University of Zurich[UZH]

Department of Informatics

University of Zürich
Department of Informatics
Binzmühlestr. 14
CH-8050 Zürich
Phone. +41 44 635 43 11
Fax +41 44 635 68 09
www.ifi.uzh.ch/dbtg

UZH, Dept. of Informatics, Binzmühlestr. 14, CH-8050 Zürich

Jasmin Ebner

Prof. Dr. Michael Böhlen
Professor
Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, 24. Februar 2016

Bachelor's Thesis
Topic: Temporal Integrity Constraints in Databases With Ongoing Timestamps

Data that is associated with time intervals is present in many real-word applications like employment contracts or insurance policies. In all these applications, it is not always possible to specify a specific start and end date of the contract or policy. Instead, they often have specific start dates and are ongoing afterwards. For those applications, databases with ongoing timestamps are required to store the data. In this project, we focus on relations with a set of non-temporal attributes and a time attribute that contains ongoing, closed-open time intervals.

Integrity constraints are a basic functionality that a database system provides. Temporal integrity constraints consider the time attribute when determining whether the constraint is fulfilled. The two integrity constraints we focus on are temporal primary key and temporal foreign key constraints. A temporal primary key, for instance, ensures that there do not exist two tuples in the same relation with the same non-temporal attributes and overlapping time intervals.

Ongoing timestamps, i.e., ongoing time points and ongoing time intervals, represent different timestamps for different current times and are therefore dependent on the current time. For instance, the ongoing time point t^+ represents t for those current times t_{curr} at which $t_{curr} \leq t$, and a time point equal to the current time at all other current times. Thus, a relation that contains ongoing timestamps in its time attribute represents different relations at different current times. Temporal integrity constraints on such relations can therefore be fulfilled for some current times, but violated for other current times. For instance, consider the two tuples t_1 and t_2 with time intervals $t_1.T = [2014, 2016^+)$ and $t_2.T = [2018, 2020)$ as shown in Figure 1; assume that both tuples have equal non-temporal attributes. When inserting tuple t_2 into a relation that already contains tuple t_1, a temporal primary key constraint is fulfilled as long as the current time is less than or equal to 2018. Afterwards, the constraint is violated, as the time intervals overlap. Thus, simply the advancing of time can cause that a fulfilled primary key constraint

gets violated.

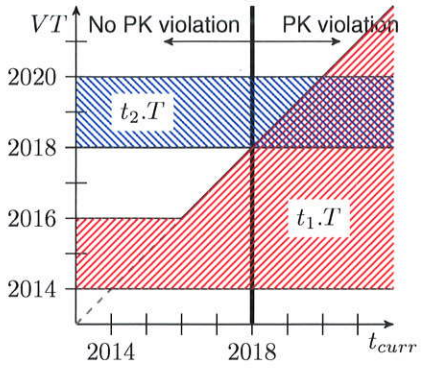

Figure 1: Primary Key (PK) violations can occur with ongoing timestamps when time advances.

For databases with ongoing timestamps, we require that the ongoing primary key constraints and foreign key constraints are fulfilled at every current time: An ongoing primary key constraint is fulfilled if the temporal primary key constraint, as introduced above, is fulfilled across all current times; analogously for an ongoing foreign key constraint: it is fulfilled if the temporal foreign key constraint is fulfilled across all current times. With these definitions of ongoing constraints, the database system needs to check only once, when the tuple is inserted, updated, or deleted, whether the ongoing constraint is fulfilled; no postponed violation of these constraints can occur when time advances. Whenever an ongoing constraint is violated, the modification that causes the violation is rejected. For instance, when tuple t_2 of our example above should be inserted in the relation that already contains t_1, the insertion is rejected. The reason is that the time intervals of t_1 and t_2 overlap at at least one current time, for instance, 2019.

In this bachelor's thesis, the student should integrate ongoing primary and foreign key constraints into the PostgreSQL kernel, so that the database system can provide native support for them.

Tasks:

1. Formalize ongoing primary and foreign key constraints; this allows the student to develop an algorithm for checking whether the constraint is fulfilled when a modification statement is processed in the database system.
2. Integrate two new keywords 'ongoing primary key' and 'ongoing temporal key' in the parser of PostgreSQL.
3. Integrate ongoing primary key constraints into the PostgreSQL kernel.
   (a) Understand how non-temporal primary key constraints are implemented in the PostgreSQL kernel.
   (b) Propose and implement an algorithm to check an ongoing primary key constraint without using indexes on the time attribute.
   (c) Use the following resolving strategy in case of a violation: reject the modification of the tuple.

4. Integrate ongoing foreign key constraints into the PostgreSQL kernel.
    - Understand how non-temporal foreign key constraints are implemented in the PostgreSQL kernel.
    - Propose and implement an algorithm to check an ongoing foreign key constraint that is not based on indexes for the ongoing primary key.
    - Use the following resolving strategy in case of a violation: reject the modification of the tuple.
5. Empirically evaluate the performance of your approach with a solution that realizes these constraints with user-defined triggers. Define these user-defined triggers for the evaluation.
6. Write the thesis and include the formalization of the two integrity constraints, a precise description of your approach for integrating them into the PostgreSQL kernel, and the evaluation results and your conclusions (approximately 50 pages).
7. Present your thesis in a DBTG meeting (25 minutes presentation)

References

[1] Clifford, James and Dyreson, Curtis and Isakowitz, Tomás and Jensen, Christian S. and Snodgrass, Richard Thomas. On the Semantics of Now in Databases. *ACM Transactions on Database Systems*, 1997.

[2] K. Kulkarni and J.-E. Michels. Temporal Features in SQL:2011. *SIGMOD Record*, 41(3):34–43, 2012.

[3] Wei Li and Snodgrass, R.T. and Shiyan Deng and Gattu, V.K. and Kasthurirangan, A. Efficient sequenced temporal integrity checking. In *Proceedings. 17th International Conference on Data Engineering.*, 2001.

Supervisor: Yvonne Mülle (muelle@ifi.uzh.ch)

Start date: 1st March 2016

End date: 31st August 2016

Presentation date: TBA

University of Zurich
Department of Informatics

Prof. Dr. Michael Böhlen
Professor