UNIX TIME

Storing and working with dates and times is difficult. You've got to deal with different time zones, different calendar systems, leap years, various approaches to daylight saving time, different ways of writing the date in different countries... Having to account for all of these things, and more, is a lot of work, and easy to get wrong, and hard to test for correctness.

Some type of universal time, that's the same the world over, would be useful. And then local users can convert this universal time to their own local time zone and calendar representation, as needed.

That's what Unix Time is. It's the number of seconds since 00:00 January 1, 1970, UTC. [UTC = temps universel coordonné in French, or universal coordinated time in English, basically the same thing as GMT, the time in London, but without daylight savings; <https://en.wikipedia.org/wiki/Coordinated_Universal_Time> ]

The instant of midnight, Jan 1, 1970, UTC, is referred to as the 'epoch' so you might hear Unix time described as 'seconds since the epoch'. Any computer, anywhere in the world, when asked for the Unix time, will produce the exact same number, even if the local clock times are different.  It's also an integer value, so compact and easy to store and manipulate.

Unix times are very common. Examples: timestamps for events in log files; modification dates for files in your file system. Typically, when computers store times, they'll store them as Unix time, and then convert to human-readable; and localize to the correct time zone and format; as needed. Similarly for databases, the data in a date or datetime column is stored as a unix time. Your database engine/driver converts it to something human-readable when the date is accessed.

**Useful things you can do with Unix time:**

-  Convert dates to unix time and save in SQLite databases; which doesn't have a date column data type.

 - Generally saving dates in a standard representation

 - Creating a random date between two boundaries - generate a random number between the Unix time values of the boundary dates.

 - Create unique timestamps for filenames or other identifiers. Just make sure you don't create more than one a second!

 - Many systems can report the unix time as milliseconds since the epoch, useful if you need finer-grained times. Useful for timing how long short tasks take, or unique timestamps.

 - Most programming languages can report the current Unix time, and convert timestamps to local time. In Python,

import time  
from datetime import datetime

print(time.time()) # Unix time  
print(datetime.fromtimestamp(1500000000)) # Convert timestamp to local time

<https://docs.python.org/3.6/library/datetime.html>

**Timestamp bugs**

If you have a weird error where a date is reported as 1969 or 1970; or something seems to have happened 47 years ago, it's probably because somewhere a Unix time is erroneously converted to 0, or a database column for time doesn't have a value stored. This Facebook bug seems to be an instance, <http://gizmodo.com/the-nerdy-explanation-for-facebooks-46-years-ago-glit-1750464185>

And if your dates seems to be a couple of centuries in the future; or decades in the past, verify if you are using the correct time - seconds or milliseconds since the epoch.

**Unix time trivia:** At 02:40:00 UTC on Friday, 14 July 2017, [20:40:00 on Thursday, 13th July CST] the Unix time value will equal 1,500,000,000 seconds.

**More info:**

<https://en.wikipedia.org/wiki/Unix_time>