

Designing a Database -- Understanding Relational Design

Contents

- OVERVIEW3**
- THE DATABASE DESIGN PROCESS3**
 - STEPS IN DESIGNING A DATABASE4
 - COMMON DESIGN PROBLEMS4
- DETERMINING THE PURPOSE.....5**
- DETERMINING THE TABLES YOU NEED.....6**
- DETERMINING THE FIELDS YOU NEED.....7**
 - TIPS FOR DETERMINING FIELDS8
 - PRIMARY KEY FIELDS9
- DETERMINING THE RELATIONSHIPS11**
- REFINING THE DESIGN17**
- ADDITIONAL READING AND PRACTICE.....18**

Overview

Microsoft® Access provides a number of tools that you can use to create a relational database even if you don't have much experience with relational design. For example, you can use the Database Wizard to create ten predefined types of databases, from an Asset Tracking database to a Time and Billing database.

Or, if you already have your data in a spreadsheet or table of some sort, but you have repeating data (unnormalized data) and you want to separate the data out into two or more relational Microsoft Access tables, you can use the Table Analyzer Wizard to help you decide which fields need to be moved to separate tables.

If you're not satisfied with having Microsoft Access do the work for you, however, and you want to know more about relational database design, this document is for you. It shows you how to plan and to design a database from the ground up. For practical examples, it uses the database design of the Northwind Traders sample database included in the Microsoft Access package.

The Database Design Process

The key to understanding the database design process lies in understanding the way a relational database management system, such as Microsoft Access, stores data. To efficiently and accurately provide you with information, Microsoft Access needs to have the facts about different subjects stored in separate tables. For example, you might have one table that stores only facts about employees, and another that stores only facts about sales.

When you use your data, you then combine and present facts in many different ways. For example, you may print reports that combine facts about employees and facts about sales.

Store all facts about Employees...

Employees : Table			
Employee ID	Last Name	First Name	Title
1	Davolio	Nancy	Sales Representative
2	Fuller	Andrew	Vice President, Sales
3	Leverling	Janet	Sales Representative
4	Peterson	Mary	Sales Representative
5	Rajnarayan	Vinay	Sales Representative
6	Shelley	Stephanie	Sales Representative
7	Todman	Michelle	Sales Representative
8	Wood	Glenn	Sales Representative
9	Zappala	Antonio	Sales Representative

... and all facts about Sales...

Orders : Table		
Order ID	Customer	Employee
10248	Vins et alcools Chevalier	Buchanan, Steven
10249	Toms Spezialitäten	Suyama, Michael
10250	Hanari Carnes	Peacock, Margaret
10251	Victuailles en stock	Leverling, Janet
10252	Suprêmes délices	Peacock, Margaret
10253	Hanari Carnes	Leverling, Janet
10254	Chop-suey Chinese	Buchanan, Steven
10255	Richter Supermarkt	Dodsworth, Anne
10256	Wellington Importadora	Leverling, Janet

...in two separate tables.

When you design a database, you first break down the information you want to keep as separate subjects, and then you tell Microsoft Access how the subjects are related to each other so that Microsoft Access can bring the right information together when you need it.

Steps in Designing a Database

Here are the steps in the database design process. Each step is discussed in greater detail in the remaining sections of this paper.

Step One: Determine the purpose of your database. This will help you decide which facts you want Microsoft Access to store.

Step Two: Determine the tables you need. Once you have a clear purpose for your database, you can divide your information into separate subjects, such as "Employees" or "Orders." Each subject will be a table in your database.

Step Three: Determine the fields you need. Decide what information you want to keep in each table. Each category of information in a table is called a *field* and is displayed as a column in the table. For example, one field in an Employees table could be Last Name; another could be Hire Date.

Step Four: Determine the relationships. Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.

Step Five: Refine your design. Analyze your design for errors. Create the tables and add a few records of sample data. See if you can get the results you want from your tables. Make adjustments to the design as needed.

Don't worry if you make mistakes or leave things out of your initial design. Think of it as a rough draft that you can refine later. Experiment with sample data and prototypes of your forms and reports. With Microsoft Access, it's easy to change the design of your database as you're creating it. However, it becomes much more difficult to make changes to tables after they're filled with data and after you've built forms and reports. For this reason, make sure that you have a sound design before pushing too far ahead.

Common Design Problems

There are several common pitfalls you may encounter when designing your database. These problems can cause your data to be harder to use and maintain. The following are signs that you should reevaluate your database design:

- You have one table with a large number of fields that don't all relate to the same subject. For example, one table might contain fields pertaining to your customers as well as fields that contain sales information. Try to make sure each table contains data about only one subject.
- You have fields that are intentionally left blank in many records because they aren't applicable to those records. This usually means that the fields belong in another table.
- You have a large number of tables, many of which contain the same fields. For example, you have separate tables for January sales and February sales, or for local customers and remote customers, in which you store the same type of information. Try consolidating all the information pertaining to a single subject in one table. You may also need to add an extra field, for example, to identify the sales date.

Determining the Tables You Need

Determining the tables in your database can be the trickiest step in the database design process. That's because the results you want from your database -- the reports you want to print, the forms you want to use, the questions you want answered -- don't necessarily provide clues about the structure of the tables that produce them. They tell you what you want to know, but not how to categorize the information into tables.

See the preceding order form as an example. It includes facts about the customer -- the customer's address and phone number -- along with facts about the order. This form provides you with a number of facts that you know you want to store in your database. But you'd run into problems if you stored the customer facts in the same table as the order facts:

- **Introducing errors in duplicate information.** Suppose that one customer places three different orders. You could add the customer's address and phone number to your database three times, once for each order. But this multiplies the chance of data entry errors.

Which address is correct?

Order ID	Order Date	Ship Name	Ship Address
11063	30-Apr-1998	Hungry Owl All-Night Grocers	8 Johnstown Road
10985	30-Mar-1998	Hungry Owl All-Night Grocers	28 Johnstown Road
10736	11-Nov-1997	Hungry Owl All-Night Grocers	8 Johnstown Road
10321	03-Oct-1996	Island Trading	Garden House
10315	26-Sep-1996	Island Trading	Garden House

For accuracy, store each fact only once.

Customer ID	Company Name	Address
HUNGO	Hungry Owl All-Night Grocers	8 Johnstown Road
ISLAT	Island Trading	Garden House
KOENE	Königlich Essen	Maubelstr. 90
LACOR	La corne d'abondance	67, avenue de l'Europe
LAMAI	La maison d'Asie	1 rue Alsace-Lorraine
LAUGB	Laughing Bacchus Wine Cellars	1900 Oak St.

Also, if the customer moves, you'd have to either accept contradictory information or find and change each of that customer's sales records in the table. It's much better to create a Customers table that stores the customer's address in your database *once*. Then if you need to change the data, you change it only once.

- **Deleting valuable information.** Suppose a new customer places an order and then cancels. When you delete the order from the table containing information on both customers and their orders, you would delete the customer's name and address as well. But you want to keep this new customer in your database so you can send the customer your next catalog. Again, it's better to put the information about the customer in a separate Customers table. That way you can delete the order without deleting customer information.

Look at the information you want to get out of your database and divide it into fundamental subjects you want to track, such as customers, employees, products you sell, services you provide, and so on. Each of these subjects is a candidate for a separate table.

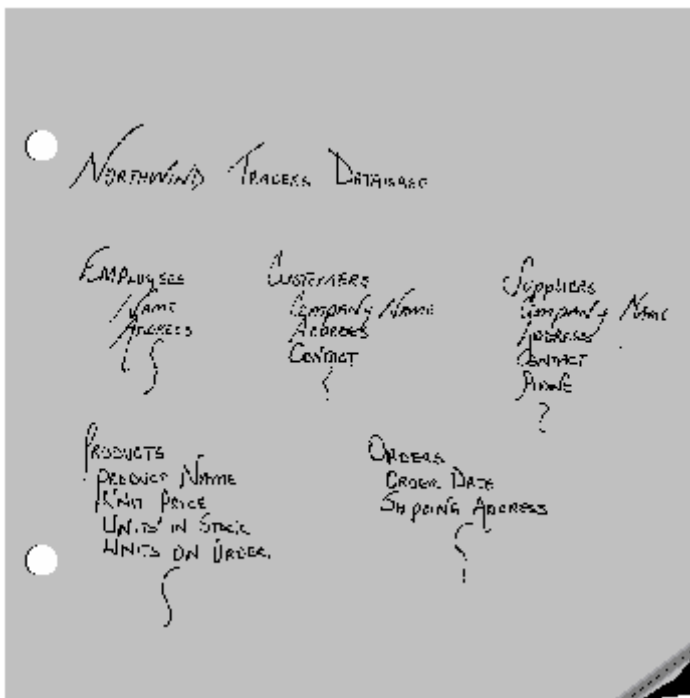
Note: One strategy for dividing information into tables is to look at individual facts and determine what each fact is actually about. For example, on the Northwind Traders order form, the customer address isn't about the sale; it's about the customer. This suggests that you need a separate table for customers. In the Products On Order report, the supplier's phone number isn't about the product in stock; it's about the supplier. This suggests that you need a separate table for suppliers.

Example: Designing Tables in the Northwind Database

The Northwind Traders order form and Products On Order report include information about these subjects:

- Customers
- Suppliers
- Products
- Orders

From this list, you can come up with a rough draft of the tables in the database and some of the fields for each table. It's a good idea as you start to design a database to write down the information you will need to have as you create your tables.



Although the finished Northwind database contains other tables, this list is a good start. Later in this paper, you'll see how to add other tables to refine your design.

Determining the Fields You Need

To determine the fields in a table, decide what you need to know about the people, things, or events recorded in the table. You can think of fields as characteristics of the table. Each record (or row) in the table contains the same set of fields or characteristics. For example, an address field in a customer table contains customers' addresses. Each record in the table contains data about one customer, and the address field contains the address for that customer.

Tips for Determining Fields

Here are a few tips for determining your fields:

- **Relate each field directly to the subject of the table.** A field that describes the subject of a different table belongs in the other table. Later, when you define relationships between your tables, you'll see how you can combine the data from fields in multiple tables. For now, make sure that each field in a table directly describes the subject of the table. If you find yourself repeating the same information in several tables, it's a clue that you have unnecessary fields in some of the tables.
- **Don't include derived or calculated data.** In most cases, you don't want to store the result of calculations in tables. Instead, you can have Microsoft Access perform the calculations when you want to see the result. For example, the Products On Order report shown earlier in this paper displays the subtotal of units on order for each category of product in the Northwind database. However, there's no Units On Order subtotal field in any Northwind table. Instead, the Products table includes a Units On Order field that stores the units on order for each individual product. Using that data, Microsoft Access calculates the subtotal each time you print the report. The subtotal itself doesn't need to be stored in a table.
- **Include all the information you need.** It's easy to overlook important information. Return to the information you gathered in the first step of the design process. Look at your paper forms and reports to make sure all the information you have required in the past is included in your Microsoft Access tables or can be derived from them. Think of the questions you will ask Microsoft Access. Can Microsoft Access find all the answers using the information in your tables?
- **Store information in its smallest logical parts.** You may be tempted to have a single field for full names, or for product names along with product descriptions. If you combine more than one kind of information in a field, it's difficult to retrieve individual facts later. Try to break down information into logical parts; for example, create separate fields for first and last name, or for product name, category, and description.

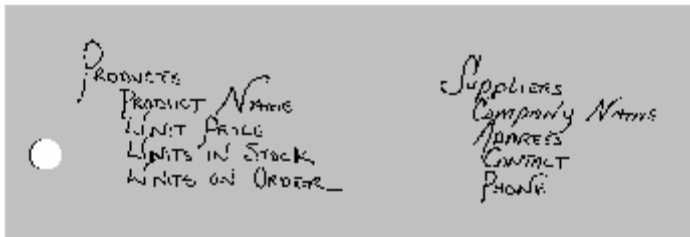
Example: Adding Fields to the Products Table

Northwind Traders sells imported specialty foods from around the world. The employees use a Products On Order report to keep track of products being ordered.

Products On Order					
14-Jun-98					
Category Name	Product Name	In Stock	On Order	Supplier Name	Phone
Beverages 16% of total inventory					
	Chai	39	0	Exotic Liquids	(71) 555-2222
	Chang	17	40	Exotic Liquids	(71) 555-2222
	ChateauBlanc	69	0	Aux joyeux ecclesiastiques	(1) 02-83-00-68
	Côte de Blaye	17	0	Aux joyeux ecclesiastiques	(1) 02-83-00-68
	Guaraná Fantástico	20	0	RefrescosAmericanos (TDA)	(11) 555-4640
	Ipiñi Coffee	17	10	Leka Trading	555-4787
	Lakkajikori	57	0	Karkku Oy	(953) 10966
	Laughing Lumberjack Lager	52	0	Bigfoot Breweries	(503) 555-9831
	Outback Lager	15	10	Pawlovs, Ltd.	(00) 444-2343
	Rhonebrau Klosterbier	125	0	PilsenerLebensmittelgroßmärkte	(069) 992755
	Sauquaticke	11	0	Bigfoot Breweries	(503) 555-9831
	Steekley Stout	20	0	Bigfoot Breweries	(503) 555-9831
Condiments 16% of Total Inventory					
	Aniseed Syrup	13	70	Exotic Liquids	(71) 555-2222
	Chef Anton's Cajun Seasoning	53	0	New Orleans Cajun Delights	(100) 555-4822
	Chef Anton's Gumbo Mix	0	0	New Orleans Cajun Delights	(100) 555-4822
	Genen Shoyuz	39	0	Miyum's	(06) 431-7877

The report indicates that the Products table, which contains facts about products sold, needs to include fields for the product name, units in stock, and units on order, among others. But what about fields for the supplier name and phone number? To produce the report, Microsoft Access needs to know which supplier goes with each product.

One approach would be to include Supplier Name and Supplier Phone fields in the Products table, but this can cause more problems than it solves. Since Northwind Traders might buy many products from the same supplier, the name and phone number of the supplier would have to be repeated in the Products table many times. If the phone number ever changed, it would have to be changed many times as well.



Instead, create a Suppliers table, with separate fields for the supplier's name and phone number. In the next step, you'll add a field to the Products table that identifies the supplier information you need.

Primary Key Fields

The power in a relational database management system such as Microsoft Access comes from its ability to quickly find and bring together information stored in separate tables. In order for Microsoft Access to work most efficiently, each table in your database should include a field or set of fields that uniquely identifies each individual record stored in the table. This is often a unique identification number, such as an employee ID number or a serial number. In database terminology, this information is called the *primary key* of the table. Microsoft Access uses primary key fields to quickly associate data from multiple tables and bring the data together for you.

If you already have a unique identifier for a table, such as a set of product numbers you've developed to identify the items in your stock, you can use that identifier as the table's primary key. But make sure

the values in this field will always be different for each record -- Microsoft Access doesn't allow duplicate values in a primary key field. For example, don't use people's names as a primary key, because names aren't unique. You could easily have two people with the same name in the same table.

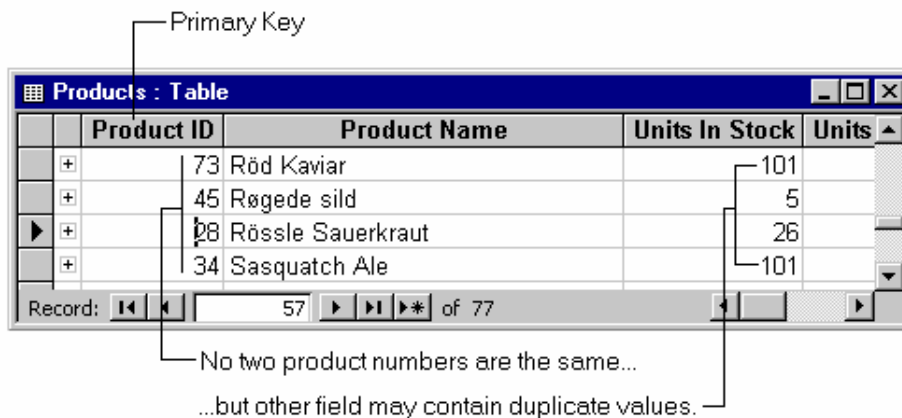
If you don't already have a unique identifier in mind for a table, you can use a field that simply numbers the records consecutively. Microsoft Access can even set up a primary key like that for you. For more information about Primary Keys, search for "Primary Key" in Microsoft Access Help.

When choosing primary key fields, keep these points in mind:

- Microsoft Access doesn't allow duplicate or null values in a primary key field. For this reason, you shouldn't choose a primary key that could contain such values.
- You may use the value in the primary key field to look up records, so it shouldn't be too long to remember or type. You may want it to have a certain number of letters or digits, or be in a certain range.
- The size of the primary key affects the speed of operations in your database. When you create primary key fields, you can set a property to limit the size of the field. For best performance, use the smallest size that will accommodate the values you need to store in the field. For more information, search for "field size" in Microsoft Access Help.

Example: Setting the Primary Key for the Products Table

The primary key of the Northwind Products table contains product ID numbers. Because each product number identifies a different product, you don't want two products with the same number.



In some cases, you may want to use two or more fields that together provide the primary key of a table. For example, the Order Details table in the Northwind database uses two fields as its primary key: Order ID and Product ID. In the next step, you'll see why.

Determining the Relationships

Now that you've divided your information into tables, you need a way to tell Microsoft Access how to bring it back together again in meaningful ways. For example, the following form includes information from several tables.

Information in this form comes from the Customers table...

... the Employees table...

... the Orders table ...

... the Products table ...

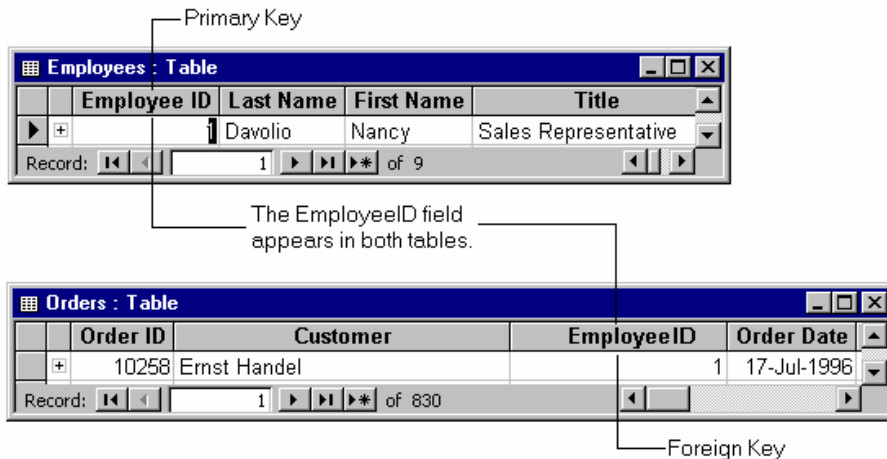
... and the Order Details table.

Product:	Unit Price:	Quantity:	Discount:	Extended Price:
Röd Kaviar	\$15.00	9	0%	\$135.00
Mozzarella di Giovanni	\$34.80	21	0%	\$730.80
Escargots de Bourgogne	\$13.25	20	0%	\$265.00
Manjimup Dried Apples	\$53.00	6	0%	\$318.00

Subtotal:	\$1,823.80
Freight	\$78.85
Total:	\$1,902.65

Microsoft Access is a *relational* database management system. That means you store related data in separate tables. Then you define relationships between the tables, and Microsoft Access uses the relationships to find associated information stored in your database.

For example, suppose that you want to phone an employee with questions about a sale the employee made. Employee phone numbers are recorded in the Employees table; sales are recorded in the Orders table. When you tell Microsoft Access which sale you're interested in, Microsoft Access can look up the phone number based on the relationship between the two tables. It works because Employee ID, the primary key for the Employees table, is also a field in the Orders table. In database terminology, the Employee ID field in the Orders table is called a *foreign key*, because it's a primary key from a different table.



Note If you open the Orders table in the sample Northwind database included with Microsoft Access in Datasheet view, you'll notice that there is an Employees field but no Employee ID field. The Employees field is actually the Employee ID field, but the Caption property is set to "Employee" and properties have been set on the Lookup tab to display the employee names rather than the ID numbers. The illustration above uses the Employee ID field to make the association between the tables clear. Later illustrations in this document also show the ID fields for education purposes rather than the lookup text that appears in the Northwind database. To learn more about how lookup tables work, search for "lookup table" in Microsoft Access Help.

So, to set up a relationship between two tables -- Table A and Table B -- you add one table's primary key to the other table, so that it appears in both tables. But how do you decide which table's primary key to use? To set up the relationship correctly, you must first determine the nature of the relationship. There are three types of relationships between tables:

- One-to-many relationships
- Many-to-many relationships
- One-to-one relationships

The rest of this section presents an example of each type of relationship and explains how to design your tables so that Microsoft Access can associate the data correctly.

Note This section explains how you determine the relationships between your tables and how you decide which fields belong in the tables to support those relationships. You use the Relationships window to create the relationships. For more information, search for "Relationships window" in Microsoft Access Help.

Example: Creating a One-to-Many Relationship

A one-to-many relationship is the most common type of relationship in a relational database. In a one-to-many relationship, a record in Table A can have more than one matching record in Table B, but a record in Table B has *at most* one matching record in Table A.

For example, the Suppliers and Products tables in the Northwind database have a one-to-many relationship.

The image shows two screenshots from Microsoft Access. The top screenshot displays the 'Suppliers : Table' window with the following data:

Supplier ID	Company Name	Contact Name
1	Exotic Liquids	Charlotte Cooper
2	New Orleans Cajun Delights	Shelley Burke
3	Grandma Kelly's Homestead	Regina Murphy
4	Tokyo Traders	Yoshi Nagase
5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra
6	Mayumi's	Mayumi Ohno
7	Pavlova, Ltd.	Ian Devling

The bottom screenshot displays the 'Products : Table' window with the following data:

Product ID	Product Name	SupplierID
1	Chai	1
2	Chang	1
3	Aniseed Syrup	1
4	Chef Anton's Cajun Seasoning	2
5	Chef Anton's Gumbo Mix	2

Annotations in the image explain the relationship: 'One supplier... ..can supply many products.' points to the SupplierID field in the Products table, and 'But each product has only one supplier.' points to the SupplierID field in the Products table.

To set up the relationship, you add the field or fields that make up the primary key on the "one" side of the relationship to the table on the "many" side of the relationship. In this case, you would add the Supplier ID field from the Suppliers table to the Products table, because *one* supplier supplies *many* products. Microsoft Access uses the supplier ID number to locate the correct supplier for each product.

Example: Creating a Many-to-Many Relationship

In a many-to-many relationship, a record in Table A can have more than one matching record in Table B, and a record in Table B can have more than one matching record in Table A. This type of relationship requires changes in your database design before you can correctly specify the relationship to Microsoft Access.

To detect many-to-many relationships between your tables, it's important that you look at *both directions* of the relationship. For example, consider the relationship between orders and products in the Northwind Traders business. One order can include more than one product. So for each record in the Orders table, there can be many records in the Products table. But that's not the whole story. Each product can appear on many orders. So for each record in the Products table, there can be many records in the Orders table.

One order can have many products.

Product:	Unit Price:	Quantity:	Discour
Gudbrandsdalsost	\$36.00	30	
Vegie-spread	\$43.90	9	
Jack's New England Clam Chowder	\$9.65	35	
Gorgonzola Telino	\$12.50	8	

Each product can appear on many orders.

The subjects of the two tables -- orders and products -- have a many-to-many relationship. This presents a problem in database design. To understand the problem, imagine what would happen if you tried to set up the relationship between the two tables by adding the Product ID field to the Orders table. To have more than one product per order, you need more than one record in the Orders table per order. You'd be repeating order information over and over for each record that relates to a single order -- an inefficient design that could lead to inaccurate data. You run into the same problem if you put the Order ID field in the Products table -- you'd have more than one record in the Products table for each product. How do you solve this problem?

The answer is to create a third table that breaks down the many-to-many relationship into two one-to-many relationships. You put the primary key from each of the two tables into the third table.

Primary key from the Orders table

Primary key from the Products table

Order ID	Product ID	Unit Price	Quantity	Discount
10248	17	\$14.00	12	0%
10248	25	\$9.80	10	0%
10248	40	\$34.80	5	0%
10249	59	\$18.60	9	0%
10249	64	\$42.40	40	0%
10250	31	\$7.70	10	0%
10250	39	\$42.40	35	15%
10250	71	\$16.80	15	15%
10251	18	\$16.80	6	5%

Record: 1 of 2155

Information that relates to both the order and the product

Each record in the Order Details table represents one line item on an order. The Order Details table's primary key consists of two fields -- the foreign keys from the Orders and Products tables. The Order ID field alone doesn't work as the primary key for this table, because one order can have many line items. The Order ID is repeated for each line item on an order, so the field doesn't contain unique values. The Product ID field alone doesn't work either, because one product can appear on many different orders. But *together* the two fields always produce a unique value for each record.

In the Northwind database, the Orders table and the Products table aren't related to each other directly. Instead, they are related indirectly through the Order Details table. The many-to-many relationship between orders and products is represented in the database using two one-to-many relationships:

- The Orders and Order Detail tables have a one-to-many relationship. Each order can have more than one line item, but each line item is connected to only one order.
- The Products and Order Detail tables have a one-to-many relationship. Each product can have many line items associated with it, but each line item refers to only one product.

Example: Creating a One-to-One Relationship

In a one-to-one relationship, a record in Table A can have no more than one matching record in Table B, and a record in Table B can have no more than one matching record in Table A. This type of relationship is unusual and may call for some changes in your database design.

One-to-one relationships between tables are unusual because in many cases, the information in the two tables can simply be combined into one table. For example, suppose you created a Ping-Pong Players table to track information about a Northwind Traders ping-pong fundraising event. Because the ping-pong players are all employees of Northwind Traders, this table has a one-to-one relationship with the Employees table in the Northwind database.

Employees : Table					
	Employee ID	Last Name	First Name	Title	Title Of Courtesy
▶ +	1	Davolio	Nancy	Sales Representative	Ms.
+	2	Fuller	Andrew	Vice President, Sales	Dr.
+	3	Leverling	Janet	Sales Representative	Ms.
+	4	Peacock	Margaret	Sales Representative	Mrs.
+	5	Buchanan	Steven	Sales Manager	Mr.
+	6	Suyama	Michael	Sales Representative	Mr.
+	7	King	Robert	Sales Representative	Mr.

Each ping-pong player has one matching record in the Employees table.

Ping-Pong Players : Table				
	EmployeeID	Player NicknaI	Preferred Date	Skill Level
▶	4	Slammin' Nan	7/7/98	2
	3	Ace	7/9/98	1
	5	Stevemiester	7/7/98	2
	7	King John	7/7/98	1

This set of data is a subset of the EmployeeID field in the Employees table.

You could add all the fields from the Ping-Pong Players table to the Employees table. But the Ping-Pong Players table tracks a one-time event, and you won't need the information after the event is over. Additionally, not all employees play ping-pong, so if these fields were in the Employees table, they would be empty for many records. For these reasons, it makes sense to create a separate table.

When you detect the need for a one-to-one relationship in your database, consider whether you can put the information together in one table. If you don't want to do that for some reason, here's how to set up the relationship:

- If the two tables have the same subject, you can probably set up the relationship by using the same primary key field in both tables.
- If the two tables have different subjects with different primary keys, choose one of the tables (either one) and put its primary key field in the other table as a foreign key.

Refining the Design

Once you have the tables, fields, and relationships you need, it's time to study the design and detect any flaws that might remain.

Create your tables, specify relationships between the tables, and enter a few records of data in each table. See if you can use the database to get the answers you want. Create rough drafts of your forms and reports and see if they show the data you expect. Look for unnecessary duplications of data and eliminate them.

As you try out your initial database, you will probably discover room for improvement. Here are a few things to check for:

- Did you forget any fields? Is there information that you need that isn't included? If so, does it belong in the existing tables? If it's information about something else, you may need to create another table.
- Did you choose a good primary key for each table? If you use it to search for specific records, is it easy to remember and type? Make sure that you won't need to enter a value in a primary key field that duplicates another value in the field.
- Are you repeatedly entering duplicate information in one of your tables? If so, you probably need to divide the table into two tables with a one-to-many relationship.
- Do you have tables with many fields, a limited number of records, and many empty fields in individual records? If so, think about redesigning the table so it has fewer fields and more records.

Example: Refining the Products Table

Each product in the Northwind Traders stock falls under a general category, such as Beverages, Condiments, or Seafood. The Products table could include a field that shows the category of each product.

Products : Table				
	Product ID	Product Name	SupplierID	Category
▶ +	1	Chai	Exotic Liquids	Beverages
+	2	Chang	Exotic Liquids	Beverages
+	3	Aniseed Syrup	Exotic Liquids	Condiments
+	4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	Condiments
+	5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights	Condiments
+	6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments
+	7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce
+	8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments

Each product has a Category _____

Suppose that in examining and refining the database, Northwind Traders decides to store a description of the category along with its name. If you add a Category Description field to the Products table, you have to repeat each category description for each product that falls under the category -- not a good solution.

A better solution is to make Categories a new subject for the database to track, with its own table and its own primary key. Then you can add the primary key from the Categories table to the Products table as a foreign key.

Categories : Table			
Category ID	Category Name	Description	
1	Beverages	Soft drinks, coffees, teas, beers, and ales	
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings	
3	Confections	Desserts, candies, and sweet breads	
4	Dairy Products	Cheeses	
5	Grains/Cereals	Breads, crackers, pasta, and cereal	
6	Meat/Poultry	Prepared meats	
7	Produce	Dried fruit and bean curd	
8	Seafood	Seaweed and fish	

Primary Key

Foreign Key

Products : Table				
Product ID	Product Name	SupplierID	Category	
1	Chai	Exotic Liquids	1	1
2	Chang	Exotic Liquids	1	1
3	Aniseed Syrup	Exotic Liquids	2	2
4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	2	2
5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights	2	2
6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	2	2
7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	7	7
8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	2	2

The Categories and Products tables have a one-to-many relationship: one category can have more than one product in it, but any individual product can belong to only one category.

Additional Reading and Practice

If you want to do additional reading on database design, you may want to look at some of the following books.

Date, C. J. *An Introduction to Database Systems*. 5th ed. Vol. 1. Reading: Addison-Wesley Publishing Co., 1990.

Kroenke, David, and Kathleen Dolan. *Database Processing: Fundamentals, Design, and Implementation*. Chicago: Science Research Associates, 1988.

Pascal, Fabian. *SQL and Relational Basics*. Redwood City: M&T Books, 1990.

Viescas, John L. *Running Microsoft Access*. Redmond: Microsoft Press, 1993.