

Getting the Most Out Of MS Access® Part 1: Understanding Relational Databases

In the world of business, data can be worth its weight in gold, but only if it's accurate and if it's stored in a way that allows meaningful reports to be generated from it. That's why good database design and maintenance is so important. A Relational Database Manager (RDBMS) like Microsoft Access does more than just store your company's valuable data. It can also be a powerful tool for maintaining the integrity of that data and transforming it into useful information.

Electronic databases store data in *tables*. Each row of the table is a single *record*. The columns of the table are called *fields*. Some traditional databases store all data in a single table. Each row contains all the necessary information relevant to that record. This type of table is sometimes referred to as a *flat file*. There are several problems with this approach, however. One problem is duplication of data. Consider the sample table in figure 1 below.

OrderID	Customer Name	Customer Address	Customer Phone	Item	Quantity
5202	William Williams	12345 Main Street	555-1212	Basket Ball	2
5202	William Williams	12345 Main Street	555-1212	Baseball Bat	1
5203	John Johnson	2525 Elm Ave	555-2121	Golf Balls	6
5204	William Williams	12345 Main St.	555-2212	Baseball Glove	1

Repeated Data

Inconsistent Data

Figure 1 Traditional Database Table

As you can see, William Williams has placed at least two orders. Each order consists of one or more items. For each item William has ordered, all data pertaining to him is entered into the table. Also, if his order consists of several items, the order information must be repeated as well. Storing the same data over and over wastes valuable space.

Another problem with the single table approach is the possibility of inconsistent data. In figure 1 you can see that in records 1 and 2, William's phone number is different from the number stored in record 4. This could be because his phone number has changed since the first order or it could be the result of a data entry error. Such errors are much more likely to occur when data must be retyped over and over again. If William's number *has* changed, then in order to maintain consistency, we would need to update all instances of the old number. This is yet another problem of traditional single-table databases.

A *relational database* stores information in a series of tables that can be joined together based on the relationships that have been defined between them. Each table represents a category. For example, one table could store information related to customers. Another

would have information related to Orders. Yet another would contain information on products. Information regarding a customer or a product only needs to be stored in one place. If something needs to be changed, such as the customer's phone number, it only needs to be changed once. Relational databases save space by eliminating the need for duplicate data and improve data accuracy by reducing data entry errors. They also allow data to be combined in a variety of ways so that additional information can be obtained.

Figure 2 shows the same information as Figure 1, however, now we are using four tables instead of one. The individual tables are tied together through a system of Primary and Foreign Keys. A *Primary Key* is a unique identifier for a record. Determining which field to use as the Primary Key is important. In the Customer table, the name field might at first seem to be a good candidate but is possible that more than one customer could have the same name. Social Security numbers are a better choice, but what if your customer does not live in the United States? For this reason it is often necessary to establish a special field to be used for this purpose. Access allows you to create a field that will automatically generate a unique number for each record. In our example, CustomerID is such a field and is the Primary Key for the Customers table.

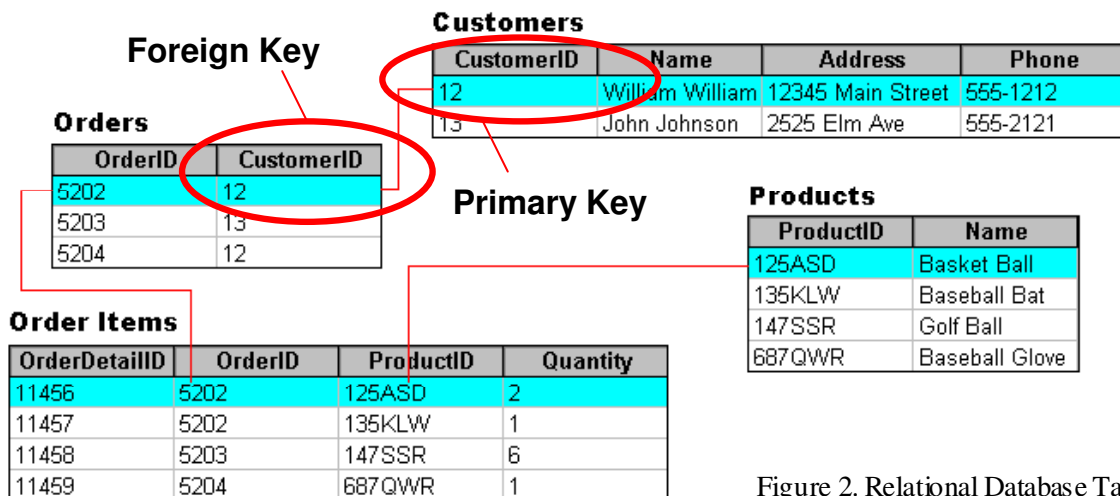


Figure 2. Relational Database Tables

A *Foreign Key* is a field in one table that contains information from the Primary Key in another table. Therefore, CustomerID would be a Foreign Key in the Orders Table. It is important that the fields' data types are the same in both tables.

Tables can be related to one another in one of three ways. The most common type of relationship is a *one-to-many* relationship. If a record in one table has many matching records in another table but each record in the second table has only one matching record

in the first table, the relationship is one-to-many. For example, a customer can have many orders but each order belongs to only one customer.

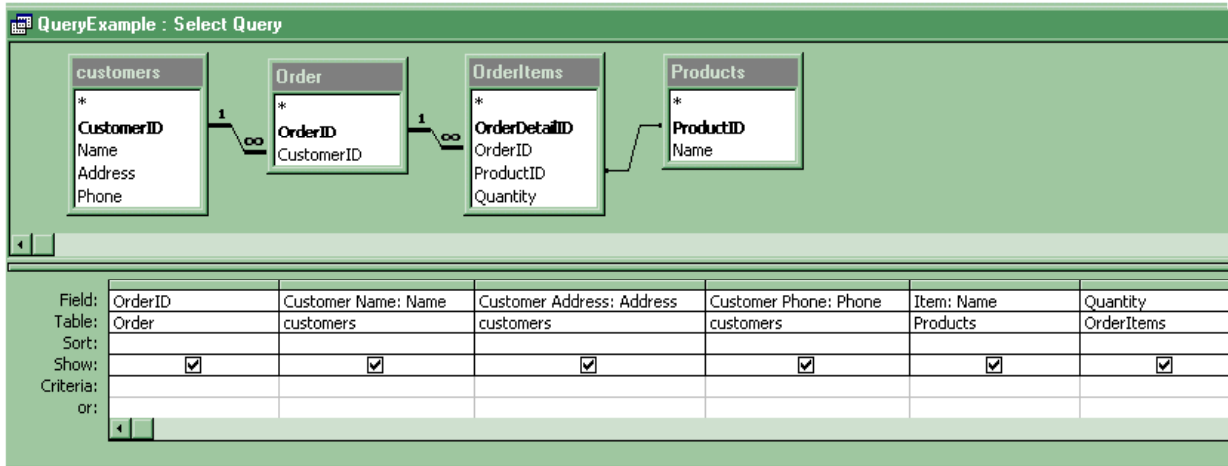
Another type of relationship is a *many-to-many* relationship. A record in one table can have many matching records in a second table and a record in the second table can have many matching records in the first table. In order to implement a many-to-many relationship between two tables it is necessary to have a third table to create a junction between them. In the database shown in figure 2, there is a many-to-many relationship between the Orders table and the Products table. Each order can consist of many products and each product could exist in many orders. The OrderItems table creates a link that allows this relationship to exist. It, in turn, has a one-to-many relationship with each of the other two tables, i.e., each OrderItem can belong to only one order and consist of only one product.

The third type of relationship is the least common. A one-to-one relationship exists when a record in the first table corresponds to only one record in the second table and vice versa. In most cases where this situation exists, it is best to simply combine the two tables into one. However, there might be instances where the data in the second table is relevant to only a subset of the first table. In this case, creating a second table and establishing a one-to-one relationship will save space.

In each of these relationships, you might decide that it is acceptable that a record in one table might have no matching records in a second table. For example, a product might be entered into the products table but if no one has purchased it, it will not have an entry in the OrderItems table. In other instances, you might decide that a record in one table must have a match in the second table. You would not want to have records in the OrderItems table that did not have a match in the Orders table as such records would be “orphaned” because there would be no way of telling what customer they belonged to.

Referential integrity is a tool used to prevent the ties between two tables from being broken. Access, like other RDBMS's, allows you to enforce referential integrity if so desired. It does this by preventing values other than null (which is the absence of a value) from being entered in the foreign key field of the related table if that value does not exist in the primary key of the parent table. It also prevents a record from being deleted from a parent table if matching records exist in a child table unless the “Cascade Delete” option has been selected. In this case, matching records in the child table are deleted along with the parent record. Finally, A Primary Key value in the parent table cannot be changed if that record has related records unless the “Cascade Update” option has been selected.

Once the relationships between tables have been established, it is possible to create *queries* that join the tables together to provide information. In Figure 3 you can see that by using a query, Access can easily join the separate tables together to create a temporary table identical to the table in Figure 1, only without the errors. Data can be combined in other ways to provide useful information such as average orders per customer, quantity sold per product, total sales per month and much, much more.



OrderID	Customer Name	Customer Address	Customer Phone	Item	Quantity
5202	William Williams	12345 Main Street	555-1212	Basket Ball	2
5202	William Williams	12345 Main Street	555-1212	Baseball Bat	1
5203	John Johnson	2525 Elm Ave	555-2121	Golf Ball	6
5204	William Williams	12345 Main Street	555-1212	Baseball Glove	1

Figure 3 Query Design

Colleen Emerick, Application Developer, immedia