

### Summary

- Introduction to Relationships
- Why Define Relationships?
- Relationships in ER Diagram vs. Relationships in MS Access
- Creating Relationships Between Tables
  - One-to-Many Relationship (1:N)
  - Many-to-Many Relationship (M:N)
- Creating a Field/Column that looks up or lists values in tables
  - Creating a field that looks up data from another table
  - Creating a field that lists values from another table

### Introduction: What is a Relationship?

*Definition in class:*

An association between 2 (or more) separate entities.

*Definition in MS Access:*

An association between 2 common fields (column) in two tables.

We know that relationships could either be:

- **One-to-One (1:1)**
- **One-to-Many (1:N)**
- **Many-to-Many (M:N)**.

### Why Define Relationships?

After you've set up different tables for each subject in your Microsoft Access Database, you need a way of telling Microsoft Access how to bring that information back together again. The first step in this process is to define relationships between your tables. After you've done that, you can create queries, forms, and reports to display information from several tables at once. For example, this form includes information from five tables:

The screenshot shows an MS Access form titled "Orders". The form is divided into several sections. At the top, there are fields for "Bill To" and "Ship To", both containing "Maison Dewey". Below these are address fields: "Rue Joseph-Bens 532" and "Rue Joseph-Bens 53", and "Bruxelles" and "Bruxelles". There are also fields for "Salesperson" (Buchanan, Steven) and "Ship Via" (Speedy, United). At the bottom, there are fields for "Order ID" (10529), "Order Date" (07-Jun-95), and "Required Date" (05-Jul-95). Below these fields is a table with columns: Product, Unit Price, Quantity, and Discount. The table contains three rows: "Gudbrandsdalsost" (Unit Price: \$24.00, Quantity: 14, Discount: 0.00%), "Scottish Longbreads" (Unit Price: \$12.50, Quantity: 20, Discount: 0.00%), and "Pâté chinois" (Unit Price: \$36.00, Quantity: 10, Discount: 0.00%).

Annotations with lines pointing to the form:

- The Employees table (points to Salesperson)
- The Customers table (points to Bill To)
- The Orders table (points to Order ID)
- The Products table (points to Product)
- The Order Details table (points to the table below)

Fig. 1: A Form Using Information from Five Tables

### How do relationships work?

In the previous example, the fields in five tables must be coordinated so that they show information about the same order. This coordination is accomplished with **relationships** between tables. A relationship works by matching data in key fields — usually 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 they're responsible for by creating a relationship between the “**Employees**” table and the “**Orders**” table using the **EmployeeID** fields (which we will show later).

### Relationships in ER Diagram vs MS Access

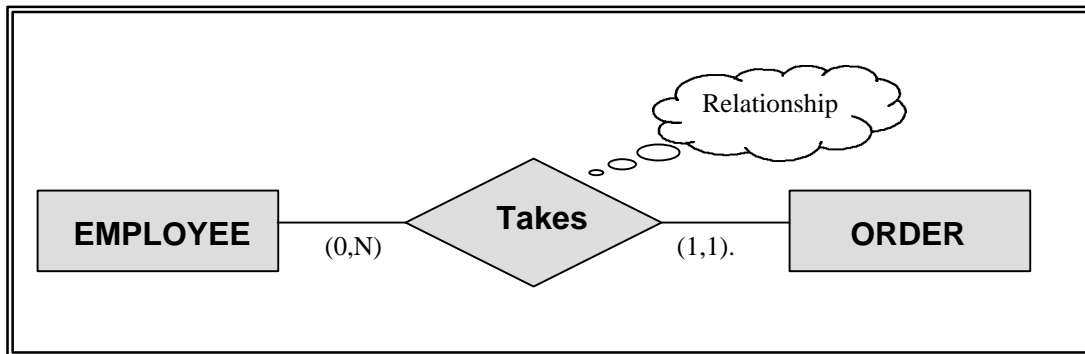
**Relationships in ER Diagram ≠ Relationships in MS Access**

**ER Diagram ≠ MS Access Relationship View**

⚡ Relationships in **MS Access** are links between attributes, but relationships in the **ER diagram** are links between entities.

### Relationships in ER Diagram

Let's go back to the relationship between the “**Employees**” entity and the “**Orders**” entity in Fig. 2:



**Fig. 2: Relationships in ER diagram**

'Takes' is a 1:N (One-to-Many) Relationship. The 'Takes' relationship can be converted into an MS Access table as shown in Fig 3.

## Relationships in MS Access

Different Menu Bar in Relationships

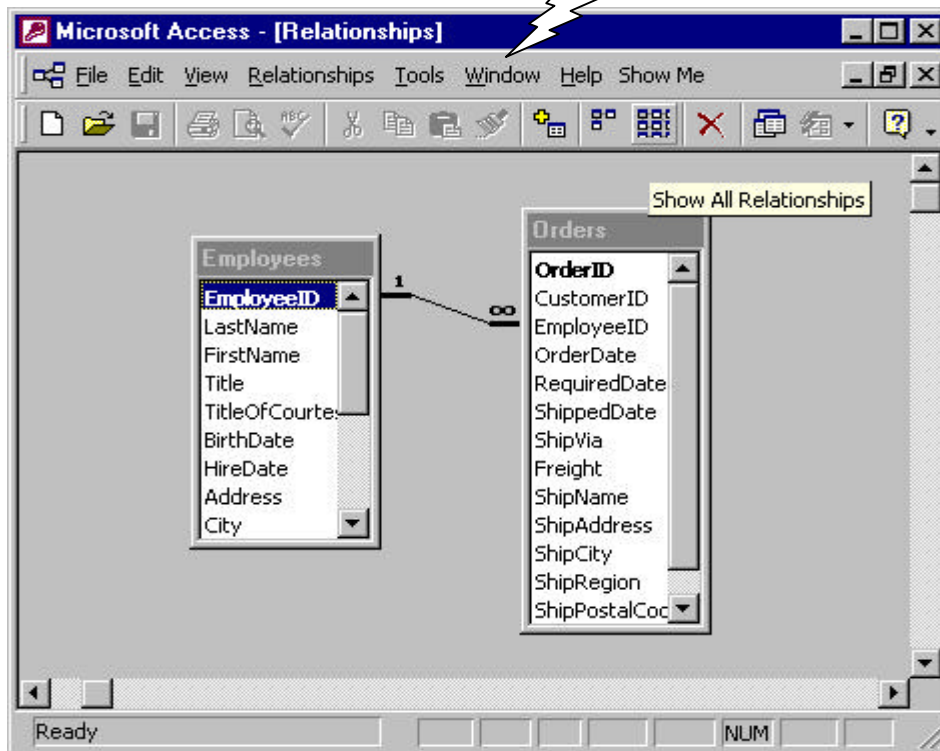


Fig. 3: Relationships in MS Access's "Relationship View"

### Creating Relationships Between Tables

Again, we will be using "Northwind.mdb". Open it from the "C:/Program Files/Microsoft Office/Office/Samples" folder.

- **One-to-One relationship**

In a one-to-one relationship, each record in Table A can have only one matching record in Table B, and each record in Table B can have only one matching record in Table A. This type of relationship is NOT common, because most information related in this way would be in one 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. For example, you might want to create a table to track employees participating in a fundraising soccer game.

- **One-to-Many Relationship**

A one-to-many relationship is the most common type of relationship. In a **one-to-many** relationship, a record in Table A can have many matching records in Table B, but a record in Table B has only one matching record in Table A.

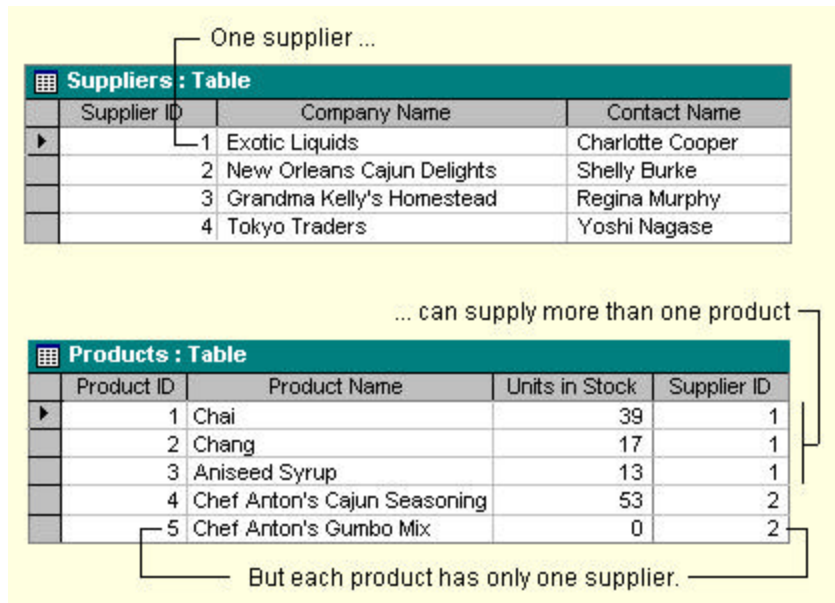



Fig. 4 One-to-Many Relationship

### Defining a One-to-Many Relationships between Tables

1. Close any tables you have open. You can't create or modify relationships between open tables.
2. If you haven't already done so, switch to the **Database Window**. You can press F11 to switch to the Database window from any other window.
3. Click on **Tools → Relationships**  (**Note**: when you do this, the toolbar will change, Fig.3)
4. If your database does not have any relationships defined, the **Show Table** dialog box will automatically be displayed (Fig. 5). Add the tables that you want to relate.

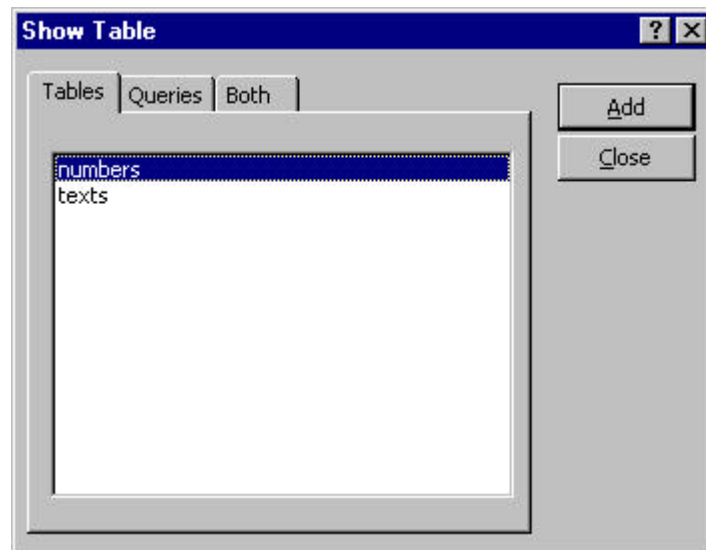

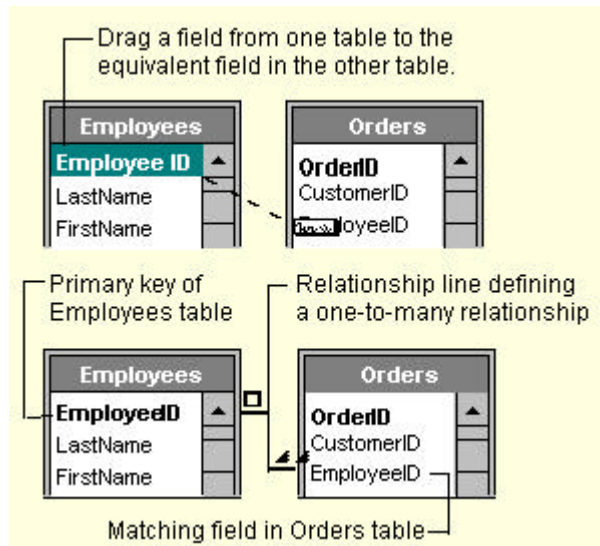


Fig. 6: A Sample "Show Table" Dialog Box

5. If you need to add tables that you want to relate and the **Show Table** dialog box isn't displayed, in the **Relationships View** (which you accessed via the original menu **Tool → Relationships**), click on **Relationships → Show Table** . If the tables you want to relate are already displayed, skip to step 6.
6. Once you have all the tables you want to relate. Define a relationship between 2 tables by **dragging** the field that you want to relate from one table to the related field in the other table (Fig. 7) (To drag **multiple fields**, press the CTRL key and click each field before dragging them.)



**Fig. 7: Creating a relationship between 2 tables**

In most cases, you drag the **primary key**<sup>1</sup> field (which is displayed in bold text) from one table to a similar field (often with the same name) called the **foreign key**<sup>2</sup> in the other table. The related fields are NOT required to have the same names (but it's good practice to do so since it reminds you where the relationship comes from), but they MUST have the same domain (or data type<sup>3</sup>) and contain the same kind of information. In addition, when the matching fields are **Number** fields, they must have the same **FieldSize** property setting. The two exceptions to matching data types:

- you can match an **AutoNumber** field with a **Number** field whose **FieldSize** property is set to **Long Integer**
- you can also match an **AutoNumber** field with a **Number** field if both fields have their **FieldSize** property set to **ReplicationID**.

7. Once you have created the relationships, the **Edit Relationships** dialog box is displayed (Fig. 8). Check the field names displayed in the two columns to ensure they are correct. You can change them if necessary.

<sup>1</sup> primary key:

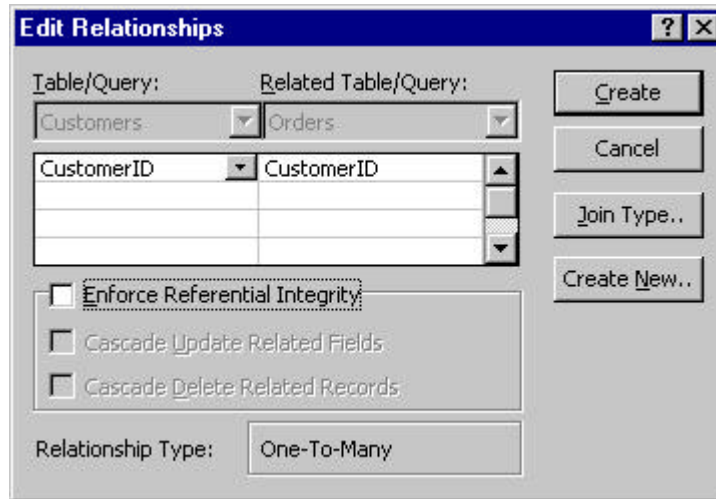
One or more fields whose value or values uniquely identify each record in a table. In a relationship, a primary key is used to refer to specific records in one table from another table.

<sup>2</sup> foreign key:

One or more table fields that refer to the primary key field or fields in another table. A foreign key indicates how the tables are related. The data in the foreign key and primary key fields must match. For example, the Products table in the Northwind sample database contains the foreign key SupplierID, which refers to the SupplierID primary key of the Suppliers table. Using this relationship, the Products table displays a supplier name from the Suppliers table for each product.

<sup>3</sup> data type:

The attribute of a variable or field that determines what kind of data it can hold. For example, the Text and Memo field data types allow the field to store either text or numbers, but the Number data type will allow only numbers to be stored in the field. Number data type fields store numerical data that will be used in mathematical calculations. Use the Currency data type to display or calculate currency values. Supported data types include field data types, Visual Basic data types, and query parameter data types.



**Fig. 8: “Edit Relationships” Dialog Box**



Set the relationship options if necessary. For information about a specific item in the **Relationships** dialog box, click on the question mark button (the cursor would now have a floating question mark next to it) and then click on the item.

8. Click the **Create** button to create the relationship.
9. Repeat steps 5 through 8 for each pair of tables you want to relate.

You can edit the relationship between the 2 tables later on, by just double-clicking on the **Relationship Line** (shown in Fig. 7) connecting the 2 tables in **Relationships View**.

When you close the Relationships window, Microsoft Access asks if you want to save the layout. Whether you save the layout or not, the relationships you create are saved in the database.

### **Notes**

- If you need to view all the relationships defined in the database, click **Show All Relationships**  on the toolbar. To view only the relationships defined for a particular table, click the table, and then click **Show Direct Relationships**  on the toolbar.
- If you need to make a change to the design of a table, you can right-click the table you want to change, and then click **Design Table**.
- You can create relationships using queries as well as tables. However, referential integrity<sup>4</sup> isn't enforced with queries.
- To create a relationship between a table and itself, add that table twice. This is useful in situations where you need to perform a lookup within the same table. For example, in the **Employees** table in the Northwind sample database, a relationship has been defined between the **EmployeeID** and **ReportsTo** fields, so that the **ReportsTo** field can display employee data from a matching **EmployeeID**.

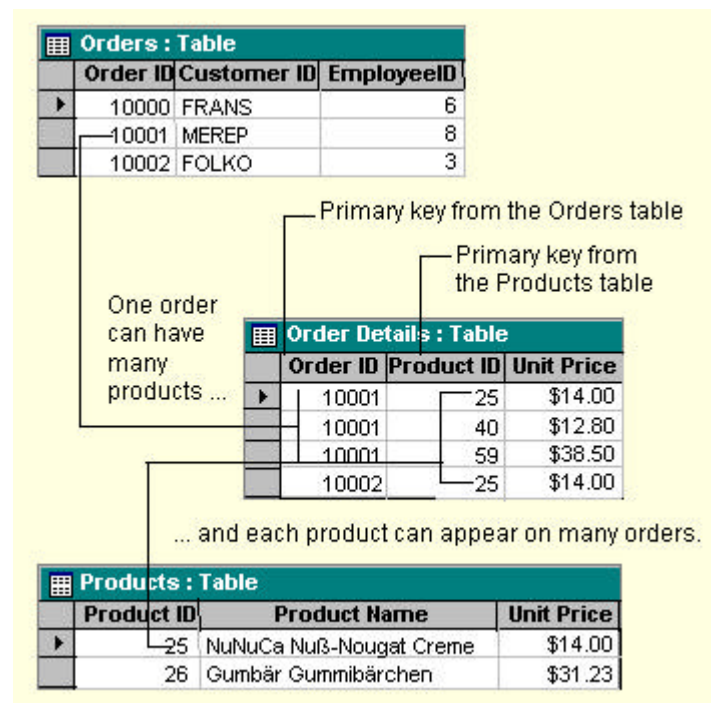
### • **Many-to-Many Relationship**

In a **many-to-many** relationship, a record in Table A can have many matching records in Table B, and a record in Table B can have many matching records in Table A. This type of relationship is **only possible** by defining a third table (called a junction table) whose primary key consists of two fields: the **primary**

<sup>4</sup> referential integrity:

Rules that you follow to preserve the defined relationships between tables when you enter or delete records. If you enforce referential integrity, Microsoft Access prevents you from adding records to a related table when there is no associated record in the primary table, changing values in the primary table that would result in orphan records in a related table, and deleting records from the primary table when there are matching related records in a related table.

**keys from both Tables A and B.** A many-to-many relationship is really 2 one-to-many relationships with a third table. For example, the **Orders** table and the **Products** table have a **many-to-many** relationship that's defined by creating 2 **one-to-many** relationships to the **Order Details** table.



**Fig. 9: Many-to-Many Relationship**

## Defining a Many-to-Many Relationship between Tables

1. Create the two tables that will have a **many-to-many** relationship.
2. Create a third table, called a **junction table**, and add fields with the same definitions as the primary key fields from each of the other two tables to this table. In the junction table, the **primary key** fields function as **foreign keys**. You can add other fields to the junction table, just as you can to any other table.
3. In the **junction table**, set the **primary key** to include the **primary key** fields from the other two tables. For example, in an **Order Details** junction table, the primary key would be made up of the **OrderID** and **ProductID** fields.  
(Note: You can set multiple fields as the **primary key** by highlighting multiple rows (which correspond to fields) which you want to be part of your **primary key** in **Design View**, and then click on **Edit** → **Primary Key**. Alternatively you can also use the **primary key** button in the tool bar)
4. Define a **one-to-many** relationship between each of the two **primary tables** and the **junction table**.
5. To add data to the tables, create a **form**<sup>5</sup> that works with more than one table.

### **Note:**

In the Northwind sample database, a **many-to-many** relationship exists between the **Orders** and **Products** tables. One order in the **Orders** table can include multiple products from the **Products** table. In addition, a single product can appear in many orders. In the sample database, the **Order Details** table is a **junction table** between the **Orders** table and the **Products** table.

<sup>5</sup> form

A Microsoft Access database object on which you place controls for taking actions or for entering, displaying, and editing data in fields.

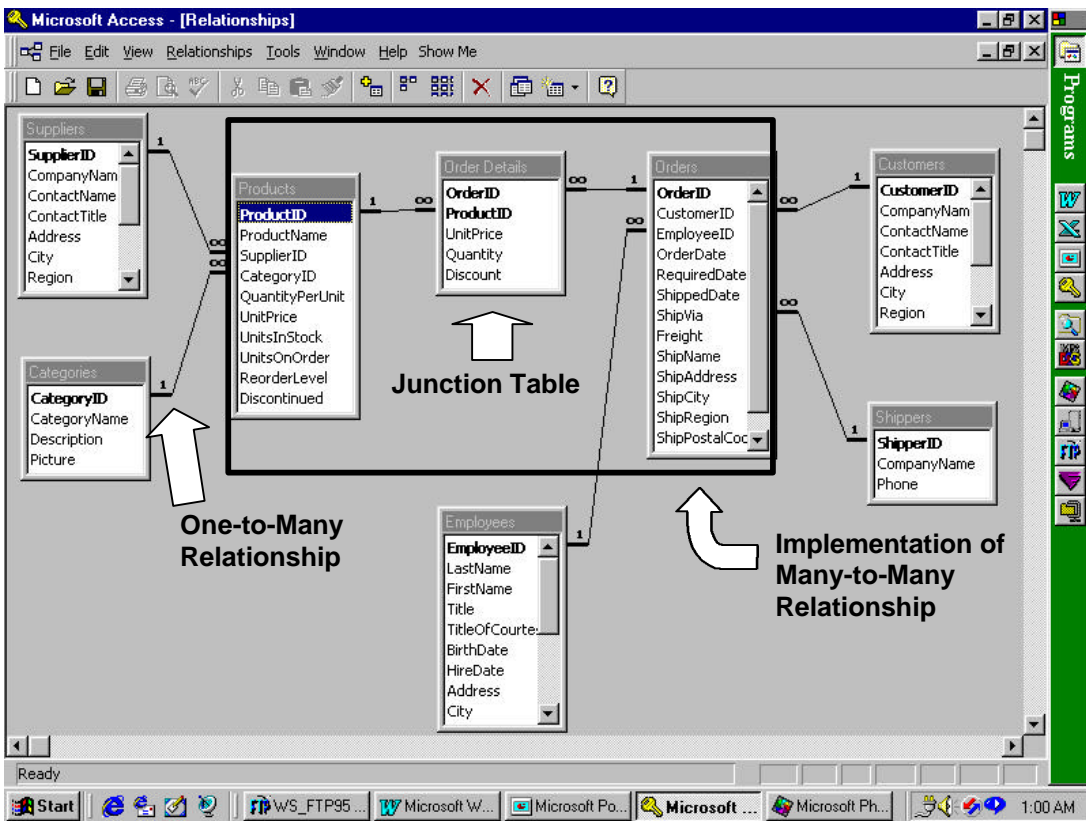


Fig. 10: Junction Table in a Many-to-Many Relationship in Northwind.mdb

### Creating a field that looks up or lists values in tables

Using the **Lookup Wizard**, you can create a field that displays either of two kinds of lists to make data entry simpler:

- **Lookup list** that displays values looked up from an existing table or query
- **Value list** that displays a fixed set of values that you enter when you create the field

### Lookup List

The most common **Lookup list** displays values looked up from a related table. For example, the **SupplierID** field in the **Products** table of the **Northwind** sample database displays this **Lookup list**:

Products: Table			
Product ID	Supplier		Category
1	Exotic Liquids	▼	Beverages
2	Exotic Liquids	▲	Beverages
3	Grandma Kelly's Homestead		Condiments
4	New Orleans Cajun Delights		Condiments
5	Tokyo Traders	▼	Condiments

—Lookup list

Fig. 11: Look up List

This list is created by looking up the **SupplierID** values in the **Suppliers** table and by displaying the corresponding **Supplier** names. Picking a value from a **Lookup list** sets the **foreign key** value in the current record (**SupplierID** in the **Products** table) to the **primary key** value of the corresponding record




in the original table (**SupplierID** in the **Suppliers** table). This creates an association to the related table to *display* (but **NOT** store) the **Supplier** names in the record. The foreign key (**SupplierID**) is stored but is not displayed. For this reason, any updates made to the data in the **Suppliers** table will be reflected in both the list and records in the **Products** table. You must define a **Lookup list** field from the table that will contain the foreign key and display the **Lookup list**. In this example, the **Lookup list** field would be defined from the **Products** table.

## Value List

A **Value list** looks the same as a **Lookup list**, but consists of a fixed set of values you type in when you create it. A value list should only be used for values that will not change very often and don't need to be stored in a table. For example, a list for a **Salutation** field containing Mr., Mrs., or Ms. would be a good candidate for a value list. Choosing a value from a value list will store that value in the record. It doesn't create an association to a related table. For this reason, if you change any of the original values in the value list later, they will not be reflected in records added before this change was made.

You can add a **new** Lookup or value list field in either table **Design view** or table **Datasheet view**. However, if the field you want to use as the foreign key for a Lookup field already exists, you must open that field's table in **Design view** to define the Lookup field. For example, if you have a **Products** table that has a **SupplierID** field already defined, and you want to change it to a Lookup field to display supplier names from your **Suppliers** table, you must open the **Products** table in **Design view** to change **SupplierID** to a Lookup field.

## Creating a field that looks up data from another table in Design view

1. Open the table in **Design view**.
2. Do one of the following:
  - To insert a new field within the table, click in the row below where you want to add the field, and then click **Insert** → **Rows**  on the toolbar, or to add a new field at the end of the table, click in the first blank row. Type the name for the field in the Field Name column, following Microsoft Access object-naming rules.
  - If the field you want to use as the foreign key for the Lookup field already exists, click in that field's row. For example, if you have a **Products** table that has a **SupplierID** field already defined, and you want to change it to a Lookup field to display supplier names from your **Suppliers** table, click in the **SupplierID** field's row.
3. In the '**Data Type**' column, click the down arrow and select '**Lookup Wizard**' from the list.
4. Click the option that indicates you want the Lookup field to look up the values in a table or query.
5. Click **Next** and follow the directions in the remaining Lookup Wizard dialog boxes.

When you click the **Finish** button, Microsoft Access creates the Lookup field and sets certain field properties based on the choices you made in the wizard. Once you've created a Lookup list field, if you add the field to a form, Microsoft Access copies its definition into the form<sup>5</sup>. You won't have to create the combo or list box and its Lookup or value list definition for the form. However, if you change the definition of a Lookup or value list field in the table after adding it to a form, those changes will NOT be reflected in that form. To correct this, delete the field from the form and then add it again. For information on the properties that the **Lookup Wizard** sets, go back to the field that was set by the **Lookup Wizard**, and click on the **Lookup** tab in the **Field Properties** window (Fig. 12).

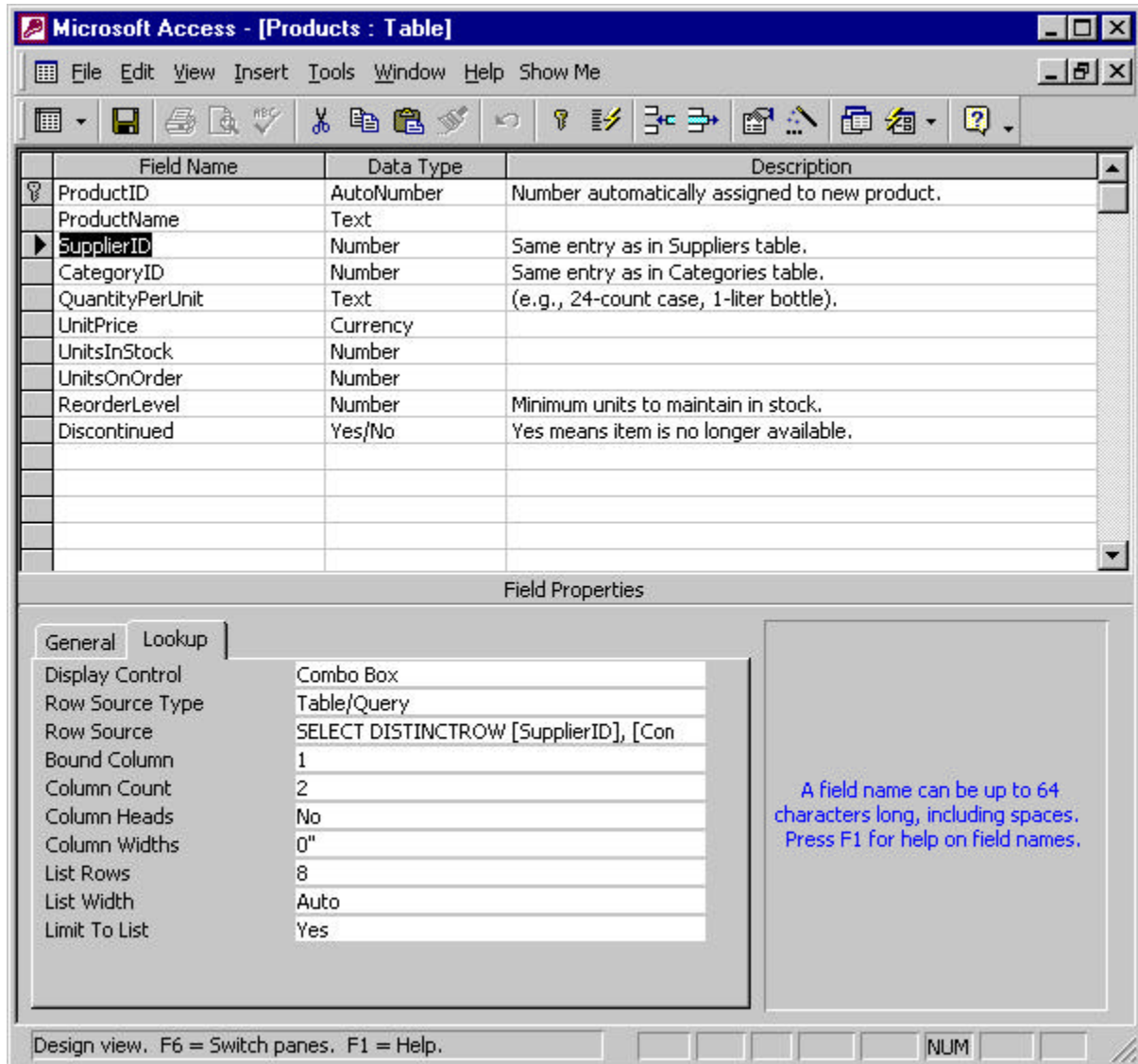


Fig. 12: Properties Listed in the Lookup Tab in the Field Properties Window

**Note:**

It is also possible to add a Lookup field to a table that displays values from the same table that contains the Lookup field. For example, in the **Employees** table of the Northwind sample database the **ReportsTo** field is a Lookup field that displays data from the **FirstName** and **LastName** fields by looking up the corresponding **EmployeeID** in the same table.

**Create a value list field in Design view**

1. Open the table in **Design view**.
2. To insert the field within the table, click in the row below where you want to add the field, and then click **Insert → Rows** on the toolbar.
3. To add the field to the end of the table, click in the first blank row.
4. In the **Field Name** column, type the name for the field, following Microsoft Access object-naming rules.
5. In the **Data Type** column, click the arrow and select **Lookup Wizard**.
6. In the first **Lookup Wizard** dialog box, click the option that indicates you will type in the values that you want.
7. Click **Next** and follow the directions in the remaining Lookup Wizard dialog boxes.

**Note:**

When you use the **Lookup Wizard** to create a fixed value list, Microsoft Access sets certain field properties based on the choices you made in the wizard. Once you've created the field, if you add it to a form, Microsoft Access copies its definition into the form. You won't have to create the combo or list box and its value list definition for the form. However, if you change the definition of the value list field in the table after adding it to a form, those changes will not be reflected in that form. To correct this, delete the field from the form and then add it again.

---

WELL DONE !! YOU HAVE JUST LEARNED HOW TO CREATE DIFFERENT RELATIONSHIPS  
BETWEEN TABLES, AND ALSO TO ENHANCE YOUR TABLES USING LOOKUP & VALUE LISTS

~~~~~