lead to data integrity problems similar to those experienced with spreadsheets.

Because building a database is a creative process involving much judgment, it helps to see an example. In the following pages, I will build a database, discuss design rules, and illustrate where judgment is required. I encourage you to work through this exercise with me. The database we will create is shown as Table 1 at the end of the article.

CASE DESCRIPTION

WebSoft is a Web-based software retailer that sells to both commercial and consumer markets. Commercial customers are typically interested in the accounting, database, spreadsheet, word processing, Web productivity, and related software products. Consumers primarily are interested in educational and computer game software, but many also buy the same products as corporate/business customers. WebSoft considers an individual to be a prospective customer as soon as he or she inquires (on the company's toll-free telephone line or on its website) about a piece of software, even though the person might not order anything. A customer can order any number of WebSoft's software products at a time. Unfortunately, WebSoft has trouble keeping up with the demand for its products, and often it must ship partial orders (each shipment is only for a single order). WebSoft considers a sale to be made at the time the software is shipped. Customers pay for the software in a variety of ways. Individuals generally pay for the products in full when they receive an invoice with a particular shipment. Corporate customers are billed at the end of the month for all the software shipped during that particular month. (The average is four shipments to a corporation per month.) Some corporations, because of the large size of their purchases, will pay for a particular shipment over several months, whereas others pay the invoice in full. All cash receipts are deposited intact at the end of the day in the bank.

Building the Database

The steps in building a database are as follows:

1. Identify the **entities** of interest (and put boundaries on the system);
2. Define the **relationships** between the entities, and establish the tables in the database;
3. Identify the **primary key** field for each table;
4. List the **attributes (fields)** for each table; and
5. Establish **referential integrity** and **foreign key** relationships.

Identify the Entities of Interest

The first step in designing a database is to decide what broad categories of information, or *entities*, you want to track. Try it with the case description on p. 19.

I have selected the following entities:

Customers,
Orders,
Sales/Shipments,
Payments,
Inventory,
Inquiries, and
Cash/Bank Activity.

There are a few uncertainties. How should prospective customer inquiries be handled? Should a customer record be set up for every phone inquiry? From an accounting perspective probably not, but from a marketing perspective the company may want to capture these prospects in the customer file. For this illustration, I won't track inquiries/prospects.

Should the bank be set up as a separate entity? Probably not, unless funds are deposited in several bank accounts. Should shipments and sales be separate entities? Probably, but in this example I'll just focus on the accounting functions and sales/billing and ignore the shipping/logistics database information. Should there be an accounts receivable entity? Accounts receivable is a function of a customer and his/her purchases and payments (and returns, but these are ignored in this exercise). My database will include that information, but accounts receivable is not an entity (more discussion will follow).

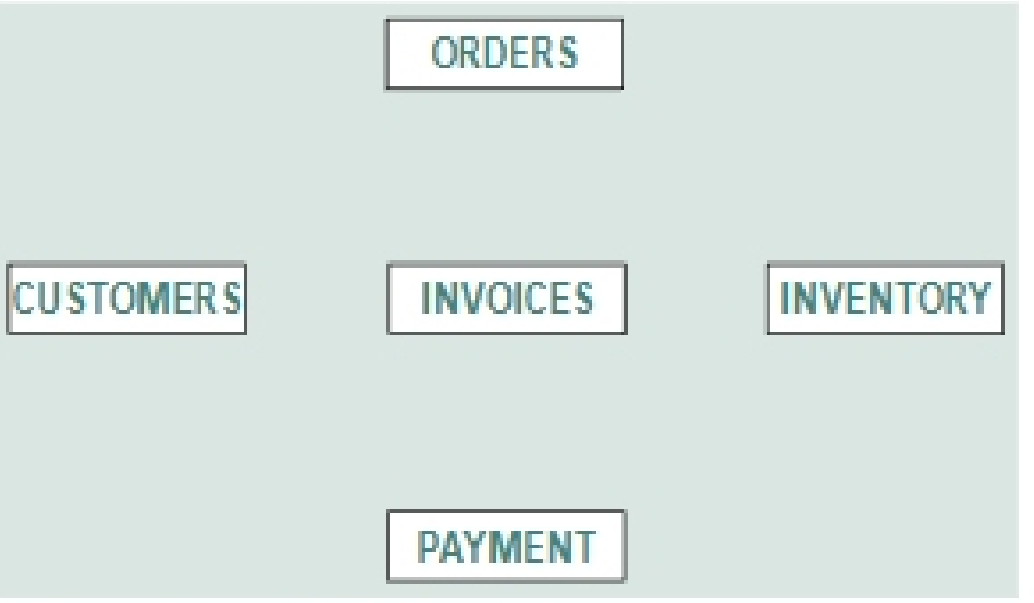
Should there be two inventory entities, software games and business software? No. The type of software is an attribute of the inventory entity. By resolving these questions, I have defined the boundaries of the database. It is a good idea to document what it includes and does not include.

So far, I see the need for five tables, one for each entity of interest:

Customers,
Orders,
Sales/Shipments,
Payments, and
Inventory.

Define the Relationships Among the Five Entities

As a database is a collection of related tables, the second step in database design is to define the relationships among the existing entities (tables). When completing this process—building an Entity-Relationship model—you must have an understanding of the underlying processes you are modeling. Create a diagram of the five entities, and draw a rectangle around each of the entity names.



Take each pair of entities, and determine how they are related to one another (if at all).

First, take Customers and Orders.

Ask these questions in the following manner:

Can a customer place more than one order?

Yes. Draw a line from the customer entity to the order entity with a double-headed arrow to signify "more than one" order can be placed.

Can an order be from more than one customer?

No. A specific order is from only one customer. Draw a single-headed arrow from order to customer.

The relationship of customers and orders is said to be *one-to-many (1:m)*. This has important implications in database design, which I will discuss later.

Next, pair Sales/Invoices and Orders.

Can an invoice be for more than one order?

No, not for this company anyway.

Can an order have more than one invoice?

Yes. We call that a backorder. Some orders require two or more shipments, and therefore invoices do as well, although the norm is probably one invoice per order. Databases, however, must be designed to deal with "the exceptions" to the norm. Once again, the relationship is one-to-many.

The questions for Inventory and Orders:

Can an order have more than one inventory item?

The answer depends on the meaning of the question. Two possibilities:

1) Can a customer order two (or more) of the