Pitfalls in transaction time

Helena Palovská
Vysoká škola ekonomická v Praze
Fakulta statistiky a informatiky
nám. W. Churchilla 4, 130 67 Praha 3
palovska@vse.cz

Abstract: *Impreciseness of the notion of "transaction time" is clarified. Uncertainty in application of "transaction time" as "the starting time of the existence of a database record" or "when we got to know" is pointed out. Questions of computer clock synchronization and currently available methods for computer clock synchronization are resumed. Examples of faults in temporal data interpreted as transaction time data in applications are introduced. Safe approaches to the design of systems with temporal data are proposed.*

Keywords: temporal database, transaction time, computer clock synchronization, chronology, chronological order

Introduction

For more than two decades, the database community deals with the temporal aspects of data. Two kinds of time information associated with data records are considered: *transaction time* and *valid time*. *Transaction time* refers to the period during which the database stores the record. It can solely represent the time at which data was recorded, optionally including an interval at which data should be removed from the database. Also, in a logging system, it may include the time at which the record actually was removed. *Valid time* refers to the period during which the fact was or still is or is intended to be true in the business domain. For instance, a time period (T_1,T_2) during which person P lived, lives or will live at address A is a valid time period for the fact "person P lives at address A". The period for which the record "person P lives at address A during time period (T_1,T_2)" is stored in the database represents the transaction time. Therefore, with a database record there could be four time fields added: starting valid time (when the fact starts to be true in the business domain), ending valid time (when the fact ends to be true in the business domain), starting transaction time (when this was recorded in the database) and ending transaction time (when this will be/should be/was removed from the database). If both valid and transaction time are recorded the term is *bitemporal database* ([6,2]). Without this specification, the more general term is *temporal database*.

While the structure design and information retrieval point of view in temporal databases is intensively investigated, the question of acquirement and processing of transaction time datum is predominantly considered unproblematic. Obviously, the intent to maintain a transaction time *period*, that is to store not only the starting time instant but also the ending time instant, leads to issues concerning database organization and operation. Against expectation, if the intention is to store only the starting time instant, problems can arise, too. This article attempts to show and explain how such situations can occur. In the following discussion, *transaction time* represents *the starting time of the existence of a database record*.

1. What is transaction time?

It is called transaction time and this itself makes it unclear. Database transaction is a process starting at some time instant and ending at another time instant. What the "starting time of a database record existence" should be? By a common sense, it should be the time instant when the transaction storing the record is committed. Unfortunately, also the "commit" is a process, and the question continues to be not solved. At some time instant of these processes, i.e. a transaction process including the commit process, the "current time" is inserted into an appropriate field of the record, but the transaction process further continues.

Let us investigate the preceding common sense argument: why should the "commit" be the starting time instant of existence of a record? From the point of view of the transaction owner, the record exists after the insert command was confirmed to him as successful, and if the transaction is rolled back, the record ceases to exist for him. From the point of view of other transactions (other "threads") running in the meantime, if the record exists depends upon *the transaction isolation level* [3,4]. If this isolation level is set to *serializable*, the record is visible, and that means it starts to exist for all threads, during the commit of the firstly mentioned transaction. If the isolation level is set other, for other threads the record can start to exist at another time and it can even cease to exist likewise.

As a conclusion, we see that the task to achieve "the real transaction time" cannot be accomplished. We can obtain only some time between the starting instant of the transaction and the ending instant of it. These two instants could only be taken from the DBMS point of view. Of these two, the only technologically unproblematic "time" is the start instant. For the business but, this time would be of no value, as we shall see from the next.

1.1 What is transaction time intended for

Usually, it is interpreted as a time when the fact was recorded into the database. If this is the case, all the impreciseness mentioned in the previous section is to be taken into consideration. What can come out of this special application?

We can ask what the state of the database was at some given time. As explained before, we cannot get a precise answer because of the difference between "transaction time" recorded and the time of the transaction commit. Eventually, "transaction time" is being used to roll back the database.

If our intention is to review what information could be retrieved from the database at a given time, an essential piece of information is missing – what was the transaction isolation level at the "transaction time". This piece of information could possibly be added to the "transaction time" data of the record. Nevertheless, the commit time is missing either.

Another usual interpretation of the transaction time is *when we got to know*. An obvious objection is that the data could be recorded later than the specific user got to know the fact. More hidden is the need to specify who of us do we mean: the user who inserted the record could perhaps know the corresponding fact in advance, while other users could get to know it later – possibly we should also take to account the corresponding transaction isolation level.

Because the transaction time can be conceived as a mere special case of general notion of valid time – that is, the valid time of a higher order fact "... was recorded in the database" or "we knew that ..." or "we considered to be true that ..." – all possible usages of valid time can take place. These are discussed in the next two sections.

1.2 Other possible business usages

Possible usages arise from temporal data comparison, to get an answer to a question "how long did it take to..." or "how long did it last". Valid time can be compared with transaction time as well as the transaction time with other transaction time. In these cases transaction time is considered the time when "the user" got to know.

1.3 Other technical usages

Technical information can be derived from transaction time comparison, in this case the transaction time being considered the time when the data was recorded. In this case the interpretation is "how long did it take to record...".

1.4 Misconceived usages

Trickery nonsense originates from mismatching the transaction time and the user time. That is, time of the database clock and time of the user's clock. These are essentially different. Next section will clarify this.

2. Clock synchronization

This section attempts to explain to what measure two different computer clocks can be saying the same. The most accurate time and frequency standard known give *atomic clocks*. Combined input of many atomic clocks around the world makes up the International Time Standard, which is the primary international time standard. *Global Positioning System* (GPS) provides for distribution of this time standard around the world.

Time servers provide for time standard distribution in computer networks. Some time servers use atomic clocks, but the most common true time source for time serves is GPS. Also another time server on the network or the Internet can be used as a time reference for a time server, and also a connected radio clock.

Other computers can utilize the service of time servers via *Network Time Protocol* (NTP) using UDP, utilize *Precision Time Protocol* on LANs, or *White Rabbit* Ethernet-based network, for instance. Any computer can adjust its clock by regulating its speed. Using true time information issuing from some source, offset of the two clocks, jitter and an observed delay of message transmission the adjustment is calculated.

2.1 Network Time Protocol clock accuracy

Following examples illustrate time precision achievable by NTP; NTP uses Internet routes. The first are two outputs from `fis2.vse.cz`, a computer in local network of University of Economics, Prague. First (offset is in milliseconds, * denotes system peer, i.e. to which the local clock is actually adjusted):

```
ntpq> pe
     remote           refid      st t when poll reach   delay   offset
jitter
=======================================================================
=======
-ca65sb.net.vse. 131.188.3.220    2 u  390  512  377    0.762   -0.599
3.833
*ca65rb.net.vse. 192.93.2.20      2 u   99  512  377    0.716    0.159
1.037
+ipv6jm.vse.cz   195.113.144.204  2 u  346  512  377    0.296    0.152
0.188
-jmnt.vse.cz     91.189.94.4      3 u   95  512  377    0.606   -4.214
0.339
-ns.infonet.cz   145.238.203.10   3 u  163  512  377    2.360    0.862
1.210
+lx.ujf.cas.cz   195.113.144.201  2 u  471  512  377    1.443    0.461
0.362
-ntp.t-mobile.cz 192.53.103.104   2 u  345  512  377    3.167    2.007
0.521
```

A while later:

```
ntpq> pe
     remote           refid      st t when poll reach   delay   offset
jitter
=======================================================================
=======
-ca65sb.net.vse. 195.113.144.201  2 u  409  512  377    0.762   -0.599
3.822
+ca65rb.net.vse. 192.93.2.20      2 u  121  512  377    0.716    0.159
1.032
*ipv6jm.vse.cz   195.113.144.204  2 u  362  512  377    0.309    0.130
0.128
-jmnt.vse.cz     91.189.94.4      3 u   99  512  377    0.606   -4.214
0.233
-ns.infonet.cz   145.238.203.10   3 u  181  512  377    2.545   -0.096
1.464
+lx.ujf.cas.cz   195.113.144.201  2 u  486  512  377    1.427   -0.039
0.372
-ntp.t-mobile.cz 192.53.103.104   2 u  356  512  377    3.167    2.007
0.415
```

In this case, the accuracy can be expected about tenths of milliseconds. Following two outputs are from a notebook in an home network connected by a leased line. First:

```
ntpq> pe
     remote           refid      st t when poll reach   delay   offset
jitter
=======================================================================
=======
*odine.cgi.cz    195.113.144.201  2 u 1003 1024  377   14.141    0.058
1.144
-bobek.sh.cvut.c 195.113.144.201  2 u  413 1024  177   42.048   11.834
33.414
+srv1.trusted.cz 195.113.144.201  2 u  602 1024  377   14.797   -1.232
35.018
```

```
+relay.qls.cz    147.231.19.43    2 u  987 1024  377   24.733    0.585
3.320
-ntp1.karneval.c 147.231.19.43    2 u  983 1024  373   12.835   -3.195
2.469
```

A while later:

```
     remote           refid      st t when poll reach   delay   offset
jitter
=======================================================================
=======
-odine.cgi.cz    195.113.144.201  2 u  879 1024  377   17.587   -3.036
0.716
+bobek.sh.cvut.c 195.113.144.201  2 u  287 1024  377   11.919   -3.662
 0.908
*srv1.trusted.cz 195.113.144.201  2 u  480 1024  377   13.608   -3.544
0.599
-relay.qls.cz    147.231.19.43    2 u  863 1024  377   14.643   -6.692
0.046
+ntp1.karneval.c 147.231.19.43    2 u  857 1024  337   13.988   -3.156
0.442
```

In this case, the expected accuracy is above one order worse.

For a computer connected to the Internet via GSM, application of NTP makes no sense because this protocol is suitable only in a case of a long-lasting connection.

2.2 LAN protocols clock accuracy

Precision Type Protocol achieves clock accuracy in the sub-microsecond range. White Rabbit aims at being able to synchronize about 1000 nodes with sub-nanoseconds accuracy over fiber and copper lengths of up to 10 km.

2.3 Conclusion on clock synchronization

As a conclusion, when comparing times from different computer clocks you must scale down to a measure corresponding to the expected divergence specific to the synchronizing connection of the two computers (see Table 1 bellow). Especially, care should be taken in case of multi-tier architecture, when application tier and database tier run on different computers. Database time-stamp precision is usually in microseconds, can be up to nanoseconds.

Table 1: Computer clock accuracies outlook

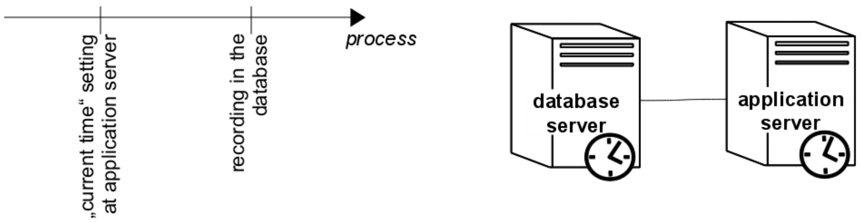
	Expected accuracy
Database timestamps	from 10^{-3}s till 10^{-9}s, usually 10^{-6}s
NTP – server in university network, Prague	10^{-4}s
NTP – notebook at home leased line, Prague	10^{-3}s
PTP type protocol	less then 10^{-6}s
White Rabbit	less then 10^{-9}s

3. Examples of fault

Following examples originate from real business applications.

3.1 Application server time-stamping instead of a database server time-stamping

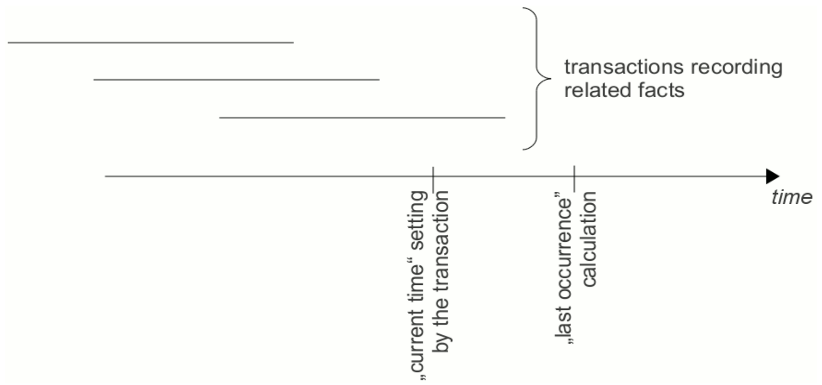
For optimization of the whole information system, some operations are performed by the application server to reduce burden of the database server. If these operations include “current time” setting intended as a transaction time for some records, the architect must be aware of significant decrease of accuracy of such “temporal” data and of reduced applicability of subsequent temporal comparisons.


„current time“ setting at application server
recording in the database
process
database server
application server

Picture 1: Two possible sources for discrepancy

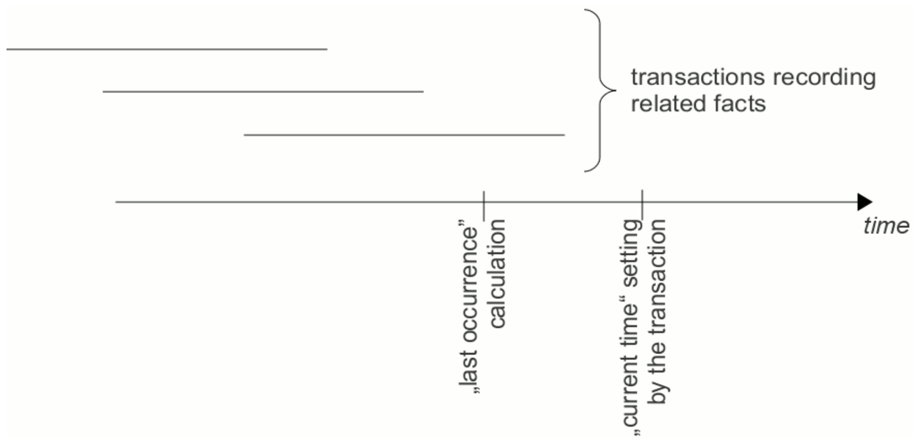
3.2 Derived data for optimization of processes

Because execution of many temporal queries is demanding, derived data are often calculated to facilitate corresponding business requirements. While many derived data are suitable to be calculated during the transaction that inserts or updates corresponding business facts, this may be not the case of temporal derivations.


transactions recording related facts
time
„current time“ setting by the transaction
„last occurrence“ calculation

Picture 2: Chronological nonsense

For instance, information, when last event of some type occurred related to currently recorded business facts, may be needed. Events of this type can yet be still in the progress of recording, and therefore not be known in the database at the beginning of the transaction. Transactions of these recordings yet can be committed before the transaction in consideration, so actually these events are to be taken in the “last occurrence” calculation. Now it depends on the order of corresponding commands in the transaction of consideration. If the “transaction time” of newly inserted facts is calculated before the “last occurrence” calculation, we can obtain chronological nonsense. If the “transaction time” is calculated after the “last occurrence” calculation, we can get not actually last occurrence, because some new occurrences can be recorded in the meantime.


transactions recording related facts
time
„last occurrence” calculation
„current time“ setting by the transaction

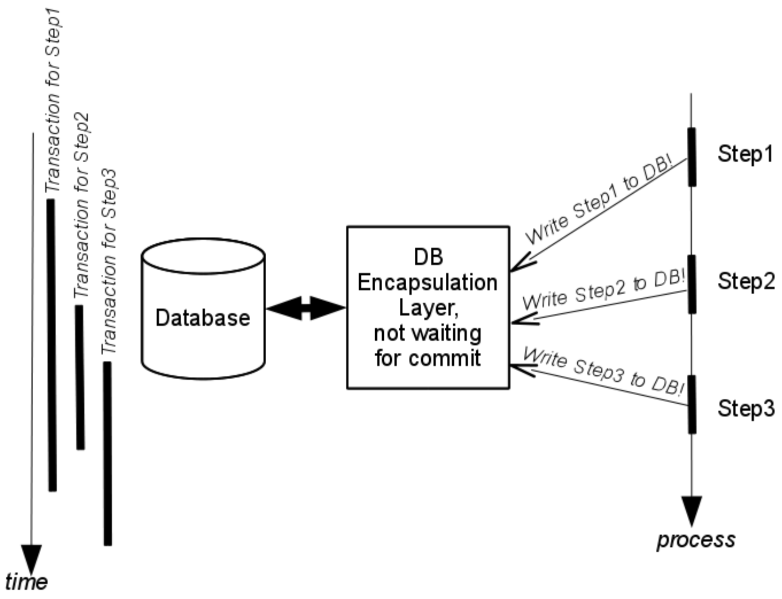
Picture 3: Not actually last occurrence

For such cases, an assessment of sufficient delay is necessary and some *cron* [7] utility can complete the “last occurrence” calculation task.

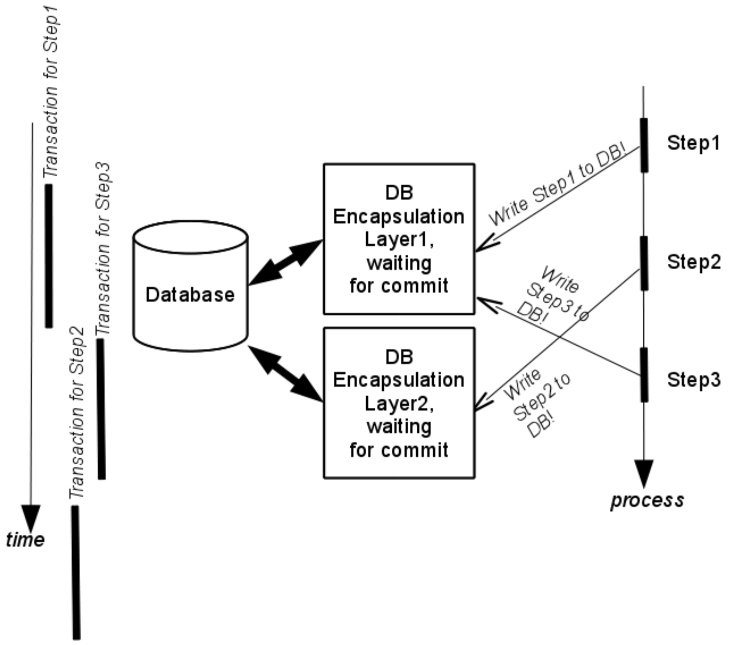
3.3 Illusive successiveness

When a set of events follows a causal order and these events are recorded to the database in a such way, that before handing over to the next step the process waits for database commit, chronological order of according database time-stamps will be preserved. When each step instead sends a request to a mediating actor (see e.g. [1]) that sends requests to a database and does not wait for the commit before continuing, the chronological order can be corrupted. When information about the order of the events is derived from database time-stamps in such a case, this information can be false.

Similar situation can happen when the mediating actor waits for a commit of a requested transaction before continuing; if the system uses more than one such an actor to speed up the operation of the system, the chronological order can be corrupted as well.



Picture 4: Corrupted chronological order, version I



Picture 5: Corrupted chronological order, version II

4. Conclusion

Comparison of time data can be relied upon only when the data come from one and only one clock, and if this comparison refers to chronology. Because computer clocks possibly adjust their speed, absolute value comparison would bear measurement error; such an error can be estimated. When data come from different clocks, achievable measure of synchronization must be taken in account; in such case small difference in time data may mean that these are chronologically incomparable. Chronological order generally is reliable upon only in the case of a causal order.

“Transaction time” can only be set by a database server and should only be interpreted as a time instant from within the period during which the transaction was in process. No exact state of database can be deduced from “transaction time” data stored in database.

References

[1] AMBLER, S.: Encapsulating Database Access: An Agile "Best" Practice. *www.agiledata.org: Techniques for Successful Evolutionary/Agile Database Development* [Cit. 27.12.2011.] http://www.agiledata.org/essays/implementationStrategies.html

[2] DATE, Ch. J., DARWEN, H., LORENTZOS, N.: *Temporal Data and the Relational Model*. 1st Ed. The Morgan Kaufmann 2002. 422pages. ISBN 1-55860-855-9.

[3] DIGITAL Equipment Corp.: *Information Technology - Database Language SQL. (Proposed revised text of DIS 9075). 1992.* [Cit. 27.12.2011)] http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[4] MELTON, J. — SIMON, A.R.: *Understanding the New SQL: A Complete Guide*. 1st Ed. The Morgan Kaufmann Series in Data Management Systems 1993. 536 pages. ISBN 1558602453

[5] ORACLE corp.: *Oracle Workspace Manager.* [Cit. 27.12.2011.] http://www.oracle.com/technetwork/database/enterprise-edition/index-087067.html

[6] SNODGRASS, R. T.: *Developing Time-Oriented Database Applications in SQL.* Morgan Kaufmann Series in Data Management Systems 1999. 504 pages. ISBN 1-55860-436-7

[7] WIKIPEDIA, The Free Encyclopedia: *cron.* [Cit. 27.12.2011.] http://en.wikipedia.org/wiki/Cron