Visual Database Tools

Visual Database Tools Usage Considerations

The following issues may affect your usage of the Visual Database Tools.

Saving Queries

To save queries designed in Visual Database Tools Query Designer, you must use SQL Query Analyzer.

To save queries designed in Query Designer

- 1. <u>Design the query</u>.
- 2. In the SQL Pane, select the query statements, right-click, and then click **Copy**.
- 3. In SQL Server Enterprise Manager, on the **Tools** menu, click **SQL Query Analyzer**.
- 4. Right-click in the Query window of SQL Query Analyzer, and click **Paste**.
- 5. On the File menu, click **Save As** to save the query.

Note Although Query Designer provides menu options for saving, these options are disabled.

For more information about SQL Query Analyzer, see the <u>Overview of SQL</u> <u>Query Analyzer</u>.

OLE DB Provider

The Microsoft OLE DB Provider for SQL Server is implemented as a dynamic link library (DLL) on the client machine and a set of stored procedures on the server. Unless both parts are installed, the OLE DB provider will not function

correctly.

- Sqloledb.dll is the DLL.
- Running Instcat.sql installs the stored procedures.

If you install a newer version of Sqloledb.dll, but do not install Instcat.sql on each SQL Server machine running OLE DB, some OLE DB functions will return incorrect results.

Visual Database Tools

Visual Database Tools and SQL Server Databases

The Microsoft® Visual Database Tools work transparently with SQL Server databases by producing SQL Server-specific SQL commands, recognizing SQL syntax unique to SQL Server, and so on.

In a few instances, however, you will find it useful to know how the Visual Database Tools differ when you are working with SQL Server databases. For example, you might find it helpful to understand how the Query Designer will treat SQL Server-specific syntax if you enter an SQL command yourself, or what operators you can use.

For more information about	See
Issues and notes for using the	Database Designer Considerations
Database Designer with SQL Server	<u>for SQL Server Databases</u>
databases	
Issues and notes for using the Query	Query Designer Considerations for
Designer with SQL Server databases	SQL Server Databases

Visual Database Tools

Database Designer Considerations for SQL Server Databases

The following guidelines outline SQL Server-specific features that you can use.

Below you will find information about:

- <u>Case Sensitivity</u>
- <u>Full-Text Indexes</u>
- <u>Changing Column Properties</u>

Case Sensitivity

Column and table names in a SQL Server database can be stored in uppercase letters, lowercase letters, or a combination of both. For example, a column name can appear as "LASTNAME," "LastName," or "lastname."

Depending on how SQL Server was installed, databases can be case-sensitive or case-insensitive. If a database is case-sensitive, you must enter owner, table, and column names using the correct combination of uppercase and lowercase characters. If you are using a case-sensitive database, you must think carefully when you refer to a database object by name, since two objects named "CUSTOMER" and "Customer" can exist in the same database.

If the server was installed with a case-insensitive option, you can enter database object names using any combination of uppercase and lowercase characters.

Tip To determine the case sensitivity of a server, execute the stored procedure sp_server_info, and then examine the contents of row 18. If the server has been installed with the case-insensitive setting, the option for sort_order will be set to nocase. You can run a stored procedure from the Query Analyzer.

Full-Text Indexes

A table can include full-text indexes. Database Designer and Table Designer provide limited support for manipulating tables with full-text indexes. You cannot use Database Designer or Table Designer to create a full-text index, but if you modify a table with full-text indexes, the Database Designer or Table Designer warns you if your modification affects the full-text index. In most cases, however, when you save your modifications, the Database Designer or Table Designer will be able to reestablish the table's full-text indexes.

For more information, see <u>Full-Text Indexing Support</u>.

Changing Column Properties

For information on the procedure you use to set column properties, see <u>Setting</u> <u>Column Properties</u>.

The following items contain information specific to Microsoft SQL Server databases:

- <u>Default Values for Data Types</u>
- Changing the Data Type Assigned to a Column
- <u>Changing the Column Length</u>
- <u>Changing the Column Precision</u>
- <u>Changing the Column Scale</u>
- Changing the Null Option Assigned to a Column
- Assigning a Default Value to a Column
- <u>Redefining a Global Default</u>

• <u>Changing a Column's Identity Properties</u>

Default Values for Data Types

The following <u>default</u> values are automatically added for a new column:

Data Type	Description	Default value
Column Name	The name of a column in a table. Column names must conform to rules for identifiers and must be unique in the table.	Blank
Datatype	The data type of the column. System- or <u>user-defined data types</u> are acceptable.	Character (char)
Length	numeric data types) or characters	Differs for different data types (e.g., 10 for Character, 50 for VARBINARY)
Precision	The maximum total number of decimal digits that can be stored, both to the left and to the right of the decimal point.	0
Scale	The maximum number of decimal digits that can be stored to the right of the decimal point. This value must be less than or equal to the precision. Applies only to DECIMAL and NUMERIC data types.	0
Allow Nulls	Whether or not the column can accept null values.	Yes (selected)
Default Value	The value that will be inserted into the column if the user does not make an entry. Default values are ignored for columns with a timestamp data type. If you do not define a default value and a	Blank

	column allows nulls, NULL will be	
	inserted.	
Identity	lentity Whether or not the column will	
	generate incremental values for new	
	rows based on the Identity Seed and	
	Identity Increment settings.	
Identity Seed	The value assigned to the first row in	Blank or 1.
	the table. If the Identity setting is No,	
	Identity Seed is blank. If the Identity	
	setting is Yes, Identity Seed defaults	
	to 1.	
Identity	The value which is added to the	Blank or 1.
Increment	Identity Seed and assigned to the	
	second row in the table. Each	
	subsequent row is increased by this	
	value. If the Identity setting is No,	
	Identity Increment is blank. If the	
	Identity setting is Yes, Identity	
	Increment defaults to 1.	

Changing the Data Type Assigned to a Column

A column's data type determines what kind of data can be stored in the column. A list of system-defined data types appears in the Data type column.

You can choose the appropriate data type for the information you want to store in the column. <u>User-defined data types</u> appear at the end of the data type list. The system-defined data type that corresponds to the user-defined data type appears in parentheses at the end of the user-defined data type name. For example: "id (varchar)."

Special considerations for assigning a user-defined data type to a column

- Selecting a data type automatically sets the length, precision, and scale for the column based on the data type's definition. You cannot change these settings for user-defined data types.
- You can change the Allow Nulls setting only if the user-defined data type allows null values.

For more information, see <u>Creating User-Defined Data Types</u>.

Note Changing the data type recreates the table in the database when you save the table or diagram.

Caution If you change the data type of a column that is related to columns in other tables, then the data type of the related columns must also be changed to preserve referential integrity. When you save the table or diagram, the <u>Datatype Change Required dialog box</u> enables you to automatically change the data type of the related columns.

For more information, see <u>Data Types</u>.

Changing the Column Length

When you select a data type, the <u>column</u> length is automatically defined. You can reset the length property for a column with a data type of binary, char, nchar, nvarchar, varbinary, or varchar if you want to increase or decrease the length of acceptable values in that column. For columns with other data types, the length is derived from the data type.

Changing the Column Precision

For most data types, the column precision is automatically defined. You can change the column precision for the decimal and numeric data types if you want to redefine the maximum number of digits these columns use. The precision of a numeric column refers to the maximum number of digits used by the selected data type. The precision of a non-numeric column generally refers to either the maximum length or the defined length of the column.

The Database Designer prevents you from changing the precision of a column

whose data type is not decimal or numeric.

Changing the Column Scale

When you select a data type, the column scale by default is set to 0. The scale of a numeric column refers to the maximum number of digits to the right of the decimal point. For columns with approximate floating point numbers, the scale is undefined because the number of digits to the right of the decimal point is not fixed.

You can change the scale for a numeric or decimal column if you want to redefine the number of digits that can appear to the right of the decimal point.

Changing the Null Option Assigned to a Column

For each column in your table, you can specify whether to allow null values or disallow null values. A null value, or NULL, is not the same as zero (0) or blank; NULL means that no entry has been made. Its presence usually implies that the value is either unknown or undefined. For example, a null value in the price column of the titles table of the pubs sample database does not mean that the book has no price; it means that the price is unknown or has not been set.

If null values are not allowed, the user entering data in the table must enter a value in the column or the table row cannot be accepted in the database.

Note You cannot change this property on a <u>primary key</u> column. Also, identity columns cannot have null values. That is, you cannot create or modify a column so that its **Identity** setting is Yes and its **Nulls Allowed** setting is Yes.

Assigning a Default Value to a Column

For each column in your table, you can specify a <u>default</u> value that will be entered in the column if the user leaves it blank. If you do not assign a default value and the user leaves the column blank, then:

• If you set the option to allow null values, NULL will be inserted into the column.

• If you did not set the option to allow null values, the column will remain blank, but you will not be able to save the row until you supply a value for the column.

For text strings, enclose the value in single quotation marks ('); do not use double quotation marks ('') because they are reserved for quoted identifiers. For example, type: 98036 or 'Paris, France'.

If your entry in the **Default Value** column replaces a bound default (which is shown without parentheses), the default will be unbound and the new value will replace it.

Redefining a Global Default

A global <u>default</u> is one that is defined for a specific database and is shared by columns of different tables. For example, suppose several of your tables have a quantity column. You can define a global default in your database that inserts a value of 1 in the quantity column whenever the user leaves that column blank in any table.

If a global default is bound to a column, you can specify a different default value for that column in a specific table. In such a case, the existing global default is unbound from the column before the new default value is bound to the column.

To redefine a global default

- 1. In your database diagram, assign a new default value to the column you want to change.
- 2. A message prompts you to permanently unbind the existing default in order for the new default to be applied. Choose **OK**.

Changing a Column's Identity Properties

You can change the identity properties of a column if you want to redefine the sequential numbers that are automatically generated and stored in that column

when new rows are added to the table. You can set the identity properties on only one column per table.

Columns that have the identity property contain system-generated sequential values that uniquely identify each row within a table (for example, employee identification numbers). When inserting values into a table with an identity column, Microsoft SQL Server automatically generates the next identifier based on the last used identity value (the identity seed property) and the increment value (the identity increment property) specified during the creation of the column.

The identity property can be set only for a column whose data type is decimal, int, numeric, smallint, bigint, or tinyint and that disallows null values.

To change a column's identity properties

- 1. In your database diagram, select the table in which you want to change the identity properties of a column.
- 2. If you are not already in **Standard** view, right-click the table and choose **Table View**, then **Standard** from the shortcut menu.
- 3. If the **Allow Nulls** property is selected, clear the check box.
- 4. Select the **Identity** cell for the column whose values you want to automatically increment.

Note Only one column per table can be defined as an identity column.

- 5. Type a value in the **Identity Seed** cell. This value will be assigned to the first row in the table. If you leave this cell blank, the value 1 will be assigned by default.
- 6. Type a value in the **Identity Increment** cell. This value is the increment that will be added to the **Identity Seed** for each subsequent row. If you leave this cell blank, the value 1 will be assigned by

default.

For example, suppose you want to automatically generate a 5-digit Order ID for each row added to the orders table, beginning with 10000 and incremented by a value of 10. To do this, you would select the **Identity** property box, type an **Identity Seed** of 10000, and type an **Identity Increment** of 10.

If you change any of the identity properties for a table, the existing identity values will be preserved. Your new settings apply only to new rows that are added to the table.

Note If an identity column exists for a table with frequent deletions, gaps can occur between identity values. If you want to avoid such gaps, do not use the identity property.

See Also

Constraints | Creating a Relationship Between Tables | Creating an Index | Database Designer | Deleting a Check Constraint | Enforcing Referential Integrity Between Tables | Setting Column Properties | Table Relationships Visual Database Tools

Query Designer Considerations for SQL Server Databases

The following guidelines provide information about SQL Server-specific features that you can use.

Below you will find information about:

- <u>SQL Syntax in Query Designer</u>
- Identifying Database Objects
- <u>Using Quotation Marks</u>
- <u>Case Sensitivity</u>
- Entering Keywords in the Grid and SQL Panes
- Entering Currency Values
- <u>Using the GUID Data Type</u>
- Entering Blanks
- Including Optimizer Hint Comments
- ANSI to OEM Character Conversion
- <u>Unsupported and Partially Supported Query Types</u>
- <u>Working with Tables from Different Data Sources</u>

SQL Syntax in Query Designer

When the Query Designer builds a statement in the <u>SQL pane</u>, it will use syntax specific to SQL Server whenever possible. For example, <u>database objects</u> such as <u>tables</u> and <u>views</u> are qualified using SQL Server owner names.

You can also type SQL Server-specific syntax in the SQL pane. In some cases when you verify a query, the Query Designer converts server-specific syntax to ANSI standard syntax. However, the changed query will always return the same results.

Identifying Database Objects

When you enter the names of database objects (tables, views, and columns) in the SQL pane, you must provide sufficient information for SQL Server to identify the object you want. Database objects are identified with unique names that consist of up to three parts (for tables and views) or four parts (for columns):

database.owner.table

database.owner.table.column

Note You can join tables from different databases on the same server. In that case, database objects can have four part names. For more details, see <u>Working with Tables from Different Data Sources</u>.

In general, you need to provide only enough qualifiers to uniquely identify the object you want to work with. For example, if you are working with a column called price in the titles table in the current database, you can simply reference the column by name, as in this SQL statement:

SELECT price FROM titles

However, if you are working with two tables, such as orders and products, and each has a column called price, you must qualify references to the column with the appropriate table name, as in this example:

SELECT products.prod_id, orders.price

```
FROM orders INNER JOIN products ON
orders.prod_id = products.prod_id
```

When you use the <u>Diagram pane</u> and <u>Grid pane</u> to work with tables in the current database, the Query Designer automatically adds owner and table qualifiers for you. If you are not the owner of a table that you are working with, the owner's name will appear in the table names. For example, if you work in the pubs database, the owner name dbo will appear in front of table names. If you are working with multiple tables, the Query Designer adds table name qualifiers to column names.

Using Quotation Marks

The standard delimiters for literal strings in SQL are single quotation marks ('). By default, SQL Server reserves double quotation marks ('') as delimiters for database objects.

The SQL Server ODBC driver supports a Quoted Identifiers setting for the session or connection. If this setting is on, double quotation marks are interpreted as delimiters for identifiers. However, if you turn this setting off, double quotation marks are interpreted instead as delimiters for literal strings.

To avoid ambiguity, the Query Designer always sets Quoted Identifiers on, so that double quotation marks are always interpreted as database object delimiters. If you have previously turned Quoted Identifiers off, the Query Designer overrides your setting.

Therefore, in the Query Designer, always use single quotation marks to enclose string literals. Use double quotation marks only as needed for database objects delimiters.

Case Sensitivity

Text information in a SQL Server database can be stored in uppercase letters, lowercase letters, or a combination of both. For example, a last name can appear as "SMITH," "Smith," or "smith."

Depending on how SQL Server was installed, databases can be case-sensitive or

case-insensitive. If a database is case-sensitive, when you search for text data, you must construct your <u>search conditions</u> using the exact combination of uppercase and lowercase letters. For example, if you are looking for a name such as "Smith," you cannot use the search conditions "=smith" or "=SMITH."

In addition, if the server was installed as case-sensitive, you must provide database, owner, table, and column names using the correct combination of uppercase and lowercase characters. If the case of the name you provide does not match exactly, SQL Server returns an error reporting an "invalid object name."

When you create queries using the Diagram and Grid panes, the Query Designer will always accurately reflect the case-sensitivity of your server. However, if you enter queries in the SQL pane, you must be careful to match names to the way they will be interpreted by the server.

If the server was installed with a case-insensitive option, you can enter database object identifiers and search conditions using any combination of uppercase and lowercase characters.

Tip To determine the case sensitivity of a server, execute the <u>stored procedure sp_server_info</u>, and then examine the contents of row 18. If the server has been installed with the case-insensitive setting, the option for sort_order will be set to nocase. You can run a stored procedure from the Query Analyzer.

Entering Keywords in the Grid and SQL Panes

The Query Designer supports the use of certain SQL Server constants, variables, and reserved column names in the Grid or SQL panes. Generally, you can enter these values by typing them in, but the Grid pane will not display them in drop-down lists. Examples of supported names include:

- **IDENTITYCOL** If you enter this name in the Grid or SQL pane, the SQL Server will recognize it as a reference to an auto-incrementing column.
- **Predefined global values** You can enter values such as @@CONNECTIONS and @@CURSOR_ROW into the Grid

and SQL panes.

- **Constants (niladic functions)** You can enter constant values such as CURRENT_TIMESTAMP and CURRENT_USER in either pane.
- **NULL** If you enter NULL in the Grid or SQL panes, it is treated as a literal value, not a constant.

Entering Currency Values in the Grid Pane

In the Grid pane, to specify that you want data interpreted as money, precede the value with \$ or \$- (for negative values). Do not include a comma or other delimiter to indicate thousands. Formatting values this way alerts the Query Designer that you are entering values to be treated as or compared to data in money or smallmoney type columns. Values are rounded to the nearest hundredth of a unit.

You can use \$ no matter what currency you are working with. When a query displays values from money columns in the Results pane, it does not include the \$ prefix. Depending on the setting in the Windows Regional Settings dialog box, currency data might or might not include a comma or other delimiter for thousands.

Using the GUID Data Type

You can include references to the GUID data type, which is used to store globally unique identifiers. In Update and Insert From queries you can call the newid() function to generate a new GUID to be stored in the database.

When you are creating a Select query, the only operations allowed with a GUID type column are comparisons based on equality (= and $\langle \rangle$).

Entering Blanks

You can specify a zero-length string in an Update or Insert Into query by entering two single quotation marks, as in the following example:

```
UPDATE employee
SET minit = "
WHERE emp_id = 'CFS88322F'
```

In versions of SQL Server 6.5 or earlier, two single quotation marks are treated as a single space. For example, you can use quotation marks in the following expression: 'abc' + " + 'def'. The resulting value would be 'abc def'.

Including Optimizer Hint Comments in the SQL Pane

If you are entering a query directly in the SQL pane, you can add optimizer hints to specify the use of specific indexes, locking methods, and so on. However, when reformatting the contents of the SQL pane, the Query Designer might not maintain these comments. Optimizer comments are not represented graphically.

For more information, see <u>Hints</u>.

ANSI to OEM Character Conversion

Data containing extended characters — that is, characters outside the ASCII range 32 (space) to 126 (~), including international characters such as "ä," "ç," "é," "ñ," and "ß" — can require special handling when you are working with SQL Server.

The representation of extended characters in a <u>result set</u> depends on the code page in use. A code page is a character set that a computer uses to interpret and display data properly. Code pages usually correspond to different platforms and languages and are used in international applications. For example, the ASCII value 174 might appear as the symbol "®" in one code page but as a chevron character in another code page.

In general, code pages are divided into ANSI code pages and OEM code pages. ANSI code pages, in which high-numbered ASCII values represent international characters, are used in Windows. OEM code pages, in which high-numbered ASCII values represent line-drawing and punctuation characters, were designed for MS-DOS®.

When data is entered into a SQL Server database, SQL Server settings on the local (client) computer specify whether the data is stored in ANSI or OEM format. The option is specified using the **Automatic ANSI to OEM conversion** option on **the DB Library Options** tab in the **SQL Server Client Configuration** dialog box. This dialog box is available by clicking the Microsoft SQL Server **Client Network Utility** from Programs on the **Start** menu. (For more information, see <u>Using the DB-Library Automatic ANSI to OEM</u> <u>Conversion Option</u>.)

By default, this option is selected for the SQL Server Client, a choice which causes the data to be converted from high-numbered ASCII characters to OEM characters. For example, if the OEM conversion option is set and you enter the name "Günther" in a column and then save the row, the character "ü" will be converted to another character before the row is stored in the database.

The results of queries that you create in the Query Designer are affected by the format in which extended-character data is stored in combination with the setting of the OEM conversion option in the SQL Server Client Configuration dialog box. Depending on these variables:

- You might not be able to search for data that includes high-order ASCII characters.
- Your query results might appear in the <u>Results pane</u> with incorrect characters substituted for high-order ASCII characters.

In general, if data is stored in OEM format, you should set the OEM conversion option so the data will display properly and so you can search it. If data is stored in ANSI format (that is, it was not converted to OEM format) but you have set the OEM conversion option, the data will not display properly and you will not be able to search for it.

To determine whether data was stored in OEM format, you can use a query to display the contents of the table or tables you are working with. If extended characters appear incorrectly, the OEM conversion setting is probably wrong. Close the query and the project, change the setting in the SQL Server Client Configuration dialog box, and then open the project and query again.

Query Designer Unsupported and Partially Supported Query Types

Some types of legal SQL Server queries cannot be represented graphically in the Query Designer. You can still enter them in the SQL pane, and they will execute correctly. However, the Query Designer will display the <u>Query Definitions</u> <u>Differ</u> dialog box and report an error when you execute your query or change panes.

Several types of SQL Server queries are not supported graphically, including:

- Queries using INTERSECT.
- Queries using UNION [ALL].
- Queries using CASE.
- Any data definition (DDL) query, including CREATE TABLE, ALTER TABLE, CREATE PROCEDURE, ALTER PROCEDURE, and so on. CREATE VIEW and ALTER VIEW queries are not supported graphically, but you can use the View Designer to create and edit views.
- Update and Delete queries that include an extra FROM clause (FROM *table* FROM *table*) that specifies the list of rows to update or delete.
- Queries using the FOR BROWSE clause.
- Queries that include UPDATE as a search condition.
- Queries including CURRENT OF.

Working with Tables from Different Data Sources

You can create distributed, heterogeneous queries —queries from tables and table-structured objects outside the server to which you have created a data connection. SQL Server can access any data source that supports OLE DB. You can use tables and table-structured objects from these outside data sources as you would any tables available on the base server (if you have proper permissions to access to the outside data source).

Microsoft SQL Server can access outside data sources in two ways. The first is using a linked server, which is defined in the SQL Server database to point to the outside data source. A linked server makes the data source accessible using a naming convention similar to that of native SQL Server data objects. The second is to use a dynamic reference to the outside source using the OpenRowset() function, which allows you to connect to any accessible data source in your query, even if no linked server is defined for it.

Tables and table-structured objects from outside servers do not appear as part of the list of tables. Instead, to use them, you use syntax in the SQL pane of the Query Designer to refer to the linked server or to include the OpenRowset() function. However, when you refer to an outside data source in the SQL pane, the Query Designer adds a rectangle representing the table or table-structured object to the Diagram pane to represent the outside data source.

To refer to an outside data source using a linked server

• In the **SQL** pane, use the following syntax to refer to the table: *linkserver.catalog.schema.object*

Where:

- *linkserver* represents the name on the local Microsoft SQL Server data source given to the linked server in OLE DB.
- *catalog* represents the name of the database containing the object.
- *schema* represents the owner of the object.
- *object* represents the table or view in the database.

Note You must define the SQL Server data source (the *linkserver* part of the name) before you use this name in the query.

The following is an example of an SQL statement that joins data from tables from the local database with a table on a server called "hrserver":

SELECT e.id, e.lname, h.hiredate FROM employee AS e INNER JOIN hrserver.hr.dbo.hiredata AS H ON e.emp_id = h.emp_id

To refer to an outside data source dynamically

In the SQL pane, use the OpenRowset() function in place of a table reference, with the following syntax:
 OpenRowset(provider,connectString,object)

Where:

- *provider* represents the friendly name of the OLE DB provider.
- *connectString* represents a string that includes information for connecting to the outside data source. This parameter is optional if an ODBC connection is already established between the local server and the outside data source. The connectString parameter can take these forms:

datasource;user id;password, which lists specific connection attributes.

-or-

provider string, which is a single string of named attributes with values for creating the connection, similar to the string used in a .dsn file.

• *object* represents the name of a database object. You can refer to a table, view, or other database object using the standard naming convention recognized by the outside data source.

Tip The data source referenced by OpenRowset() is easier to

work with if you assign it a table alias.

The following examples illustrate variations on using OpenRowset() to dynamically access data from an outside data source. The first shows access using a set of attributes for the connect string to a Microsoft® Jet (Access) database. The second example shows how you can pass a connect string.

```
SELECT n.*
FROM OpenRowset('Microsoft.jet.OLEDB.3.51',
    'c:\nwind.mdb';'admin';'pwd', authors)
AS n
SELECT a.*
FROM OpenRowset('MSDASQL', 'Driver=SQL Server;Server=Test;
    UID=user1;PWD=pwd', pubs.dbo.authors)
```

AS a

See Also

<u>Creating Make Table Queries</u> | <u>Creating Queries</u> | <u>Designing Queries</u> | <u>Specifying</u> <u>Parameter Marker Characters</u> | <u>Supported Query Types</u> | <u>Using Expressions in a</u> <u>Query</u> Visual Database Tools

Database Development and Visual Database Tools

As you design a database, you create database objects such as tables, columns, keys, indexes, relationships, constraints, and views. To help you create these objects, the Visual Database Tools provides three mechanisms: the Database Designer, the Table Designer, and the View Designer.

- **The Database Designer** A visual tool allowing you to create tables, columns, keys, indexes, relationships, and constraints. Within the Database Designer, you interact with database objects through database diagrams, which graphically show the structure of the database. With the Database Designer you can create and modify objects that are visible on diagrams (tables, columns, relationships, and keys) and some objects that are not visible on diagrams (indexes and constraints).
- **The Table Designer** A visual tool allowing you to create an individual table. Although you can create tables with the Database Designer, the Table Designer is sometimes more convenient for this task, because it devotes a larger portion of the screen to the table and shows more detail about the table as you design it.
- **The View Designer** A visual tool that helps you create views. Because the SQL syntax for creating views is almost identical to the syntax for creating queries, the View Designer is very similar to the Query Designer.

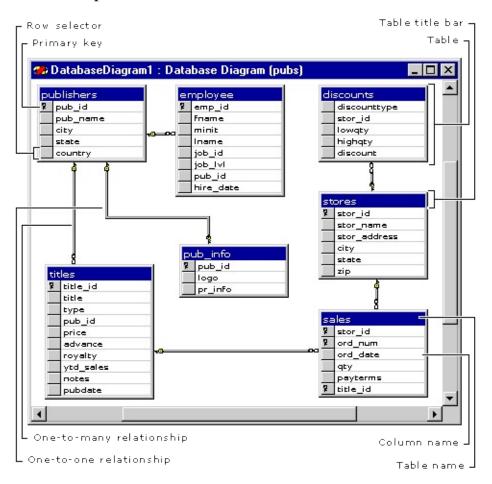
See Also

Database Designer | Table Designer | Query and View Designer Layout

Visual Database Tools

Database Designer

The Database Designer is a visual tool allowing you to design and visualize a database to which you are connected. When designing a database, you can use the Database Designer to create, edit, or delete tables, columns, keys, indexes, relationships, and constraints. To visualize a database, you can create one or more diagrams illustrating some or all of the tables, columns, keys, and relationships in it.



For any database, you can create as many database diagrams as you like; each database table can appear on any number of diagrams. Thus, you can create different diagrams to visualize different portions of the database, or to accentuate different aspects of the design. For example, you can create a large diagram showing all tables and columns, and you can create a smaller diagram showing all tables without showing the columns.

Each database diagram you create is stored in the associated database.

Tables and Columns in a Database Diagram

Within a database diagram, each table can appear with three distinct features: a title bar, a row selector, and a set of property columns.

Title Bar The title bar shows the name of the table. If another user owns the table, then that user's name appears in parentheses at the end of the table name. For information about table owners, see <u>Ownership of Database Objects</u>.

authors (JohnDoe)

If you have modified a table and have not yet saved it, an asterisk (*) appears at the end of the table name to indicate unsaved changes. For information about saving modified tables and diagrams, see <u>Working with Databases</u>.

```
authors * Or authors (JohnDoe) *
```

Row Selector You can click the row selector to select a database column in the table. The row selector displays a key symbol

8

if the column is in the table's primary key. For information about primary keys, see <u>Defining a Primary Key</u>.

Property Columns The set of property columns is visible only in the certain views of your table. You can view a table in any of four different views to help you manage the size and layout of your diagram.

For more information about table views, see <u>Changing a Table View in a</u> <u>Database Diagram</u>.

Relationships in a Database Diagram

Within a database diagram, each relationship can appear with three distinct features: endpoints, a line style, and related tables.

Endpoints The endpoints of the line indicate whether the relationship is one-to-one or one-to-many. If a relationship has a key at one endpoint and a figure-

eight at the other, it is a one-to-many relationship. If a relationship has a key at each endpoint, it is a one-to-one relationship.

Line Style The line itself (not its endpoints) indicates whether the Database Management System (DBMS) enforces referential integrity for the relationship when new data is added to the foreign-key table. If the line appears solid, the DBMS enforces referential integrity for the relationship when rows are added or modified in the foreign-key table. If the line appears dotted, the DBMS does not enforce referential integrity for the relationship when rows are added or modified in the foreign-key table.

Related Tables The relationship line indicates that a foreign-key relationship exists between one table and another. For a one-to-many relationship, the foreign-key table is the table near the line's figure-eight symbol. If both endpoints of the line attach to the same table, the relationship is a reflexive relationship. For more information, see <u>Drawing a Reflexive Relationship</u>.

See Also

Database Designer Properties Pages | Database Designer Dialog Boxes

Visual Database Tools

Table Designer

The Table Designer is a visual tool allowing you to design and visualize a single table in a database to which you are connected.

╊ 2:Design Table 'pu	blishers'			_ 🗆 ×			
🖬 🕾 🎬 X 🗈 🖻 7 🏙 🍣 寻 🦓							
Column Name	Data Type	Length	Allow Nulls				
💦 pub_id	char	4					
pub_name	varchar	40					
city	varchar	20					
state	char	2					
country	varchar	30		•			
Columns Description Default Value Precision Scale Identity Identity Seed Identity Increment Is RowGuid Formula Collation	0 0 No No <database defa<="" th=""><th>ult></th><th></th><th></th></database>	ult>					

The Table Designer has two parts. The upper part shows a grid; each row of the grid describes one database column. For each database column, the grid displays its fundamental characteristics: column name, data type, length, and nulls-allowed setting.

The lower portion of the Table Designer shows additional characteristics for whichever data column is highlighted in the upper portion.

From the Table Designer, you can also access property pages through which you can create and modify relationships, constraints, indexes, and keys for the table.

See Also

Table Designer Properties Pages | Database Designer Dialog Boxes

Interactions Among Database Diagrams and Table Design Windows

When you connect to a database and begin designing or modifying a database diagram or a table, the Visual Database Tools retain your work in memory. That is, the tools do not transmit your work to the database until you explicitly save the work there. Regardless of how many database diagrams you open or how many tables you design, the Visual Database Tools retain a single in-memory model of the database structure. There are several ramifications:

• You can experiment with different object definitions

Because your modifications are not saved to the database immediately, you can experiment to see how a proposed modification will affect the database. When you complete you modifications, you can either save your changes to the database, save your changes to a script file, or discard your changes.

• Your modifications can appear in many diagrams or table design windows

When you modify a database object, every open diagram containing that object will reflect that modification. For example, if you add a column to a table, the new column appears on every open diagram containing that table. If you modify an object and later add that object to another diagram, the added object reflects the modifications — even if you have not yet saved the modifications to the database.

• Your modifications can exist in memory only

The in-memory model of the database structure endures until you close all database diagrams and table design windows for that database. Thus, it is possible that the in-memory model retains modifications that are not visible on any open database diagrams. Even if you remove a modified object from the only open diagram, the modification remains in memory. If you later add the same object to any diagram, the modification will be visible. **Note** Query Designer windows and View Designer windows use a different strategy to retain your work in memory. For more information, see Interactions Among Query and View Designer Windows

Database Objects

When you use Visual Database Tools to design a database, you create database objects such as the following.

- <u>Tables</u>
- <u>Columns</u>
- <u>Keys</u>
- <u>Table Relationships</u>
- <u>Indexes</u>
- <u>Constraints</u>
- <u>Triggers</u>
- <u>Stored Procedures</u>

See Also

Queries and Views

Uniqueness of Database Object Names

The database catalog in your database contains one row for each object (constraint, default, log, rule, stored procedure, etc.) created within a database. Because the DBMS enforces certain rules to ensure that objects have unique names, you must take care when naming objects.

When you name database objects in a database diagram or table diagram, you will be alerted if the name you choose is already used by another object. There are three possible outcomes when you name an object:

- You choose a duplicate name detected by the Visual Database Tools. In this case, you receive an error as soon as you try to name the object.
- You choose a duplicate name detected by the DBMS. In this case, the Visual Database Tools initially accept the name, but the DBMS rejects the name when you try to save the object to the database; you receive an error.
- You choose a unique name. In this case, you receive no error.

See Also

Ownership of Database Objects

Tables

A database consists of one or more tables. A <u>table</u> is a collection of data, arranged in <u>rows</u> and <u>columns</u>. For example, you might have a table for author information called authors. Each column would contain a certain type of information, such as the author's last name. Each row would contain all the information about a specific author: first name, last name, address, and so on.

In a database, you might have a number of tables, each devoted to a specific topic. For example, the pubs database might contain tables for authors, titles, and so on. Using a separate table for each topic can eliminate duplicate data, make data storage more efficient, and reduce data-entry errors.

Tables are the basic building blocks of database diagrams. In a database diagram, each table is laid out in a matrix so that you can see all the <u>properties</u> defined for every column in your database table.

Tables in a Database Diagram

Each table in a database diagram has three distinct features: a title bar, a row selector, and a set of property columns.

• **Title Bar** The title bar shows the name of the table. If another user owns the table, then that user's name appears in parentheses at the end of the table name.

authors (JohnDoe)

For information about table owners, see <u>Multiuser Environments</u>.

or

If you have modified a table and have not yet saved it, an asterisk (*) appears at the end of the table name to indicate unsaved changes. For information about saving modified tables and diagrams, see <u>Working with Databases</u>.

authors *

authors (JohnDoe) *

• **Row Selector** You can click the row selector to select a database column in the table. The row selector displays a key symbol 🕅 if the

column is in the table's <u>primary key</u>. For information about primary keys, see <u>Defining a Primary Key</u>.

• **Property Columns** The set of property columns is visible only in certain views of your table. You can view a table in any of five different views to help you manage the size and layout of your diagram.

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls	
•	pub_id	char	4	0	0	· · · ·	
	pub_name	varchar	40	0	0	\checkmark	
	city	varchar	20	0	0	\checkmark	
	state	char	2	0	0	\checkmark	
	country	varchar	30	0	0	\checkmark	

For more information about table views, see <u>Changing a Table View</u>.

Before you begin defining the columns for a table, determine what type of data the table will hold and how that table relates to the other tables in your database.

Designing Tables

To determine the structure of a new table, you need to decide:

- What type of data the table will contain.
- What columns you need in the table and the data type (and length, if required) for each column.
- Which columns should accept null values. For information about data types, allowing null values, and other column properties, see <u>Working</u> <u>with Columns</u>.
- Whether to use constraints and if so, where. For more information, see <u>Constraints</u>.

• What types of <u>indexes</u> you need, where you need them, and which columns should be the <u>primary key</u> and <u>foreign key</u>. For more information, see <u>Indexes</u>.

After you decide on the structure of your table, you can create the table and define its columns in your database diagram or with the Table Designer. You can also alter the table's appearance in your diagram so that the information you need is visible when you need it. When you save your table or the diagram, the table is created in your database.

If you know exactly what you want in a table, it is often most efficient to define everything you need at the beginning, including the table's data restrictions and additional properties. However, in many cases, you will do best to first create a basic table and save it so it is created in your database. You can then add some test data to the table and experiment with the table in the database diagram to fine-tune its design.

The Database Designer lets you try out different designs by working with tables in your diagram. Through experimentation, you can determine what types of data are frequently entered and queried and then redesign your table accordingly.

When you change a table's design in a database diagram or in Table Designer, any data that is stored in the table is preserved to the extent possible. When you are satisfied with your basic design, you can add constraints, indexes, and any additional columns that you require. For more information, see <u>Developing</u> <u>Database Structure</u>.

То	See
Add new, existing, or related tables	Adding Tables
to your database diagram	
Copy a table from one database	Copying a Table Across Database
diagram to another diagram	Diagrams
Create a new table that contains some	Duplicating a Table
of the same columns as an existing	
table in your diagram	
Delete a table from a database	Deleting a Table from a Database
diagram and the database	Diagram and the Database
Change the name of a table in a	Renaming a Table

database diagram and in the database	
Change the amount of information	<u>Changing a Table View in a Database</u>
shown for the tables in the diagram	Diagram
Add or delete columns to a table,	Working with Columns
change column property settings	

Columns

In a table, data is arranged into columns. Each column stores one data element, such as a first name, one line of an address, a price, or any similar discrete unit of information.

When columns are created in a table, they are given a name that identifies their purpose, such as FirstName or Address1. In most databases, you must also specify additional properties, such as how long the longest entry in the column will be, and what type of data the column will contain - characters, integers, floating-point numbers, dates or times, and so on. Other column properties can include whether the column is the table's primary key, whether users must enter a value into it, and what its default value is.

See Also

Column Properties

Column Properties

Each <u>column</u> in a table has a set of properties. Each <u>property</u> defines one characteristic, such as the name, data type or length, of a column. The entire set of properties for a column makes up that column's definition in your database table.

You can set column properties directly in your database table in a database diagram. Three column properties are required — the column name, data type, and length — before you can save a table in your database. You can redefine a column by editing any of its properties. For example, you can rename a column, alter its length, specify a default value, and so on.

When you view your table in Standard view, the column properties are laid out in a grid. In this grid, you can choose which properties you want to see. You can hide the properties that you rarely define and show the properties that you define more frequently. You can also resize the columns in the grid so that the properties are easy to read.

Column Name	Datatype	Length	Precision	Scale	Allow Nulls	-
▶ pub_id	char	4	0	0	2	
pub_name	varchar	40	0	0	\checkmark	
city	varchar	20	0	0	\checkmark	
state	char	2	0	0	\checkmark	
country	varchar	30	0	0		

For more information about showing, hiding, and resizing property columns, see:

- <u>Setting Column Properties</u>
- <u>Resizing Property Columns</u>
- Changing Which Properties Appear
- <u>Renaming a Column</u>

Keys

There are two kinds of keys. A primary key is a set of columns from a table that are guaranteed to have unique values for each row of that table. A primary key is also called a primary key constraint, because it effectively constrains the values you can add to the table: it prevents you from adding a row to the table whose primary key columns are all equal to the corresponding values of some other row in that table.

A foreign key is a correspondence between a set of columns in one table and the set of primary key columns in some other table. When discussing foreign keys, the two participating tables are sometimes called the foreign-key table and the primary-key table. A foreign key is also called a foreign key constraint because it constrains table rows: it ensures that any row you add to the foreign-key table has a corresponding row in the primary-key table. That is, it requires that any row added to the foreign-key table have values in the foreign-key column that correspond to the respective values of the primary key columns for some row in the primary-key table.

For more information about primary keys, see <u>Primary Key Constraints</u>. For more information about foreign keys, see <u>Foreign Key Constraints</u>. For details about working with keys, see <u>Working with Keys</u>.

Table Relationships

You can create <u>relationships</u> between your tables in a database diagram to show how the columns in one table are linked to columns in another table.

In a relational database, relationships enable you to prevent redundant data. For example, if you are designing a database that will track information about books, you might have a table called titles that stores information about each book, such as the book's title, date of publication, and publisher. There is also information you might want to store about the publisher, such as the publisher's phone number, address, and zip code. If you were to store all of this information in the titles table, the publisher's phone number would be duplicated for each title that the publisher prints.

A better solution is to store the publisher information only once in a separate table, publishers. You would then put a pointer in the titles table that references an entry in the publisher table.

To make sure your data is not out of sync, you can enforce <u>referential integrity</u> between the titles and publishers tables. Referential integrity relationships help ensure information in one table matches information in another. For example, each title in the titles table must be associated with a specific publisher in the publishers table. A title cannot be added to the database for a publisher that does not exist in the database.

For a better understanding of table relationships, see:

- <u>Types of Table Relationships</u>
- **Overview of Referential Integrity**

Types of Table Relationships

A relationship works by matching data in key columns — usually columns with the same name in both tables. In most cases, the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table. For example, sales can be

associated with the specific titles sold by creating a relationship between the title_id column in the titles table (the primary key) and the title_id column in the sales table (the foreign key).

There are three types of relationships between tables. The type of relationship that is created depends on how the related <u>columns</u> are defined.

- <u>One-to-Many Relationships</u>
- Many-to-Many Relationships
- <u>One-to-One Relationships</u>

One-to-Many Relationships

A <u>one-to-many relationship</u> is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A. For example, the publishers and titles tables have a one-to-many relationship: each publisher produces many titles, but each title comes from only one publisher.

A one-to-many relationship is created if only one of the related columns is a <u>primary key</u> or has a <u>unique constraint</u>.

The primary key side of a one-to-many relationship is denoted by a key \square symbol. The foreign key side of a relationship is denoted by an infinity \square symbol.

Many-to-Many Relationships

In a <u>many-to-many relationship</u>, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table, called a <u>junction table</u>, whose primary key consists of the foreign keys from both table A and table B. For example, the authors table and the titles table have a many-to-many relationship that is defined by a one-to-many relationship from each of these tables to the titleauthors table. The primary key of the titleauthors table is the combination of the au_id column (the authors table's primary key) and the title_id column (the titles table's primary key).

One-to-One Relationships

In a <u>one-to-one relationship</u>, a row in table A can have no more than one matching row in table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.

This type of relationship is not common because most information related in this way would be all in one table. You might use a one-to-one relationship to:

- Divide a table with many columns.
- Isolate part of a table for security reasons.
- Store data that is short-lived and could be easily deleted by simply deleting the table.
- Store information that applies only to a subset of the main table.

The primary key side of a one-to-one relationship is denoted by a key symbol. The foreign key side is also denoted by a key symbol.

Overview of Referential Integrity

Referential integrity is a system of rules that ensure relationships between rows in related tables are valid and that you do not accidentally delete or change related data.

When referential integrity is enforced, you must observe the following rules:

• You cannot enter a value in the foreign key column of the related table if that value does not exist in the primary key of the related table. However, you can enter a null in the foreign key column. For example, you cannot indicate that a job is assigned to an employee who is not

included in the employee table, but you can indicate that an employee has no assigned job by entering a null in the job_id column of the employee table.

- You cannot delete a row from a primary key table if rows matching it exist in a related table. For example, you cannot delete a row from the jobs table if there are employees assigned to the job represented by that row in the employee table.
- You cannot change a primary key value in the primary key table if that row has related rows. For example, you cannot delete an employee from the employee table if that employee is assigned to a job in the jobs table.

You can set referential integrity when all of the following conditions are met:

- The matching column from the primary table is a primary key or has a <u>unique constraint</u>.
- The related columns have the same data type and size.
- Both tables belong to the same database.

Enforced and Unenforced Relationships in Database Diagrams

Creating a relationship line in a database diagram automatically enforces referential integrity by creating a foreign key constraint on the related table. An enforced relationship appears in your database diagram as a solid line. For example:

An unenforced relationship, whose foreign key constraint is disabled, appears in your diagram as a dashed line. For example:

____00

Depending on the features of your database, you can set options to disable the foreign key constraint for certain conditions, for example, during INSERT and UPDATE transactions.

То	See	
Create relationships between	Creating a Relationship Between	
database tables in a database diagram	<u>Tables</u>	
Ensure each value entered in a	Enforcing Referential Integrity	
foreign key column matches an	Between Tables	
existing value in the related primary		
key column		
Link a column in a table with another	Drawing a Single-Table Reflexive	
column in the same table	Relationship	
Create a many-to-many relationship	Mapping Many-to-Many	
	Relationships to a Database Diagram	
Change the name of a relationship	Renaming a Relationship	
Remove the relationship between	Deleting a Relationship	
two tables		
Disable a foreign key constraint	Disabling a Foreign Key Constraint	
	with INSERT and UPDATE	
	Statements and Disabling a Foreign	
	Key Constraint for Replication	

Indexes

You can use an index to gain fast access to specific information in a database table. An index is a structure that orders the values of one or more <u>columns</u> in a database table, for example the last name (lname) column of the employee table. If you were looking for a specific employee by his or her last name, the index would help you get that information faster than if you had to search all the <u>rows</u> in the table.

The index provides pointers to the data values stored in specified columns of the table, and then orders those pointers according to the sort order you specify. The database uses the index much as you use an index in a book: it searches the index to find a particular value and then follows the pointer to the row containing that value.

In database diagrams, you can create, edit, or delete each type of index in the <u>Indexes/Keys Property Page</u> for a selected table. An index is saved in the database when you save the table that it is attached to, or when you save the diagram in which that table appears. For details, see <u>Creating an Index</u>.

As a general rule, you should create an index on a table only if the data in the indexed columns will be queried frequently. Indexes take up disk space and slow the adding, deleting, and updating of rows. In most situations, the speed advantages of indexes for data retrieval greatly outweigh these disadvantages. However, if your application updates data very frequently or if you have disk space constraints, you might want to limit the number of indexes.

Before creating an index, you must determine what columns to use and what type of index to create. For more information, see:

- Index Columns
- <u>Types of Index</u>

Index Columns

You can create indexes based on a single column or on multiple columns in a database table. Multiple-column indexes enable you to distinguish between rows

in which one column may have the same value.

Indexes are also helpful if you often search or sort by two or more columns at a time. For example, if you often set criteria for last name and first name columns in the same query, it makes sense to create a multiple-column index on those two columns.

To determine the usefulness of an index:

- Examine the WHERE and JOIN clauses of your queries. Each column included in either clause is a possible candidate for an index.
- Experiment with the new index to examine its effect on the performance of running queries.
- Consider the number of indexes already created on your table. It is best to avoid a large number of indexes on a single table.
- Examine the definitions of the indexes already created on your table. It is best to avoid overlapping indexes that contain shared columns.
- Examine the number of unique data values in a column and compare that number with the number of rows in the table. The result is the selectivity of that column, which can help you decide if a column is a candidate for an index and, if so, what type of index.

Types of Index

Depending on the functionality of your database, you can create three types of indexes - unique, primary key, and clustered - in Database Designer.

Tip Although a unique index will help locate information, for the best performance results it is recommended that you use primary key or unique constraints instead. For more information about these constraints, see <u>Primary Key Constraints</u> and <u>Unique Constraints</u>.

Unique Index

A unique index is one in which no two rows are permitted to have the same index value.

Most databases prevent you from saving a table with a newly created unique index when there are duplicate key values in the existing data. Your database may also prevent the addition of new data that would create duplicate key values in the table. For example, if you create a unique index on the employee's last name (lname) in the employee table, then no two employees can share the same last name.

For more information about unique indexes, see <u>Creating a Unique Index</u>.

Primary Key Index

A database table often has a <u>column</u> or combination of columns whose value uniquely identifies each row in the table. This column is called the <u>primary key</u> of the table.

Defining a primary key for a table in a database diagram automatically creates a primary key index that is a specific type of unique index. This index requires each value in the primary key to be unique. It also permits fast access to data when you use the primary key index in queries. For more information about primary keys, see <u>Defining a Primary Key</u>.

Clustered Index

In a clustered index, the physical order of the rows in the table is the same as the logical (indexed) order of the key values. A table can contain only one clustered index.

If an index is not clustered, the physical order of the rows in the table does not match the logical order of the key values. A clustered index usually provides faster access to data than does a nonclustered index.

For more information about using a clustered index, see <u>Creating a Clustered</u> <u>Index</u>.

See Also

<u>Constraints</u>

Constraints

Constraints are business logic that your database server enforces for you. They limit the possible values that users can enter into specified <u>columns</u>, enforcing <u>referential integrity</u>. When you create constraints in the Database Designer, they conform to the ANSI standard for creating and altering <u>tables</u>.

The Database Designer accepts five types of constraints:

- <u>Check Constraints</u>
- Default Constraints
- <u>Unique Constraints</u>
- <u>Primary Key Constraints</u>
- Foreign Key Constraints

Check Constraints

A <u>check constraint</u> specifies the data values or formats that are acceptable in one or more columns in a table. For example, you can require the **zip** column of the authors table to allow only five-digit numeric entries.

You can define many check constraints for a table. You use the Tables property pages to create, modify, or delete each check constraint.

То	See
Attach a check constraint to a table to	Attaching a New Check Constraint to
specify the data values that are	<u>a Table or Column</u>
acceptable in one or more columns	
Create a constraint expression to	Defining a Check Constraint
check data for a condition	Expression
Change the constraint expression or	Modifying a Check Constraint
the options that enable or disable the	
constraint for specific conditions	
Apply constraints either to new data	Checking Existing Data When
only or to existing data as well	Creating a Check Constraint
Disable a check constraint when data	Disabling a Check Constraint with
is added to, updated in, or deleted	INSERT and UPDATE Statements
from a table	
Disable a check constraint when your	Disabling a Check Constraint for
table is replicated in another database	<u>Replication</u>
Remove the limitations on data	Deleting a Check Constraint
values in a column	

Default Constraints

A <u>default</u> constraint enables you to define the value that will be supplied for a column whenever a user fails to enter a value. For example, in a table with a column called payterms, you can instruct your database server to enter "???" or "fill in later" if the user leaves it blank.

In database diagrams, you define a default constraint as a property of a column in your table. You define this type of constraint for a column by specifying a default value inside a table in Standard view. Be sure to specify the constraint with the correct delimiters. For example, strings must be surrounded with single quotes.

For more information about defining default constraints, see <u>Setting Column</u> <u>Properties</u>.

See Also

Constraints

Unique Constraints

A <u>unique</u> constraint ensures no duplicate values are entered into specified columns that are not a table's <u>primary key</u>. For example, in the employee table in which the emp_id column is the primary key, you can define a unique constraint that requires entries in the Social Security number (SSN) column to be unique within the table.

In database diagrams, you use the Indexes/Keys property page to create, modify, or delete unique constraints.

То	See
Ensure no duplicate values are	Creating a Unique Constraint
entered in specific columns	
Change the columns that the	Modifying a Unique Constraint
constraint is attached to, change the	
constraint name, or set additional	
properties for the constraint	
Remove the requirement for	Deleting a Unique Constraint
uniqueness for values entered in the	
column	

Primary Key Constraints

A <u>primary key</u> constraint ensures no duplicate values are entered in particular <u>columns</u> and that NULL values are not entered in those columns. You can use primary key constraints to enforce uniqueness as well as <u>referential_integrity</u>. For example, the au_id column uniquely identifies each author stored in the authors table.

You create primary key constraints directly in a database diagram.

То	See
Enforce uniqueness for values	Defining a Primary Key
entered in specified columns	
Change the column order, index	Modifying a Primary Key
name, clustered option, or fill factor	
Copy column properties from a	Copying Column Properties to a
primary key column to a foreign key	<u>Foreign Key Column</u>
column to relate the two columns	
Remove the requirement for	Deleting a Primary Key Constraint
uniqueness for the values entered in a	
column	

Foreign Key Constraints

A <u>foreign_key</u> constraint works in conjunction with <u>primary key</u> or <u>unique constraints</u> to enforce referential integrity among specified tables. For example, you can place a foreign key constraint on the title_id column in the publishers table to ensure that a value entered in that column matches an existing value in the title_id column of the titles table.

In database diagrams, a foreign key constraint is automatically placed on specified columns when you create a relationship to a table from another table to which a primary key or unique constraint is attached. For more information about creating relationships, see <u>Creating a Relationship Between Tables</u>.

То	See
See which columns participate in the foreign key side of a relationship	<u>Viewing Foreign Key Attributes</u>
Change which columns are related to columns in the primary key table	<u>Modifying a Foreign Key</u>
Check existing data when creating a	Checking Existing Data when
relationship	Creating a Relationship
Disable a foreign key constraint	Disabling a Foreign Key Constraint
during INSERT and UPDATE	with INSERT and UPDATE
transactions	<u>Statements</u>
Disable a foreign key constraint	Disabling a Foreign Key Constraint
during replication of the table	for Replication
Remove the requirement to enforce	Deleting a Foreign Key Constraint
referential integrity between primary	
key columns and the related columns	
in another table	

Triggers

A trigger is a special kind of <u>stored procedure</u> that goes into effect when you modify data in a specified table using one or more data modification operations: UPDATE, INSERT, or DELETE. Triggers can query other tables and can include complex SQL statements. They are primarily useful for enforcing complex business rules or requirements. For example, you could control whether to allow an order to be inserted based on a customer's current account status.

Triggers are also useful for enforcing <u>referential integrity</u>, which preserves the defined relationships between tables when you add, update, or delete the rows in those tables. However, the best way to enforce referential integrity is to define primary key and foreign key constraints in the related tables. If you use database diagrams, you can create a relationship between tables to automatically create a foreign key constraint. For details, see <u>Table Relationships</u>.

Advantages of Using Triggers

Triggers are useful in these ways:

- Triggers are automatic: they are activated immediately after any modification to the table's data, such as a manual entry or an application action.
- Triggers can cascade changes through related tables in the database. For example, you can write a delete trigger on the title_id column of the titles table to cause a deletion of matching rows in other tables. The trigger uses the title_id column as a unique key to locate matching rows in the titleauthor, sales, and roysched tables.
- Triggers can enforce restrictions that are more complex than those defined with <u>check constraints</u>. Unlike check constraints, triggers can reference columns in other tables. For example, a trigger can roll back updates that attempt to apply a discount (stored in the discounts table)

to books (stored in the titles table) with a price of less than \$10.

For details about working with triggers, see the following topics:

То	See
Create new triggers	Creating a Trigger
View existing triggers	Viewing a Trigger
Change the name of a trigger	Modifying and Renaming a Trigger
Delete stored triggers	Deleting a Trigger

Stored Procedures

Stored procedures can make managing your database and displaying information about that database and its users much easier. Stored procedures are a precompiled collection of SQL statements and optional control-of-flow statements stored under a name and processed as a unit. Stored procedures are stored within a database; can be executed with one call from an application; and allow user-declared variables, conditional execution, and other powerful programming features.

Stored procedures can contain program flow, logic, and queries against the database. They can accept parameters, output parameters, return single or multiple result sets, and return values.

You can use stored procedures for any purpose for which you would use SQL statements, with these advantages:

- You can execute a series of SQL statements in a single stored procedure.
- You can reference other stored procedures from within your stored procedure, which can simplify a series of complex statements.
- The stored procedure is compiled on the server when it is created, so it executes faster than individual SQL statements.

The functionality of a stored procedure is dependent on the features offered by your database. For more details about what a stored procedure can accomplish for you, see <u>Stored Procedures</u>.

For details about working with stored procedures, see the following topics:

То	See
Create stored procedures to be executed from the database	Creating a Stored Procedure
Set execute permissions to allow access to the stored procedures by	Executing a Stored Procedure

specific users	
Use parameters in stored procedures	Specifying Parameters
View a stored procedure	Viewing a Stored Procedure
Delete stored procedures	Deleting a Stored Procedure
Run stored procedures against your	Executing a Stored Procedure
database	
Change the name of a stored	Modifying and Renaming a Stored
procedure	Procedure

User-Defined Functions

Microsoft SQL Server 2000 lets you create user-defined functions. Like any function, a user-defined function is a routine that returns a value. Based on what kind of value it returns, each user-defined function falls into one of three categories:

• Functions that return an updateable table of data

If a user-defined function contains a single SELECT statement and that statement is updateable, then the tabular result returned by the function is also updateable.

• Functions that return a non-updateable table of data

If a user-defined function contains more than one SELECT statement, or contains a SELECT statement that is not updateable, then the tabular result returned by that function is not updateable.

• Functions that return a scalar value

A user-defined function can return a scalar value.

If a function returns a table, you can use that function in the FROM clause of a query. For more information, see <u>Using Something Else In Place of a Table</u>. If a function returns a scalar value, you can use it in a query anywhere you would use a column name. For more information, see <u>Expressions in Queries</u>.

Large Database Projects

When you use the Visual Database Tools as part of a large software development effort, you can encounter several noteworthy situations:

- **Multiple People Designing a Single Database** Several users can connect to a database and use the Visual Database Tools to alter the database design. For information about coordinating the simultaneous activities of several users, see <u>Multiuser Database Design</u>.
- Evolutionary Development of a Deployed Database After a database is deployed, design changes can become necessary as users expand the set of tasks they want to perform with the data. For information about making changes to a deployed database, see <u>Issues of Database</u> <u>Evolution</u>.
- Multiple Versions of a Particular Database Large projects can include many databases. Even a project with a single database can have several copies of it a development database, a test database, and a deployed production database. During the lifetime of a deployed database application, changes and improvements are first made in the development database, then propagated to the test database, then propagated to the production database. For information about using Visual Database Tools to propagate design changes from one database to another, see Development, Test, and Production Databases.

Multiuser Database Design

You can work with database diagrams in a multiuser environment; that is, an environment in which more than one user at a time may make changes to a database diagram and the database. When you save a diagram, the Database Designer verifies that the database has not been modified since you last saved the diagram. If another user has made changes, you will be notified that the database has been modified. You may need to reconcile these changes, both in the database diagram and in the database itself.

То	See
Work with others on shared database	Multiuser Environments
diagrams	
Resolve changes made to a database	Reconciling a Database Diagram
diagram by multiple users	with a Modified Database

Multiuser Environments

A multiuser environment is one in which other users can connect and make changes to the same database that you are working with. As a result, several users might be working with the same <u>database objects</u> at the same time. Thus, a multiuser environment introduces the possibility of your <u>database diagrams</u> being affected by changes made by other users, and vice versa. Such changes could include changes to copies of your diagrams, other users' diagrams that share database objects with your diagrams, or the underlying database.

A key issue when working with databases in a multiuser environment is access permissions. The permissions you have for the database determine the extent of the work you can do with the database. For example, to make changes to objects in a database, you must have the appropriate write permissions for the database. For more information about permissions in your database, see <u>Managing</u> <u>Permissions</u>.

As one of multiple users, you may need to address any of the following issues when working with the Database Designer:

- <u>Ownership of Database Objects</u>
- <u>Diagrams Affected by Another User's Changes</u>
- Database Objects Deleted by Another User

Ownership of Database Objects

Each database object is owned by either a user or a role. If the owner is a role, the object is co-owned by every user belonging to that role.

Each object also has privileges associated with it. A privilege grants a particular authority to an object for either an individual user or for all users belonging to a particular role.

Depending on the privileges defined in the database, you can view objects that you do not own. For example, you can include on a database diagram an object owned by another user. When such an object appears on a diagram or in the Table Designer, it is labeled with the owner name as well as the object name.

If the database privileges let you view but not modify the database structure, you can use change scripts to design your modifications without transmitting them to the database. For more information, see <u>Saving a Change Script</u>.

For details about changing the owner of a table or view, see <u>Tables Property</u> <u>Page</u> and <u>Query Tab</u>, <u>Properties Window (View Designer</u>), respectively.

Diagrams Affected by Another User's Changes

In a multiuser environment, your database diagram can be affected by changes other users have saved to:

- Your diagram, which other users changed since you opened the diagram.
- Database objects shared between diagrams.
- The database.

For example, your diagram might contain a table that another user deleted or renamed. In such a case, your diagram will no longer reflect the current state of the database. When you attempt to save your database diagram or selected tables, the <u>Database Changes Detected dialog box</u> notifies you that the database has been updated since you opened your diagram.

This dialog box also displays a list of database objects that will be affected as a result of saving your diagram or selection. At this point, you can take one of these actions:

• Save your diagram or selection and update the database with all the changes in the list. This action will also affect other diagrams that share the same database objects.

For example, suppose you edit the au_id column in the titleauthors table on your diagram and another user's diagram contains the authors table which is related to the titleauthors table by the au_id column. Saving your diagram will affect the other user's diagram. Similarly, suppose that another user defined a <u>check constraint</u> for the qty column in the sales table. If you delete the qty column and save the sales table, the other user's check constraint will be affected.

• **Cancel the save action.** You can then close the diagram without

saving it. When you reopen the diagram, it will be in synch with the database.

• Save a list of the changes. You can save the list of database changes shown in the Database Changes Detected dialog box to a text file so that you can investigate the cause of other users' changes. For example, if another user edited a table that you marked for deletion, you may want to research whether the table should be deleted before updating the database.

See Also

Interactions Among Database Diagrams and Table Design Windows

Database Objects Deleted by Another User

In a multiuser environment, deleting database objects from the database can affect other users. If another user deletes a database object that appears in your database diagram, the effect of the deletion on your diagram depends on where the object was deleted.

Where the object was deleted	How your database diagram is affected
In another copy of your database diagram	The deleted object will be removed from your diagram next time you open it. If your diagram is open when the deletion occurs, it still shows the deleted item.
In a different database diagram	The deleted object will be removed from your diagram next time you open it. If your diagram is open when the deletion occurs, it still shows the deleted item.
In the database	If your diagram has no unsaved changes to that object, the object will be removed from your diagram the next time you open it. If your diagram contains unsaved changes to that object, the object will remain in your diagram. You can recreate it in the database by saving your diagram.

Note Recreating a deleted object creates a new definition of that object in the database; it does not restore the data that was deleted when the object was deleted.

For example, if the titles table is deleted in the database and you have unsaved changes to the titles table in your database diagram, then saving your diagram will create a new titles table in the database. Any data that existed in the titles table before it was deleted is not restored. For details about saving a diagram when the database has changed, see <u>Reconciling a Database Diagram with a Modified Database</u>.

Issues of Database Evolution

If you change the structure of a deployed database, you must take special care that your alteration is compatible with the existing data and database structure. You might need to take special steps when you make the following modifications:

- Adding a Constraint If you add a constraint, the database might already contain data that does not satisfy it. When you try to save the new constraint, the <u>Save Incomplete dialog box</u> informs you that the database server could not create the constraint. To force the database to accept the new constraint, you can clear the Check existing data on creation check box. For more information, see <u>Checking Existing Data</u> <u>When Creating a Check Constraint</u>.
- Adding a Relationship If you add a relationship, the database might already contain rows of the foreign-key table that do not have corresponding rows in the primary-key table. That is, the existing data might not satisfy referential integrity. When you try to save the new relationship, the <u>Save Incomplete dialog box</u> informs you that the database server could not save the revised foreign-key table. To force the database to accept the modification, you can clear the Check existing data on creation check box. For details, see <u>Checking Existing Data when Creating a Relationship</u>.
- **Modifying a Table Contributing to an Indexed View** If you modify a table that contributes to an SQL Server indexed view, the indexes on the view will be lost. For more information about recreating indexes, see <u>Rebuilding an Index</u>.

No matter how you alter the database design, you should retain a history of the alterations. One approach is to retain SQL scripts for all modifications that you ever make to your production database. For more information about scripts, see <u>Working with Scripts</u>.

Development, Test, and Production Databases

If you have two databases with identical structure, you can make changes in one database and propagate those changes to the other. For example, if you have a personal development database and a group-wide test database, you can modify the development database, then propagate those changes to the test database. To accomplish this:

• You can perform all the modifications in a single session with the development database, save an SQL script file of your session and later run the script on the test database. For more information, see <u>Saving a</u> <u>Change Script</u>.

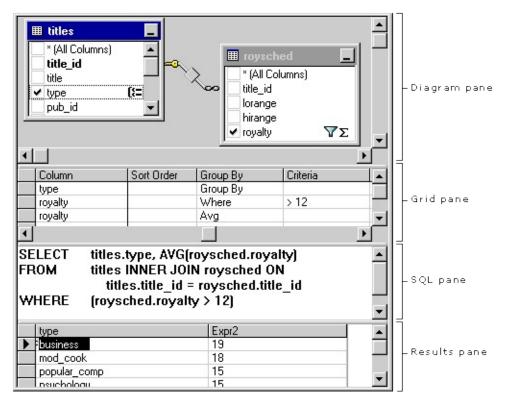
Database Queries and Visual Database Tools

After a database is designed and populated with data, you can write queries to retrieve information from the database. To help you write queries, the Visual Database Tools include the Query Designer. For more information, see the following topics:

- <u>Query and View Designer Layout</u>
- <u>Queries and Views</u>
- Comparison of Queries and Views
- Indexed Views
- <u>Supported Query Types</u>
- <u>Structure of Retrieval Queries</u>

Query and View Designer Layout

Both the Query Designer and View Designer consist of four panes: the Diagram pane, the Grid pane, the SQL pane, and the Results pane.



- **The Diagram pane** displays the tables and other table structuredobjects that you are querying. Each rectangle represents a table or tablestructured object and shows the available data columns as well as icons that indicate how each column is used in the query. Joins are indicated by lines between the rectangles. For more information, see <u>Diagram</u> <u>Pane</u>.
- **The Grid pane** contains a spreadsheet-like grid in which you specify options, such as which data columns to display, what rows to select, how to group rows, and so on. For more information, see <u>Grid Pane</u>.
- **The SQL pane** displays the SQL statement for the query or view. You can edit the SQL statement created by the Designer or you can enter

your own SQL statement. It is particularly useful for entering SQL statements that cannot be created using the Diagram and Grid panes, such as <u>Union queries</u>. For more information, see <u>SQL Pane</u>.

• **The Results pane** shows a grid with data retrieved by the query or view. In the Query Designer, the pane shows the results of the most recently executed <u>Select query</u>. You can modify the database by editing values in the cells of the grid, and you can add or delete rows. For more information, see <u>Results Pane</u>. In the View Designer, the results pane shows the contents of the view.

You can create a query or view by working in any of the panes: you can specify a column to display by choosing it in the Diagram pane, entering it into the Grid pane, or making it part of the SQL statement in the SQL pane. The Diagram, Grid, and SQL panes are synchronized — when you make a change in one pane, the other panes automatically reflect the change.

See Also

Diagram Pane | Grid Pane | Navigating in the Query Designer | Results Pane | SQL Pane

Diagram Pane

The Diagram pane presents a graphic display of the tables or table-structured objects you have selected from the data connection. It also shows any join relationships among them.

In the Diagram pane you can:

- Add or remove tables and table-structured objects and specify data columns for output.
- Specify columns for ordering the query.
- Specify that you want to group rows in the result set.
- Create or modify joins between tables and table-structured objects.

When you make a change in the Diagram pane, the Grid pane and SQL pane are updated to reflect your change. For example, if you select a column for output in a table or table-structured object window in the Diagram pane, the Query Designer adds the data column to the Grid pane and to the SQL statement in the SQL pane.

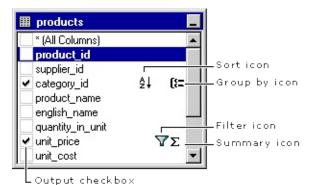
About the Diagram Pane

Each table or table-structured object appears as a separate window in the Diagram pane. The icon in the title bar of each rectangle indicates what type of object the rectangle represents, as illustrated in the following table.

Icon	Object type
	Table
	Query or View
-	Linked Table
	User-Defined Function

Ø	Subquery (in FROM clause)
-X ^B	Linked View

Each rectangle shows the data columns for the table or table-structured object. Check boxes and symbols appear next to the names of columns to indicate how the columns are being used in the query. ToolTips display information such as data type and size for columns.



each table or table-structured object.

The following table lists the check boxes and symbols used in the rectangle for

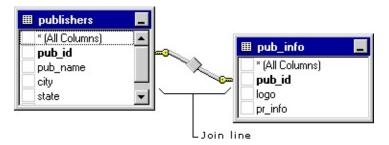
Check box or symbol	Description
~	Specifies whether a data column appears in the
+	query result set (Select query) or is used in an
0	Update, Insert From, Make Table, or Insert Into
×	query. Select the column to add it to the results. If (All Columns) is selected, all data columns appear in the output.
	The icon used with the check box changes according to the type of query you are creating. When creating a Delete query, you cannot select individual columns.
Ê↓	Indicates that the data column is being used to
₹↓	order the query results (is part of an ORDER BY clause). The icon appears as A-Z if the sort order is ascending or Z-A if sort order is descending.
[}=	Indicates that the data column is being used to

	create a grouped result set (is part of a GROUP BY clause) in an aggregate query.
A	Indicates that the data column is included in a search condition for the query (is part of a WHERE or HAVING clause).
Σ	Indicates that the contents of the data column are being summarized for output (are included in a SUM, AVG, or other aggregate function).

Note The Query Designer will not display data columns for a table or tablestructured object if you do not have sufficient access rights to it or if the database driver cannot return information about it. In such cases, the Query Designer displays only a title bar for the table or table-structured object.

Joined Tables on the Diagram Pane

If the query involves a join, a join line appears between the data columns involved in the join. If the joined data columns are not displayed (for example, the table or table-structured object window is minimized or the join involves an expression), the Query Designer places the join line in the title bar of the rectangle representing the table or table-structured object. The Query Designer displays one join line for each join condition.



The shape of the icon in the middle of the join line indicates how the tables or table-structured objects are joined. If the join clause uses an operator other than equal (=), the operator is displayed in the join line icon. The following table lists the icons that can be displayed in a join line.

Join line icon	Description
۶×	

	Inner join (created using equal sign).
Ň	Inner join based on the "greater than" operator.
	(The operator displayed in the join line icon
	reflects the operator used in the join.)
\bowtie	Outer join in which all rows from the table
	represented on the left will be included, even if
	they do not have matches in the related table.
	Outer join in which all rows from the table
	represented on the right will be included, even if
	they do not have matches in the related table.
<u>ک</u>	A full outer join in which all rows from both
~	tables will be included, even if they do not have
	matches in the related table.

Icons on the ends of the join line indicate the type of join. The following table lists the types of joins and the icons that can be displayed on the ends of the join line.

Icon on ends of join line	Description
	One-to-one join
	One-to-many join
	Query Designer cannot determine join type

See Also

Adding Tables | Creating a Query | Designing Queries | Grid Pane | Grouping Rows in Query Results | Removing Columns from Query Output | Removing Tables | Removing Joins | Results Pane | SQL Pane

Grid Pane

The Grid pane allows you to specify query options — such as which data columns to display, how to order the results, and what rows to select — by entering your choices into a spreadsheet-like grid. In the Grid pane you can specify:

- Columns to display and column name aliases.
- The table that a column belongs to.
- Expressions for calculated columns.
- The sort order for the query.
- Search conditions.
- Grouping criteria, including aggregate functions to use for summary reports.
- New values for Update or Insert Into queries.
- Target column names for Insert From queries.

Changes you make in the Grid pane are automatically reflected in the Diagram pane and SQL pane. Similarly, the Grid pane is updated automatically to reflect changes made in the other panes.

About the Grid Pane

The rows in the Grid pane display the data columns used in your query; columns in the Grid pane display query options.

_ Query	options	appear	ίn	grid	columns
---------	---------	--------	----	------	---------

Column	Output	Sort Type	Sort Order	Criteria	0r
employee_id	\checkmark				
last_name	\checkmark	Ascending	1		
first_name					
birth_date	\checkmark	Ascending	2	> "'01 Dec 26'"	
			1.10.00		

L Data columns appear as rows in the grid

The specific information that appears in the Grid pane depends on the type of query you are creating. If you are creating a Select query, the Grid pane contains different columns than if you are creating an Update query.

The following table lists the grid columns that can appear in the Grid pane.

Column	Query type	Description
Column	All	Displays either the name of a data column used for the query or the expression for a computed column. This column is locked so that it is always visible as you scroll horizontally.
Alias	Select, Insert From, Update, Make Table	Specifies either an alternate name for a column or the name you can use for a computed column.
Table		Specifies the name of the table or table- structured object for the associated data column. This column is blank for computed columns.
Output	Select, Insert From, Make Table	 Specifies whether a data column appears in the query output. Note If the database allows, you can use a data column for sort or search clauses without displaying it in the result set.
Sort Type	Select, Insert From	Specifies that the associated data column is used to sort the query results and whether the sort is ascending or descending.
Sort Order	Select, Insert From	Specifies the sort priority for data columns used to sort the result set. When you change the sort order for a data column, the sort order

		for all other columns is updated accordingly.
Group By	Select, Insert From, Make Table	Specifies that the associated data column is being used to create an aggregate query. This grid column appears only if you have chosen Group By from the Tools menu or have added a GROUP BY clause to the SQL pane.
		By default, the value of this column is set to Group By , and the column becomes part of the GROUP BY clause.
		When you move to a cell in this column and select an aggregate function to apply to the associated data column, by default the resulting expression is added as an output column for the result set.
Criteria	All	Specifies a search condition (filter) for the associated data column. Enter an operator (the default is "=") and the value to search for. Enclose text values in single quotation marks.
		If the associated data column is part of a GROUP BY clause, the expression you enter is used for a HAVING clause.
		If you enter values for more than one cell in the Criteria grid column, the resulting search conditions are automatically linked with a logical AND.
		To specify multiple search condition expressions for a single database column (for example, (fname > 'A') AND (fname < 'M'), add the data column to the Crid page
		'M'), add the data column to the Grid pane twice and enter separate values in the Criteria grid column for each instance of the data column.
Or	All	Specifies an additional search condition

		expression for the data column, linked to previous expressions with a logical OR. You can add more Or grid columns by pressing the TAB key in the rightmost Or column.
Append	Insert From	Specifies the name of the target data column for the associated data column. When you create an Insert From query, the Query Designer attempts to match the source to an appropriate target data column. If the Query Designer cannot choose a match, you must provide the column name.
New Value	Update, Insert Into	Specifies the value to place into the associated column. Enter a literal value or an expression.

See Also

<u>Creating Column Aliases | Designing Queries | Diagram Pane | Entering Search</u> <u>Values | Grouping Rows in Query Results | Results Pane | Specifying Search</u> <u>Conditions | SQL Pane</u>

SQL Pane

The SQL pane displays the SQL statement for the current query. As you build your query, the SQL pane is automatically updated and reformatted to be easy to read.

In the SQL pane you can:

- Create new queries by entering SQL statements.
- Modify the SQL statement created by the Query Designer based on settings you make in the Diagram and Grid panes.
- Enter statements that take advantage of features specific to the database you are using.

Note Be sure you know the rules for identifying database objects in the database you are using. For details about SQL Server, see <u>Query Designer</u> <u>Considerations for SQL Server Databases</u>.

Statements in the SQL Pane

You can edit the current query directly in the SQL pane. When you move to another pane, the Query Designer automatically formats your statement, and then changes the Diagram and Grid panes to match your statement.

Note You can enter optimizer hints for SQL statements, but the Query Designer might reformat them. For details about SQL Server, see <u>Query</u> <u>Designer Considerations for SQL Server Databases</u>.

If your statement cannot be represented in the Diagram and Grid panes, and if those panes are visible, the Query Designer displays an error and then offers you two choices:

• Return to the SQL pane and edit the statement.

• Discard your changes and revert to the most recent version of the SQL statement.

If you return to the SQL pane and continue editing the statement, the Query Designer dims the other panes to indicate that they no longer reflect the contents of the SQL pane.

You can also use the SQL pane to enter SQL statements that cannot be represented graphically in the Query Designer. In such cases, the Query Designer displays the same behavior as it does when it detects an error — it dims the Diagram and Grid panes to indicate that they do not represent the current statement. You can continue to edit the statement and execute it as you would any SQL statement. For details about unsupported query types in SQL Server, see <u>Query Designer Considerations for SQL Server Databases</u>.

Note If you enter an SQL statement, but then make further changes to the query by changing the Diagram and Grid panes, the Query Designer rebuilds and redisplays the SQL statement. In some cases, this action results in an SQL statement that is constructed differently from the one you originally entered (though it will always yield the same results). This difference is particularly likely when you are working with search conditions that involve several clauses linked with AND and OR.

See Also

<u>Creating Queries</u> | <u>Designing Queries</u> | <u>Diagram Pane</u> | <u>Executing a Query</u> | <u>Grid</u> <u>Pane</u> | <u>Results Pane</u> | <u>Using Expressions in a Query</u>

Results Pane

The Results pane shows the results of the most recently executed Select query. (The results of other query types are displayed in message boxes.)

In the Results pane you can:

- View the result set for the most recently executed Select query in a spreadsheet-like grid.
- Edit the values in individual columns in the result set, add new rows, and delete existing rows. For details, see <u>Editing Rows in the Results</u> <u>Pane</u>.

If you change the query definition (for example, add another output column to the query), the Query Designer dims the Results pane to indicate that it no longer reflects the current query. However, you can still navigate in the Results pane grid to edit, add, or delete rows.

Data in the Results Pane

When you execute a Select query, open a view, or open a table, the result set appears in the Results pane.

The Results pane uses these conventions:

- Columns containing no value display the word <NULL>.
- Columns containing binary data display the word <Binary>. You cannot edit the contents of these of columns.
- Columns containing long varchar-type data display up to 900 characters of data. If the data is longer, the cell displays <Long Text>.
- Columns format number, currency, time, and date information

according to the options set in the Regional Settings dialog box in the Microsoft® Windows® Control Panel.

See Also

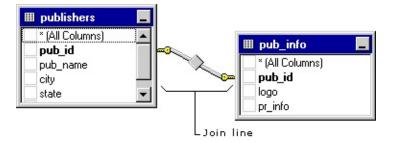
Adding New Rows in the Results Pane | Deleting Rows in the Results Pane | Determining When Query Results Can Be Updated | Editing Rows in the Results Pane | Removing Columns from Query Output | Reordering Output Columns

How the Query Designer Represents Joins

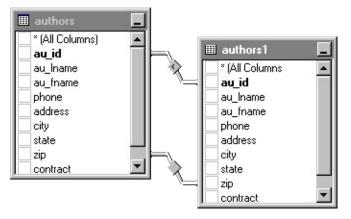
If tables are joined, the Query Designer represents the join graphically in the Diagram Pane and by using SQL syntax in the <u>SQL Pane</u>.

Diagram Pane

In the **Diagram** pane the Query Designer displays a join line between the data columns involved in the join. The Query Designer displays one join line for each <u>join condition</u>. For example, the following illustration shows a join line between two tables that are joined:



If tables are joined using more than one join condition, the Query Designer displays multiple join lines, as in the following example:



If the joined data columns are not displayed (for example, the rectangle representing the table or table-structured object is minimized or the join involves an <u>expression</u>), the Query Designer places the join line at the title bar of the rectangle representing the table or table-structured object.

The shape of the icon in the middle of the join line indicates how the tables or

table-structured objects are joined. If the join clause uses an operator other than equal (=), the operator appears in the join line icon. The following table lists the icons that appear in the join line.

Join line icon	Description	
X	Inner join (created using an equal sign).	
Ň	Inner join based on the "greater than" operator.	
Ď	Outer join in which all rows from the table represented on the left will be included, even if they do not have matches in the related table.	
	Outer join in which all rows from the table represented on the right will be included, even if they do not have matches in the related table.	
	Full outer join in which all rows from both tables will be included, even if they do not have matches in the related table.	

The symbols on the ends of the join line indicate the type of join. The following table lists the types of joins and the icons displayed on the ends of the join line.

Icon on ends of join line	Type of join
	<u>One-to-one</u> join.
	<u>One-to-many</u> join.
	Query Designer cannot determine the
	join type. This situation occurs most
	often when you have created a join
	manually.

SQL Pane

A join can be expressed in a number of ways in an SQL statement. The exact syntax depends on the database you are using and on how you have defined the join.

Syntax options for joining tables include:

• **JOIN qualifier for the FROM clause**. The keywords INNER and OUTER specify the join type. This syntax is standard for ANSI 92 SQL.

For example, if you join the publishers and pub_info tables based on the pub_id column in each table, the resulting SQL statement might look like this:

SELECT *
FROM publishers INNER JOIN pub_info ON
publishers.pub_id = pub_info.pub_id

If you create an <u>outer join</u>, the words LEFT OUTER or RIGHT OUTER appear in place of the word INNER.

• WHERE clause compares columns in both tables. A WHERE clause appears if the database does not support the JOIN syntax (or if you entered it yourself). If the join is created in the WHERE clause, both table names appear in the FROM clause.

For example, the following statement joins the publishers and pub_info tables.

SELECT *
FROM publishers, pub_info
WHERE publishers.pub_id = pub_info.pub_id

Note SQL Server databases support *= and =* syntax. For details, see <u>Query Designer Considerations for SQL Server</u>.

See Also

<u>Creating Outer Joins | Creating Self-Joins | Joining Tables Automatically |</u> <u>Joining Tables Manually | Modifying Join Operators | Types of Joins</u>

Queries and Views

A query is a specific request for the retrieval, creation, modification, or deletion of data in a database. A database accepts queries that are written in SQL, a language that is powerful but challenging. To capitalize on the power of SQL without enduring the challenge of writing it, you can use the Visual Database Tools. With the tools, you can create SQL queries without directly writing SQL.

SQL is also used to create views, which are specific subsets of database data. Because views and retrieval queries are defined with the same statement (the SQL SELECT statement), they are necessarily similar. But there are important distinctions between queries and views. For more information see <u>Comparison</u> <u>of Queries and Views</u>.

The SQL SELECT statement is the foundation of views and retrieval queries, but there are other kinds of queries based on other SQL statements. For more information about the types of queries, see <u>Supported Query Types</u>.

See Also

Comparison of Queries and Views

Comparison of Queries and Views

Because queries and views have so many similarities, it is easy to overlook their differences. This section briefly compares queries and views.

Storage Views are stored as part of a database design, but queries are not. As you design a database, you can include views in the design for the following reasons:

- **Some subsets of data are of interest to many users.** Because each view is stored in the database, it establishes a particular subset of data that can be used by any database user.
- Views can conceal base tables. You can disallow all user access to database tables, requiring users to manipulate data through views only. Such an approach can protect users and application programs from certain database modifications. For example, if you can create a view called "Current Month Sales." On the first of each month, you can modify the view definition accordingly. Without such a view, users would each month need to rewrite their queries to select Sales rows from the appropriate month.

Updating results The restrictions on updating result sets are different for views and queries. For more information, see <u>Rules for Updating Results</u>.

Sorting results You can sort any query result, but you can sort a view only if the view includes the TOP clause. For more information about the TOP clause, see Limiting Result Sets Using TOP and PERCENT.

Query plan generation A query plan is an internal strategy by which a database server tries to create result sets quickly. A database server can establish a query plan for a view as soon as the view is saved. For a query, however, a database server cannot establish a query plan until the query is actually run — that is, until the user explicitly demands the result set.

Parameterization You can create parameters for a query, but not for a view. For more information about parameterized queries, see <u>Creating General Purpose</u> <u>Queries</u>.

Encryption A view can be encrypted, but a query cannot. For more information see <u>Encrypting Views</u>.

See Also

Queries and Views

Indexed Views

SQL Server 2000 supports indexed views. Indexed views are views whose results are persisted in the database and indexed for fast access.

To learn how to create an indexed view, see <u>Creating Indexed Views</u>.

As with any other views, indexed views depend on base tables for their data. Such dependency means that if you change a base table contributing to an indexed view, the indexed view might become invalid. For example, renaming a column that contributes to a view invalidates the view. To prevent such problems, SQL Server supports creating views with "schema binding." Schema binding prohibits any table or column modification that would invalidate the view. Any indexed view you create with the View Designer automatically gets schema binding, because SQL Server requires that indexed views have schema binding.

Schema binding does not mean you cannot modify the view; it means you cannot modify the underlying tables or views in ways that would change the view's result set.

Using the Table Designer or Database Designer, you might attempt to modify a base table or column that contributes to a view defined with schema binding. If your attempted modification could invalidate the view, the Designer warns you and asks you whether you want to proceed. If you choose to proceed, these things happen:

- Your modifications to the base table occur.
- All views depending on the base table views are changed so that "schema binding" is removed. Thus, your subsequent changes to the base table will proceed without warning.
- If the dependent views were indexed, the indexes are deleted.

The Table Designer and Database Designer warn you before modifying a base table only if that base table contributes to a view with schema binding and if

your modification satisfies one or more of the following conditions:

- Your modification deletes the base table.
- Your modification renames the base table.
- Your modification recreates the base table.
- Your modification removes a column from the base table and the view includes that column.
- Your modification renames a column from the base table and the view includes that column.

See Also

Creating Indexed Views

Interactions Among Query and View Designer Windows

When you design a view, the View Designer retains your work in memory. That is, the View Designer does not transmit your work to the database until you explicitly save the view there. For each open Query Designer and View Designer Window, the Visual Database Tools retain a separate portion of memory to retain your work. Because each window has its own memory, the modifications you make in one window are not available to any other window.

Remember, you can use a view as part of a query or as part of another view. After you modify a view, your modifications will not affect queries or other views until you save the modified view to the database.

Note Database diagrams and Table Designer windows use a different strategy to retain your work in memory. For more information, see <u>Interactions Among</u> Database Diagrams and Table Design Windows.

See Also

Interactions Among Database Diagrams and Table Design Windows

Interaction Between the Results Pane and the Database

When you run a query or view, the database prepares the result set and returns the results to you. If the result set contains many rows, they are returned to you in batches. There are several details you should know about:

- **The database runs the query in its entirety** The query processing on the database server is not run in batches. Only the transmission of the query result from the server to your computer runs in batches.
- Unless you demand otherwise, only the first batch of results is returned After the first batch of rows is returned to you, you can continue to work in other panes of the query or view. If you want to view more rows of the result set, you can begin to scroll through the results pane to induce the database server to transmit more rows to your computer.

Some operations in the results pane require the entire result set If you attempt to navigate to the last row of the result set, or if you edit a value within the result set, you effectively induce the database server to return the entire result set to you. For an especially large result set, this can be time-consuming.

Supported Query Types

You can create the following types of queries in the Diagram and Grid panes (the graphical panes) of the Query Designer:

- **Select query** Retrieves data from one or more <u>tables</u> or <u>views</u>. This type of query creates an <u>SQL</u> SELECT statement.
- **Insert Into query** Creates a new row and inserts values into specified columns. This type of query creates an SQL INSERT INTO...VALUES statement.
- **Insert From query** Creates new rows by copying existing rows from one table into another, or into the same table as new rows. This type of query creates an SQL INSERT...SELECT statement.
- **Update query** Changes the values of individual columns in one or more existing rows in a table. This type of query creates an SQL UPDATE statement.
- **Delete query** Removes one or more rows from a table. This type of query creates an SQL DELETE statement.

Note A Delete query removes entire rows from the table. If you want to delete values from individual data <u>columns</u>, use an Update query.

• **Make Table query** Creates a new table and creates rows in it by copying the results of a query into it. This type of query creates an SQL SELECT...INTO statement.

In addition to the queries you can create using the graphical panes, you can enter any SQL statement into the SQL pane, such as <u>Union</u> queries.

When you create queries using SQL statements that cannot be represented in the

graphical panes, the Query Designer dims those panes to indicate that they do not reflect the query you are creating. However, the dimmed panes are still active and, in many cases, you can make changes to the query in those panes. If the changes you make result in a query that can be represented in the graphical panes, those panes are no longer dimmed.

For details about unsupported query types in SQL Server, see <u>Query Designer</u> <u>Considerations for SQL Server Databases</u>.

See Also

<u>Creating Delete Queries</u> | <u>Creating Insert From Queries</u> | <u>Creating Update</u> <u>Queries</u> | <u>Creating Insert Into Queries</u> | <u>Creating Make Table Queries</u>

Structure of Retrieval Queries

A retrieval query requests data from the database. At its simplest, a retrieval query merely fetches all data from a single table. But as you create more complex (and more typical) retrieval queries, you can assemble exactly the data you want in the following ways:

- Including or Excluding Rows
- Including or Excluding Columns
- <u>Combining Tables</u>
- Collapsing Groups of Rows
- <u>Using a Table Twice in One Query</u>
- <u>Using Something Else in Place of a Table</u>
- Sorting Rows

Including or Excluding Rows

To restrict the number of rows a <u>Select query</u> should return, you create <u>search conditions</u> or filter criteria. In SQL, search conditions appear in the WHERE clause of the statement, or if you are creating an <u>aggregate query</u>, in the HAVING clause.

Note You can also use search conditions to indicate which rows are affected by an Update, Insert Into, Insert From, Delete, or Make Table query.

When the query runs, the database engine examines and applies the search condition to each row in the tables you are searching. If the row meets the condition, it is included in the query. For example, a search condition that would find all the employees in a particular region might be:

region = 'UK'

To establish the criteria for including a row in a result, you can use multiple search conditions. For example, the following search criterion consists of two search conditions. The query includes a row in the result set only if that row satisfies both of the conditions.

region = 'UK' AND product_line = 'Housewares'

You can combine these conditions with AND or OR. The previous example uses AND. In contrast, the following criterion uses OR. The attendant query result will include any row that satisfies either or both of the search conditions:

```
region = 'UK' OR product_line = 'Housewares'
```

You can even combine search conditions on a single column. For example, the following criterion combines two conditions on the region column:

region = 'UK' OR region = 'US'

For details about combining search conditions, see the following topics:

• <u>Combining Search Conditions</u>

- <u>Specifying Multiple Search Conditions for One Column</u>
- Specifying Multiple Search Conditions for Multiple Columns
- <u>Combining Conditions when AND Has Precedence</u>
- <u>Combining Conditions when OR Has Precedence</u>

Predicates in Search Conditions

A search condition consists of one or more predicates, each specifying a single condition. If the search condition includes more than one predicate, the predicates are linked with a logical AND (to narrow the search) or OR (to broaden it). The following example shows how you can use multiple conditions when searching an employee table to find the employee (or employees) with the specified first and last names:

WHERE lname = 'Smith' AND fname = 'Jean'

A single predicate follows this format:

search_expression operator search_value

In most instances, *search_expression* is the name of a column to search. Similarly, the most common form of *search_value* is a literal value to search for, which can be either a string of characters or a number.

The following two examples show literal values. The first searches for all the employees who are in the United Kingdom, and the second searches for all employees with a specific job level:

WHERE region = 'UK'

WHERE job_lvl = 100

Both *search_expression* and *search_value* can consist of any (or any combination) of the following:

Literal A single text, numeric, date, or logical value. The following example uses a literal to find all rows for employees in the United Kingdom:
 WITEDE region = 'LW'

WHERE region = 'UK'

• **Column reference** The name of a column in one of the tables being searched. The following example searches a products table for all rows in which the value of the production cost is lower than the shipping cost:

WHERE prod_cost < ship_cost

- **Function** A reference to a function that the database back end can resolve to calculate a value for the search. The function can be a function defined by the database server or a user-defined function that returns a scalar value. The following example searches for orders placed today (the GETDATE() function returns the current date): WHERE order_date = **GETDATE()**
- NULL The following example searches an authors table for all authors who have a first name on file: WHERE au_fname IS NOT NULL
- Calculation The result of a calculation that can involve literals, column references, or other expressions. The following example searches a products table to find all rows in which the retail sales price is more than twice the production cost:
 WHERE sales_price > (prod_cost * 2)
- **Subquery** A <u>result set</u> generated by another query. The following example searches a products table to find all the products from Swedish suppliers. The <u>subquery</u> first searches the <u>suppliers</u> table to build a list of the suppliers located in that country. The second search then searches the products table, matching the product's supplier ID against the list created by the subquery: WHERE supplier_id IN

(SELECT supplier.supplier_id FROM supplier WHERE (supplier.country = 'Sweden'))

For more details about creating search conditions, see the following topics.

For information about	See
Specifying <u>search conditions</u> in the	Specifying Search Conditions
Query Designer	
Creating <u>expressions</u> that you can use	Using Expressions in Queries
in search conditions	
Using operators in search conditions	Comparison Operators, Logical
	<u>Operators</u> , and <u>Wildcard Characters</u>
Entering text, numbers, dates, or	Entering Search Values
logical values	
Finding rows that do not match a	Selecting Rows that Do Not Match a
value	Value
Removing duplicate rows from	Excluding Duplicate Rows
Select queries	
Applying multiple search conditions	Specifying Multiple Search
to the same data column	Conditions for One Column
Including several data columns as	Specifying Multiple Search
part of the search condition for a	Conditions for Multiple Columns
query	
Linking search conditions with AND	Combining Search Conditions
and OR operators	
Using <u>subqueries</u>	Creating Subqueries

Including or Excluding Columns

You can choose which columns appear in a query result. When choosing which columns to include, there are several things to keep in mind:

- You can include all of a table's columns For example, you can include everything about each employee. The resulting SQL looks like this:
 SELECT * FROM employee
- You can include exactly the columns you want For example, you can list the name of all the employees. The resulting SQL looks like this: SELECT fname, minit, lname FROM employee

The list of columns you include might not even include a column from every table in the query. This does not mean that the table does not contribute to the query. For an example of a query that uses a table without including any of that table's columns, see <u>Using a Table Twice in One Query</u>.

• You can include all columns from all tables For example, when you combine data from the sales and stores tables, you can include every column from either table in the result. The resulting SQL might look like this:

SELECT * FROM sales INNER JOIN stores ON sales.stor_id = stores.stor_id

• You can include derived columns That is, you can include columns that are not part of any database table of the query. For example, you can create a result set containing the job description and the average job level for each job. The resulting SQL might look like this: SELECT job_desc, (max_lvl + min_lvl) / 2

FROM jobs

You can use SQL syntax to define the derived column (as in the preceding sample query) or you can employ a user-defined function that returns a scalar value.

For more information on including columns in a query result, see <u>Adding</u> <u>Columns</u>. For more information on user-defined functions, see <u>User-Defined</u> <u>Functions</u>.

See Also

Structure of Retrieval Queries

Combining Tables

A query result can include data from multiple tables. To combine data from tables, you use the JOIN operation from SQL. The JOIN operation matches rows of one table with rows of another table, based on values in those rows. For example, you can join the table titles with the table publishers. Each row in the result set will describe a title, including information about that title's publisher, as shown in the following illustration:

			Publishe	rs table		
			pub_id	pub_name		city
			0736	New Moon Book	s	Boston
			1389	Algodata Infosys	stems	Berkeley
Titles tab		_	_			
title_id	title	pub_id				
BU1111	Cooking with Computers	1389				
BU1032	Executive's Database Guide, The	1389				
BU2075	You Can Combat Computer Stress!	0736				
Results ba	ased on joining Titles and Publish	ners 🖡	-			
title_id	title	pub_id	pub_nan	ne	city	
BU1111	Cooking with Computers	1389	Algodata	Infosystems	Berkel	ey
BU1032	Executive's Database Guide, The	1389	Algodata	Infosystems	Berkel	ey
BU2075	You Can Combat Computer Stress!	0736	New Moo	in Books	Bostor	1

When you use JOIN, you have several decisions to make. The following topics describe the types of joins and how to use them:

- <u>Types of Joins</u>
- JOIN Columns
- JOIN Comparison Operators
- JOIN Tables

Types of Joins

When you join tables, the type of join that you create affects the rows that appear in the <u>result set</u>. You can create the following types of joins:

• **Inner join** A join that displays only the rows that have a match in both joined tables. (This is the default type of join in the Query Designer.) For example, you can join the titles and publishers tables to create a result set that shows the publisher name for each title. In an inner join, titles for which you do not have publisher information are not included in the result set, nor are publishers with no titles. The resulting SQL for such a join might look like this:

SELECTtitle, pub_nameFROMtitles INNER JOINpublishersON titles.pub_id = publishers.pub_id

Note Columns containing NULL do not match any values when you are creating an inner join and are therefore excluded from the result set. Null values do not match other null values.

For more information on creating an inner join, see <u>Joining Tables</u> <u>Automatically</u>.

- **Outer join** A join that includes rows even if they do not have related rows in the joined table. You can create three variations of an outer join to specify the unmatched rows to be included:
 - Left outer join All rows from the first-named table (the "left" table, which appears leftmost in the JOIN clause) are included. Unmatched rows in the right table do not appear. For example, the following SQL statement illustrates a left outer join between the titles and publishers tables to include all titles, even those you do not have publisher information for: SELECT titles.title_id,

titles.title,

publishers.pub_name FROM titles LEFT OUTER JOIN publishers ON titles.pub_id = publishers.pub_id

• **Right outer join** All rows in the second-named table (the "right" table, which appears rightmost in the JOIN clause) are included. Unmatched rows in the left table are not included. For example, a right outer join between the titles and publishers tables will include all publishers, even those who have no titles in the titles table. The resulting SQL might look like this:

SELECT titles.title_id,

titles.title, publishers.pub_name FROM titles RIGHT OUTER JOIN publishers ON titles.pub_id = publishers.pub_id

• Full outer join All rows in all joined tables are included, whether they are matched or not. For example, a full outer join between titles and publishers shows all titles and all publishers, even those that have no match in the other table. SELECT titles.title_id, titles.title,

publishers.pub_name FROM titles FULL OUTER JOIN publishers ON titles.pub_id = publishers.pub_id

For more information on creating an outer join, see <u>Creating Outer</u> <u>Joins</u>.

• **Cross join** A join whose result set includes one row for each possible pairing of rows from the two tables. For example, authors CROSS

JOIN publishers yields a result set with one row for each possible author/publisher combination. The resulting SQL might look like this: SELECT * FROM authors CROSS JOIN publishers

For more information on creating a cross join, see <u>Removing Joins</u>.

See Also

<u>Creating Outer Joins | Creating Self-Joins | How the Query Designer Represents</u> Joins | Joining Tables Automatically | Joining Tables Manually | Modifying Join Operators | Querying Using Multiple Tables | Removing Joins

Join Columns

The JOIN operator matches rows by comparing values in one table with values in another. You decide which columns from each table should be matched. You have several choices:

Related Columns Typically, you join tables by matching values in columns for which a foreign-key relationship exists. For example, you can join discounts to stores by matching the values of stor_id in the respective tables. The resulting SQL might look like this:
 SELECT *
 FROM discounts INNER JOIN stores

ON stores.stor_id = discounts.stor_id

For more information on joining tables on related columns, see <u>Joining</u> <u>Tables Automatically</u>.

• **Unrelated Columns** You can also join tables by matching values in columns for which no foreign-key relationship exists. For example, you can join publishers to authors by matching the values of state in the respective tables. Such a join yields a result set in which each row describes an author-publisher pair located in the same state.

SELECT au_lname,

au_fname, pub_name, authors.state FROM authors INNER JOIN publishers ON authors.state = publishers.state

For more information on joining tables on unrelated columns, see <u>Joining Tables Manually</u>.

Note also that you use multiple columns to match rows from the joined tables.

For example, to find the author-publisher pairs in which the author and publisher are located in the same city, you use a join operation matching the respective state columns and the respective city columns of the two tables. You need to match both city and state because it is possible that different states could have like-named cities (e.g., Springfield, Illinois and Springfield, Massachusetts).

For more information on joining tables on multiple columns, see <u>Joining Tables</u> <u>on Multiple Columns</u>.

See Also

Structure of Retrieval Queries

Join Comparison Operators

The JOIN operator matches rows by comparing values in one table with values in another. You can decide exactly what constitutes a match. Your choices fall into two broad categories:

• Match on Equality Typically, you match rows when the respective column values are equal. For example, to create a result set in which each row contains a full description of each publisher, (that is, with columns from the publishers table and the pub_info table) you use a join matching rows where the values of pub_id in the respective tables are equal. The resulting SQL might look like this: SELECT *

FROM publishers INNER JOIN pub_info
ON publishers.pub_id
= pub info.pub id

• **Other** You can match rows using some test other than equality. For example, to find the employees and the jobs for which they are underqualified, you can join employee with jobs, matching rows in which the job's minimum required level exceeds the employee's job level. The resulting SQL might look like this:

```
SELECT
fname, minit, lname,
job_desc, job_lvl, min_lvl
FROM employee INNER JOIN jobs
ON employee.job_lvl
< jobs.min_lvl
```

For more information on comparison operators, see <u>Comparison Operators</u>.

See Also

Structure of Retrieval Queries

Join Tables

When combining data from multiple tables, you must decide what tables to use. There are several noteworthy considerations:

• **Combining Three or More Tables** Each JOIN operation combines two tables. However, you can use multiple JOIN operations within one query to assemble data from any number of tables. Because the result of each JOIN operation is effectively a table, you can use that result as an operand in a subsequent join operation. For example, to create a result set in which each row contains a book title, an author, and the percentage of that book's royalties the author receives, you must combine data from three tables: authors, titles, and titleauthor. The resulting SQL might look like this:

SELECT title, au_fname, au_lname, royaltyper FROM authors INNER JOIN titleauthor ON authors.au_id = titleauthor.au_id INNER JOIN titles ON titleauthor.title_id = titles.title id

• **Using a Table merely to join others** You can include a table in a join even if you do not want to include any of that table's columns in a result set. For example, to establish a result set in which each row describes a

```
title-store pair in which that store sells that title, you include columns
from two tables: titles, and stores. But you must use a third table, sales,
to determine which stores have sold which titles. The resulting SQL
might look like this:
SELECT title, stor_name
FROM titles
    INNER JOIN
    sales
    ON titles.title_id = sales.title_id
    INNER JOIN
    stores
    ON
    sales.stor_id = stores.stor_id
```

Notice that the sales table contributes no columns to the result set.

- Using a table twice in one query You can use the same table two (or more) times within a single query. For more information, see <u>Using a</u> <u>Table Twice in One Query</u>.
- Using something else in place of a table In place of a table, you can use a query, a view, or a user-defined function that returns a table. For more information, see <u>Using Something Else in Place of a Table</u>.

For more information on adding tables to a query, see <u>Adding Tables</u>.

See Also

Structure of Retrieval Queries

Collapsing Groups of Rows

You can create a query result in which each result row corresponds to an entire group of rows from the original data. When collapsing rows, there are several things to keep in mind:

• You can eliminate duplicate rows Some queries can create result sets in which multiple identical rows appear. For example, you can create a result set in which each row contains the city and state name of a city containing an author – but if a city contains several authors, there will be several identical rows. The resulting SQL might look like this: SELECT city, state FROM authors

The result set generated by the preceding query is not very useful. If a city contains four authors, the result set will include four identical rows. Since the result set does not include any columns other than city and state, there is no way to distinguish the identical rows from each other. One way to avoid such duplicate rows is to include additional columns that can make the rows different. For example, if you include author name, each row will be different (provided no two like-named authors live within any one city). The resulting SQL might look like this:

SELECT city, state, fname, minit, lname FROM authors

Of course, the preceding query eliminates the symptom, but does not really solve the problem. That is, the result set has no duplicates, but it is no longer a result set about cities. To eliminate duplicates in the original result set, and still have each row describe a city, you can create a query returning only distinct rows. The resulting SQL might look like this:

SELECT DISTINCT city, state FROM authors

For details about eliminating duplicates, see <u>Excluding Duplicate Rows</u>.

You can calculate on groups of rows That is, you can summarize information in groups of rows. For example, you can create a result set in which each row contains the city and state name of a city containing an author, plus a count of the number of authors contained in that city. The resulting SQL might look like this:
 SELECT city, state, COUNT(*)
 FROM authors
 GROUP BY city, state

For details about calculating on groups of rows, see <u>Aggregate</u> <u>Functions</u> and <u>Querying on Groups of Rows</u>.

• You can use selection criteria to include groups of rows For example, you can create a result set in which each row contains the city and state name of a city containing several authors, plus a count of the number of authors contained in that city. The resulting SQL might look like this:

SELECT city, state, COUNT(*) FROM authors GROUP BY city, state HAVING COUNT(*) > 1

For details about applying selection criteria on groups of rows, see <u>Specifying Conditions for Groups</u> and <u>Using HAVING and WHERE</u> <u>Clauses in the Same Query</u>.

See Also

Structure of Retrieval Queries

Using a Table Twice in One Query

You can use the same table two (or more) times within a single query. There are several situations in which you do this.

• Creating a self-join with a reflexive relationship You can join a table to itself using a reflexive relationship — a relationship in which the referring foreign-key columns and the referred-to primary-key columns are in the same table. For example, suppose the employee table contains an additional column, employee.manager_emp_id, and that a foreign key exists from manager_emp_id to employee.emp_id. Within each row of the employee table, the manager_emp_id column indicates the employee's boss. More precisely, it indicates the employee's boss's emp_id.

By joining the table to itself using this reflexive relationship, you can establish a result set in which each row contains a boss's name and the name of one of that boss's employees. The resulting SQL might look like this:

SELECT boss.lname, boss.fname, employee.lname, employee.fname FROM employee INNER JOIN employee boss ON employee.manager_emp_id = boss.emp_id

For more information about creating joins using reflexive relationships, see <u>Creating Self-Joins Automatically</u>.

• **Creating a self-join without a reflexive relationship** You can join a table to itself without using a reflexive relationship. For example, you

```
can establish a result set in which each row describes an employee and a
potential mentor for that employee. (A potential mentor is an employee
with a higher job level.) The resulting SQL might look like this:
SELECT
employee.fname,
employee.lname,
mentor.fname,
mentor.lname
FROM
employee
INNER JOIN
employee mentor
ON employee.job_lvl
< mentor.job_lvl
```

Notice that the join uses a condition other than equality. For more information about joining tables using conditions other than equality, see <u>Join Comparison Operators</u>.

For more information about creating self-joins using unrelated columns, see <u>Creating Self-Joins Manually</u>.

• Using a table twice without a self-join Even without a self join, you can use the same table twice (or more) in a query. For example, you can establish a result set containing the other books by the author or authors of your favorite book. In this case, you use the titleauthors table twice — once to find the authors of your favorite book (*Is Anger the Enemy?*), and once to find the other books by those authors. The resulting SQL might look like this:

SELECT other title.title

FROM

titles favorite_title

INNER JOIN

titleauthor favorite_titleauthor

```
ON favorite_title.title_id
```

= favorite_titleauthor.title_id **INNER JOIN** authors ON favorite_titleauthor.au_id = authors.au id **INNER JOIN** titleauthor other titleauthor ON authors.au id = other titleauthor.au id **INNER JOIN** titles other title ON other titleauthor.title id = other_title.title_id WHERE favorite title.title = 'Is Anger the Enemy?' AND favorite title.title <> other title.title

Note To distinguish between the multiple uses of any one table, the preceding query uses the following aliases: favorite_title, favorite_titleauthor, other_titleauthor, and other_title. For more information about aliases, see <u>Creating Table Aliases</u>.

See Also

Structure of Retrieval Queries | Drawing a Reflexive Relationship

Using Something Else in Place of a Table

Whenever you write a retrieval query, you articulate what columns you want, what rows you want, and where the query processor should find the original data. Typically, this original data consists of a table or several tables joined together. But the original data can come from sources other than tables. In fact, it can come from views, queries, or user-defined functions that return a table.

More precisely, the original data can come from any joined combination of tables, views, queries, and user-defined functions that return tables.

Using a View in Place of a Table

You can select rows from a view. For example, suppose the database includes a view called "ExpensiveBooks," in which each row describes a title whose price exceeds 19.99. The view definition might look like this:

```
SELECT *
FROM titles
WHERE price > 19.99
```

You can select the expensive psychology books merely by selecting the psychology books from the ExpensiveBooks view. The resulting SQL might look like this:

```
SELECT *
FROM ExpensiveBooks
WHERE type = 'psychology'
```

Similarly, a view can participate in a JOIN operation. For example, you can find the sales of expensive books merely by joining the sales table to the ExpensiveBooks view. The resulting SQL might look like this:

SELECT * FROM sales INNER JOIN

```
ExpensiveBooks
ON sales.title_id
= ExpensiveBooks.title_id
```

For more information about adding a view to a query, see <u>Adding Tables</u>.

Using a Query in Place of a Table

You can select rows from a query. For example, suppose you have already written a query retrieving titles and identifiers of the coauthored books — the books with more than one author. The SQL might look like this:

```
SELECT

titles.title_id, title, type

FROM

titleauthor

INNER JOIN

titles

ON titleauthor.title_id

= titles.title_id

GROUP BY

titles.title_id, title, type

HAVING COUNT(*) > 1
```

You can then write another query that builds on this result. For example, you can write a query that retrieves the coauthored psychology books. To write this new query, you can use the existing query as the source of the new query's data. The resulting SQL might look like this:

```
SELECT
title
FROM
(
SELECT
titles.title_id,
```

```
title,
    type
  FROM
    titleauthor
      INNER JOIN
       titles
      ON titleauthor.title_id
      = titles.title_id
  GROUP BY
    titles.title_id,
    title,
    type
  HAVING COUNT(*) > 1
  )
  co_authored_books
WHERE
          type = 'psychology'
```

The emphasized text shows the existing query used as the source of the new query's data. Note that the new query uses an alias ("co_authored_books") for the existing query. For more information about aliases, see <u>Creating Table</u> <u>Aliases</u> and <u>Creating Column Aliases</u>.

Similarly, a query can participate in a JOIN operation. For example, you can find the sales of expensive coauthored books merely by joining the ExpensiveBooks view to the query retrieving the coauthored books. The resulting SQL might look like this:

```
SELECT
ExpensiveBooks.title
FROM
ExpensiveBooks
INNER JOIN
(
SELECT
titles.title_id,
```

```
title,
type
FROM
titleauthor
INNER JOIN
titles
ON titleauthor.title_id
= titles.title_id
GROUP BY
titles.title_id,
title,
type
HAVING COUNT(*) > 1
)
```

For more information about adding a query to a query, see <u>Adding Tables</u>.

Using a User-Defined Function in Place of a Table

In SQL Server 2000 or higher, you can create a user-defined function that returns a table. Such functions are useful for performing complex or procedural logic.

For example, suppose the employee table contains an additional column, employee.manager_emp_id, and that a foreign key exists from manager_emp_id to employee.emp_id. Within each row of the employee table, the manager_emp_id column indicates the employee's boss. More precisely, it indicates the employee's boss's emp_id. You can create a user-defined function that returns a table containing one row for each employee working within a particular high-level manager's organizational hierarchy. You might call the function fn_GetWholeTeam, and design it to take an input variable — the emp_id of the manager whose team you want to retrieve.

You can write a query that uses the fn_GetWholeTeam function as a source of data. The resulting SQL might look like this:

SELECT *

FROM fn_GetWholeTeam ('VPA30890F')

("VPA30890F" is the emp_id of the manager whose organization you want to retrieve.) For more information about adding a user-defined function to a query, see <u>Adding Tables</u>.

See Also

<u>User-Defined Functions</u> | <u>Queries and Views</u>

Sorting Rows

You can order the rows in a query result. That is, you can name a particular column or set of columns whose values determine the order of rows in the result set. There are several ways in which you can use ordering:

 You can arrange rows in ascending or descending order By default, SQL uses order-by columns to arrangesrows in ascending order. For example, to arrange the book titles by ascending price, simply sort the rows by the price column. The resulting SQL might look like this: SELECT * FROM titles ORDER BY price

On the other hand, if you want to arrange the titles with the more expensive books first, you can explicitly specify a highest-first ordering. That is, you indicate that the result rows should be arranged by descending values of the price column. The resulting SQL might look like this:

SELECT * FROM titles ORDER BY price DESC

- You can sort by multiple columns For example, you can create a result set with one row for each author, ordering first by state and then by city. The resulting SQL might look like this:
 SELECT *
 FROM authors
 ORDER BY state, city
- You can sort by columns not appearing in the result set For example, you can create a result set with the most expensive titles first, even though the prices do not appear. The resulting SQL might look like this:

SELECT title_id, title FROM titles ORDER BY price DESC

• You can sort by derived columns For example, you can create a result set in which each row contains a book title — with the books that pay the highest royalty per copy appearing first. The resulting SQL might look like this: SELECT title, price * royalty / 100 as royalty_per_unit

FROM titles ORDER BY royalty_per_unit DESC

(The formula for calculating the royalty that each book earns per copy is emphasized.)

To calculate a derived column, you can use SQL syntax, as in the preceding example, or you can use a user-defined function that returns a scalar value. For more information about user-defined functions, see <u>User-Defined Functions</u>.

 You can sort grouped rows For example, you can create a result set in which each row describes a city, plus the number of authors in that city — with the cities containing many authors appearing first. The resulting SQL might look like this: SELECT city, state, COUNT(*) FROM authors GROUP BY city, state ORDER BY COUNT(*) DESC, state

(Notice that the query uses state as a secondary sort column. Thus, if two states have the same number of authors, those states will appear in alphabetical order.)

• You can sort using international data That is, you can sort a column using collating conventions that differ from the default conventions for that column. For example, you can write a query that retrieves all the book titles by the Icelandic novelist Halldor Laxness. To display the

```
titles in alphabetical order, you use an Icelandic collating sequence for
the title column. The resulting SQL might look like this:
SELECT title
FROM
  authors
  INNER JOIN
     titleauthor
     ON authors.au id
     = titleauthor.au id
     INNER JOIN
       titles
       ON titleauthor.title_id
       = titles.title_id
WHERE
   au fname = 'Halldor' AND
   au lname = 'Laxness'
ORDER BY
   title COLLATE SQL_Icelandic_Pref_CP1_CI_AS
```

For more information about sorting result rows, see Ordering Query Results.

Expressions in Queries

You can use an <u>expression</u> anywhere in a query where you can use a column name. Expressions can calculate values to display, be part of <u>search conditions</u>, or combine the contents of data columns. An expression can consist of a mathematical calculation or a string, and can involve any combination of column names, literals, operators, or functions.

Examples of the use of expressions in a query include:

- In a products table, displaying a discounted price that is calculated by taking 10% off the retail price.
- Displaying only the first three digits the area code of a phone number.
- Displaying employee names in the format *last_name*, *first_name*.
- Joining two tables, an orders table and a products table, then sorting the query on the total price (order quantity times product price).
- In an orders table, copying and then deleting all orders that were shipped more than one year ago.

For more information about creating and using expressions, refer to the topics listed in the following table.

For information about	See	
Creating expressions	Creating an Expression	
Entering expressions into a query	Using Expressions in a Query	
Creating summaries of data	Summarizing and Grouping	
Using operators in expressions	Operators for Expressions	
Using functions in expressions	Functions for Expressions	

Using predefined variables in	Predefined Variables for Expressions
expressions	

Parameter Queries

In some cases you want to create a query that you can use many times, but with a different value each time. For example, you might frequently query a titles table to find all the books written by one author. You could run the same query for each request, except that the author's ID or name would be different each time.

To create a query that can have different values at different times, you use parameters in the query. A parameter is a placeholder for a value that is supplied when the query runs. An SQL statement with a parameter might look like the following, where "?" represents the parameter for the author's state:

SELECT au_lname, au_fname FROM state WHERE state = ?

Where You Can Use Parameters

You can use parameters as placeholders for literal values — for either text or numeric values. Most commonly, parameters are used as placeholders in <u>search conditions</u> for individual rows or for groups (that is, in the WHERE or HAVING clauses of an SQL statement).

Some databases allow you to use parameters as placeholders in <u>expressions</u>. For example, you might want to calculate discounted prices by supplying a different discount value each time you run a query. To do so, you could specify the following expression:

```
(price * ?)
```

For details about where you can use parameters, see Parameters.

Specifying Unnamed and Named Parameters

You can specify two types of parameters: unnamed and named. An unnamed parameter is a question mark (?) that you put anywhere in the query that you

want to prompt for or substitute a literal value. For example, if you use an unnamed parameter to search for an author's id in a titles table, the resulting statement in the <u>SQL pane</u> might look like this:

SELECT au_lname, au_fname FROM state WHERE state = ?

When you run the query in the Query Designer, the <u>Define Query Parameters</u> dialog box appears with "?" as the name of the parameter, as in the following illustration.

Parameter Name	Parameter Value
?	
	OK Cancel

Alternatively, you can assign a name to a parameter. Named parameters are particularly useful if you have multiple parameters in a query. For example, if you use named parameters to search for an author's first and last names in an authors table, the resulting statement in the SQL pane might look like this:

```
SELECT au_id
FROM authors
WHERE au_fname = %first name% AND
au lname = %last name%
```

When you run the query in the Query Designer, the <u>Define Query Parameters</u> dialog box appears with a list of named parameters, as in the following illustration.

Parameter Name	Parameter Value
first name	
last name	

See Also

<u>Creating a Query with Named Parameters | Creating a Query with Unnamed</u> <u>Parameters | Creating General Purpose Queries | Specifying Parameter Marker</u> <u>Characters | Supported Query Types</u>

Summary and Grouping Behavior in the Query Designer

When you create <u>aggregate queries</u>, certain logical principles apply. For example, you cannot display the contents of individual rows in a summary query. The Query Designer helps you comply with these principles in the way the <u>Diagram</u> and <u>Grid</u> panes behave.

By understanding the principles of aggregate queries and the Query Designer's behavior, you can create logically correct aggregate queries. The overriding principle is that aggregate queries can result only in summary information. Thus, most of the principles that follow describe the ways that you can reference individual data columns within an aggregate query.

- <u>Referencing Columns for Output and Sorting</u>
- <u>Referencing Columns in Search Conditions</u>
- Working with Columns in Aggregate Queries

Referencing Columns for Output and Sorting

The following principles describe how you can reference columns in an aggregate query for output and for sorting:

- If you include an aggregate function anywhere in a query, the query is considered an aggregate query. (This principle does not necessarily apply to <u>subqueries</u>, which can include nonaggregate information.)
- You cannot display the contents of individual rows in an aggregate query; you can display only summary data. As a consequence, all columns marked for output must also be assigned to either an <u>aggregate function</u> or to the GROUP BY clause.
- Columns used in aggregate functions cannot appear in the GROUP BY

clause.

Referencing Columns in Search Conditions

The following principles describe how you can reference columns in an aggregate query in search conditions. In general, you can use a column in a search condition to filter the rows that should be summarized (a WHERE clause) or to determine which grouped results appear in the final output (a HAVING clause).

- Individual data columns can appear in either the WHERE or HAVING clause, depending on how they are used elsewhere in the query.
- WHERE clauses are used to select a subset of rows for summarizing and grouping and are thus applied before any grouping is done. Therefore, you can use a data column in a WHERE clause even if it is not part of the GROUP BY clause or contained in an aggregate function. For example, the following statement selects all titles that cost more than \$10.00 and averages the price: SELECT AVG(price) FROM titles
 WHERE price > 10
- If you create a <u>search condition</u> that involves a column also used in a GROUP BY clause or aggregate function, the search condition can appear as either a WHERE clause or a HAVING clause you can decide which when you create the condition. For example, the following statement creates an average price for the titles for each publisher, then displays the average for the publishers in which the average price is greater than \$10.00:

```
SELECT pub_id, AVG(price)
FROM titles
GROUP BY pub_id
HAVING (AVG(price) > 10)
```

• If you use an aggregate function in a search condition, the condition involves a summary and must therefore be part of the HAVING clause.

Working with Columns in Aggregate Queries

When you create aggregate queries using the <u>Diagram</u> and <u>Grid</u> panes, the Query Designer makes certain assumptions so that your query adheres to the principles outlined earlier. For example, if you are creating an aggregate query and mark a data column for output, the Query Designer automatically makes the column part of the GROUP BY clause so that you do not inadvertently attempt to display the contents of an individual row in a summary.

Even though the Query Designer works to help prevent logical errors, it is possible to create aggregate queries that will not execute. Therefore, be sure that you understand the principles listed earlier in order to avoid errors when you create queries.

The Query Designer uses the following guidelines for working with columns:

• When you choose the Group By option or add an aggregate function to a query, all columns marked for output or used for sorting are automatically added to the GROUP BY clause. Columns are not automatically added to the GROUP BY clause if they are already part of an aggregate function.

If you do not want a particular column to be part of the GROUP BY clause, you must manually change it by selecting a different option in the Group By column of the Grid pane. (However, the Query Designer will not prevent you from choosing an option that can result in a query that will not run.)

• If you manually add a query output column to an aggregate function in either the Grid or SQL pane, the Query Designer does not automatically remove other output columns from the query. Therefore, you must remove the remaining columns from the query output or make them part of the GROUP BY clause or of an aggregate function.

When you enter a search condition into the Criteria column of the Grid pane, the Query Designer follows these rules:

- If the **Group By** column of the grid is not displayed (because you have not yet specified an aggregate query), the search condition is placed into the WHERE clause.
- If you are already in an aggregate query and have selected the option **Where** in the **Group By** column, the search condition is placed into the WHERE clause.
- If the **Group By** column contains any value other than **Where**, the search condition is placed in the HAVING clause.

See Also

Counting Rows in a Table | Grouping Rows in Query Results | Querying on Groups of Rows | Specifying Conditions for Groups | Summarizing and Grouping | Summarizing Values for All Rows in a Table | Summarizing Values Using Custom Expressions | Using HAVING and WHERE Clauses in the Same Query

Using the Query Designer with International Data

You can use the Query Designer with data in any language and in any version of the Windows® operating system. The following guidelines outline the differences you will notice and provide information about managing data in international applications.

Note For additional information about ANSI to OEM character conversion in SQL Server, see <u>Query Designer Considerations</u>.

Localized Information in the Grid and SQL Panes

If you are using the Grid pane to create you query, you can enter information in the format that corresponds to the Windows Regional Settings for you computer. For example, if you are searching for data, you can enter the data in the Criteria columns using whatever format you are accustomed to using, with these exceptions:

- Long data formats are not supported.
- Currency symbols should not be entered in the Grid pane. However, the correct symbol will appear in the Results pane.

Note You may enter the \$ currency symbol, but the Results pane will return the currency symbol that corresponds to the Windows Regional Settings for your computer.

• Unary minus always appears on the left side (for example, -1) regardless of the Regional Settings options.

In contrast, data and keywords in the SQL pane must always be in ANSI (U.S.) format. For example, as the Query Designer builds a query, it inserts the ANSI form of all SQL keywords such as SELECT and FROM. If you add elements to the statement in the SQL pane, be sure to use the ANSI standard form for the elements.

When you enter data using local-specific format in the Grid pane, the Query

Designer automatically translates it to ANSI format in the SQL pane. For example, if your Regional Settings are set to Standard German, you can enter data in the Grid pane in a format such as "31.12.96." However, the date will appear in the SQL pane in ANSI datetime format as { ts '1996-12-31 00:00:00' }. If you enter data directly in the SQL pane, you must enter it in ANSI format.

Sort Order

The sort order of data in your query is determined by the database. Options that you set in the Windows Regional Settings dialog box do not affect sort order for queries. Within any particular query, however, you can request that rows be returned in a particular order. For more information, see <u>Sorting Rows</u>.

Using Double-Byte Characters

You can enter <u>DBCS</u> characters for literals and for database object names such as table and view names or <u>aliases</u>. You can also use DBCS characters for parameter names and parameter marker characters. However, you cannot use DBCS characters in <u>SQL</u> language elements such as function names or SQL keywords.

Developing Database Structure

The Database Designer provides tools to help you create and maintain the structure of the database. The Table Designer devotes an entire window to the design of an individual table. For more information about database and table structure, see <u>Database Development and the Visual Database Tools</u>.

For details about the specific steps in developing database structure, see the following topics:

- <u>Working with Databases</u>
- <u>Working with Tables</u>
- <u>Working with Columns</u>
- Working with Keys
- <u>Working with Relationships</u>
- <u>Working with Indexes</u>
- Working with Constraints
- <u>Working with Scripts</u>

Working with Databases

The Database Designer provides tools to help you change your database. You can control when and how changes to a database made in a database diagram are saved. You control these changes by noting which objects have changed and which remain unchanged in the database diagram, by saving changes only to selected tables, and by discarding unwanted changes. You can also use SQL change scripts to track, discard, postpone, and apply unsaved changes.

То	See
Save changes made on a database	Saving an Entire Database Diagram
diagram	
Save changes to a selected table on a	Saving Selected Tables on a Diagram
diagram	
Save your work in table designer	Saving Your Work in Table Designer
Return to a previous version of a	Discarding Modifications
database diagram without saving	
changes	
Ensure that database objects have a	Uniquely Naming Database Objects
unique name	

Saving Selected Tables on a Diagram

You can save a specific table or a set of tables if you do not want to save all the changes you made in a database diagram.

To save selected tables

- 1. In your database diagram, select the tables you want to save.
- 2. Right-click one of the selected tables, and choose **Save Selection**.
- The Save dialog box displays the list of tables that will be updated in the database when you save your selection.
 Choose Save Text File if you want to save the list of tables in a text file in the project directory before continuing. When a message box displays the name of the saved text file, choose OK.
- 4. In the **Save** dialog box, confirm the list of tables and choose **Yes** to save these tables.

Note The list of tables may contain tables in addition to those selected. For example, if you change the data type of a column that participates in a relationship with another table, both tables will be included in this list.

See Also

Saving a Change Script

Saving an Entire Database Diagram

By saving a database diagram, you can save all the changes you made to it, including any changes you made to the tables, columns and other database objects.

To save the database diagram

- 1. Right-click inside the database diagram, then click **Save**.
- If this is a new diagram that you have never before saved, the Save As dialog box appears. Type a name for the diagram.
 If you made changes to tables in an existing diagram, the Save dialog box appears and displays a list of changes that will be saved in the database when you save your diagram.
- 3. Click **Yes** to update the database to match your diagram.

See Also

Saving a Change Script

Saving Your Work in Table Designer

You can save all the changes you made to a table in the Table Designer.

To save a table

1. Click the **Save** button on the toolbar.

If you made changes to the table, the **Save** dialog box appears and displays a list of changes that will be saved in the database when you save your diagram.

Note The list of tables may contain tables other than the table you modified. For example, if you change the data type of a column that participates in a relationship with another table, both tables will be included in this list.

2. Click **Yes** to update the database.

See Also

Saving a Change Script

Discarding Changes Made in Database Designer or Table Designer

You can discard changes in your database diagram that you do not want to save in the database.

To discard pending modifications

- 1. Close your database diagram or table design window.
- 2. A message prompts you to save your changes. Choose **No**.

Caution If you have other open database diagrams or table design windows connected to the same database, the modifications that you discard in this manner remain in your local memory. To remove these pending modifications from memory, you must close all database diagrams and table design windows connected to the same database. For more information, see Interactions Among Database Diagrams and Table Design Windows.

Reconciling a Database Diagram with a Modified Database

You save your database diagram when you are ready to update the database to match your diagram. However, if other users have updated the database since you opened your diagram, their changes might affect your diagram and vice versa. In such cases, saving your diagram will automatically:

- Recreate <u>database objects</u> that are referenced in your diagram but that another user deleted.
- Preserve <u>triggers</u> that were added to a table; triggers that were deleted from a table are not preserved.
- Delete an object you deleted in your diagram, even if another user edited that object.

Saving your diagram will reconcile the database with your diagram by overwriting other users' changes so that the database will match your diagram.

To reconcile your diagram with a modified database

- Save your database diagram.
 If you have not previously saved your diagram, type a name for the diagram in the Save New Database Diagram dialog box and choose OK.
- 2. The **Save** dialog box lists the tables that will be affected when you save your diagram. Choose **Yes** to continue.
- 3. The <u>Database Changes Detected dialog box</u> lists the objects that were modified and will be changed to match your diagram. Choose **Yes** to save the diagram and accept the list of changes.

Note If your diagram contains tables and columns that were deleted in the

database, only their definitions are recreated in the database when you save your diagram. This process does not restore any data that existed in these objects before their deletion.

See Also

Multiuser Environments

Working with Tables

The Visual Database Tools help you change your database. You can control when and how changes to a database made in a database diagram are saved. You control these changes by noting which objects have changed and which remain unchanged in the database diagram, by saving changes only to selected tables, and by discarding unwanted changes. You can also use <u>SQL change scripts</u> to track, discard, and apply unsaved changes.

То	See
Add a table	Adding Tables
Remove a table from a diagram	Removing a Table from a Database
	Diagram
Remove a table from the database	Deleting a Table from a Database
	Diagram and the Database
Rename a table	Renaming a Table
Duplicate a table	Duplicating a Table
Copy a table to another diagram	Copying a Table Across Database
	<u>Diagrams</u>
Change how tables appear in the	Changing a Table View in a Database
diagram	<u>Diagram</u>
Change which per-column properties	Changing Which Properties Appear
appear in a diagram	
Move columns to another table	Moving Columns from One Table to
	Another
Resize per-column property columns	Resizing Property Columns

Adding Tables

You can add a table to your database diagram to edit its structure or relate it to other tables in your diagram. You can either add existing database tables to a diagram or insert a new table that has not yet been defined in the database. Alternatively, you can create a table or modify an existing table with the Table Designer.

- <u>Creating a New Table with Table Designer</u>
- Inserting a New Table in a Diagram
- Adding an Existing Table to a Diagram
- Adding Related Tables to a Diagram

Creating a New Table with Table Designer

Using Table Designer, you can create a new table in the database without opening a database diagram.

To create a new table with Table Designer

- 1. Make sure you are connected to the database in which you want to create the table. Right-click the Tables node of the target database and choose **New Table**.
- 2. Add the columns and other details of the table.

Note For each column you add, a new row appears in the top portion of the Table Designer. Within that row, you can edit the column's basic properties. To edit the columns other properties, you can use the bottom portion of the Table Designer. Simply click on a particular row in the top portion of the Table Designer, then add or edit the values for the properties appearing in the bottom portion.

Inserting a New Table

Adding a new table to the database diagram means that you are defining a new table that does not already exist in the database. To create a new table, you must define the individual <u>columns</u> that make up the table. The table is created in the database when you save the table or the diagram.

To insert a new table into a diagram

1. Make sure you are connected to the database in which you want to create the table.

To create a table in your current diagram, click the **New Table** button in the toolbar.

-or-

Right-click in the diagram and select New Table.

- Modify or accept the system-assigned table name, in the <u>Choose Name</u> <u>dialog box</u>, and then choose **OK**. A new table appears in the diagram in Standard view.
- 3. In the first cell of the new table, type a column name. Then press the TAB key to move to the next cell.
- 4. Under Datatype, select a data type for the column. Each column must have a name and data type.You can set the column's other properties, such as Scale, by using the Columns property page. See <u>Setting Column Properties</u> for details.
- 5. Repeat steps 3 and 4 for each column you want to add to the table.

When you save your database diagram, the new table will be added to your database.

Note If you create a new table, then remove it from the diagram before saving it to the database, the table name remains in memory until you close the database diagram. Also, if you delete an existing table from the database,

the table name remains in memory. To use the table name again, close and restart the Database Designer.

Adding an Existing Table

Adding an existing table means that the table you want to appear in your diagram already exists in your database. If you add a group of tables to a diagram, any <u>relationships</u> that exist between the tables are also added to the diagram.

To add an existing table to a diagram

- 1. Open a database diagram.
- 2. Right-click inside the diagram, then choose **Add Table...**.
- 3. In the **Add Table** dialog box, click the table, then click **Add**. Alternatively, you can double-click any tables you want to add before closing the dialog box.
- 4. Click **Close** to close the **Add Table** dialog box.

If relationships exist between the selected table and other tables in your diagram, relationship lines are automatically drawn.

When you add a table to a diagram, the definition of the table (not the data that is stored in the table) is loaded from your database into memory. At that point you can edit the table's definition. For example, you can add new columns or modify its indexes, key, relationships, or constraints.

Adding Related Tables

For tables with existing <u>foreign key</u> constraints, you can easily add the related tables to the database diagram.

To add related tables to a diagram

- 1. Select one or more tables with foreign key constraints in the database diagram.
- 2. Right-click on any of the selected tables and choose **Add Related Tables**.

Both those tables referenced by a foreign key constraint from the selected table(s) and those referencing the selected table(s) with a foreign key constraint are added to the diagram.

See Also

Copying a Table Across Database Diagrams | Deleting a Table from a Database Diagram and the Database | Removing a Table from a Database Diagram | Renaming a Table | Tables

Removing a Table from a Database Diagram

You can remove a table from your database diagram. Removing a table does not alter your database. The table and its relationships to other tables continue to exist in the database.

If you remove a table that has been changed either by you or another user, a message prompts you to save the table before removing it:

- If you save the table before removing it, the database is updated with the changes.
- If you discard changes to the table before removing it, the table remains in memory in its modified state until you save the table or close the last open database diagram, even though the table no longer appears in your diagram. For more information, see <u>Interactions Among Database</u> <u>Diagrams and Table Design Windows</u>.

To remove a table from a database diagram

- 1. In your database diagram, select the table you want to remove.
- 2. Right-click the table and choose **Remove Table from Diagram** from the shortcut menu.

If the table has unsaved changes as a result of edits you made in the database diagram, a message prompts you to save the table before removing it.

The table is removed from your diagram but it continues to exist in the database.

See Also

Database Designer | Deleting a Table from a Database Diagram and the Database | Tables

Deleting a Table from a Database Diagram and the Database

Delete a <u>table</u> from your database diagram when you want to do all of the following:

- Remove the table from your diagram.
- Remove the table from every other diagram in which it appears.
- Mark the table for deletion from the database.

A table that is marked for deletion is permanently deleted from the database when you save your diagram. A reference to that table continues to exist in memory until you save the diagram. If you close the diagram without saving it, the table will continue to exist in your database and appear in your diagram and every other diagram in which it appeared before you marked it for deletion.

To delete a table from the database

- 1. In your database diagram, select the table you want to delete.
- 2. Right-click the table and choose **Delete Table from Database** from the shortcut menu.
- 3. A message box prompts you to confirm the deletion. Choose **Yes**.

Note Deleting a table automatically removes any relationships to it.

See Also

<u>Adding Tables to a Database Diagram</u> | <u>Database Designer</u> | <u>Removing a Table</u> <u>from a Database Diagram</u> | <u>Tables</u>

Renaming a Table

When you rename a table, the table name is automatically updated in every database diagram in which the table appears. It is updated in the database when you save the table or diagram.

Caution Think carefully before you rename a table. If existing queries, views, user-defined functions, stored procedures, or programs refer to that table, the name modification will make these objects invalid.

To rename a table

- 1. In your database diagram, select the table you want to rename.
- 2. Right-click the table, and choose **Property Pages** from the shortcut menu.
- 3. Choose the **Tables** tab.
- 4. In the **Table name** box, type a new name. Be sure to choose a name that does not duplicate one in the **Selected table** list. To cancel this action, press the ESC key before leaving this field.
- 5. Click **Close**.

The table is renamed in the diagram as soon as you exit the **Table name** box.

See Also

<u>Tables</u>

Duplicating a Table

A table can appear only once in a database diagram. However, if you want to create a new table that contains some or all of the same <u>columns</u> as an existing table in your diagram, you can duplicate the existing table as the first step in creating your new table. Then in your new table, you can delete unwanted columns, add new columns, and assign a unique table name.

Note This operation duplicates only a table's structure; it does not duplicate any table rows.

Use the same technique for duplicating tables across database diagrams.

To duplicate a table

In the database diagram where you want to duplicate the table, right-click the diagram and choose **New Table** from the shortcut menu. -or-

Click the **New Table** button in the Database Diagram toolbar.

- 6. In the <u>Choose Name dialog box</u>, accept or change the system-assigned name of the new table and click **OK**. A blank table is added to your diagram.
- 7. In the table you want to duplicate, select all the columns. This table can be in the same diagram, a different diagram in the same database, or a diagram in a different database.
- 8. Copy the columns to the blank table. For details about copying columns, see <u>Copying Columns from One Table to Another</u>.

See Also

<u>Tables</u>

Copying a Table Across Database Diagrams

You can copy a table from one database diagram to another in the same database.

Copying a table from one database diagram to another diagram merely adds a reference to the table in the second diagram. The table is not duplicated in your database. For example, if you copy the authors table from one database diagram to another, each diagram references the same authors table in the database.

To copy a table from another database diagram

- 1. Make sure you are connected to the database whose table you want to copy.
- 2. Open the source and target database diagrams and within the source diagram, select the table that you want to copy to the target diagram.
- 3. Click the **Copy** button on the toolbar. This action places the selected table definition on the Clipboard.
- 4. Switch to the target diagram. This diagram must be in the same database as the source diagram.
- 5. Click the **Paste** button on the toolbar. The Clipboard contents appear at the new location and remain highlighted until you click elsewhere. If relationships exist between the selected tables and other tables in the target diagram, relationship lines are automatically drawn.

When you edit the table in either diagram, your changes are reflected in both diagrams. Similarly, once you save the table in either diagram, the table is no longer considered "modified" in either diagram. For more information, see <u>Interactions Among Table Diagrams and Table Designer Windows</u>.

See Also

Adding Tables to a Database Diagram | Database Designer | Duplicating a Table | Tables

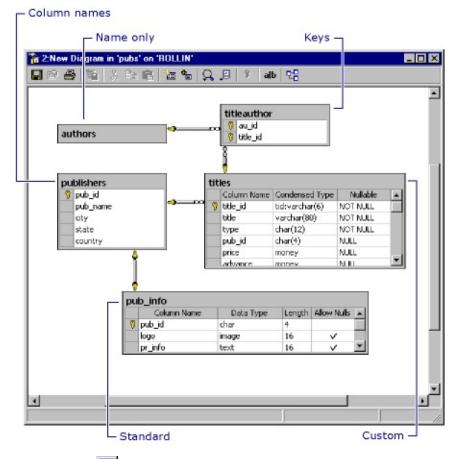
Changing a Table View in a Database Diagram

When you are working with only a few tables in a database diagram, it is usually helpful to view as much column information as possible. Such a view not only gives you more information, but it also enables you to edit the column definitions.

When you are working with a large number of tables, however, it is usually preferable to view only the column names, or only the table name, so that you can see more of the tables in your diagram at once.

In a database diagram, you can expand the tables you want to edit as needed and then collapse them again to save space on the diagram.

Each table in a database diagram can be displayed in one of five views.



Standard Shows every database column of the table, but shows only the column name and other fundamental properties of each database column. You

can edit the column properties directly by typing in a cell or using the **Cut**, **Copy**, and **Paste** commands. You can also move and delete entire database column definitions.

When you insert a new table into a diagram, it is shown in this view so that you can begin defining each column.

Column Names Shows only the name of each database column. You can rename the database columns in this view. When you add an existing table to a diagram, it is shown in this view.

Keys Shows only the names of only some of the table's columns. The columns whose names appear are those participating in a primary key, a foreign key, or a unique constraint. You can rename some of the database columns in this view.

Name Only Shows only the table name.

Custom Shows all of the table's database columns, but shows only the properties you choose. This gives you the flexibility to display only the information you want. This view is also useful for printing the database diagrams, because you can display more tables on one page, and will not normally see horizontal scroll bars on the tables.

The default set of properties for the Custom view includes Column Name, Condensed Datatype, and Nullable.

Property	Setting
Column Name	The name of the field in the table.
Condensed Datatype	Information about the field's data type, in the same format as the SQL Create Table statement. For example, a field containing a variable-length string with a maximum length of 20 characters would be represented as varchar(20).
Nullable	Null (the field can contain a Null value) or Not Null (the field must contain a non-null value).

To change a table view

- 1. In your database diagram, select the tables you want to see in another view. To change the view of all the tables, right-click in the diagram and choose **Select All**.
- 2. Right-click the table and select the desired view from the shortcut menu.

When you finish editing a table, you can change it to a different view or resize it to create more space on the diagram.

To modify the Custom view

- 1. In your database diagram, right-click the table you want to see in Custom view.
- From the shortcut menu, choose Table View... then Modify Custom View. The <u>Column Selection dialog box</u> appears.
- 3. Use the arrows to move the desired columns from the **Available columns** box to the **Selected columns** box.

Note To add a column, highlight the column in the **Available columns** box and click the > button. The column will then be listed in the **Selected columns** box. To display all columns, click the >> button. To remove a column from the **Selected columns** box, highlight the column and click the < button.

- 4. The columns will appear in the table in the order they are displayed in the **Selected columns** box. Use the **Sort** arrows to rearrange the columns in the **Selected columns** box to the desired order.
- 5. To make this setting the default Custom view, select the **Save as default** box.

Note If the **Save as default** box is not selected, the columns listed in the **Selected columns** box will be displayed for all the

database tables displayed in Custom view on the current diagram. However, this setting will not be saved and is not applied to other diagrams when you choose the **Custom view** command.

6. Click **OK**.

The selected table is redrawn to display the columns chosen in the dialog box.

See Also

<u>Tables</u>

Changing Which Properties Appear

When you show a database table in Standard view, some but not all of that table's <u>property</u> columns are visible. You can customize this view by choosing whether to hide or show each property column in a database table.

Note When you change which properties appear, you change only the displayed properties for that table in that diagram.

To hide a property column

1. In your database diagram, select the table whose property column you want to hide.

Position the pointer along the right border of the column header for the column that you want to hide. The cursor changes from an arrow to a splitter pointer \leftrightarrow .

- 2. Drag the column border to the left until it meets the column's left border.
- 3. Release the mouse button.

To show a property column that is hidden

- 1. In your database diagram, select the table whose property column you want to show.
- 2. Position the pointer along the right border of the previous column's header. The cursor changes from an arrow to a splitter pointer.
- 3. Double-click the splitter pointer. The hidden column resizes to fit the widest entry.

Note You can also set which column properties you want to display for database tables in a database diagram by using Custom view. For information on this view, see <u>Changing a Table View</u>.

See Also

Column Properties | Resizing Property Columns

Moving Columns from One Table to Another

You can move columns from one table to another table in the same database diagram or in a different diagram. Moving columns in the database diagram moves the column definition only. The data itself is not automatically transferred to the second table as part of this process.

Tip To move both the column's definition and its data, you can use the Query Designer in conjunction with the Database Designer. For information about the Query Designer, see <u>Designing Queries</u>.

To move columns without data from one table to another

- 1. In your database diagram, select the columns that you want to move.
- 2. Click the **Cut** button on the toolbar. This action deletes the selection from the table and places the column and its current set of <u>properties</u> on the Clipboard.
- 3. Position the cursor in the new table at the location where you want to insert the columns.
- 4. Click the **Paste** button on the toolbar. The columns are inserted at the new location and remain highlighted until you click elsewhere.

To move columns with data from one table to another

- 1. In your database diagram, select the columns that you want to move.
- 2. Click the **Copy** button on the toolbar. This action places the selection with its current set of properties on the Clipboard.
- 3. Position the cursor in the new table at the location where you want to insert the columns.

- 4. Click the **Paste** button on the toolbar. The columns are inserted at the new location and remain highlighted until you click elsewhere.
- 5. Run an <u>update query</u> to add the data to the table to which you copied the columns. For details, see <u>Creating Update Queries</u>.
- 6. Return to the database diagram and delete the columns from the original table. For details, see <u>Deleting Columns from a Table</u>.

See Also

<u>Tables</u>

Resizing Property Columns

You can manually change the width of a <u>property</u> column in a database table by resizing it in Standard view. For example, you can widen the Datatype column if it contains user-defined data type names that are too long to see in their entirety. You can also automatically resize a property column to fit its widest entry.

To manually change the width of a property column

1. In your database diagram or in Table Designer, select the table whose property columns you want to resize.

Position the pointer along the right border of the column header for the column that you want to resize. The cursor changes from an arrow to a splitter pointer \leftrightarrow .

- 2. Drag the column border to the left to reduce its width or to the right to increase its width.
- 3. Release the mouse button.

To resize a property column to fit the widest entry

- 1. In your database diagram, select the table whose columns you want to resize.
- 2. Position the pointer along the right border of the column header for the column that you want to resize. The cursor changes from an arrow to a splitter pointer.
- 3. Double-click the splitter pointer.

Note You can also resize property columns in Custom view for any columns you have chosen to display in this view.

See Also

Changing Which Properties Appear | Column Properties

Working with Columns

You can edit the column properties for tables directly in the database diagram or Table Designer.

То	See
Add new column definitions to the	Inserting Columns into a Table
end of a table	
Copy the column definitions from	Copying Columns from One Table to
one table to another table in the same	Another
diagram or in different diagrams	
Move the column definitions from	Moving Columns from One Table to
one table to another table in the same	Another
database diagram or in a different	
diagram	
Delete columns from a table and	Deleting Columns from a Table
from the database	
Set or edit the column definition	Column Properties
(properties) for a database table	

Inserting Columns into a Table

You can add new column definitions to a table to capture additional data that is not already stored in an existing <u>column</u>. The Standard view, which shows a <u>table</u> with all of its currently defined columns, provides blank grid rows so that you can easily add new column definitions to your database table.

To insert columns into a table from a database diagram

- 1. In your database diagram, select the table that you want to add new columns to.
- 2. If the table is not already in **Standard** view, right-click the table and choose **Standard** from the shortcut menu.
- 3. Right-click a row in the table and select **Insert Column** from the shortcut menu. A blank column row is inserted above the selected row.
 -orPlace the cursor in the first blank cell in the **Column Name** column.
- 4. In the **Column Name** column, type the column name in the cell. The Column Name is a required value.
- 5. Press the TAB key to go to the **Datatype** cell. This is a required value.

The default values for your database are added when you create a new column. Define other properties for the column as needed. You can change any of these values and set additional properties at any time according to the rules of your database.

To insert columns into a table from the Table Designer

1. Open the Table Designer for the table to which you want to add a column.

- 2. Right-click a row in the table and select **Insert Column** from the shortcut menu. A blank column row is inserted.
 -orPlace the cursor in the first blank cell in the **Column Name** column.
- 3. In the **Column Name** column, type the column name in the cell. The Column Name is a required value.
- 4. Press the TAB key to go to the **Datatype** cell. This is a required value.

The default values for your database are added when you create a new column. Define other properties for the column as needed. To define any of the fundamental properties, enter the property value on the grid. To define any of the other properties, highlight the grid row describing that database column, then enter the property value in the appropriate control on the tab appearing under the grid.

See Also

Setting Column Properties | Tables

Copying Columns from One Table to Another

You can copy columns from one table to another table in the same diagram or in different diagrams. Copying a column involves only the column definition. The data itself is not automatically transferred to the second table as part of this process.

Tip You can use database queries to copy the column's data from the original column to the new column. Use the Query Designer to run an <u>update query</u> to add the data to the table to which you copied the columns. For details, see <u>Creating Update Queries</u>. For information about the Query Designer, see <u>Designing Queries</u>.

To copy columns from one table to another

- 1. In your database diagram, select the columns that you want to copy.
- 2. Click the **Copy** button on the toolbar. This action copies the selection from the table and places the column and its current set of <u>properties</u> on the Clipboard.
- 3. Position the cursor in the table at the location where you want to insert the columns.
- 4. Click the **Paste** button on the toolbar. The column and its properties are inserted at the new location.

When you copy a database column that has a <u>user-defined data type</u> from one database to another, the user-defined data type may not be available in the destination database. In such a case, the column will be assigned the nearest matching base data type available in that database.

See Also

Duplicating a Table | Tables

Deleting Columns from a Table

Delete columns from a table when they are no longer needed to store data. When a column is deleted from the table in the database diagram, it and all the data it contains are deleted from the database.

Caution This action cannot be undone once the database diagram or table is saved. The only way to restore a deleted column is to close the table or database diagram without saving changes.

To delete columns from a table

- 1. In your table or database diagram, select the column or columns you want to delete.
- 2. Right-click the column and choose **Delete Column** from the shortcut menu.
- 3. If the column participates in a relationship, a message prompts you to confirm the deletion of the selected columns and their relationships. Choose **OK**.

If the column does not participate in a check constraint, then the column, any constraints attached to it, any relationships it participates in, and any data contained in the column are removed from the database and the diagram. They are deleted from the database when you save the table or database diagram. If the column does participate in a check constraint, the database server will reject your modification when you save your work. That is, the commit operation will fail. To delete a column that participates in a check constraint, you must first modify or remove the check constraint before deleting the column.

See Also

Deleting a Table from a Database Diagram and the Database | Inserting Columns into a Table | Tables

Setting Column Properties

You can set or change the properties of a column.

To set column properties in a database diagram

- 1. In your database diagram, select the table whose column properties you want to define.
- 2. Right-click the table and choose **Properties** from the shortcut menu.
- 3. Select the **Columns** tab.
- 4. For each property you want to change, modify the value or setting.
- 5. Click **Close** to close the property page.

The new setting takes effect in the database diagram as soon as you press the TAB key or click outside the cell in the **Properties** dialog box. The new setting is saved in the database when you save the table or your diagram.

Note If a column property is visible on the database diagram, you can edit that property value directly — without opening the property page. For information about controlling which column properties are visible on a diagram, see <u>Changing a Table View in a Database Diagram</u>.

To set column properties in Table Designer

- 1. Open the Table Designer for the table containing the column whose property you want to modify.
- To set a fundamental property, click in the grid cell for that property of the database column you want to modify, then enter the value.
 -or-

To set another property, select the grid row describing the database

column you want to modify, then modify the property in the tab appearing below the grid.

The new setting takes effect in the database diagram as soon as you press the TAB key or click outside the edited grid cell or control. The new setting is saved in the database when you save the table or your diagram.

See Also

Column Properties

Renaming a Column

The name of a column in a table in your database diagram shows the name of the column as it is stored in the database. You can rename a column directly in your table in the database diagram and the database will be updated with the new name when you save the table or diagram.

Column names are stored in the case (uppercase or lowercase) in which they appear in your diagram.

Caution Renaming a column may affect <u>triggers</u>, <u>stored procedures</u>, and <u>constraints</u>. Consider renaming a column before you create these other objects. For more information about these database objects, see <u>Database</u> <u>Objects</u>.

To rename a column

- 1. In your database diagram, select the table whose column you want to rename.
- 2. If only the table name is shown in the database diagram, right-click the table and select **Standard**, **Column Names**, or **Keys** from the shortcut menu.
- 3. In the cell that shows the column name you want to change, type a new column name.

The column is renamed in your table or diagram as soon as you exit the cell that shows the column name. The column is renamed in your database when you save the table or diagram.

See Also

Column Properties | Uniqueness of Database Object Names

Working with Relationships

A relationship is a type of association between table rows. For an overview of relationships, see <u>Table Relationships</u>.

For details about working with relationships, see the following topics:

То	See
Create relationships between	Creating a Relationship Between
database tables in a database diagram	Tables
Ensure each value entered in a	Enforcing Referential Integrity
foreign key column matches an	Between Tables
existing value in the related primary	
key column	
Link a column in a table with another	Drawing a Single-Table Reflexive
column in the same table	Relationship
Create a many-to-many relationship	Mapping Many-to-Many
	Relationships to a Database Diagram
Change the name of a relationship	Renaming a Relationship
Remove the relationship between	Deleting a Relationship
two tables	
Disable a foreign key constraint	Disabling a Foreign Key Constraint
	with INSERT and UPDATE
	Statements and Disabling a Foreign
	Key Constraint for Replication

Creating a Relationship Between Tables

You create a relationship between two tables when you want to associate rows of one table with rows of another. For more background about relationships, see <u>Table Relationships</u>.

To create a relationship in a database diagram

In your database diagram, click the row selector **I** for the database column or combination of columns that you want to relate to a column in another table.

- 1. While the pointer is positioned over the row selector, click and drag to the related table.
- 2. Release the mouse button. The <u>Create Relationship</u> dialog box appears and attempts to match the columns you selected with columns of the same name and data type in the related table.
- 3. In the **Create Relationship** dialog box, confirm that the columns you want to relate are shown in the **Primary key table** and **Foreign key table** lists.
- 4. Choose **OK** to create the relationship.

On the diagram, the primary key side of the relationship is denoted by a key symbol. In one-to-one relationships, the table that initiated the relationship determines the primary key side. For example, if you create a relationship from the pub_id column in the publishers table to the pub_id column in the publishers table is on the primary key side of the relationship.

To create a relationship in Table Designer

1. Open the Table Designer for the table that will be on the foreign key side of the relationship.

- 2. Right-click in the Table Designer and choose **Relationships**.
- 3. Click the **New** button.
- 4. From the drop-down list in **Primary Key Table**, choose the table that will be on the primary-key side of the relationship. In the grid beneath, enter the columns contributing to the table's primary key. In the adjacent grid cell to the left of each column, enter the corresponding foreign-key column of the foreign-key table. The table designer suggests a name for the relationship. To change this name, edit the contents of the **Relationship Name** text box.
- 5. Choose **Close** to create the relationship.
- 6. For more details, see the <u>Relationships Property Page</u>.

See Also

Defining a Primary Key | Deleting a Relationship | Mapping Many-to-Many Relationships to a Database Diagram | Renaming a Relationship | Table Relationships

Enforcing Referential Integrity Between Tables

<u>Referential integrity</u> between tables is enforced by default when you create a relationship in your database diagram. An enforced relationship ensures each value entered in a foreign key column matches an existing value in the related primary key column.

You can change the conditions under which referential integrity is enforced by editing the relationship's properties.

To change referential integrity options for a new relationship

- 1. In your database diagram, create a relationship. For details, see <u>Creating a Relationship Between Tables</u>.
- 2. In the **Create Relationship** dialog box, clear or select one or more of the options.

For information about the options available for SQL Server databases, see <u>Database Designer Considerations for SQL Server Databases</u>.

To change referential integrity options for an existing relationship

- 1. In your database diagram, select the relationship line.
- 2. Right-click the relationship line and select **Properties**.
- 3. Choose the **Relationships** tab.
- 4. Select the relationship from the **Selected relationship** list.
- 5. Clear or select one or more of the options.

For information about the options available for SQL Server databases, see <u>Database Designer Considerations for SQL Server Databases</u>.

The relationship is updated in the database when you save the diagram or either of the related tables.

See Also

<u>Foreign Key Constraints</u> | <u>Primary Key Constraints</u> | <u>Table Relationships</u> | <u>Unique Constraints</u>

Drawing a Reflexive Relationship

You create a <u>reflexive relationship</u> to link a column or columns in a table with another column or columns in the same table. For example, suppose the employee table has an emp_id column and a mgr_id column. Because each manager is also an employee, you relate these two columns by drawing a relationship line from the table to itself. This relationship ensures each manager ID that is added to the table matches an existing employee ID.

Before you create a relationship, you must first define a primary key or unique constraint for your table. You then relate the primary key column to a matching column. Once you create the relationship, the matching column becomes a foreign key of the table.

To draw a reflexive relationship

- 1. In your database diagram, click the row selector for the database column or columns that you want to relate to another column or columns.
- 2. While the pointer is positioned over the row selector, drag the pointer outside the table until a line appears.
- 3. Drag the line back to the selected table.
- 4. Release the mouse button. The <u>Create Relationship</u> dialog box appears and attempts to match the primary key columns with the nonkey columns you dragged the line to.
- 5. Confirm that the columns you want to relate are shown in the **Primary key table** and **Foreign key table** lists.
- 6. Choose **OK** to create the relationship.

When you run queries against a table, you can use a reflexive relationship to create a self-join. For information about querying tables with joins, see <u>Combining Tables</u>.

See Also

Table Relationships

Deleting a Relationship

Delete a relationship when you no longer want to relate columns in two related tables. When you redesign tables, it is often necessary to delete relationships and then recreate them after your new design is complete. For example, if you decide to normalize a database and store all address data in one table, you would delete all the relationships to address columns between existing tables, create a new table containing the address columns, and then create relationships from the new address table to every table that requires an address.

To delete a relationship

- 1. In your database diagram, select the line that represents the relationship that you want to delete from the diagram.
- 2. Right-click the relationship line and choose **Delete Relationship from Database** from the shortcut menu.
- 3. A message box prompts you to confirm the deletion. Choose **Yes**. -or-
- 1. Right-click on any table in a database diagram or right-click within the Table Designer and choose **Relationships** from the shortcut menu.
- 2. Select the relationship from the **Selected relationship** drop-down list.
- 3. Click the **Delete** button.

Note When you delete a relationship, the relationship line is removed from every diagram in which it appears. It is deleted from the database when you save the diagram or when you save either of the tables that it related.

See Also

Creating a Relationship Between Tables | Table Relationships

Renaming a Relationship

You can rename a relationship. When you rename a relationship, the relationship name is automatically updated in the current diagram and any other diagram in which it appears.

To rename a relationship

- 1. In your database diagram, select the line representing the relationship that you want to rename.
- 2. Right-click the relationship line and select **Properties**.
- 3. Choose the **Relationships** tab.
- 4. In the **Relationship name** box, type a new name.

The relationship is renamed in the diagram and any other diagram in which it appears as soon as you exit the **Relationship name** box; it is renamed in the database when you save either of the related tables or the diagram.

See Also

Table Relationships

Checking Existing Data when Creating a Relationship

Select the option to check existing data when you create a relationship if the <u>foreign key</u> constraint should apply to existing data as well as to new data.

To check existing data when creating a relationship

- 1. In your database diagram, select the table that the foreign key constraint is attached to.
- 2. Right-click the table and select **Properties**.
- 3. Choose the **Relationships** tab.
- 4. Select the relationship from the **Selected relationship** list.
- 5. Select the **Check existing data on creation** check box.

The foreign key constraint is applied when you save the table or diagram. If any constraint violations are encountered during the save process, the table cannot be saved.

See Also

<u>Check Constraints | Constraints | Creating a Relationship.</u>

Mapping Many-to-Many Relationships to a Database Diagram

Many-to-many relationships let you relate each row in one table to many rows in another table, and vice versa. For example, you could create a many-to-many relationship between the authors table and the titles table to match each author to all of his or her books and to match each book to all of its authors. Creating a <u>one-to-many relationship</u> from either table would incorrectly indicate that every book can have only one author, or that every author can write only one book.

Many-to-many relationships between tables are accommodated in databases by means of junction tables. A junction table contains the primary key columns of the two tables you want to relate. You then create a relationship from the primary key columns of each of those two tables to the matching columns in the junction table. In the pubs database, the titleauthor table is a junction table.

To create a many-to-many relationship between tables

- 1. In your database diagram, add the tables that you want to create a many-to-many relationship between.
- Create a third table by right-clicking the diagram and choosing New Table from the shortcut menu. This will become the junction table. For details, see <u>Adding Tables to a Database Diagram</u>.
- 3. In the <u>Choose Name</u> dialog box, change the system-assigned table name. For example, the junction table between the titles table and the authors table is now named titleauthors. For details, see <u>Renaming a Table</u>.
- 4. Copy the primary key columns from each of the other two tables to the junction table. You can add other columns to this table, just as you can to any other table. For details, see <u>Copying Columns from One Table to Another</u>.

- 5. In the junction table, set the primary key to include all the primary key columns from the other two tables. For details, see <u>Defining a Primary</u> <u>Key</u>.
- 6. Define a one-to-many relationship between each of the two primary tables and the junction table. The junction table should be at the "many" side of both of the relationships you create. For details, see <u>Creating a Relationship Between Tables</u>.

Note The creation of a junction table in a database diagram does not insert data from the related tables into the junction table. For information about inserting data into a table, see <u>Creating Insert Queries</u>.

See Also

Table Relationships

Working with Indexes

An index is a mechanism for providing fast access to table rows or for enforcing certain constraints. For an overview of indexes, see <u>Indexes</u>.

For details about working with indexes, see the following topics:

То	See
Create an index	Creating an Index
Create an indexed view	Creating Indexed Views
Create a unique index	Creating a Unique Index
Create a clustered index	Creating a Clustered Index
Rename an index	Renaming an Index
Delete an index	Deleting an Index
Save, display, and update an index	Saving, Displaying, and Updating an
definition	Index Definition
Specify a fill factor for an index	Specifying a Fill Factor for an Index

Creating an Index

You can use an index to speed access to data in a database <u>table</u>. You create an index by selecting one or more <u>columns</u> in a table that you want to be able to search on. You can use the index once you save the table.

To create an index

1. In your database diagram, select the table you want to index, rightclick the table, and choose **Indexes/Keys** from the shortcut menu.

-or-

Open the Table Designer for the table you want to index, right-click in the Table Designer, and choose **Indexes/Keys** from the shortcut menu.

- 2. Choose **New**. The **Selected index** box displays the system-assigned name of the new index.
- 3. Under **Column name**, select the columns you want to index. You can select up to 16 columns. For optimal performance, select only one or two columns. For each column you select, you can indicate whether the index organizes its values in ascending or descending order.
- 4. Specify any other desired settings for the index and then click **OK**.

The index is created in the database when you save the table or diagram.

See Also

<u>Creating a Clustered Index | Creating a Unique Index | Deleting an Index |</u> <u>Indexes | Renaming an Index | Saving, Displaying, and Updating an Index</u> <u>Definition | Specifying a Fill Factor for an Index</u>

Creating a Unique Index

In SQL Server, you can create a unique <u>index</u> when uniqueness is a characteristic of the data itself, but the combination of indexed <u>columns</u> is not the same as the table's <u>primary key</u>. For example, if you plan to query frequently on the Social Security number (SSN) column in the employee <u>table</u> (where the primary key is emp_id), and you want to ensure Social Security numbers are unique, create a unique index on SSN. If the user enters the same Social Security number for more than one employee, the database displays an error and cannot save the table.

When you create or modify a unique index, you can set an option to ignore duplicate keys. If this option is set and you attempt to create duplicate keys by adding or updating data that affects multiple <u>rows</u> (with the INSERT or UPDATE statement), the row that causes the duplicates is not added or, in the case of an update, discarded.

For example, if you try to update "Smith" to "Jones" in a table where "Jones" already exists, you end up with one "Jones" and no "Smith" in the resulting table. The original "Smith" row is lost because an UPDATE statement is actually a DELETE followed by an INSERT. "Smith" was deleted and the attempt to insert an additional "Jones" failed. The whole transaction cannot be rolled back because the purpose of this option is to allow a transaction in spite of the presence of duplicates.

To create a unique index

1. In your database diagram, select the table you want to index, rightclick the table, and choose **Indexes/Keys** from the shortcut menu.

-or-

Open the Table Designer for the table you want to index, right-click in the Table Designer, and choose **Indexes/Keys** from the shortcut menu.

- 2. Choose **New**. The **Selected index** list displays the system-assigned name of the new index.
- 3. Under **Column name**, select the columns you want to index. You can

select up to 16 columns. For optimal performance, select only one or two columns per index. For each column you select, indicate whether the index arranges values of this column in ascending or descending order.

- 4. Select the **Create UNIQUE** check box.
- 5. Select the **Index** option.
- 6. Select the **Ignore duplicate keys** option if you want to ignore new or updated data that would create a duplicate key in the index (with the INSERT or UPDATE statement).

The index is created in the database when you save the table or diagram.

Note You cannot create a unique index on a single column if that column contains NULL in more than one row. Similarly, you cannot create a unique index on multiple columns if the combination of columns contains NULL in more than one row. These are treated as duplicate values for indexing purposes.

See Also

<u>Creating a Unique Constraint | Defining a Primary Key | Indexes | Saving,</u> <u>Displaying, and Updating an Index Definition | Unique Constraints</u>

Creating a Clustered Index

In Microsoft® SQL ServerTM databases you can create a clustered index. In a clustered index, the physical order of the rows in the table is the same as the logical (indexed) order of the index key values. A table can contain only one clustered index. UPDATE and DELETE operations are often accelerated by clustered indexes because these operations require large amounts of data to be read. Creating or modifying a clustered index can be time-consuming, because it is during these operations that the table's rows are reorganized on disk.

Consider using a clustered index for:

- Columns that contain a limited number of unique values, such as a state column that contains only 50 unique state codes.
- Queries that return a range of values, using operators such as BETWEEN, >, >=, <, and <=.
- Queries that return large <u>result sets</u>.

To create a clustered index

1. In your database diagram, select the table you want to index, rightclick the table, and choose **Indexes/Keys** from the shortcut menu.

-or-

Open the Table Designer for the table you want to index, right-click in the Table Designer, and choose **Indexes/Keys** from the shortcut menu.

- Create a new index. For details, see <u>Creating an Index</u>. To modify an existing index, select the index from the **Selected index** list.
- 3. Select the **Create as CLUSTERED** check box.

The index is created in the database when you save the table or diagram.

See Also

Indexes | Saving, Displaying, and Updating an Index Definition

Renaming an Index

New indexes are automatically given system-defined names based on the database table name. If you create multiple indexes on a table, the index names are appended with "_1", "_2", and so on. You can rename an index as long as index names are unique within the table.

Note When you create a <u>primary key</u> or <u>unique constraint</u> for a table, an index with the same name as the constraint is automatically created for the table. Because index names must be unique for a table, you cannot create or rename an index to have the same name as the primary key or unique constraint for the table.

To rename an index

1. In your database diagram, select the table whose index you want to rename, right-click the table, and choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table whose index you want to rename, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. Select the index from the **Selected index** list.
- 4. Type a new name in the **Index name** box. Make sure that it does not duplicate a name in the **Selected index** list.

The index is renamed in your diagram when you exit the **Index name** box. It is renamed in the database when you save the table or diagram.

See Also

Deleting an Index | Indexes | Saving, Displaying, and Updating an Index Definition

Deleting an Index

Indexes can slow INSERT, UPDATE, and DELETE performance. If you find that an index hinders overall performance or you no longer need it, you can delete it.

To delete an index

1. In your database diagram, select the table whose index you want to delete, right-click the table, and choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table whose index you want to delete, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. From the **Selected index** list, select the index you want to delete.
- 4. Choose **Delete**.

Caution Choosing **Delete** will result in an action that cannot be undone without losing all other changes made to the database diagram. To undo this action, close this database diagram and all other open database diagrams and Table Designer windows without saving the changes.

The index is deleted from the database when you save your table or diagram.

See Also

<u>Creating an Index</u> | <u>Indexes</u> | <u>Saving, Displaying, and Updating an Index</u> <u>Definition</u>

Saving, Displaying, and Updating a Table's Index Definition

Your index is automatically saved in the database when you save your table or database diagram. The index is available for modification as soon as you create it.

To display and update index definitions

1. In your database diagram, right-click the table and choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table whose index you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. To update the definition, select the field you want to update and change its value or setting.

Your changes are saved to the database when you save the table or diagram.

Note that you can also manage indexes on some views. For more information, see <u>Indexed Views</u>.

See Also

Indexes

Specifying a Fill Factor for an Index

You can identify a fill factor to specify how full each <u>index</u> page can be. The amount of empty space on an index page is important because when an index page fills up, the system must take time to split it to make room for new <u>rows</u>.

It is seldom necessary to specify a fill factor when you create an index. The option is provided for fine-tuning performance. It is useful when you are creating a new index on a table with existing data, and particularly when you can accurately predict future changes in that data.

To specify a fill factor for an index on a table

1. In your database diagram, right-click the table containing the index for which you want to specify a fill factor, and choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the index for which you want to specify a fill factor, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. Select the index from the **Selected index** list.
- 4. In the **Fill factor** box, type a percentage from 0 to 100.

To specify a fill factor for an index on a view

- 1. Open the View Designer for the view containing the index for which you want to specify a fill factor, right-click in the View Designer, and choose **Manage Indexes** from the shortcut menu.
- 2. Select the index from the **Selected index** list.

3. In the **Fill factor** box, type a percentage from 0 to 100.

For more information about fill factors and their uses, see *SQL Server Books Online*.

See Also

<u>Indexes</u>

Working with Keys

A primary key is a constraint that assures that each table contains no duplicate rows. A foreign key is a constraint that enforces referential integrity. For more information about keys, see <u>Keys</u>.

То	See
Define a primary key	Defining a Primary Key
Modify a primary key	Modifying a Primary Key
Delete a primary key	Deleting a Primary Key Constraint
Define a foreign key	Creating a Relationship Between
	<u>Tables</u>
Modify a foreign key	Modifying a Foreign Key
Delete a foreign key	Deleting a Foreign Key Constraint
Copy primary-key columns to a	Copying Column Properties to a
foreign-key table	Foreign Key Column
Ensure each value entered in a	Enforcing Referential Integrity
foreign key column matches an	Between Tables
existing value in the related primary	
key column	
Disable a foreign key constraint	Disabling a Foreign Key Constraint
	with INSERT and UPDATE
	Statements and Disabling a Foreign
	Key Constraint for Replication

For details about working with keys, see the following topics:

Defining a Primary Key

Define a <u>primary key</u> to enforce uniqueness for values entered in specified <u>columns</u> that do not allow nulls. If you define a primary key for a <u>table</u> in your database, you can relate that table to other tables, thus reducing the need for redundant data. A table can have only one primary key.

To define a primary key

- 1. In your database diagram or Table Designer, click the row selector for the database column you want to define as the primary key. If you want to select multiple columns, hold down the CTRL key while you click the row selectors for the other columns.
- 2. Right-click the row selector for the column and select **Set Primary Key**. A primary key <u>index</u>, named "PK_" followed by the table name, is automatically created; you can find it on the **Indexes/Keys** tab of the property pages.

Warning If you want to redefine the primary key, any <u>relationships</u> to the existing primary key must be deleted before the new primary key can be created. A message will warn you that existing relationships will be automatically deleted as part of this process.

A primary key column is identified by a primary key symbol 🕅 in its row selector.

If a primary key consists of more than one column, duplicate values are allowed in one column, but each combination of values from all the columns in the primary key must be unique.

If you define a compound key, the order of columns in the primary key matches the order of columns as shown in the table in your diagram. However, you can change the order of columns after the primary key is created. For more information, see <u>Modifying a Primary Key</u>.

See Also

Constraints | Deleting a Primary Key Constraint | Enforcing Referential Integrity Between Tables | Modifying a Primary Key | Primary Key Constraints | Table Relationships

Modifying a Primary Key

Modify a <u>primary key</u> if you want to change the <u>column</u> order, <u>index</u> name, clustered option, or <u>fill factor</u>.

To modify a primary key

1. In your database diagram, right-click the table whose primary key you want to modify, and choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table whose primary key you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. Select the primary key index from the **Selected index** list.
- 4. Complete an action from the following table:

То	Follow these steps
Change the column	In the Column name grid, remove the
order	columns from the primary key. Then add the
	columns back in the order you want. To
	remove a column from the key, simply
	remove the column name from the Column
	name list.
Rename the	Type a new name in the Index name box.
primary key	Make sure that your new name does not
	duplicate a name in the Selected index list.
Set the clustered	Select the Create as CLUSTERED check
option	box. Only one clustered index can exist per
	table. If this option is not available for your
	index, you must first clear this setting on the
	existing clustered index.

Define a fill factor	Type an integer from 0 to 100 in the Fill
	factor box.

The primary key is updated in the database when you save your table or diagram.

See Also

<u>Constraints</u> | <u>Deleting a Primary Key Constraint</u> | <u>Enforcing Referential Integrity</u> <u>Between Tables</u> | <u>Primary Key Constraints</u>

Deleting a Primary Key Constraint

Delete a <u>primary key constraint</u> when you want to remove the requirement for uniqueness for the values entered in a <u>column</u> or a combination or columns.

To delete a primary key constraint

In your database diagram or Table Designer, select the primary key columns for the table whose primary key constraint you want to delete. A primary key column is identified by a primary key symbol 🗊 in its row selector.

1. Right-click the row selector for the column and select **Set Primary Key**.

-or-

- 2. In your database diagram, select the table whose primary key constraint you want to delete.
- 3. Right-click the table and select **Indexes/Keys**.
- 4. Select the primary key index from the **Selected index** list.
- 5. Choose **Delete**.

Caution Choosing **Delete** will result in an action that cannot be undone without losing all other changes made to the database diagram. To undo this action, close this database diagram and all other open database diagrams without saving the changes.

The constraint is deleted from the database when you save the table or diagram.

See Also

Constraints | Defining a Primary Key | Primary Key Constraints

Modifying a Foreign Key

Modify the foreign key side of a relationship if you want to change which <u>columns</u> are related to columns in the <u>primary key</u> table.

To modify a foreign key

1. In your database diagram, right-click the relationship corresponding to the foreign key you want to modify, then choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the foreign key you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Relationships** tab.
- 3. Select the relationship from the **Selected relationship** list.
- 4. In the **Foreign key table** column, expand the list in the first row.
- 5. Select a different table column from the list. The foreign key column must match the data type and size of the primary key column, with these exceptions:
 - A char column or sysname column can relate to a varchar column.
 - A binary column can relate to a varbinary column.
 - A <u>user-defined data type</u> can relate to its base type.

Any changes you make to the relationship's properties take effect as soon as you move outside the grid in the property pages. The constraint is updated in the

database when you save your table or diagram.

Note Modifying a relationship in a database diagram marks both of the related tables as modified. Consequently, each table will also be marked as modified in any other diagrams in which it appears.

See Also

<u>Constraints</u> | <u>Deleting a Foreign Key Constraint</u> | <u>Foreign Key Constraints</u> | <u>Modifying a Primary Key</u>

Viewing Foreign Key Attributes

View the <u>foreign key</u> attributes of a relationship if you want to see which <u>columns</u> participate in the foreign key side of a relationship. If the foreign key columns are related to a <u>primary key</u>, the primary key columns are identified in your database diagram by a primary key symbol in the row selector.

To view the foreign key attributes of a relationship

1. In your database diagram, right-click the relationship line corresponding to the foreign key, then choose **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the foreign key you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Relationships** tab.
- 3. Make sure the relationship is selected in the **Selected relationship** list.
- 4. The **Foreign key table** column displays the name of each column that participates in the foreign key side of the relationship.

See Also

Constraints | Foreign Key Constraints | Modifying a Foreign Key

Disabling a Foreign Key Constraint for Replication

SQL Server supports replication. Select the option to disable a <u>foreign key</u> <u>constraint</u> during replication if the constraint is specific to the source database and may unnecessarily prevent new data from being entered into the destination database.

To disable a foreign key constraint for replication

1. In your database diagram, right-click the table containing the foreign key, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the foreign key you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Relationships** tab.
- 3. Select the relationship from the **Selected relationship** list.
- 4. Clear the **Enable relationship for replication** check box.

See Also

Constraints | Foreign Key Constraints

Disabling a Foreign Key Constraint with INSERT and UPDATE Statements

Select the option to disable a <u>foreign key constraint</u> during INSERT and UPDATE transactions if you know that new data will violate the constraint or if the constraint applies only to the data already in the database.

To disable a foreign key constraint for INSERT and UPDATE statements

1. In your database diagram, right-click the table containing the foreign key, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the foreign key you want to modify, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Relationships** tab.
- 3. Select the relationship from the **Selected relationship** list.
- 4. Clear the **Enforce relationship for INSERTs and UPDATEs** check box.

After you add or modify data, you should select this option if you want to ensure the constraint applies to subsequent data modifications.

Note If you plan to use <u>triggers</u> to implement database operations, you must disable foreign key constraints in order for the trigger to run.

See Also

Constraints | Foreign Key Constraints

Deleting a Foreign Key Constraint

Delete a <u>foreign key</u> constraint when you want to remove the requirement to enforce <u>referential integrity</u>.

To delete a foreign key constraint

• In your database diagram, delete the relationship line that represents the foreign key constraint you want to delete. For details, see <u>Deleting a</u> <u>Relationship</u>.

Note Deleting a relationship from a database diagram marks the related tables as modified in all diagrams in which they appear.

See Also

Constraints | Foreign Key Constraints

Copying Column Properties to a Foreign Key Column

When you copy a <u>column</u> from one table to another table, the column name and many of the other properties you defined for that column in the original table are copied to the new location. You can copy column properties from a <u>primary key</u> column to a <u>foreign key</u> column if you want to relate the two columns. Related columns must have the same data type (or data types that can be related) and length property settings.

To copy primary key column properties to foreign key columns

In your database diagram, select the primary key columns that you want to copy. Primary key columns are indicated by a key symbol 🗊 in their row selector.

- 5. Copy the columns to the foreign key table. For details, see <u>Copying</u> <u>Columns from One Table to Another</u>.
- 6. Drag a relationship line from the columns in the primary key table to the same columns in the foreign key table. For details, see <u>Creating a</u> <u>Relationship Between Tables</u>.

See Also

Copying Columns from One Table to Another

Working with Constraints

Constraints are rules that the database server enforces for you. For more information see <u>Constraints</u>.

For details about working with constraints, see the following topics:

То	See
Attach a check constraint to a table to	Attaching a New Check Constraint to
specify the data values that are	<u>a Table or Column</u>
acceptable in one or more columns	
Create a constraint expression to	Defining a Check Constraint
check data for a condition	Expression
Change the constraint expression or	Modifying a Check Constraint
the options that enable or disable the	
constraint for specific conditions	
Apply constraints either to new data	Checking Existing Data When
only or to existing data as well	Creating a Check Constraint
Disable a check constraint when data	Disabling a Check Constraint with
is added to, updated in, or deleted	INSERT and UPDATE Statements
from a table	
Disable a check constraint when your	Disabling a Check Constraint for
table is replicated in another database	<u>Replication</u>
Remove the limitations on data	Deleting a Check Constraint
values in a column	
Ensure no duplicate values are	Creating a Unique Constraint
entered in specific columns	
Change the columns that the	Modifying a Unique Constraint
constraint is attached to, change the	
constraint name, or set additional	
properties for the constraint	
Remove the requirement for	Deleting a Unique Constraint
uniqueness for values entered in the	
column	
Enforce uniqueness for values	<u>Defining a Primary Key</u>

entered in specified columns	
Change the column order, index	Modifying a Primary Key
name, clustered option, or fill factor	
Copy column properties from a	Copying Column Properties to a
primary key column to a foreign key	<u>Foreign Key Column</u>
column to relate the two columns	
Remove the requirement for	Deleting a Primary Key Constraint
uniqueness for the values entered in a	
column	
See which columns participate in the	Viewing Foreign Key Attributes
foreign key side of a relationship	
Change which columns are related to	Modifying a Foreign Key
columns in the primary key table	
Check existing data when creating a	Checking Existing Data when
relationship	Creating a Relationship
Disable a foreign key constraint	Disabling a Foreign Key Constraint
during INSERT and UPDATE	with INSERT and UPDATE
transactions	<u>Statements</u>
Disable a foreign key constraint	Disabling a Foreign Key Constraint
during replication of the table	for Replication
Remove the requirement to enforce	Deleting a Foreign Key Constraint
referential integrity between primary	
key columns and the related columns	
in another table	

Attaching a New Check Constraint to a Table or Column

Attach a <u>check constraint</u> to a table to specify the data values that are acceptable in one or more columns.

To attach a new check constraint

1. In your database diagram, right-click the table that will contain the constraint, then select **Constraints** from the shortcut menu.

-or-

Open the Table Designer for the table that will contain the constraint, right-click in the Table Designer, and choose **Constraints** from the shortcut menu.

- 2. Choose **New**. The **Selected constraint** box displays the systemassigned name of the new constraint. System-assigned names begin with "CK_" followed by the table name.
- 3. In the **Constraint expression** box, type the SQL expressions for the check constraint. For example, to limit the entries in the **State** column of the authors table to New York, type: state = 'NY'

Or, to require entries in the zip column to be 5 digits, type:

zip LIKE '[0-9][0-9][0-9][0-9]'

Note Make sure to enclose any non-numeric constraint values in single quotation marks ('). For additional details, see <u>Defining a Check Constraint Expression</u>.

4. If you want to give the constraint a different name, type the name in the **Constraint name** box.

- 5. Use the check boxes to control when the constraint is enforced:
 - To test the constraint on existing data before creating the constraint, check **Check existing data on creation**.
 - To enforce the constraint whenever a replication operation occurs on this table, check **Enforce constraint for replication**.
 - To enforce the constraint whenever a row of this table is inserted or updated, check **Enforce constraint for INSERTs and UPDATEs**.

See Also

<u>Check Constraints</u> | <u>Constraints</u> | <u>Deleting a Check Constraint</u> | <u>Disabling a</u> <u>Check Constraint for Replication</u>

Defining a Check Constraint Expression

When you attach a <u>check constraint</u> to a table or column, you must include an SQL expression. For details about this operation, see <u>Attaching a New Check</u> <u>Constraint to a Table or Column</u>.

You can create a simple constraint expression to check data for a simple condition; or you can create a complex expression, using Boolean operators, to check data for several conditions. For example, suppose the authors table has a zip column where a 5-digit character string is required. This sample constraint expression guarantees that only 5-digit numbers are allowed:

zip LIKE '[0-9][0-9][0-9][0-9]'

Or suppose the sales table has a column called qty which requires a value greater than 0. This sample constraint guarantees that only positive values are allowed:

qty > 0

Or suppose the orders table limits the type of credit cards accepted for all credit card orders. This sample constraint guarantees that if the order is placed on a credit card, then only Visa, MasterCard, or American Express is accepted:

```
NOT (payment_method = 'credit card') OR
(card_type IN ('VISA', 'MASTERCARD', 'AMERICAN EXPRESS')
```

To define a constraint expression

- 1. Create a new check constraint. For details on how to do this, see <u>Attaching a New Check Constraint to a Table or Column</u>.
- 2. In the Check Constraints tab of the property pages, type an expression in the Constraint expression box using the following syntax: {constant | column_name | function | (subquery)} [{operator | AND | OR | NOT}

{constant | column_name | function | (subquery)}...]

The SQL syntax is made up of the following parameters:

Parameter	Description
constant	A literal value, such as numeric or character data. Character data must be enclosed within single quotation marks (').
column_name	Specifies a column.
function	A built-in function.
operator	An arithmetic, bitwise, comparison, or string operators.
AND	Use in Boolean expressions to connect two expressions. Results are returned when both expressions are true.
	When AND and OR are both used in a statement, AND is processed first. You can change the order of execution by using parentheses.
OR	Use in Boolean expressions to connect two or more conditions. Results are returned when either condition is true.
	When AND and OR are both used in a statement, OR is evaluated after AND. You can change the order of execution by using parentheses.
NOT	Negates any Boolean expression (which can include keywords, such as LIKE, NULL, BETWEEN, IN, and EXISTS).
	When more than one logical operator is used in a statement, NOT is processed first. You can change the order of execution by using parentheses.

See Also

Check Constraints | Constraints

Checking Existing Data When Creating a Check Constraint

When you create a check constraint, you can set an option to apply it either to new data only or to existing data as well. The option of applying the constraint to new data only is useful when you know that the existing data already meets the new check constraint, or when a business rule requires the constraint to be enforced only from this point forward.

For example, you may have required zip codes to be limited to five digits in the past, but now want new data to allow nine-digit zip codes. Old data with five-digit zip codes will coexist with new data that contains nine-digit zip codes.

To check existing data when creating a check constraint

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Check Constraints tab**.
- 3. Select the constraint from the **Selected constraint** list.
- 4. Select the **Check existing data on creation** check box. This option is selected by default.

The check constraint will be applied when you save the table or the database diagram. If any constraint violations are encountered during the save process, the table cannot be saved.

See Also

Attaching a New Check Constraint to a Table or Column | Check Constraints | Constraints

Disabling a Check Constraint for Replication

You can disable a check constraint when your <u>table</u> is replicated in another database. When you replicate a table, the table definition and data are copied from the source database to a destination database. These two databases are usually (but not necessarily) on separate servers. If the check constraints are specific to the source database, they may unnecessarily prevent new data from being entered in the destination database. When you replicate a database at a remote site, you should not reapply check constraints because:

- The integrity of the data was checked when it was entered into the original database.
- The replication will fail if data violates the check constraints.

To disable a check constraint for replication

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Check Constraints tab**.
- 3. Select the constraint from the **Selected constraint** list.
- 4. Clear the **Enforce constraint for replication** check box.

See Also

<u>Attaching a New Check Constraint to a Table or Column | Check Constraints |</u> <u>Constraints</u>

Disabling a Check Constraint with INSERT and UPDATE Statements

You can disable a check constraint when data is added to, updated in, or deleted from a <u>table</u>. Disabling a constraint enables you to perform the following transactions:

- Add a new row of data to a table (using the INSERT statement) where the existing rows were required to meet specific business rules that no longer apply. For example, you may have required postal codes to be limited to five digits in the past, but now want new data to allow nine-digit postal codes. Old data with five-digit postal codes will coexist with new data that contains nine-digit postal codes.
- Modify existing rows (using the UPDATE statement) where the existing rows were required to meet specific business rules that no longer apply. For example, you may want to update all existing five-digit postal codes to nine-digit postal codes.

Select the option to disable a check constraint during INSERT and UPDATE transactions if you know that new data will violate the constraint, or if the constraint applies only to the data already in the database.

To disable a check constraint with INSERT and UPDATE statements

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Check Constraints** tab.
- 3. Select the constraint from the **Selected constraint** list.

4. Clear the **Enforce constraint for INSERTs and UPDATEs** check box.

You can select this option after you add or modify data to guarantee that the constraint applies to subsequent data modifications.

See Also

<u>Attaching a New Check Constraint to a Table or Column | Check Constraints |</u> <u>Constraints</u>

Modifying a Check Constraint

Modify a <u>check constraint</u> when you want to change the constraint <u>expression</u> or the options that enable or disable the constraint for specific conditions.

To modify a check constraint

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Check Constraints** tab.
- 3. Select the constraint you want to change from the **Selected constraint** list.
- 4. Complete an action from the following table:

То	Follow these steps	
Edit the constraint	Type the new expression in the Constraint	
expression	expression box. For details, see <u>Defining a</u>	
	Check Constraint Expression.	
Rename the constraint	Type a new name in the Constraint name	
	box.	
Apply the constraint	Select the Check existing data on	
to existing data	creation option. For details, see <u>Checking</u>	
	Existing Data When Creating a Check	
	<u>Constraint</u> .	
Disable the constraint	Clear the Enforce constraint for	
when new data is	INSERTs and UPDATEs option. For	
added to the table or	details, see <u>Disabling a Check Constraint</u>	

when existing data is	with INSERT and UPDATE Statements.
updated in the table	
Disable the constraint	Clear the Enforce constraint for
when the table is	replication option. For details, see
replicated in another	Disabling a Check Constraint for
database	Replication.

The constraint is updated in the database when you save your table or diagram.

See Also

Check Constraints | Constraints

Deleting a Check Constraint

Delete a <u>check constraint</u> when you want to remove the limitations on data values that are accepted in the <u>column</u> or columns included in the constraint expression.

To delete a check constraint

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Check Constraints tab**.
- 3. Select the constraint from the **Selected constraint** list.

4. Choose **Delete**.

Caution Choosing **Delete** will result in an action that cannot be undone without losing all other changes made to the database diagram. To undo this action, close this database diagram and all other open database diagrams without saving the changes.

The constraint is deleted from the database when you save the table or diagram.

See Also

<u>Attaching a New Check Constraint to a Table or Column</u> | <u>Check Constraints</u> | <u>Constraints</u>

Creating a Unique Constraint

Create a <u>unique constraint</u> to ensure no duplicate values are entered in specific <u>columns</u> that do not participate in a <u>primary key</u>. While both a unique constraint and a primary key enforce uniqueness, you should attach a unique constraint instead of a primary key constraint to a <u>table</u> if:

- You want to enforce uniqueness in a column or combination of columns. You can attach multiple unique constraints to a table, whereas you can attach only one primary key constraint to a table.
- You want to enforce uniqueness in a column that allows null values. You can attach unique constraints to columns that allow null values, whereas you can attach primary key constraints only to columns that do not allow null values. When you attach a unique constraint to a column allowing null values, you ensure that at most one row will have a null value in the constrained column.

To create a unique constraint

1. In your database diagram, right-click the table that will contain the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table that will contain the constraint, right-click in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. Choose **New**. A system-assigned name appears in the **Index name** box.
- 4. Under **Column name**, expand the list of columns and select the column that you want to attach the constraint to. To attach the

constraint to multiple columns, select the additional columns in subsequent rows.

- 5. Select the **Create UNIQUE** check box.
- 6. Select the **Constraint** option.

The unique constraint is created in the database when you save the table or the diagram.

If you are using SQL Server, you can control the sort order of key values and the action taken when duplicate keys exist. To do this, you should create a unique <u>index</u> instead of a unique constraint. For more information, see <u>Creating a</u> <u>Unique Index</u>.

See Also

<u>Constraints</u> | <u>Creating a Unique Index</u> | <u>Deleting a Unique Constraint</u> | <u>Unique</u> <u>Constraints</u>

Modifying a Unique Constraint

Modify a <u>unique constraint</u> when you want to change the <u>columns</u> that the constraint is attached to, change the constraint name, or set additional properties for the constraint.

To modify a unique constraint

1. In your database diagram, right-click the table containing the constraint, then select **Properties** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Properties** from the shortcut menu.

- 2. Choose the **Indexes/Keys** tab.
- 3. Select the constraint you want to change from the **Selected index** list.
- 4. Complete an action from the following table:

То	Follow these steps
Change the columns	In the Column name grid, expand the list
that the constraint is	and select the columns you want to attach
attached to	the constraint to.
Rename the constraint	Type a new name in the Index name box.
	Make sure that your new name does not
	duplicate a name in the Selected index list.
Set the clustered	Select the Create as CLUSTERED check
option	box. Only one clustered index can exist per
	table. If a clustered index already exists,
	you must first clear this setting on that
	index.
Define a <u>fill factor</u>	Type an integer from 0 to 100 in the Fill

factor box.

The constraint is updated in the database when you save your table or diagram.

See Also

Constraints | Unique Constraints

Deleting a Unique Constraint

Delete a <u>unique constraint</u> when you want to remove the requirement for uniqueness for values entered in the <u>column</u> or combination of columns included in the constraint <u>expression</u>.

To delete a unique constraint

1. In your database diagram, right-click the table containing the constrained column or columns, then select **Indexes/Keys** from the shortcut menu.

-or-

Open the Table Designer for the table containing the constraint, rightclick in the Table Designer, and choose **Indexes/Keys** from the shortcut menu.

- 2. Select the unique constraint from the **Selected index** list.
- 3. Choose **Delete**.

Caution Choosing **Delete** will result in an action that cannot be undone without losing all other changes made to the database diagram. To undo this action, close this database diagram or table design window and all other open database diagrams and table design windows without saving the changes.

The constraint is deleted from the database when you save the table or diagram.

See Also

Constraints | Creating a Unique Constraint | Unique Constraints

Working with User-Defined Functions

You can employ user-defined functions in a number of places in your applications.

For details about working with user-defined functions, see the following topics:

То	See
Include data from a user-	Including User-Defined Functions in a Query
defined function in a query	

Working with Scripts

You can preserve your work in SQL Scripts. In so doing, you can retain your work without affecting the underlying database. Later, you or another user can run the scripts to affect the database.

То	See
Create a script capturing all the	Saving a Change Script
changes you made since you opened	
the database diagram or last saved	
your work to the database	

Saving a Change Script

You can save an SQL <u>change script</u> if you do not have security permissions to alter a database or if you are not ready to update the database (for example, if you have made changes to the database diagram that conflict with changes made by other users). An <u>SQL</u> change script provides a record of your changes that can be applied to the database at a later time using a database tool (for example, the Microsoft[®] SQL Server[™] command-line utility **osql**).

To save a change script

- 1. Right-click inside the database diagram, and then click **Save Change Script**. This command is available whenever you have unsaved database changes in your diagram.
- 2. In the **Save Change Script** dialog box, choose **Yes**.

Note If the option to automatically generate a change script is selected, a change script is generated whenever you save your database diagram or any changed database object in the diagram. This is helpful if you need to track the changes you have made to the database.

3. A message box displays the file name of the saved change script. Choose **OK**.

Each time you save a change script, a new text file named DbDgm*N*.sql (where *N* equals 1 for the first change script you generate and N+1 for each subsequent change script) is created and saved in the current working directory.

The change script file lists any changed tables and how they were changed (using the DROP TABLE, ALTER TABLE, or CREATE TABLE SQL statements). The change script file also contains any error handling code that is required to clean up temporary tables or to rollback transactions that were part of unsuccessful changes (changes that could not be saved). In addition, the change script file contains code to make the script run successfully against the database and code (Print statements) that describe what the script is doing when it runs.

Any error messages that occurred while the changes were saved are stored in a log file, with the same name as the script file, except with a .log extension. You can also view messages showing which tables were successfully and unsuccessfully saved in the Log Viewer.

See Also

Database Changes Detected Dialog Box | Saving Selected Tables

Designing Data Retrieval and Manipulation

The Microsoft® Visual Database Tools Query Designer and View Designer can help you create and maintain the data retrieval and data manipulation portions of your application.

For more information about designing the data retrieval and manipulation portions of an application, see <u>Database Queries and the Visual Database Tools</u>.

For details about the specific steps in designing data retrieval and manipulation solutions, see the following topics:

- **Designing Queries**
- <u>Working with Views</u>
- Manipulating Data

Designing Queries

Query design is an important part of developing database applications, because after the database is designed and populated with data, it is through queries that you put that data to use. To learn more about the specific steps of query design, see the following topics:

- <u>Performing Basic Operations with Queries</u>
- Specifying Search Criteria
- <u>Querying on Groups of Rows</u>
- <u>Querying Using Multiple Tables</u>
- <u>Creating General Purpose Queries</u>
- <u>Using Expressions in Queries</u>
- Including User-Defined Functions in a Query

Performing Basic Operations with Queries

The following fundamental operations apply to most or all queries.

То	See
Create a new query	Creating a Query
Add a table, view, or any other	Adding Tables
column source to a query	
Remove a column source from a	Removing Tables
query	
Add a column to a query	Adding Columns
Remove a column from the query	Removing Columns from Query
output	Output
Remove a column from the query	Removing Columns from the Query
Change the order of columns in	Reordering Output Columns
query output	
Create column aliases in a query	Creating Column Aliases
Create table aliases in a query	Creating Table Aliases
Verify a query	Verifying a Query
Add comments to a query	Commenting a Query
Execute a query	Executing a Query
Stop a query	Stopping a Query
Clear query results	Clearing Query Results
Print query results	Printing Query Results
Order result rows of a query	Ordering Query Results

Creating a Query

To create a new query

- 1. In the SQL Server Enterprise Manager, right-click any table or view that will be included in the query.
- 2. On the shortcut menu, point to **Open Table** and choose **Query**, or point to **Open View** and choose **Query**.

The Query Designer appears with the table or view already present on the Diagram Pane.

See Also

Performing Basic Operations with Queries

Adding Tables

When you create a query, you are retrieving data from a table or other objects structured like tables — views and certain user-defined functions. To work with any of these objects in your query, you add them to the <u>Diagram pane</u>.

Note For information about adding tables or objects structured like tables from different data sources while working with SQL Server, see <u>Query</u> <u>Designer Considerations for SQL Server Databases</u>.

To add a table, view, or user-defined function to the query

- 1. In the Diagram pane of the Query Designer or View Designer, rightclick the background and choose **Add Table** from the shortcut menu.
- 2. In the **Add Table** dialog box, select the tab for the type of object you want to add to the query.
- 3. In the list of items, double-click each item you want to add.
- 4. When you finish adding items, click **Close**.

The Query Designer updates the Diagram pane, Grid pane, and SQL pane accordingly.

Alternatively, you can drag objects onto the Diagram pane. You can drag a table, view, or user-defined function from the SQL Server Enterprise Manager.

You can also drag columns or tables from the Database Designer or paste them from the Clipboard.

Tables and views are automatically added to the query when you reference them in the statement in the SQL pane.

The Query Designer will not display data columns for a table or table-structured object if you do not have sufficient access rights to it or if the <u>ODBC driver</u>

cannot return information about it. In such cases, only a title bar and the * (All Columns) check box are displayed for the table or table-structured object.

To add an existing query to a new query

- 1. Make sure the SQL pane is displayed in the new query you are creating.
- 2. In the SQL pane, type a right and left parentheses () after the word FROM.
- 3. Open the Query designer for the existing query . (You now have two Query Designers open.)
- 4. Display the SQL pane for the inner query the existing query you are including in the new, outer query.
- 5. Select all the text in the SQL pane, and copy it to the Clipboard.
- 6. Click in the SQL pane of the new query, situate the cursor between the parentheses you added, and paste the contents of the Clipboard.
- 7. Still in the SQL pane, add an alias after the right parenthesis.

See Also

Removing Tables

Removing Tables

You can remove a table — or any table-structured object — from the query.

Note Removing a table or table-structured object does not delete anything from the database; it only removes it from the current query. For details about removing a table from a database, see <u>Deleting a Table from a</u> <u>Database Diagram and the Database</u>.

To remove a table or table-structured object

• In the <u>Diagram pane</u>, select the table, view, user-defined function, or query, and then press DELETE, or right-click the object and then choose **Remove**. You can select and remove multiple objects at one time.

–or–

• Remove all references to the object in the SQL pane.

When you remove a table or table-structured object, the Query Designer automatically removes joins that involve that table or table-structured object and removes references to the object's columns in the <u>SQL</u> and <u>Grid panes</u>. However, if the query contains complex <u>expressions</u> involving the object, the object is not automatically removed until all references to it are removed.

See Also

Adding Tables

Adding Columns

To use a column in a query, you must add it to the query. You might add a column to display it in query output, to use it for sorting, to search the contents of the column, or to summarize its contents.

If you are creating a <u>Select query</u> and add a column in the <u>Diagram</u> or <u>Grid</u> panes, the column you add becomes part of the query output. You can remove the column from the output and still use it for sorting, searching, and so on. For example, to find all employees in the accounting department, you might search the department column but not display it in the output.

Tip Wherever you use a column in a query, you can also use an <u>expression</u> that can consist of any combination of columns, literals, operators, and functions. For details, see <u>Using Expressions in Queries</u>.

You can add columns individually or as a group. Your choices are:

- An individual column from one table or table-structured object, to use for sorting, searching, or summarizing.
- All columns from one table or table-structured object. For tables, this is the equivalent of specifying "*tablename*.*" in the SQL statement.
- All columns from all table-structured objects in the query, which can be useful if you are working with joins. This option is the equivalent of specifying "*" alone in the SQL statement when more than one table or table-structured object is used in the query.
 When you add all columns, the Query Designer does not add all individual columns to the query; it instead uses the asterisk ("*"). If you need to work with a specific column, you must add it separately.

Note When you choose "*", all currently defined columns for the tables you are using are included. If a table definition changes, the list of columns returned by "*" changes as well. It is recommended that, if practical, you specify the columns you want to work with instead of using "*".

To add an individual column

• In the **Diagram** pane, select the check box next to the data column that you want to include.

-or-

• In the **Grid** pane, move to the first blank grid row where you want to add the column, click the field in the **Column** column, and select a column name from the list.

Note To add a data row at a specific location in the **Grid** pane, select the grid row where you want to add the new column and press **INS**. A new column is added above that row.

To add all columns for one table or table-structured object

In the **Diagram** pane, select the check box ✓ next to **(All Columns)**.

-or-

• Specify *objectname*.* in the SQL statement in the <u>SQL pane</u>, substituting the name of your table or table-structured object for *objectname*.

To add all columns for all tables and table-structured objects

- 1. Make sure no join lines in the **Diagram** pane are selected.
- 2. Right-click in the query window and choose **Properties** from the shortcut menu. Then choose the <u>Query tab</u>.

3. Select **Output all columns**.

-0r-

Specify ***** in the output list of the SQL statement in the <u>SQL pane</u>.

See Also

Creating Column Aliases | Removing Columns from Query Output | Removing

Columns from the Query | Reordering Output Columns

Removing Columns from Query Output

If you are using a column in a <u>Select query</u> but do not want to display it in the <u>result set</u> (that is, you do not want it in the query's select list), you can remove it from output. After you remove the column from the query's output, you can still use it in search conditions or as a sorting field.

Note If you want to remove a column from the query altogether, see <u>Removing Columns from the Query</u>.

To remove a column from the query output

• In the <u>Diagram pane</u>, clear the check box next to the name of the column you want to remove.

-or-

• In the <u>Grid pane</u>, clear the check box in the **Output** column for the data column you want to remove. (If you want to add the column back to the query output, you can check the **Output** column again.)

-or-

• Remove the column from the output list in the <u>SQL pane</u>.

See Also

Adding Columns | Using Expressions in Queries

Removing Columns from the Query

If you no longer want to use a column in a query, you can remove it. If you do, the Query Designer removes references to the column in the select list, the sort specification, the search criteria, SQL pane, and any grouping specifications.

Note If you want to remove a column from just the output of a <u>Select query</u>, you can do so without removing it from the query altogether. For details, see <u>Removing Columns from Query Output</u>.

To remove a column from the query

• In the <u>Grid pane</u>, select the grid row containing the column you want to remove and then press DELETE.

-or-

• Remove all references to the column in the <u>SQL pane</u>.

See Also

Adding Columns

Reordering Output Columns

The order in which you add data columns to a <u>Select query</u> determines the order in which they appear in the results. The first column you add to the query appears leftmost in the results, the second column next, and so on.

If you are creating <u>Update</u> or <u>Insert</u> queries, the order in which you add columns affects the order in which data is processed.

To control where a data column appears in the <u>result set</u>, or in what order it is used, you can reorder the columns.

To reorder columns for output

In the <u>Grid pane</u>, select the row containing the column by clicking the row selector button to the left of the row .

4. Point to the row selector button and drag the row to a new location.

-or-

• Edit the order of the column names in the <u>SQL pane</u>.

Tip You can add a data row at a specific location in the Grid pane by inserting a blank row into the Grid pane, and then specifying the data column to insert. For details, see <u>Adding Columns</u>.

See Also

Adding Columns

Creating Column Aliases

You can create <u>aliases</u> for column names to make it easier to work with column names, calculations, and summary values. For example, you can create a column alias to:

- Create a column name, such as "Total Amount," for an <u>expression</u> such as (quantity * unit_price) or for an <u>aggregate function</u>.
- Create a shortened form of a column name, such as "d_id" for "discounts.stor_id."

After you have defined a column alias, you can use the alias in a <u>Select query</u> to specify query output.

To create a column alias

- 1. In the **Grid** pane, locate the row containing the data column for which you want to create an alias, and if necessary, mark it for output. If the data column is not already in the grid, add it.
- 2. In the **Alias** column for that row, enter the alias. The alias must follow all naming conventions for SQL. If the alias name you enter contains spaces, the Query Designer automatically puts delimiters around it.

See Also

Adding Columns

Creating Table Aliases

<u>Aliases</u> can make it easier to work with table names. Using aliases is helpful when:

- You want to make the statement in the <u>SQL pane</u> shorter and easier to read.
- You refer to the table name often in your query such as in qualifying column names and want to be sure you stay within a specific character-length limit for your query. (Some databases impose a maximum length for queries.)
- You are working with multiple instances of the same table (such as in a <u>self-join</u>) and need a way to refer to one instance or the other.

For example, you can create an alias "e" for a table name employee_information, and then refer to the table as "e" throughout the rest of the query.

To create an alias for a table or table-structured object

- 1. Add the table or table-structured object to your query.
- 2. In the Diagram Pane, right-click the object for which you want to create an alias, then select **Properties** from the shortcut menu.
- 3. In the **Properties** dialog box, enter the alias in the **Alias** box.

When you create a table alias, the Query Designer substitutes the alias for the corresponding table name in the Table column of the Grid pane.

Note The SQL standard specifies that when you create an alias for a table name, you must use the alias to refer to the table in the rest of the SQL

statement (that is, you cannot use the original table name).

Verifying a Query

To avoid problems, you can check the query you have built to ensure its syntax is correct. This option is especially useful when you enter statements in the <u>SQL</u> <u>pane</u>.

Note A statement can be valid, and therefore be verified successfully, even if it cannot be represented in the <u>Diagram</u> and <u>Grid</u> panes.

Note SQL Verification can detect some, but not all SQL errors. If a query contains an error not detected during SQL verification, the database will detect the error when you run the query.

To verify an SQL statement

Right-click in the **SQL** pane, and select **Verify SQL Syntax** from the shortcut menu.

See Also

Executing a Query

Commenting a Query

You can document a query or view by adding a comment to it. You can enter a comment with a property page or in the SQL pane.

You should avoid entering the two-character strings "/*" and "*/" within your comments.

If you change the type of query (for example, from a SELECT query to an UPDATE query), the query's comment is deleted.

If you add a comment to a view, the text that you add is returned by the <u>sp_help</u> stored procedure.

To add a comment to a query with the property page

- 1. Right-click in the Query Designer or View Designer, then select **Property Pages** from the shortcut menu.
- 2. If you are working on a query, select the Query tab. If you are working on a View, select the View tab.
- 3. Enter the comment in the **SQL Comment** field.

To add a comment to a query in the SQL pane

- Situate the text cursor at the beginning or end of the SQL pane. Enter the comment, preceded by "/*" and followed by "*/". You must situate the cursor at the beginning or end of the SQL pane because you cannot enter embedded comments.
- 2. Click in the Diagram pane or Grid pane. The comment is automatically removed from the SQL pane and appended to the existing comment text in the **SQL Comment** field on the Query or View property page.

Executing a Query

When you have finished designing your query, you can run it.

Note If you want to test whether the syntax of the query you are creating is correct, you can verify the query. For details, see <u>Verifying a Query</u>.

To execute a query

• Right-click anywhere in the query window, and select **Run** from the shortcut menu.

If you are creating a <u>Select query</u>, the results of the query appear in the <u>Results</u> <u>pane</u>.

The Query Designer returns results to your computer in batches (incrementally) so that you can begin viewing results as soon as possible, and so that you can perform other tasks while the query is underway. For more information, see Interaction Between the Results Pane and the Database.

If you are creating an Update, Insert From, Insert Into, Delete, or Make Table query, the Query Designer displays a message indicating how many rows were affected by the query.

For information on viewing results and navigating in the Results pane, see <u>Results Pane</u>.

See Also

Clearing Query Results | Editing Rows in the Results Pane | Verifying a Query

Stopping a Query

If you see that the query is taking too long or is not returning the results you expect, you can stop the query if it has not already finished.

To stop a query

• Right-click anywhere in the <u>Results pane</u>, and then choose **Clear Results**.

The Query Designer also stops the current query if you execute a new query.

See Also

Clearing Query Results | Executing a Query

Clearing Query Results

If you change the current query definition, the Query Designer dims the Results pane to indicate that the contents of the pane no longer reflect the query you are editing. You can also clear the contents of the Results pane.

To clear query results

• Right-click in the <u>Results pane</u>, and then choose **Clear Results**.

If a query is being executed when you clear the Results pane, the Query Designer stops the query.

See Also

Executing a Query | Stopping a Query

Printing Query Results

To print the results of your query, you can copy the contents of the <u>Results pane</u> to another Windows program, such as a word processing program, then format and print it there.

The Query Designer puts query results onto the Clipboard using tabs as delimiters between columns and carriage return and linefeed characters as delimiters between rows.

To print query results

- 1. Select the columns or rows that you want to print.
- 2. Use the **Copy** command to move them to the Clipboard.
- 3. Switch to the Windows program you want to use to print the results.
- 4. Use the **Paste** command to move the query results from the Clipboard.
- 5. Format and print the results.

See Also

Page Setup Dialog Box

Ordering Query Results

You can order your query by the contents of a data column or by an <u>expression</u>. The Query Designer allows you to specify the sort types ascending and descending.

You can also order the query results by more than one column or expression and specify the sort order for each. For example, if you are querying an employee table, you can order the results by department (sort order = 1) and within each department by last name (sort order = 2).

Note You cannot sort by the contents of a <u>memo</u> or binary (<u>BLOB</u>) column.

To order query results

- If you have not done so already, add the columns or expressions that you want to sort by to the <u>Grid pane</u>.
 If you do not want the columns or expressions to be part of the result set, remove them as output columns.
- 2. In the **Grid** pane, locate the row containing the first data column or expression to sort by, and then in the **Sort Type** grid column, choose **Ascending** or **Descending**.

	Column	Table	Output	Sort Type	Sort Order
	emp_id	employees	\checkmark	a 6,757.8	
	fname	employees	\checkmark		
a. 7.	minit	employees	\checkmark		
	Iname	employees	\checkmark		
	hire_date	employees	\checkmark	Ascending	1
. 1					
•					

3. If you are sorting by multiple columns or expressions, specify the sort order in the **Sort Order** column of the grid.

-0r-

• In the <u>Diagram pane</u>, right-click the column to sort by and then choose **Sort Ascending** or **Sort Descending** from the shortcut menu. You can

select multiple columns before right-clicking them. If you use this method, the columns are sorted in the order you select them.

See Also

Collapsing Groups of Rows

Creating Queries

The Query Designer features an easy-to-use interface for working with your queries. The topics in this section describe the fundamentals of working with any query, including how to choose a table or table-structured object to work with, how to add or remove data columns from the query, and how to execute the query.

То	See
Add or remove tables or table-	Adding Tables Removing Tables
structured objects from a query	
Add or remove columns from a	Adding Columns
query or from the query output	Removing Columns from Query
	Output
	Removing Columns from the Query
Change how query results are	Reordering Output Columns
displayed	Ordering Query Results
Clear or print the query results	Clearing Query Results
	Printing Query Results
Create aliases for tables or columns	Creating Table Aliases
	Creating Column Aliases
Check the syntax of a query	Verifying a Query
To run or stop a query	Executing a Query
	Stopping a Query

Specifying Search Criteria

You can use search criteria to restrict the number of rows returned by a query. For more information about choosing rows for inclusion in a result set, see <u>Including or Excluding Rows</u>.

For details about the particular steps to creating search criteria, refer to the topics listed in the following table.

For information about	See
Specifying <u>search conditions</u> in the	Specifying Search Conditions
Query Designer	
Creating expressions that you can use	Using Expressions in Queries
in search conditions	
Using operators in search conditions	Comparison Operators, Logical
	Operators, and Wildcard Characters
Entering text, numbers, dates, or	Entering Search Values
logical values	
Finding rows that do not match a	Selecting Rows that Do Not Match a
value	Value
Removing duplicate rows from	Excluding Duplicate Rows
Select queries	
Applying multiple search conditions	Specifying Multiple Search
to the same data column	Conditions for One Column
Including several data columns as	Specifying Multiple Search
part of the search condition for a	Conditions for Multiple Columns
query	
Linking search conditions with AND	Combining Search Conditions
and OR operators	
Using <u>subqueries</u>	Creating Subqueries

Specifying Search Conditions

You can specify the data rows that appear in your query by specifying <u>search conditions</u>. For example, if you are querying an employee table, you can specify that you want to find only the employees who work in a particular region.

You specify search conditions using an <u>expression</u>. Most commonly the expression consists of an operator and a search value. For example, to find employees in a particular sales region, you might specify the following search criterion for the region column:

='UK'

Note If you are working with multiple tables, the Query Designer examines each search condition to determine whether the comparison you are making results in a join. If so, the Query Designer automatically converts the search condition into a join. For more information, see Joining Tables Automatically.

To specify search conditions

- If you have not done so already, add the columns or expressions that you want to use within your search condition to the <u>Grid pane</u>. If you are creating a <u>Select query</u> and do not want the search columns or expressions to appear in the query output, clear the **Output** column for each search column or expression to remove them as output columns.
- 2. Locate the row containing the data column or expression to search, and then in the **Criteria** grid column, enter a search condition.

Note If you do not enter an operator, the Query Designer automatically inserts the equality operator "=".

The Query Designer updates the SQL statement in the <u>SQL pane</u> by adding or

modifying the WHERE clause.

See Also

Wildcard Characters | Including or Excluding Rows

Selecting Rows that Do Not Match a Value

To find <u>rows</u> that do not match a value, use the NOT operator. For example, to find all the rows in a **products** table where the values in the product code column begin with a character other than "A," you can enter a search condition such as the following:

NOT LIKE 'A%'

See Also

Entering Search Values | Specifying Search Conditions | Specifying Search Criteria | Including or Excluding Rows

Specifying Multiple Search Conditions for One Column

In some instances, you might want to apply a number of <u>search conditions</u> to the same data column. For example, you might want to:

- Search for several different names in an employee table or for employees who are in different salary ranges. This type of search requires an OR condition.
- Search for a book title that both starts with the word "The" and contains the word "Cook." This type of search requires an AND condition.

Note The information in this topic applies to search conditions in both the WHERE and HAVING clauses of a query. The examples focus on creating WHERE clauses, but the principles apply to both types of search conditions. For details about creating HAVING clauses, see <u>Specifying Conditions for Groups</u>.

To search for alternative values in the same data column, you specify an OR condition. To search for values that meet several conditions, you specify an AND condition.

Specifying an OR Condition

Using an OR condition enables you to specify several alternative values to search for in a column. This option expands the scope of the search and can return more rows than searching for a single value.

Tip You can often use the IN operator instead to search for multiple values in the same data column. For details, see <u>Comparison Operators</u>.

To specify an OR condition

1. In the <u>Grid pane</u>, add the column to search.

- 2. In the **Criteria** column for the data column you just added, specify the first condition.
- 3. In the **Or** ... column for the same data column, specify the second condition.

Column	Criteria	0r
salary	< 30000	> 100000

The Query Designer creates a WHERE clause that contains an OR condition such as the following:

SELECT fname, lname FROM employees WHERE (salary < 30000) **OR** (salary > 100000)

Specifying an AND Condition

Using an AND condition enables you to specify that values in a column must meet two (or more) conditions for the row to be included in the <u>result set</u>. This option narrows the scope of the search and usually returns fewer rows than searching for a single value.

Tip If you are searching for a range of values, you can use the BETWEEN operator instead of linking two conditions with AND. For details, see <u>Comparison Operators</u>.

To specify an AND condition

- 1. In the <u>Grid pane</u>, add the column to search.
- 2. In the **Criteria** column for the data column you just added, specify the first condition.
- 3. Add the same data column to the **Grid** pane again, placing it in an empty row of the grid.

4. In the **Criteria** column for the second instance of the data column, specify the second condition.

Criteria		
LIKE '%Cook%'		
LIKE '%Recipe%'		

The Query Designer creates a WHERE clause that contains an AND condition such as the following:

SELECT title_id, title FROM titles WHERE (title LIKE '%Cook%') **AND** (title LIKE '%Recipe%')

See Also

<u>Combining Search Conditions | Specifying Multiple Search Conditions for</u> <u>Multiple Columns</u>

Specifying Multiple Search Conditions for Multiple Columns

You can expand or narrow the scope of your query by including several data columns as part of your <u>search condition</u>. For example, you might want to:

- Search for employees who either have worked more than five years at the company or who hold certain jobs.
- Search for a book that is both published by a specific publisher and pertains to cooking.

To create a query that searches for values in either of two (or more) columns, you specify an OR condition. To create a query that must meet all conditions in two (or more) columns, you specify an AND condition.

Note If you are creating queries that include combinations of AND and OR clauses, you must be aware of how the query is interpreted when you execute it. For details, see <u>Combining Search Conditions</u>.

Specifying an OR Condition

To create multiple conditions linked with OR, you put each separate condition in a different column of the <u>Grid pane</u>.

To specify an OR condition for two different columns

- 1. In the **Grid** pane, add the columns you want to search. For details, see <u>Adding Columns</u>.
- 2. In the **Criteria** column for the first column to search, specify the first condition. For details, see <u>Specifying Search Conditions</u>.
- 3. In the **Or** ... column for the second data column to search, specify the second condition, leaving the **Criteria** column blank.

Column	Criteria	Or
job_lvl	>= 200	
hire_date		< '01 Jan 1990'
	_	

The Query Designer creates a WHERE clause that contains an OR condition such as the following:

SELECT job_lvl, hire_date FROM employee WHERE (job_lvl >= 200) **OR** (hire_date < '01/01/90')

4. Repeat Steps 3 and 4 for each additional condition you want to add. Use a different **Or** ... column for each new condition.

Specifying an AND Condition

To search different data columns using conditions linked with AND, you put all the conditions in the **Criteria** column of the grid.

To specify an AND condition for two different columns

- 1. In the <u>Grid pane</u>, add the columns you want to search.
- 2. In the **Criteria** column for the first data column to search, specify the first condition.
- 3. In the **Criteria** column for the second data column, specify the second condition.

3-1	Column	Criteria	Or
	pub_id	= '0877'	
	title	LIKE '%Cook%'	
_			

The Query Designer creates a WHERE clause that contains an AND condition such as the following:

SELECT pub_id, title FROM titles WHERE (pub_id = '0877') **AND** (title LIKE '%Cook%') 4. Repeat Steps 2 and 3 for each additional condition you want to add.

See Also

Combining Conditions when AND Has Precedence | Combining Conditions when OR Has Precedence | Combining Search Conditions | Comparison Operators | Specifying Search Criteria

Combining Conditions when AND Has Precedence

To combine conditions with AND, you put the conditions in the same column of the Grid pane. To combine conditions with OR, you put the first one in the Criteria column and additional conditions into an Or ... column.

For example, imagine that you want to find either employees who have been with the company for more than five years in lower-level jobs or employees with middle-level jobs regardless of their hire date. This query requires three conditions, two of them linked with AND:

• Employees with a hire date earlier than five years ago and with a job level of 100

-or-

• Employees with a job level of 200

The following procedure illustrates how to create this type of query in the Grid pane.

To combine conditions when AND has precedence

- 1. In the **Grid** pane, add the data columns you want to search. If you want to search the same column using two or more conditions linked with AND, you must add the data column name to the grid once for each value you want to search.
- 2. In the **Criteria** column, enter all the conditions that you want to link with AND. For example, to link conditions with AND that search the hire_date and job_lvl columns, enter values as shown here:

Column	Table	Output	Criteria	Or
hire_date	employee		< '1/1/91'	
job_lvl	employee		= 100	

These grid entries produce the following WHERE clause in the statement in the <u>SQL pane</u>:

WHERE (hire_date < '01/01/91') **AND** (job_lvl = 100)

3. In the **Or** ... grid column, enter conditions that you want to link with OR. For example, to add a condition that searches for another value in the job_lvl column, enter an additional value as shown here:

	Column	Table	Output	Criteria	0r	
	hire_date	employee		< '1/1/91'		
	job_lvl	employee		= 100	= 200	
-						

Adding a value in the **Or** ... column adds another condition to the WHERE clause in the statement in the **SQL** pane:

WHERE (hire_date < '01/01/91') **AND** (job_lvl = 100) **OR** (job_lvl = 200)

See Also

<u>Combining Conditions when OR Has Precedence | Combining Search</u> <u>Conditions | Creating an Expression | Entering Search Values | Specifying Search</u> <u>Criteria</u>

Combining Conditions when OR Has Precedence

To link conditions with OR and give them precedence over conditions linked with AND, you must repeat the AND condition for each OR condition.

For example, imagine that you want to find all employees who have been with the company more than five years and have lower-level jobs or are retired. This query requires three conditions, a single condition linked to two additional conditions with AND:

- Employees with a hire date earlier than five years ago, and
- Employees with a job level of 100 or whose status is "R" (for retired).

The following procedure illustrates how to create this type of query in the Grid pane.

To combine conditions when OR has precedence

- 1. In the **Grid** pane, add the data columns you want to search. If you want to search the same column using two or more conditions linked with AND, you must add the data column name to the grid once for each value you want to search.
- 2. Create the conditions to be linked with OR by entering the first one into the **Criteria** grid column and the second (and subsequent ones) into separate **Or** ... columns. For example, to link conditions with OR that search the job_lvl and status columns, enter values as shown here:

Column	Table	Output	Criteria	Or (
job_lvl	employee		= 100	
status				= 'B'
			di someti	

Entering the values shown in the grid above produces the following WHERE clause in the statement in the SQL pane:

WHERE (job_lvl = 100) OR (status = 'R')

3. Create the AND condition by entering it once for each OR condition. Place each entry in the same grid column as the OR condition it corresponds to. For example, to add an AND condition that searches the hire_date column and applies to both OR conditions, enter values as shown here:

	Column	Table	Output	Criteria	Or
	job_lvl	employee		= 100	
	status				= 'R'
	hire_date	employee		< '1/1/91'	< '1/1/91'
8					

Entering the values shown in the grid above produces the following WHERE clause in the statement in the SQL pane:

WHERE (job_lvl = 100) **AND** (hire_date < '01/01/90') **OR** (status = 'R') **AND** (hire_date < '01/01/91')

Tip You can repeat an AND condition by adding it once, and then using the **Cut** and **Paste** commands from the **Edit** menu to repeat it for other OR conditions.

The WHERE clause created by the Query Designer is equivalent to the following WHERE clause, which uses parentheses to specify the precedence of OR over AND:

WHERE (job_lvl = 100 **OR** status = 'R') **AND** (hire_date < '01/01/91')

Note If you enter the search conditions in the format shown immediately above in the SQL pane, but then make a change to the query in the Diagram or Grid panes, the Query Designer recreates the SQL statement to match the form with the AND condition explicitly distributed to both OR conditions.

See Also

<u>Combining Conditions when AND Has Precedence | Combining Search</u> <u>Conditions | Comparison Operators | Specifying Search Criteria</u>

Creating Subqueries

You can use the results of one query as the input for another. Typically, you use the results of a <u>subquery</u> as a <u>search condition</u> that uses the IN() function or EXISTS operator. However, you can also use a subquery in the FROM clause.

You can create a subquery by entering it in either the Grid pane or SQL pane.

To define an EXISTS subquery in the Grid pane

- 1. Create the primary query.
- 2. In the **Column** column for the first empty row in the **Grid** pane, enter EXISTS followed by the subquery in parentheses.
- 3. In the **Criteria** column for the row containing the subquery, enter TRUE, FALSE, =TRUE, or =FALSE. Entering FALSE or =FALSE results in a NOT EXISTS query.

Note To create a NOT EXISTS query, create an EXISTS query as listed in the above steps, and set the Criteria column to FALSE. If you enter NOT EXISTS in the Grid pane, the Query Designer will display an error.

To define a subquery in the SQL pane

- 1. Create the primary query.
- 2. In the **SQL** pane, select the SQL statement, and then use **Copy** to move the query to the Clipboard.
- 3. Start the new query, and then use **Paste** to move the first query into the new query's WHERE or FROM clause.

For example, imagine you have two tables, products and suppliers, and you want to create a query showing all products for suppliers in Sweden. Create the first query on the suppliers table to find all Swedish suppliers:

```
SELECT supplier_id
FROM supplier
WHERE (country = 'Sweden')
```

Use the Copy command to move this query to the Clipboard. Create the second query using the **products** table, listing the information you need about products:

SELECT product_id, supplier_id, product_name FROM products

In the SQL pane, add a WHERE clause to the second query, then paste the first query from the Clipboard. Place parentheses around the first query, so that the end result looks like this:

```
SELECT product_id, supplier_id, product_name
FROM products
WHERE supplier_id IN
(SELECT supplier_id
FROM supplier
WHERE (country = 'Sweden'))
```

Note When you add a subquery to the WHERE clause, the subquery appears in the **Criteria** column of the Grid pane. You can edit it further in either the Grid pane or SQL pane. However, the tables and table-structured objects, columns, and <u>expressions</u> referenced in the subquery are not displayed in the Diagram or Grid pane.

See Also

Supported Query Types

Using Expressions in a Query

To use an <u>expression</u> in a query, you can type it directly into the Grid pane or you can enter it in the <u>SQL pane</u> as part of the statement. Entering expressions is similar to entering <u>column</u> names. For details about how to create an expression, see <u>Creating an Expression</u>.

Tip To make it easier to see long expressions on your screen, you can resize the columns in the Grid pane. For details, see <u>Grid Pane</u>.

In this topic you can read about:

- Displaying Expressions in the Result Set
- Sorting Using Expressions
- <u>Searching Using Expressions</u>

Displaying Expressions in the Result Set

You can display an expression in the <u>result set</u> by specifying the expression in place of a column in the Grid pane.

To display an expression in the result set

- 1. In the **Grid** pane, insert a new grid row.
- 2. In the **Column** column of the new grid row, type the expression whose results you want to display.

3	Column	Alias	Table	Output
	Iname + ', ' + fname	full_name		V

When you display the results of an expression in the result set, the database assigns a column heading to it using the format "Exprn," where *n* is the number of the expression in the current query. You can replace this with a more meaningful <u>alias</u> for the expression. For details, see <u>Creating Column Aliases</u>.

Sorting Using Expressions

In SQL Server you can sort by the results of an expression. As with columns, you specify the sort type and sort order.

To sort using an expression

- 1. In the **Grid** pane, insert a new grid row.
- 2. In the **Column** column of the new grid row, type the expression you want to sort by.
- 3. If you do not want to display the expression in the query, clear the **Output** column of the new row.
- 4. In the **Sort Type** column, choose **Ascending** or **Descending**, and then in the **Sort Order** column, choose the sort priority for the expression.

Column	Alias	Table	Output	Sort Type	Sort Order
 sales.qty * titles.price	10-			Ascending	1

Searching Using Expressions

There are two ways to use an expression for searching. The expression can be the condition against which you compare values, or it can function as the value you are comparing.

The following example illustrates how you can use an expression as the condition in a WHERE clause:

SELECT ord_num, ord_date FROM sales WHERE (price * .9) > 20

In contrast, the following example illustrates the opposite use of an expression, in which the expression is the value you are comparing:

SELECT ord_num, ord_date FROM sales WHERE (ord_date >= DATEADD(day, -10, GETDATE()))

The way you specify an expression for searching depends on whether it appears as a condition or as a value to search.

To use an expression as a condition

- 1. In the <u>Grid pane</u>, insert a new grid row.
- 2. In the **Column** column of the new grid row, type the expression you want to use as the condition.
- 3. In the **Criteria** column for the new row, type the value to compare against the condition.

Column	Alias	Table	Output	Criteria
price * .9				> 20

To use an expression as a search value

- 1. If it is not already in the **Grid** pane, add the data column or expression you want to search.
- 2. In the **Criteria** column for that data column or expression, enter the expression to use as a search value.

8	Column	Alias	Table	Output	Criteria
	ord_date		sales		>= DATEADD (day, - 10, GETDATE ())

See Also

<u>Creating an Expression</u> | <u>Functions for Expressions</u> | <u>Operators for Expressions</u> | <u>Predefined Variables for Expressions</u> | <u>Query Designer Considerations for SQL</u> <u>Server Databases</u>

Including User-Defined Functions in a Query

You can include a user-defined function in a query. You can include any of the three types of functions:

- Non-updateable functions returning a table
- Updateable functions returning a table
- Functions returning a scalar

To include an updateable or non-updateable table function in a query or view

- 1. Design the query to which you want to add the function. In the Query Designer, be sure the Diagram pane is visible.
- 2. In the SQL Server Enterprise Manager, expand the Databases node and the particular node for the database you are working on.
- 3. Click the User Defined Functions node.
- 4. From the list of functions, drag the function onto the Diagram pane.

To include a scalar function in a query or view

- 1. Design the query to which you want to add the function. In the Query Designer, be sure the Grid pane is visible.
- 2. In the SQL Server Enterprise Manager, expand the Databases node and the particular node for the database you are working on.
- 3. Click the User Defined Functions node.

4. In the first blank row of the Grid pane, enter the name of the function in the **Column** column.

See Also

User-Defined Functions

Querying on Groups of Rows

You can create a query result in which each result row corresponds to an entire group of rows from the original data. To learn about the logical principles for creating such queries, see <u>Collapsing Groups of Rows</u>.

To learn the details about creating such queries, see the topics listed in the following table:

То	See
Learn about excluding duplicate	Excluding Duplicate Rows
rows	
Create subsets of summary	Grouping Rows in Query Results
information by organizing data into	
groups	
Count the number of rows that meet	Counting Rows in a Table
specific conditions	
Create an average, sum, or other	Summarizing Values for All Rows in
summary from information in all	<u>a Table</u>
rows in a table	
Use calculations to create summary	Summarizing Values Using Custom
information	Expressions
Create search conditions that apply to	Specifying Conditions for Groups
groups of rows	
Use search conditions on both	Using HAVING and WHERE
individual rows and on groups of	<u>Clauses in the Same Query</u>
rows	

Excluding Duplicate Rows

If you want to see only unique values in a result set, you can specify that you want to exclude duplicates from the result set.

To exclude duplicate rows from the result set

- 1. Right-click the background of the Diagram pane, then choose **Property Pages** from the shortcut menu.
- 2. In the <u>Query tab</u> of the **Property pages** window, select **Distinct values**.

The Query Designer inserts the keyword DISTINCT in front of the list of display columns in the SQL statement.

Note If you use the DISTINCT keyword, you cannot modify the result set in the results pane.

See Also

<u>Selecting Rows that Do Not Match a Value | Specifying Search Criteria |</u> <u>Collapsing Groups of Rows</u>

Grouping Rows in Query Results

If you want to create subtotals or show other summary information for subsets of a table, you create groups using an <u>aggregate query</u>. Each group summarizes the data for all the rows in the table that have the same value.

For example, you might want to see the average price of a book in the titles table, but break the results down by publisher. To do so, you group the query by publisher (for example, pub_id). The resulting query output might look like this:

pub_id	Avg Price
0736	9.784
0877	15.41
1389	18.976

When you group data, you can display only summary or grouped data, such as:

- The values of the grouped columns (those that appear in the GROUP BY clause). In the example above, pub_id is the grouped column.
- Values produced by <u>aggregate functions</u> such as SUM() and AVG(). In the example above, the second column is produced by using the AVG() function with the price column.

You cannot display values from individual rows. For example, if you group only by publisher, you cannot also display individual titles in the query. Therefore, if you add columns to the query output, the Query Designer automatically adds them to the GROUP BY clause of the statement in the <u>SQL pane</u>. If you want a column to be aggregated instead, you can specify an aggregate function for that column.

If you group by more than one column, each group in the query shows the aggregate values for all grouping columns.

For example, the following query against the titles table groups by publisher (pub_id) and also by book type (type). The query results are ordered by

publisher and show summary information for each different type of book that the publisher produces:

SELECT pub_id, type, SUM(price) Total_price FROM titles GROUP BY pub_id, type

The resulting output might look like this:

pub_id	type	Total_price
0736	business	2.99
0736	psychology	45.93
0877	mod_cook	22.98
0877	psychology	21.59
0877	trad_cook	47.89
1389	business	51.93
1389	popular_com-	

To group rows

- 1. Start the query by adding the tables you want to summarize to the <u>Diagram pane</u>.
- 2. Right-click the background of the Diagram pane, then choose **Group By** from the shortcut menu. The Query Designer adds a **Group By** column to the grid in the <u>Grid pane</u>.
- 3. Add the column or columns you want to group to the **Grid** pane. If you want the column to appear in the query output, be sure that the **Output** column is selected for output.

The Query Designer adds a GROUP BY clause to the statement in the SQL pane. For example, the SQL statement might look like this:

SELECT pub_id FROM titles GROUP BY pub_id

4. Add the column or columns you want to aggregate to the **Grid** pane. Be sure that the column is marked for output.

5. In the **Group By** grid cell for the column that is going to be aggregated, select the appropriate aggregate function.

The Query Designer automatically assigns a column <u>alias</u> to the column you are summarizing. You can replace this automatically generated alias with a more meaningful one. For more details, see <u>Creating Column Aliases</u>.

	Column	Alias	Table	Output	Group By
	pub_id		titles (dbo)	\checkmark	Group By
9	price	Total price	titles (dbo)	\checkmark	Sum
~					

The corresponding statement in the **SQL** pane might look like this:

SELECT pub_id, SUM(price) AS Totalprice FROM titles GROUP BY pub_id

See Also

Specifying Conditions for Groups | Summarizing and Grouping | Summarizing Values for All Rows in a Table | Summary and Grouping Behavior in the Query Designer | Querying on Groups of Rows.

Counting Rows in a Table

You can count rows in a table to determine:

- The total number of rows in a table, for example, a count of all the books in a titles table.
- The number of rows in a table that meet a specific condition, for example, the number of books by one publisher in a titles table.
- The number of values in a particular column.

When you count values in a column, nulls are not included in the count. For example, you might count the number of books in a titles table that have values in the advance column. By default, the count includes all values, not just unique values.

The procedures for all three types of counts are similar.

To count all the rows in a table

- 1. Be sure the table you want to summarize is already present in the Diagram pane.
- Right-click the background of the Diagram pane, then choose Group By from the shortcut menu. The Query Designer adds a Group By column to the grid in the <u>Grid pane</u>.
- 3. Select ***** (All Columns) in the rectangle representing the table or tablestructured object.

The Query Designer automatically fills the term **Count** into the **Group By** column in the **Grid** pane and assigns a column <u>alias</u> to the column you are summarizing. You can replace this automatically generated

alias with a more meaningful one. For more details, see <u>Creating</u> <u>Column Aliases</u>.

To count all the rows that meet a condition

- 1. Be sure the table you want to summarize is already present in the Diagram pane.
- Right-click the background of the Diagram pane, then choose Group By from the shortcut menu. The Query Designer adds a Group By column to the grid in the Grid pane.
- 3. Select ***** (All Columns) in the rectangle representing the table or tablestructured object.

The Query Designer automatically fills the term **Count** into the **Group By** column in the **Grid** pane and assigns a column alias to the column you are summarizing. To create a more useful column heading in query output, see <u>Creating Column Aliases</u>.

4. Add the data column that you want to search, and then clear the check box in the **Output** column.

The Query Designer automatically fills the term **Group By** into the **Group By** column of the grid.

- 5. Change **Group By** in the **Group By** column to **Where**.
- 6. In the **Criteria** column for the data column to search, enter the search condition.

To count the values in a column

- 1. Be sure the table you want to summarize is already present in the Diagram pane.
- Right-click the background of the Diagram pane, then choose Group By from the shortcut menu. The Query Designer adds a Group By

column to the grid in the Grid pane.

3. Add the column that you want to count to the **Grid** pane.

The Query Designer automatically fills the term **Group By** into the **Group By** column of the grid.

4. Change **Group By** in the **Group By** column to **Count**.

Note To count only unique values, choose **Count Distinct**.

See Also

<u>Summarizing and Grouping | Summarizing Values for All Rows in a Table |</u> <u>Summary and Grouping Behavior in the Query Designer</u>

Summarizing or Aggregating Values for All Rows in a Table

Using an <u>aggregate function</u>, you can create a summary for all the values in a table. For example, you can create a query such as the following to display the total price for all books in the titles table:

```
SELECT SUM(price)
FROM titles
```

You can create multiple aggregations in the same query by using aggregate functions with more than one column. For example, you can create a query that calculates the total of the price column and the average of the discount column.

You can also aggregate the same column in different ways (such as totaling, counting, and averaging) in the same query. For example, the following query averages and summarizes the price column from the titles table:

SELECT AVG(price), SUM(price) FROM titles

If you add a <u>search condition</u>, you can aggregate the subset of rows that meet that condition.

Note You can also count all the rows in the table or the ones that meet a specific condition. For details, see <u>Counting Rows in a Table</u>.

When you create a single aggregation value for all rows in a table, you display only the aggregate values themselves. For example, if you are totaling the value of the price column of the titles table, you would not also display individual titles, publisher names, and so on.

Note If you are subtotaling — that is, creating groups — you can display column values for each group. For details, see <u>Grouping Rows in Query</u> <u>Results</u>.

To aggregate values for all rows

- 1. Be sure the table you want to aggregate is already present in the Diagram pane.
- Right-click the background of the Diagram pane, then choose Group By from the shortcut menu. The Query Designer adds a Group By column to the grid in the <u>Grid pane</u>.
- 3. Add the column you want to aggregate to the **Grid** pane. Be sure that the column is marked for output.

The Query Designer automatically assigns a column <u>alias</u> to the column you are summarizing. You can replace this alias with a more meaningful one. For details, see <u>Creating Column Aliases</u>.

4. In the **Group By** grid column, select the appropriate aggregate function, such as: **Sum**, **Avg**, **Min**, **Max**, **Count**. If you want to aggregate only unique rows in the result set, choose an aggregate function with the DISTINCT options, such as **Min Distinct**. Do not choose **Group By**, **Expression**, or **Where**, because those options do not apply when you are aggregating all rows.

The Query Designer replaces the column name in the statement in the <u>SQL pane</u> with the aggregate function that you specify. For example, the SQL statement might look like this:

SELECT SUM(price) FROM titles

5. If you want to create more than one aggregation in the query, repeat steps 3 and 4.

When you add another column to the query output list or order by list, the Query Designer automatically fills the term **Group By** into the **Group By** column of the grid. Select the appropriate aggregate function.

6. Add search conditions, if any, to specify the subset of rows you want

to summarize.

When you execute the query, the Results pane displays the aggregations that you specified.

Note The Query Designer maintains aggregate functions as part of the SQL statement in the SQL pane until you explicitly turn off Group By mode. Therefore, if you modify your query by changing its type or by changing which tables or table-structured objects are present in the diagram pane, the resulting query might include invalid aggregate functions.

See Also

<u>Counting Rows in a Table | Grouping Rows in Query Results | Summarizing</u> <u>Values for All Rows in a Table | Summarizing Values Using Custom Expressions</u> | <u>Summary and Grouping Behavior in the Query Designer</u>

Summarizing or Aggregating Values Using Custom Expressions

In addition to using <u>aggregate functions</u> to aggregate data, you can create custom expressions to produce aggregate values. You can use custom expressions in place of aggregate functions anywhere in an <u>aggregate query</u>.

For example, in the titles table you might want to create a query that shows not just the average price, but what the average price would be if it were discounted.

You cannot include an <u>expression</u> that is based on calculations involving only individual rows in the table; the expression must be based on an aggregate value, because only the aggregate values are available at the time the expression is calculated.

To specify a custom expression for a summary value

- 1. Specify the groups for your query. For details, see <u>Grouping Rows in</u> <u>Query Results</u>.
- 2. Move to a blank row of the <u>Grid Pane</u>, and then type the expression in the **Columns** column.

The Query Designer automatically assigns a column <u>alias</u> to the expression to create a useful column heading in query output. For more details, see <u>Creating Column Aliases</u>.

3. In the **Group By** column for the expression, select **Expression**.

See Also

<u>Counting Rows in a Table | Summarizing Values for All Rows in a Table |</u> <u>Summary and Grouping Behavior in the Query Designer</u>

Specifying Conditions for Groups

You can limit the groups that appear in a query by specifying a condition that applies to groups as a whole — a HAVING clause. After the data has been grouped and aggregated, the conditions in the HAVING clause are applied. Only the groups that meet the conditions appear in the query.

For example, you might want to see the average price of all books for each publisher in a titles table, but only if the average price exceeds \$10.00. In that case, you could specify a HAVING clause with a condition such as AVG(price) > 10.

Note In some instances, you might want to exclude individual rows from groups before applying a condition to groups as a whole. For details, see <u>Using HAVING and WHERE Clauses in the Same Query</u>.

You can create complex conditions for a HAVING clause by using AND and OR to link conditions. For details about using AND and OR in <u>search conditions</u>, see <u>Specifying Multiple Search Conditions for One Column</u>.

To specify a condition for a group

- 1. Specify the groups for your query. For details, see <u>Grouping Rows in</u> <u>Query Results</u>.
- 2. If it is not already in the <u>Grid pane</u>, add the column on which you want to base the condition. (Most often the condition involves a column that is already a group or summary column.) You cannot use a column that is not part of an aggregate function or of the GROUP BY clause.
- 3. In the **Criteria** column, specify the condition to apply to the group.

The Query Designer automatically creates a HAVING clause in the statement in the <u>SQL pane</u>, such as in the following example:

SELECT pub_id, AVG(price)

FROM titles GROUP BY pub_id HAVING (AVG(price) > 10)

4. Repeat steps 2 and 3 for each additional condition you want to specify.

See Also

<u>Grouping Rows in Query Results</u> | <u>Summarizing and Grouping</u> | <u>Summary and</u> <u>Grouping Behavior in the Query Designer</u> | <u>Using HAVING and WHERE</u> <u>Clauses in the Same Query</u>

Using HAVING and WHERE Clauses in the Same Query

In some instances, you might want to exclude individual rows from groups (using a WHERE clause) before applying a condition to groups as a whole (using a HAVING clause).

A HAVING clause is like a WHERE clause, but applies only to groups as a whole (that is, to the rows in the <u>result set</u> representing groups), whereas the WHERE clause applies to individual rows. A query can contain both a WHERE clause and a HAVING clause. In that case:

- The WHERE clause is applied first to the individual rows in the tables or table-structured objects in the diagram pane.. Only the rows that meet the conditions in the WHERE clause are grouped.
- The HAVING clause is then applied to the rows in the result set that are produced by grouping. Only the groups that meet the HAVING conditions appear in the query output. You can apply a HAVING clause only to columns that also appear in the GROUP BY clause or in an aggregate function.

For example, imagine that you are joining the titles and publishers tables to create a query showing the average book price for a set of publishers. You want to see the average price for only a specific set of publishers — perhaps only the publishers in the state of California. And even then, you want to see the average price only if it is over \$10.00.

You can establish the first condition by including a WHERE clause, which discards any publishers that are not in California, before calculating average prices. The second condition requires a HAVING clause, because the condition is based on the results of grouping and summarizing the data. The resulting <u>SQL</u> statement might look like this:

SELECT titles.pub_id, AVG(titles.price)

```
FROM titles INNER JOIN publishers
   ON titles.pub_id = publishers.pub_id
WHERE publishers.state = 'CA'
GROUP BY titles.pub_id
HAVING AVG(price) > 10
```

You can create both HAVING and WHERE clauses in the <u>Grid pane</u> of the Query Designer. By default, if you specify a <u>search condition</u> for a column, the condition becomes part of the HAVING clause. However, you can change the condition to be a WHERE clause.

You can create a WHERE clause and HAVING clause involving the same column. To do so, you must add the column twice to the **Grid** pane, then specify one instance as part of the HAVING clause and the other instance as part of the WHERE clause.

To specify a WHERE condition in an aggregate query

- 1. Specify the groups for your query. For details, see <u>Grouping Rows in</u> <u>Query Results</u>.
- 2. If it is not already in the **Grid** pane, add the column on which you want to base the WHERE condition.
- 3. Clear the **Output** column unless the data column is part of the GROUP BY clause or included in an aggregate function.
- 4. In the **Criteria** column, specify the WHERE condition. The Query Designer adds the condition to the HAVING clause of the SQL statement.

Note The query shown in the example for this procedure joins two tables, titles and publishers.

At this point in the query, the SQL statement contains a HAVING clause:

SELECT titles.pub_id, AVG(titles.price) FROM titles INNER JOIN publishers ON titles.pub_id = publishers.pub_id GROUP BY titles.pub_id HAVING publishers.state = 'CA'

5. In the **Group By** column, select **Where** from the list of group and summary options. The Query Designer removes the condition from the HAVING clause in the SQL statement and adds it to the WHERE clause.

The SQL statement changes to include a WHERE clause instead:

```
SELECT titles.pub_id, AVG(titles.price)
FROM titles INNER JOIN publishers
ON titles.pub_id = publishers.pub_id
WHERE publishers.state = 'CA'
GROUP BY titles.pub_id
```

See Also

<u>Grouping Rows in Query Results | Specifying Conditions for Groups | Summary</u> and Grouping Behavior in the Query Designer

Querying Using Multiple Tables

A query result can include data from multiple tables or table-structured objects. To combine data from multiple table-structured objects, you use the JOIN operation from SQL. For more information, see <u>Combining Tables</u>.

For information about creating queries using multiple tables, see the following topics:

То	See
Learn the different ways in which	Types of Joins
tables can be joined	
Learn how the Query Designer	How the Query Designer Represents
displays join information in the	Joins
Diagram pane	
Let the Query Designer determine if	Joining Tables Automatically
tables should be joined	
Join tables yourself	Joining Tables Manually
Specify that tables should be joined	Modifying Join Operators
using an operator other than equal	
(=)	
Specify that joined tables should	<u>Creating Outer Joins</u>
include rows even when they do not	
match rows in the corresponding	
table	
Use a join to find subsets of data	Creating Self-Joins
within a single table	
Remove a join between tables	Removing Joins

Joining Tables Automatically

When you add two or more tables to a query, the Query Designer attempts to determine if they are related. If they are, the Query Designer automatically puts join lines between the rectangles representing the tables or table-structured objects.

The Query Designer will recognize tables as joined if:

- The database contains information that specifies that the tables are related.
- If two columns, one in each table, have the same name and data type. The column must be a <u>primary key</u> in at least one of the tables. For example, if you add employee and jobs tables, if the job_id column is the primary key in the jobs table, and if each table has a column called job_id with the same data type, the Query Designer will automatically join the tables.

Note The Query Designer will create only one join based on columns with the same name and data type. If more than one join is possible, the Query Designer stops after creating a join based on the first set of matching columns that it finds.

• The Query Designer detects that a search condition (a WHERE clause) is actually a join condition. For example, you might add the tables employee and jobs, then create a search condition that searches for the same value in the job_id column of both tables. When you do, the Query Designer detects that the search condition results in a join, and then creates a join condition based on the search condition.

If the Query Designer has created a join that is not suitable to your query, you can modify the join or remove it. For details, see <u>Modifying Join Operators</u> and <u>Removing Joins</u>.

If the Query Designer does not automatically join the tables in your query, you

can create a join yourself. For details, see <u>Joining Tables Manually</u>.

See Also

<u>Creating Outer Joins | Creating Self-Joins | How the Query Designer Represents</u> Joins | Joining Tables Automatically | Joining Tables Manually | Modifying Join Operators | Querying Using Multiple Tables | Removing Joins | Types of Joins

Joining Tables Manually

When you add two (or more) tables to a query, the Query Designer attempts to join them based on common data or on information stored in the database about how tables are related. For details, see <u>Joining Tables Automatically</u>. However, if the Query Designer has not joined the tables automatically, or if you want to create additional join conditions between tables, you can join tables manually.

You can create joins based on comparisons between any two columns, not just columns that contain the same information. For example, if your database contains two tables, titles and roysched, you can compare values in the ytd_sales column of the titles table against the lorange and hirange columns in the roysched table. Creating this join would allow you to find titles for which the year-to-date sales falls between the low and high ranges for the royalty payments.

Tip Joins work fastest if the columns in the join condition have been indexed. In some cases, joining on unindexed columns can result in a slow query. For information about creating indexes using the Visual Database Tools, see <u>Indexes</u>.

To manually join tables or table-structured objects

- 1. Add to the <u>Diagram pane</u> the objects you want to join.
- 2. Drag the name of the join column in the first table or table-structured object and drop it onto the related column in the second table or table-structured object. You cannot base a join on text, ntext, or image columns.

Note The join columns must be of the same (or compatible) data types. For example, if the join column in the first table is a date, you must relate it to a date column in the second table. On the other hand, if the first join column is an integer, the related join column must also be of an integer data type, but it can be a different size. The Query Designer will not check the data

types of the columns you use to create a join, but when you execute the query, the database will display an error if the data types are not compatible.

3. If necessary, change the join operator; by default, the operator is an equal sign (=). For background, see Join Comparison Operators. For details, see Modifying Join Operators.

The Query Designer adds an INNER JOIN clause to the SQL statement in the <u>SQL pane</u>. You can change the type to an <u>outer join</u>. For details see <u>Creating</u> <u>Outer Joins</u>.

See Also

How the Query Designer Represents Joins | Querying Using Multiple Tables | Types of Joins

Joining Tables on Multiple Columns

You can join tables with multiple columns. That is, you can create a query that matches rows from the two tables only if they satisfy multiple conditions. For background information, see Join Columns. If the database contains a relationship matching multiple foreign-key columns in one table to a multicolumn primary key in the other table, you can use this relationship to create a multicolumn join. For details, see Joining Tables Automatically.

Even if the database contains no multi-column foreign-key relationship, you can create the join manually.

To manually create a multicolumn join

- 1. Add to the <u>Diagram pane</u> the tables you want to join.
- 2. Drag the name of the first join column in the first table window and drop it onto the related column in the second table window. You cannot base a join on text, ntext, or image columns.

Note In general, the join columns must be of the same (or compatible) data types. For example, if the join column in the first table is a date, you must relate it to a date column in the second table. On the other hand, if the first join column is an integer, the related join column must also be of an integer data type, but it can be a different size. However, SQL Server provides implicit data type conversions so that many joins between seemingly incompatible columns will work. The Query Designer will not check the data types of the columns you use to create a join, but when you execute the query, the database will display an error if the data types are not compatible.

3. Drag the name of the second join column in the first table window and drop it onto the related column in the second table window.

4. Repeat step 3 for each additional pair of join columns in the two tables.

See Also

Joining Tables Automatically | Joining Tables Manually | Querying Using Multiple Tables

Modifying Join Operators

By default, the Query Designer joins tables using an equal sign (an <u>equijoin</u>), which matches values in the two join columns. If you want, you can change the operator used to compare values in the join columns.

To modify join operators

- 1. In the <u>Diagram pane</u>, right-click the join line you want to modify, and then choose **Properties** from the shortcut menu.
- In the Join Line tab of the **Properties** dialog box, select a new operator from the list.
 –or–

Change the operator in the SQL statement in the <u>SQL pane</u>.

See Also

Joining Tables Automatically | Joining Tables Manually | Querying Using Multiple Tables

Creating Outer Joins

By default, the Query Designer creates an <u>inner join</u> between tables. If you want to include data rows in the result set that do not have a match in the joined table, you can create an <u>outer join</u>.

When you create an outer join, the order in which tables appear in the SQL statement (as reflected in the SQL pane) is significant. The first table you add becomes the "left" table and the second table becomes the "right" table. (The actual order in which the tables appear in the <u>Diagram pane</u> is not significant.) When you specify a left or right outer join, you are referring to the order in which the tables were added to the query and to the order in which they appear in the SQL statement in the <u>SQL pane</u>.

To create an outer join

- 1. Create the join, either automatically or manually. For details, see <u>Joining Tables Automatically</u> or <u>Joining Tables Manually</u>.
- 2. Select the join line in the **Diagram** pane, and then choose **Select All Rows from** *table* from the shortcut menu, selecting the command that includes the table whose extra rows you want to include.

-0r-

In the **Diagram** pane, right-click the join line you want to change to an outer join, and then choose **Properties** from the shortcut menu. Under **Include rows** in the Join Line tab of the **Properties** dialog box, choose the option specifying the table from which you want to include all rows.

- Choose the first table to create a left outer join.
- Choose the second table to create a right outer join.
- Choose both tables to create a full outer join.

When you specify an outer join, the Query Designer modifies the join line to indicate an outer join.

In addition, the Query Designer modifies the SQL statement in the SQL pane to reflect the change in join type, as shown in the following statement:

```
SELECT employee.job_id, employee.emp_id,
employee.fname, employee.minit, jobs.job_desc
FROM employee LEFT OUTER JOIN jobs ON
employee.job_id = jobs.job_id
```

Because an outer join includes unmatched rows, you can use it to find rows that violate <u>foreign key</u> constraints. To do so, you create an outer join and then add a <u>search condition</u> to find rows in which the <u>primary key</u> column of the rightmost table is null. For example, the following outer join finds rows in the <u>employee</u> table that do not have corresponding rows in the jobs table:

SELECT employee.emp_id, employee.job_id FROM employee LEFT OUTER JOIN jobs ON employee.job_id = jobs.job_id WHERE (jobs.job_id IS NULL)

See Also

<u>How the Query Designer Represents Joins | Querying Using Multiple Tables |</u> <u>Types of Joins</u>

Creating Self-Joins Automatically

If a table has a reflexive relationship in the database, you can join it to itself automatically.

To create a self-join automatically

- 1. Add to the <u>Diagram pane</u> the table you want to work with.
- Add the same table again, so that the **Diagram** pane shows the same table twice within the Diagram pane.
 The Query Designer assigns an alias to the second instance by adding a sequential number to the table name. In addition, the Query Designer creates a join line between the two rectangles representing the two different ways the table participates in the query.

See Also

Drawing a Reflexive Relationship

Creating Self-Joins Manually

You can join a table to itself even if the table does not have a reflexive relationship in the database. For example, you can use a self-join to find pairs of authors living in the same city.

As with any join, a self-join requires at least two tables. The difference is that, instead of adding a second table to the query, you add a second instance of the same table. That way, you can compare a column in the first instance of the table to the same column in the second instance, which allows you to compare the values in a column to each other. The Query Designer assigns an <u>alias</u> to the second instance of the table.

For example, if you are creating a self-join to find all pairs of authors within Berkeley, you compare the City column in the first instance of the table against the City column in the second instance. The resulting query might look like the following:

```
SELECT
authors.au_fname,
authors.au_lname,
authors1.au_fname AS Expr2,
authors1.au_lname AS Expr3
FROM
authors
INNER JOIN
authors authors1
ON authors.city
= authors1.city
WHERE
authors.city = 'Berkeley'
```

Creating a self-join often requires multiple join conditions. To understand why, consider the result of the preceding query:

Cheryl CarsonCheryl CarsonAbraham BennetAbraham BennetCheryl CarsonAbraham BennetAbraham BennetCheryl Carson

The first row is useless; it indicates that Cheryl Carson lives in the same city as Cheryl Carson. The second row is equally useless. To eliminate this useless data, you add another condition retaining only those result rows in which the two author names describe different authors. The resulting query might look like this:

```
SELECT
```

```
authors.au_fname,
authors.au_lname,
authors1.au_fname AS Expr2,
authors1.au_lname AS Expr3
FROM
authors
INNER JOIN
authors
authors authors1
ON authors.city
= authors1.city
AND authors.au_id
<> authors1.au_id
WHERE
authors.city = 'Berkeley'
```

The result set is improved:

Cheryl CarsonAbraham BennetAbraham BennetCheryl Carson

But the two result rows are redundant. The first says Carson lives in the same city as Bennet, and the second says the Bennet lives in the same city as Carson. To eliminate this redundancy, you can alter the second join condition from "not

equals" to "less than". The resulting query might look like this:

```
SELECT
authors.au_fname,
authors.au_lname,
authors1.au_fname AS Expr2,
authors1.au_lname AS Expr3
FROM
authors INNER JOIN
authors authors1
ON authors.city
= authors1.city
AND authors.au_id
< authors1.au_id
WHERE
authors.city = 'Berkeley'
```

And the result set looks like this:

Cheryl Carson Abraham Bennet

To create a self-join manually

- 1. Add to the <u>Diagram pane</u> the table or table-structured object you want to work with.
- 2. Add the same table again, so that the **Diagram** pane shows the same table or table-structured object twice within the Diagram pane..

The Query Designer assigns an alias to the second instance by adding a sequential number to the table name. In addition, the Query Designer creates a join line between the two occurrences of the table or tablestructured object within the Diagram pane.

3. Right-click the join line, choose **Properties** from the shortcut menu, and then change the comparison operator between the primary keys as

required. For example, you might change the operator to less than (<).

- 4. Create the additional join condition (for example, authors.zip = authors1.zip) by dragging the name of the primary join column in the first occurrence of the table or table-structured object and dropping it on the corresponding column in the second occurrence.
- 5. Specify other options for the query such as output columns, search conditions, and sort order.

See Also

Querying Using Multiple Tables

Removing Joins

If you do not want tables to be joined via an inner join or an outer join, you can remove the join between them. For example, you might remove a join that the Query Designer has created automatically between two tables.

Note Removing a join from a query does alter the underlying relationship in the database.

If two joined tables are part of your query and you remove all join conditions between them, the resulting query becomes the product of both tables — that is, it becomes a CROSS JOIN. For more information, see <u>Types of Joins</u>.

To remove a join

• In the <u>Diagram pane</u>, select the join line for the join to remove, and then press the DELETE key. You can select and delete multiple join lines at one time.

The Query Designer removes the join line and alters the statement in the <u>SQL</u> <u>pane</u>.

See Also

Joining Tables Automatically | Joining Tables Manually | Querying Using Multiple Tables

Manipulating Data

In addition to viewing the contents of tables, you can use the Query Designer or View Designer to modify data, such as editing the contents of columns, adding new rows, and deleting existing rows. Using the graphical features of the Query Designer and View Designer, you can modify data in tables two ways:

- Execute a <u>Select query</u>, then edit the data directly in the <u>Results pane</u>.
- Create a query to update, delete, or copy data in tables or updateable views.

Note If you are familiar with <u>SQL</u>, you can also execute any statement in the <u>SQL pane</u>, including those that update or delete data in tables or updateable views. The features discussed in this section relate primarily to capabilities that are available by using the <u>Diagram</u>, <u>Grid</u>, and <u>Results</u> panes.

Editing directly in the Results pane is similar to editing in a form or spreadsheet. You can see the changes immediately, and if there is a problem (for example, you attempt to enter data that does not match the data type of a column), you are notified as soon as you try to move to another row. For details, see <u>Editing Rows</u> in the Results Pane.

Alternatively, you can create a query to update, delete, or copy the data of many rows in one operation. For example, you can create one query that changes the status of all the employees who have worked at the company more than five years, or a set of two queries that first copy and then delete all orders more than two years old.

See Also

Adding New Rows in the Results Pane | Creating Delete Queries | Deleting Rows in the Results Pane | Editing Rows in the Results Pane

Editing Rows in the Results Pane

In many cases, you can edit the data in the Results pane grid and save changes to the table or tables represented by the query. However, whether you can actually change, add, or delete rows depends on the type of query you have defined and on <u>constraints</u>, <u>triggers</u>, and permissions maintained by the database.

For details about the types of queries you can update, see <u>Rules for Updating</u> <u>Results</u>.

Note You cannot update timestamp or binary columns using the Query Designer.

The Query Designer or View Designer can perform only minor validation of the data you enter in the Results pane. For example, if the price column of the titles table accepts only positive values, the Query Designer or View Designer might not enforce this restriction, and it would be possible to enter a negative value. However, when you save a row, the database will report an error if any column in the row contains invalid data.

To edit data in the Results pane

1. Navigate to the cells containing the data you want to change.

Type in the new data. While you are editing, the leftmost column displays *s* to indicate that the row is being edited.

While you are editing, the following rules apply:

- When entering number, currency, time, or date information, use a format that will be recognized according to the specification in the **Regional Settings** dialog box in the Windows Control Panel.
- To enter a NULL value into a cell, press CTRL+0.
- You can edit a memo-type column (such as a text, memo, or long character column) if the column does not display <Long Text>. The Results pane can accept up to 900 characters of text in memo-type

columns. If you are typing in a cell and exceed the amount of text that the Results pane can accept, the Query Designer will beep to indicate that you have exceeded this limit.

- You cannot edit <u>BLOB</u> data.
- 2. Save your changes by moving to another row in the grid.

Note The Query Designer or View Designer does not automatically save your changes if you switch to another pane. If you edit the current query (for example, by making a change in the <u>Grid Pane</u>), the **Results** pane is dimmed. However, it is still active, and you can still edit and save the row on which you had been working.

To cancel your changes for the current row, press ESC. If you press ESC while in a cell that you have changed, the changes for only that cell are canceled. If you press ESC while in a cell that you have not changed, the changes for the entire row are canceled and all the cells in that row revert to their old values.

It is possible that a result you are using conflicts with changes being made by other users. For example, you might be using the Results pane to edit a row while another user executes an <u>Update query</u> that modifies the row you are editing.

When you save the row you have edited, the Query Designer or View Designer compares the row against the version currently in the database. If there is a difference, a message appears indicating that the target row could not be located in the database. You can run the query or view a second time to refresh the Results pane and see the other user's modifications.

See Also

<u>Adding New Rows in the Results Pane</u> | <u>Deleting Rows in the Results Pane</u> | <u>Navigating in the Query Designer</u> | <u>Rules for Updating Results</u>

Adding New Rows in the Results Pane

If the data in the Results pane can be modified, you can also add new rows.

Note You cannot add <u>BLOB</u> or binary columns using the Results pane.

The Query and View Designers can perform only simple data validation on the data you enter in the Results pane. For example, if the price column of the titles table accepts only positive values, the Query or View Designer might be able to enforce this restriction and might display an error message if you enter a negative value. But if the column is defined in the database to accept prices in only a certain range (such as between 100 and 1000), the Query or View Designer cannot enforce this restriction. When you save a row, the database will display an error message if any column in the row contains invalid data.

When you add a row, some columns might be filled in by <u>default</u> values or <u>triggers</u>. In many cases the Query Designer or View Designer can refresh the display of the new row to show you the automatically generated values. However, rows cannot be refreshed if the row's primary key:

- Is not part of the result set.
- Is generated automatically.
- Is modified by a trigger.

In these cases, if you want to see the row as it is stored in the database, you must re-execute the query. Even then, however, the new row might not appear if it does not match the <u>search conditions</u> in the query you execute.

You can add new data either by typing it in or by pasting it from the Clipboard. A row to be pasted must have exactly the same number and types of columns as the table into which you are pasting. You can paste multiple rows into the Results pane at once.

To add a new data row

1. Navigate to the bottom of the **Results** pane, where a blank row is

available for adding a new data row.

Tip You can jump from anywhere in the **Results** pane to the bottom by pressing INS or right-clicking the **Results** pane, and then choosing **New**.

- 2. If you are pasting rows from the Clipboard, select the new row by clicking the button to its left.
- 3. Enter the data for the new row. If you are pasting, choose **Paste** from the shortcut menu.

Tip To enter null into a cell, press CTRL+0.

4. Save the new row by moving to another row in the grid.

Note The Query Designer or View Designer does not automatically save your changes if you switch to another pane. If you edit the current query or view (for example, by making a change in the <u>Grid pane</u>), the **Results** pane is dimmed. However, it is still active, and you can still edit and save the row on which you had been working.

If an error occurs when you save the row (for example, you left a column blank that does not accept null values), the Query Designer displays the error message provided by the database, and then returns you to the row you were editing. You can then:

- Resolve the error by making further edits in the row.
- Cancel the edit by pressing ESC. If you press ESC while in a cell that you changed, the changes for that cell are canceled. If you press ESC while in a cell you have not changed, the changes for the entire row are canceled.

See Also

Deleting Rows in the Results Pane | Editing Rows in the Results Pane |

Navigating in the Query Designer | Rules for Updating Results

Deleting Rows in the Results Pane

If the data in the Results pane can be modified, you can also delete existing rows.

To delete a row

- 1. Select the box to the left of the row or rows you want to delete in the **Results** pane grid.
- 2. Press DELETE.

Caution Rows you delete in this way are permanently removed from the database and cannot be recalled.

See Also

Adding New Rows in the Results Pane | Editing Rows in the Results Pane | Navigating in the Query Designer | Rules for Updating Results

Creating Update Queries

You can change the contents of multiple rows in one operation by using an <u>Update query</u>. For example, in a titles table you can use an Update query to add 10% to the price of all books for a particular publisher.

When you create an Update query, you specify:

- The table to update
- The columns whose contents you want to update
- The value or <u>expression</u> to use to update the individual columns
- <u>Search conditions</u> to define the rows you want to update

For example, the following query updates the titles table by adding 10% to the price of all titles for one publisher:

UPDATE titles SET price = price * 1.1 WHERE (pub_id = '0766')

Caution You cannot undo the action of executing an Update query. As a precaution, back up your data before executing the query.

To create an Update query

- 1. Add the table you want to update to the <u>Diagram pane</u>.
- 2. Right-click in the Query Designer window, point to **Change Type**, and then choose **Update**.

Note If more than one table is displayed in the **Diagram** pane when you start the Update query, the Query Designer displays the <u>Update Table</u> dialog box to prompt you for the name of the

table to update.

- 3. Define the data columns to update by adding them to the query. For details, see <u>Adding Columns</u>. Columns will be updated only if you add them to the query.
- 4. In the **New Value** column of the <u>Grid pane</u>, enter the update value for the column. You can enter literal values, column names, or expressions. The value must match (or be compatible with) the data type of the column you are updating.

Caution The Query Designer cannot check that a value fits within the length of the column you are updating. If you provide a value that is too long, it might be truncated without warning. For example, if a name column is 20 characters long but you specify an update value of 25 characters, the last 5 characters might be truncated.

 Define the rows to update by entering search conditions in the Criteria column. For details, see <u>Specifying Search Conditions</u>. If you do not specify a search condition, all rows in the specified table will be updated.

Note When you add a column to the Grid pane for use in a search condition, the Query Designer also adds it to the list of columns to be updated. If you want to use a column for a search condition but not update it, clear the check box next to the column name in the rectangle representing the table or table-structured object.

When you execute an Update query, no results are reported in the <u>Results pane</u>. Instead, a message appears indicating how many rows were changed.

See Also

Creating Delete Queries | Supported Query Types

Creating Insert From Queries

You can copy rows from one table to another or within a table using an Insert From query. For example, in a titles table, you can use an Insert From query to copy information about all the titles for one publisher to a second table that you can make available to that publisher. An Insert From query is similar to a <u>Make</u> <u>Table</u> query, but copies rows into an existing table.

Tip You can also copy rows from one table to another using cut and paste. For details, see <u>Adding New Rows in the Results Pane</u>.

When you create an Insert From query, you specify:

- The database table to copy rows to (the destination table).
- The table or tables to copy rows from (the source table). The source table or tables become part of a <u>subquery</u>. If you are copying within a table, the source table is the same as the destination table.
- The columns in the source table whose contents you want to copy.
- The target columns in the destination table to copy the data to.
- <u>Search conditions</u> to define the rows you want to copy.
- Sort order, if you want to copy the rows in a particular order.
- Group By options, if you want to copy only summary information.

For example, the following query copies title information from the titles table to an archive table called archivetitles. The query copies the contents of four columns for all titles belonging to a particular publisher:

INSERT INTO archivetitles

(title_id, title, type, pub_id) SELECT title_id, title, type, pub_id FROM titles WHERE (pub_id = '0766')

Note To insert values into a new row, use an Insert Into query.

You can copy the contents of selected columns or of all columns in a row. In either case, the data you are copying must be compatible with the columns in the rows you are copying to. For example, if you copy the contents of a column such as price, the column in the row you are copying to must accept numeric data with decimal places. If you are copying an entire row, the destination table must have compatible columns in the same physical position as the source table.

When you create an Insert From query, the <u>Grid pane</u> changes to reflect options available for copying data. An Append column is added to allow you to specify the columns into which data should be copied.

Caution You cannot undo the action of executing an Insert From query. As a precaution, back up your data before executing the query.

To create an Insert From query

- 1. Right-click in the Query Designer window, point to **Change Type**, and then choose **Insert From**.
- 2. In the <u>Choose Table for INSERT FROM Query</u> dialog box, select the table to copy rows to (the destination table).

Note The Query Designer cannot determine in advance which tables and <u>views</u> you can update. Therefore, the **Table Name** list in the **Choose Table for Insert From Query** dialog box shows all available tables and views in the data connection you are querying, even those that you might not be able to copy rows to.

3. Add to the query the table to copy rows from (the source table). For details, see <u>Adding Tables</u>. If you are copying rows within a table, you can add the source table as a destination table.

The data columns from the source table appear in an input window in the <u>Diagram pane</u>.

4. In the rectangle representing the table or table-structured object, choose the names of the columns whose contents you want to copy. To copy entire rows, choose ***** (All Columns).

The Query Designer adds the columns you choose to the **Column** column of the **Grid** pane.

- 5. In the **Append** column of the **Grid** pane, select a target column in the destination table for each column you are copying. Choose *tablename*.* if you are copying entire rows. The columns in the destination table must have the same (or compatible) data types as the columns in the source table.
- 6. If you want to copy rows in a particular order, specify a sort order. For details, see <u>Ordering Query Results</u>.
- Specify the rows to copy by entering search conditions in the Criteria column. For details, see <u>Specifying Search Conditions</u>. If you do not specify a search condition, all rows from the source table will be copied to the destination table.

Note When you add a column to search to the **Grid** pane, the Query Designer also adds it to the list of columns to copy. If you want to use a column for searching but not copy it, clear the check box next to the column name in the rectangle representing the table or table-structured object **+**

8. If you want to copy summary information, specify Group By options. For details, see <u>Summarizing Values for All Rows in a Table</u>.

When you execute an Insert From query, no results are reported in the <u>Results</u> <u>pane</u>. Instead, a message appears indicating how many rows were copied.

See Also

Creating Delete Queries | Creating Update Queries | Supported Query Types

Creating Delete Queries

You can delete multiple rows in one operation by using a <u>Delete query</u>. When you create a Delete query, you specify the database table to delete rows from and the <u>search condition</u> to define the rows you want to delete.

Note Deleting all rows from a table clears the data in the table but does not delete the table itself. You can delete tables using the Database Designer. For information about using the Database Designer, see <u>Database Designer</u>.

When you create a Delete query, the <u>Grid pane</u> changes to reflect the options available for deleting rows. Because you do not display data in a Delete query, the Output, Sort By, and Sort Order columns are removed. In addition, the check boxes next to the column names in the rectangle representing the table or tablestructured object are removed because you cannot specify individual columns to delete.

Caution You cannot undo the action of executing a Delete query. As a precaution, back up your data before executing a Delete query.

To create a Delete query

- 1. Add the table to delete rows from to the <u>Diagram pane</u>.
- 2. Right-click in the Query Designer window, point to **Change Type**, and then choose **Delete**.

Note If more than one table is displayed in the **Diagram** pane when you start the Delete query, the Query Designer displays the <u>Delete Table</u> dialog box to prompt you for the name of the table to delete rows from.

3. Enter search conditions to define the rows to be deleted. For details, see <u>Specifying Search Conditions</u>. If you do not specify a search condition, all rows in the specified table will be deleted.

Note Because you cannot delete the contents of individual columns, the check boxes in the rectangle representing the table or table-structured object are removed when you start a

Delete query. To add columns to the <u>Grid pane</u> to use in search conditions, drag them from the rectangle representing the table or table-structured object to the **Columns** column, or select a data column in the **Columns** column.

When you execute the Delete query, no results are reported in the <u>Results pane</u>. Instead, a message appears indicating how many rows were deleted.

See Also

Supported Query Types

Creating Insert Into Queries

You can create a new row in the current table using an Insert Into query. When you create an Insert Into query, you specify:

- The database table to add the row to.
- The columns whose contents you want to add.
- The value or <u>expression</u> to insert into the individual columns.

For example, the following query adds a row to the titles table, specifying values for the title, type, publisher, and price:

INSERT INTO titles

(title_id, title, type, pub_id, price) VALUES ('BU9876', 'Creating Web Pages', 'business', '1389', '29.99')

When you create an Insert Into query, the <u>Grid pane</u> changes to reflect the only options available for inserting a new row: the column name and the value to insert.

Caution You cannot undo the action of executing an Insert Into query. As a precaution, back up your data before executing the query.

To create an Insert Into query

- 1. Add the table you want to update to the <u>Diagram pane</u>.
- 2. Right-click in the Query Designer window, point to **Change Type**, and then choose **Insert Into**.

Note If more than one table is displayed in the **Diagram** pane when you start the Insert Into query, the Query Designer displays the <u>Insert Into</u> dialog box to prompt you for the name of the table to update.

- 3. Define the data columns to create by adding them to the query. For details, see <u>Adding Columns</u>. Columns will be updated only if you add them to the query.
- 4. In the **New Value** column of the <u>Grid pane</u>, enter the new value for the column. You can enter literal values, column names, or expressions. The value must match (or be compatible with) the data type of the column you are updating.

Caution The Query Designer cannot check that a value fits within the length of the column you are inserting. If you provide a value that is too long, it might be truncated without warning. For example, if a name column is 20 characters long but you specify an insert value of 25 characters, the last 5 characters might be truncated.

When you execute an Insert Into query, no results are reported in the <u>Results</u> <u>pane</u>. Instead, a message appears indicating how many rows were changed.

See Also

Manipulating Data

Creating Make Table Queries

You can copy rows into a new table using a Make Table query, which is useful for creating subsets of data to work with or copying the contents of a table from one database to another. A Make Table query is similar to an <u>Insert From</u> query, but creates a new table to copy rows into.

When you create a Make Table query, you specify:

- The name of the new database table (the destination table).
- The table or tables to copy rows from (the source table). You can copy from a single table or from joined tables.
- The columns in the source table whose contents you want to copy.
- Sort order, if you want to copy the rows in a particular order.
- <u>Search conditions</u> to define the rows you want to copy.
- Group By options, if you want to copy only summary information.

For example, the following query creates a new table called uk_customers and copies information from the customers table to it:

SELECT * INTO uk_customers FROM customers WHERE country = 'UK'

In order to use a Make Table query successfully:

• You must have permission to create a table in the target database.

To create a Make Table query

- 1. Add the source table or tables to the <u>Diagram pane</u>.
- 2. Right-click in the Query Designer window, point to **Change Type**, and then choose **Create Table**.
- In the Make Table dialog box, type the name of the destination table. The Query Designer does not check whether the name is already in use or whether you have permission to create the table. To create a destination table in another database, specify a fully qualified table name including the name of the target database, the owner (if required), and the name of the table.
- 4. Specify the columns to copy by adding them to the query. For details, see <u>Adding Columns</u>. Columns will be copied only if you add them to the query. To copy entire rows, choose * (All Columns). The Query Designer adds the columns you choose to the Column column of the Grid pane.
- 5. If you want to copy rows in a particular order, specify a sort order. For details, see <u>Ordering Query Results</u>.
- Specify the rows to copy by entering search conditions. For details, see <u>Specifying Search Conditions</u>. If you do not specify a search condition, all rows from the source table will be copied to the destination table.

Note When you add a column to search to the **Grid** pane, the Query Designer also adds it to the list of columns to copy. If you want to use a column for searching but not copy it, clear the check box next to the column name in the rectangle representing the table or table-structured object ✓

7. If you want to copy summary information, specify Group By options. For details, see <u>Summarizing Values for All Rows in a Table</u>. When you execute a Make Table query, no results are reported in the <u>Results</u> <u>pane</u>. Instead, a message appears indicating how many rows were copied.

See Also

Manipulating Data

Creating General Purpose Queries

In many instances, it is useful to create a query that you can reuse many times in your applications with different input each time. In this section you will find information about how to create parameter queries for these general purposes.

То	See
Learn about general-purpose queries	Parameter Queries
Create a query using unnamed	Creating a Query with Unnamed
parameters	Parameters
Specify names for the parameters in	Creating a Query with Named
your query to help distinguish	Parameters
multiple parameters	
Mark a name as a parameter instead	Specifying Parameter Marker
of a string of literal characters	<u>Characters</u>

Creating a Query with Unnamed Parameters

You can create a query with an unnamed parameter by specifying a question mark (?) as a placeholder for a literal value. You can specify as many unnamed parameters in the query as you need.

Note If you are using multiple parameters, you can use named parameters to help you distinguish them. For details, see <u>Creating a Query with Named Parameters</u>.

When you run the query in the Query Designer, the <u>Define Query Parameters</u> dialog box is displayed with "?" as the name of the parameter.

?	

To specify an unnamed parameter

1. Add the columns or <u>expressions</u> that you want to search to the <u>Grid</u> <u>pane</u>. If you do not want the search columns or expressions to appear in the query output, remove them as output columns.

Locate the row containing the data column or expression to search, and then in the **Criteria** grid column, enter a question mark (?).

	Column	Criteria
	fname	=?
ļ	<u> </u>	

By default, the Query Designer adds the "=" operator. However, you can edit the cell to substitute ">", "<", or any other SQL comparison operator.

See Also

Creating a Query with Named Parameters | Parameter Queries

Creating a Query with Named Parameters

You can specify names for the parameters in your query to help you distinguish multiple parameters. For example, if you create a parameter query that prompts for an author's name, you can use the parameter names "first name" and "last name." When you run the query in the Query Designer, you are prompted for the parameter by name:

Parameter Name	Parameter Value
first name	
last name	
	OK Cancel

To mark the parameter name as a name and not a string of literal characters, you specify parameter marker characters and then place those characters around the parameter name. For details, see <u>Specifying Parameter Marker Characters</u>.

If you use both prefix-marker and suffix-marker characters, you can name your parameters using any combination of characters, including text, numbers, punctuation, and spaces. (However, do not use null or a question mark.) If you specify only a prefix-marker character, the parameter name cannot contain spaces.

Note The parameter name you specify is used as a variable in the environment from which you run the query. Be sure that the parameter names you use are acceptable in all environments in which you intend to use the query.

To specify a named parameter

1. If you have not done so already, specify the parameter marker characters. For details, see <u>Specifying Parameter Marker Characters</u>.

- 2. Add the columns or <u>expressions</u> that you want to search to the **Grid** pane. If you are creating a <u>Select query</u> and do not want the search columns or expressions to appear in the query output, remove them as output columns.
- 3. Locate the row containing the data column or expression to search, and then in the **Criteria** grid column, enter the name of the parameter. Be sure to mark the parameter using the appropriate characters:

Column	Criteria
Iname	= [last name]

By default, the Query Designer adds the "=" operator. However, you can edit the cell to substitute ">", "<", or any other SQL comparison operator.

See Also

<u>Creating a Query with Unnamed Parameters</u> | <u>Parameter Queries</u> | <u>Specifying</u> <u>Parameter Marker Characters</u>

Specifying Parameter Marker Characters

To mark a name as a parameter and not as a string of literal characters, you place a prefix in front of the parameter name (and, as an option, a suffix after it). For example, parameter marker characters might be "@", ":", or "%".

For example, in your query you might define a parameter called "last name." To indicate that "last name" is a parameter and not a string to search for in the lname column, you might specify that the parameter marker characters are "[" and "]". You can then enter "[last name]" as a <u>search condition</u> value, and the Query Designer will correctly interpret "last name" as a parameter.

In rare instances, the parameter marker character you define might also appear in the name of the parameter. If so, you can specify an <u>escape character</u>. For details, see <u>Parameters Tab</u>, <u>Properties Window</u>.

To specify parameter marker characters

- 1. Right-click in any pane of the Query Designer, and then from the shortcut menu choose **Properties**.
- 2. In the **Properties** dialog box choose the **Parameters** tab.
- 3. Enter the parameter prefix character or characters and, if you intend to use one, a suffix character. (You can specify a suffix character only if you have specified a prefix character.) If you need to specify one, enter an escape character.

See Also

Creating a Query with Named Parameters | Parameter Queries

Working With Views

A view is simply a SELECT query saved in the database. Thus, most operations you can perform on queries you can also perform on views. However, there are some operations that apply only to one or the other. For background information, see <u>Comparison of Queries and Views</u>.

For details about most of the operations you can perform on views, see <u>Designing Queries</u>. For details about View-specific operations, see the topics listed in the following table.

То	See
Create a view	Creating Views
Create an indexed view	Creating Indexed Views
Modify an indexed view	Modifying Indexed Views

Creating Views

Because a view is saved in the database and a query is not, the process of creating a new view is different from the process of creating a query.

To create a view

- 1. In the SQL Server Enterprise Manager, right-click the **Views** node, then choose **New View** from the shortcut menu.
- 2. Proceed to design the view as you would design a SELECT query.

Note Although you design the view as you would design a SELECT query, there are some restrictions. For more information, see <u>Comparison of Queries and Views</u>.

See Also

Creating Indexed Views

Creating Indexed Views

An indexed view is a view whose result set is stored in the database for fast access. For more background on indexed views, see <u>Indexed Views</u>.

To create an indexed view

- 1. In the SQL Server Enterprise Manager, make sure the Databases node is expanded, and that the node for the target database is expanded, too.
- 2. Right-click on the **Views** node, then choose **New View** from the shortcut menu.
- 3. Add tables to the Diagram pane. Because you intend this to be an indexed view, be sure to add only tables that you own.
- 4. Select which columns you want included in the view. Do not use the asterisk (*); you must explicitly select each column that you want to appear in the indexed view.
- 5. Right-click the background of the Diagram pane, then choose **Manage Indexes** from the shortcut menu. The **Indexes** dialog box appears.

Note There are many situations in which the Manage Indexes command is unavailable, because there are many restrictions on which views can be indexed. For example, you cannot index a view unless you are the owner of each table contributing to it. For more information about indexed views, see <u>View Indexes</u>.

- 6. Within the dialog box, click **New**.
- 7. Supply the information for the index definition index name, index columns and their order, index file group, and the other index settings.

For a complete description of these settings, see <u>Indexes Dialog Box</u>.

8. Click **OK**.

See Also

Indexed Views | Modifying Indexed Views | Creating Views

Modifying Indexed Views

There are two ways you modify an indexed view. You can modify the view, or you can modify the indexes on the view.

To modify an indexed view

- 1. In the SQL Server Enterprise Manager, make sure the Databases node is expanded, and that the node for the target database is expanded, too.
- 2. Click the **Views** node, right-click the target view, then choose **Design View** from the shortcut menu.
- 3. Proceed to modify the view by manipulating its definition on the Diagram pane, Grid pane, and SQL pane.

Note When you save the modified view, its indexes will be deleted and recreated.

To modify the index of an indexed view

- 1. In the SQL Server Enterprise Manager, make sure the Databases node is expanded, and that the node for the target database is expanded, too.
- 2. Click the **Views** node, right-click the target view, then choose **Design View** from the shortcut menu.
- 3. Right-click the background of the Diagram pane and choose **Manage Indexes** from the shortcut menu.
- 4. Revise the information for the index definition index name, index columns and their order, index file group, and the other index settings. For a complete description of these settings, see <u>Indexes Dialog Box</u>.

5. Click **OK**.

See Also

Indexed Views | Creating Indexed Views

Encrypting Views

You can encrypt a view. That is, you can permanently conceal the text of the view definition.

Note This operation is irreversible. After you encrypt a view, you can never again modify it, because you can never again see the view definition. If you need to modify an encrypted view, you must delete it and recreate another one.

To encrypt a view

- 1. In the SQL Server Enterprise Manager, make sure the Databases node is expanded, and that the node for the target database is expanded, too.
- 2. Click the **Views** node, right-click the target view, then choose **Design View** from the shortcut menu.
- 3. Right-click the background of the Diagram pane and choose **Properties** from the shortcut menu.
- 4. Select **Encrypt View** and click **Close**.

See Also

Creating Views

Reference

The following reference topics are available for the Visual Database Tools:

- Rules for Updating Query Results
- Navigating in the Query Designer
- <u>Comparison Operators</u>
- Logical Operators
- <u>Wildcard Characters</u>
- <u>Rules and Grid-Pane Conventions for Combining Search Conditions</u>
- <u>Rules for Entering Search Values</u>
- <u>Aggregate Functions</u>
- <u>Structure of Expressions</u>
- <u>Properties Pages</u>
- <u>Dialog Boxes</u>
- Visual Database Tools and SQL Server Databases
- Error Messages

Rules for Updating Results

In many cases, you can update the <u>result set</u> displayed in the <u>Results pane</u>. However, in some cases you cannot.

In general, in order to update results, the Query Designer must have sufficient information to uniquely identify the row in the table. An example is if the query includes a <u>primary key</u> in the output list. In addition, you must have sufficient permission to update the database.

If your query is based on a <u>view</u>, you might be able to update it. The same guidelines apply, except that they apply to the underlying tables in the view, not just to the view itself.

Note The Query Designer cannot determine in advance whether you can update a result set based on a view. Therefore, it displays all views, even though you might not be able to update them.

The following table summarizes specific instances in which you might and might not be able to update query results in the Results pane.

	Can results be
Query	updated?
Query based on one table with primary key in the	Yes (except as listed
output list	below).
Query based on a table with no unique <u>index</u> and	Query must contain
without a primary key	sufficient information
	to uniquely identify
	records.
Query based on multiple tables which are not joined	No.
Query based on data marked as read-only in the	No.
database	
Query based on a view that involves one table with	Yes (except as listed
no <u>constraints</u>	below).
Query based on tables joined with a	Yes (except as listed
one-to-one relationship	below).
Query based on tables joined with a	Usually.

one-to-many relationship	
Query based on three or more tables in which there	No.
is a <u>many-to-many relationship</u>	
Query based on a table for which update permission	Can delete but not
is not granted	update.
Query based on a table for which delete permission	Can update but not
is not granted	delete.
Aggregate query	No.
Query based on a <u>subquery</u> that contains totals or	No.
aggregate functions	
Query that includes the DISTINCT keyword to	No.
exclude duplicate rows	
Query whose FROM clause includes a user-defined	No.
function that returns a table and the user-defined	
function contains multiple select statements	
Query whose FROM clause includes an inline use-	Yes.
defined function	

In addition, you might not be able to update specific columns in the query results. The following list summarizes specific types of columns that you cannot update in the Results pane.

- Columns based on <u>expressions</u>
- Columns based on scalar user-defined functions
- Rows or columns deleted by another user
- Rows or columns locked by another user (locked rows can usually be updated as soon as they are unlocked)
- Timestamp or <u>BLOB</u> columns

See Also

Reference

Navigating in the Query Designer

You can work in the Query Designer using the keyboard or the mouse. Refer to the following tables for specific methods.

Any Pane

То	Press	Click
Move among the Query	F6, SHIFT+F6	Anywhere in the target pane
Designer panes		

Diagram Pane

То	Press	Click
Move among tables, other	TAB, or	The table, table-structured
table-structured objects,	SHIFT+TAB	object, or join line to move to
(and to join lines, if		
available)		
Move between columns in	Arrow keys	The column to go to
a table or table-structured		
object		
Choose the selected data	SPACEBAR ¹ or	The check box next to the
column for output	PLUS key	name of the column
Remove the selected data	SPACEBAR ¹ or	The check box next to the
column from the query	MINUS key	name of the column
output		
Remove the selected table,	DELETE	Right-click, and then choose
table-structured object, or		Remove
join line from the query		

1 If multiple items are selected, pressing this key affects all selected items. Select multiple items by holding down the SHIFT key while clicking them. Toggle the selected state of a single item by holding down CTRL while clicking it.

Grid Pane

То	Press	Click
Move among cells	Arrow keys or TAB or SHIFT+TAB	The target cell
Move to the last row in the current column	CTRL+DOWN ARROW	
Move to the first row in the current column	CTRL+UP ARROW	
Move to the top left cell in the visible portion of grid	CTRL+HOME	
Move to the bottom right cell	CTRL+END	
Move in a drop-down list	UP ARROW or DOWN ARROW	The button in the cell
Select an entire grid column	CTRL+SPACEBAR	The column header
Select an entire grid row		The button to the left of the row
Toggle between edit mode and cell selection mode	F2	
Copy selected text in cell to the Clipboard (in edit mode)	CTRL+C	
Cut selected text in cell and place it on the Clipboard (in edit mode)	CTRL+X	
Paste text from the Clipboard (in edit mode)	CTRL+V	
Toggle between insert and overstrike mode while editing in a cell	INS	
Toggle the check box in	SPACEBAR ¹	The check box

the Output column		
Clear the selected	DELETE	
contents of a cell		
Remove row containing	DELETE ¹	
selected data column		
from the query		
Clear all values for a	DELETE	
selected grid column		
Insert row between	INS after you select	
existing rows	grid row	
Add an Or column	INS after you select	
	any Or column	

1 If multiple items are selected, pressing this key affects all selected items.

SQL Pane

You can use the standard Windows editing keys when working in the SQL pane, such as CTRL+Arrow keys to move between words, and the Cut, Copy, and Paste commands on the Edit menu.

Note You can only insert text; there is no overstrike mode.

Results Pane

То	Press	Click
Move between cells	Arrow keys or TAB	The target cell
	or SHIFT+TAB	
Move to first or last cell	HOME or END	
in current row		
Move to the first row in	CTRL+UP	
the current column	ARROW	
Move to the top left cell	CTRL+HOME	
Move to the bottom cell	CTRL+DOWN	
in the first column	ARROW	
Select to first character	SHIFT+HOME	

in a cell		
Select to last character in a cell	SHIFT+END	
Select an entire grid column	CTRL+SPACEBAR	The column header
Select an entire grid row		The button to the left of the row
Toggle between edit mode and cell selection mode	F2	
Toggle between insert and overstrike mode while editing in a cell	INS	
Insert a new row into the grid (moves to last row in grid)	INS while cell is selected	
Delete a row from the table	DELETE ¹	
Undo changes for the current cell	ESC in cell that has changed	
Undo changes for the current row	ESC in any cell that has not changed	
Enter null into a cell	CTRL+0	
Copy selected columns or rows to the Clipboard	CTRL+C	
Copy selected text in cell to the Clipboard (in edit mode)	CTRL+C	
Cut selected text in cell to the Clipboard (in edit mode)	CTRL+X	
Paste text from the Clipboard (in edit mode)	CTRL+V	

1 If multiple items are selected, pressing this key affects all selected items.

See Also

<u>Reference</u>

Comparison Operators

You can use any standard SQL operators in a <u>search condition</u>. When you use operators in a search condition, the following rules apply:

- The data types of the data used in the comparison must match. That is, only text can be compared to text, numbers to numbers, and so on. For information about converting data types, see <u>Data Type Conversion</u>.
- If you compare text data, the result depends on the character set currently in use. For example, if a table was created using Scandinavian characters, the search results might differ depending on whether your current character set (code page) is Scandinavian or another character set.
- If a comparison value is null, the result is unknown. Nulls are not matched to any value, including other instances of null. For example, if you are searching for a name beginning with the letter "M" or higher (name >= 'M'), and some of the rows contain no value, those rows do not appear, no matter what comparison operator you use.

The following table summarizes search condition operators that are defined for standard SQL and how the operators are entered in the <u>Grid Pane</u>. For more information, see <u>Operators</u>.

Operator	Meaning	Grid pane example ¹	SQL pane example
=	Equal.	= 'Smith'	SELECT fname, lnam FROM employees WHERE lname = 'Sm
<>> !=	Not equal to.	<> 'Active'	SELECT fname, lnam FROM employees WHERE status <> 'Ao

>	Greater than.	> '01 Jan 1995' ²	SELECT fname, lnam FROM employees WHERE hire_date > '12/31/90'
<	Less than.	< 100	SELECT fname, lnam FROM employees WHERE job_lvl < 10
>= !<	Greater than or equal to.	>= 'T'	SELECT au_lname FROM authors WHERE au_lname >=
<= !>	Less than or equal to.	<= '01 Jan 1995' ²	SELECT fname, lnam FROM employees WHERE hire_date <= '01/01/95'
BETWEEN expr1 AND expr2	of values.	BETWEEN '01 Jan 1995' AND '31 Dec 1995' ²	SELECT fname, lnam FROM employees WHERE hire_date BETWEEN '12/31/9(AND '12/31/91'
IS [NOT] NULL	Tests whether contents of column or result of <u>expression</u> is null.	IS NULL	SELECT fname, lnam FROM employees WHERE photo_on_fi
[NOT] LIKE	Performs pattern matching	LIKE ('MAC%')	SELECT fname, lnam FROM employees

	(usually restricted to character data types).		WHERE lname LIKE
expr1 [NOT] IN (val1, val2,) – or – expr1 [NOT] IN (subquery)	Matches list of specific values by testing whether <i>expr1</i> appears either in a list of values or in the <u>result set</u> of a	IN ('SW', 'SE') supplier_id IN (subquery)	SELECT fname, lnam FROM employees WHERE sales_region SELECT product_nan FROM products WHERE supplier_id I (SELECT supplier_id FROM supplier WHERE (country =
ANY (SOME)	subquery. Tests whether one or more rows in the result set of a subquery meet the specified condition. (ANY and SOME are synonyms; the Query	<> ANY (subquery)	SELECT au_lname, a FROM authors where city <> any (SELECT city FROM

	Designer will use ANY when creating an SQL statement.)		
ALL	Tests whether all rows in the result set of a subquery meet the specified condition.	advance > ALL (subquery)	SELECT title FROM where advance > all (SELECT advance F publishers,titles where titles.pub_id = publishers.pub_id AND pub_name = 'Alogdata Infosyster
[NOT] EXISTS	Tests whether a subquery returns any results whatsoever (not a specific result).	EXISTS (subquery)	SELECT product_nan FROM products WHERE EXISTS (SELECT * FROM orders, products WHERE orders.prod = products.prod_id)

1 For clarity, the <u>Grid pane</u> examples include only one example for each operator and do not indicate which data column is being searched.

2 Dates can be entered in the Grid pane using the format specified in the Windows Regional Settings dialog box. For details, see <u>Entering Search Values</u>.

See Also

Reference

Logical Operators

You can combine or modify <u>search conditions</u> using the standard logical operators listed in the following table. The operators are listed in the order that they are evaluated.

Operator	Meaning	Example	
NOT	Logical opposite of condition	SELECT * FROM employee	
		WHERE NOT (fname = 'Ann')	
AND	Both conditions	SELECT * FROM employee	
	must apply	WHERE lname = 'Smith' AND fname =	= 'A:
OR	Either condition	SELECT * FROM employee	
	can apply	WHERE region = 'UK' OR region = 'FI	RA'

See Also

<u>Reference</u>

Wildcard Characters

You can search for patterns within data columns or <u>expressions</u> by using <u>wildcard characters</u>. For example, you can search for all employees whose last names begin with "Mac" or end with "son."

In this topic, you can read about:

- <u>Wildcard Characters</u>
- <u>Searching for Characters Used as Wildcard Characters</u>
- <u>Searching Datetime Columns</u>
- Examples of Wildcard Searches

Wildcard Characters

You can use wildcard characters to search any columns that can be treated as text strings. Columns with the data type character can always be treated as text strings.

To search for patterns, use the LIKE operator, and then substitute wildcard characters for one or more characters in the search string. You can use either of the following wildcard characters:

Wildcard character	Meaning
% (percent symbol)	Zero or more characters in that position
_ (underscore)	One character at that position

For example, to search for all names beginning with "Mac," you could specify the search condition LIKE 'Mac%'. To find names such as "Jan," "Jen," and "Jon," you could specify the search condition LIKE 'J_n'.

Searching for Characters Used as Wildcard Characters

In some cases, you might need to search for a string that contains one of the characters used as a wildcard character. For example, in a titles table you might want to find all the publications that contain the string "10%" as part of the title. Because "%" is part of the string you are searching for, you must specify that you mean it as a literal string and not a wildcard character.

To search for characters that can be interpreted as wildcard characters, you can specify an <u>escape character</u>. Place the escape character immediately in front of the "%" or "_" character that you mean literally. To specify the escape character, include an ESCAPE clause immediately after the LIKE search criterion. You can do this in either the Criteria column of the <u>Grid pane</u> or in the <u>SQL pane</u>.

For example, imagine that you want to find all titles that contain the string "10%". Suppose you want to define the character "#" as an escape character, which allows you to include "#" in front of the "%" character that is meant literally. You can enter this in the Grid pane:

LIKE '%10#%%' ESCAPE '#'

The resulting WHERE clause in the SQL statement looks like this:

WHERE title LIKE '%10#%%' ESCAPE '#'

Note You can define an escape character only in the **SQL** pane.

Searching Datetime Columns

When working with a datetime data type column, you can search any portion of the date or time, including text abbreviations of the month and complete years.

For example, you can use the following LIKE clause to search for all rows in which the date falls within 1994:

LIKE '%1994%'

The following searches for all rows in which the date falls within the month of January, regardless of year:

LIKE 'Jan%'

For more information, see LIKE.

Examples of Wildcard Searches

The following examples illustrate the use of wildcard characters.

Search expression	Description	Sample matches
LIKE 'Mac%'	Finds values beginning with "Mac"	Mac MacIntosh Mackenzie
LIKE 'J%n'	Finds values starting with "J" and ending with "n"	Jon Johnson Jason Juan
LIKE '%son'	Finds values ending with "son"	Son Anderson
LIKE '%sam%'	Finds values with "sam" anywhere in the string	Sam Samson Grossam
LIKE '%Mar%'	Finds values in a datetime column that fall in the month of March, regardless of year	3/1/94 01 Mar 1992
LIKE '%1994%'	Finds values in a datetime column for the year 1994.	12/1/94 01 Jan 1994
LIKE 'Mac_'	Finds values with exactly four characters, the first three being "Mac"	Mack Macs
LIKE '_dam'	Finds values with exactly four characters, the last three being "dam"	Adam Odam
LIKE '%s_n'	Finds values containing "s" and "n" at the end of the value, with any one character between them and any number of	Anderson Andersen Johnson san sun

rs in front of them

See Also

<u>Reference</u>

Rules and Grid-Pane Conventions for Combining Search Conditions

You can create queries that include any number of <u>search conditions</u>, linked with any number of AND and OR operators. A query with a combination of AND and OR clauses can become complex, so it is helpful to understand how such a query is interpreted when you execute it, and how such a query is represented in the <u>Grid</u> and <u>SQL</u> panes.

Note For details about search conditions that contain only one AND or OR operator, see <u>Specifying Multiple Search Conditions for One Column</u> and <u>Specifying Multiple Search Conditions for Multiple Columns</u>.

Below you will find information about:

- The precedence of AND and OR in queries that contain both.
- How the conditions in AND and OR clauses relate logically to one another.
- How the Query Designer represents in the Grid pane queries that contain both AND and OR.

To help you understand the discussion below, imagine that you are working with an employee table containing the columns hire_date, job_lvl, and status. The examples assume that you need to know information such as how long an employee has worked with the company (that is, what the employee's hire date is), what type of job the employee performs (what the job level is), and the employee's status (for example, retired).

Precedence of AND and OR

When a query is executed, it evaluates first the clauses linked with AND, and then those linked with OR.

Note The NOT operator takes precedence over both AND and OR.

For example, to find either employees who have been with the company for more than five years in lower-level jobs or employees with middle-level jobs without regard for their hire date, you can construct a WHERE clause such as the following:

WHERE

```
hire_date < '01/01/90' AND
job_lvl = 100 OR
job_lvl = 200
```

To override the default precedence of AND over OR, you can put parentheses around specific conditions in the SQL pane. Conditions in parentheses are always evaluated first. For example, to find all employees who have been with the company more than five years in either lower or middle-level jobs, you can construct a WHERE clause such as the following:

WHERE

hire_date < '01/01/90' AND (job_lvl = 100 OR job_lvl = 200)

Tip It is recommended that, for clarity, you always include parentheses when combining AND and OR clauses instead of relying on the default precedence.

How AND Works with Multiple OR Clauses

Understanding how AND and OR clauses are related when combined can help you construct and understand complex queries in the Query Designer.

If you link multiple conditions using AND, the first set of conditions linked with AND applies to all the conditions in the second set. In other words, a condition linked with AND to another condition is distributed to all the conditions in the second set. For example, the following schematic representation shows an AND condition linked to a set of OR conditions:

A AND (B OR C)

The representation above is logically equivalent to the following schematic representation, which shows how the AND condition is distributed to the second set of conditions:

(A AND B) OR (A AND C)

This distributive principle affects how you use the Query Designer. For example, imagine that you are looking for all employees who have been with the company more than five years in either lower or middle-level jobs. You enter the following WHERE clause into the statement in the **SQL** pane:

WHERE (hire_date < '01/01/90') AND (job_lvl = 100 OR job_lvl = 200)

The clause linked with AND applies to both clauses linked with OR. An explicit way to express this is to repeat the AND condition once for each condition in the OR clause. The following statement is more explicit (and longer) than the previous statement, but is logically equivalent to it:

WHERE (hire_date < '01/01/90') AND (job_lvl = 100) OR (hire_date < '01/01/90') AND (job_lvl = 200)

The principle of distributing AND clauses to linked OR clauses applies no matter how many individual conditions are involved. For example, imagine that you want to find higher or middle-level employees who have been with the company more than five years or are retired. The WHERE clause might look like this:

WHERE

(job_lvl = 200 OR job_lvl = 300) AND (hire_date < '01/01/90') OR (status = 'R')

After the conditions linked with AND have been distributed, the WHERE clause will look like this:

WHERE

(job_lvl = 200 AND hire_date < '01/01/90') OR (job_lvl = 200 AND status = 'R') OR (job_lvl = 300 AND hire_date < '01/01/90') OR (job_lvl = 300 AND status = 'R')

How Multiple AND and OR Clauses Are Represented in the Grid Pane

The Query Designer represents your search conditions in the <u>Grid pane</u>. However, in some cases that involve multiple clauses linked with AND and OR, the representation in the Grid pane might not be what you expect. In addition, if you modify your query in the Grid or <u>Diagram</u> panes, you might find that your SQL statement has been changed from what you entered.

In general, these rules dictate how AND and OR clauses appear in the **Grid** pane:

- All conditions linked with AND appear in the **Criteria** grid column or in the same **Or** ... column.
- All conditions linked with OR appear in separate **Or** ... columns.
- If the logical result of a combination of AND and OR clauses is that the AND is distributed into several OR clauses, the **Grid** pane represents this explicitly by repeating the AND clause as many times as necessary.

For example, in the SQL pane you might create a search condition such as the following, in which two clauses linked with AND take precedence over a third one linked with OR:

WHERE (hire_date < '01/01/90') **AND** (job_lvl = 100) **OR** (status = 'R')

The Query Designer represents this WHERE clause in the **Grid** pane as follows:

Column	Criteria	Or	
hire_date	< '1/1/90'		
job_lvl	= 100		
status		='B'	

However, if the linked OR clauses take precedence over an AND clause, the AND clause is repeated for each OR clause. This causes the AND clause to be distributed to each OR clause. For example, in the **SQL** pane you might create a WHERE clause such as the following:

WHERE (hire_date < '01/01/90') **AND** ((job_lvl = 100) **OR** (status = 'R'))

The Query Designer represents this WHERE clause in the **Grid** pane as follows:

Column	Criteria	Or
hire_date	< '1/1/90'	< '1/1/90'
job_lvl	= 100	
status		='R'

If the linked OR clauses involve only one data column, the Query Designer can place the entire OR clause into a single cell of the grid, avoiding the need to repeat the AND clause. For example, in the SQL pane you might create a WHERE clause such as the following:

WHERE (hire_date < '01/01/90') **AND** ((status = 'R') **OR** (status = 'A'))

The Query Designer represents this WHERE clause in the **Grid** pane as follows:

	Column	Criteria
a	hire_date	< '1/1/90'
	status	= ' R' OR = ' A'

If you make a change to the query (such as changing one of the values in the Grid pane), the Query Designer recreates the SQL statement in the SQL pane. The recreated SQL statement will resemble the Grid pane display rather than your original statement. For example, if the Grid pane contains distributed AND clauses, the resulting statement in the SQL pane will be recreated with explicit distributed AND clauses. For details, see "How AND Works with Multiple OR Clauses" earlier in this topic.

See Also

<u>Reference</u>

Rules for Entering Search Values

This topic discusses the conventions you must use when entering the following types of literal values for a <u>search condition</u>:

- <u>Text values</u>
- <u>Numeric values</u>
- <u>Dates</u>
- Logical values

Note The information in this topic is derived from the rules for standard SQL-92. If you have questions about how to enter search values, see the <u>Transact-SQL Reference</u>.

Searching on Text Values

The following guidelines apply when you enter text values in search conditions:

• **Quotation marks** Enclose text values in single quotation marks, as in this example for a last name: 'Smith'

If you are entering a search condition in the <u>Grid pane</u>, you can simply type the text value and the Query Designer will automatically put single quotation marks around it.

Note In SQL Server, the Query Designer always interprets double quotation marks as database object delimiters. For details, see <u>Query Designer Considerations for SQL Server</u> <u>Databases</u>.

• **Embedding apostrophes** If the data you are searching for contains a single quotation mark (an apostrophe), you can enter two single

quotation marks to indicate that you mean the single quotation mark as a literal value and not a delimiter. For example, the following condition searches for the value "Swann's Way": ='Swann''s Way'

- **Length limits** Do not exceed the maximum length of the SQL statement when entering long strings. The SQL standard does not impose a maximum length on literal strings, but most ODBC drivers have a maximum length limit for a statement.
- **Case sensitivity** Follow the case sensitivity rules for the database you are using. The database you are using determines whether text searches are case sensitive.

If you are unsure about whether the database uses a case-sensitive search, you can use the UPPER or LOWER functions in the search condition to convert the case of the search data, as illustrated in the following example:

```
WHERE UPPER(lname) = 'SMITH'
```

For details about the functions to convert to uppercase and lowercase letters, see <u>Functions for Expressions</u>.

Searching on Numeric Values

The following guidelines apply when you enter numeric values in search conditions:

- **Quotation marks** Do not enclose numbers in quotation marks.
- **Non-numeric characters** Do not include non-numeric characters except the decimal separator (as defined in the **Regional Settings** dialog box of Windows Control Panel) and negative sign (-). Do not include digit grouping symbols (such as a comma between thousands) or currency symbols.

- **Decimal marks** If you are entering whole numbers, you can include a decimal mark, whether the value you are searching for is an integer or a real number.
- Scientific notation You can enter very large or very small numbers using scientific notation, as in this example: > 1.23456e-9

Note For details about entering currency values in SQL Server, see <u>Query</u> <u>Designer Considerations for SQL Server Databases</u>.

Searching on Dates

The format you use to enter dates depends on the database you are using and in what pane of the Query Designer you are entering the date. The Query Designer can work with the following date formats:

- Locale-specific The format specified for dates in the Windows Regional Settings Properties dialog box.
- **Database-specific** Any format understood by the database.
- ANSI standard date A format that uses braces, the marker 'd' to designate the date, and a date string, as in the following example: { d '1990-12-31' }
- ANSI standard datetime Similar to ANSI-standard date, but uses 'ts' instead of 'd' and adds hours, minutes, and seconds to the date (using a 24-hour clock), as in this example for December 31, 1990:
 { ts '1990-12-31 00:00:00' }

In general, the ANSI standard date format is used with databases that represent dates using a true date data type. In contrast, the datetime

format is used with databases that support a datetime data type.

The following table summarizes the date format that you can use in different panes of the Query Designer.

Pane	Date format
Grid	Locale-specific Database-specific ANSI standard Dates entered in the Grid pane are converted to a database-compatible format in the SQL pane.
SQL	Database-specific ANSI standard Dates entered into the SQL pane are converted to the locale-specific format in the Grid pane.
Results	Locale-specific

Searching on Logical Values

The format of logical data varies from database to database. Very frequently, a value of False is stored as zero (0). A value of True is most frequently stored as 1 and occasionally as -1. The following guidelines apply when you enter logical values in search conditions:

- To search for a value of False, use a zero, as in the following example: SELECT * FROM authors WHERE contract = 0
- If you are not sure what format to use when searching for a True value, try using 1, as in the following example: SELECT * FROM authors WHERE contract = 1

 Alternatively, you can broaden the scope of the search by searching for any non-zero value, as in the following example: SELECT * FROM authors WHERE contract <> 0

See Also

Reference

Aggregate Functions

To summarize all the data in a table, you create an <u>aggregate query</u> that involves a function such as SUM() or AVG(). When you run the query, the <u>result set</u> contains a single row with the summary information. For example, you can calculate the total price of all books in the titles table by creating a query that sums the contents of the price column. The resulting query output might look like this:

	total_price
	236.26
*	

The corresponding <u>SQL</u> statement might look like this:

SELECT SUM(price) total_price FROM titles

You can use the following <u>aggregate functions</u>:

Aggregate function	Description	
AVG(expr)	Average of the values in a column. The column can contain only numeric data.	
COUNT(<i>expr</i>), COUNT(*)	 A count of the values in a column (if you specify a column name as <i>expr</i>) or of all rows in a table or group (if you specify *). COUNT(<i>expr</i>) ignores null values, but COUNT(*) includes them in the count. 	
MAX(expr)	Highest value in a column (last value alphabetically for text data types). Ignores null values.	
MIN(<i>expr</i>)	Lowest value in a column (first value alphabetically for text data types). Ignores null values.	
SUM(expr)	Total of values in a column. The column can contain only numeric data.	

When you use an aggregate function, by default the summary information includes all specified rows. In some instances, a result set includes non-unique rows. You can filter out non-unique rows by using the DISTINCT option of an aggregate function.

You can combine aggregate functions with other expressions to calculate other summary values. For details, see <u>Summarizing Values Using Custom</u> <u>Expressions</u>.

See Also

Reference

Structure of Expressions

An expression consists of any combination of <u>column</u> names, literals, operators, or functions. Follow these guidelines in combining elements to form expressions:

• Reference columns by typing their names. If your query uses more than one table and if you use a column name that is not unique, you must add the table name and a period to the column name. The following example shows the column name job_id qualified with the table name employee:

employee.job_id

• Include literal text by enclosing it in single quotation marks; no quotation marks are necessary for numbers.

Note In SQL Server, the Query Designer always interprets double quotation marks as database object delimiters. For details, see <u>Query Designer Considerations for SQL Server</u> <u>Databases</u>.

- Use standard arithmetic operators for numbers and a concatenation operator for combining strings. For details, see <u>Operators for Expressions</u>.
- Include parentheses to establish precedence of operators.
- If you include a function, use these same guidelines for the arguments passed to the function. That is, reference columns by typing their names, enclose literal text in single quotation marks, and so on. For more information, see <u>Functions for Expressions</u>.
- If you pass column names as function arguments, be sure the data type of the column is appropriate for the function argument.

• You can include user-defined functions returning a scalar value in an expression. For more information about user-defined functions, see <u>User-Defined Functions</u>.

Expression	Result
SELECT (price * .9)	Displays a discounted price (10% off
FROM products	the value in the price column).
SELECT (lname + ', ' + fname) FROM employee	Displays the concatenated values of the last name and first name columns with a comma between them.
SELECT sales.qty, titles.price	After joining two tables, sorts the
FROM sales INNER JOIN titles ON	result set by the total value of an order (quantity times price).
sales.title_id = titles.title_id ORDER BY	
(sales.qty * titles.price)	
SELECT au_lname, au_fname	Displays authors whose area code is
FROM authors	in the San Francisco area.
WHERE	
(SUBSTRING(phone, 1, 3) = '415')	
SELECT ord_num, ord_date	Finds all orders in the sales table that
FROM sales	were made in the last 10 days. Today's
WHERE	date is returned by the GETDATE()
(ord_date >=	function.
DATEADD(day, -10,	
GETDATE()))	

The following table illustrates the use of expressions in a query.

Reference

Operators for Expressions

You can use a variety of operators in constructing <u>expressions</u> for your query, including mathematical and text operators.

Mathematical Operators

The following table lists the mathematical operators you can use in constructing an expression.

Operator	Meaning
+, -	Unary positive, negative
+	Addition
-	Subtraction
*	Multiplication
/	Division

Note For more information, see <u>Operators</u>.

If you use more than one mathematical operator in an expression, the Query Designer processes the expression according to the following operator precedence. To override the default precedence, use parentheses around the portion of the expression that is to be evaluated first. If more than one operator of the same level is included, the operators are evaluated left to right.

1. Unary + and -

2. * and /

3. + and -

Text Operator

You can perform one operation on text: concatenation or the linking together of

strings. To concatenate a string, use the "+" operator in the Grid pane. For more information, see <u>Operators</u>.

You can also use the <u>ODBC</u> CONCAT function, which is supported by ODBCcompliant drivers. For example, the following expression illustrates how you can use the CONCAT function to combine authors' last name and first name with a comma between them:

{ fn CONCAT(au_lname,{ fn CONCAT (', ', au_fname)})}

For more details about functions you can use, see <u>Functions for Expressions</u>.

See Also

Reference

Functions for Expressions

You can call a number of functions when you are building an <u>expression</u>, including:

- String (character) functions
- Date functions
- Mathematical functions
- System functions
- Other functions, such as those to convert data from one type to another
- User-defined functions

For more information, see <u>Functions</u>.

If you are creating queries (not views, stored procedures, or triggers) that might be run against different databases, you can also use <u>ODBC</u> functions. ODBC syntax includes the "fn" qualifier in front of the function name and braces around the entire function. For example, the following expression uses an ODBC function to convert text to lowercase letters:

```
{fn LCASE ( address ) }
```

The Query Designer can help you work with functions by:

- Correctly inserting quotation marks in function arguments
- Validating the data types of arguments
- Validating the data types of return values

For information on ODBC, see the Data Access Services section of the MSDN® Online Library <u>Microsoft Web site</u>. For more information on functions, see <u>User-Defined Functions</u>.

Note You can use a special set of functions, the <u>aggregate functions</u> such as SUM() and AVG(), to create queries that summarize data. For details, see <u>Summarizing and Grouping</u>.

String Functions

The following table contains samples of string functions. For more information, see <u>String Functions</u> and <u>Using String Functions</u>.

Function	Description	Example
LCASE() ¹ , LOWER()	Converts strings to	SELECT UPPER(substring(lname, 1, 1)) +
	lowercase	LOWER(substring (lname, 2, 99)) FROM employee
		Displays a last name after the first character is conve
LTRIM()	Removes leading spaces from	SELECT stor_name, LTRIM(stor_address) FROM stores
	a string	Displays an address <u>column</u> after extraneous spaces
SUBSTRING()	Extracts one or more characters from a string	SELECT SUBSTRING(phone,1,3) FROM employee Displays the first three characters (the area code) of a
UCASE() ¹ , UPPER()	Converts strings to uppercase	SELECT * FROM employee WHERE UPPER(lname) = 'SMITH' Converts the contents of the lname column t

¹ If calling as an ODBC function, use syntax such as: { fn LCASE(*text*) }.

Date Functions

The following table contains samples of date functions. For more information, see <u>Date and Time Functions</u>.

Function	Description	Example
DATEDIFF()	Calculates an interval between two dates.	SELECT fname, lname, hire_date FROM employee WHERE DATEDIFF(year, hire_date, getdat Locates all employees hired more than five years ago.
DATEPART(Returns the specified portion of a date or datetime column, including the day, month, or year.	SELECT DATEPART(year, hire_date) FROM employee Displays only the year in which an employee was hired (not the full date).
CURDATE() ¹ , GETDATE() or DATE()	Returns the current date in datetime format. This function is useful as input for many other date functions, such as	SELECT order_id FROM orders WHERE order_date = GETDATE() Displays orders placed today.

calculating	
an interval	
forward or	
backward	
from today.	

¹ If calling as an ODBC function, use syntax such as: { fn CURDATE() }.

Mathematical Functions

The following functions are typical of those available in many databases. Refer to <u>Mathematical Functions</u> for more information.

Note You can use the <u>aggregate functions</u> AVG(), COUNT(), MAX(), MIN(), and SUM() to create averages and totals in your report. For details, see <u>Summarizing and Grouping</u>.

Function	Description	Example
ROUND()	number off to the specified	SELECT ROUND(qty * (price * discount), 2) FROM sales Displays a total price based on a discount, then rounds the results off to two decimal places.
FLOOR()	number down to the nearest (cmallest)	UPDATE titles SET price = FLOOR(price) Rounds all prices in the titles table down to the nearest whole number.
CEILING()	Rounds a number up to the nearest whole number	INSERT INTO archivetitle SELECT title, CEILING(price) FROM titles

	Copies the title and the price (rounded up to the nearest integer) from the titles table to the
	archivetitle table.

System Functions

The following functions are typical of those available in many databases. For more information, see <u>System Functions</u>.

Function	Description	Example
DATALENGTH()	number of bytes used by the specified	SELECT DATALENGTH(au_lname + ', ' + au_fname) FROM authors Lists the number of bytes required for the combination of last and first names.
USER() ¹ , USER_NAME()	current user name	SELECT company_name, city, phone FROM customers WHERE salesperson = USER_NAME() Creates a list of customers for the salesperson who runs the query.

¹ If calling as an ODBC function, use syntax such as: { fn USER() }.

Other Functions

The following functions illustrate utility functions available in many databases. For more information, see <u>Functions</u>.

Function	Description	Example

CONVERT(Converts	SELECT 'Hired: ' + CONVERT(char (11),
)	data from	
	one data	hire_date)
	type into	FROM employee
	another.	r restri employee
	Useful to	Displays a date with a caption in front of it; the
	format data	CONVERT() function creates a string out of the
	or to use the	date so that it can be concatenated with a literal
	contents of	string.
	a data	0
	column as	
	an	
	argument in	
	a function	
	that .	
	requires a	
	different	
	data type.	
SOUNDEX(SELECT au_lname, au_fname
)	Soundex	FROM authors
	code for the	WHERE SOUNDEX(au_fname) = 'M240'
	specified	
	expression, which you	Searches for names that sound like "Michael".
	can use to	
	create	
	"sounds	
	like''	
	searches.	
STR()	Converts	SELECT str(job_id) + ' ' +
	numeric	str(job_lvl)
	data into a	FROM employee
	character	
	string so	Displays the job_id and job_lvl columns (both
	you can	numeric) in a single string.
	manipulate	numerie) in a single string.
	it with text	

operators.

See Also

<u>Reference</u>

Predefined Variables for Expressions

In addition to using <u>column</u> names, literals, operators, and functions in an <u>expression</u>, you can use predefined variables that have defined meanings or values. For example, you can use a predefined variable to display the user name for the current user or to search for data columns that contain no value (null).

See Also

Reference | Reserved Keywords | Functions

Dialog Boxes

For information about the dialog boxes used by the Visual Database Tools, see the following topics:

Database Designer Dialog Boxes

Query Designer Dialog Boxes

Database Designer Dialog Boxes

Documentation is available for the following Database Designer user interface elements:

- Add Table Dialog Box
- <u>Choose Name Dialog Box</u>
- Column Selection Dialog Box
- <u>Create Relationship Dialog Box</u>
- Database Changes Detected Dialog Box
- <u>Datatype Change Required Dialog Box</u>
- <u>Page Setup Dialog Box</u>
- <u>Save Change Script Dialog Box</u>
- <u>Save Dialog Box</u>
- Save Incomplete Dialog Box
- <u>Unsaved Changes Exist Dialog Box</u>
- <u>Validation Warnings Dialog Box</u>

Add Table Dialog Box (Database Designer)

Enables you to add tables in Database Designer.

Add

Adds the selected table or tables.

If you add a table or view that already appears on the Diagram pane, you will get a message warning you that the table already exists on the diagram.

Note If you want to add several tables to the diagram, you can select them all before clicking Add. Alternatively, you can double-click each table you want to add, then click Close when you are finished.

Close

Closes the dialog box without adding further tables.

See Also

Database Designer Dialog Boxes | Add Table Dialog Box (Query and View Designers)

Choose Name Dialog Box

Enables you to change the system-assigned name of a new table before that table is created in the database.

Enter a name for the table

Displays the system-assigned name of the table. Type a new name for the table in the text box. Although you can later use the Table property page to change the system-assigned name, it is best if you make any name changes now. Changing the name of an existing database object is risky, because you can invalidate programs and queries that refer to that object by name.

OK

Creates the table with the name you specified. The table is created in the database when you save the table or diagram.

Cancel

Cancels the creation of the new table.

For more information about creating tables, see <u>Adding a Table to a Database</u> <u>Diagram</u>.

See Also

Database Designer Dialog Boxes | Renaming a Table

Create Relationship Dialog Box

Enables you to confirm the related columns and to set properties for a new relationship.

This dialog box appears when you draw a relationship line between two tables in your database diagram.

Relationship name

Displays the system-assigned name of the relationship. To rename the relationship, type a new name in the text box.

Primary key table

Shows the name of the primary key table in the relationship, followed by the columns that make up the primary key. You can select different columns to match the columns shown under **Foreign key table**.

Foreign key table

Shows the name of the foreign key table in the relationship, followed by the columns that make up the foreign key. You can select different columns to match the columns shown under **Primary key table**.

Check existing data on creation

Applies the relationship to existing data in the foreign key table when the relationship is created. An error message will notify you of any data that violates the constraint if this box is selected.

Enforce relationship for INSERT and UPDATE

Selecting this options enforces the constraint whenever data is added to or

updated in the foreign key table using these statements.

Enforce relationship for replication

Selecting this option enforces referential integrity for the relationship whenever the foreign key table is replicated to a different database.

Cascade update related fields

Instructs the database to propagate new key values to corresponding foreign key fields whenever a primary key value is updated.

Cascade delete related fields

Instructs the database to delete corresponding rows from the foreign key table whenever rows from the primary key table are deleted.

OK

Creates the relationship with the properties you selected.

Cancel

Erases the relationship line from your database diagram. The relationship is not created.

For more information about creating relationships between tables, see <u>Creating a</u> <u>Relationship Between Tables</u>.

See Also

Database Designer Dialog Boxes | Relationships Property Page

Column Selection Dialog Box

Lets you change the Custom view for tables in the database diagram. Custom view shows only the column properties identified by the user.

This dialog box appears when you right-click a table and then choose **Modify Custom View** from the shortcut menu.

Available columns

Lists all columns existing in the selected database table. The columns listed here depend on the properties of the database table and the type of database. Highlight the desired column and use the arrow buttons to move the columns to the **Selected columns** box.

Selected columns

Displays the column properties currently defined for the Custom view. Use the arrow buttons to add and remove column properties to the Selected Columns list.

Move

Use the up and down arrow buttons to move highlighted columns up or down in the **Selected columns** list. The column properties will be displayed in the table in the order shown in the **Selected columns** list.

Save as default

Replaces the default Custom view with the columns selected in this dialog box. If not selected, the column selection specified in the dialog box will be applied only to the selected table in the database diagram.

OK

Saves the Custom view.

Cancel

Cancels the modification of Custom view.

For more information about Custom views, see <u>Changing a Table View</u>.

See Also

Database Designer Dialog Boxes

Database Changes Detected Dialog Box

Appears if you attempt to save a database diagram or selected tables but some of the database objects that will be affected by the save action are out of date with the database. Accepting the changes shown in this dialog box updates the database to match your diagram and overwrites other users' changes.

Note Although you cannot undo changes made to a table or database diagram, the changes are not saved to the database until you save the table or diagram. You can discard any unsaved changes by choosing **No** and closing all open diagrams without saving them.

Yes

Updates the database with all the changes shown in the list.

No

Cancels the save action.

Save Text File

Displays the Save As dialog box, letting you specify a location for a text file containing a list of the database changes.

For more information about saving database objects that have changed since you began working in the Database Designer, see <u>Reconciling a Database Diagram</u> with a Modified Database.

See Also

Database Designer Dialog Boxes

Datatype Change Required Dialog Box

Appears when you change the data type, length, scale, precision, or collation of a column that participates in a relationship.

Yes

Changes the data type property of the related columns shown in the list so that existing relationships are preserved.

No

Cancels the data type change and restores the previous data type to the column you just changed.

See Also

Database Designer Dialog Boxes

Define Column Collation Dialog Box

Lets you specify a collation sequence for the column. A column's collation sequence is used in any operation that compares values of the column to each other or to constant values. It is also affects the behavior of some string functions, such a SUBSTRING, and CHARINDEX. For a complete list of the effects of a column's collation setting, see <u>SQL Server Collation Fundamentals</u>.

This dialog box appears in any of several situations:

- If you enter an invalid collation name in the **Collation** box in the bottom portion of the Table Designer or in the **Columns** property page
- If you click in the **Collation** box in the Columns property page or in the bottom portion of the Table Designer, then click the button appearing to the right of the control.

SQL Collation

Select SQL Collation if you want to choose among the collation sequences defined by SQL Server. Then select a collation sequence from the drop-down list.

Windows Collation

Select Windows Collation if you want to choose among the collation sequences defined by Windows. Then select a collation sequence from the drop-down list.

Binary Sort

Applies only if you select Windows collation. Select this if you want comparison operations to use the binary codes of character values for comparison. If you select this option, certain alphabetic comparison options become unavailable. For example, making case-insensitive comparisons becomes unavailable because uppercase letters and lowercase letters have different binary encodings.

Dictionary Sort

Applies only if you select Windows collation. Select this if you want comparison operations to use certain alphabetic comparison options. The alphabetic comparisons options are:

- **Case Sensitive** Select this if you want comparisons to consider uppercase and lower case letters to be unequal.
- Accent Sensitive Select this if you want comparisons to consider accented and unaccented letters to be unequal. If you select this, comparisons will also consider differently accented letters to be unequal
- **Kana Sensitive** Select this if you want comparisons to consider katakana and hiragana Japanese syllables to be unequal.
- **Width Sensitive** Select this if you want comparisons to consider half-width and full-width characters to be unequal.

Reset Defaults

Applies to the column the default collation sequence for the database.

See Also

Database Designer Dialog Boxes

Page Setup Dialog Box

Use this dialog box to specify a diagram's printer settings, such as orientation, scale, and paper size. The print setup options you set for a diagram are saved with the diagram file. When you change the Print scale and Orientation options for the diagram, the page breaks for the diagram are automatically updated.

Name

Shows the name of the default printer and lists the printers that are set up on your computer.

Properties

Sets the options for the selected printer. The options available depend on the features of the printer.

Status

Shows information about the selected printer.

Туре

Displays the model and make information for the selected printer.

Where

Displays the network path or serial port information for the selected printer.

Comment

Displays additional information about the selected printer.

Size

Specifies the size of the paper or envelope you want to use.

Source

Specifies where the paper you want to use is located in the printer. Different printer models support different paper sources, such as the upper tray, envelope feed, and manual feed.

Portrait

Sets a taller-than-wide layout for the document on the page.

Landscape

Sets a wider-than-tall layout for the document on the page.

Percent

Sets the size of the diagram based on the settings as applied to a 100-percent diagram. Choose a percentage between 10 and 400.

See Also

Database Designer Dialog Boxes

Save Change Script Dialog Box

Displays a change script whenever you save a table or diagram while there are unsaved database changes in it. The dialog box lets you save the change script as a text file with an .sql extension.

This dialog box appears when you choose **Save Change Script** from the **File** menu or click the Save Change Script button **Content** on the Database Designer toolbar.

Yes

Displays the Save As dialog box, prompting you for a destination file to contain the change script.

No

Cancels the save action.

Automatically generate change script on every save

Select this option to create a change script each time you save the database diagram or selected tables. This provides a record of all changes made to the database.

If you have selected to have a change script created automatically each time you save, this dialog box will not appear when you save your database diagram.

For more information about saving SQL change scripts, see <u>Saving a Change</u> <u>Script</u>.

See Also

Database Designer Dialog Boxes | Saving a Change Script

Save Dialog Box

Appears when you save a database diagram or selected tables. This dialog box lets you confirm the tables that will be saved. It does not appear when your diagram contains only one modified table.

Note Although you cannot undo changes made to a table or database diagram, the changes are not saved to the database until you save the table or diagram. You can discard any unsaved changes by closing all open diagrams without saving them.

Yes

Saves all the tables shown in the list.

No

Cancels the save action.

Save Text File

Displays the Save As dialog box, prompting you for a location in which to save a text file listing the tables. This file provides a record of the tables that were affected by the changes you made.

See Also

Database Designer Dialog Boxes | Saving Selected Tables

Save Incomplete Dialog Box

Appears when errors are encountered while trying to save a database diagram or selected tables. This dialog box lists the following: the objects that were successfully saved in the database, the objects that were not saved, and the errors that were encountered.

OK

Returns to the diagram.

Save Text File

Displays the Save As dialog box, prompting you for a location in which to save a text file listing the tables. This file provides a record of the database changes that were successfully saved as well as the changes that could not be saved due to errors.

For more information about the errors that can occur, see:

- Error modifying column properties.
- Invalid cursor state
- <u>Unable to add constraint.</u>
- <u>Unable to create index.</u>
- <u>Unable to create relationship.</u>
- <u>Unable to modify table.</u>
- <u>Unable to preserve trigger.</u>

If the error you want to troubleshoot does not appear in this list, see <u>System</u> <u>errors.</u> for additional messages returned by Microsoft® SQL Server[™].

Error modifying column properties

Appears when your constraint expression contains an error.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]The name '[column value]' is illegal in this context. Only constants, constant expressions, or variables allowed here. Column names are illegal.

Cause

A <u>default</u> value defined for a character column is not enclosed in single quotation marks (').

Recommended solution

Enclose the value in single quotation marks in the database column's Default Value cell and then save the table.

Invalid cursor state

Appears when Microsoft® SQL Server[™] runs out of resources while attempting to save selected <u>tables</u> or a database diagram.

ODBC error text

[Microsoft][ODBC SQL Server Driver]Invalid cursor state.

Cause

There is insufficient space in your database or <u>transaction log</u> to complete the save process.

Recommended solution

Check to see if the database or the transaction log is full. If so, increase the size of the database to accommodate the change. Check other system resources or contact your System Administrator.

For more information about increasing the size of your database, see <u>Expanding</u> <u>a Database</u>.

Unable to add constraint

Appears when a new <u>constraint</u> has failed on existing data or your constraint <u>expression</u> contains an error. Compare the ODBC error text that appears in the <u>Save Incomplete dialog box</u> with the error text shown below to determine the appropriate solution.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Data exists in table '[table name]', database '[database name]', that violates CHECK constraint '[constraint name]' being added. ALTER command has been aborted.

[Microsoft][ODBC SQL Server Driver][SQL Server]Unable to create constraint. See previous errors.

Cause

Existing data does not match the <u>check constraint</u>.

Recommended solution

Change the data (for example, by using Query Designer) to match the constraint. For details, see <u>Query Designer</u>.

-or-

Clear the **Check existing data on creation** check box in the Tables property page for the check constraint in question. For more information about disabling this property, see <u>Checking Existing Data when Creating a Relationship</u>.

-or-

Change the constraint expression in the Tables property page for the check constraint in question.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column '[column name]' specified in constraint definition.

[Microsoft][ODBC SQL Server Driver][SQL Server]Unable to create constraint. See previous errors.

Cause

The text value in the check constraint expression on the Tables property page is not enclosed in single quotation marks (').

-or-

A column that participates in the check constraint has been renamed. For example, if the original constraint had the expression (cityname = 'Paris') and you renamed the column to City, you would see this error.

Recommended solution

Correct the expression and save the table.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Line [line number]: Incorrect syntax near '[operator]'.

Cause

The expression defined for the check constraint (in the Tables property page) or the <u>default</u> constraint (in the Default Value cell) is not valid <u>SQL</u> syntax. For example, the check constraint expression 'city equals Paris' was typed instead of 'city = Paris'.

Recommended solution

Correct the expression and save the table.

For more information about constraints, see <u>Constraints</u>.

Unable to create index

Appears when a new <u>index</u> has failed on existing data.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Create unique index aborted on duplicate key. Primary key is '[primary key data]'

[Microsoft][ODBC SQL Server Driver][SQL Server]Unable to create constraint. See previous errors.

Cause

A unique index was created in the Indexes/Keys property page but duplicate data exists in the database.

Recommended solution

Remove duplicate data from the database. For more information, see <u>Designing</u> <u>Queries</u>.

-or-

Change the option in the Indexes/Keys property page to allow duplicate rows in the index.

For more information about creating unique indexes, see <u>Creating a Unique</u> <u>Index</u>.

Unable to create relationship

Appears when a new referential integrity constraint has failed on existing data.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Data exists in table '[table name]', database '[database name]', that violates FOREIGN KEY constraint '[constraint name]' being added. ALTER command has been aborted.

[Microsoft][ODBC SQL Server Driver][SQL Server]Unable to create constraint. See previous errors.

Cause

Existing data fails the <u>foreign key</u> constraint.

Recommended solution

Change the data that fails the foreign key constraint by running a query to show all the foreign key values that do not match <u>primary key</u> values. For example, to find foreign key values in the job_id column of the employee table that do not match primary key values in the jobs table, run a query with this SQL syntax:

SELECT employee.emp_id, employee.job_id FROM employee LEFT OUTER JOIN jobs ON employee.job_id = jo WHERE (jobs.job_id IS NULL)

For more information, see <u>Creating Queries</u>.

-or-

Clear the **Check existing data on creation** check box in the **Relationships** property page. For more information, see <u>Checking Existing Data when Creating</u> <u>a Relationship</u>.

Unable to modify table

Appears when a new <u>constraint</u> has failed on existing data. Compare the <u>ODBC</u> error text that appears in the <u>Save Incomplete dialog box</u> with the two ODBC errors shown below to determine the appropriate solution.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]The column [column name] in table Tmp_ [table name] may not be null.

Cause

A new database <u>column</u> has been added that does not allow null values and does not provide a <u>default</u> value. The table name in question appears after "Tmp_".

Recommended solution

Change the column properties. Either select the **Allow Nulls** property or type a **Default Value** setting. For more information about setting properties for database columns, see <u>Column Properties</u>.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Attempt to insert the value NULL into column '[column name]', table '[database name] TMP_ [table name]'; column does not allow nulls. INSERT fails.

[Microsoft][ODBC SQL Server Driver][SQL Server]Command has been aborted.

Cause

The **Allow Nulls** property on an existing database column has been cleared, but the column has existing null values in it.

Recommended solution

Go to the column and select the **Allow Nulls** property.

For more information about setting properties for database columns, see <u>Column</u> <u>Properties</u>.

Unable to preserve trigger

Appears when your <u>trigger</u> text references a <u>column</u> that has been renamed, deleted, or assigned a different data type.

ODBC error text

[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column name '[column name]'.

- Unable to preserve trigger '[trigger name]'.

Cause

A change to the table required the table to be recreated. When a table is recreated, the triggers attached to that table are automatically recreated as well.

Recommended solution

The recommended solution depends on the type of change made to the column referenced by the trigger.

To preserve a trigger that references a renamed column

• Rename the column to its original name and then save the table. This action will allow the table to be recreated. You can now rename the

column, save the table again, and then edit the trigger to fix the renamed columns.

To preserve a trigger that references a deleted column

- 1. Expand the table that the trigger is attached to.
- 2. Right-click the trigger you want to change and choose **Open** from the shortcut menu.
- 3. Edit the trigger text and save the trigger.
- 4. Save the table or database diagram.

For more information about triggers, see <u>Triggers</u> and <u>Enforcing Business Rules</u> <u>with Triggers</u>.

System errors

System errors can appear in the <u>Save Incomplete dialog box</u> when you exceed Microsoft® SQL Server[™] limitations that are not controlled by the Database Designer. One such error is described here.

ODBC error text

[Microsoft][ODBC SQL Server Driver]Timeout expired.

Cause

The timeout can occur when you are updating the database with any Transact-SQL changes.

Recommended solution

Try again later to save the diagram or selected tables.

-or-

Save a <u>change script</u> and apply it to the database at a later time. For more information, see <u>Saving a Change Script</u>.

-or-

Increase the SQL Query Time-out value and try to save the diagram or selected tables again.

See Also

Database Designer Dialog Boxes | Database Designer Considerations for SQL Server Databases

Unsaved Changes Exist Dialog Box

Appears if you close the last table of a database diagram or the diagram itself in the Database Designer when there are unsaved tables in memory.

Note Although you cannot undo changes made to a table or database diagram, the changes are not saved to the database until you save the table or diagram. You can discard any unsaved changes by choosing **No** and closing all open diagrams without saving them.

Yes

Saves all the tables shown in the list.

No

Closes the diagram without saving the changes.

Save Text File

Displays the Save As dialog box, letting you specify a location for a text file containing a record of the tables that were affected by the changes you made.

See Also

Database Designer Dialog Boxes | Saving Selected Tables

Validation Warnings Dialog Box

Appears if you attempt to save modifications with potentially damaging side effects—or if the database commit operation is likely to fail. This dialog box indicates what those side effects might be or why the commit operation might fail. It gives you the chance to proceed with the modification or cancel the operation.

Note This dialog box appears when you attempt to transmit your modifications to the database **OR** when you save a change script.

The dialog box can appear for any of these reasons:

- You might not have database permissions to commit all the modifications.
- Your modifications would result in improperly formed computed columns, default constraints, or check constraints.
- A modification to a column's data type might cause data loss.
- A modification would result in an index greater than 900 bytes.
- A modification would change a table or column contributing to a schema-bound view or user-defined function.

Yes

Proceeds with the operation. That is, selecting **Yes** proceeds to generate the change script or transmit the modifications to the database. Remember, even if you are transmitting the modifications to the database and you select **Yes**, the commit operation could still fail if you do not have privileges to modify the database; if your modifications would result in an index greater than 900 bytes; or if your modifications would result in an improperly formed computed column,

default constraint, or check constraint.

No

Cancels the save action.

Save Text File

Displays the Save As dialog box, letting you specify a location for a text file containing a list of the warnings.

See Also

Database Designer Dialog Boxes | Indexed Views | Saving a Change Script

Query Designer Dialog Boxes

Documentation is available for the following Database Designer user interface elements:

Add Table Dialog Box

Database Changes Detected Dialog Box

Define Query Parameters Dialog Box

Delete Table Dialog Box

Go To Row Dialog Box

Indexes Dialog Box

Choose Target Table for INSERT FROM Dialog Box

Insert Into Dialog Box

Make Table Dialog Box

Query Definitions Differ Dialog Box

Returning Query Results Dialog Box

Returning Query Results (Edit) Dialog Box

SQL Syntax Errors Encountered Dialog Box

Update Table Dialog Box

Add Table Dialog Box (Query and View Designers)

Enables you to add tables, views, or user-defined functions returning a table in the Query or View Designer.

Tables

Lists the tables you can add to the Diagram pane. To add a table, select it and click **Add**. To add several tables at once, select them and click **Add**.

Views

Lists the views you can add to the Diagram pane. To add a view, select it and click **Add**. To add several views at once, select them and click **Add**.

Functions

Lists the user-defined functions you can add to the Diagram pane. To add a function, select it and click **Add**. To add several functions at once, select them and click **Add**.

Add

Adds the selected item or items.

Close

Closes the dialog box without adding further items, including the currently highlighted items.

See Also

Query Designer Dialog Boxes | Add Table Dialog Box (Database Designer)

Database Changes Detected Dialog Box

Notifies you that the data row you are editing in the Results pane is no longer current. This dialog box appears when you attempt to save changes to a row by moving to another row in the Results pane and the Query Designer determines that the corresponding row in the database has changed. This situation occurs most often when another user has edited the row and saved the changes since you executed your query.

The dialog box displays a message asking how you want to proceed.

Yes

Choose this button to specify that you want to save your changes and overwrite the changes made by the other user.

No

Choose this button to discard the changes you have made and fetch the most current version of the row. The Query Designer returns you to the Results pane after rereading the row from the database.

Cancel

Choose this button to return to the Results pane without saving the row and without discarding your changes. If you choose this option, you can attempt to save the row again or make further changes, such as matching your edits to the version of the row in the database.

Tip To see the most current version of the row without discarding your changes, execute a second query against the data you are editing.

See Also

Query Designer Dialog Boxes

Define Query Parameters Dialog Box

Allows you to enter values for the parameters defined in the query. This dialog box appears when you execute a query that contains parameters to be filled in at run time.

Parameter name

Lists the parameters defined for the query being executed. If the query contains named parameters, the names appear in the list. If the query contains unnamed parameters, a question mark appears for each parameter in the query.

Parameter value

Enter the value for each parameter listed under Parameter name. The value used most recently appears as the default parameter value.

See Also

<u>Query Designer Dialog Boxes</u> | <u>Creating a Query with Named Parameters</u> | <u>Creating a Query with Unnamed Parameters</u> | <u>Parameter Queries</u> | <u>Specifying</u> <u>Parameter Marker Characters</u>

Delete Table Dialog Box

Allows you to specify the table from which to delete rows. This dialog box appears if more than one table is displayed in the Diagram pane when you start a Delete query.

Select the table to delete rows from, and then choose OK.

Note A Delete query removes entire rows from the table. If you want to clear values from individual data columns, use an Update query. If you want to delete columns from a table definition, or delete the table itself, use the Database Designer or the database design tools for your database.

See Also

Query Designer Dialog Boxes

Go To Row Dialog Box

Allows you to navigate to a specific row in the Results pane. This dialog box appears when you right-click in the Results pane and choose **Row**.

Row

Enter the row number of the row you want to navigate to. If the query has finished fetching query results, the total number of rows in the result set is displayed. If you enter a number higher than the total number of rows in the result set, the Query Designer moves to the last row.

See Also

<u>Query Designer Dialog Boxes</u> | <u>Navigating in the Query Designer</u> | <u>Results Pane</u>

Indexes Dialog Box

Allows you to specify create, delete, and modify indexes.

Selected Index

Shows the name of the first index for the selected view. To show properties for a different index, select an index from the drop-down list.

New

Choose this button to create a new index. For more information, see <u>Creating</u> <u>Indexed Views</u>.

Delete

Choose this button to remove the selected index.

Index Name

Shows the name of the selected item. You can rename the item by entering a new name in this box.

Column Name/Order

Shows the columns contributing to the index, along with whether each column's values are arranged in ascending or descending order within the item. You can add, change, or remove column names in this list. You can also change the ascending/descending setting for each column.

Index File Group

Select the name of the file group in which you want to store the selected index.

You must have at least one user-defined file group for this setting to be enabled. This setting is only available for Microsoft® SQL Server[™] 7.0 or higher databases. If you create a database object and do not specify its file group, SQL Server will assign it to the default file group. Initially, the default file group is the Primary file group.

For more information on creating and using file groups, see <u>Files and Filegroups</u>.

Create UNIQUE

Select this option to create a unique index for the selected view.

• **Ignore duplicate key** If you create a unique index, you can set this option to ensure each index value is unique.

Create as CLUSTERED

Select this option to create a clustered index for the selected view. For more information, see <u>Creating a Clustered Index</u>.

Do not automatically recomputed statistics

Select this option to tell SQL Server to use previously created statistics. This choice, available only for Microsoft® SQL Server[™] 7.0 and higher databases, may not produce optimal results and is not recommended.

For more information, see <u>Statistical Information</u>.

Fill factor

Shows the fill factor that specifies how full each index page can be. If a fill factor is not specified, the database's default fill factor is used. For more information, see <u>Specifying a Fill Factor for an Index</u>.

Pad Index

If you specified a fill factor of more than zero percent, and you selected the option to create a unique index, you can tell SQL Server to use the same percentage you specified in fill factor as the space to leave open on each interior node. By default, SQL Server sets a two-row index size.

See Also

Query Designer Dialog Boxes

Choose Target Table for INSERT FROM Dialog Box

Allows you to specify a table to insert new rows into. This dialog box appears when you start an INSERT FROM query.

Table name

Select from the list the name of the table to add rows to. You can specify only one table for the INSERT FROM query.

Note You can change the table into which you want to insert rows in the Properties window. For details, see <u>Query Tab, Properties Window</u>.

See Also

Query Designer Dialog Boxes

Insert Into Dialog Box

Allows you to specify the table to add a row to. This dialog box appears if more than one table is displayed in the Diagram pane when you start an Insert Into query.

Select the table to add a row to, and then choose **OK**.

See Also

Query Designer Dialog Boxes

Make Table Dialog Box

Allows you to name a table that will be created and that you will copy rows to. This dialog box appears when you start a Make Table query.

Table Name

Type the name of the table to create. The Query Designer does not check whether the name is already in use or whether you have permission to create the table.

To create a destination table in another database, specify a fully qualified table name, including the name of the target database, the owner (if required), and the name of the table.

Note Before you execute the query, from the Property Pages window, you can change properties of the table you want to create. For details, see <u>Query</u> <u>Tab, Properties Window</u>.

See Also

Query Designer Dialog Boxes | Creating Make Table Queries

Query Definitions Differ Dialog Box

Notifies you that your query cannot be represented graphically in the Diagram and Grid panes and that you can edit your query only in the SQL pane. This dialog box appears when you enter or edit an SQL statement in the SQL pane; then you either switch to another pane, verify the query, or attempt to execute the query; and one of the following conditions applies:

- The SQL command is incomplete or contains one or more syntax errors.
- The SQL command is valid but is not supported in the graphical panes (for example, a Union query).
- The SQL command is valid but contains syntax specific to the data connection you are using.

Tip You can check whether a statement is valid using the Verify SQL Statement button on the Query toolbar.

The dialog box displays a message with the reason that the SQL statement cannot be represented, and then asks how you want to proceed.

Note The Query Definitions Differ dialog box does not appear if you have hidden the Diagram and Grid panes, because the Query Designer assumes that you are editing only in the SQL pane.

Yes

Choose this button to specify that you want to accept the SQL statement, either to edit it further or to execute it. If you accept the statement, the Diagram and Grid panes appear dimmed to indicate that they do not represent the statement in the SQL pane.

No

Choose this button to discard your changes to the SQL pane.

Note If the statement is correct but not supported graphically by the Query Designer, you can execute it even though it cannot be represented in the Diagram and Grid panes. For example, if you enter a Union query, the statement can be executed but not represented in the other panes.

See Also

Query Designer Dialog Boxes

Returning Query Results (Edit) Dialog Box

Notifies you that the Query Designer or View Designer cannot save changes to the row you have edited because the result set is still being transmitted from the database to your computer. For more information, see <u>Interaction Between the Results Pane and the Database</u>.

To help you estimate how long you must wait, the dialog box displays a progress counter.

Cancel

Choose this button to specify that you want to cancel your attempt to move to another row. The Query Designer returns you to the row you have edited.

If you do not choose Cancel, the Query Designer saves your changes after the result set has been completely transmitted to your computer.

See Also

Query Designer Dialog Boxes

Returning Query Results Dialog Box

Notifies you that the Query Designer or View Designer cannot perform the requested action because the result set is still being transmitted from the database to your computer. This dialog box appears when you have executed a query, and then before the result set has been fully transmitted to your computer, you attempt one of the following operations:

- To go to a result-set row that has not yet been returned.
- To insert a new row in the Results pane.
- To delete one or more rows in the Results pane.
- To use the Copy command to place selected columns on the Clipboard.

For more information, see <u>Interaction Between the Results Pane and the</u> <u>Database</u>.

To help you estimate how long you must wait, the dialog box displays a progress counter.

Cancel

Cancels the attempted operation and leaves you at the row you were on when you requested the operation.

See Also

Query Designer Dialog Boxes

SQL Syntax Errors Encountered Dialog Box

Notifies you that the Query Designer cannot parse the SQL statement in the SQL pane. This dialog box appears when you enter or edit an SQL statement in the SQL pane; and then you either switch to another pane, verify the query, or attempt to execute the query; and one of the following conditions applies:

- The SQL command is incomplete or contains one or more syntax errors.
- The SQL command is valid but is not supported in the graphical panes (for example, a Union query).
- The SQL command is valid but contains syntax specific to the data connection you are using.

Tip You can check whether a statement is valid using the Verify SQL Statement button on the Query toolbar.

The dialog box displays a message with the reason that the SQL statement cannot be parsed. Choose **OK** to proceed.

See Also

Query Designer Dialog Boxes

Update Table Dialog Box

Allows you to specify the table to be updated. This dialog box appears if more than one table is displayed in the Diagram pane when you start an Update query.

Select the table to update, and then choose **OK**.

See Also

Query Designer Dialog Boxes | Creating Update Queries

Properties Pages

For information about the properties pages used by the Visual Database Tools, see the following topics:

Database Designer and Table Designer Properties Pages

<u>Query Designer and View Designer Properties Pages</u>

Database Designer and Table Designer Properties Pages

The Database Designer and Table Designer provide property pages with which you can create, delete, and modify database objects, and set other options that affect them.

То	See
Set properties for database tables	Tables Property Page
Set properties for columns	Columns Property Page
Create and manipulate relationships	Relationships Property Page
between tables, including choosing	
primary and foreign keys for the	
relationship	
Create and manipulate indexes,	Indexes/Keys Property Page
primary keys, and unique constraints	
attached to a table	
Create and manipulate check	Constraints Property Page
constraints	

Columns Property Page

This property page contains a set of properties for a column within a table.

Table name

Shows the name of the table containing the column whose properties you are viewing. The Table Name option is editable only within the Database Designer — not from within the Table Designer. If more than one table is selected in your diagram, only the name of the first table is visible.

Column Name

Shows the name of the selected column of the selected table in your diagram. If more than one table is selected in your diagram, only the name of the first column of the first table is visible. To show properties for a different column, expand the Column Name list.

Description

Shows the text description of the selected column.

Default Value

Shows the default for this column whenever a row with a null value for this column is inserted into the table. The value of this field can be either the value of a SQL Server default constraint or the name of a global constraint to which the column is bound. The drop-down list contains all global defaults defined in the database. To bind the column to a global default, select from the drop-down list. Alternatively, to create a default constraint for the column, enter the default value directly as text.

Precision

Shows the maximum number of digits for values of this column.

Scale

Shows the maximum number of digits that can appear to the right of the decimal point for values of this column.

Identity

Shows whether the column is used by SQL Server as an identifier column. Possible values are:

- **No** The column is not used as an identity column.
- **Yes** The column is used as an identity column.
- **Yes (Not For Replication)** The column is used as an identity column, except while a replication agent is inserting data into the table.

Identity Seed

Shows the seed value of an identity column. This option applies only to columns whose **Identity** option is set to **Yes** or **Yes (Not For Replication)**.

Identity Increment

Shows the increment value of an identity column. This option applies only to columns whose **Identity** option is set to **Yes** or **Yes (Not For Replication)**.

Is RowGuid

Shows whether the column is used by SQL Server as a ROWGUID column. You can set this value to **Yes** only for a column that is an identity column.

Formula

Shows the formula for a computed column.

Collation

Shows the collating sequence that SQL Server applies by default to the column whenever the column values are used to sort rows of a query result. To use the default collating sequence for the database, choose **<database default>**.

Indexed

Shows whether an index exists on the column. Possible values are:

- **No** No index exists on the column.
- **Yes (duplicates OK)** A non-unique index exists on the column.
- **Yes (no duplicates)** A unique index exists on the column.

See Also

Database Designer and Table Designer Properties Pages

Check Constraints Property Page

This property page contains a set of properties for constraints (except unique constraints) attached to the tables in your database. Properties applying to unique constraints appear on the Indexes/Keys property page.

Table Name

Shows the name of the selected table in your diagram. If more than one table is selected in your diagram, only the name of the first table is visible.

Selected Constraint

Shows the name of the constraint whose properties you are viewing. To view the properties of a different constraint, select a constraint from the drop-down list.

New

Choose this button to create a new constraint for the selected database table. Enter properties for the index. For more information, see <u>Attaching a New</u> <u>Check Constraint to a Table or Column</u>.

Delete

Choose this button to remove the selected constraint from the database. For more information, see <u>Deleting a Check Constraint</u>.

Constraint name

Shows the name of the constraint whose properties you are viewing. Use this control to modify the name of the constraint. For more information, see <u>Modifying a Check Constraint</u>.

Constraint expression

Shows the SQL syntax of the selected check constraint. For new constraints, you must enter the SQL syntax before exiting this box. You can also edit existing check constraints. For more information, see <u>Defining a Check Constraint</u> <u>Expression</u>.

Check existing data on creation

When selected, this option ensures that all data that exists in the table before the constraint was created is verified against the constraint.

Enforce constraint for INSERT and UPDATE

Enforces the constraint when data is inserted into or updated in the table.

Enforce constraint for replication

Enforces the constraint when the table is replicated into a different database.

See Also

Database Designer and Table Designer Properties Pages

Indexes/Keys Property Page

This property page contains a set of properties for the indexes, primary keys, and unique constraints attached to the tables in your database diagram. Indexes and constraints are not graphically represented in database diagrams.

Table name

Shows the name of the selected table in your diagram. If more than one table is selected in your diagram, only the name of the first table is visible.

Selected index

Shows the name of the first index for the selected table in your diagram. If more than one table is selected in your diagram, only the name of the first index for the first table is visible. To show properties for a different index, expand the drop-down list.

Туре

Shows the index or key object type for the selected table: index, primary key, or unique.

New

Choose this button to create a new index, key, or unique constraint for the selected database table. For more information, see <u>Creating an Index</u>.

Delete

Choose this button to remove the selected index, key, or constraint from the table. For more information, see <u>Deleting an Index</u>.

Note If you try to delete a primary key that participates in relationships, a

message appears asking you if you want to delete all the relationships, too. You cannot delete a primary key without first deleting the relationships it participates in.

Column name/Order

Shows the columns contributing to the index, primary key, or unique constraint, along with whether each column's values are arranged in ascending or descending order within the item. You can add, change, or remove column names in this list. You can also change the ascending/descending setting for each column.

Index name

Shows the name of the selected index. You can rename the index by entering a new name in this box. For more information, see <u>Renaming an Index</u>.

Index file group

Select the name of the file group in which you want to store the selected index. You must have at least one user-defined file group for this setting to be enabled. If you create a database object and do not specify its file group, SQL Server will assign it to the default file group. Initially, the default file group is the Primary file group.

For more information on creating and using file groups, see <u>Placing Indexes on</u> <u>Filegroups</u>.

Create UNIQUE

Select this option to create a unique constraint or index for the selected database table. Specify whether you are creating a constraint or index by selecting either the **Constraint** or **Index** button.

• **Ignore duplicate key** If you create a unique index, you can set this option to ensure each value in an indexed column is unique.

Fill factor

Shows the fill factor that specifies how full each index page can be. If a fill factor is not specified, the database's default fill factor is used. For more information, see <u>Specifying a Fill Factor for an Index</u>.

Pad Index

If you specified a Fill Factor of more than zero percent, and you selected the option to create a unique index, you can tell SQL Server to use the same percentage you specified in Fill Factor as the space to leave open on each interior node. By default, SQL Server sets a two-row index size.

Create as CLUSTERED

Select this option to create a clustered index for the selected database table. For more information, see <u>Creating a Clustered Index</u>.

Don't automatically recompute statistics

Select this option to tell SQL Server to use previously created statistics. This choice may not produce optimal results and is not recommended.

For more information, see <u>Statistical Information</u>.

See Also

Database Designer and Table Designer Properties Pages

Relationships Property Page

This property page contains a set of properties for the relationships between the tables in your database.

Table name

Shows the name of the selected table in your diagram. If more than one table is selected in your diagram, only the name of the first table is visible.

Selected relationship

Shows the name of the selected relationship in your diagram. If more than one relationship is selected in your diagram, only the name of the first relationship in your selection is visible. Expand the list to view or modify the properties of a different relationship.

Each entry in the drop-down list is preceded by an icon. A key icon indicates that the table participates in the relationship as the referred-to table. An infinity icon indicates that the table participates as the referring table. (The referring table has the relationship's foreign-key constraint.)

New

Choose this button to create a new relationship for the selected database table. For more information, see <u>Creating a Relationship</u>.

Delete

Choose this button to remove the selected relationship from the database. For more information, see <u>Deleting a Relationship</u>.

Relationship name

Shows the name of the selected relationship. You can rename the relationship by entering a new name in this box. For more information, see <u>Renaming a</u> <u>Relationship</u>.

Primary key table

Shows the name of the primary key table in the relationship, followed by the columns that make up the primary key. For information about changing the primary key, see <u>Modifying a Primary Key</u>.

Foreign key table

Shows the name of the foreign key table in the relationship, followed by the columns that make up the foreign key. For information about changing the foreign key, see <u>Modifying a Foreign Key</u>.

Check existing data on creation

Applies the constraint to data that already exists in the database when the relationship is added to the foreign key table.

Enforce relationship for INSERTs and UPDATEs

Applies the constraint to data that is inserted into, deleted, or updated in the foreign key table. Also prevents a row in the primary key table from being deleted when a matching row exists in the foreign key table.

Enforce relationship for replication

Applies the constraint when the foreign key table is copied to a different database.

Cascade Update Related Fields

Instructs the DBMS to automatically update foreign-key values of this

relationship whenever the primary-key value is updated.

Cascade Delete Related Fields

Instructs the DBMS to automatically delete rows of the foreign-key table whenever the referred-to row of the primary-key table is deleted.

See Also

Database Designer and Table Designer Properties Pages

Tables Property Page

This property page contains a set of properties for the tables in your database.

Selected table

Shows the name of the selected table. If more than one table is selected in your diagram, only the name of the first table in your selection is visible. Expand the list to choose a different table whose properties you want to inspect or modify.

Owner

Shows the name of the table's owner. The owner name is either a SQL Server role or SQL Server user. The drop-down list contains all the users and roles defined in the database. Within the drop-down list, the users and roles have different icons; the role icon shows two faces, the user icon shows only one.

This control is editable only if you are connected to a database as a user that is a member of the db_owner role or is a member of both the db_ddladmin and db_securityadmin roles.

Table name

Shows the name of the selected table. To rename the table, enter a new name in this box. For more information, see <u>Renaming a Table</u>.

Table Identity Column

Shows the column used by SQL Server as the table's identity column. To change the identity column, choose from the drop-down list. Within the drop-down list, you can choose the blank entry to indicate that the table has no identity column.

Table ROWGUID Column

Shows the column used by SQL Server as the table's ROWGUID column. To change the ROWGUID column, choose from the drop-down list. Within the drop-down list, you can choose the blank entry to indicate that the table has no ROWGUID column.

Table File Group

Select the name of the file group in which you want to store the selected table data. You must have at least one user-defined file group for this setting to be enabled. If you create a database object and do not specify its file group, SQL Server will assign it to the default file group. Initially, the default file group is the Primary file group.

For more information, see <u>Placing Tables on Filegroups</u>.

Text File Group

Select the name of the file group you want to store the text and images from the selected table in. You must have at least one user-defined file group for this setting to be enabled. If you create a database object and do not specify its file group, SQL Server will assign it to the default file group. Initially, the default file group is Primary file group.

For more information, see <u>Placing Tables on Filegroups</u>.

Description

You can enter any text in this field. The text that you enter is implemented as a SQL Server 2000 extended property.

See Also

Database Designer and Table Designer Properties Pages

Query Designer Properties Pages

Query Designer provides property pages to set options that affect how queries will be executed against your database.

То	See
Control the behavior of your query	Query Tab, Properties Window
by setting its properties using the	(Query Designer)
Query Designer	
Control the behavior of your query	Query Tab, Properties Window
by setting its properties using the	<u>(View Designer)</u>
View Designer	
Mark parameters in your query with	Parameters Tab, Properties Window
special characters	
Specify options for joining tables in a	Join Line Tab, Properties Window
multitable query	

Query Tab, Properties Window (Query Designer)

Contains options for controlling the behavior of the query you are building or modifying.

Query name

Displays the name for the current query. You cannot change the query name in this box. For more information, see <u>Visual Database Tools Usage</u> <u>Considerations</u>.

Output all columns

Specifies that all columns from all tables in the current query will be in the result set. Choosing this option is equivalent to specifying an asterisk (*) in place of individual column names after the SELECT keyword in the SQL statement.

DISTINCT values

Specifies that the query will filter out duplicates in the result set. This option is useful when you are using only some of the columns from the table or tables and those columns might contain duplicate values, or when the process of joining two or more tables produces duplicate rows in the result set. Choosing this option is equivalent to inserting the word DISTINCT into the statement in the SQL pane.

Destination table

Specifies the name of the table into which you are inserting rows. This list appears if you are creating an Insert query or Make Table query. For an Insert query, select a table name from the list.

For a Make Table query, type the name of the table to create. The Query Designer does not check whether the name is already in use or whether you have permission to create the table. To create a destination table in another database, specify a fully qualified table name, including the name of the target database, the owner (if required), and the name of the table.

GROUP BY extension

Specifies that additional options for views based on <u>aggregate queries</u> are available.

WITH CUBE

Specifies that the aggregate query should produce summary values for groups specified in the GROUP BY clause. The groups are created by cross-referencing columns included in the GROUP BY clause, and then applying the query's aggregate function to produce summary values for the additional super-aggregate rows. The WITH CUBE option is multidimensional, creating summaries from all combinations of aggregate functions and columns in the query.

WITH ROLLUP

Specifies that the aggregate query should produce summary values for the groups specified in the GROUP BY clause. The WITH ROLLUP option is similar to the WITH CUBE option, but creates only one type of summary for each combination of column and aggregate function.

ALL

Specifies that the aggregate query will include any duplicate rows in the summary.

Тор

Specifies that the query will include a TOP clause, which returns only the first *n* rows or first *n* percentage of rows in the result set. The default is that the query returns the first 10 rows in the result set.

Use this box to specify a different number of rows to return or to specify a

percentage.

Percent

Specifies that the query will include a TOP clause, returning only the first *n* percentage of rows in the result set.

With Ties

Specifies that the query will include a WITH TIES clause. WITH TIES is useful if a query includes an ORDER BY clause and a TOP clause based on percentage. If this option is set, and if the percentage cutoff falls in the middle of a set of rows with identical values in the ORDER BY clause, the query extends the percentage until all such rows are included.

See Also

Query Designer Properties Pages

Query Tab, Properties Window (View Designer)

Contains options for controlling the behavior of the view you are building or modifying.

View name

Displays the name for the current query. If you have not yet saved your view, the default name is displayed. You cannot change the name in this box. To assign a name, choose **Save** or **Save As** from the **File** menu.

Output all columns

Specifies that all columns from all tables currently displayed will be in the view. Choosing this option is equivalent to specifying an asterisk (*) in place of individual column names.

DISTINCT values

Specifies that the query will filter out duplicates in the view. This option is useful when you are using only some of the columns from a table and those columns might contain duplicate values, or when the process of joining two or more tables produces duplicate rows in the result set. Choosing this option is equivalent to inserting the word DISTINCT into the statement in the SQL pane.

Encrypt view

Encrypts the view when you save it. You will continue to be able to work with the view until you close the View Designer.

Important After you close an encrypted view, you will no longer be able to open it in the View Designer. If you need to modify an encrypted view, you must delete it and recreate another.

GROUP BY extension

Specifies that additional options for views based on <u>aggregate queries</u> are available.

WITH CUBE

Specifies that the aggregate query should produce summary values for groups specified in the GROUP BY clause. The groups are created by cross-referencing columns included in the GROUP BY clause, and then applying the query's aggregate function to produce summary values for the additional super-aggregate rows. The WITH CUBE option is multidimensional, creating summaries from all combinations of aggregate functions and columns in the query.

WITH ROLLUP

Specifies that the aggregate query should produce summary values for the groups specified in the GROUP BY clause. The WITH ROLLUP option is similar to the WITH CUBE option, but creates only one type of summary for each combination of column and aggregate function.

ALL

Specifies that the aggregate query will include any duplicate rows in the summary.

Check Option

Indicates that whenever you open this view and modify the results pane, the DBMS checks that the added or modified data satisfies the WHERE clause of the view definition.

Owner

Shows the name of the view's owner. The owner name is either a SQL Server

role or SQL Server user. The drop-down list contains all the users and roles defined in the database. Within the drop-down list, the users and roles have different icons; the role icon shows two faces, the user icon shows only one.

This control is editable only if you are connected to the database as a user that is a member of the db_owner role or is a member of both the db_ddladmin and db_securityadmin roles.

Тор

Specifies that the query will include a TOP clause, which returns only the first *n* rows or first *n* percentage of rows in the result set. The default is that the query returns the first 10 rows in the result set.

Use this box to specify a different number of rows to return or to specify a percentage.

Percent

Specifies that the query will include a TOP clause, returning only the first *n* percentage of rows in the result set.

With Ties

Specifies that the view will include a WITH TIES clause. WITH TIES is useful if a view includes an ORDER BY clause and a TOP clause based on percentage. If this option is set, and if the percentage cutoff falls in the middle of a set of rows with identical values in the ORDER BY clause, the view is extended to include all such rows.

See Also

Query Designer Properties Pages

Data Source Tab, Properties Window

Specifies options for any table or table-structured object appearing in the Diagram pane of a query or view.

Object Name

If the selected object is a table, view, or function, this control contains the object's name.

If the selected object is a subquery, this control contains the text of the subquery.

Alias

Contains the Alias (if any) of the selected object. You can create or modify an alias for the object by typing in this box.

Full Name

If the selected object is a table, view, or function, this control contains the object's full name (e.g., Northwind.dbo.Invoices).

If the selected object is a subquery, this control contains the text of the subquery.

See Also

<u>Query Designer Properties Pages | Using Something Else in Place of a Table</u>

Function Parameters Tab, Properties Window

Specifies options for parameters of a table-valued user-defined function. This property page contains a grid, each row of which describes one parameter.

Options

Name

Contains the name of the parameter. You cannot edit this box.

Data Type

Contains the data type of the parameter. You cannot edit this box.

Default

Contains the default value of the parameter. You cannot edit this box.

Value

Contains the value for this parameter used by this query.

See Also

Query Designer Properties Pages

Parameters Tab, Properties Window

Specifies options for marking parameters in the query.

To include a parameter in a query, you must mark the parameter using special characters so that the Query Designer does not mistake your parameter name for text data. For example, if you specify square brackets ([and]) as parameter markers, you can enter a parameter by specifying a search expression such as the following:

= [last_name]

When you run the query, the Query Designer prompts you for a value, and then substitutes the value you provide for the parameter last_name.

For more information about using parameters in queries, see <u>Parameter Queries</u>.

Options

Prefix characters

Specifies the character or characters that mark the beginning of a parameter.

Suffix characters

Specifies the character or characters that mark the end of a parameter.

Escape character

Specifies an escape character that is used to indicate that a parameter marker character in a name is meant literally. For example, if you specify "%" as a parameter marker, you would not normally be able to include that character in a parameter name such as "% complete." However, you can specify "\" as an escape character, and then use the following string as the parameter name:

%\% complete%

See Also

Query Designer Properties Pages

Join Line Tab, Properties Window

Specifies options for the joining of tables in a query.

By default, related tables are joined using an inner join that creates a result set based on rows containing matching information in the join columns. By setting options in the Join Line tab, you can specify a join based on a different operator, and you can specify an outer join.

For more information about joining tables, see <u>Querying Using Multiple Tables</u>.

Options

Table

The names of the tables involved in the join. You cannot change the names of the tables here — this information is displayed for information only.

Column

The names of the columns used for joining the tables. The operator in the Operator list specifies the relationship between the data in the columns. You cannot change the names of the columns here — this information is displayed for information only.

Operator

Specifies the operator used to relate the join columns. To specify an operator other than equal (=), select it from the list. When you close the dialog box, the operator you selected will appear in the diamond graphic of the join line, as in the following:

Ś

Include rows

Specifies whether unmatched rows appear in the **Results** pane.

All rows from <table1>

Specifies that all the rows from the left table appear in the output, even if there are no corresponding matches in the right table. Columns with no matching data in the right table appear as null. Choosing this option is equivalent to specifying LEFT OUTER JOIN in the SQL statement.

All rows from <table2>

Specifies that all the rows from the right table appear in the output, even if there are no corresponding matches in the left table. Columns with no matching data in the left table appear as null. Choosing this option is equivalent to specifying RIGHT OUTER JOIN in the SQL statement.

Selecting both All rows from <table1> and All rows from <table2> is equivalent to specifying FULL OUTER JOIN in the SQL statement.

When you select an option to create an outer join, the diamond graphic in the join line changes to indicate that the join is a left outer, right outer, or full outer join.

Note The words "left" and "right" do not necessarily correspond to the position of tables in the Diagram pane. "Left" refers to the table whose name appears to the left of the keyword JOIN in the SQL statement, and "right" refers to the table whose name appears to the right of the JOIN keyword. If you move tables in the Diagram pane, you do not change which table is considered left or right.

See Also

Query Designer Properties Pages

Error Messages

You may encounter message boxes while using the Microsoft® Visual Database Tools. For more information about those messages, browse the following topics.

Database Designer Troubleshooting Errors

Query Designer Troubleshooting Errors

Note The Visual Database Tools are dependent upon many other applications and components. If you receive an error message not listed in the Visual Database Tools documentation, see <u>Error Messages</u> in the Troubleshooting section.

Database Designer Troubleshooting Errors

Error messages may occur during the process of saving a database diagram or selected tables in a diagram. For more information about those messages, see the <u>Save Incomplete dialog box</u>.

For information about additional Visual Database Tools error messages, see <u>Query Designer Troubleshooting Errors</u>.

Note The Visual Database Tools are dependent upon many other applications and components. If you receive an error message not listed in the Visual Database Tools documentation, see <u>Error Messages</u> in the Troubleshooting section.

A pair of matching columns is required to create a relationship.

You must select matching columns from both tables participating in a foreign key relationship. The columns you have selected don't match, so they cannot be related.

Choose two columns that match, and then attempt to establish the relationship. In order to match, there must be the same number of columns and each partner in a pair of columns must have a comparable data type.

For more information, see <u>Foreign Key</u> and <u>Creating a Relationship Between</u> <u>Tables</u>.

A primary key can't be created on column '<0s>' because it allows null values.

The primary key is used to relate the table to foreign keys in other tables. All columns participating in a primary key must contain defined values other than NULL.

To create a primary key on the selected column, first clear the Allow Nulls check box for the column.

For more information, see <u>Primary Key</u> and <u>Column Properties</u>.

A primary key or index cannot be created on columns with a datatype of <0s>.

A column's datatype defines what kind of information can be stored in a column. Some datatypes such as TEXT or Image cannot be used in an index because servers will not allow indexes on these types of data.

For more information, see <u>Indexes</u>, <u>Creating a Unique Index</u>, and <u>Defining a</u> <u>Primary Key</u>.

A primary key or index cannot have more than <0d> columns.

Microsoft® SQL Server[™] does not allow more than 16 columns in an index. Because a primary key has an associated index, there can be no more than 16 columns in a primary key.

For more information, see <u>Indexes</u> and <u>Defining a Primary Key</u>.

A primary key or unique constraint must be defined for table '<0s>' before it can participate in a relationship.

In order to create a foreign key relationship, at least one of the two tables must have a primary key or unique constraint defined. To create the relationship you are attempting to create, first create a primary key for one of the tables.

For more information, see <u>Defining a Primary Key</u> and <u>Creating a Relationship</u> <u>Between Tables</u>.

A relationship cannot contain more than '<0d>' columns.

Select fewer than 16 columns when defining the foreign key relationship.

For more information, see <u>Creating a Relationship Between Tables</u>.

An index already exists for table '<0s>' with the columns '<1s>'.

Oracle only allows one index for each ordered set of columns.

As a rule, you should create an index only if the data in the selected columns will be queried frequently. Indexes take up disk space and can slow the adding, deleting, and updating of rows.

In most cases, the benefits of indexes will far outweigh the performance overhead, but if your application updates data very frequently, or if you have disk space constraints, you might also want to limit the number of indexes.

For more information, see <u>Indexes</u> and <u>Creating an Index</u>.

An index can't exist on a blank column.

An index must contain at least one column, and that column must contain unique data values if you are creating a unique index. The column names selected must be entered under Column name in the Selected index box.

For more information, see <u>Indexes</u> and <u>Creating an Index</u>.

Are you sure you want to delete the current selection from your database?

This will permanently delete the table and all its data from your database.

For more information, see <u>Deleting a Table from a Database Diagram and the</u> <u>Database</u>.

Are you sure you want to permanently delete table '<0s>' from your database?

This will permanently delete the table and all its data from the database.

For more information, see <u>Deleting a Table from a Database Diagram and the</u> <u>Database</u>.

Are you sure you want to permanently delete the selected tables from your database?

The table and all its data will be deleted from the database.

For more information, see <u>Deleting a Table from a Database Diagram and the</u> <u>Database</u>.

Are you sure you want to remove the selected table from the diagram?

This will remove the table from the diagram but leave the table in the database.

For more information, see <u>Removing a Table from a Database Diagram</u>.

Are you sure you want to remove the selected tables from the diagram?

This will remove the table from the diagram, but not from your database.

For more information, see <u>Removing a Table from a Database Diagram</u>.

Both sides of a relationship must have the same number of columns.

When creating a composite foreign key relationship, both sides of the relationship must have the same number of columns. Make sure the number of columns you've selected matches the number of columns in the target table.

For more information, see <u>Creating a Relationship Between Tables</u>.

Column '<1s>' in table '<0s>' participates in index '<2s>'. Columns with a datatype of '<3s>' cannot participate in indexes or primary keys.

You have attempted to change the datatype of a column that is part of an index or a primary key. Indexes and primary keys cannot be created for columns with datatypes such as TEXT or IMAGE. To change the datatype, first remove the index or primary key.

For more information see <u>Indexes</u>, <u>Creating a Unique Index</u>, and <u>Defining a</u> <u>Primary Key</u>.

Editing this default will permanently unbind the named default.

Changing from a named default to a default constraint is not allowed on Microsoft® SQL Server[™]. For more information, see <u>Constraints</u>.

Identity column '<0s>' in table '<1s>' must have a datatype of int, smallint, tinyint, decimal or numeric with scale of 0.

Only certain datatypes can be used for Identity columns. Columns that are Identity columns contain system-generated, sequential values that uniquely identify each row within a table.

For more information, see <u>Column Properties</u>.

Invalid name. You must provide a name for this object.

An object name is required to continue. Valid names begin with an underscore (_) or a character and contain a combination of characters, numbers, and underscores.

The maximum length for a name based on the server is 32 for Microsoft® SQL Server[™] 6.5 and earlier and 32 for Oracle.

In general, avoid names that contain periods.

For more information, see <u>Uniquely Naming Database Objects</u>.

ODBC error: <0s>.

An ODBC error has been generated. You might have deleted a record that has a foreign key value related to it, or you might have violated a check constraint.

For details, refer to your ODBC documentation.

One or more selected tables are already on the diagram.

You can only have one copy of any table on a diagram. If you want to create a new table that has some of the same columns as an existing table, you can duplicate an existing table as the first step in creating a new table.

For more information, see <u>Duplicating Tables</u>.

Only one clustered index can be created on table '<0s>'.

A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore only one clustered index can be created on each table.

To create a second index on the same table, change the Clustered property setting before creating the second index.

For more information, see <u>Indexes</u> and <u>Creating a Clustered Index</u>.

Only one ROWGUID column is allowed per table.

You have attempted to assign more than one ROWGUID. The ROWGUID is a special property similar to IDENTITY and only one ROWGUID is allowed per table.

For more information, see <u>Setting Column Properties</u>.

Relationship '<0s>' was modified or deleted since the diagram was loaded.

A modified relationship is about to be saved. It is possible that a table involved in the relationship has been deleted from the database by another user.

Verify whether this relationship is meant to be modified. Changing or deleting a relationship for a table may affect other tables.

For more information about relationships, see <u>Table Relationships</u> and <u>Deleting a</u> <u>Relationship</u>.

Table <0s> already exists.

The table name you specified already exists in the database. Type another name for the new table.

Table '<0s>' is marked for deletion and was not added to the diagram.

A table that has been removed from the diagram will be deleted from the diagram when the changes are saved. The table will not, however, be deleted from the database.

For more information, see <u>Removing a Table from a Database Diagram</u>.

-or-

The table has been deleted from the database by another user and cannot be added to your diagram.

For more information, see <u>Deleting a Table from a Database Diagram and the</u> <u>Database</u>.

Table '<0s>' no longer exists in the database.

The table has been deleted from the database and cannot be used in your diagram.

For more information, see <u>Deleting a Table from a Database Diagram and the</u> <u>Database</u>.

The Allow Nulls property can't be set on a column that is part of the primary key.

All columns that are part of a table's a primary key must contain aggregate unique values other than NULL. To add the column to a primary key, first clear the Allow Nulls check box on the Column Property dialog box.

For more information, see <u>Column Properties</u>.

The Allow Nulls property can't be set on column '<0s>' because it is an identity column.

All identity columns must contain unique values other than NULL. To make the column an identity column, first clear the Allow Nulls check box on the Column Property dialog box.

For more information, see <u>Column Properties</u>.

The columns in table '<0s>' do not match an existing enabled primary key or UNIQUE constraint.

At least one table participating in a relationship must have either a <u>Primary Key</u> or a <u>Unique Constraint</u>. After setting up a Primary Key or a Unique constraint for one of the tables you've selected, you can then define other relationships for that table.

For more information, see <u>Creating a Unique Constraint</u> and <u>Defining a Primary</u> <u>Key</u>.

The columns in table '<0s>' do not match an existing primary key or UNIQUE constraint.

The columns on the primary key side of a foreign key relationship must participate in either a <u>Primary Key</u> or a <u>Unique Constraint</u>. After setting up a Primary Key or a Unique constraint for one of the tables you've selected, you can then define other relationships for that table.

For more information, see <u>Creating a Unique Constraint</u> and <u>Defining a Primary</u> <u>Key</u>.

The datatype of column '<1s>' in the '<0s>' table can't be changed because it participates in index '<2s>'.

The sum of the lengths of all columns in an index can not exceed 256 bytes. Changing to the specified datatype would exceed the maximum index size of <3d> bytes.

For more information, see <u>Column Properties</u> and <u>Creating an Index</u>.

The following datatype or size property of '<0s>.<1s>' doesn't match '<2s>.<3s>'.

When creating a foreign key relationship, the data types and other properties of selected columns must match. Select a column with the same data types and other properties as the related column. The properties that must match are Data type, Length, Precision, Scale, and Collation.

The Default Value property can't be set on column '<0s>' because it is an identity column.

Identity columns contain system-generated sequential values that uniquely identify each row within the table. Therefore, identity columns cannot have default values.

For more information, see <u>Column Properties</u>.

The existing relationship must have at least one pair of related columns.

There must be matching columns on both sides of a foreign key relationship. Deleting a column that is part of a relationship will disable the relationship.

For more information, see <u>Creating a Relationship</u>.

The Identity property can't be set on column '<0s>' because it allows null values.

Identity columns contain system-generated sequential values that uniquely identify each row within the table. Therefore, identity columns cannot have null values.

For more information, see <u>Column Properties</u>.

The Identity property can't be set on column '<0s>' because it has a default value.

Identity columns contain system-generated sequential values that uniquely identify each row within the table. Therefore, identity columns cannot have default values.

For more information, see <u>Column Properties</u>.

The number of selected columns exceeds the number of columns in the target table.

You cannot create a compound foreign key relationship using more columns than exist in the target table. Make sure the number of columns you've selected matches the number in the target table.

For more information, see <u>Creating a Relationship Between Tables</u>.

The primary key or UNIQUE constraint cannot be changed until its existing relationships are deleted.

You cannot change a table's primary key or unique constraint while other foreign key relationships are still in place. To change a table's primary key or unique constraint, first delete all existing relationships for the table.

For more information, see <u>Deleting a Relationship</u>.

The primary key or Unique constraint cannot be changed while relationships to the existing primary key or Unique constraint are enforced.

You cannot change a table's primary key or unique constraint while other foreign key relationships are still in place. To change a table's primary key or unique constraint, first delete all existing relationships for the table.

For more information, see <u>Deleting a Relationship</u>.

The total size of an index or primary key cannot exceed 256 bytes.

The sum of the lengths of all columns in an index can not exceed 256 bytes. Changing to the specified datatype would exceed the maximum index size of <3d> bytes.

For more information, see <u>Indexes</u>, <u>Column Properties</u>, and <u>Creating an Index</u>.

You are not logged in as the database owner or as a user that is a member of the db_owner role. You will not be able to save changes to tables that you do not own.

Because you are not logged on as the system administrator, database owner, or a user that is a member of the db_owner role, you have limited privileges to the database. The privileges you have are determined by the permissions granted to your logon ID and the privileges granted to the roles that your logon ID is a member of.

Even though you are not the database owner, you will still be able to use any tables that you have permissions to see. For example, you can create diagrams using such tables. However you won't be able to perform all edits. Certain edits require SQL Server CREATE TABLE permission, which gives you permission to create new tables and modify tables that you own.

Even if you have CREATE TABLE permission, there are limitations to the modifications you can make. Remember, as you modify an existing table or design a new one, your work can induce attendant modifications in other tables. For example, if you change the data type of a foreign-key column, the corresponding column in the primary-key table will be automatically modified by the Visual Database Tools. If you do not own the primary-key table, and you are not logged in as the system administrator, database owner, or a user that is a member of the db_owner role, your modification will fail.

For more information, see <u>Working in a Multi-User Environment</u>.

A blank constraint expression is not allowed.

You must enter a constraint in the **Constraints Expression** dialog box before fcontinuing to define properties for your table.

For more information, see <u>Creating a Unique Constraint</u>.

Are you sure you want to delete the selected relationship from your database?

Answering YES will delete the selected relationship.

For more information about relationships, see <u>Table Relationships</u> and <u>Deleting a</u> <u>Relationship</u>.

Deleting the selected columns will also delete relationships.

The columns you have chosen to delete have existing relationships. Deleting these columns will delete the relationships associated with them.

For more information about relationships, see <u>Table Relationships</u> and <u>Deleting a</u> <u>Relationship</u>.

Do you want to save changes in this diagram?

You have made changes in this diagram. You will lose the changes if you close this diagram without saving. Do you want to save changes?

For more information about database diagrams, see <u>Database Designer</u>.

Do you want to save changes in this table?

You have made changes in this table. You will lose the changes if you close this table without saving. Do you want to save changes?

For more information about working with tables, see <u>Tables</u>.

Database Designer Error

No specific help is available for this error. If the text in the error message does not provide enough information for you to solve the problem, please consider the following options:

- Verify that your settings are correct and try the action again.
- Use the index to search for keywords related to the error message.
- If the problem is due to external applications or software, consult the documentation for that area. For example, errors may be caused by your database software, your server setup, or even the network you are using.
- For database modification errors, such as invalid parameters in a database table, consult your database software for details.
- Check the Readme for late-breaking information concerning the error.

Internal Error.

An unexpected error has occurred. You can try any of the following remedial actions:

- Try the operation again.
- Restart your application and try again.
- Restart your computer and try again.
- Check the Knowledge Base at the <u>Microsoft Web site</u> for any available new information on internal errors with the Visual Database Tools.

The name is too long.

The name supplied for a database object is too long. Enter a shorter name.

This backend version is not supported to design database diagrams or tables.

The database to which you are connected does not allow schema modification with the Visual Database Tools. Be sure you are connected to the correct database.

Changing the column to the selected data type will delete relationships.

The columns whose data types you are changing participate in foreign-key relationships. If the data types are changed, the relationships will be deleted. Choose Yes to proceed with the operation and delete the relationships. Choose No to stop the operation.

Note: Choosing No does not cancel the portion of the operation that has already been completed in the database.

Adding a formula to the selected column will delete all indexes and relationships it participates in.

The column you are modifying participates in one or more relationships or contributes to one or more indexes. When you modify the column, it will become a computed column, which cannot participate in relationships or contribute to indexes. Choose Yes to proceed with the operation and delete the indexes and relationships. Choose No to cancel the operation.

Removing the formula from the selected column leaves it with data type that is disallowed in indexes and relationships.

The column whose formula you are removing participates in one or more indexes or relationships. The modification will result in a column whose data type precludes it from participating in them. Choose Yes to proceed with the modification and delete the indexes and relationships. Choose No to cancel the modification.

Changing the column to the selected data type will delete its indexes.

The column whose data type you are changing contributes to one or more indexes. The modification will delete the indexes. Choose Yes to proceed with the modification and delete the indexes. Choose No to cancel the modification.

Changing a column data type results in an index that is too large.

The column whose data type you are changing contributes to one or more indexes. The new data type enlarges the column, making one or more of that column's indexes too large. Choose Yes to proceed with the modification and delete the index or indexes. Choose No to cancel the modification.

Adding a formula to the selected column will delete all relationships it participates in as a foreign key.

The column to which you are adding a formula contributes to one or more foreign keys. The modification will delete the attendant foreign-key relationships. Choose Yes to proceed with the modification and delete the relationships. Choose No to cancel the modification.

Error validating the formula for column.

The formula contains an error. Common errors include mismatched parentheses, misspelled column names, and using an operator on a column whose data type does not support that operator. Choose Yes to modify the formula. Choose No to leave it as is.

Column has no name and will be deleted.

Because the column-name field is blank, the column will be deleted. Choose Yes to delete the column. Choose No to return to the Table Designer or database diagram to supply a column name.

Primary key or index cannot be created on column '<0s>' because its data type is '<1s>'.

You have included in an index or primary key a column whose data type precludes its participation in keys or indexes. Change the column's data type or remove it from the index or primary key.

For the rules restricting the data types of index and key columns, see <u>Creating</u> <u>and Modifying PRIMARY KEY Constraints</u> and <u>Table Indexes</u>.

Primary key or index cannot be created on a column with no name.

You have included in a primary key a column with no name. Provide a name for the column before including that column in the primary key.

You entered <0d> characters for '<1s>'. The maximum number of characters allowed is <2d>.

You entered a property value that is longer than the maximum allowed for that property. Choose a shorter value.

Values for '<0s>' must lie within the range <1d> to <2d>.

You entered a property value that is outside the range of legal values for that property. Choose a value within the range specified.

Table '<0s>' has no columns.

You tried to save a table with no columns. Either add a column to the table, delete the table from the diagram, or close the Table Designer without saving the table.

Default Value property cannot be set on column '<0s>' because it is a computed column.

You tried to set a default value on a column that has a formula. Remove the column's formula before setting a default.

Column '<0s>' is a computed column, and cannot participate in indexes or relationships.

You tried to use a column with a formula in an index or relationship. In SQL Server 7.0 databases, a computed column cannot participate in an index or a relationship. In SQL Server 2000, a computed column can participate in indexes, but not in relationships. Either clear the column's **Formula** property or remove the column from the index or relationship.

Column '<0s>' is a computed column, and cannot contribute to foreign keys.

You tried to use a column with a formula in a foreign key. Either clear the column's **Formula** property or remove the column from the foreign key.

The index is used to enforce the full-text key for this table. Deleting this index will disable full-text indexing for the table. Do you want to proceed?

You are performing an operation that will delete an index that is used as a key for full-text indexing of this table. If you proceed with the operation, full-text indexing for this table will be disabled.

Column '<0s>' is used to enforce the full-text key on table '<1s>' and must be <2d> bytes or less.

You are performing an operation that will enlarge a column used as a key for full-text indexing of this table beyond the maximum size for such a column. Choose Yes to proceed with the operation and disable full-text indexing on the table. Choose No to cancel the operation.

Column '<0s>' participates in full-text indexing on table '<1s>'. After this change, the column will no longer participate in the table's full-text index.

You are performing an operation that disallows full-text searching of the data in this column. Choose Yes to proceed with the operation and disable full-text searches of this column.

Even if you choose Yes, other columns that participate in this table's full-text indexing will be unaffected. That is, you will still be able to perform full-text searches of the data in those columns.

The following error was encountered while changing the collation: <0s>.

The collation was not changed because of the underlying problem described in the message. Changing a column's collation is akin to changing its data type. That is, after you change the collation, the Visual Database Tools validate the column's data type. This validation can uncover a number of problems, including:

- The modification enlarges the column, which in turn enlarges an index beyond the maximum size for indexes.
- The modification is incompatible with existing data values in the column.

Choose a different collation for the column or fix the underlying problem.

Index '<0s>' is used to enforce the full-text key on table '<1s>' and must not be null.

You are performing an operation that will allow null values within the column that is used as a key for full-text indexing of this table. Proceed with the operation to disable full-text indexing for this table or cancel the operation to keep full-text indexing.

Index '<0s>' is used to enforce the full-text key on table '<1s>' and must be single-column.

You are adding a column to the index that is used as a key for full-text indexing of this table. Proceeding with the operation will disable full-text indexing for this table. Choose Yes to proceed with the operation and disable full-text indexing. Choose No to cancel the operation.

Index '<0s>' is used to enforce the full-text key on table '<1s>' and must be unique.

This operation will allow duplicate values on the index that is used as a key for full-text indexing of this table. If you proceed with the operation, full-text indexing for this table will be disabled. Choose Yes to proceed with the operation and disable full-text indexing. Choose No to cancel the operation.

Index '<0s>' is used to enforce the full-text key on table '<1s>' and must be <2d> bytes or less.

This action will enlarge the index that is used as a key for full-text indexing of this table beyond the maximum size for such an index. If you proceed with the operation, full-text indexing for this table will be disabled. Choose Yes to proceed with the operation and disable full-text indexing. Choose No to cancel the operation.

The identity increment must be a number containing <0d> digits or less.

You have entered an invalid value for the identity increment field. Enter an integer whose magnitude is less than the maximum indicated in the error message. If the data type of the column is TINYINT, you must enter a positive number. For any other data type, you can enter a positive or negative number.

Identity seed must be a number containing <0d> digits or less.

You have entered an invalid value for the identity seed field. Enter an integer whose magnitude is less than the maximum indicated in the error message. If the data type of the column is TINYINT, you must enter a positive number. For any other data type, you can enter a positive or negative number.

The table must have at least one column that is not computed.

You cannot create a table all of whose columns are computed. You can, however, create a view containing entirely computed data.

For more information, see <u>Queries and Views</u>.

The new relationship must have at least one pair of related columns.

The relationship must relate at least one column from the foreign-key table to at least one column in the primary key table. Select a column from each table or delete the relationship.

The collation properties of columns <0s> and <1s> do not match.

Each pair of corresponding columns from the primary-key table and foreign-key table of a relationship must have identical collation settings, but this relationship includes a pair of corresponding columns with different settings. Remove the columns from the relationship or change the collation settings to be identical.

Select both primary key table and foreign key table before selecting any field for the relationship.

You tried to select the columns from one table of a relationship before indicating what the other table is. You must first indicate both tables before proceeding to choose columns.

Your Logon does not have CREATE TABLE permission; you might not be able to make certain edits that require this permission.

You are creating or modifying a table in a database in which your logon ID does not have CREATE TABLE permissions and is not a member of the DBO role. Check with the database administrator to make sure that you have the necessary permissions to create tables in the database.

Remember, a seemingly innocuous modification to an existing table can require CREATE TABLE permission. For example, if you add a column to a table, the Visual Database Tools instruct SQL Server to delete the table, re-create it, and re-insert the rows it had contained.

Setting for Length must be from <0d> to <1d>.

You must enter a whole number within the range specified in the error message.

Setting for Precision must be from <0d> to <1d>.

You must enter a whole number within the range specified in the error message.

Setting for Scale must be from <0d> to <1d>.

You must enter a whole number within the range specified in the error message.

Property cannot be modified.

You attempted to change a property that is incompatible with other properties already specified.

A diagram with that name already exists in the database.

A diagram of that name already exists in the database. Choose Yes to replace the existing diagram with the one you are saving.

Another user modified this diagram while you were working on it.

Since you opened the diagram and began working on it, another user has saved modifications to it in the database. If you save your modifications, you will overwrite that user's changes.

For more information, see <u>Multiuser Environments</u>.

The table being loaded into memory has a userdefined data type that is not recognized.

The table you are loading into memory refers to a recently created user-defined data type, but your in-memory list of user-defined data types does not include it.

Because the data type was created recently, it is not present in local memory. All user-defined data types are loaded from the database into local memory when you open the first Table Designer window or database diagram within your connection to that database. To refresh your in-memory copy of the database's user-defined data types, close all open database diagrams and Table Designer windows within that database connection. After you close them all, reopen them in turn. When you reopen the first one, the list of user-defined data types will be refreshed.

For more information, see <u>Multiuser Environments</u>.

Your diagram will be updated with the following changes to match the database before the following tables can be loaded.

A table you are trying to add to the diagram is related to a table already on the diagram, and that relationship somehow contradicts the contents of your diagram. This occurs because another user has added or modified the relationship since you began working on the diagram.

Remember, when you add a table to a diagram, the Database Designer automatically includes any relationships between the existing tables and the newly added table. In attempting to add such a relationship, the Database Designer has discovered that your in-memory copy of the database structure no longer matches the contents of the database. Before adding the table, the Database Designer will update your diagram accordingly.

For example, suppose your diagram is initially consistent with the database. The diagram includes two tables, Student and Course, and a one-to-one relationship between them, called TutoringAssignment. The relationship indicates that each student can tutor a course and each course can be tutored by a student. Initially, your diagram includes no other tables.

While you are working on the diagram, another user changes the database structure. That user deletes the TutoringAssignment relationship and adds another relationship of the same name between the tables Student and SectionOfCourse. (This change improves the database structure, because it more faithfully represents your organization's information needs. Students are assigned to tutor particular sections of courses, not courses in general.) The other user commits these changes to the database.

Now you add the SectionOfCourse table to your diagram. Because this table has a relationship to the Student table already present on your diagram, the Database Designer tries to include this relationship on the diagram. The conflict arises because the to-be-included relationship has the same name as a now obsolete relationship already present on your diagram.

Because of these conflicts, the Database Designer will update your in-memory copy of the database structure accordingly. That is, it will remove the TutoringAssignment relationship between the tables Student and Course. Only then will the Database Designer add to your diagram the SectionOfCourse table and the TutoringAssignment relationship between the Student and SectionOfCourse tables.

Note The Database Designer does not reconcile all differences between your diagram and the database. It reconciles only those differences that would otherwise prevent you from adding the table to the diagram.

Your table will be updated with the following changes to match the database.

The Table Designer needs to load information about the table to which you are trying to create a relationship. For example, it needs to load the details about the other table's columns, so you can choose which columns from the table you are designing correspond to columns in the other table.

The contents of the database and the in-memory contents of the Table Designer are in conflict. This conflict occurs because another user has recently modified the table you are working on, but the Table Designer retains the older version of the table in your local memory.

The body of the message shows the details of the conflict. Before proceeding to load the other table into your local memory, the Table Designer will reconcile these conflicts accordingly.

Note The Table Designer does not reconcile all differences between your local memory and the database. It reconciles only those differences that would otherwise prevent you from creating the relationship you are trying to create.

The following schema-bound objects will be modified.

The object you are modifying contributes to the definition of one or more schema-bound objects. Those schema-bound objects will be modified or dropped. The body of the message gives the details.

Column is the full-text filter for columns that participate in full-text indexing on a table.

You are trying to delete a column or change its data type, but that column is the full-text filter for one or more other columns for which a full-text index exists. If you proceed with the operation, the columns will no longer participate in the full-text index.

If you later want to reestablish full-text indexing for these columns, you must choose or create a new full-text filter field, and you must add these columns back to the full-text index. For more information about accomplishing these tasks, see <u>Full-Text Indexes</u>.

Query Designer Troubleshooting Errors

This section contains a list of possible messages returned while creating and running queries using the Query Designer. These errors often occur while trying to save or execute a query.

For information about additional Microsoft Visual Database Tools error messages, see <u>Database Designer Troubleshooting Errors</u>.

Note If you receive an error message not listed in the Visual Database Tools section, see <u>Error Messages</u>.

(+) operator ignored.

The Oracle outer join operator that you entered is not appropriate and will be ignored.

See <u>Types of Joins</u> and <u>Creating Outer Joins</u> for more information.

(+) table reference cannot be joined with more than one table.

There is an error in the way that you are attempting to join a table reference. This table reference can only be joined with one other table.

For more information, see <u>Types of Joins</u>.

<0s> in expression is not part of the query.

An expression within your query is not properly formed. This often occurs when the criteria expression that you entered doesn't match the available columns for the input sources. Clear the value out and then type in a column reference that matches your input sources.

For more information, see <u>Using Expressions in Queries</u>.

The Query Designer does not support the <0s> SQL construct.

The syntax you entered is valid but is not supported visually by the Query Designer. Be sure to verify your syntax before saving.

For more information, see <u>Supported Query Types</u>.

<0s> cannot be used in this query type.

The action you are attempting is not permitted with the type of query you have selected. Verify the syntax in your query or change the query type.

For more information, see <u>Supported Query Types</u>.

<0s> support not available in this server version.

You have attempted to use a feature that isn't supported by your server software.

A number of rows were affected.

The query executed successfully. The specified rows were affected by a make table process.

Ambiguous outer join (+) operator.

The Query Designer requires that you follow specific rules when designing queries. The Query Designer is unable to process the outer join operator you have entered. Check the syntax in your SQL statement.

For more information, see <u>Creating Outer Joins</u>.

Appropriate SQL cannot be generated.

The Microsoft Visual Database Tools Query Designer requires that you follow specific rules when designing queries. This error may appear if you have used the Diagram or Grid pane to create a query that cannot be expressed in an SQL statement. Verify the query options you have selected.

For more information, see <u>Creating Queries</u> or <u>Specifying Search Criteria</u>.

You are about to delete a row or rows.

Choosing Yes will delete the selected row from the table. If you say Yes, the deletion will be immediately transmitted to the database. You will not be able to Undo this change.

For more information about deleting rows, see <u>Deleting Rows in the Results</u> <u>Pane</u>.

For more information about changing data using queries, see <u>Manipulating Data</u>.

Bad top value.

The top value in your query is invalid or out of range. The value for the top clause must be an integer value if the PERCENT clause is not used. To use the PERCENT clause, just type in the word PERCENT after the numeric value in the cell.

For more information, see Entering Search Values.

Cannot put expression on select list.

There are limitations to what expressions are allowed in the select list. The expression you are trying to add may be invalid. Verify that the syntax in the expression is correct. It is possible that the type of expression is invalid.

For more information, see <u>Using Expressions in Queries</u>.

Cannot assign alias to this field.

An alias can only be created for a suitable column or table. The field you have chosen is not valid. Verify that the settings for the field are correct.

For more information, see <u>Creating Column Aliases</u> or <u>Creating Table Aliases</u>.

Cannot convert entry to valid date/time.

The Microsoft Visual Database Tools Query Designer requires that you follow specific rules when designing queries. Verify that the entry is correct.

For more information, see <u>Manipulating Data</u>.

Cannot convert to proper type.

The Microsoft Visual Database Tools Query Designer requires that you follow specific rules when converting values. The query cannot be converted. Verify that the information in the query and the query type are appropriate.

For more information, see <u>Functions for Expressions</u>.

Cannot delete rows with unknown keys.

The row you've attempted to delete has an unknown key. This row cannot be deleted at this time.

For more information, see <u>Deleting Rows in the Results Pane</u>.

Cannot edit rows with unknown keys.

The row you've attempted to edit has an unknown key. For more information, see <u>Editing Rows in the Results Pane</u>.

Cannot edit this cell.

The cell you've selected cannot be edited. It may be the result of a calculation or part of a joined table.

For more information, see <u>Manipulating Data</u>.

Cannot filter this expression.

The expression you've entered cannot be filtered. This typically occurs when the user types in a criteria against the * or a criteria against a field that is blank in the QBE. The user should only enter criteria against fields that are already in the QBE.

For more information, see <u>Specifying Search Criteria</u>.

Cannot insert into this expression.

You cannot insert data into the selected expression. This function is not supported at this time.

For more information, see <u>Creating Insert Queries</u>.

Cannot open encrypted *<***0***s><***1***s>***.**

After a view has been encrypted and closed it can't be opened again.

For more information, see <u>Creating Views</u>.

Cannot update this expression.

The selected expression cannot be updated.

For more information, see <u>Using Expressions in Queries</u>.

Cannot use column whose data type is IMAGE or TEXT in this context.

The Microsoft Visual Database Tools Query Designer requires that you follow specific rules when designing queries.

For more information, see <u>Creating Queries</u> and <u>Specifying Search Criteria</u>.

Cannot use LONG data type in this context.

The Microsoft Visual Database Tools Query Designer requires that you follow specific rules when designing queries.

For more information, see <u>Creating Queries</u> and <u>Specifying Search Criteria</u>.

Column list not supported for Make Table query.

The Query Designer does not support inclusion of the optional column list for the Oracle Make Table query. Remove the column list and use column aliases in your select statement instead.

For more information, see <u>Creating Make Table Queries</u>.

Column with (+) operator does not reference a valid table.

The operator you've used in the query column must reference a valid table. The Query Designer cannot validate the table currently referenced. Verify that the table exists in the database and that the Query Designer has access to it.

For more information, see <u>Specifying Search Conditions</u>.

Data source alias is read only.

Read-only values cannot be edited. To use this data source, you need a writeenabled version of the data source available to the Query Designer.

For more information, see <u>Manipulating Data</u>.

Data source base name is read only.

Read-only values cannot be edited. To use this data source, you need a writeenabled version of the data source available to the Query Designer.

For more information, see <u>Manipulating Data</u>.

Data type error in expression.

Expressions with different data types can generate errors. To work with different data types in a single expression, you need to convert to the same data type.

For more information, see <u>Creating an Expression</u>.

Data type mismatch - no conversion possible.

To work with different data types in a single expression, you need to convert to the same data type. The selected data types cannot be converted.

For more information, see <u>Using Expressions in Queries</u>.

Data type mismatch - use the CONVERT function.

To work with different data types in a single expression, you need to convert to the same data type. One of the data types must be converted before you can continue.

For more information, see <u>Using Expressions in Queries</u>.

DELETE statement conflicted with COLUMN REFERENCE constraint.

A conflict occurred while your query attempted to execute. The QueryDesigner has tried to locate the direct source of the conflict. Please check the syntax of your query to attempt to resolve the conflict.

Do you want to suppress further error messages telling you why records can't be pasted?

If you answer Yes, you will not receive messages explaining why records cannot be pasted correctly.

Error after function '<0s>'

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error after predicate near '<0s>'

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error before EXISTS clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error before EXISTS clause: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error before FROM clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error following UNION operator.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error in column list.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Creating Queries</u>.

Error in column list: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in destination table specification.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For best results, do not include periods in the destination table specification.

Error in FROM clause near '<0s>'

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in GROUP BY clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Summarizing and</u> <u>Grouping</u>.

Error in GROUP BY clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Summarizing and</u> <u>Grouping</u>.

Error in HAVING clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Querying on Groups</u> <u>of Rows</u>.

Error in HAVING clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Querying on Groups</u> <u>of Rows</u>.

Error in INSERT statement.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in INSERT statement: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in join expression.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Querying Using</u> <u>Multiple Tables</u>.

Error in join expression: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Querying Using</u> <u>Multiple Tables</u>.

Error in list of function arguments: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in list of values.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in list of values in IN clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in list of values in IN clause: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in list of values: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in ON clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in optional FROM clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in ORDER BY clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in ORDER BY clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in SELECT clause: alias '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in SELECT clause: alias not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in SELECT clause: expression near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in set list in UPDATE clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in table name or view name in DELETE clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query. Be sure to use Quoted Identifiers as needed.

Error in table name or view name in DELETE clause: <0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query. Be sure to use Quoted Identifiers as needed.

Error in table name or view name in INSERT clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query. Be sure to use Quoted Identifiers as needed.

Error in table name or view name in INSERT clause: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query. Be sure to use Quoted Identifiers as needed.

Error in table name or view name in UPDATE clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query. Be sure to use Quoted Identifiers as needed.

Error in text following query statement: '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in values list in INSERT INTO clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Error in values list in INSERT INTO clause: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Make sure the number of items in the Column list matches the number of items in the Values list.

Error in WHERE clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Querying on Groups</u> <u>of Rows</u>.

Function argument count error.

A function in your query has the wrong number of arguments. Review you query syntax to locate the function.

For more information, see <u>Functions for Expressions</u> and <u>Specifying Search</u> <u>Criteria</u>.

Ignoring illegal use of ALL.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Although the query will run, you should review your query syntax and correct the error.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Ignoring ODBC syntax.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Illegal use of expression list.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Illegal sequence use.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Illegal use of outer join operator.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Outer Joins</u>.

Incomplete column list.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Incomplete SET clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Incomplete VALUES list.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid entries must be resolved before you can exit this pane.

Exiting a pane causes other panes to be updated. If there is an invalid entry in the pane you are editing, you need to change or delete it.

Invalid escape character.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Your escape character must be a single character.

Invalid identifier '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid or missing expression.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid or missing expression near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid prefix or suffix characters.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid row index: Goto failed.

You have entered a non-numeric value into the GoTo record field. Enter a numeric value instead. Entering a number that is greater than the number of rows in the table, goes to the last record.

For more information, see <u>Specifying Search Criteria</u>.

Invalid text or symbol.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Invalid view name.

The View name you've specified is not valid. Valid names begin with an underscore (_) or a character and contain a combination of characters, numbers and underscores.

The maximum length for View names is 32 for Microsoft® SQL Server[™] 6.5, 32 for Oracle, and 128 for SQL Server 7.0 and SQL Server 2000. Do not use names with a period.

For more information, see <u>Working With Views</u>.

Missing escape character in LIKE predicate.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Missing FROM clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Missing FROM clause near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Missing FROM keyword.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing FROM keyword near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing INTO keyword.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing INTO keyword near '<0s>'.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing or incomplete SELECT clause.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing pattern in LIKE predicate.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing SET keyword.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u> and <u>Specifying Search</u> <u>Conditions</u>.

Missing subquery.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Subqueries</u>.

Missing subquery correlation name.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Subqueries</u>.

Missing subquery or the operator you entered requires parenthesis.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Subqueries</u>.

Must enter either TRUE or FALSE.

When specifying a criteria for the EXISTS statement, you must use either True for EXISTS or False for NOT EXISTS.

For more information, see <u>Specifying Search Criteria</u>.

Outer join column may not be used with an IN predicate or subquery.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Outer Joins</u>.

Query Designer cannot open this query file.

Query Designer cannot open the specified query file. Verify that file is a query file. The file may be corrupt or from a non-supported version of the Query Designer.

For more information, see <u>Supported Query Types</u>.

Query has executed successfully.

The query executed successfully.

Row limit value should be between <**xxx**> **and** <**xxx**>.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see Entering Search Values.

SQL statement could not be parsed.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Queries</u>.

The SQL syntax has been verified against the data source.

The SQL verified successfully.

For more information, see <u>Creating Queries</u> and <u>Specifying Search Criteria</u>.

Syntax error in table reference.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Queries</u> and <u>Specifying Search Criteria</u>.

Syntax error in table reference: '<0s>' not recognized.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u>.

ODBC driver returned an invalid **ODBC** version and needs to be updated.

Your ODBC driver is not current. For best results, use the ODBC driver provided with the Microsoft Visual Database Tools installation.

For more information refer to your ODBC documentation.

Use of GROUP BY function in WHERE clause not allowed.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Querying on Groups of Rows</u>.

The NOT keyword may not be used in a column cell.

When entering a NOT EXIST subquery into the Column cell, enter just the EXIST clause. Then go to the Criteria cell and enter False for it.

For more information, see <u>Specifying Search Criteria</u>.

ORDER BY not supported for CREATE TABLE query.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Reordering Output Columns</u>.

Outer join operator (+) not allowed as OR operand.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Outer Joins</u> and <u>Specifying Search Criteria</u>.

Outer join operator (+) not allowed in the grid pane.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Outer Joins</u>.

The query cannot be executed because some files are missing or not registered.

Query Designer cannot locate your query files and cannot run your query. You may need to reinstall the product.

The Query Designer does not support the critical ODBC APIs.

The API you are attempting to use is not supported. Refer to your ODBC driver documentation. Use Microsoft provided ODBC drivers whenever possible.

The Query Designer does not support the current ODBC API.

The API you are attempting to use is not supported. Refer to your ODBC driver documentation. Use Microsoft provided ODBC drivers whenever possible.

Query Designer supports no more than 1 table for this type of query.

Only one data source can be used with this query type.

For more information, see <u>Supported Query Types</u>.

The specified OR group number is already in use.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Specifying Search Criteria</u>.

The value you entered is not consistent with the data type or length of the column.

Entered values must match the column data types.

For more information, see Entering Search Values.

There are not enough values in the subquery select list.

The SELECT statement needs to include the same number of columns that the embedded subquery returns.

For more information, see <u>Creating Subqueries</u>.

There are not enough columns to match the value list.

An UPDATE or INSERT statement contains more values than the table being updated. Verify your SQL statement to make sure the number of columns matches the table.

There are no columns selected. Please select one or more columns and re-run the query

The <u>SELECT</u> list does not contain any output columns. Specify the columns you want to be returned and run the query again.

There is no unique table in this query.

The query needs to unambiguously identify which table to return values from.

ALIAS name is already being used.

Alias names must be unique. Enter a different alias name.

For more information, see <u>Creating Table Aliases</u> and <u>Creating Column Aliases</u>.

This cell contains the text string "<NULL>" which may not be processed correctly.

Use NULL or NOT NULL for comparing null data. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Queries</u>.

IS operator can only be used with NULL or NOT NULL.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

This operator cannot be used with columns with data type "uniqueidentifier".

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Table is not in the query definition.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Queries</u>.

Too many characters for field width.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

For more information, see <u>Creating Queries</u>.

Unable to locate data source.

The specified data source cannot be found. Check the data source location and the entered location.

Unable to parse expression.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Unable to parse query text.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Unable to parse statement.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Unknown column.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Unknown conversion specification.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Use of CONVERT function might be unnecessary.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

SQL text cannot be represented in the grid pane and diagram pane.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

View already exists in the database.

Views must have unique names within the database. Enter a different name for the view.

For more information, see <u>Creating Views</u>.

You might not have permission to perform this operation, or the object <0s> might no longer exist in the database.

The object you want to work with may be listed, but you may not have SELECT permission to this object. Or, the object may no longer exist in the database.

Refresh the list of objects to find out if this object still exists in the database. If this object still exists, then contact the owner of the object or the Database Administrator to get permission to this object.

For more information, see <u>Ownership of Database Objects</u>.

Link server object cannot be used as a destination with this query type.

The selected query type does not support using link server objects as destinations.

For more information, see <u>Creating Queries</u>.

ALL cannot be used with CUBE or ROLLUP.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

OpenRowset cannot be used as a destination with this query type.

There is an error in the syntax of your query. The Query Designer has attempted to locate the source of the error. Review your query syntax and correct the error before running your query.

Enter an expression in the Column cell first.

Enter an expression in the column cell.

For more information, see <u>Using Expressions in Queries</u>.

Your entry cannot be converted to a valid date time value.

In general, the ANSI standard date format is used with databases that represent dates using a true date data type. In contrast, the datetime format is used with databases that support a datetime data type.

For more information, see Entering Search Values.

Query Designer Error.

No specific help is available for this error. If the text in the error message does not provide enough information for you to solve the problem, please consider the following options:

- Verify that your settings are correct and try the action again.
- Use the index to search for keywords related to the error message.
- If the problem is due to external applications or software, consult the documentation for that area. For example, errors may be caused by your server setup, or even the network you are using.
- For SQL statement errors, consult the <u>Transact-SQL Reference</u>.
- Check the Readme for late-breaking information concerning the error.

The alias name is too long.

The name you supplied as an alias for a table or table-structured object is too long. Enter a shorter name.

Poorly formed comment.

The comment contains an error. Be sure that the body of the comment does not contain any comment markers (/* or */).

An expression cannot be used as a parameter value.

An expression appears as a parameter value. Parameter values must be scalar quantities or simple text. Supply a text value or number for the quantity, as appropriate.

Incomplete parameters or column list.

An argument is missing from an insert statement. Rewrite the insert statement to supply the missing argument.

Incomplete parameters list.

A parameter is missing from a user-defined function that returns a table. Rewrite the statement to supply the missing argument.

Warning: It is likely that your modification will result in a view that cannot be indexed.

The modifications you are trying to transmit to the database might result in a view that cannot be indexed. If you proceed with the modifications, the view's existing indexes will be deleted.

Warning: It is likely that the view definition will result in a view that cannot be indexed.

The view definition you are trying to transmit to the database might result in a view that cannot be indexed. You can proceed to save the view, but SQL Server might not create the indexes for it.

Index already exists.

The database already contains an index with the name you supplied. Choose another name.

Unquoted alias contains white space.

An alias you supplied contains embedded blanks or other white space. Remove the white space or put double-quotation marks around the alias.

If you save the view encrypted, you will no longer be able to alter the view definition.

By encrypting a view, you disallow all users (regardless of database privileges), including yourself, from ever seeing the view definition. Thus, you will not be able to alter the view definition. Choose OK to save the view in encrypted form. Choose Cancel to cancel the Save operation.

View has indexes. If you remove schema binding, the indexes will be dropped.

The operation you are performing will automatically remove schema binding from an indexed view. When schema binding is removed, the indexes will be dropped. Click OK to proceed with the operation; click Cancel to cancel the operation.

For more information, see <u>Indexed Views</u>.

Edits not allowed. HAVING clause not allowed in an indexed view.

You cannot use the HAVING clause in the view definition of an indexed view. You can remove the indexes or you can remove the HAVING clause.

Database Server Version Runtime Error.

The database server has returned a run-time error. See <u>Error Messages</u>.

SQL Verify failed.

The database server could not verify the validity of the SQL statement. For more information, consult the <u>Transact-SQL Reference</u>.

Poorly formed cast function.

An expression contains a bad <u>CAST</u> function.

Illegal use of collation clause.

The SQL statement uses the collation clause illegally. For more information, see <u>COLLATE</u>.

Columns in this expression have incompatible collations.

An expression you supplied in an indexed view definition uses two columns with different collation settings, so a collation clause has been added to one of the columns. But an indexed view definition cannot contain the collation clause. Modify the view definition by either removing the expression with the incompatible columns or remove the indexes from the view.

Collate clause may not be used in an indexed view.

The COLLATE clause appears within the definition of an indexed view, but an indexed view definition cannot contain the COLLATE clause. Modify the view definition by either removing the COLLATE clause or removing the indexes from the view.

Asterisk (*) may not be used in an indexed view.

Because an indexed view is stored on disk, the view definition must explicitly refer to the columns to be included in the view. Replace the asterisk with the columns you want to include in the view.

Index must have at least one column.

Your modification would result in an index without columns, which is illegal. You can delete the index or modify it by adding a column to it.

Invalid fill factor; enter an integer between 0 and 100.

Because fill factor must be a percentage, its minimum value is zero and its maximum value is 100. You can use only whole-number percentages. For example, 67 is acceptable, but 66.7 is not.

Indexed View must contain a clustered index.

Because an indexed view is stored on disk, it must contain a clustered index (rather than a nonclustered one). Modify the index to make it a clustered index.

The table-valued function used as target is not an inline function.

The operation you are attempting requires an updateable user-defined function (an in-line function), but the function you are using is not updateable. You can replace the function with an updateable one, or you can change the query type to a SELECT query.

WITH TIES clause requires an ORDER BY clause.

The query does not conform to SQL syntax, because it uses the WITH TIES clause without an attendant ORDER BY clause. Either remove the WITH TIES clause or add an ORDER BY clause.

There are too many values from the sub-query select list.

The number of columns in the target of the INSERT operation exceeds the number of data values you supplied. Modify the query to include one value for each column.

NOT cannot be used in a column cell.

To negate a criterion that contains a subquery, you cannot place the NOT operator in the Column cell of the Grid pane. Instead, you should put the keyword FALSE in the Criteria cell of that same row of the Grid pane.

View definition includes no output columns or no items in the FROM clause.

A view definition must have at least one table or table-structured object in the FROM clause, and must have at least one column in the select list. The view definition is missing one or both. Modify the view definition accordingly.

You cannot create a view which is self-referenced.

The view definition's FROM clause includes the name of this view. Remove the view name from the FROM clause.

This SQL statement type cannot be used in a view.

A view definition must be a SELECT statement (rather than an INSERT, UPDATE, DELETE, or CREATE TABLE statement). Alter the view definition accordingly.

Query or View has been modified. Save changes before closing?

You have changed the query or view. If you do not save it, your modifications will be lost. Choose Yes to save the changes; choose no to cancel the operation.

Query or View already exists. Do you want to overwrite it?

A query or view with that name already exists. If you choose Yes, you will replace the existing query with the new one.

You are about to paste n rows.

When you paste rows in the results pane, the rows are immediately transmitted to the database. Because the database commits these modifications when you transmit them, you cannot Undo this change.

Rows pasted.

The rows you pasted in the results pane have been transmitted to the database. Because the database committed these modifications when you transmitted them, you cannot Undo this change.

Column cannot be updated.

You cannot update values in this column – perhaps because it is a computed column, and identiTy column, or a column whose values are RowGuids.

The database row you are modifying no longer exists in the database.

After you retrieved data into the Results pane, another user deleted a row from the database. You are now trying to modify that row from within the Results pane. Your modification will not work, because the row that is the target of your modification no longer exists.

Too many or too few parameters specified.

You are using a user-defined function, but you are not supplying the correct number of parameters. Check the definition of the user-defined function to determine how many parameters it takes.

View name cannot begin with #.

The view name cannot begin with a pound sign (#). Choose a different name.

Table name cannot include the double-quote character.

The table name cannot include the double-quote character ("). Choose a different name.