Sqlite: CURRENT_TIMESTAMP is in GMT, not the timezone of the machine

Ask Question

I have a sqlite (v3) table with this column definition:

```
"timestamp" DATETIME DEFAULT CURRENT_TIMESTAMP
```

The server that this database lives on is in the CST time zone. When I insert into my table without including the timestamp column, sqlite automatically populates that field with the current timestamp in GMT, not CST.

Is there a way to modify my insert statement to force the stored timestamp to be in CST? On the other hand, it is probably better to store it in GMT (in case the database gets moved to a different timezone, for example), so is there a way I can modify my select SQL to convert the stored timestamp to CST when I extract it from the table?

sql sqlite timezone timestamp

edited Jul 13 '14 at 22:56 Benjamin 14.6k • 26 • 112 • 213

asked Dec 19 '08 at 15:53 BrianH 4,284 • 7 • 44 • 70

12

Storing timestamps in UTC is considered best practice. Convert to local time when *presenting* them. – csl Oct 12 '15 at 21:56

POSIX defines timestamps as UTC: stackoverflow.com/a/34107910/895245 – Ciro Santilli 包子露宪 六四事件 法轮功 Dec 5 '15 at 18:53

11 Answers

I found on the sqlite documentation (https://www.sqlite.org/lang_datefunc.html) this text:

> Compute the date and time given a unix timestamp 1092941466, and compensate for your local timezone.

```
SELECT datetime(1092941466, 'unixepoch', 'localtime');
```

That didn't look like it fit my needs, so I tried changing the "datetime" function around a bit, and wound up with this:

```
select datetime(timestamp, 'localtime')
```

That seems to work - is that the correct way to convert for your timezone, or is there a better way to do this?

edited Oct 17 '15 at 3:04 Stephan 25.3k • 28 • 141 • 240

answered Dec 19 '08 at 16:04 BrianH 4,284 • 7 • 44 • 70

This is the correct way to convert timestamps to local time in SQLite. – Doug Currie Dec 21 '08 at 17:49

second one was perfect solution thanks. – j2me Jun 27 '12 at 7:26

that is some powerfull stuff the datetime function. +1 for helping me. I noticed it's really powerfull, you can throw this at it: datetime(date_field , time_field) ex: 2012-10-13|04:15 and it will spit out this: 2012-10-13 04:15:00. very powerfull. – Glenn Plas Oct 12 '12 at 15:21

SELECT datetime(CURRENT_TIMESTAMP,'localtime') would be exactely what the question needed – kurast Jun 14 '13 at 14:42

simply use local time as the default:

```
CREATE TABLE whatever(
     ....
     timestamp DATE DEFAULT (datetime('now','localtime')),
     ...
);
```

answered May 30 '10 at 14:42 hoju 12.3k • 26 • 103 • 158

This does have the problem of not being transferable across timezones however, no? – Vadi Sep 9 '13 at 7:40

Good to know, but it seems like a dangerous practice... – iconoclast Nov 4 '14 at 23:45

yes it is, its not working for me either, I am using sqlite in ZF2 – rohu2187 May 13 '15 at 15:19

@iconoclast it shouldn't be dangerous where if you are aware of this and if it makes your implementation easier, should it? if this is what i need only to store the local time zone into the database and i know my application is never going to need other way then i will. – xFighter Dec 15 '16 at 7:31

@xFighter *I know my application is never going to need other way*... Until you forget about that or someone else is using it. – MKesper May 29 '17 at 7:44

You should, as a rule, leave timestamps in the database in GMT, and only convert them to/from local time on input/output, when you can convert them to the user's (not server's) local timestamp.

It would be nice if you could do the following:

```
SELECT DATETIME(col, 'PDT')
```

...to output the timestamp for a user on Pacific Daylight Time. Unfortunately, that doesn't work. According to this SQLite tutorial, however (scroll down to "Other Date and Time Commands"), you can ask for the time, and then apply an offset (in hours) at the same time. So, if you do know the user's timezone offset, you're good.

Doesn't deal with daylight saving rules, though...

edited Dec 20 '08 at 8:11

answered Dec 19 '08 at 16:08 Roger Lipscombe 52.9k • 37 • 178 • 298

Oh, that would be handy! I tried using 'PDT' and 'PST' but it returned nulls. Apparently that isn't available in sqlite. – BrianH Dec 19 '08 at 16:53

In the (admitted rare) case that a local datatime is wanted (I, for example, store local time in one of my database since all I care is what time in the day is was and I don't keep track of where I was in term of time zones...), you can define the column as

```
"timestamp" TEXT DEFAULT (strftime('%Y-%m-%dT%H:%M','now', 'localtime'))
```

The %Y-%m-%dT%H:%M part is of course optional; it is just how I like my time to be stored. [Also, if my impression is correct, there is no "DATETIME" datatype in sqlite, so it does not really matter whether TEXT or DATETIME is used as data type in column declaration.]

answered Feb 21 '09 at 3:38 polyglot 5,657 • 5 • 37 • 59

When having a column defined with "`NOT NULL DEFAULT CURRENT_TIMESTAMP`," inserted records will always get set with UTC/GMT time.

Here's what I did to avoid having to include the time in my INSERT/UPDATE statements:

```
--Create a table having a CURRENT_TIMESTAMP:
CREATE TABLE FOOBAR (
    RECORD_NO INTEGER NOT NULL,
    TO_STORE INTEGER,
    UPC CHAR(30),
    QTY DECIMAL(15,4),
    EID CHAR(16),
    RECORD_TIME NOT NULL DEFAULT CURRENT_TIMESTAMP)

--Create before update and after insert triggers:
CREATE TRIGGER UPDATE_FOOBAR BEFORE UPDATE ON FOOBAR
    BEGIN
       UPDATE FOOBAR SET record_time = datetime('now', 'localtime')
       WHERE rowid = new.rowid;
    END

CREATE TRIGGER INSERT_FOOBAR AFTER INSERT ON FOOBAR
    BEGIN
       UPDATE FOOBAR SET record_time = datetime('now', 'localtime')
       WHERE rowid = new.rowid;
    END
```

Test to see if it works...

```
--INSERT a couple records into the table:
INSERT INTO foobar (RECORD_NO, TO_STORE, UPC, PRICE, EID)
    VALUES (0, 1, 'xyz1', 31, '777')

INSERT INTO foobar (RECORD_NO, TO_STORE, UPC, PRICE, EID)
    VALUES (1, 1, 'xyz2', 32, '777')

--UPDATE one of the records:
UPDATE foobar SET price = 29 WHERE upc = 'xyz2'

--Check the results:
SELECT * FROM foobar
```

Hope that helps.

edited May 30 '10 at 15:07 Donal Fellows 96.7k • 13 • 106 • 164

answered Aug 11 '09 at 18:11 Patrick

```
SELECT datetime('now', 'localtime');
```

answered Jan 16 '14 at 17:53 liquide 798 • 3 • 15 • 25

```
SELECT datetime(CURRENT_TIMESTAMP, 'localtime')
```

answered Apr 22 '15 at 14:36 Jens A. Koch 23.5k • 7 • 75 • 89

I think this might help.

```
SELECT datetime(strftime('%s','now'), 'unixepoch', 'localtime');
```

edited May 19 '11 at 12:53 Alex 26.9k • 9 • 57 • 113

answered Feb 20 '09 at 23:46 Kenneth Navarro

for unxitime; SELECT strftime('%s','now','localtime'); – Hariboo Oct 17 '17 at 11:20

The current time, in your machine's timezone:

```
select time(time(), 'localtime');
```

As per http://www.sqlite.org/lang_datefunc.html

answered Nov 14 '12 at 2:22 Joseph 149 • 2 • 10

`Time ( 'now', 'localtime' )` and Date `( 'now', 'localtime' )` works.

edited Nov 11 '15 at 12:16 Matt 59.5k • 16 • 114 • 154

answered Nov 7 '14 at 5:26 Julian 11 • 1

You can also just convert the time column to a timestamp by using strftime():

```
SELECT strftime('%s', timestamp) as timestamp FROM ... ;
```

Gives you:

> 1454521888

'timestamp' table column can be a text field even, using the `current_timestamp` as DEFAULT.

Without strftime:

```
SELECT timestamp FROM ... ;
```

Gives you:

> 2016-02-03 17:51:28

answered Feb 3 '16 at 18:01 danger89 1,366 • 10 • 14

protected by Josh Crozier Aug 15 '17 at 5:24

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?