

 免費電子書

學習

sqlite

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#sqlite

.....	1
<b>1: sqlite</b> .....	<b>2</b>
.....	2
Examples.....	2
.....	2
.....	2
<b>2: PRAGMA</b> .....	<b>3</b>
.....	3
Examples.....	3
PRAGMA.....	3
<b>3: sqlite3_stmtC API</b> .....	<b>4</b>
.....	4
Examples.....	4
.....	4
.....	4
.....	5
<b>4:</b> .....	<b>6</b>
.....	6
Examples.....	6
SQL.....	6
<b>5:</b> .....	<b>7</b>
.....	7
Examples.....	7
TYPEOF.....	7
.....	7
.....	7
/.....	8
<b>ISO8601</b> .....	<b>8</b>
.....	8
<b>Unix</b> .....	<b>8</b>
.....	9



---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [sqlite](#)

It is an unofficial and free sqlite ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sqlite.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# 1: sqlite

3.0		2004-06-18
3.7.11	SELECT maxxy	<a href="#">2012-03-20</a>
3.8.3		<a href="#">2014211</a>

## Examples

SQLiteC `sqlite3.c` ◦

[Perl](#) [Python](#) [Ruby](#) [Android](#) SQLite;SQLite ◦

SQLshell `sqlite3``sqlite3.exe` ◦ [Linux](#); [Windows](#) `sqlite-tools` ◦

SQLite ◦

[sqlite](https://riptutorial.com/zh-TW/sqlite/topic/1753/sqlite) <https://riptutorial.com/zh-TW/sqlite/topic/1753/sqlite>

---

## 2: PRAGMA

SQLite [PRAGMA](#) ◦

### Examples

#### PRAGMA

PRAGMA◦

PRAGMA

- [APPLICATION\\_ID](#)
- [WALjournal\\_mode](#)
- [schema\\_version](#)
- [user\\_version](#)
- [wal\\_checkpoint](#)

PRAGMA

- [auto\\_vacuum VACUUM](#)
- 
- [legacy\\_file\\_format](#)
- [page\\_size VACUUM](#)

```
-- open a connection to a not-yet-existing DB file
PRAGMA page_size = 4096;
PRAGMA auto_vacuum = INCREMENTAL;
CREATE TABLE t(x);           -- database is created here, with the above settings
```

[PRAGMA https://riptutorial.com/zh-TW/sqlite/topic/5223/pragma](https://riptutorial.com/zh-TW/sqlite/topic/5223/pragma)

# 3: sqlite3\_stmtC API

## Examples

[sqlite3\\_prepare\\_v2](#)◦

[sqlite3\\_finalize](#)◦ ◦

[sqlite3\\_bind\\_xxx](#)◦

[sqlite3\\_step](#)◦

```
const char *sql = "INSERT INTO MyTable(ID, Name) VALUES (?, ?)";
sqlite3_stmt *stmt;
int err;

err = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
if (err != SQLITE_OK) {
    printf("prepare failed: %s\n", sqlite3_errmsg(db));
    return /* failure */;
}

sqlite3_bind_int(stmt, 1, 42); /* ID */
sqlite3_bind_text(stmt, 2, "Bob", -1, SQLITE_TRANSIENT); /* name */

err = sqlite3_step(stmt);
if (err != SQLITE_DONE) {
    printf("execution failed: %s\n", sqlite3_errmsg(db));
    sqlite3_finalize(stmt);
    return /* failure */;
}

sqlite3_finalize(stmt);
return /* success */;
```

SELECT ◦ [sqlite3\\_step](#) ◦

- SQLITE\_ROW
- SQLITE\_DONE
- ◦

SQLITE\_DONE◦

[sqlite3\\_column\\_xxx](#)

```
const char *sql = "SELECT ID, Name FROM MyTable";
sqlite3_stmt *stmt;
int err;

err = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
if (err != SQLITE_OK) {
    printf("prepare failed: %s\n", sqlite3_errmsg(db));
```

```

    return /* failure */;
}

for (;;) {
    err = sqlite3_step(stmt);
    if (err != SQLITE_ROW)
        break;

    int          id   = sqlite3_column_int (stmt, 0);
    const char *name = sqlite3_column_text(stmt, 1);
    if (name == NULL)
        name = "(NULL)";
    printf("ID: %d, Name: %s\n", id, name);
}

if (err != SQLITE_DONE) {
    printf("execution failed: %s\n", sqlite3_errmsg(db));
    sqlite3_finalize(stmt);
    return /* failure */;
}

sqlite3_finalize(stmt);
return /* success */;

```

## sqlite3\_reset

```

const char *sql = "INSERT INTO MyTable(ID, Name) VALUES (?, ?)";
sqlite3_stmt *stmt;
int err;

err = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
if (err != SQLITE_OK) {
    printf("prepare failed: %s\n", sqlite3_errmsg(db));
    return /* failure */;
}

for (...) {
    sqlite3_bind_int (stmt, 1, ...); /* ID */
    sqlite3_bind_text(stmt, 2, ...); /* name */

    err = sqlite3_step(stmt);
    if (err != SQLITE_DONE) {
        printf("execution failed: %s\n", sqlite3_errmsg(db));
        sqlite3_finalize(stmt);
        return /* failure */;
    }

    sqlite3_reset(stmt);
}

sqlite3_finalize(stmt);
return /* success */;

```

sqlite3\_stmtC API <https://riptutorial.com/zh-TW/sqlite/topic/5456/sqlite3-stmt--c-api->



---

# 4:

[sqlite3shell](#) [SQLite](#) [sqlite3](#)

## Examples

### SQL

```
sqlite> .output mydatabase_dump.sql  
sqlite> .dump
```

```
sqlite> .output mytable_dump.sql  
sqlite> .dump mytable
```

[.dump.output](#) ◦

```
sqlite> .read mytable_dump.sql
```

<https://riptutorial.com/zh-TW/sqlite/topic/3789/>

# 5:

## SQLite3

### Examples

#### TYPEOF

```
sqlite> SELECT TYPEOF(NULL);
null
sqlite> SELECT TYPEOF(42);
integer
sqlite> SELECT TYPEOF(3.141592653589793);
real
sqlite> SELECT TYPEOF('Hello, world!');
text
sqlite> SELECT TYPEOF(X'0123456789ABCDEF');
blob
```

#### SQLite01

```
sqlite> SELECT 2 + 2 = 4;
1
sqlite> SELECT 'a' = 'b';
0
sqlite> SELECT typeof('a' = 'b');
integer
```

```
> CREATE TABLE Users ( Name, IsAdmin );
> INSERT INTO Users VALUES ('root', 1);
> INSERT INTO Users VALUES ('john', 0);
> SELECT Name FROM Users WHERE IsAdmin;
root
```

#### SQLite

```
> CREATE TABLE Test (
    Col1 INTEGER,
    Col2 VARCHAR(2),      -- length is ignored, too
    Col3 BLOB,
    Col4,                -- no type required
    Col5 FLUFFY BUNNIES -- use whatever you want
);
> INSERT INTO Test VALUES (1, 1, 1, 1, 1);
> INSERT INTO Test VALUES ('xxx', 'xxx', 'xxx', 'xxx', 'xxx');
> SELECT * FROM Test;
1  1  1  1  1
xxx xxx xxx xxx xxx
```

o

## typeof

```
CREATE TABLE Tab (  
  Col1 TEXT CHECK (typeof(Col1) = 'text' AND length(Col1) <= 10),  
  [...]  
);
```

NULLable 'null'。

/

SQLite。

---

## ISO8601

CURRENT\_DATE CURRENT\_TIMECURRENT\_TIMESTAMPISO8601

```
> SELECT CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP;  
CURRENT_DATE  CURRENT_TIME  CURRENT_TIMESTAMP  
-----  
2016-07-08    12:34:56      2016-07-08 12:34:56
```

/

```
> SELECT strftime('%Y', '2016-07-08');  
2016
```

---

## /Julian

```
> SELECT datetime(2457578.02425926);  
2016-07-08 12:34:56
```

julianday()/Julian

```
> SELECT julianday('2016-07-08 12:34:56');  
2457578.02425926
```

---

## Unix

/unixepochUnix

```
> SELECT datetime(0, 'unixepoch');  
1970-01-01 00:00:00
```

strftime()/Unix

```
> SELECT strftime('%s', '2016-07-08 12:34:56');  
1467981296
```

---

//NULL

```
> SELECT time('1:30:00');    -- not two digits  
  
> SELECT datetime('8 Jul 2016');  
[]
```

<https://riptutorial.com/zh-TW/sqlite/topic/5252/>

---

S. No		Contributors
1	sqlite	<a href="#">CL.</a> , <a href="#">Community</a> , <a href="#">e4c5</a> , <a href="#">H. Pauwelyn</a>
2	PRAGMA	<a href="#">CL.</a> , <a href="#">springy76</a>
3	sqlite3_stmtC API	<a href="#">CL.</a>
4		<a href="#">CL.</a> , <a href="#">e4c5</a> , <a href="#">James Toomey</a> , <a href="#">Lasse Vågsæther Karlsen</a> , <a href="#">ravenspoint</a> , <a href="#">Thinkeye</a>
5		<a href="#">CL.</a>