SQLite AUTOINCREMENT : Why You Should Avoid Using It

Summary: in this tutorial, you will learn about SQLite AUTOINCREMENT attribute and when to use it in your in your table.

Introduction to SQLite ROWID table

Whenever you create a table without specifying the `WITHOUT ROWID` option, you get an implicit auto increment column called rowid.

The rowid column store 64-bit signed integer that uniquely identify a row within the table.

Let’s see the following example.

First, create a new table name people with two columns: `first_name,` `last_name`

```
CREATE TABLE people (
 first_name text NOT NULL,
 last_name text NOT NULL
);
```

Try It >

Second, insert a row into the `people` table using the following INSERT statement:

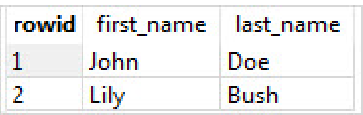
```
INSERT INTO people (first_name, last_name)
VALUES
 ('John', 'Doe');
```

Try It >

Third, query data from the people using the following SELECT statement.

```
SELECT
```

Try It >



rowid	first_name	last_name
1	John	Doe
2	Lily	Bush

So SQLite automatically creates an implicit column named rowid and automatically assigns an integer value whenever you insert a new row.

You can refer to the `rowid` column using its two alias such as `_rowid_` or `oid`

If you create a table that has an `INTEGER PRIMARY KEY` column, this will column points to the rowid column.

The following statement removes the `people` table and recreates it. However, this time, we add another column named `person_id` with `INTEGER PRIMARY KEY` attribute.

```
DROP TABLE people;

CREATE TABLE people (
 person_id INTEGER PRIMARY KEY,
 first_name text NOT NULL,
 last_name text NOT NULL
);
```

Try It >

The `person_id` column is now actually the `rowid` column.

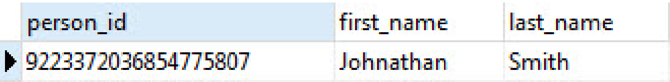
How does SQLite assign an integer value to the rowid column?

If you don't specify the rowid value or you use a `NULL` value when you insert a new row, SQLite assigns an integer that is one larger than the largest rowid in the table. In case you have never inserted any row, the rowid value is 1.

First, insert a row with the maximum value into the `people` table.

```
INSERT INTO people (
 person_id,
 first_name,
 last_name
)
VALUES
 (
 9223372036854775807,
 'Johnathan',
 'Smith'
 );
```

Try It >



person_id	first_name	last_name
9223372036854775807	Johnathan	Smith

Second, insert another row without specifying `person_id`.

```
INSERT INTO people (
 first_name,
 last_name
)
VALUES
 (
 'William',
 'Gate'
 );
```

Try It >

SQLite recommends that you should not use `AUTOINCREMENT` attribute because:

> The `AUTOINCREMENT` keyword imposes extra CPU, memory, disk space, and disk I/O overhead and should be avoided if not strictly needed. It is usually not needed.

In addition, the way SQLite assigns a value for the `AUTOINCREMENT` column is slightly different from the way it used for rowid column.

See the following example.

First, drop and recreate the people table again. This time, we use `AUTOINCREMENT` column.

```
DROP TABLE people;

CREATE TABLE people (
 person_id INTEGER PRIMARY KEY AUTOINCREMENT,
 first_name text NOT NULL,
 last_name text NOT NULL
);
```

Try It >

Second, add a row with the maximum row id value into the `people` table.

```
INSERT INTO people (
 person_id,
 first_name,
 last_name
)
VALUES
 (
 9223372036854775807,
 'Johnathan',
 'Smith'
 );
```

```
  first_name,
  last_name
)
VALUES
  (
  'John',
  'Smith'
  );
```

Try It >

This time, SQLite issued an error message:

```
[Err] 13 - database or disk is full
```

Because it does not reuse the number that has not been used.

The main purpose of using `AUTOINCREMENT` attribute is…

To prevent SQLite to reuse value that has not been used or from the previously deleted row.

If you don’t have any requirements like this, you should not use SQLite AUTOINCREMENT attribute in the primary key.

In this tutorial, you have learned how SQLite `AUTOINCREMENT` attribute works and how it influences the way SQLite assigns value to the primary key column.

Was this tutorial helpful ? 👍 Yes 👎 No

« Previous Tutorial:
SQLite Primary Key

Next Tutorial: »
SQLite Foreign Key

GETTING STARTED

What Is SQLite

Download & Install SQLite

SQLite Sample Database

SQLITE TUTORIAL

SQLite Select

SQLite Order By

SQLite Select Distinct

SQLite Where

SQLite Limit

SQLite IN

SQLite Like

SQLite GLOB

SQLite Left Join

SQLite Inner Join

SQLite Cross Join

SQLite Self-Join

SQLite Group By

SQLite Union

SQLite Full Outer Join

SQLite Having

SQLite Case

SQLite Insert

SQLite Update

SQLite Delete

SQLite Date & Time

SQLite Create Table

SQLite Primary Key

SQLite Foreign Key

SQLite AUTOINCREMENT

SQLite Alter Table

SQLite Drop Table

SQLite Create View

SQLite Index

SQLite Expression Based Index

SQLite Trigger

SQLite VACUUM

SQLite Transaction

SQLite Full-text Search

SQLITE TOOLS

SQLite Commands

SQLite Dump

SQLite Import CSV

SQLite Export CSV

SQLITE FUNCTIONS

SQLite AVG

SQLite COUNT

SQLite MAX

SQLite MIN

SQLite Node.js

SQLite Java

SQLite Python

ABOUT SQLITE TUTORIAL

We created the SQLite Tutorial website to help you master SQLite quickly and easily. In each tutorial, we explain the complex concepts in simple and easy-to-understand ways so that you can not only understand SQLite but also apply it in your development work.

Search this website …

SQLITE PROGRAMMING INTERFACES

SQLite Java

SQLite PHP

SQLite Node.js

SQLite Python

LOOKING FOR A TUTORIAL…

If you did not find the tutorial that you are looking for, you can use the following search box. In case the tutorial is not available, you can request for it using the request for a SQLite tutorial form.

SITE LINKS

Home

About

Contact

Resources

Privacy Policy