Using SQLite

by Sophia

WHAT'S COVERED

This tutorial explores the unique features of SQLite in four parts:

1. Introduction
2. Data Types
3. Primary Keys
4. Table Management

1. Introduction

SQLite is a unique database with some approaches that distinguish it from other databases. One of the greatest advantages of SQLite is that it can be run nearly anywhere. It has been ported to a variety of platforms like Windows, Mac OS, Linux, iOS, Android, and many others. The applications that make use of the SQLite database don't have to be written in a specific language, as you would typically see with other databases. As long as there is some way to bind and work with the external libraries, it works correctly. All of the source code for SQLite is public domain, so it can be reused in other programs with no restrictions.

2. Data Types

SQLite is very flexible when it comes to data types. We have seen with our PostgreSQL database that we have very specific types of data and sizes. SQLite only has a few data types, including REAL, BLOB, NULL, INTEGER, and TEXT.

SQLite is quite forgiving of the type of data that you enter. If a column has the data type of an integer and you try to insert a text string into that column, SQLite will try to convert the text string into an integer. For example, if the user enters a string of '9876' into an integer column, that value is converted to an integer of 9876 and stored in the column. If you try to insert a non-numeric string like 'abcd' into an integer column, most other databases would throw an error. However, SQLite will just store the string value in the column.

You may remember that many of our columns have a VARCHAR(40) that allows up to 40 characters. If we try to insert data into a column that has more than, many databases would either throw an error or truncate the string to 40 characters. SQLite instead stores the entire string without the loss of information. This is viewed as a feature of SQLite rather than a bug. However, it does make it very difficult for applications built using SQLite to be ported to other databases, as situations like these can create problems.

Booleans are another instance where SQLite does not store a separate data type. Instead, true and false are represented by integers of 0 and 1. Dates are also different in SQLite, where they are stored in a TEXT string. If a date is stored as an integer, SQLite stores the number of seconds since 1970. There are built-in date and time features to change between those values, which should make it easier to follow. Tables can even be created without any data types at all, which can be quite confusing. For example, you could run a command like

CREATE TABLE myTable (a, b, c);
This would create a table named myTable with the column names a, b, and c that have no data types defined. Anything could be stored in those columns.

3. Primary Keys

Primary keys should be unique and cannot be empty. This is a big piece of any database. However, with SQLite, a primary key can be a null value. This was originally a bug in the program. However, when the bug was identified, so many databases had already made use of that bug as a feature that the bug was kept.

4. Table Management

In SQLite, after tables have been created, you can only rename a table with RENAME TABLE. You can only rename a column in the table using RENAME COLUMN, and add a new column at the end of a table with ADD COLUMN. You cannot add any constraints to a table once the table has been created, nor can you drop a column or alter a column. The joins between tables in SQLite can also be a bit of a challenge, especially with outer joins. The only outer join you can run is an OUTER LEFT join. SQLite only supports an INNER join, LEFT join, OUTER LEFT join, CROSS join and OUTER join.

As you can see, there are a lot of benefits to SQLite, but there are also many intricacies to be aware of when using the database.

SUMMARY

SQLite is a popular serverless database with many unique features, including its flexible typing.

Source: Authored by Vincent Tran