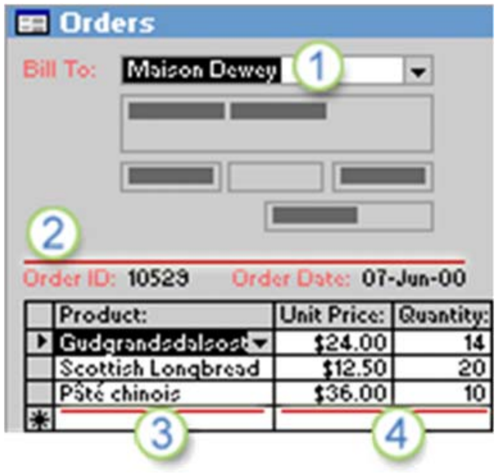
Relationship Types in Access

Introduction

After you have created a table for each subject in your database, you must provide Office Access with the means by which to bring that information back together again when needed. You do this by placing common fields in tables that are related, and by defining relationships between your tables. You can then create queries, forms, and reports that display information from several tables at once. For example, the form shown here includes information drawn from several tables:



Bill To: Maison Dewey

Order ID: 10529 Order Date: 07-Jun-00

Product:	Unit Price:	Quantity:
Gudgrandsdalsost	$24.00	14
Scottish Longbread	$12.50	20
Pâté chinois	$36.00	10

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

2. ...the Orders table...

3. ...the Products table...

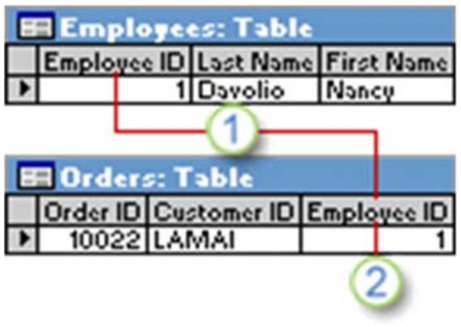
4. ...and the Order Details table.

The customer name in the **Bill To** box is retrieved from the Customers table, the Order ID and the Order Date values come from the Orders table, the Product name comes from the Products table, and the Unit Price and Quantity values come from the Order Details table. These tables are linked to each other in a variety of ways to bring information from each into the form.

Primary Key and Foreign Key

In the preceding example, the fields in the tables must be coordinated so that they show information about the same order. This coordination is accomplished by using table relationships. A table relationship works by matching data in key fields — often a field with the same name in both tables. In most cases, these matching fields are the **primary key** from one table, which provides a unique identifier for each record, and a **foreign key** in the other table.

For example, employees can be associated with orders for which they are responsible by creating a table relationship between the Employee ID fields in the Employees and the Orders tables.



Employees: Table

Employee ID	Last Name	First Name
1	Davolio	Nancy

Orders: Table

Order ID	Customer ID	Employee ID
10022	LAMAI	1

1. EmployeeID appears in both tables — as a primary key ...

2. ... and as a foreign key.

Types of Table Relationships

There are three types of table relationships.

- **A one-to-many relationship**

  Consider an order tracking database that includes a Customers table and an Orders table. A customer can place any number of orders. It follows that for any customer represented in the Customers table, there can be many orders represented in the Orders table. The relationship between the Customers table and the Orders table is, therefore, a one-to-many relationship.

  To represent a one-to-many relationship in your database design, take the primary key on the "one" side of the relationship and add it as an additional field or fields to the table on the "many" side of the relationship. In this case, for example, you add a new field — the ID field from the Customers table — to the Orders table and name it Customer ID. Access can then use the Customer ID number in the Orders table to locate the correct customer for each order.

- **A many-to-many relationship**

  Consider the relationship between a Products table and an Orders table. A single order can include more than one product. On the other hand, a single product can appear on many orders. Therefore, for each record in the Orders table, there can be many records in the Products table. In addition, for each record in the Products table, there can be many records in the Orders table. This type of relationship is called a many-to-many relationship because, for any product, there can be many orders and, for any order, there can be many products. Note that to detect existing many-to-many relationships between your tables, it is important that you consider both sides of the relationship.

To represent a many-to-many relationship, you must create a third table, often called a junction table, that breaks down the many-to-many relationship into two one-to-many relationships. You insert the primary key from each of the two tables into the third table. As a result, the third table records each occurrence, or instance, of the relationship. For example, the Orders table and the Products table have a many-to-many relationship that is defined by creating two one-to-many relationships to the Order Details table. One order can have many products, and each product can appear on many orders.

- **A one-to-one relationship**

In a one-to-one relationship, each record in the first table can have only one matching record in the second table, and each record in the second table can have only one matching record in the first table. This type of relationship is not common because, most often, the information related in this way is stored in the same table. You might use a one-to-one relationship to divide a table with many fields, to isolate part of a table for security reasons, or to store information that applies only to a subset of the main table. When you do identify such a relationship, both tables must share a common field.

Understanding Referential Integrity

When you design a database, you divide your information into many subject-based tables to minimize data redundancy. You then provide Office Access with the means by which to bring the data back together by placing common fields into related tables. For example, to represent a one-to-many relationship you take the primary key from the "one" table and add it as an additional field to the "many" table. To bring the data back together, Access takes the value in the "many" table and looks up the corresponding value in the "one" table. In this way the values in the "many" table reference the corresponding values in the "one" table.

Suppose you have a one-to-many relationship between Shippers and Orders and you want to delete a Shipper. If the shipper you want to delete has orders in the Orders table, those orders will become "orphans" when you delete the Shipper record. The orders will still contain a shipper ID, but the ID will no longer be valid, because the record that it references no longer exists.

The purpose of referential integrity is to prevent orphans and keep references in sync so that this hypothetical situation never occurs.

You enforce referential integrity by enabling it for a table relationship. Once enforced, Access rejects any operation that would violate referential integrity for that table relationship. This means Access will reject both updates that change the target of a reference, and deletions that remove the target of a reference. However, it is possible you might have a perfectly valid need to change the primary key for a shipper that has orders in the Orders table. For such cases, what you really need is for Access to automatically update all the effected rows as part of a single operation. That way, Access ensures that the update is completed in full so that your database is not left in an inconsistent state, with some rows updated and some not. For this reason Access supports the Cascade Update Related Fields option. When you enforce referential integrity and choose the Cascade Update Related Fields option, and you then update a primary key, Access automatically updates all fields that reference the primary key.

It's also possible you might have a valid need to delete a row and all related records — for example, a Shipper record and all related orders for that shipper. For this reason, Access supports the Cascade Delete Related Records option. When you enforce referential integrity and choose the Cascade Delete Related Records option, and you then delete a record on the primary key side of the relationship, Access automatically deletes all records that reference the primary key.