Important Database Concepts

In This Chapter

- Using a database to get past Excel limitations
- Getting familiar with database terminology
- Understanding relational databases
- How databases are designed

Although Excel is traditionally considered the premier tool for data analysis and reporting, it has some inherent characteristics that often lead to issues revolving around scalability, transparency of analytic processes, and confusion between data and presentation. Over the last several years, Microsoft has recognized this and created tools that allow you to develop reporting and business intelligence by connecting to various external databases. Microsoft has gone a step further with Excel 2013, offering business intelligence (BI) tools like Power Pivot natively; it effectively allows you to build robust relational data models within Excel.

With the introduction of these BI tools, it's becoming increasingly important for you to understand core database fundamentals. Unlike traditional Excel concepts, where the approach to developing solutions is relatively intuitive, good database-driven development requires a bit of prior knowledge. There are a handful of fundamentals you should know before jumping into the BI tools. These include database terminology, basic database concepts, and database best practices.

The topics covered in this chapter explain the concepts and techniques necessary to successfully use database environments and give you the skills needed to normalize data and plan and implement effective tables.

If you're already familiar with the concepts involved in database design, you may want to skim this chapter. If you're new to the world of databases, spend some time in this chapter gaining a thorough understanding of these important topics.

Traditional Limits of Excel and How Databases Help

Managers, accountants, and analysts have had to accept one simple fact over the years: Their analytical needs had outgrown Excel. They all met with fundamental issues that stemmed from one or more of Excel's three problem areas: scalability, transparency of analytical processes, and separation of data and presentation.

Scalability

Scalability is the ability for an application to develop flexibly to meet growth and complexity requirements. In the context of Excel, scalability refers to Excel's ability to handle ever-increasing volumes of data. Most Excel aficionados are quick to point out that as of Excel 2007, you can place 1,048,576 rows of data into a single Excel worksheet. This is an overwhelming increase from the limitation of 65,536 rows imposed by previous versions of Excel. However, this increase in capacity does not solve all of the scalability issues that inundate Excel.

Imagine that you're working in a small company and using Excel to analyze your daily transactions. As time goes on, you build a robust process complete with all the formulas, PivotTables, and macros you need to analyze the data that is stored in your neatly maintained worksheet.

As your data grows, you start to notice performance issues. Your spreadsheet becomes slow to load and then slow to calculate. Why does this happen? It has to do with the way Excel handles memory. When an Excel file is loaded, the entire file is loaded into RAM. Excel does this to allow for quick data processing and access. The drawback to this behavior is that each time something changes in your spreadsheet, Excel has to reload the entire spreadsheet into RAM. A large spreadsheet takes a great deal of RAM to process even the smallest change. Eventually, each action you take in your gigantic worksheet will result in an excruciating wait.

Your PivotTables will require bigger *pivot caches* (memory containers), almost doubling your Excel workbook's file size. Eventually, your workbook will become too big to distribute easily. You may even consider breaking down the workbook into smaller workbooks (possibly one for each region). This causes you to duplicate your work.

In time, you may eventually reach the 1,048,576-row limit of your worksheet. What happens then? Do you start a new worksheet? How do you analyze two datasets on two different worksheets as one entity? Are your formulas still good? Will you have to write new macros?

These are all issues that need to be dealt with.

You can find various clever ways to work around these limitations. In the end, though, they are just workarounds. Eventually you will begin to think less about the most effective way to perform and present analysis of your data and more about how to make something "fit" into Excel without breaking your formulas and functions. Excel is flexible enough that you can make most things "fit" into Excel just fine. However, when you think only in terms of Excel, you're limiting yourself, albeit in an incredibly functional way.

In addition, these capacity limitations often force you to have the data prepared for you. That is, someone else extracts large chunks of data from a large database, then aggregates and shapes the data for use in Excel. Should you always depend on someone else for your data needs? What if you have the tools to "access" vast quantities of data without relying on others to provide data? Could you be more valuable to the organization? Could you focus on the accuracy of the analysis and the quality of the presentation instead of routing Excel data maintenance?

A relational database system (like Access or SQL Server) is a logical next step. Most database system tables take very few performance hits with larger datasets and have no predetermined row limitations. This allows you to handle larger datasets without requiring the data to be summarized or prepared to fit into Excel. Also, if a process becomes more crucial to the organization and needs to be tracked in a more "enterprise-acceptable" environment, it's easier to upgrade and scale up if that process is already in a relational database system.

Transparency of analytical processes

One of Excel's most attractive features is its flexibility. Each individual cell can contain text, a number, a formula, or practically anything else you define. Indeed, this is one of the fundamental reasons Excel is such an effective tool for data analysis. You can use named ranges, formulas, and macros to create an intricate system of interlocking calculations, linked cells, and formatted summaries that work together to create a final analysis.

The problem with that is there is no transparency of analytical processes, meaning it is extremely difficult to determine what is actually going on in a spreadsheet. If you've ever had to work with a spreadsheet created by someone else you know all too well the frustration that comes with deciphering the various gyrations of calculations and links being used to perform an analysis. Small spreadsheets that perform a modest analysis are painful to decipher but are usually still workable, while large, elaborate, multi-worksheet workbooks are virtually impossible to decode, often leaving you to start from scratch.

Compared to Excel, database systems might seem rigid, strict, and unwavering in their rules. However, all this rigidity comes with a benefit.

Because only certain actions are allowable, you can more easily come to understand what is being done within structured database objects, such as queries or stored procedures. If a dataset is being edited, a number is being calculated, or any portion of the dataset is being affected as a part of an analytical process, you can readily see that action by reviewing the query syntax or reviewing the stored procedure code. Indeed, in a relational database system, you never encounter hidden formulas, hidden cells, or dead named ranges.

Separation of data and presentation

Data should be separate from presentation; you do not want the data to become too tied into any one particular way of presenting it. For example, when you receive an invoice from a company, you don't assume that the financial data on that invoice is the true source of your data. It is a presentation of your data. It can be presented to you in other manners and styles on charts or on Web sites, but such representations are never the actual source of the data.

What exactly does this concept have to do with Excel? People who perform data analysis with Excel tend to fuse the data, the analysis, and the presentation together. For example, you often see an Excel workbook that has 12 worksheets, each representing a month. On each worksheet, data for that month is listed along with formulas, PivotTables, and summaries. What happens when you're asked to provide a summary by quarter? Do you add more formulas and worksheets to consolidate the data on each of the month worksheets? The fundamental problem in this scenario is that the worksheets actually represent data values that are fused into the presentation of your analysis. The point here is that data should not be tied to a particular presentation, no matter how apparently logical or useful it may be. However, in Excel, it happens all the time.

In addition, because all manners and phases of analysis can be done directly within a spreadsheet, Excel cannot effectively provide adequate transparency to the analysis. Each cell has the potential of holding hidden formulas and containing links to other cells. In Excel, the line between analysis and data is blurred, which makes it difficult to determine exactly what is going on in a spreadsheet. Moreover, it takes a great deal of effort in the way of manual maintenance to ensure that edits and unforeseen changes don't affect previous analyses.

Relational database systems inherently separate analytical components into tables, queries, and reports. By separating these elements, databases make data less sensitive to changes and create a data analysis environment where you can easily respond to new requests for analysis without destroying previous analyses.

In these days of big data, there are more demands for complex data analysis, not fewer. You have to add some tools to your repertoire to get away from being simply "spreadsheet mechanics." Excel can be stretched to do just about anything, but maintaining such "creative" solutions can be a tedious manual task. You can be sure that the exciting part of data analysis is not in routine data management within Excel. Rather, it is in leveraging of BI tools to provide your clients with the best solution for any situation.

Database Terminology

The terms *database, table, record, field,* and *value* indicate a hierarchy from largest to smallest. These same terms are used with virtually all database systems, so you should learn them well.

Databases

Generally, the word *database* is a computer term for a collection of information concerning a certain topic or business application. Databases help you organize this related information in a logical fashion for easy access and retrieval. Some older database systems used the term *database* to describe individual tables. Current use of *database* applies to all elements of a database system.

Databases aren't only for computers. There are also manual databases; sometimes they're referred to as manual filing systems or manual database systems. These filing systems usually consist of people, folders, and filing cabinets — and paper, which is the key to a manual database system. In a real manual database system, you probably have in/out baskets and some type of formal filing method.

You access information manually by opening a file cabinet, taking out a file folder, and finding the correct piece of paper. Customers fill out paper forms for input, perhaps by using a keyboard to input information that is printed on forms. You find information by manually sorting the papers or by copying information from many papers to another piece of paper (or even into an Excel spreadsheet). You may use a spreadsheet or calculator to analyze the data or display it in new and interesting ways.

Tables

Databases store information in carefully defined structures called *tables.* A table is just a container for raw information (called *data*), similar to a folder in a manual filing system. Each table in a database contains information about a single entity, such as a person or product, and the data in the table is organized into rows and columns. A relational database system stores data in related tables. For example, a table containing employee data (names and addresses) may be related to a table containing payroll information (pay date, pay amount, and check number).


Note

In database-speak, a table is an object. As you design and work with databases, it's important to think of each table as a unique entity and consider how each table relates to the other objects in the database.

In most database systems, you can view the contents of a table in a spreadsheet-like form, called a *datasheet*, comprising rows and columns (known as *records* and *fields*, respectively — see the following section, "Records, fields, and values"). Although a datasheet and a spreadsheet are superficially similar, a datasheet is a very different type of object. You typically cannot make changes or add calculations directly within a table. Your interaction with tables primarily comes in the form of queries or views (see the later section, "Queries").

Records, fields, and values

A database table is divided into rows (called *records*) and columns (called *fields*), with the first row (the heading at the top of each column) containing the names of the fields in the database.

Each row is a single record containing fields that are related to that record. In a manual system, the rows are individual forms (sheets of paper), and the fields are equivalent to the blank areas on a printed form that you fill in.

Each column is a field that includes many properties that specify the type of data contained within the field, and how the database should handle the field's data. These properties include the name of the field (for example, CompanyName) and the type of data in the field (for example Text). A field may include other properties as well. For example, a field's Size property tells the database the maximum number of characters allowed for the address.

At the intersection of a record and a field is a *value* — the actual data element. For example, if you have a field called CompanyName, a company name entered into that field would represent one data value.


Note

When working with Access, the term *field* is used to refer to an attribute stored in a record. In many other database systems, including SQL Server, *column* is the expression you'll hear most often in place of field. Field and column mean the same thing. The exact terminology used relies somewhat on the context of the database system underlying the table containing the record.

Queries

Most relational database systems allow the creation of queries (sometimes called *views*). Queries extract information from the database tables. A query selects and defines a group of records that fulfill a certain condition. Most database outputs are based on queries that combine, filter, or sort data before it's displayed. Queries are often called from other database objects, such as stored procedures, macros, or code modules. In addition to extracting data from tables, queries can be used to change, add, or delete database records.

An example of a query is when a person at the sales office tells the database, "Show me all customers, in alphabetical order by name, who are located in Massachusetts and bought something over the past six months." Or "Show me all customers who bought Chevrolet car models within the past six months and sort them by customer name and then by sale date."

Instead of asking the question in words to query a database, you use a special syntax such as SQL (Structured Query Language).

How Databases Are Designed

The better a database is designed or structured, the better the reporting solutions are able to leverage the data within it. The design process of a database is not all that mysterious. The basic design steps described in this section provide a solid understanding of how best to think about and even design your own databases.

Step 1: The overall design — from concept to reality

All solution developers face similar problems, the first of which is determining how to meet the needs of the end client. It's important to understand the overall client's requirements before zeroing in on the details.

For example, a client may ask for a database that supports the following tasks:

- Entering and maintaining customer information (name, address, and financial history)
- Entering and maintaining sales information (sales date, payment method, total amount, customer identity, and other fields)
- Entering and maintaining sales line-item information (details of items purchased)
- Viewing information from all the tables (sales, customers, sales line items, and payments)

- Asking questions about the information in the database
- Producing a monthly invoice report
- Producing a customer sales history
- Producing mailing labels and mail-merge reports

When reviewing these eight tasks, database designers need to consider other peripheral tasks that weren't mentioned by the client. Before jumping into design, database designers typically prepare a series of questions that provide insight to the client's business and how the client uses data. For example, a database designer might ask these questions:

- What reports and forms are currently used?
- How are sales, customers, and other records currently stored?
- How are invoices processed?

As these types of questions get answered, database designers get a feel for the business process, how data should be structured, and what, if any, integration with other data systems need to be considered.

Step 2: Report design

Database designers often consider the types of reports needed when modeling a database. Although it may seem odd to start with output reports, in many cases, customers are more interested in the printed output from a database than they are in any other aspect of the application. Reports often include every bit of data managed by an application. Because they tend to be comprehensive, reports are often the best way to gather important information about a database's requirements.

Step 3: Data design

The next step in the design phase is to take an inventory of all the information needed by the reports. One of the best methods is to list the data items in each report. As database designers do so, they take careful note of items that are included in more than one report, making sure they keep the same name for a data item that is in more than one report because the data item is really the same item.

For example, note all the customer data needed for each report shown in in Table 1-1.

Table 1-1: Customer-Related Data Items Found in the Reports

Customer Report	Invoice Report
Customer Name	Customer Name
Street	Street
City	City
State	State

continued

Table 1-1: Customer-Related Data Items Found in the Reports *(continued)*

Customer Report	Invoice Report
Zip Code	Zip Code
Phone Number	Phone Number
E-Mail Address	
Web Site	
Discount Rate	
Customer Since	
Last Sales Date	
Sales Tax Rate	
Credit Information (four fields)	

As you can see by comparing the type of customer information needed for each report, there are several common fields. Most of the customer data fields are found in both reports. Table 1-1 shows only some of the fields that are used in each report — those related to customer information. Because the related row and the field names are the same, a database designer can make sure all the data items are included in a customer table in the database. Table 1-2 lists the fields in a needed Invoice Report that contains sales information.

Table 1-2: Sales Data Items Found in the Reports

Invoice Report	Line Item Data
Invoice Number	
Sales Date	
Invoice Date	
Payment Method	
Salesperson	
Discount (overall for sale)	
Tax Location	
Tax Rate	
Product Purchased (multiple lines)	Product Purchased
Quantity Purchased (multiple lines)	Quantity Purchased
Description of Item Purchased (multiple lines)	Description of Item Purchased
Price of Item (multiple lines)	Price of Item
Discount for Each Item (multiple lines)	Discount for Each Item
Payment Type (multiple lines)	
Payment Date (multiple lines)	
Payment Amount (multiple lines)	
Credit Card Number (multiple lines)	
Expiration Date (multiple lines)	

As you can see when you examine the type of sales information needed for the report, there are a few repeating items (fields) — for example, Product Purchased, Quantity Purchased, and Price of Item. Each invoice can have multiple items, and each of these items needs the same type of information — number ordered and price per item. Many sales have more than one purchased item. Also, each invoice may include partial payments, and it's possible that this payment information will have multiple lines of payment information, so these repeating items can be put into their own grouping.

This type of report leads you to create two tables: one table to hold the top-level invoice data such as invoice number, invoice data, and sales person; and another table to hold line item details such as the products purchased, quantity purchased, and purchase price.

Step 4: Table design

After determining the tables needed, you evaluate the fields and calculations that are needed to fulfill the reporting requirements. Initially, only the fields included in the reports are added to the tables. Other fields may be added later (for various reasons), although certain fields won't appear in any table.

It's important to understand that not every little bit of data must be added into the database's tables. For example, clients may want to add vacation and other out-of-office days to the database to determine which employees are available on a particular day. However, it's easy to burden a database's initial design by incorporating too many ideas during the initial development phases. In general, you can accommodate client requests after the database development project is underway.

After all the tables and fields are determined, database designers consolidate the data by purpose (for example, grouped into logical groups) and then compare the data across those functions. For example, customer information is combined into a single set of data items. The same action is taken for sales information and line-item information. Table 1-3 compares data items from these three groups of information.

Table 1-3: Comparing the Data Items

Customer Data	Invoice Data	Line Items
Customer Company Name	Invoice Number	Product Purchased
Street	Sales Date	Quantity Purchased
City	Invoice Date	Description of Item Purchased
State	Payment Method	Price of Item
Zip Code		Discount for Each Item
Phone Numbers (two fields)	Discount (overall for this sale)	Taxable?
E-Mail Address	Tax Rate	
Web Site	Payment Type (multiple lines) Payment Date (multiple lines)	
Discount Rate	Payment Amount (multiple lines)	
Customer Since	Credit Card Number (multiple lines)	

continued

Table 1-3: Comparing the Data Items *(continued)*

Customer Data	Invoice Data	Line Items
Last Sales Date	Expiration Date (multiple lines)	
Sales Tax Rate		
Credit Information (four fields)		

Consolidating and comparing data is a good way to start creating the individual table, but the customer data must be split into two groups. Some of these items are used only once for each customer, while other items have multiple entries. For example, in the Sales column, the payment information can have multiple lines of information.

For example, one customer can have multiple contacts with the company. Another customer may make multiple payments toward a single sale. Of course, for this example, the data goes into three categories: customers, invoices, and sales line items.

Keep in mind that one customer may have multiple invoices, and each invoice may have multiple line items on it. The invoice category contains information about individual sales and the line items category contains information about each invoice. Notice that these three columns are all related; for example, one customer can have multiple invoices and each invoice may require multiple detail lines (line items).

Why multiple tables?

The prospect of creating multiple tables almost always intimidates beginning database users. Most often, beginners want to create one huge table that contains all the information they need — for example, a customer table with all the sales placed by the customer and the customer's name, address, and other information. After all, if you've been using Excel to store data so far, it may seem quite reasonable to take the same approach when building tables in a database.

A single large table for all customer information quickly becomes difficult to maintain, however. You have to input the customer information for every sale a customer makes (repeating the name and address information in every row). The same is true for the items purchased for each sale when the customer has purchased multiple items as part of a single purchase. This makes the system more inefficient and prone to data-entry mistakes. The information in the table is inefficiently stored — certain fields may not be needed for each sales record — and the table ends up with a lot of empty fields.

You want to create tables that hold the minimum of information while still making the system easy to use and flexible enough to grow. To accomplish this, you need to consider making more than one table, with each table containing fields that are only related to the focus of that table. Then, after you create the tables, you link them so that you're able to glean useful information from them. Although this process sounds complex, the actual implementation is relatively easy.

The relationships between tables can be different. For example, each sales invoice has one and only one customer, while each customer may have multiple sales. A similar relationship exists between the sales invoice and the line items of the invoice.

Database table relationships require a unique field in both tables involved in a relationship. A unique identifier in each table helps the database engine to properly join and extract related data.

Only the sales table has a unique identifier (InvoiceNumber), which means at least one field must be added to each of the other tables to serve as the link to other tables; for example, adding a CustomerID field to tblCustomers, adding the same field to the invoice table, and establishing a relationship between the tables through CustomerID in each table. The database engine uses the relationship between customers and invoices to connect customers with their invoices. Relationships between tables is done through key fields. Chapter 3 shows you how to create relationships between key fields in tables.

When you understand the need for linking one group of fields to another group, you can add the required key fields to each group. Table 1-4 shows two new groups and link fields created for each group of fields. These linking fields, known as *primary* and *foreign* keys, are used to link these tables.

Table 1-4: Tables with Keys

Customer Data	Invoice Data	Line Items Data	Sales Payment Data
CustomerID	InvoiceID	InvoiceID	InvoiceID
Customer Name	CustomerID	Line Number	Payment Type
Street	Invoice Number	Product Purchased	Payment Date
City	Sales Date	Quantity Purchased	Payment Amount
State	Invoice Date	Description of Item Purchased	Credit Card Number
ZIP Code	Payment Method	Price of Item	Expiration Date
Phone Numbers (two fields)	Salesperson	Discount for Each Item	
E-Mail Address			
Web Site			
Discount Rate			
Customer Since			
Last Sales Date			
Sales Tax Rate	Tax Rate		

The field that uniquely identifies each row in a table is the primary key. The corresponding field in a related table is the foreign key. In this example, CustomerID in tblCustomers is a primary key, while CustomerID in tblInvoices is a foreign key.

Assume a certain record in tblCustomers has 12 in its CustomerID field. Any records in Invoices with 12 as its CustomerID is "owned" by customer 12.

With the key fields added to each table, you can now find a field in each table that links it to other tables in the database. For example, Table 1-4 shows CustomerID in both the Customer table (where it's the primary key) and the Invoice table (where it's a foreign key).

This way, you're not repeating information on every row, you're just providing a link back to the table containing the information that may be required to show up on the report or invoice. The database will handle retrieving all related information so you don't have to worry about it — the only thing you have to define is the link between the tables.