

Access 2003 - Intermediate Guide

Relationships

Using Related Tables

Tables can be joined, or related, in order to access and coordinate information in all the fields of the related tables. Joining tables is a useful way to avoid the need to enter duplicate information in various, related tables. In addition, it allows you to create reports, forms, and queries from the related data tables and save them in the database file. Relating tables allows you to create smaller, more efficient tables that can be related when you need access to the data.

When you relate tables, the table from which you select the first join field is the primary table and the table to which you drag the join field is the related table. The tables must have some common fields that contain the same type of data. One of the fields in the primary table must be the primary key so that Access does not allow duplicate entries. The common fields in both tables must have the same or equivalent data types and; if they are **Number** fields, they must have the same field size.

For example, you can create a table consisting of customer names and addresses. You can also include a unique identification number for each customer, which would be the primary key in the table. You can create this number or allow Access to create it for you.

ID #	Names	Address	City	State	Zip
1	Jones	6352 Oak Tree Drive	Edmonds	WA	98026
2	White	563 S. Monroe Blvd.	Fort Wayne	IN	46825
3	Smith	21458 Clark Road	Valencia	CA	91355
4	Lee	1622 16 th Street West	Washington	DC	20520
5	Fulbright	875 Walker Court	Golden Valley	MN	55422

You could then create a separate table consisting only of orders placed by customers.

This table would also contain the field for the unique customer identification number, but not the customers' names and addresses.



ID #	Order
1	PC Label Printer
2	12- Sheet Crosscut Shredder
3	Vertical Wood File Cabinet
4	Black Leather Executive Chair
5	Fire-Safe Security File Cabinet

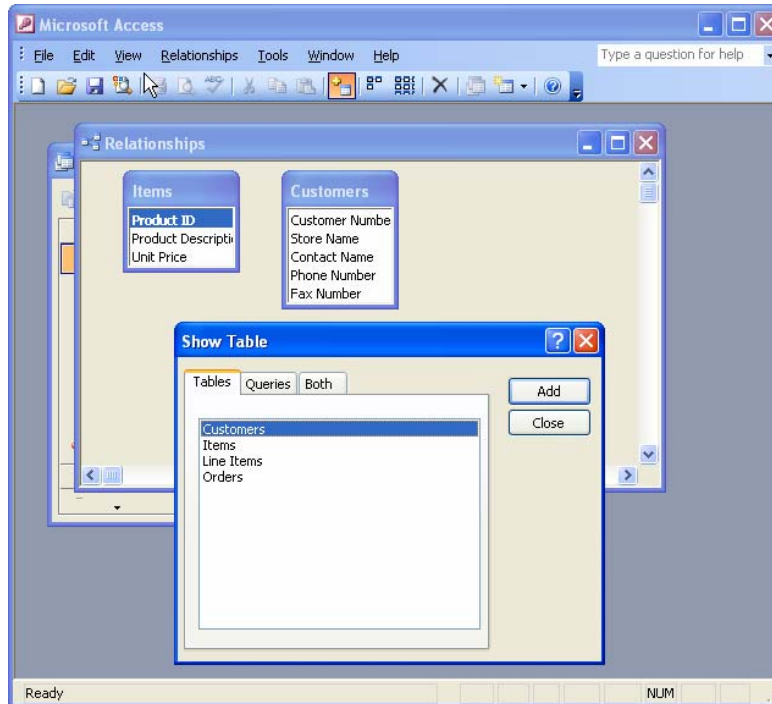
By relating the two tables through the common customer identification number field, the customers' names, addresses, and telephone numbers do not have to be entered for every order.

Access includes two basic types of relationships: one-to-many and one-to-one. A one-to-many relationship occurs when one record from the primary table matches many records from the related table; for example, one customer record matches many order records. A one-to-one relationship occurs when one record from the primary table matches one record from the related table. Access determines the relationship type automatically when you create the relationship.

Creating a Relationship

You create relationships between tables or queries in the Relationships window. The Relationships window displays a graphic representation of the relationships in the database.

To establish a relationship, click the **Relationships** button  on the Database toolbar. To add tables that need to be related, click the **Show Table**  button on the **Relationships** toolbar. The **Show Table** dialog box opens. You may then choose from the list of available tables or queries.



When you are working in the Relationships window, you can reposition the field lists so that you can view the relationships more easily. In addition, the field name representing the primary key appears in bold in the tables. All tables must be closed before you can create relationships.

Notes:

The Show Table Dialog box opens automatically if no tables have been added to the Relationships window.

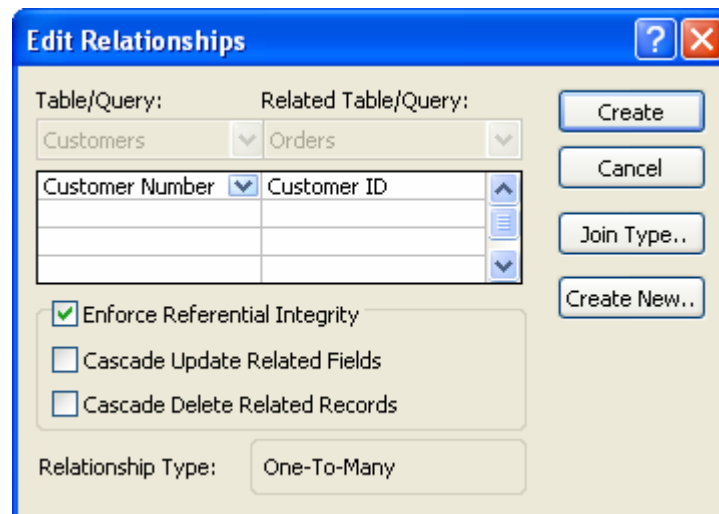
*You can also open the Relationships window by selecting the **Relationships** option from the **Tools** drop-down menu.*

*You can select multiple tables in the Show Table dialog box by holding the [Ctrl] key and clicking on each table. Clicking the **Add** button adds all the selected tables to the Relationships window.*

Setting Referential Integrity

When you create a relationship between two tables, you can set referential integrity. Referential integrity is a built-in set of rules Access uses to make sure that the relationship is valid. Referential integrity can also prevent accidental

deletion or editing of data. In order to use referential integrity, the following conditions must be true: the related field in the primary table is the primary key, the related fields in both tables have the same data type, and both tables belong to the same database.

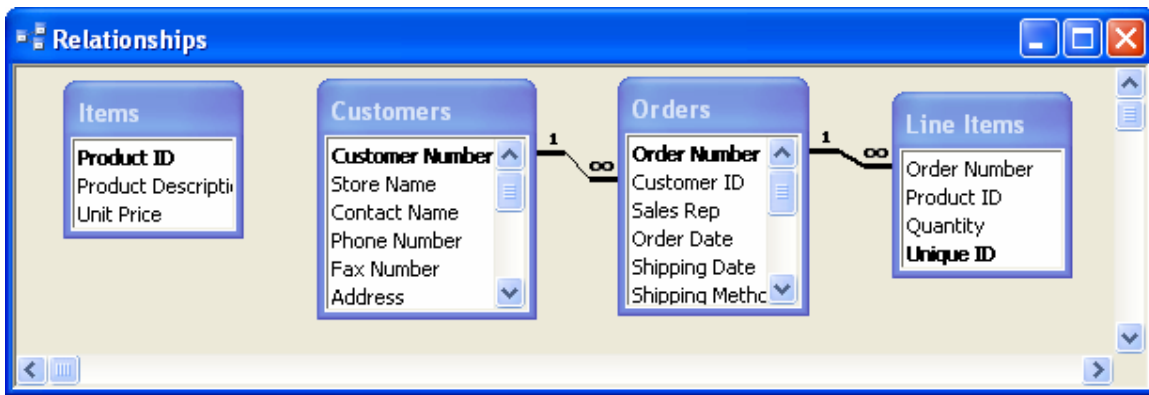


When you set referential integrity, you must observe the following three rules:

1. You cannot enter data in the join field in the related table that does not have a match in the join field in the primary table.
2. You cannot delete records from the primary table if there are matching records in the related table.
3. You cannot edit primary key values in the primary table if related records exist.

However, if you want to perform the changes listed above and still maintain referential integrity, you can select the **Cascade Update Related Fields** and **Cascade Delete Related Records** options in the Edit Relationships dialog box. When either or both of these options are selected, Access makes the necessary changes to the related tables automatically to maintain referential integrity. It is recommended that these two options be used after careful consideration since the changes cannot be undone.

When the referential integrity option is activated, Access displays symbols above the join line in the Relationships window to indicate the type of relationship: one-to-one or one-to-many. The number 1 above the join line next to a field list indicates "one", while the mathematical symbol for infinity (which resembles a horizontal 8) indicates "many".



Notes:

Double-clicking the middle segment of any join line opens the *Edit Relationships* window with the selected join displayed. Double-clicking the beginning or ending segment of any join line opens the *Edit Relationships* window, but with no join selected. You can then use the *Table* or *Query* list to select the desired join.

You can also open the *Edit Relationships* dialog box by right-clicking the middle segment of the join line and selecting the **Edit Relationship...** command.

Deleting a Join Line

Deleting a join line removes the relationship between two tables. You must want to delete a join line if you no longer need to relate the tables or you want to create a different relationship.

You must select a join line before you can delete it:

- Click the middle segment of a join line to select it. The middle line is bolded. Clicking the beginning or ending segment does not select the join line.
- Press the **Delete** key on your keyboard.
- Click **Yes** to permanently remove the selected relationship from the database.

Note:

You can also delete a join line by right-clicking its middle segment and selecting the **Delete** option.

Using Operators in Queries

Using Comparison Operators

You can enter criteria in the **Criteria** row of the query design grid in order to select specific records. The simplest criteria requires that records match a single value to be included in the RecordSet.

You can also use comparison operators to select a specific group of records in a table. For example, if you want to find all customers with credit limits less than \$1000, or all customers with a contract date on or before January 2001, you can write an expression that defines the criteria using a combination of comparison operators and field values, such as <1000 or <=1/1/01. Comparison operators are symbols that represent conditions recognized by Access. The available comparison operators are:

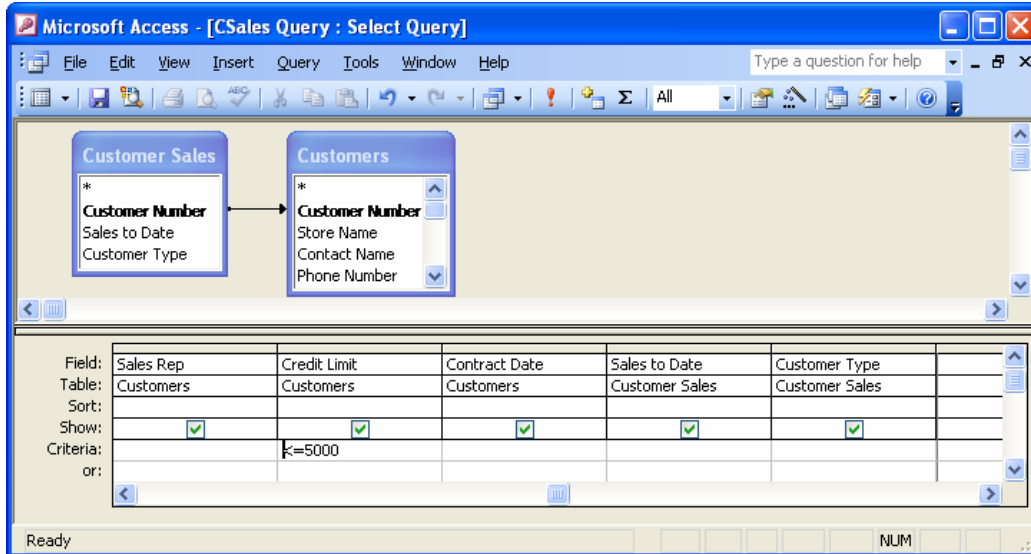
<i>Operator</i>	<i>Description</i>
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
=	equal to
<>	not equal to
Not	reverse logic

You can use a comparison operator to compare a specified value with all the values in a field. For example, you may want to find all customers with credit limits of less than \$1000 or all customers with a contract date on or before January 2001. You can use a combination of comparison operators and field values to write an expression defining the desired criteria (e.g., <1000 or <=1/1/01, respectively.)

When you run the query, only the records with values meeting the criteria you specify appear in the RecordSet.

Note:

Access automatically inserts number symbols (#) around date values and quotation marks (“ ”) around alphanumeric values. Access does not insert any symbols or characters around numeric values.



Using an AND Condition

Many times, a query requires more than one condition to obtain the desired result. For example, if you want to find all customers in PA with sales to date over \$10,000, you would need two conditions: State=PA and Sales to Date>10000. The records must meet both conditions in order to be included in the RecordSet. To combine two criteria in this way, you use the **And** logical operator.

You can use the **And** operator in a single field or in different fields. In a single field, you can use the **And** operator to find records that fall into a range. For example, to find customers whose contract dates fall between 9/1/99 and 9/30/99, you type both criteria on a single line in the **Criteria** row under the appropriate field (i.e., >=9/1/99 **And** <=9/30/99 in the **Contract Date** field).

The **And** operator also allows you to impose conditions in two or more different fields. For example, to find customers in PA with sales to date over \$10,000, you type each criterion on a single line in the **Criteria** row under the appropriate fields (i.e., =PA in the **State/Province** field and >10000 in the **Sales to Date** field).

Notes:

*Criteria entered in the same Criteria row of the design grid create an **And** condition.*

*Criteria entered in different Criteria rows create an **Or** condition.*

Using an OR Condition

Many times, a query requires more than one condition to obtain the desired result. For example, if you want to find all customers in PA or all customers with sales to date over \$10,000, you would need two conditions: State=PA as well as Sales to Date>10000. The records only need to meet one of the conditions in order to be included in the RecordSet. To combine two criteria in this way, you use the **Or** logical operator.

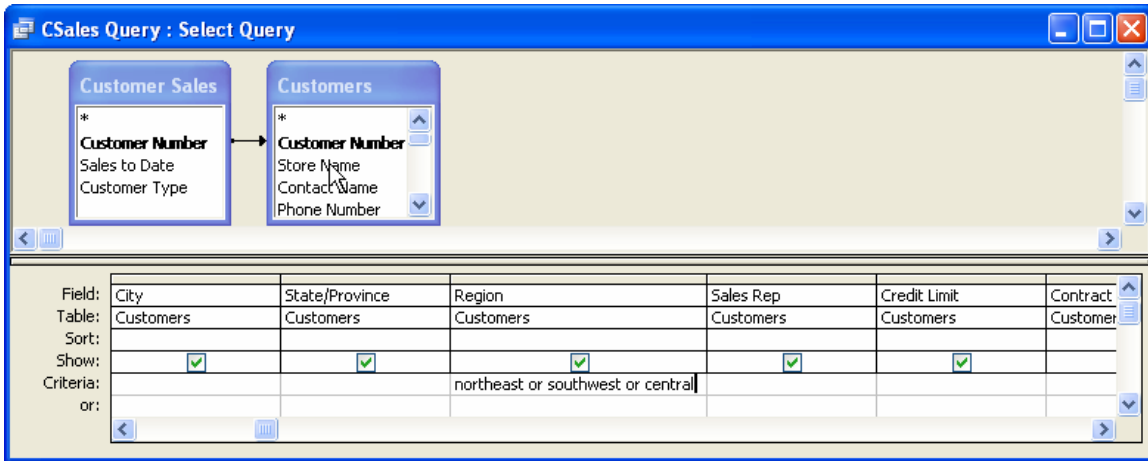
You can use the **Or** operator in a single field or in different fields. In a single field, you type the criteria on two separate lines under the same field. In different fields, you type the criteria on two separate lines under the appropriate fields. For example, to find all customers with contract dates on or before January 1, 2001 or credit limits above \$3,000, you type <=1/1/01 in the **Criteria** row under the **Contract Date** field and >3000 in the **or** row under the **Credit Limit** field.

You can create additional Or and And operators by typing criteria into the **Criteria** row, the **or** row, or any row below the **or** row.

- Criteria entered into the same **Criteria** row across the design grid create an And condition.
- Criteria entered into different **Criteria** rows create an Or condition.
- You can also create a combination of And and Or conditions.

Note:

*You can create an Or condition for a single field in the **Criteria** row using the **Or** operator. Typing **northeast or southeast or central** in the **Criteria** row of the **Region** field finds all records in any of the three regions.*



Using the BETWEEN AND Operator

You can use the **Between And** operator in a query to find data that is between two values. You can use this operator with a text, numeric, or date field. For example, to find all records of customers with credit limits between 1000 and 2000, you would enter **Between 1000 And 2000** in the **Criteria** row under the **Credit Limit** field.

The **Between And** operator is inclusive. All records with values that include or fall between the stated criteria are included in the RecordSet.

Space must be included between the criteria and the words **Between** and **And**. If you do not include the proper spacing, the **Data type mismatch in criteria expression** error message will appear.

Using the Expression Builder

When you enter criteria, you are actually creating an expression. Expressions are calculations and can include database objects, operators, and values. Objects range from table fields to controls in a form. Operators are standard mathematical operators used in calculations, such as +, -, *, /, (), < >. Values can be numbers, dates, text, and built-in functions, as well as fields, control, and property identifiers.

You can create an expression by typing the expression elements, or you can use the Expression Builder. The Expression Builder is a tool that provides all the elements needed to build the expression.

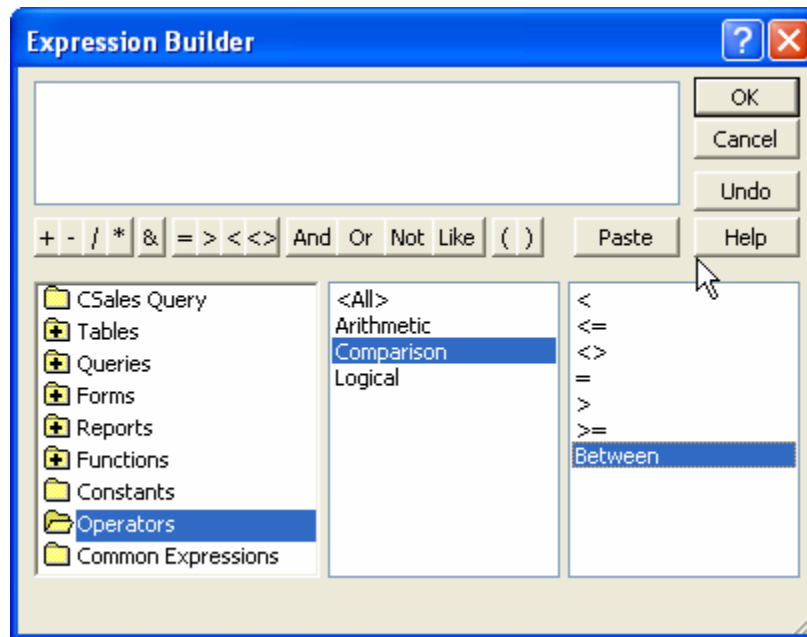
The Expression Builder displays the **Expression** box in its top pane, a row of operator buttons below the **Expression** box, and three lower panes that display categories, subcategories, and values, respectively.

New expressions appear in the **Expression** box. You can use a combination of methods to build a new expression. You can type some elements of the expression and select others, such as operators, functions, and values, from the element panes. If you make a mistake, the **Undo** button allows you to undo previous actions, one at a time. Additionally, you can select and delete any elements you want to remove from the **Expression** box.

Open up the Expression Builder right-clicking the Criteria row and selecting the Build option. The Expression Builder window, as seen below, will be displayed.

Notes: *The Expression Builder is available in **Design** view for any database object for which you need to create an expression. For example, you can use the Expression Builder to create a validation rule for a table field.*

*You can insert an element from an element pane into the **Expression** box by double-clicking the element or by selecting it and clicking the **Paste** button in the Expression Builder.*



Using a Wildcard Character

Wildcard characters are used in a query to find records when the criteria contains a pattern (such as all last names beginning with M) or is only partly known (such as the proper spelling - Kline or Klein). Wildcards take the place of one or several letters in a Text field or numbers in a Date/Time field.

The two most common wildcards are listed in the following table:

Wildcard	Representing	Example
?	Any single letter or number	Sm?th finds Smith and Smyth, whereas ?andy finds Sandy, Randy, etc.
*	One or more letters or numbers	M* finds all records that start with M. 8*/00 finds all dates in August, 2000. *ball* finds all records that have the word ball anywhere in the field.

Notes: Wildcards are not case-sensitive. For example, **ill* finds *Bill* and *bill*.

When you use wildcard characters (*?* and ***) Access automatically inserts the word **Like** before the criteria, and quotation marks (“ ”) around text.

Designing Advanced Queries

Setting Top Values in a Query

You can limit the results of a query so that only the highest or lowest values for a field appear in a RecordSet. For example, you can set the top values of a **Quantity Sold** field to 10 to find the top ten best selling products.

You can limit the number of records to a specific number or a percentage of all records being queried (i.e. top 25%). The field for which you are setting the top or bottom values must be sorted. If the field is sorted in descending order (Z to A, 9 to 0), the top values will be found. If the field is sorted in ascending order (A to Z, 0 to 9), the bottom values will be found.

Notes:

If other fields in the query are sorted, they must appear to the right of the field for which you are finding top or bottom values in the design grid.

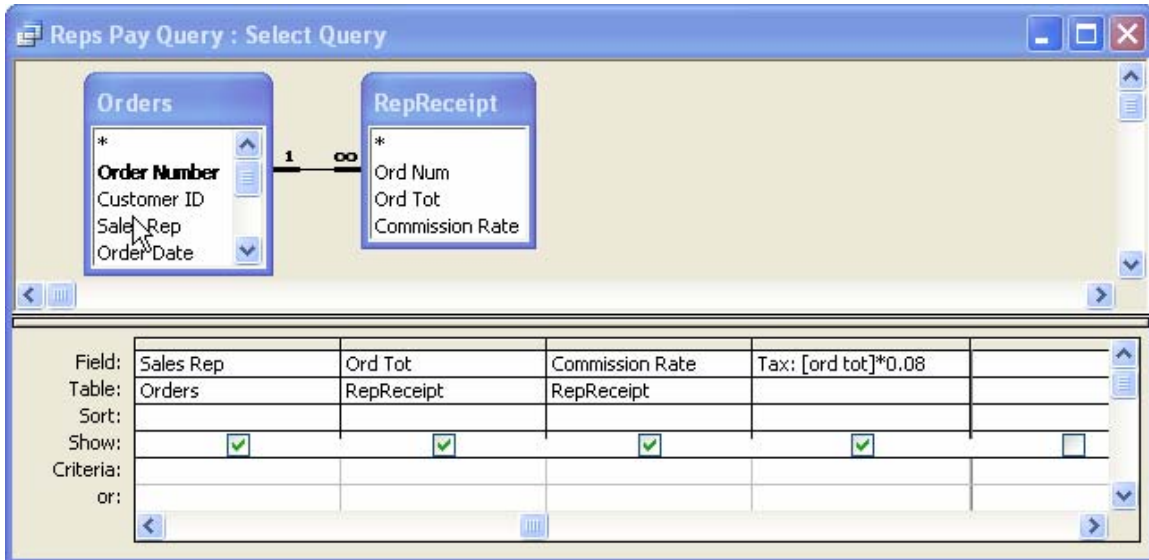
You can also type a value, e.g. 7, into the Top Value box on the Query Design toolbar.

Creating a Calculated Field

Access allows you to create expressions that calculate new field values. For example, you can create an expression that multiplies the value in the **Quantity** field by the value in the **Price** field to calculate total sales. You can also create an expression that adjusts a value in a single field, such as doubling a **Wholesale Price** field to calculate values for a **Retail Price** field.


In expressions, field names are enclosed in square brackets (**[]**); numbers are not. For example, to calculate 8% of order total amount and display the results in a column named **Tax**, enter **Tax:[ord tot]*.08** in the design grid. The colon separates the column from the expression.

Calculated fields are created in queries. You can also use criteria to remove nonessential records, thereby allowing the query to run faster. The results of your query can then be used to generate a report.



Notes:

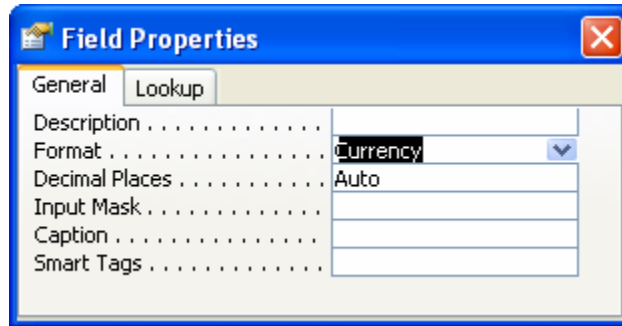
The field names used in an expression must be the same as the field names that appear in the table.

*You can also use the Expression Builder to create a calculated field by selecting any blank **Field** row and clicking the **Build** button  on the **Query Design** toolbar.*

*Another way of using the Expression Builder is by right-clicking any blank **Field** row and selecting the **Build** option from the drop-down menu.*

Formatting a Calculated Field

Once you have created a calculated field, you can change its properties as you would any other field on the design grid. The **Format** property determines how data appear in **Datasheet** view. For example, you can change the properties of a calculated field so that the field values display as currency.



You can also change the format of a calculated field by right-clicking anywhere in it and selecting the **Properties** command.

Creating a Function Query

Access allows you to create a query that groups records by a selected field and applies a function that calculates values on other fields in the query according to your needs. For example, you can group records in a table by state and then select the **Count** function to find out how many customers (records) are in each state (field). You can also group by customer name (field) and calculate the **Sum** of each customer's orders (record values.)

There are several types of functions from which you can choose. The most commonly used functions are listed in the following table:

<i>Function</i>	<i>Description</i>
Sum	Sums the values in the calculated field
Average	Finds the average value of the calculated field
Count	Counts the number of records in the calculated field
Max	Finds the highest value in the calculated field
Min	Finds the lowest value in the calculated field

Notes:

You can perform more than one calculation on a specific field. In such a case, you must add the field to the design grid a second time and create the desired expression.

You can add a **Total** row by right-clicking anywhere in the design grid and selecting the **Totals** option.

Creating a Parameter Query

If you want to run a query with different criteria each time, you can create a parameter query. A parameter query is a query that prompts the user for information when the query is run. Access then uses the information as the criteria and runs the query. The resulting RecordSet only includes those records that meet the criteria. This option allows you to avoid displaying the query in **Design** view each time you want to change the criteria.

You enter the text that will display in the prompt in the **Criteria** row under the appropriate field in the design grid, followed by a colon (:) and enclosed in square brackets ([]). You can set up a parameter query to prompt the user for more than one piece of information as well.

Note:

You can add multiple parameters to a query. When you run the query, a prompt will appear for each parameter in it.

Creating a Concatenation in a Query

Access allows you to combine two or more fields into one field. This process, known as concatenation, enables you to add field name text strings to one another. The text strings can follow each other with or without spaces. You can add other characters between the text strings if needed. For example, you can combine the individual **City**, **State**, and **Postal Code** fields into one field called **Address**. You can have the comma and space characters appear between the field text strings. This concatenation process can be performed by creating a query to combine two or more fields.


When typing expressions for concatenation, the first part of the expression defines the name of the new field. The second part of the expression defines the fields which are to be concatenated. These field names must be surrounded by brackets. The ampersand (&) appears between the field name brackets. Any additional characters that are to appear between the fields are surrounded by double quotes.

For example, the expression **Names: [Last Name]&", "&[First Name]** concatenates the **Last Name** and **First Name** fields and inserts the concatenated

text string into a field called **Names**. The new field displays the last name, a comma, a space, and the first name for each record in the table.

Notes:


You may use concatenation to create text strings in forms and reports.

*Expression Builder can be used to concatenate text. Select any blank **Field** row and click the **Build** button  on the **Query Design** toolbar.*

Filtering a Query

You can apply filters to queries in the same way you apply filters to a table or form. Since the data you want to filter sometimes appears in two or more tables, you might need to create a multiple table query. Once the query is created, you can apply a filter to temporarily isolate the records that you want to view.

The **Filter By Selection** feature allows you to quickly and easily filter a query to display only those records in which the selected value appears. Conversely, the **Filter Excluding Selection** feature filters out the selected value, leaving only those records that do not contain the selected value.

To filter a query in **Datasheet** view, click in the field containing the text you would like to filter. Then, click the **Filter By Selection** button  on the **Query Datasheet** toolbar.

To remove the filter, click the **Remove Filter** button  on the **Query Datasheet** toolbar.

Note:

*To filter a query, you can also right-click the field you want to filter, and select either the **Filter By Selection** or **Filter Excluding Selection** option from the drop-down list.*

Creating Action Queries

Creating a Make Table Query

A make-table query creates a new table consisting of data from existing tables and queries. Make-table queries have many uses. The new table can be a duplicate of

an existing table and serve as a backup for the data. The backup can then be used to store old records that can then be deleted from the current table.

Make-table queries can create a compact version of an existing table, displaying only the fields you need to see. They can also bring fields from many tables into one table, making reports and queries on the single table run faster.

You can limit both records and fields in a new table. If you use criteria in a make-table query, only those records that meet the criteria will be added to the new table.

Note: *The fields in the new table retain the data type and field size properties assigned to them in the source table.*

Creating an Update Query

You can use an update query to change the values of data in an existing table. Update queries save time by updating a large number of records in a table at once. For example, you can use an update query to increase the values in a **Unit Price** field by 10%.

You can also create a new field in the table and use an update query to provide the values for the field based on a calculation. For example, you could create a **Sale Price** field in a table and then use an update query to add the values to the field based on 75% of the **Unit Price** field.

If you do not want to update all the records in the table, you can use criteria to select only the records you want to update. For example, you can use an update query with criteria to increase the unit price on only one line of products, instead of on all the products.

In an update query, you do not need to include all the fields from the table in the design grid. You only need to include the fields you want to update and the fields you want to limit with criteria.

Notes:

When you use field names in an expression, they must be enclosed in square brackets ([]).

*An update query permanently alters field values. Before you run the query, make sure that the correct records are selected and that the expression in the **Update To** row of the design grid is correct.*

Creating an Append Query

An append query copies records from a table or query and adds them to the end of another table. Append queries are useful if you want to transfer data from one table to another.

You can also use an append query to archive all or part of a table's contents. For example, you have two tables, **Current Orders** and **Old Orders**. You can use an append query to copy the records from the **Current Orders** table to the **Old Orders** table. This query prevents you from having to manually add the records to the **Old Orders** table.

If the fields in both tables have the same names, the data is added automatically to the table. If the fields have different names, you must specify the fields to which you want to "append" the data. The data is copied to the field(s) indicated in the **Append To** row of the appropriate field(s) in the design grid.

You can also use criteria to select the records you want to append. Only those records that meet the criteria are appended to the table.

Creating a Delete Query

You can use delete queries to maintain the appearance, usefulness, and efficiency of tables in a database. When records have been appended or archived to another table, or they are simply no longer of use to you, it is a good idea to delete them from the table. Deleting records saves disk space and makes tables more efficient. The more data in a table, the more time it takes to save, sort, and query.

Delete queries enable you to apply criteria to select and delete groups of records at one time.

Notes: A *delete* query always deletes entire record(s.) You cannot use a delete query to delete data only in specific fields.

The records deleted by a delete query are not retrievable. Therefore, you should always preview the selected records by first running a select query. Only after you have confirmed that the selected records are correct should you run the delete query.