Sculptor 2.7 Release Notes – 31 January 2018

Sculptor 2.7 is a major upgrade that allows existing Sculptor 2 applications to work in client/server mode using a Sculptor 6 server. It enables Sculptor 2 programs to read and update Sculptor files and SQL tables in proprietary SQL databases such as Access, SQL Server, Oracle, Postgres and MySQL across both local and wide area networks.

The Sculptor 2.7 client is available for most Unix and Linux platforms. Currently, it is not available for use in a DOS shell on Microsoft Windows. This would require a new screen module. A DOS version will only be developed if there is sufficient demand.

Sculptor 2.7 is supplied as a single, integrated package that includes Sculptor 6 for the same platform but see note about the resolution of name conflicts below. All utility programs (newkf, kfcheck, kfcopy, reformat, etc) are the Sculptor 6 versions.

Sculptor 2.7 cannot run programs compiled with earlier Sculptor 2 compilers. Existing Sculptor 2 programs must be recompiled using the Sculptor 2.7 version of scc. Sculptor 2.7 is source code upward compatible except that !informix is no longer supported.

Availability

The release version of Sculptor 2 module is available free of charge to Sculptor users who have a current Sculptor maintenance contract. Users with expired maintenance contracts must purchase a new licence in order to restart maintenance and use the Sculptor 2.7 module.

Name Conflicts

In a Sculptor 2.7 package:

scc is the Sculptor 2 version and **scc6** is the Sculptor 6 version.
sage is the Sculptor 2 version and **sage6** is the Sculptor 6 version.
menu is the Sculptor 2 version and **menu6** is the Sculptor 6 version.
There is no conflict with **sagerep** as the Sculptor 6 equivalent is named **srep**.
All other commonly named programs are the Sculptor 6 versions.

Note About Locking

The commands readu, nextu, prevu, findu, matchu read records without locks regardless of whether another user holds a lock. The commands read, next, prev, find, match read and lock records when using Sculptor files or using a SQL database that supports row level locks. Currently, only SQL Server is known to support row level locks.

If a file is open read only, the commands read, next, prev, find, match behave exactly the same as readu, nextu, prevu, findu, matchu for that file.

If a SQL database that does not support row level locks is being used, an RIU (record in use) trap is generated only if an attempt is made to read a record that has been read by another user inside an open transaction (begin ...). Otherwise, if more than one user reads the same record, then one user updates the record and later another user tries to update the same record without re-reading it, the second update fails and a UE (update error) trap is generated.

New Declarations in Sculptor 2.7

!database *dbname*

Declares a database.

dbname	A given name used to identify this database in the program.

All subsequent **!file** and **!cfile** declarations define files or SQL tables inside this database. Database attributes can be defined using the declarations below or at run-time by calling the function **setdbattr**() before the database is opened.

All the files in a database are closed when the program starts, regardless of their declaration. When the database is opened using the **opendb** command, files declared with **!file** are then opened automatically but files declared with **!cfile** must be opened using either the **open** or the **openfile** command.

!dbtype *type*

Defines the type of the current database. This must be **DBT_ODBC** or **DBT_SCULPTOR**, which are constants defined in the header file **sculptor2.h**.

!riu_timeout *value*

Sets the SQL query timeout to *value* seconds. This must be an integer constant. An attempt to read and lock record will fail and generate an **riu** trap after this number of seconds. For SQL Server, Sculptor creates stored procedures that do not use this value. For other SQL databases, Sculptor uses a sensible default and this attribute does not normally need to be specified and may be ignored.

!dblogfile "pathname"

By default, the Sculptor server logs errors in "$SCULPTOR\log\sculptor.log" and "$SCULPTOR\log\kfserver.log". This declaration causes errors related to this database to logged into the specified file instead.

!dbflags *flags*

Defines flags that alter the behaviour of the current database. These flags are defined in the header file **sculptor2.h**. Multiple flags must be combined with logical or. As this operator is not supported in Sculptor 2, the required value will need to be calculated and defined.

DBFL_CASE_SENSITIVE

Forces field and index names to be case sensitive. Sculptor names are always case sensitive and this flag is ignored for Sculptor databases. By default, SQL names are not case sensitive and this flag must be set if case sensitive names are required.

DBFL_OPTIMISTIC_LOCKING

Sculptor files use pessimistic locking, which means that a record cannot be read for update if it is locked by another user; an attempt to do so generates an **riu** trap. By default, Sculptor tries to implement the same behaviour when reading SQL tables. If this is not possible for a particular SQL database, Sculptor defaults to optimistic locking, which means that a locked record can be read for update but a subsequent update will fail if the record has since been updated by another user. This flag forces optimistic locking if supported by the SQL database. *This flag is not yet implemented for some SQL databases.*

DBFL_UNSIGNED_KEYS

Sculptor was originally implemented on early microcomputers with very limited amounts of memory and low processor speeds. For this reason, key fields are combined into a single byte string for comparison purposes, which means that in Sculptor negative integers are sorted unsigned and follow the positive integers. Most SQL databases sort negative integers first, which becomes an issue if SQL tables are required to sort in the same order as their equivalent Sculptor files. If this flag is set, Sculptor maps all integer key fields to binary fields in the SQL tables, forcing the SQL tables to sort in the same order as Sculptor. If this behaviour is only required for some integer keys, set the flag "p" for the required key fields in the Sculptor data dictionary and do not set this flag.

SQL Server only: Since binary fields are an inconvenient type, Sculptor adds a computed integer field to the table that is set automatically to the same value as the sort key. The integer field has the same name as the sort key field except that it is prefixed with **int_**.

DBFL_DRIVER_COMPLETE

Causes the ODBC driver to prompt for a connection string at run-time (if this feature is supported by the driver). Probably not useful in most applications but can be helpful for debugging connection strings.

!dbsource “*connection_string*”

For ODBC databases, *connection_string* specifies an ODBC connection string. This can be a DSN (data source name) or a full driver connection string. The syntax of the connection string depends on the ODBC driver and the SQL database but must start with a Sculptor server definition. The server clause is stripped and the rest of the string is then sent to the specified Sculptor server, which must be running on the target computer.

For Sculptor databases, *connection_string* specifies an optional Sculptor server name and an optional path prefix for all files in the database. Unless the files are being accessed locally, this must start with a Sculptor server name.

The / character is a universal separator in path names that works in system calls on Unix, Linux and Windows. It is not necessary to use the \ character on Windows. If \ is used in a quoted string then it must be escaped, each \\ being one backslash. Using a single \ by mistake is a common cause of difficult to diagnose errors in path names.

In the examples below, be careful to distinguish between the ; used in ODBC connection strings and the : used in Sculptor connection strings.

<u>Example ODBC Connection Strings</u>

“SCULPTOR_SERVER=localhost; DSN=mydsn”

“SCULPTOR_SERVER= mylinuxserver; DSN=mydsn”

“SCULPTOR_SERVER= mywinserver; DSN=mydsn”

“SCULPTOR_SERVER=MYWINSERVER;DRIVER=SQL Native Client;
SERVER={MYWINSERVER\\MSSQL2008};DATABASE=master;UID=myname;PWD=mypass”

<u>Example Sculptor Connection Strings</u>

“/home/myapp/mydata”

“C:/apps/myapp/mydata”

“mylinuxserver: /home/myapp/mydata”

“mywinserver:C:/apps/myapp/mydata”

!**end_database**

Defines the end of declarations that belong to the current **!database**. Multiple databases can be defined. All files declared outside a **!database** / **!end_database** pair are normal Sculptor files but these can also be opened in a Sculptor server by using the **openfile** command.

New and Modified Commands in Sculptor 2.7

opendb *dbname*

Opens the specified database and all files/tables in the database declared using **!file**. Files and tables declared using **!cfile** can be opened using the **open** or **openfile** command.

dbname	The name of the database (not quoted).

Note: An **err**=*label* trap may be added before the release version is finalised.

openfile *fileid* "*pathname*" *mode*

Opens a file/table in the specified mode with an optional, alternative pathname.

fileid	A file identifier or file number.
pathname	A full, alternative pathname. Specify "" to use the file's declared pathname and also for SQL tables where a pathname is not appropriate. The pathname can specify a Sculptor server. Example: "myserver: /home/myapps/mydata/myfile".
mode	The mode in which the file is to be opened. This must be **create**, **createbig, read** or **update**.

The maximum size of a Sculptor file is 4Gb or 16,777,215 records, whichever is reached first. If the **createbig** mode is used, these limits increase to 128Gb or 134,217,727 records. A big file cannot be used be Sculptor versions prior to 2.7 and 5.8.6.

Note: An **err**=*label* trap may be added before the release version is finalised.

close *fileid*

This command now sets a file to be logically closed but does not close its system files. Subsequent file access commands will return an error as before. This change in behaviour is needed to support client/server mode. The file needs to remain open on the server because it's the server and not the client program that now stores the find/match position and this is documented to be preserved when a file is closed and re-opened. The **closefile** command can be used to close a file and it's system files.

closefile *fileid*

Closes a file and all its system files. The find/match position is lost.

begin

Defines the start of a set of file updates to be processed as a single transaction. All records read for update remain locked until either **commit** or **rollback** is executed.

commit

Defines the end of a transaction. All records locked during the transaction are unlocked.

rollback

Defines the end of a transaction that is to be reversed as if it never happened. All records locked during the transaction are unlocked.

New Function in Sculptor 2.7

stat = **setdbattr**(*dbname, attrib, value*)

This function can be called to set a database attribute that has not been declared or to change the value of a declared attribute. Attributes must be set before the database is opened.

dbname	The name of the database (not quoted).
attrib	A text string defining the attribute to be set: “dbtype”, “dbflags”, “dbsource”, “dblogfile” or “riu_timeout”.
value	A text value for dbsource and dblogfile. An integer value for dbtype, dbflags and riu_timeout.

Return value is 0 if successful. Non-zero indicates a syntax error.

prev_value = **setfileattr(***fileid, attrib, value*)

This function can be called to set a file/table attribute at run time. Most file attributes start with default values when the file is opened, so unlike database attributes, the file must be open when this function is called.

fileid	The file identifier or number as used with file access commands.
attrib	A text string defining the attribute to be set: “batch_read”, “limitb”, “limitu” or “limit”.
value	An integer value.

Returns the previous value of the attribute.

“batch_read”	If non-zero, turns batch_read ON. If zero, turns batch_read OFF. For Sculptor files, ON tells Sculptor to perform read ahead for nextu/prevu/findu/matchu and to send records in batches across the client/server interface. For SQL databases, ON tells Sculptor to use a fast, read only cursor that retrieves records in batches. This can improve performance substantially but reduces concurrency, which means that the program will not see an update made by another user after the batch has been sent but before the record has been read. For SQL databases, batch_read is ON by default.
“limitb”	Sets the SQL LIMIT value used for nextu/prevu/findu/matchu when batch_read is ON. The default is 8000. If there is a noticeable delay when the first command is executed, try a lower value. A value of 0 is allowed but behaviour depends on the database. SQL Server is very fast but other SQL databases can fail completely with large tables. See special note about Microsoft Access below.
“limitu”	Sets the SQL LIMIT value used for nextu/prevu/findu/matchu when batch_read is OFF. The default is 1, which provides maximum compatibility with Sculptor files. Normally there is little point changing this value. It is better to turn batch_read ON when performance is more important than seeing recent updates made by other users.
“limit”	Sets the SQL LIMIT value used for next/prev/find/match. The default is 1, which provides maximum compatibility with Sculptor files. This value is worth changing if performance is important and it’s known that other users will not be updating the file at this time. In this case, a value of 8000 would be a good choice. See special note about Microsoft Access below.

Microsoft Access

We do not recommend using Access for multi-user updates or large tables. A user can overwrite an update made by another user and the Microsoft ODBC driver used by Sculptor seems to provide no way to detect or prevent this. If more than one user is using the database, even if only one is updating it, the database flag DBFL_ACCESS_MULTIUSER can be set to maximise concurrency at the expense of performance. Access is very slow processing large tables. The default LIMIT values are optimal and it is recommended not to change these for Access.

New Environment Variables in Sculptor 2.7

SC_DBSOURCE

This optional environment variable defines a default database connection string. It has the effect of modifying the path to all Sculptor files that are not declared inside a **!database** / **!end_database** pair. This is an easy way to open all files in existing Sculptor programs either through a Sculptor 6 server or as tables in an ODBC database. If a Sculptor 6 server is used, the Sculptor files must all be in the same folder on the server or be relative to that folder and declared as such in the source program.

Sculptor files examples:

```
localhost:/usr/appdata

mylinuxserver:/home/mydata
```

ODBC tables example:

```
SCULPTOR_SERVER= mylinuxserver; DSN=mydsn
```

When setting an environment variable on Windows, do not enclose in “” unless the quote marks are required as part of the value.

When setting an environment variable on Linux and Unix, use “”if the value contains punctuation characters. The “” will be stripped from the value.

The following environment variables modify the default database. These variables are all optional and are ignored if SC_DBSOURCE is not defined.

SC_DBNAME	Name of the default database. The default name is **sculptor**.
SC_DBTYPE	Type of the default database. The default type is 2 (DBT_SCULPTOR). To use an ODBC database, this must be set to 6 (DBT_ODBC).
SC_DBFLAGS	Option flags for the default database.
SC_DBLOGFILE	Pathname for dedicated log file.
SC_RIU_TIMEOUT	riu_timeout value (default 0).