libSceSqlite Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Purpose and Features

libSceSqlite is a library that has added PlayStation®Vita porting layers to SQLite. SQLite is a Relational Database Management System (RDBMS). Applications can use libSceSqlite to enable easy use of the operation function of the SQLite database.

SQLite has the following features.

- SQLite is a RDBMS comprised of only a library and database files.
- Implements major parts of SQL-92, a SQL standard.
- Provides transaction functions.
- Content established in the database is stored in a single database file on the media. File format is not platform dependent.

Main Functions

The libSceSqlite library provides the following main functions.

- Functions for operating a database with SQL
- Transaction functions

Additional Function

In order to facilitate processing for giving memory allocation functions to libSceSqlite, we have provided a function in addition to the SQLite standard function. For details, refer to the "libSceSqlite Reference" document.

Used Resources

The following are the system resources consumed by the libSceSqlite library.

Resource	Description
Footprint	The total of the text and data regions is approximately 450 KiB.
Work memory	When the page size is set to 4KiB, the cache is set to 16 pages, and there are 10000 records: When 1000 records are selected in a field with an index, the work memory is approximately 80.6 KiB. When 1000 records are selected in a field with an index and sorting is performed in a field without an index, the work memory is approximately 718.3 KiB.
Thread	An internal thread is not created implicitly. The thread calling the API performs the operation.
File descriptor	In addition to database files, SQLite uses journaling files and temporary files used to save intermediate results. Refer to "Temporary Files Used by SQLite" (http://www.sqlite.org/tempfiles.html) on the SQLite Web site for details. (The above reference destination has been confirmed as of February 19, 2014. Note that pages may have been subsequently moved or its contents modified.)

Embedding into a Program

Include sqlite3.h in the source program (stdarg.h will also be included implicitly).

libSceSqlite.a is a PRX-format library. When building the program, link stub library libSceSqlite_stub.a. In the program, use the PRX load function `sceSysmoduleLoadModule()` and the identifier `SCE_SYSMODULE_SQLITE` to load PRX.

Sample Programs

Refer to the following sample program that uses the libSceSqlite library.

sample_code/system/api_sqlite/

This sample shows basic uses of the libSceSqlite library. Refer to the sample readme for details.

2 Using the Library

Basic Procedure

This describes the basic procedure for libSceSqlite processing. The following is an overview of the process flow.

(1) Load PRX
(2) Configure the required settings for PlayStation®Vita.
(3) Configure additional settings as necessary.
(4) Use the database.

Preparations

Link stub library libSceSqlite_stub.a to the program. Only savedata0: and below can be used for the file path of database files.

(1) Loading PRX

Use the PRX load function `sceSysmoduleLoadModule()` and the identifier `SCE_SYSMODULE_SQLITE` to load PRX.

(2) Required settings for PlayStation®Vita

SQLite may create temporary files. The directory for creating these files must be set in SQLite, but because there is no appropriate directory in the PlayStation®Vita, this is left unset. Set the appropriate path of the write-enabled directory to `sqlite3_temp_directory`. If this setting is not configured, the SQLite APIs may return `SQLITE_ERROR`.

For example, specify `sqlite3_temp_directory="savedata0:"` to specify a temporary directory for the creation of files directly under the mount point savedata0:.

Since no default memory allocation function is set for libSceSqlite, set memory allocation functions when initializing. Use the `SQLITE_CONFIG_MALLOC` option of `sqlite3_config()` or `sceSqliteConfigMallocMethods()` for setting memory allocation functions. The usage method of `sceSqliteConfigMallocMethods()` is the same as that of the `SQLITE_CONFIG_MALLOC` option of `sqlite3_config()`, but it allows easier setting since there are less functions to be registered.

If memory allocation functions are not set, libSceSqlite will return the error code `SQLITE_NOMEM` for every operation. Also, messages stating "`WARNING: malloc() is unavailable.`" or "`failed to allocate xx bytes of memory`" will be displayed on the console.

(3) Additional settings

SQLite is the recommended state by default, but the various settings, such as the memory allocation method, can be changed. Refer to `sqlite3_config()` of the SQLite API or PRAGMA Statements, which are SQL SQLite extensions, for details.

(4) Using the database

Operate the database according to the SQLite style.

Major APIs Used in Basic Processing

API	Description
`sqlite3_temp_directory`	Global variable indicating the directory in which temporary files are created
`sqlite3_config()`	Sets memory allocation functions.
`sqlite3_open()`	Opens a database file.
`sqlite3_exec()`	Executes a SQL statement.
`sqlite3_close()`	Closes a database file.

3 Reference Information

SQLite Version and Build Settings

The libSceSqlite library was built by adding PlayStation®Vita porting layers to an amalgamation file (a single C code file that contains all C code for SQLite library) distributed by SQLite. The version of the built SQLite is 3.6.23.1. The following are the settings at the time of the build.

- `SQLITE_OS_OTHER`
- `SQLITE_MUTEX_APPDEF`
- `SQLITE_ENABLE_EMMORY_MANAGEMENT`

SQLite 3.6.23 Document

http://www.sqlite.org/sqlite_docs_3_6_23.zip

(The above URL has been confirmed as of February 19, 2014. Note that pages may have been subsequently moved.)

Database Portability

A database created on the PlayStation®Vita can be used with SQLite on a PC.

List of APIs

```
sqlite3_libversion              sqlite3_sourceid
sqlite3_libversion_number       sqlite3_threadsafe
sqlite3_close                   sqlite3_exec
sqlite3_initialize              sqlite3_shutdown
sqlite3_os_init                 sqlite3_os_end
sqlite3_config                  sqlite3_db_config
sqlite3_extended_result_codes   sqlite3_last_insert_rowid
sqlite3_changes                 sqlite3_total_changes
sqlite3_interrupt               sqlite3_complete
sqlite3_complete16              sqlite3_busy_handler
sqlite3_busy_timeout            sqlite3_get_table
sqlite3_free_table              sqlite3_mprintf
sqlite3_vmprintf                sqlite3_snprintf
sqlite3_malloc                  sqlite3_realloc
sqlite3_free                    sqlite3_memory_used
sqlite3_memory_highwater        sqlite3_randomness
sqlite3_set_authorizer          sqlite3_trace
sqlite3_profile                 sqlite3_progress_handler
sqlite3_open                    sqlite3_open16
sqlite3_open_v2                 sqlite3_errcode
sqlite3_extended_errcode        sqlite3_errmsg
sqlite3_errmsg16                sqlite3_limit
sqlite3_prepare                 sqlite3_prepare_v2
sqlite3_prepare16               sqlite3_prepare16_v2
sqlite3_sql                     sqlite3_bind_blob
sqlite3_bind_double             sqlite3_bind_int
sqlite3_bind_int64              sqlite3_bind_null
sqlite3_bind_text               sqlite3_bind_text16
sqlite3_bind_value              sqlite3_bind_zeroblob
sqlite3_bind_parameter_count    sqlite3_bind_parameter_name
sqlite3_bind_parameter_index    sqlite3_clear_bindings
```

sqlite3_column_count	sqlite3_column_name
sqlite3_column_name16	sqlite3_column_decltype
sqlite3_column_decltype16	sqlite3_step
sqlite3_data_count	sqlite3_column_blob
sqlite3_column_bytes	sqlite3_column_bytes16
sqlite3_column_double	sqlite3_column_int
sqlite3_column_int64	sqlite3_column_text
sqlite3_column_text16	sqlite3_column_type
sqlite3_column_value	sqlite3_finalize
sqlite3_reset	sqlite3_create_function
sqlite3_create_function16	sqlite3_aggregate_count
sqlite3_expired	sqlite3_transfer_bindings
sqlite3_global_recover	sqlite3_thread_cleanup
sqlite3_memory_alarm	sqlite3_value_blob
sqlite3_value_bytes	sqlite3_value_bytes16
sqlite3_value_double	sqlite3_value_int
sqlite3_value_int64	sqlite3_value_text
sqlite3_value_text16	sqlite3_value_text16le
sqlite3_value_text16be	sqlite3_value_type
sqlite3_value_numeric_type	sqlite3_aggregate_context
sqlite3_user_data	sqlite3_context_db_handle
sqlite3_get_auxdata	sqlite3_set_auxdata
sqlite3_result_blob	sqlite3_result_double
sqlite3_result_error	sqlite3_result_error16
sqlite3_result_error_toobig	sqlite3_result_error_nomem
sqlite3_result_error_code	sqlite3_result_int
sqlite3_result_int64	sqlite3_result_null
sqlite3_result_text	sqlite3_result_text16
sqlite3_result_text16le	sqlite3_result_text16be
sqlite3_result_value	sqlite3_result_zeroblob
sqlite3_create_collation	sqlite3_create_collation_v2
sqlite3_create_collation16	sqlite3_collation_needed
sqlite3_collation_needed16	sqlite3_sleep
sqlite3_get_autocommit	sqlite3_db_handle
sqlite3_next_stmt	sqlite3_commit_hook
sqlite3_rollback_hook	sqlite3_update_hook
sqlite3_enable_shared_cache	sqlite3_release_memory
sqlite3_soft_heap_limit	sqlite3_load_extension
sqlite3_enable_load_extension	sqlite3_auto_extension
sqlite3_reset_auto_extension	sqlite3_create_module
sqlite3_create_module_v2	sqlite3_declare_vtab
sqlite3_overload_function	sqlite3_blob_open
sqlite3_blob_close	sqlite3_blob_bytes
sqlite3_blob_read	sqlite3_blob_write
sqlite3_vfs_find	sqlite3_vfs_register
sqlite3_vfs_unregister	sqlite3_mutex_alloc
sqlite3_mutex_free	sqlite3_mutex_enter
sqlite3_mutex_try	sqlite3_mutex_leave
sqlite3_db_mutex	sqlite3_file_control
sqlite3_test_control	sqlite3_status
sqlite3_db_status	sqlite3_stmt_status
sqlite3_backup_init	sqlite3_backup_step
sqlite3_backup_finish	sqlite3_backup_remaining
sqlite3_backup_pagecount	sqlite3_strnicmp
sqlite3_version	sqlite3_temp_directory

For details on the each API above and the list of error codes, refer to the following URL.

http://www.sqlite.org/c3ref/funclist.html

http://www.sqlite.org/c3ref/c_abort.html

(The above URL has been confirmed as of February 19, 2014. Note that pages may have been subsequently moved or its contents modified.)

When There Is Not Enough Free Space

When write to a database fails due to insufficient free space, the `SQLITE_FULL` error is returned. When this happens, follow the instructions described in the "How to Handle the Situation Where File System Free Space Becomes Insufficient" section of the "Save Data Free Space" chapter in the "Save Data User's Guide" document and display a system message of Message Dialog or Save Data Dialog for indicating the insufficient file system free space error.

Performance Measurement

The effect of the page size and cache on the processing speed and memory consumption was measured on the PDEL-1000 under the current SDK version. The database files were placed on a memory card. The lookaside memory allocation is disabled with `sqlite3_config(SQLITE_CONFIG_LOOKASIDE, 0, 0)`.

These measurement results are reference values under the current SDK version and do not guarantee future performance. Device and file system performance and behaviors are subject to change.

Scenario ID	Scenario Description
s1	Select 1 record in a field with an index
s2	Select 10 records in a field with an index
s3	Select 100 records in a field with an index
s4	Select 1000 records in a field with an index
s5	Select 10000 records in a field without an index
s6	Select 1 record in a field with an index and sort in a field without an index
s7	Select 10 records in a field with an index and sort in a field without an index
s8	Select 100 records in a field with an index and sort in a field without an index
s9	Select 1000 records in a field with an index and sort in a field without an index
s10	Select 10000 records in a field without an index and sort in a field without an index
s11	Select 1 record in a field with an index and sort in a field with an index
s12	Select 10 records in a field with an index and sort in a field with an index
s13	Select 100 records in a field with an index and sort in a field with an index
s14	Select 1000 records in a field with an index and sort in a field with an index
s15	Select 10000 records in a field without an index and sort in a field with an index
s16	Insert 1 record
s17	Insert 10 records
s18	Insert 100 records
s19	Insert 1000 records

	Memory card 8GB (savedata0:(ux0:data/savedata))							
	Page size: 1K Cache size: 64 pages		**Page size: 2K Cache size: 32 pages**		**Page size: 4K Cache size: 16 pages**		**Page size: 8K Cache size: 8 pages**	
Scen ario ID	**Max. memory consumpti on (bytes)**	**Time required (sec)**	**Max. memory consumpti on (bytes)**	**Time required (sec)**	**Max. memory consumpti on (bytes)**	**Time required (sec)**	**Max. memory consumpti on (bytes)**	**Time required (sec)**
s1	17,160	0.026	23,304	0.019	35,592	0.027	51,840	0.018
s2	43,056	0.169	67,360	0.170	82,520	0.120	85,528	0.097
s3	85,976	1.214	82,648	0.822	82,520	0.743	85,528	0.721
s4	85,976	3.847	82,648	2.306	82,520	1.934	85,528	1.618
s5	85,632	9.392	82,304	3.828	82,176	2.362	85,184	1.795
s6	17,208	0.025	23,352	0.018	35,640	0.026	51,888	0.018
s7	69,336	0.170	93,640	0.171	108,800	0.122	111,808	0.099
s8	227,104	1.231	223,776	0.836	223,648	0.759	226,656	0.739
s9	683,648	58.015	688,536	57.072	735,512	56.779	809,176	55.553
s10	683,264	1464.102	673,544	1503.114	673,256	1489.178	676,264	1485.241
s11	17,208	0.025	23,352	0.019	35,640	0.026	51,888	0.018
s12	69,328	0.170	93,632	0.172	108,792	0.120	111,800	0.098
s13	230,560	1.231	227,232	0.838	227,104	0.760	230,112	0.738
s14	681,600	49.550	690,560	49.740	738,584	49.056	816,344	48.297
s15	86,112	10.795	82,784	4.612	82,656	2.784	85,664	2.033
s16	34,640	1.793	43,176	1.741	71,848	1.740	120,864	1.790
s17	86,744	9.570	98,672	6.950	112,816	6.329	129,192	6.958
s18	142,000	48.935	99,168	29.104	112,896	26.954	129,200	25.458
s19	155,312	283.986	99,672	186.622	114,416	166.184	202,552	192.315

4 Precautions

Restrictions

- Available file paths for database files are limited to savedata0: and below.
- SCE will only fix bugs on the PlayStation®Vita porting layers of SQLite and no other bugs on SQLite.
- The LoadExtension function for shared library is not supported.
- Writing with the file system of the current SDK version is done using write-through. The policy for future SDK writing is to be determined. Whether sync and other synchronization commands operate as expected is to be determined. The behavior of sync and other synchronization commands affects the effectiveness of SQLite journaling.