ExcelSQL Add-In

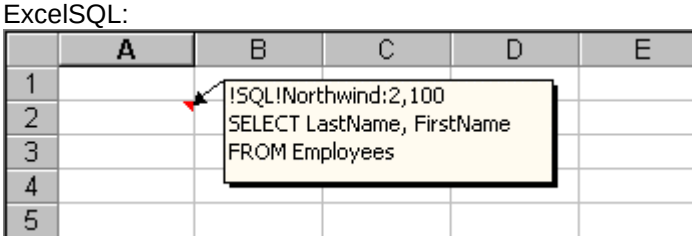
Version 2.5
August 31st 2000

Table of contents

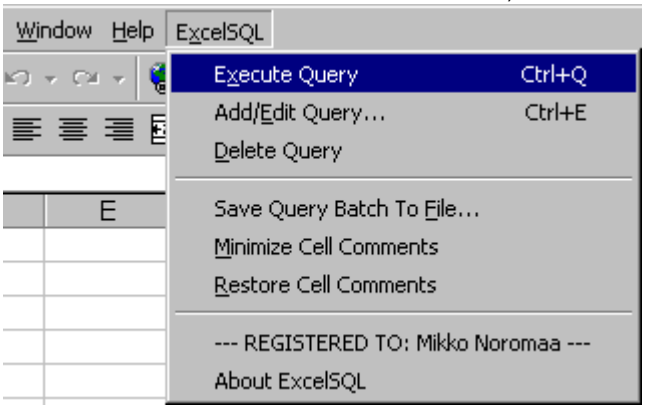
1. Introduction

ExcelSQL is an add-in for Microsoft Excel. It can be used to fetch data from different SQL databases into Microsoft Excel. ExcelSQL works as follows:

1. User adds a cell comment into a cell in Excel. The comment contains a header recognized by ExcelSQL:



2. User selects the cell with the cell comment, and selects Execute Query from ExcelSQL menu:



3. At this point, ExcelSQL reads the text from the cell comment, parses it, and sends it to the ODBC datasource defined in the cell comment (Northwind).

4. The database sends back results of the query. ExcelSQL processes the results, and writes them onto the Excel sheet starting from the cell where the comment was entered:

	A	B	C
1			
2	Davolio	Nancy	
3	Fuller	Andrew	
4	Leverling	Janet	
5	Peacock	Margaret	
6	Buchanan	Steven	
7	Suyama	Michael	
8	King	Robert	
9	Callahan	Laura	
10	Dodsworth	Anne	
11			

2. Installing ExcelSQL

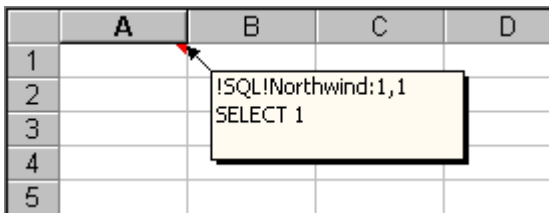
ExcelSQL works on Microsoft Excel 97 and 2000. It requires that Data Access Objects (DAO) is installed.

To install ExcelSQL, do as follows:

1. Unzip the excelsql25.zip file to a directory on your computer (for example, C:\ExcelSQL). Extract folder structure as well.
2. Start Microsoft Excel, and select the Add-Ins command from the Tools menu.
3. In the Add-Ins dialog, click the Browse button. Select the ExcelSQL.xla file from your local directory (C:\ExcelSQL), either from the 97 or 2000 folder, depending on your Excel version.

After installation, you should see the ExcelSQL menu appear. If you get any errors, see the Troubleshooting section at the end of this manual. The most common cause for unsuccessful installation is that Data Access Objects (DAO) is not installed, or a wrong version of it is installed. You can install DAO via the Microsoft Office Setup program.

Sometimes installation problems do not surface at the installation yet. It is strongly recommended that you test any new installation of ExcelSQL by executing a real query from a database. The query can be very simple, for example:



Even if you don't have a datasource called Northwind defined on your computer, this query will test the most error-prone part of ExcelSQL, the link to DAO. This query should prompt for a username and password (to which you can enter anything), and then report an error about ODBC-connection failing. If this happens, you know that ExcelSQL works fine on your computer. If ExcelSQL reports another error, see the Troubleshooting section at the end of this manual.

3. Using ExcelSQL

Inserting Basic SQL Queries

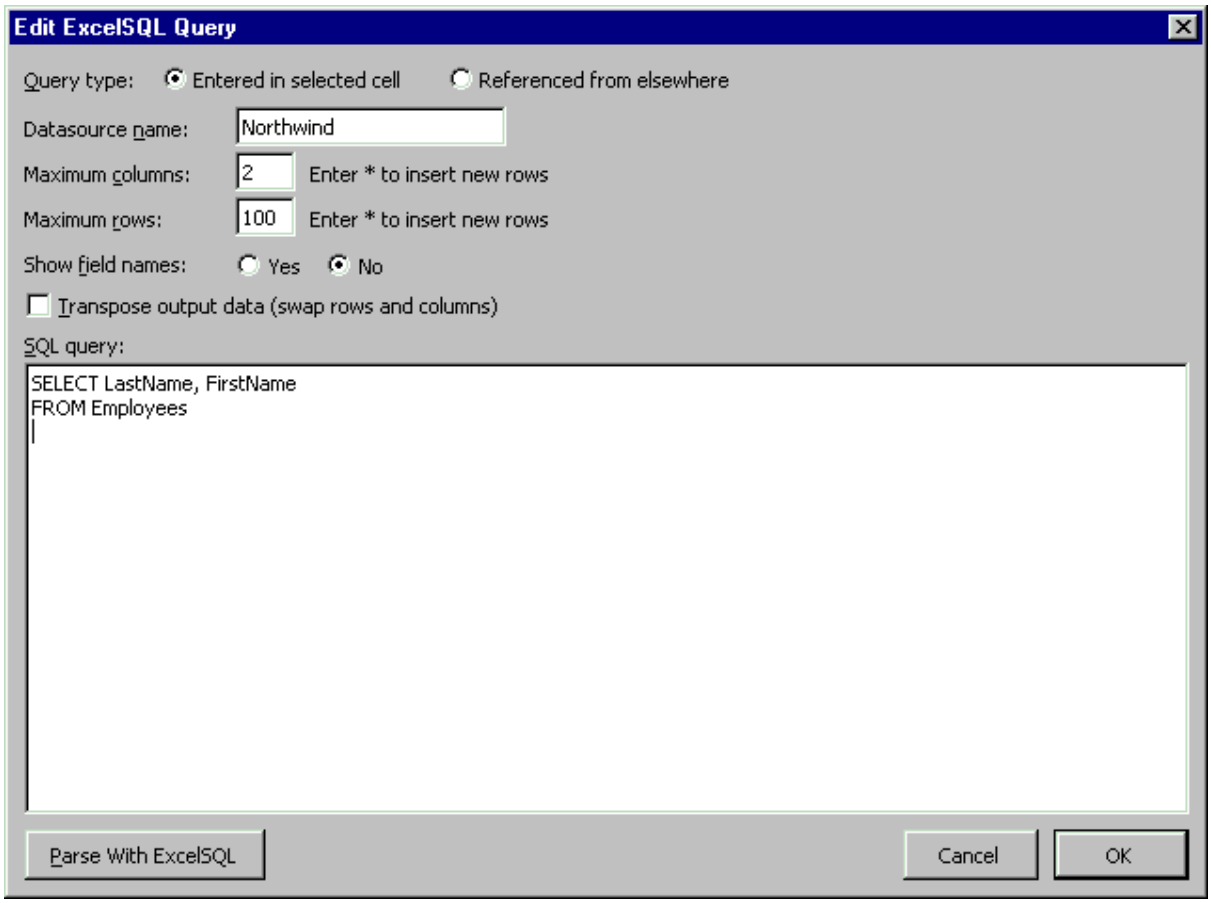
SQL queries executed by ExcelSQL are entered as cell comments in Excel. There are two ways to enter cell comments: 1) with Excel's own functinality (select Comment from the Insert menu), or 2) with ExcelSQL's query editor. The first option is most straightforward, and can be used on computers where ExcelSQL is not installed. The latter option is easiest, as the user does not have to remember the syntax of the ExcelSQL cell comment.

For details on entering ExcelSQL queries without the query editor, see the section ExcelSQL query syntax.

To start the query editor, select Add/Edit Query from the ExcelSQL menu.

ExcelSQL Query Components

The ExcelSQL query editor looks as follows:



The options in the ExcelSQL query editor are explained below:

Query type	Either "Entered in selected cell" or "Referenced from elsewhere". For details about this option, see the section Sharing Identical Queries Without Rewriting Them later in this manual.
Datasource name	This option specified the name of the datasource where the query will be sent. Datasources are defined in the ODBC Data Sources applet in Control Panel.
Maximum columns	Specifies the maximum number of columns to return.
Maximum rows	Specifies the maximum number of rows to return. If you don't know how many rows the query is going to return, enter an asterisk (*) here. ExcelSQL will then insert space for new rows as they are returned.
Show field names	If you select to show field names, ExcelSQL will write the column labels as part of its output. If you select No here, ExcelSQL will write only the data out.
Transpose output	This setting specifies to "swap" rows with columns, and vice versa. For more information on the transpose setting, see the section "Transposing Output Data".
SQL query	This is where you enter your SQL query. Most of this manual describes the different options you can use in constructing the SQL query.

Executing ExcelSQL Queries

ExcelSQL queries are executed by selecting the Execute Query command from the ExcelSQL menu. You can also use the shortcut key, Ctrl+Q.

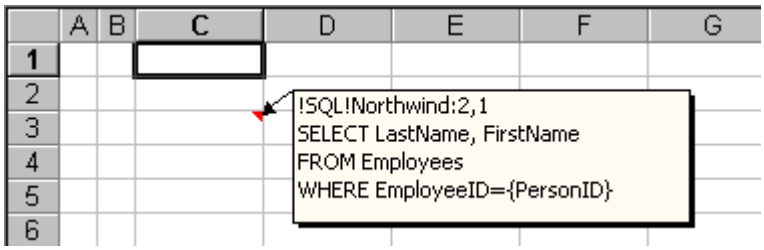
There can be two kinds of errors when executing a query: those that come from ExcelSQL and those that come from the SQL database. For example, if a reference cannot be parsed, the error is coming from ExcelSQL. On the other hand, if there is a comma missing from an SQL field list, it is an error coming from the SQL database.

4. References In ExcelSQL Queries

Because queries are stored in Excel cell comments, there is no Excel functionality for adding references to other cells. Since references are so essential in constructing any kind of an Excel solution, ExcelSQL includes the same features for adding references to cell comment as Excel does in using references in normal cells.

Using Absolute References

All references are entered between curly brackets ({}'s). For example:



In this example, cell C1 is named PersonID. If cells are moved, rows are removed/added, or the ExcelSQL is placed in a different cell, the name still refers to the correct cell. This is why all absolute references used in ExcelSQL queries should use named cells. It is also the suggested practise to use in normal Excel formulas as it makes the formulas much more readable.

Another way to use absolute references is to use the A1-style notation:

SELECT FirstName, LastName FROM Employees WHERE EmployeeID={C1}

Note that in this case the reference is *absolute*. Normally, the reference "C1" in Excel is relative, and C1 would be the absolute form. Relative reference is a reference which changes if it is copied to other cells. For example, if the reference C1 is copied from cell A1 to A2, it becomes C2 (pointing two columns right from the cell where the reference resides). Absolute reference stays always the same when copied between cells.

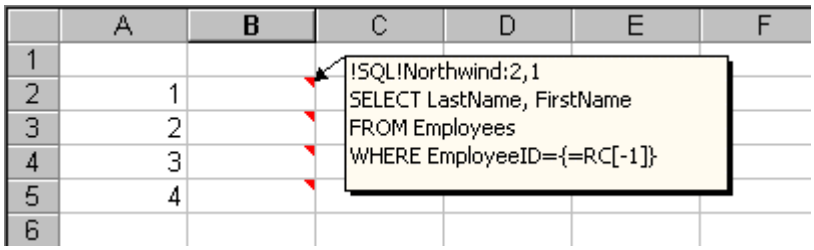
You can also use the normal absolute A1-syntax in ExcelSQL queries. In fact, A1, A1, $A1 and A$1 are all equal to ExcelSQL.

ExcelSQL will never change references when cell notes are copied, and thus absolute/relative distinction will not work for A1-style references. Note that if you add or remove rows/columns, Excel normally updates all formulas so that they refer to the same cells they originally referred to. For example, if there is a reference C1 in cell A1, and a new column is added between columns A and C, the reference is automatically updated to D1. ExcelSQL cannot do such updates, and all affected ExcelSQL query references must be manually changed.

Hint: ExcelSQL simply replaces all references written in {}'s before sending the query to the database server. If a referenced cell is empty, the resulting query could be, for example "...WHERE EmployeeID=". Of course, this will cause an SQL error to be returned from the server. You can prevent the error from occurring by preceding the reference with a 0: "...WHERE EmployeeID=0{PersonID}". In this case, the resulting query would be "...WHERE EmployeeID=0", which is syntactically correct, and would just return an empty recordset.

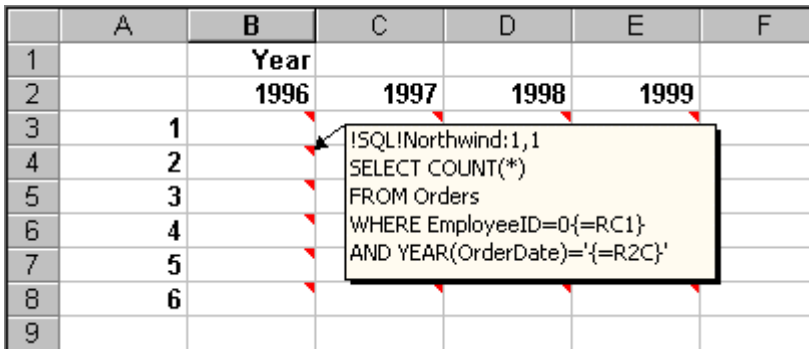
Using Relative References

As absolute references, relative references are also entered between curly brackets ({}'s). To make the distinction between absolute and relative references, relative references include an equal sign before the reference. For example:



Relative references must always be written in the R1C1-style, no matter what style is used in Excel formulas. In this example, "=RC[-1]" references the current row and the cell that is one cell left from the current column. You can easily see that you can copy this query to other cells, and it will work in them without modifications.

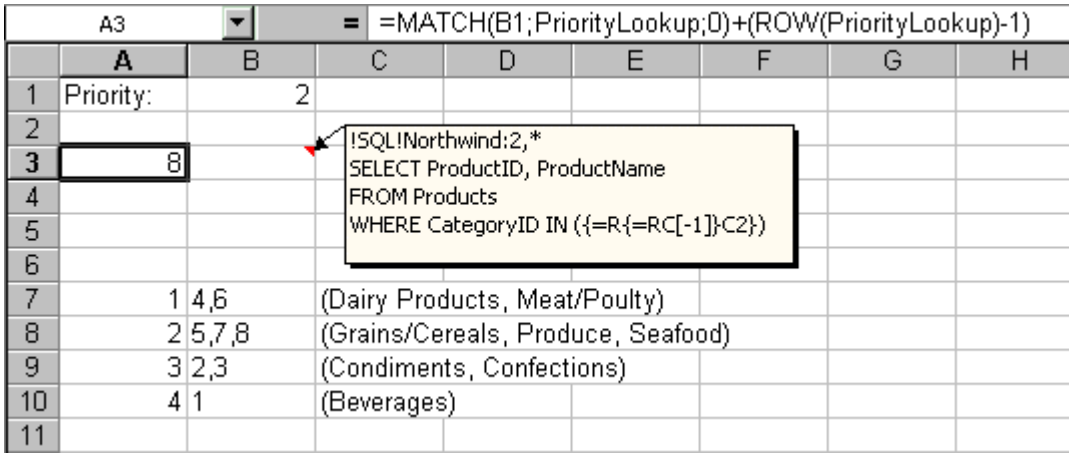
Standard R1C1-style constructs can be used in ExcelSQL references: Either row/column can be absolute, or even both. For example, "=RC1" references the column 1 of the current row. Thus, row is relative and column is absolute. A practical example is a matrix that can be filled with the same ExcelSQL query:



This example shows the number of orders generated by an employee by year. Again, the same query can be used in all cells, and it will return the data for the correct employee and year.

More Complex References

References can be nested inside each other. For example:



	A	B	C	D	E	F	G	H
1	Priority:	2						
2								
3	8							
4								
5								
6								
7	1	4,6	(Dairy Products, Meat/Poulty)					
8	2	5,7,8	(Grains/Cereals, Produce, Seafood)					
9	3	2,3	(Condiments, Confections)					
10	4	1	(Beverages)					
11								

Here the innermost reference (=RC[-1]) is evaluated first. It gets the value from cell A3, which is the result of the MATCH formula. The outermost reference is then =R8C2, whose values is substituted into the SQL query. This system implements a simple "addition" to the SQL database by grouping product categories into 4 priority levels. Note that this kind of addition requires no changes to the SQL database structure.

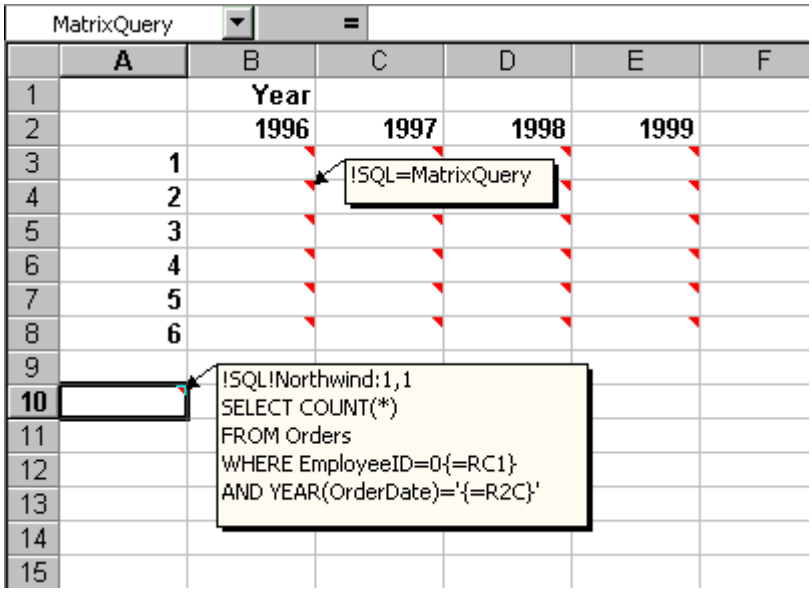
After doing reference substitutions, ExcelSQL does **not** scan the substituted part for other references. Otherwise it would be completely impossible to use curly brackets in your SQL statements. If you want to create references based on reference substitutions, you must enclose the double-reference in two pairs of brackets. For example: "{{ReferenceCell}}". If ReferenceCell contains the string "AnotherName", and AnotherName is a named cell on your worksheet, the reference will expanded to the contents of the AnotherName cell.

If you want to use curly brackets ({}'s) in your SQL statements, you must define two cells which contain the opening and closing bracket, and then reference these cells from your ExcelSQL query.

Sharing Identical Queries Without Rewriting Them

There may be cases where you have long and complex queries which you want to use in several different cells. By using relative references, you can simplify your task so that all cells can use the same query (see the second example under "Using relative references").

If the same long queries occupy hundredrs of cells, the Excel workbook may grow enormously large in size (several megabytes). To reduce the filesize, you can write just one copy of your query and reference it from other cells. For example:



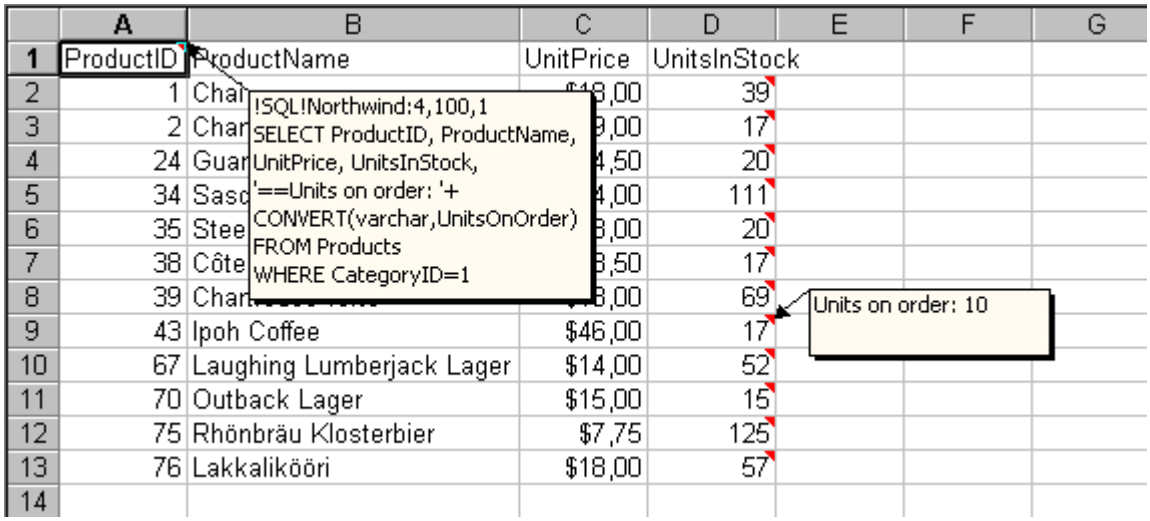
Referencing cell notes this way is equivalent to typing the same query in the referencing cell note. Thus, the placement of the actual query (in this case, cell A10) has no effect. When the query in cell B4 is executed, all references are relative to cell B4, **not** cell A10.

You can also use A1-style references, and R1C1-style references (relative) as the query reference. The notes given under section "Using absolute references" applies here as well.

5. Returning Special Data With Queries

Returning Cell Notes

Before writing query results out to the Excel sheet, ExcelSQL can process the results. For one, ExcelSQL write a specially formatted data as a cell comment. For example:



	A	B	C	D
1	ProductID	ProductName	UnitPrice	UnitsInStock
2	1	Chai	$18,00	39
3	2	Chan	9,00	17
4	24	Guar	4,50	20
5	34	Sasq	4,00	111
6	35	Stee	8,00	20
7	38	Côte	8,50	17
8	39	Char	8,00	69
9	43	Ipoh Coffee	$46,00	17
10	67	Laughing Lumberjack Lager	$14,00	52
11	70	Outback Lager	$15,00	15
12	75	Rhönbräu Klosterbier	$7,75	125
13	76	Lakkalikööri	$18,00	57
14				

When the data returned from a query starts with a double equal sign (==), the value will be inserted as a cell comment in the **previous** result cell. Note that the width of the output is 4 in this example, even though the query has 5 columns.

You cannot return a cell comment for the first column because otherwise the ExcelSQL query cell comment might be lost.

Also note that if you return cell comments, you should be sure to return one for **every** row returned by the query. Otherwise some rows would be longer than others. Be especially careful if your query can contain a NULL value: in this case '=='+NULL equal NULL, and there would thus be no cell comment returned.

To remove an existing cell note, return '==' without any other text. To insert a line break into the formula, insert a line break in the ExcelSQL query.

If you want to return a real value beginning with two equal signs, prefix it with an aposthrope ('). This way ExcelSQL won't interpret is as a cell comment, and Excel will show it as text without the aposthrope.

Note: To create really advanced ExcelSQL solutions, you can return other ExcelSQL queries from your ExcelSQL queries.

Returning Line Breaks

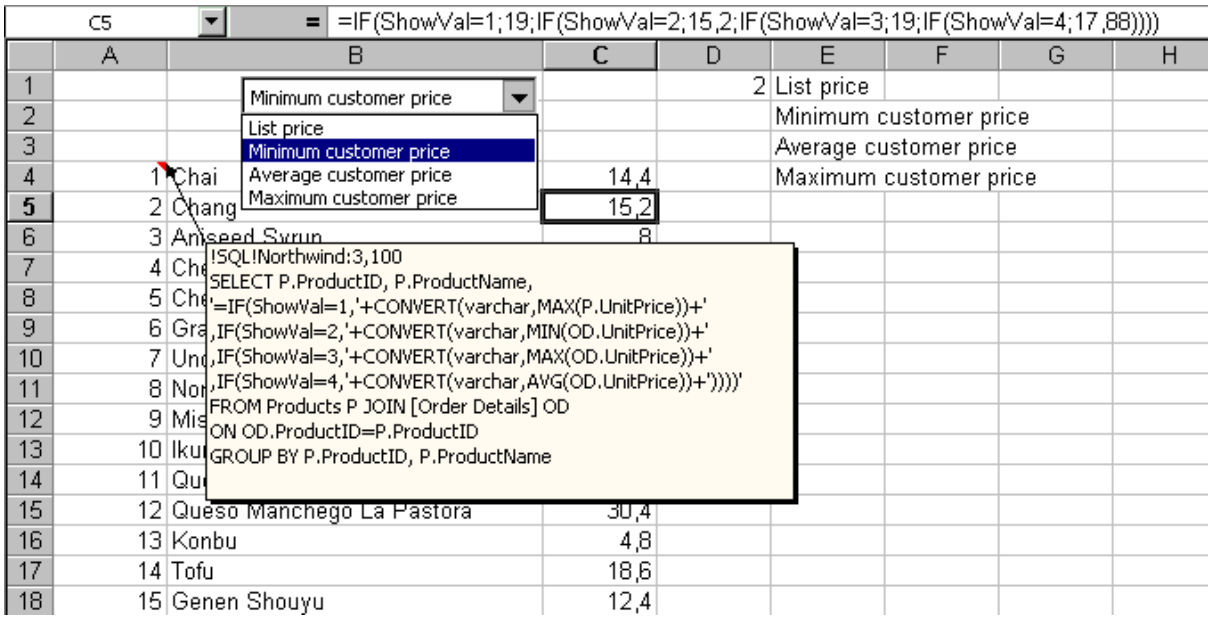
Sometimes you want the data on a single row of the resultset to be shown on several rows in Excel. For example, your query may return lots of fields and you must fit all of them into a single A4 page in portrait format. This is when you might want to take advantage of breaking the output to several lines.

When the data returned from a query contains three equal signs (===), ExcelSQL will wrap the rest of the fields to the next row. The three equal signs will **not** be written out to the Excel sheet. Be sure to return "===" for **every** row returned. Otherwise some rows will be longer than others.

Returning Formulas

To return formulas from cell notes, prefix the formula with an equal sign (=). This is not really even ExcelSQL functionality, because Excel interprets all strings starting with an equal sign as formulas.

Following is an example of a query that returns formulas:



Here cell D2 is named as "ShowVal", and the list in E1–E4 contains the choices in the dropdown box. When the dropdown box selection is changed, the value in cell D2 (ShowVal) changes from 1 to 4. The value shown in column C then changes depending on the selection.

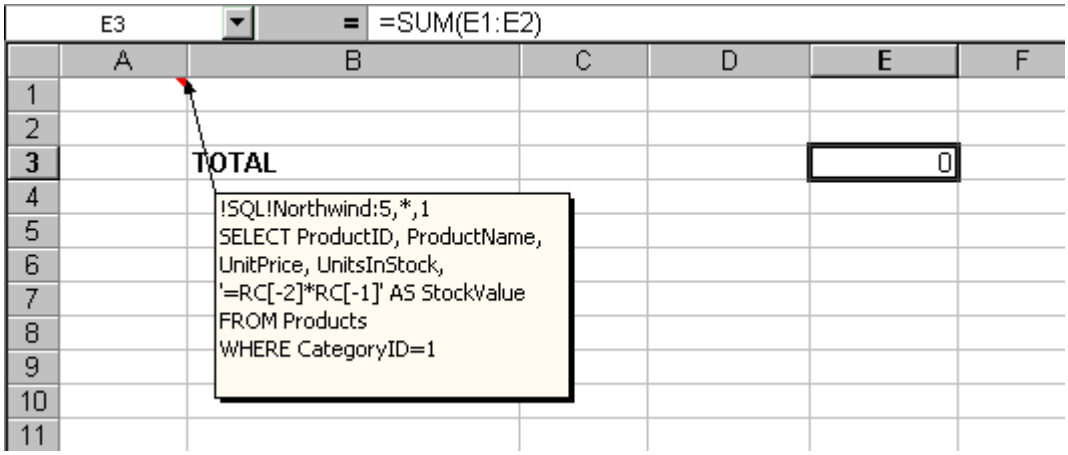
Note that this query would have been better written to just return the 4 different values in different columns. However, sometimes kind of a selection possibility may be useful: for example, if you have lots of other formulas that calculate the data based on the returned products, maybe even drawing charts from that. Then you can simply change the "view" of the data from one point, and see how that affects the charts, and everything else. This example also demonstrates a way to implement a 3-dimensional view in Excel.

Formulas returned from SQL queries must use the English language syntax even if you use a localized version of Excel. All functions must be in English (like IF), decimal separator must be a period (.), array separator must be comma (,) and semicolon (;), etc.

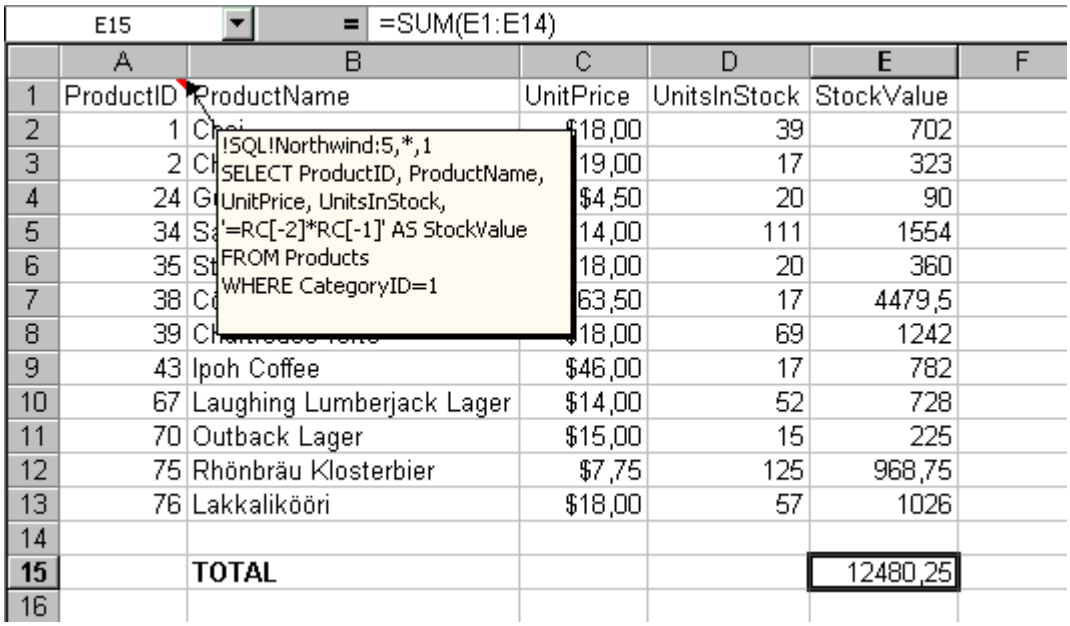
6. Controlling Returned Rows And Columns

Returning Variable Numbers Of Rows

In the ExcelSQL query editor, you are can specify the number of rows (and columns) your query returns. However, often the number of rows returned is not known beforehand. This may be a problem if you want to include some data below your query results. To circumvent this problem, you can use an asterisk (*) as the number of rows to return. For example:



When this query is run, it will insert rows into the Excel sheet for each row returned by the query, except the first row. The result will look as follows:



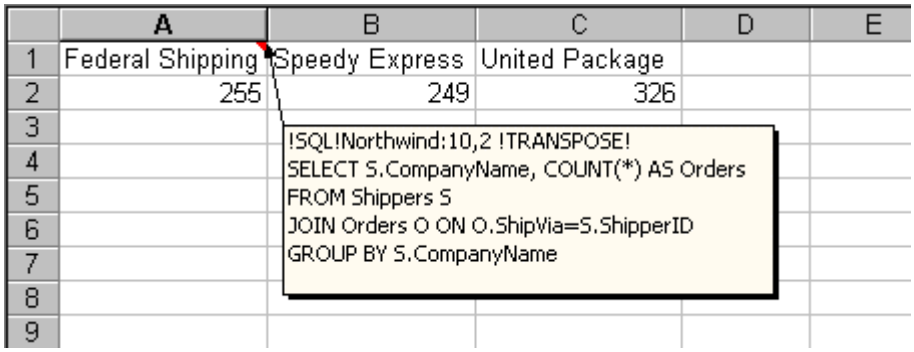
When the query is run again, previously returned rows are deleted first. This way you can place sum formulas directly below the query results, as done in the example. Note, however, that you may not place any other data on the same rows with the query (that is, alongside with the query data), because that data would be deleted when the query rows are deleted!

When a query that returns variable numbers of rows is run, it will add a new named range onto your worksheet. This range has a name of the format: Z_ExcelSQL_cell where "cell" is the cell in which the query resides (Z is there to make it sort last in the list of names). If you move your query or delete this name, ExcelSQL will no longer be able to delete the correct rows before running the query. In this case, you must first delete any previous results manually.

If the query where the cell resides is named, the Z_ExcelSQL range added by ExcelSQL will contain the name of the cell. For example, if the query cell is named as QueryCell, the range will be called Z_ExcelSQL_QueryCell. This means that if the query moves to another cell, ExcelSQL can still remove the correct rows when running the query. For this reason, it is highly recommended that queries containing cells be named.

Transposing Output Data

Normally, columns returned from the database become columns in Excel, and rows returned from the database become rows in Excel. Sometimes, however, you may want to reverse this so that columns become rows, and rows become columns. For example:



Note that Maximum columns and Maximum rows value are expressed in rows and columns in Excel. Thus, when you toggle the Transpose output data setting, you should swap the Maximum columns and Maximum rows values.

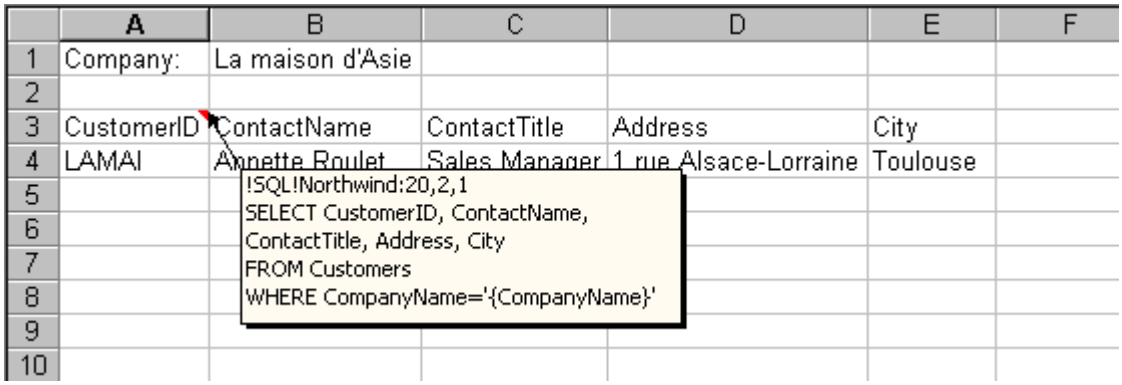
When the Transpose mode is on, all dimensions are reversed. For example, if you specify an asterisk (*) as the number of columns, new columns are added as needed (normally rows are added, see section "Returning Variable Numbers Of Rows").

Some ExcelSQL functionality is not available when the Transpose mode is on. Grouping and crosstabbing is not available in Transpose mode. For more information on grouping and crosstabbing, see section "Grouping And Crosstabbing Data".

7. Special Considerations With SQL Statements

Single Quotes In SQL References

If the cells you reference contain single quotes, they will most probably be mis-interpreted in the SQL text. Take for example the following query:



When the CompanyName reference is substituted into the ExcelSQL query, it would look as follows:

... WHERE CompanyName='La maison d'Asie'

Clearly, this would cause an SQL error, because there are non-matched quotes. However, ExcelSQL will correct this situation, and duplicate the single quote in the referenced value:

... WHERE CompanyName='La maison d''Asie'

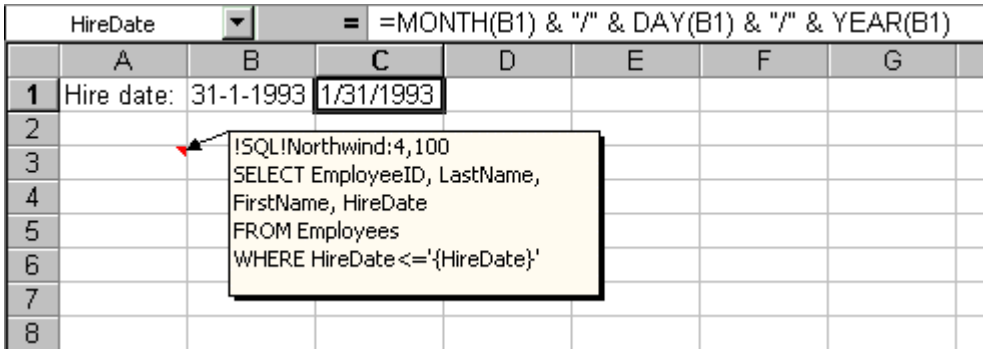
ExcelSQL always substituted single quotes, if the reference is enclosed in single quotes. This automatic substitution works only if the single quotes are **immediately** before and after the reference (opening and closing curly brackets). If this is not the case, you will have to take care of quote substitution yourself. For example, Excel's SUBSTITUTE worksheet function could be used for this.

Note that if you use double quotes in your SQL statements (which is not ANSI-compatible anyway), automatic substitution will not happen.

Date Formats

Always remember that values are passed to the SQL database as they appear on the worksheet. In some cases, however, the appearance can change from one computer to another, for example, date formats change depending on the computer's international settings. Thus, special care must be taken when dates are referenced from ExcelSQL queries.

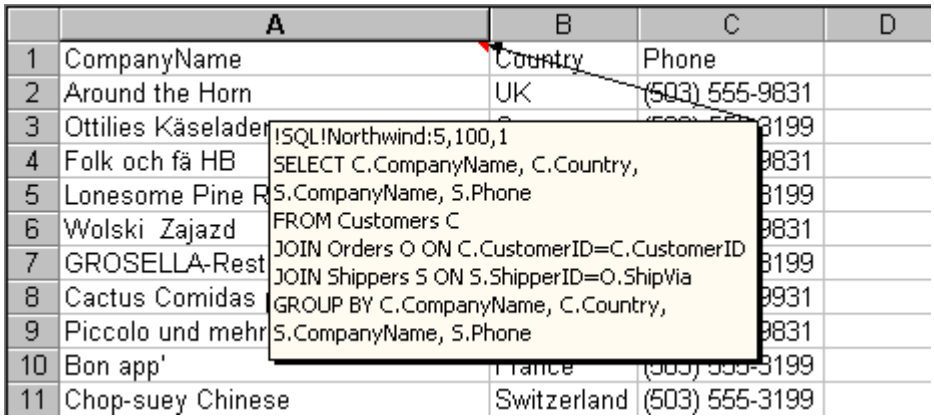
Following is one possible solution of how to handle dates:



Here the date (entered in ANSI format in cell B1) is explicitly converted to the mm/dd/yyyy format.

Problems With Duplicate Column Names

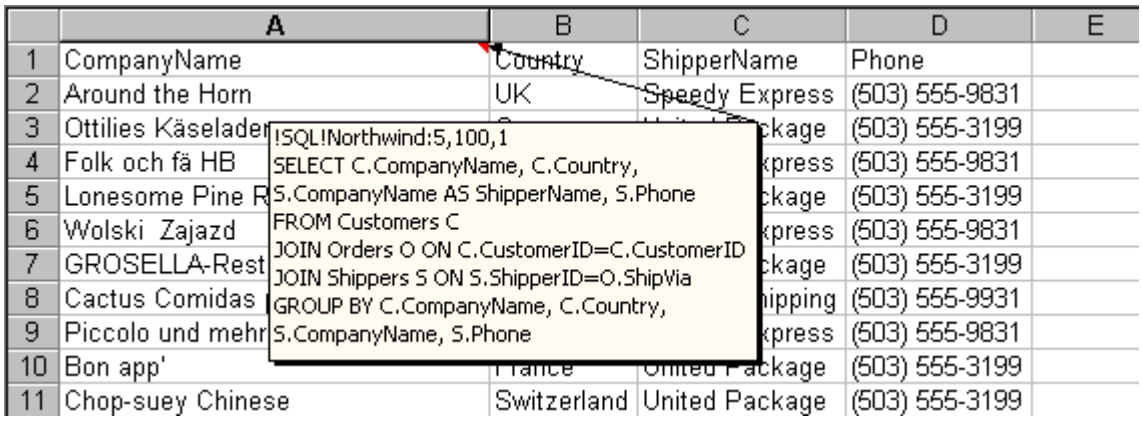
Sometimes ExcelSQL fails to return all columns specified in the query. For example:



	A	B	C	D
1	CompanyName	Country	Phone	
2	Around the Horn	UK	(503) 555-9831	
3	Ottilies Käselader		3199	
4	Folk och fä HB		9831	
5	Lonesome Pine R		3199	
6	Wolski Zajazd		9831	
7	GROSELLA-Rest		3199	
8	Cactus Comidas		9931	
9	Piccolo und mehr		9831	
10	Bon app'	France	(503) 555-3199	
11	Chop-suey Chinese	Switzerland	(503) 555-3199	

This query is supposed to return the companies and which shippers they have used. However, the CompanyName field is the the same in both tables, which causes ExcelSQL to process only the first one of them (this is a limitation in Data Access Objects, DAO).

You can make this query work by defining an alias for the other CompanyName column:

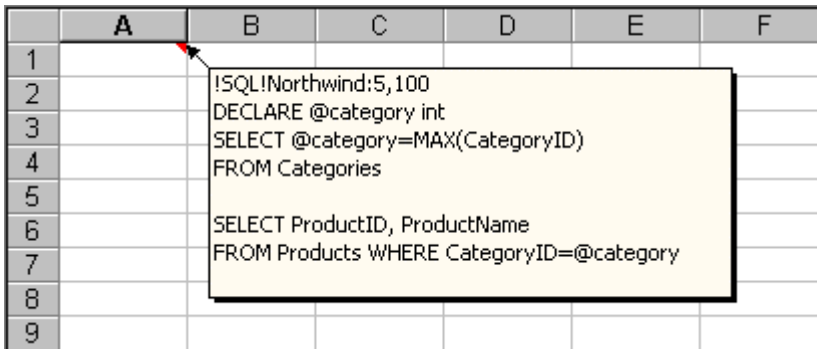


	A	B	C	D	E
1	CompanyName	Country	ShipperName	Phone	
2	Around the Horn	UK	Speedy Express	(503) 555-9831	
3	Ottilies Käselader		ckage	(503) 555-3199	
4	Folk och fä HB		xpress	(503) 555-9831	
5	Lonesome Pine R		ckage	(503) 555-3199	
6	Wolski Zajazd		xpress	(503) 555-9831	
7	GROSELLA-Rest		ckage	(503) 555-3199	
8	Cactus Comidas		ipping	(503) 555-9931	
9	Piccolo und mehr		xpress	(503) 555-9831	
10	Bon app'	France	United Package	(503) 555-3199	
11	Chop-suey Chinese	Switzerland	United Package	(503) 555-3199	

Queries That Contain Several SELECTs

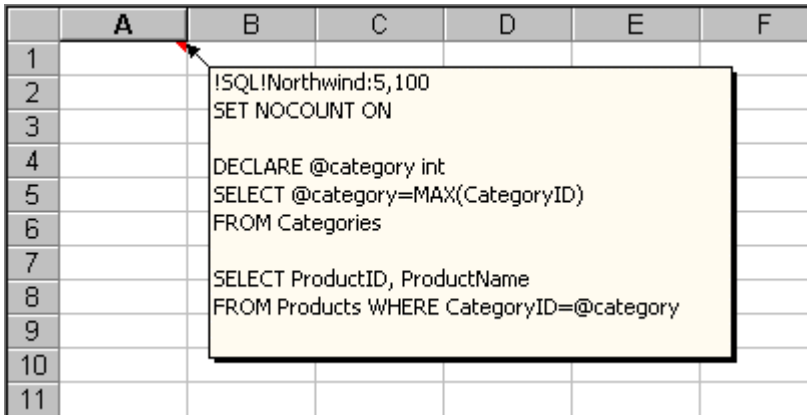
ExcelSQL requires that all queries return a recordset. ExcelSQL cannot process several recordsets returned by a query; only the first recordset is processed.

If the first recordset is empty, ExcelSQL will report the error "Object invalid or no longer set". A common example is a variable assignment:



This query will report the error "Object invalid or no longer set", because ExcelSQL cannot process the results of the first query. The first query in this case is the SELECT which sets the @Category variable.

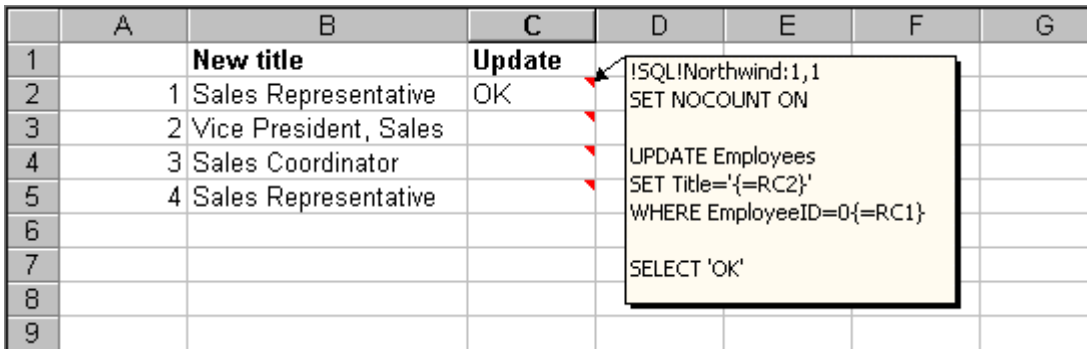
To make this query work, modify it as follows:



Note: The SET NOCOUNT ON clause is specific to Microsoft SQL Server. Other SQL products may work differently.

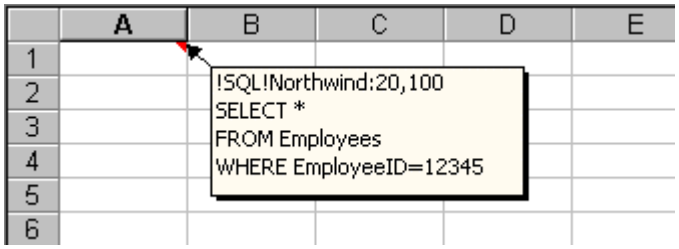
Queries That Return No Values

Some perfectly valid SQL statements, for example UPDATEs, do not return anything. As described in the section "Queries That Contain Several SELECTs" above, ExcelSQL only produces the first recordset returned. UPDATE queries return an empty recordset, and ExcelSQL will thus report the "Object invalid or no longer set" error. To suppress the error message, do a "dummy" select in addition to the UPDATE. For example:



SET NOCOUNT ON causes the UPDATE clause not to return a recordset, and the last SELECT will thus be visible to ExcelSQL.

Note that a query whose WHERE conditions does not meet any records, is totally different from the case discussed here. For example:



This query will work perfectly, but it will not return anything.

8. Miscellaneous Information

Useful Excel settings

To see a clear indication of cells that contain cell notes, you should have the cell note indicator on. This can be set from Options (in the Tools menu) under the View tab.

To make it easy to edit your SQL statement, turn In-Cell Editing off (Options, View tab). When In-Cell Editing is off, you can edit cell notes by double-clicking cells.

Using ExcelSQL macros

If you want execute all SQL statements in a workbook in one go, run the macro "ExcelSQL.xla!ExecuteAllSQLStatements". You can also add this macro as a button into your workbook. Note, that generally it is not a good idea to run all the SQL statements at once, because it may take a very long time.

When you select Execute Selected SQL Statements from the ExcelSQL menu, the macro "ExcelSQL.xla!ExecuteSelectedSQLStatements" gets executed. You can run this macro from your own macros, for example to cause some queries to run automatically when your workbook is opened. If you want to use the ExcelSQL macros from your own macros, you must set a reference to Excelsql.xla first (select References from the Tools menu).

9. ExcelSQL Cell Comment Syntax

Syntax Of The First Line

The complete ExcelQL syntax for the first line in the cell note is defined as follows. Brackets ([]) mean optional values, vertical bars (|) separate interchangable parts, and parenthesis (()) are used to group related items. The vertical bar before the **connectstr** parameter is part of the syntax.

!SQL(**!**[_]**DSN**[**$driver**][**|connectstr**])|[**=ref**]**:maxx,maxy**[**,headers**] [**!TRANSPOSE!**]

!SQL	This is the string by which ExcelSQL recognizes the cell note as an SQL query. Regular cell notes are skipped, and not processed by ExcelSQL.
DSN	Specifies the datasource for the query. You can specify any datasource that has previously been registered via the ODBC Administrator (32-bit). Prefix the datasource name with an underscore to specify that no login information should be asked from the user (in case the datasource asks for such information itself or just does not need a login). If you include an underscore, enclose the datasource definition in double quotes, for example: "_MyDSN".
driver and **connectstr**	You can register your own datasource by specifying **driver** and one or more connection string entries (**connstr**) separated by vertical bars (	). If the datasource is not registered, and you include **driver**, the datasource will be registered on the computer with the connection string you specify.
ref	Specifies an absolute or a relative reference to a cell whose cell note will be used as the query to execute for this cell. If you use this argument, you may not specify any other settings (maxx, maxy or the SQL query) in the cell note.
maxx	Specifies the maximum number of columns to return. Asterisk (*) means that columns are to be inserted/deleted as needed (can only be used when the transpose mode is on).
maxy	Specifes the maximum number of rows to return. Specify a very large value (16000, for example) if you want to allow returning as many rows as possible. That the area defined by **maxx** and **maxy** is cleared before executing the query. Asterisk (*) means that rows are to be inserted/deleted as needed (can only be used when the transpose mode is off).
headers	Specify 1 to insert column headers from the query results. Omit this parameters or specify 0 to not insert any headers.
!TRANSPOSE!	If the SQL statement returns data in a "vertical" format but you would like to get it horizontally onto your worksheet, add this keyword to the cell note. Since the query will now return more columns than rows (typically), you must probably swap the **maxx** and **maxy** values.

10. Troubleshooting

Information About Installation Problems

ExcelSQL relies heavily on the database functionality provided by Microsoft Data Access Objects (DAO). Because of different Excel and DAO versions, installing ExcelSQL may sometimes be tricky. This section describes the technical details behind these problems, and suggests solutions to the most common problems.

There are two versions of the ExcelSQL.xla file available: one for Excel 97 and one for Excel 2000. The only difference between these versions is the version of DAO (Data Access Objects) they use. Excel 97 version uses DAO 3.5 and Excel 2000 version uses DAO 3.6. As long as the correct DAO library is installed on the computer, they can be used on either version of Excel, that is, ExcelSQL for 97 can be used on Excel 2000, and ExcelSQL for 2000 can be used on Excel 97.

For example, if you have upgraded Excel 97 to Excel 2000, you will have both versions of DAO installed. If you have not installed any Office 2000 software, you must use the 97 version of ExcelSQL. If you have not installed any Office 97 software, you must use the 2000 version of ExcelSQL.

There is an additional problem with Excel 97 and DAO: In typical installation of Office 97, DAO is **not** installed. Thus, ExcelSQL will not work on such an installation. You should ensure that all computer where ExcelSQL is to be used have DAO installed.

Common Installation Problems

Problem:
"Can't find project or library" error is displayed, and Excel prompts for an ExcelSQLProject password when ExcelSQL is installed.

Description:
The DAO version used by ExcelSQL is not found. Make sure you use the correct version of ExcelSQL (97 or 2000), and that DAO is installed. You can tell whether DAO is installed by looking at the directory C:\Program Files\Common Files\Microsoft Shared\DAO. DAO 3.5 (installed by Excel 97 and used by ExcelSQL for 97) is contained in file Dao350.dll, and DAO 3.6 (installed by Excel 2000 and used by ExcelSQL for 2000) is contained in file Dao360.dll.

Problem:
"Compile error in hidden module: ExcelSQLModule" error is displayed.

Description:
The DAO version used by ExcelSQL is not found. See the above problem for more information.

Problem:
"ActiveX component can't create object" error is displayed when a query is run.

Description:
The DAO library used by ExcelSQL is not found. The cause may be same as described in the first problem above, but it may also be caused by other causes. One such cause is that DAO license information is missing from the Windows Registry. To restore missing DAO license information, the license information can be entered into the Registry manually. For instructions on how to do this, see the Microsoft Knowledge Base article number Q240377. You can search for this article on the Microsoft Web site at http://support.microsoft.com.