

Using Gsharp with Microsoft Office

Course Notes & Exercises

John Stringer
Advanced Visual Systems

UAUUG Technical Briefing Day
Tuesday 6th February 2001
Manchester

Gsharp 3.2

Same Interface as UNIX version

- Resize Browser
- Cut & Paste in Browser
- Resize DataManager
- DataManager supports folders
- Resize ScriptBuilder
- Modular Resource Editors with Apply Button
- Implemented input_file, input_selection

Enhancements

- Support for double precision
- More symbols for scatter plot
- Excel Spreadsheet reader and paste Excel data from clipboard
- New GUI object: combobox.
- Flat icons and support for separators on the toolbar
- Implemented input_dataset
- Implemented filelist(“*.mcd”)
- Support for TrueType fonts

Schedule

- Internal Alpha testing just begun.
- Beta-testing in about 1 month – to join e-mail support@avsuk.com
- Release April, May, ...

Available Methods

Getting data from Office to Gsharp

- Importing CSV (comma separated) files – *New in Gsharp 3.1*
- Importing Excel data files (*.xls) – *New in Gsharp 3.2*
- Paste an Excel selection directly into the Gsharp DataManager - *New in Gsharp 3.2*
- Using ODBC to connect to Microsoft Access (or Excel)

Getting data from Gsharp to Office

- Writing ASCII files
- Using ODBC to connect to Microsoft Access (or Excel)

Getting graphics from Gsharp to Office

- Copy bitmaps onto the clipboard – *New in Gsharp 3.1*
- Copying enhanced metafiles to the clipboard – *Hopefully new in Gsharp 3.2*
- Saving as image files such as GIF, TIFF, PNG and JPEG
- Saving as Encapsulated PostScript or CGM

Getting tables and text from Gsharp to Office

- Write HTML using fprintf or similar functions

Future methods

- Please let us know what you want – johns@avs.com

Cutting and Pasting

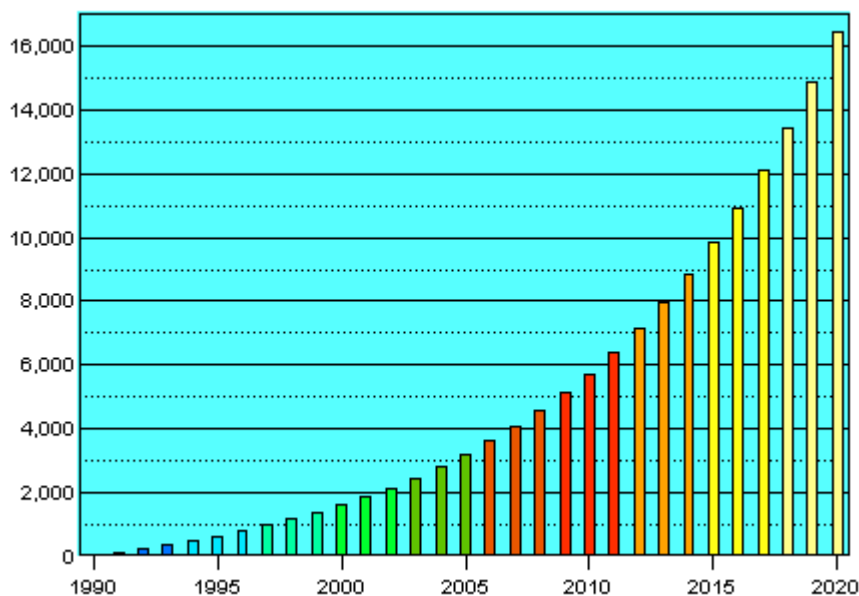
Cutting and Pasting is one of the fastest ways of moving data and graphics between Gsharp and Office.

In this exercise we will:

- Create a small spreadsheet in Excel.
- Cut and Paste the data into Gsharp
- Plot the data in Gsharp
- Paste the plot into PowerPoint.

It should be a very easy exercise that you will finish in no time and continue onto the next exercise bursting with confidence.

If all goes well you should finish with a plot like this:



It's probably quite easy to make a plot like this in Excel, but for this exercise I've concentrated on the cutting and pasting rather than trying to persuade you that Gsharp is better at plotting than Excel.

Exercise

1. Bring up the Windows Explorer (Windows-E) and create a folder for storing your working files. Create a shortcut on your desktop that will run `c:\uniras\7v2\bin\Gsharp.exe` starting in your working folder.
2. Start Excel.
3. Enter the following values:

	A	B	C	D
1	Interest:	10	Save	100
2				
3	Year	Add	Interest	Balance
4	1990.00	0.00	0.00	0.00
5	<code>=+A4+1</code>	<code>=+\$D\$1</code>	<code>=+\$D4*\$B\$1/100</code>	100.00

(Feel free to make your own spreadsheet if you prefer something a little more adventurous).

4. Copy line 5 to lines 6 to 20. Your spreadsheet should now look like this (please make sure that your numbers are not formatted as currencies nor include commas – these can't be pasted into the alpha version of Gsharp!)

Interest: 10 Save 100

Year	Add	Interest	Balance
1990.00	0.00	0.00	0.00
1991.00	100.00	0.00	100.00
1992.00	100.00	10.00	210.00
1993.00	100.00	21.00	331.00
1994.00	100.00	33.10	464.10
1995.00	100.00	46.41	610.51
1996.00	100.00	61.05	771.56
1997.00	100.00	77.16	948.72
1998.00	100.00	94.87	1143.59
1999.00	100.00	114.36	1357.95
2000.00	100.00	135.79	1593.74
2001.00	100.00	159.37	1853.12
2002.00	100.00	185.31	2138.43
2003.00	100.00	213.84	2452.27
2004.00	100.00	245.23	2797.50
2005.00	100.00	279.75	3177.25
2006.00	100.00	317.72	3594.97

5. Drag your mouse over cells A4 to D20 and press Ctrl-V. A stippled line should appear around your selection.
6. If Gsharp is not running – start Gsharp.
7. Bring up the DataManager. You will notice that the normally greyed out Excel icon is shining green.

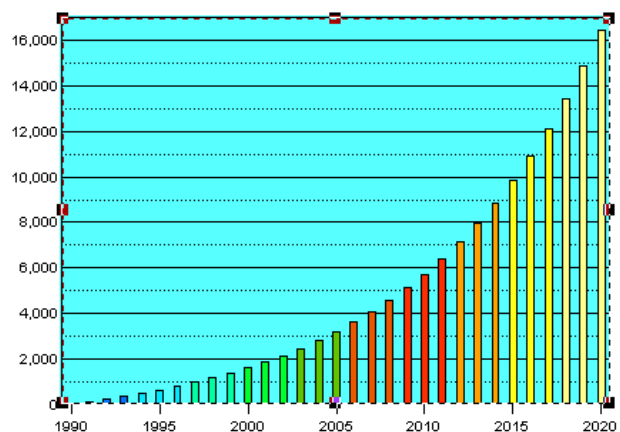
8. Click on the Excel icon and your selection will appear in the DataManager as T1, T2, T3, T4
9. Convert T1 into a date by typing the following command on the command line:

```
dates = todate(T1,1,1);
```

10. Create a bar graph with xData = "dates" and yData = "T4" and colorData = "T3". Turn on major and minor ticklines. Format the y axis labels as integers. The following GSL could be useful:

```
create Viewport page_1.viewport_1
( XuNbackgroundColor = 54,
  XuNfirstDiagonalPoint = (12,12) %,
  XuNframeWidth = 0.1 mm,
  XuNsecondDiagonalPoint = (96,94) %
);
create Domain page_1.viewport_1.domain_1
( XuN1st2DYAxisTicklines = "behind",
  XuN1st2DYAxisTickmarks = false
);
set page_1.viewport_1.domain_1.yaxis1
( XuNaxisLabelsDecimals = 0,
  XuNaxisLabelsPowerFactor = 0,
  XuNticklinesMajor = true,
  XuNticklinesMinor = true,
  XuNticklinesMinorStyle = "lineStyle1",
  XuNtickmarksMajor = false
);
create Graph page_1.viewport_1.domain_1.graph_1
( XuNcolorData = "T3",
  XuNgraphType = "bar",
  XuNxData = "dates",
  XuNyData = "T4"
);
```

11. Once your plot looks something like this press Ctrl-C. Your plot will be copied to the clipboard as a bitmap.
12. Start PowerPoint. Create a new slide and then press Ctrl-V. Your Gsharp plot will be added to the slide.



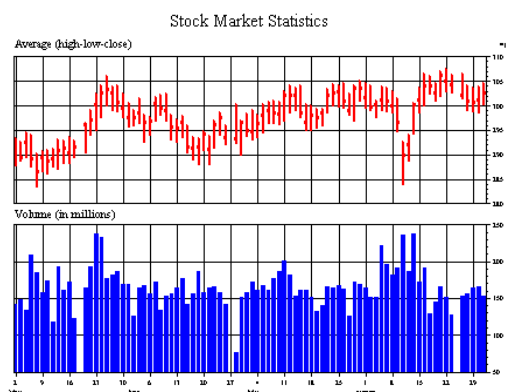
Creating HTML

HTML is useful for formatting output from Gsharp. Gsharp can create HTML files that can be used on the web or imported into Microsoft Word. HTML can be used to specify the layout of a report or to format our results into a table.

Gsharp can also create the images, which are then referenced by the HTML.

Exercise

We want to make a report in Microsoft Word based on the graphics and data created by the Gsharp Example, stocks.gsl



1. Start Gsharp and open up the ScriptBuilder. Paste in the following code.
2. Save your GSL code as stockhtml.gsl. Run it and make sure that stocks.gif is created correctly.
3. Add calls to fopen, fwrite and fclose to write a file called stocks.html. The HTML file must contain the picture stocks.gif and a table of the data created by the example (hi, lo, cls, vol and dates). You can use the guide to HTML include in the Resources section.

TIP: fwrite(f, "<TAG>" + vol + "<TAG>" + cls);
will write out one line for every member of vol/cls.
4. Once stocks.html is created without errors, read it into Word.
5. If you have time use the mask() function to remove any undefined rows from your table.

```
mymask = (vol<>undef);  
cls = mask(cls, mymask);  
hi = mask(hi, mymask);  
etc.
```

Resources

A Simple HTML File:

```
<HTML>
<HEAD><TITLE>Stock Exchange Example</TITLE></HEAD>
<BODY>
<H1>Output from Stock Exchange Example</H1>
<H2>The Graphics</H2>
  reference the picture now
<H2>The Data</H2>
  create a table of the data here
</BODY>
```

In HTML we reference an image like this:

```
<IMG src=stocks.gif>
```

and a table of data can be created like this:

```
<TABLE>
<TR><TH>Date<TH>Volume<TH>High<TH>Low<TH>Close
<TR><TD>02-MAY-1994<TD>140<TD>2932<TD>2879<TD>2887
<TR><TD>03-MAY-1994<TD>146<TD>2927<TD>2887<TD>2893
<TR><TD>04-MAY-1994<TD>132<TD>2945<TD>2895<TD>2925
<TR><TD>05-MAY-1994<TD>208<TD>2940<TD>2875<TD>2890
<TR><TD>06-MAY-1994<TD>184<TD>2902<TD>2835<TD>2865
...
</TABLE>
```

In the above exercise you will need the following Gsharp functions:

fopen - *open file*

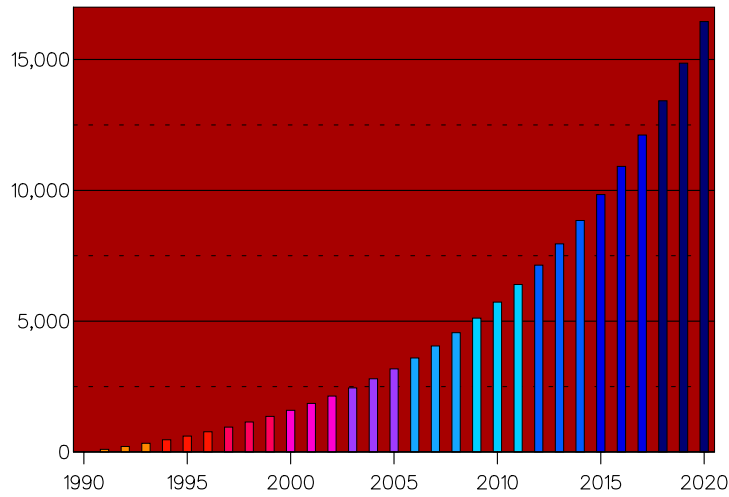
Syntax:	float fopen (float <i>filename</i> , string <i>mode</i>)
Description:	<p>Open <i>filename</i> with the specified <i>mode</i>.</p> <ul style="list-style-type: none">• The <i>filename</i> can include environment variables, e.g. \$UNIDIR• <i>mode</i> can be any of the values taken by the C function fopen. These include:<ul style="list-style-type: none">○ "r" - read○ "w" - write○ "a" - append, create if necessary○ "r+" - read or write○ "w+" - create new, then read or write○ "a+" - read or write, open at end• fopen returns a float identifier to the file which can be used with fprintf, fread, fwrite, fclose, etc.• The following identifiers are set up by Gsharp: 0 (stdin), 1 (stdout) and 2 (stderr).

fwrite - *write to a file*

Syntax:	float fwrite (float <i>fptr</i> , float <i>var1</i> , float <i>var2</i> , ...)
Description:	<p>Write data to the file specified by <i>fptr</i>.</p> <ul style="list-style-type: none">• Each variable is written in turn• Floats are written in binary format• Strings, dates and times are written as ASCII strings.• The written file is most easily read by fread.• The function returns the total number of bytes written. <p>e.g. RecordSize = fwrite(f, Title, Name, Occupation, Address, Age, Sex, Salary);</p>
Code sample:	<pre>float f; f = fopen("newfile.txt", "w"); fwrite(f, "My header"); fwrite(f, "Line"+(1:10)); fclose(f);</pre>

Output from Stock Exchange Example

The Graphics



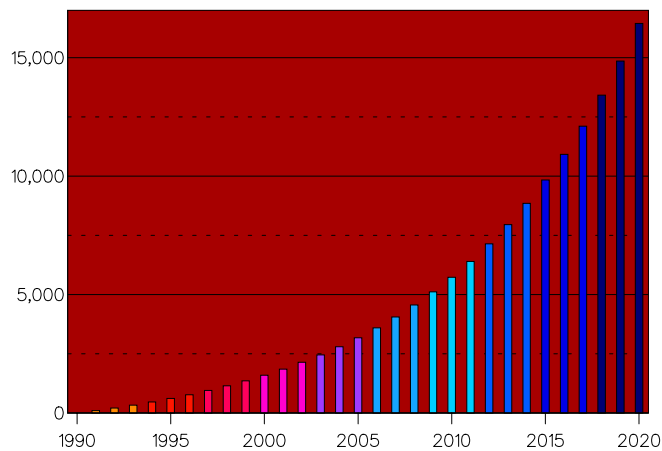
The Data

Date	Volume	High	Low	Close
02-MAY-1994	140	2932	2879	2887
03-MAY-1994	146	2927	2887	2893
04-MAY-1994	132	2945	2895	2925
05-MAY-1994	208	2940	2875	2890
06-MAY-1994	184	2902	2835	2865
09-MAY-1994	156	2906	2865	2894
10-MAY-1994	172	2910	2860	2885
11-MAY-1994	116	2912	2875	2890
12-MAY-1994	192	2930	2871	2908
13-MAY-1994	160	2928	2883	2912
16-MAY-1994	170	2935	2883	2902
17-MAY-1994	122	2929	2895	2915
19-MAY-1994	162	2965	2905	2960
20-MAY-1994	192	2990	2940	2970
23-MAY-1994	236	3025	2950	3000
24-MAY-1994	232	3040	2975	3025
25-MAY-1994	175	3060	3000	3032
26-MAY-1994	180	3039	2990	3025

Time to Spare

If you have time to spare please feel free to try out any of the following tasks that you think might be useful:

- Create a spreadsheet in Excel. Save it. Try and read it with the Gsharp Excel (*.xls) reader.
- Create a spreadsheet in Excel. Save it as a comma separated file. Try and read it with the Gsharp Comma Separated File (*.csv) reader.
- Use the File/Output Image to make an Encapsulated PostScript file and a Binary CGM. Which one looks better in Word? Note that the images can be resized and will still print at a high resolution.



ODBC

Gsharp can connect to an ODBC Data Source using SQL embedded into the Gsharp Script Language.

If you want to access an Access database – you must first define it as a data source. Here is how to do it:

Creating an ODBC Data Source

Use the following instructions to connect Gsharp to your Microsoft Access database, uauug.mdb

1. **From the Control Panel - select ODBC Data Sources.** The ODBC Data Source Administrator should appear
2. Make sure that User DSN is selected. **Click on the Add ... button.**
3. **Select the appropriate driver** - in this case Microsoft Access Driver (*.mdb) and then **click on Finish**
4. **Give the data source a name** e.g. tbd
5. **Click on Select ...** and then enter the appropriate filename.
6. Click on OK.

You can connect to the database with the following command:

```
exec sql connect 'admin/@uauug';
```

Exercise

We want to build a small application for processing and visualising web server logs.

1. Start Microsoft Access and create a new blank database called **tbd.mdb**
2. Create a table called **Master** which contains the following fields:

Filename	Text	100
Date	Date/Time	Short Date
Time	Date/Time	Long Time
Page	Text	100
Code	Text	10
Bytes	Text	10
IPAddress	Text	20

N.B. Due to a bug in the Gsharp 3.2 alpha version it is not possible to use numbers with ODBC. Numbers should be converted to strings before passing with `valustr()` and back to numbers after retrieving with `strvalue()`.

3. Save your database and exit Access.
4. Add tbd.mdb as an ODBC data source as shown above.

5. Start Gsharp as normal.
6. Open up the ScriptBuilder and switch on Command Logging
7. File/Open set file type to ASCII (*.dat)
8. Click on options. Set Datasets to **Lines** and Type to **string**. Click on OK.
9. Select a log file. Click on OK. The file is read into a dataset called Lines.

We now want to process this text dataset and create arrays corresponding to each of the fields in our database: Filename, Date, Time, File, Code, Bytes, IP address. For speed we have to avoid using the GSL for loop.

10. Bring up the DataManager. Type the following commands into the command line of the DataManager and use the DataEditor to look at the result of each command:
 - numLines = size(WORK.Lines);
 - Filename = repeatx("sportsvl20001223.dat", numLines);

```
209.1.13.231 - - [23/Dec/2000:00:19:47 -0500] "GET / HTTP/1.0" 200 949
64.208.37.30 - - [23/Dec/2000:00:26:25 -0500] "GET /robots.txt HTTP/1.0" 404 204
```

We will tokenise each line using spaces as delimiters.

- WORK.Lines = tokenize(WORK.Lines," ");
- WORK.Lines = reshape(WORK.Lines, 10, numLines, 1);
-
- IPAddress = WORK.Lines[1];
- IPAddress = list(IPAddress);
- Date = list(WORK.Lines[4]);
- File = list(WORK.Lines[7]);
- Code = list(WORK.Lines[9]);
- Bytes = list(WORK.Lines[10]);
-
- Date = tokenize(Date-"[";":"");
- Date = reshape(Date, 4, numLines, 1);
- Hour = strvalue(list(Date[2]));
- Minute = strvalue(list(Date[3]));
- Second = strvalue(list(Date[4]));
- Date = dateformat1(list(Date[1]));
- Time = totime(Hour, Minute, Second);

11. Go back to the ScriptBuilder turn off command logging. Edit your code so that it looks like this:

```
function ReadLogFile(string filename)
float numLines, Hour, Minute, Second;

if exists(WORK.Lines) destroy WORK.Lines;
import_ascii(filename,,,,,"WORK.Lines","text");

numLines = size(WORK.Lines);
Filename = repeatx(filename, numLines);
```

```

WORK.Lines = tokenize(WORK.Lines," ");
WORK.Lines = reshape(WORK.Lines, 10, numLines, 1);

IPAddress = list(WORK.Lines[1]);
Date = list(WORK.Lines[4]);
File = list(WORK.Lines[7]);
Protocol = list(WORK.Lines[8]);
Code = list(WORK.Lines[9]);
Bytes = list(WORK.Lines[10]);

Date = tokenize(Date-"[";":"");
Date = reshape(Date, 4, numLines, 1);
Hour = strvalue(list(Date[2]));
Minute = strvalue(list(Date[3]));
Second = strvalue(list(Date[4]));
Date = dateformat1(list(Date[1]));
Time = totime(Hour, Minute, Second);

if exists(WORK.Lines) destroy WORK.Lines;

endfunction

```

12. Reset the DataManager. Run our script in the ScriptBuilder and then try:
ReadLogFile(input_file("Choose log file", "*.dat"));
13. Look in the DataManager to make sure that our Datasets have been created correctly and then ...

```

EXEC SQL INSERT INTO Master VALUES (:Filename, :Date, :Time, :File, :Code,
:Bytes, :IPAddress);

```

14. Start Access again and check that the data has been successfully entered into our Access table.
15. Start a text editor, such as notepad and paste in our ReadLogFile function. Add the following function to perform the ODBC commands:

```

function SaveLogFile()

if not exists(WORK.Connected) EXEC SQL CONNECT 'admin/@tbd';
WORK.Connected = true;

EXEC SQL DELETE FROM Master WHERE Filename = :filename;
EXEC SQL INSERT INTO Master VALUES (:Filename, :Date, :Time, :File, :Code, :Bytes, :IPAddress);
EXEC SQL COMMIT;

endfunction

```

16. Finally add the following code to use the two functions. Save the file as logs.gsl:

```

for filename in filelist("sportsv1*.dat")
echo("Reading "+filename);    ReadLogFile(filename);
echo("Saving "+filename);     SaveLogFile();
endfor

```

17. Run logs.gsl and examine the records in Access.

We would like to build a small application for processing web server logs. We'll start by creating the File menu for our application.

```
function CreateLogMenus()

if exists(gsharp_1.menubar.File2) destroy gsharp_1.menubar.File2;
create Menu gsharp_1.menubar.File2
( XuNguiLabel = "File"
);
create Button gsharp_1.menubar.File2.Add
( XuNguiLabel = "Add Log File ...",
  XuNguiCallback = "AddLogFileCB"
);
create Button gsharp_1.menubar.File2.Reset
( XuNguiLabel = "Remove all log files",
  XuNguiCallback = "ResetLogFilesCB"
);
endfunction

function AddLogFileCB(string o, string d, float r)
string filename;
filename = input_file("Enter log file", "r", true, "*.dat");
ReadLogFile(filename);
SaveLogFile(filename);
endfunction

function ResetLogFilesCB(string o, string d, float r)
if not exists(WORK.Connected) EXEC SQL CONNECT 'admin/@tbd';
WORK.Connected = true;
EXEC SQL DELETE FROM Master;
endfunction
```

Finally we want to build a dialog for probing our data.

Appendix A: ODBC - SQL Syntax

It is possible to embed SQL statements in a Gsharp script language program. This allows easy access to data stored in an external relational database such as Oracle. The syntax of the SQL statements conforms to the ANSI X3.1681989 with a number of exceptions.

- The embedded SQL statements is only available when the Gsharp Database Option has been licensed. If the option is not licensed, then execution of an SQL statement will result in an error message and Gsharp will abort the script language program.
- There is no support for cursors, dynamic SQL or the Data Definition Language part of SQL. Instead of cursors, Gsharp provides the ability to use host arrays. Gsharp also allows you to embed the SELECT, INSERT, UPDATE and DELETE SQL statements.
- Set operations are now supported in select statements and subqueries. The Oracle operations UNION [ALL], INTERSECT and MINUS are all supported by the GSL interpreter, but not all relational databases will support them.

Using Embedded SQL

An SQL statement is introduced by the keywords EXEC SQL and is terminated by the semicolon.

```
if a < 10 then
  EXEC SQL SELECT y_data INTO :y FROM experiment_data
  WHERE threshold > :a;
  echo("Data collected");
endif
```

Whenever the script language parser sees the EXEC SQL keywords, it switches into SQL mode and understands SQL statements. Gsharp's script language keywords are no longer reserved unless they are reserved in SQL as well.

Variables and Keywords

The following guidelines apply to using variables and keywords in SQL statements.

- Unlike Gsharp keywords, SQL keywords can be entered both in upper and lower case. For example, the following two statements are both acceptable.

```
EXEC SQL SELECT yData INTO :y FROM myData WHERE threshold > :a;
exec sql select yData into :y from myData where threshold > :a;
```

- The dollar sign (\$) and the # are valid characters in SQL data names. (This is not the case with Gsharp variable names.)
- You must use host variables to communicate with SQL.
A host variable is formed by adding a colon in front of a Gsharp variable

name. For example, if myData and yData are Gsharp variables, you would refer to them as :myData and :yData when using them in SQL statements.

- SQL strings are always enclosed in single quotes. You cannot use strings starting and ending with double quotes as SQL literals. However, strings delimited by double quotes are used in SQL to explicitly name table and columns which contains spaces, reserved keyword, or is in mixed case. Thus, you can write SQL statements like:
 - EXEC SQL SELECT "MyVar" FROM "SELECT"
WHERE "Strange NAME" IS LIKE '_YO_';

Note that `_YO_` is a literal string and "MyVar", "SELECT" and "Strange NAME" are actual objects in the database. This use is greatly discouraged by database vendors. Beware that if you use host variables containing strings, it doesn't matter if the string was initialized using single or double quoted strings.

Commenting Your SQL Statements

You can use either ANSI style or C style comments in SQL statements.

The following example demonstrates the ANSI style comment. Note that the ANSI comment extends to the end of the line.

```
EXEC SQL UPDATE tabl SET num = :x Set num to a calculated  
value  
WHERE c = 123;
```

The following example demonstrates the C style comment.

```
EXEC SQL UPDATE tabl SET num = :x/* Set num to a value */  
WHERE c = 123;
```

IMPORTANT: You cannot use the # character to initiate a comment.

Data Definition Language

GSL now supports part of the data definition language of SQL. It is now possible to create, alter, and drop tables and views. The syntax for performing these tasks is shown below.

```
EXEC SQL CREATE TABLE table (  
column datatype opt_constraints [, ...] );  
EXEC SQL ALTER TABLE table  
[ ADD (  
column datatype opt_constraints [, ...]  
) ]  
[ MODIFY (  
columns opt_datatype opt_constraints [, ...]  
) ];  
EXEC SQL DROP TABLE table;
```

Where table is the table name optionally including a schema, column is a column name, datatype is a valid data type, opt_constraints is optional column constraints, and opt_datatype is an optional data type.

Valid data types include:

```
CHAR, CHARACTER, CHAR(n), CHARACTER(n)
CHAR VARYING(n), CHARACTER VARYING(n)
NUMBER, NUMBER(p), NUMBER(p,s)
NUMERIC, NUMERIC(p), NUMERIC(p,s)
DECIMAL, DECIMAL(p), DECIMAL(p,s)
DEC, DEC(p), DEC(p,s)
INT, INTEGER, SMALLINT
FLOAT, FLOAT(b), DOUBLE PRECISION, REAL
```

Valid constraints include:

```
EXEC SQL CREATE VIEW view AS subquery;
EXEC SQL DROP VIEW view;
```

Where a subquery is a select statement with no ORDER BY clause.

Note: You must have the appropriate permissions in the database system in order to create, alter, or drop tables and views.

Schema Definition Statement

GSL supports the schema definition statement.

```
EXEC SQL CREATE SCHEMA AUTHORIZATION user
[CREATE TABLE command]
[CREATE VIEW command]
[GRANT command];
```

The CREATE TABLE, CREATE VIEW, and GRANT command are valid statements providing they are issued as part of the CREATE SCHEMA command.

Connecting to the Database

Oracle and most other relational databases require that you identify yourself to access the data in the database. Gsharp uses the same method and syntax as does Oracle. To connect to the database use the CONNECT statement, which is defined as:

```
EXEC SQL CONNECT literal|:hostvar [IDENTIFIED BY
literal|:hostvar];
```

The following code demonstrates how you can automate the process of connecting to a database. The code prompts for a user name and password and then uses that information to connect to the database:

```
/* Log on to database */
user = input_text("Enter username");
pwd = input_text("Enter password");
```

```
exec sql connect :user identified by :pwd;
```

If the database refuses the connection, Gsharp aborts the current program.

Retrieving Data

The SELECT statement is used to retrieve data from the database. A query of the database may result in one or more rows each containing one or more columns. To use the data in Gsharp, each column must refer to a Gsharp host variable. This reference is established using the INTO clause. Here is an example of a query, retrieving four columns:

```
EXEC SQL SELECT x, y, z, v INTO :x, :y, :z, :v FROM  
coords;
```

If the coords table contains the following values:

```
x y z v  
10.4 11.5 5.3 20.45  
11.4 10.6 3.4 15.67  
9.5 6.3 0 0
```

The SELECT will result in four real datasets each containing three values. This is an exception to normal embedded SQL, where you usually retrieve each row separately using a cursor.

For further explanation of the SQL SELECT statement refer to the ANSI standard document. Note that you cannot use Oracle extensions to the SQL language.

If you enter the SQL SELECT statement without the INTO clause, Gsharp will display the retrieved rows in the message area.

IMPORTANT: If you include a WHERE clause, all host variables in the WHERE clause must be scalars.

Entering Data into the Database

You store data in a database using the INSERT statement. You can use Gsharp datasets as well as literals in the VALUES clause of the INSERT statement:

```
EXEC SQL INSERT INTO coords VALUES ( :x, :y, :z, :v );  
  
EXEC SQL INSERT INTO coords VALUES ( :x, :y, 1.0, 0.0 );
```

If one or more Gsharp dataset is an array, they must be of equal length. If arrays are given, an INSERT statement is executed for each data value in the array. If scalars and arrays are specified together, the scalar is duplicated for each inserted row.

The example below demonstrates using arrays and scalars to populate an empty table, TDATA:

```

TDATA:
NUMBER NUM
CHAR(10) TXT
NUMBER C

a = 1//2//3//4//5;
b = "This"//"is"//"a"//"database"//"program";

EXEC SQL INSERT INTO TDATA VALUES (:a, :b, 123);

EXEC SQL SELECT * FROM TDATA ORDER BY NUM;

```

The result of running the code above is shown below. Notice how the scalar, 123 is duplicated for each row in the table.

```

NUM TXT C
1 This 123
2 is 123
3 a 123
4 database123
5 program123

```

Updating Data in the Database

You can update rows using the UPDATE statement. You can use host variables in the UPDATE statement in both the SET clause and the WHERE clause. An example of an UPDATE statement is shown below.

```

EXEC SQL UPDATE tabl SET num = :x WHERE c = 123;

```

If one or more Gsharp datasets is an array, they must be of equal length. If arrays are given, an UPDATE statement is executed for each data value in the array. If scalars and arrays are specified together, the scalar is duplicated for each updated row.

Deleting Data from the Database

You delete rows from a database using the DELETE statement. You can use host variables in WHERE clause.

If one or more Gsharp datasets is an array, they must be of equal length. If arrays are given, a DELETE statement is executed for each data value in the array. If scalars and arrays are specified together, the scalar is duplicated for each deleted row.

The following statement deletes all rows from the TDATA table where C is equal to 123.

```

EXEC SQL DELETE FROM TDATA WHERE C = 123;

```

The statement below deletes all data from the TDATA table.

```

EXEC SQL DELETE FROM TDATA;

```

Handling SQL Errors

The GSL can now receive error information from the SQL. To work with this error information, you must create a variable, `SQLCODE`, in your current folder. This variable will receive the status of the SQL statement processed at the RDBMS.

- If `SQLCODE` is not defined, Gsharp will abort the runtime with an error message.
- If `SQLCODE` is defined, processing continues even if an error occurs (except for internal errors such as no memory available).

Committing or cancelling changes

Changes to the database are not committed automatically by Gsharp - you are responsible for managing the contents of `SQLCODE` and for either committing or cancelling the changes to the database with the `COMMIT` and `ROLLBACK` statements.

```
EXEC SQL COMMIT [WORK];  
EXEC SQL ROLLBACK [WORK];
```

Note: The `WORK` keyword is optional in GSL to be compatible with Oracle practice.

The following code demonstrates how to use the `SQLCODE` variable to manage the changes to your database.

```
SQLCODE=0;  
user = `Paul`;  
  
EXEC SQL DELETE FROM user_work WHERE user_name = :user;  
if not SQLCODE < 0 then  
    EXEC SQL DELETE FROM user_master WHERE user_name =  
:user;  
    If SQLCODE < 0 then  
        EXEC SQL ROLLBACK WORK;  
    else  
        EXEC SQL COMMIT WORK;  
    endif  
endif
```