← SQLite 命令 SQLite 数据类型 →

SQLite 语法

SQLite 是遵循一套独特的称为语法的规则和准则。本教程列出了所有基本的 SQLite 语法，向您提供了一个 SQLite 快速入门。

大小写敏感性

有个重要的点值得注意，SQLite 是**不区分大小写**的，但也有一些命令是大小写敏感的，比如 **GLOB** 和 **glob** 在 SQLite 的语句中有不同的含义。

注释

SQLite 注释是附加的注释，可以在 SQLite 代码中添加注释以增加其可读性，他们可以出现在任何空白处，包括在表达式内和其他 SQL 语句的中间，但它们不能嵌套。

SQL 注释以两个连续的 "-" 字符（ASCII 0x2d）开始，并扩展至下一个换行符（ASCII 0x0a）或直到输入结束，以先到者为准。

您也可以使用 C 风格的注释，以 "/*" 开始，并扩展至下一个 "*/" 字符对或直到输入结束，以先到者为准。SQLite的注释可以跨越多行。

```
sqlite>.help -- 这是一个简单的注释
```

SQLite 语句

所有的 SQLite 语句可以以任何关键字开始，如 SELECT、INSERT、UPDATE、DELETE、ALTER、DROP 等，所有的语句以分号（;）结束。

SQLite ANALYZE 语句：

```
ANALYZE;
or
ANALYZE database_name;
or
ANALYZE database_name.table_name;
```

SQLite AND/OR 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

SQLite ALTER TABLE 语句：

```
ALTER TABLE table_name ADD COLUMN column_def...;
```

SQLite ALTER TABLE 语句（Rename）：

```
ALTER TABLE table_name RENAME TO new_table_name;
```

SQLite ATTACH DATABASE 语句：

```
ATTACH DATABASE 'DatabaseName' As 'Alias-Name';
```

SQLite BEGIN TRANSACTION 语句：

```
BEGIN;
or
BEGIN EXCLUSIVE TRANSACTION;
```

SQLite BETWEEN 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

SQLite COMMIT 语句：

```
COMMIT;
```

SQLite CREATE INDEX 语句：

```
CREATE INDEX index_name
ON table_name ( column_name COLLATE NOCASE );
```

SQLite CREATE UNIQUE INDEX 语句：

```
CREATE UNIQUE INDEX index_name
ON table_name ( column1, column2,...columnN);
```

SQLite CREATE TABLE 语句：

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

SQLite CREATE TRIGGER 语句：

```
CREATE TRIGGER database_name.trigger_name
BEFORE INSERT ON table_name FOR EACH ROW
BEGIN
   stmt1;
   stmt2;
   ....
END;
```

SQLite CREATE VIEW 语句：

```
CREATE VIEW database_name.view_name  AS
SELECT statement....;
```

SQLite CREATE VIRTUAL TABLE 语句：

```
CREATE VIRTUAL TABLE database_name.table_name USING weblog( access.log );
or
CREATE VIRTUAL TABLE database_name.table_name USING fts3( );
```

SQLite COMMIT TRANSACTION 语句：

```
COMMIT;
```

SQLite COUNT 子句：

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

SQLite DELETE 语句：

```
DELETE FROM table_name
WHERE  {CONDITION};
```

SQLite DETACH DATABASE 语句：

```
DETACH DATABASE 'Alias-Name';
```

SQLite DISTINCT 子句：

```
SELECT DISTINCT column1, column2....columnN
FROM   table_name;
```

SQLite DROP INDEX 语句：

```
DROP INDEX database_name.index_name;
```

SQLite DROP TABLE 语句：

```
DROP TABLE database_name.table_name;
```

SQLite DROP VIEW 语句：

```
DROP VIEW view_name;
```

SQLite DROP TRIGGER 语句：

```
DROP TRIGGER trigger_name
```

SQLite EXISTS 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name EXISTS (SELECT * FROM   table_name );
```

SQLite EXPLAIN 语句：

```
EXPLAIN INSERT statement...;
or
EXPLAIN QUERY PLAN SELECT statement...;
```

SQLite GLOB 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name GLOB { PATTERN };
```

SQLite GROUP BY 子句：

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

SQLite HAVING 子句：

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithematic function condition);
```

SQLite INSERT INTO 语句：

```
INSERT INTO table_name( column1, column2....columnN)
VALUES ( value1, value2....valueN);
```

SQLite IN 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

SQLite Like 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

SQLite NOT IN 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name NOT IN (val-1, val-2,...val-N);
```

SQLite ORDER BY 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

SQLite PRAGMA 语句：

```
PRAGMA pragma_name;

For example:

PRAGMA page_size;
PRAGMA cache_size = 1024;
PRAGMA table_info(table_name);
```

SQLite RELEASE SAVEPOINT 语句：

```
RELEASE savepoint_name;
```

SQLite REINDEX 语句：

```
REINDEX collation_name;
REINDEX database_name.index_name;
REINDEX database_name.table_name;
```

SQLite ROLLBACK 语句：

```
ROLLBACK;
or
ROLLBACK TO SAVEPOINT savepoint_name;
```

SQLite SAVEPOINT 语句：

```
SAVEPOINT savepoint_name;
```

SQLite SELECT 语句：

```
SELECT column1, column2....columnN
FROM   table_name;
```

SQLite UPDATE 语句：

```
UPDATE table_name
SET column1 = value1, column2 = value2....columnN=valueN
[ WHERE  CONDITION ];
```

SQLite VACUUM 语句：

```
VACUUM;
```

SQLite WHERE 子句：

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

← SQLite 命令 SQLite 数据类型 →

点我分享笔记