Final Research Pape

Shivani Gowda KS

January 18, 2024

Abstract

In this paper, we present a new way of implementing temporal databases in SQLite. We will brief about our methods and motivation, describe the investigation that we have conducted and about the future work that can be done to improve our present approach.

1 Introduction

The research in database community has gotten strong over years. Even tough there is a large amount of progress in the past few years, yet in terms of technology change is always constant. As the amount of data that is been gathered each year is exponentially increasing, we need advanced way to maintain this data in order to gather insights from the data. In the present society most of our decisions are data driven, giving us a new opportunity to think about our decision in a new perspective.

The latest Seattle reportAbadi et al. [2022], shows that there is still some gap in the database community that should be focused on further. In this paper we would be specifically focusing on database auditing. Database auditing is a method which records all the ongoing actions in the database like logging each and every user activities on the database Huang and Liu [2009]. By keeping track of all the activities the database, it detects malicious activities Lu et al. [2013], which is very important in the security of the database system. Even tough it has a huge impacts over the database system and also provide security which is briefed in this paper [Matthew and Dudley, 2015] along with procedure and techniques, we further wanted to look into scenarios where the user who has good intention and who had access/rights to modify the database, but but what if there was a value that was modified several times from a user, and it was all happened by mistake, in that case audit alone will not help.

2 Motivation

The database roll back method Sarda [1993] will roll back the database to one specific time, but we want to track specific value that was modified, and keep rest of the changes as it is. So we think that while modifying a data in the database, we need to keep track of all the updated values like maintaining a temporal database Snodgrass [1992] to refer back to the values. The different form of temporal databases are explained in Elmasri et al. [2000]. From our further studies we realized that PostgreSQL already provides extensions like maintaining history, where when the value in the database is updated, the older value is stored in the history tables. The history tables might not be as same as original table. All of the given approaches uses a new table to store the updated values, keeping this in mind, we came up with a different type of solution to handle temporal table, where we did not want to create a new table, instead handle the updated value in the same table and when the user wanted to derive insights from the updated values, then only we want to create temporary views to handle this request.

3 Methods

We want to first start with SQLite which is a very basic database to start with. For the SQLite database instead of directly connecting to SQLite server, we want to provide a wrapper class around the SQLite like a middle layer. Our previous thoughts was to add the wrapper class, where we wanted

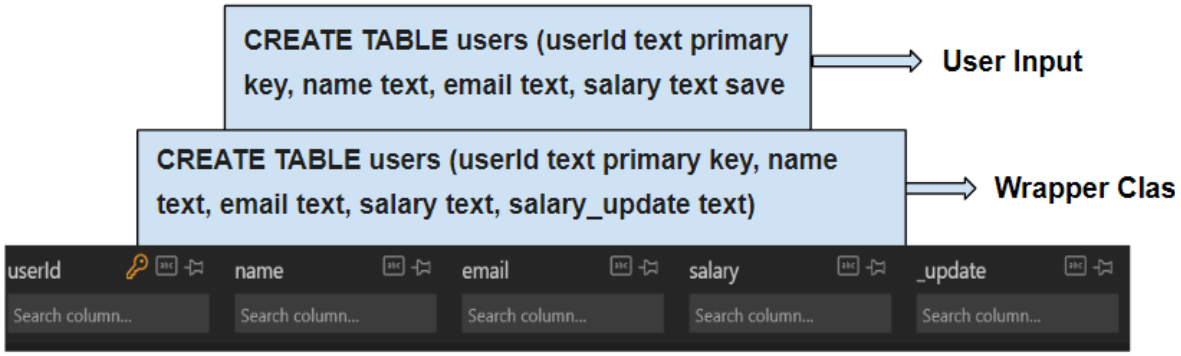

Figure 1: Implementation of the wrapper class.

to store time stamp with its previous value of any values that gets updated, inserted or deleted. Like adding a new dimension that has a list that stores the values and try to keep it as efficient as possible.

But for now we have limited our database where only one value gets updated. So, we started with a small database, assuming that only one column in the table get updated. We want to add one new column to the table, whose datatype is a string, this string gets updated every time the column is updated.

ex:

Id	Name	Salary	salary_update
1	"Bob"	1000	"900:(11/21/2021-11/21/2022),800:(11/21/2020-11/21/2021)"

Table 1: Example database

salary_update is the new column that we will be added automatically, which maintains the previous values along with the time stamp. This whole idea will be implemented in SQLite, which does not provide any type of temporal database extension.

4 Implementation

We implemented our idea in SQLite which implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. We have written the wrapper class in python language. We have added the wrapper class as shown in the Fig. 1, where it converts the given query as needed like adding the new field depending on the user information. The Fig. 1 explains only about the CREATE query, same was followed for SELECT, UPDATE and DELETE.

We have written the documentation on how to use the implementation. The user will have to follow some of the specification given, like adding the special keys for the values that may be updated in the future and follow a specific syntax [1].

5 Result

The database seems to handle well the wrapper class we have implemented. But, during the implementation we realized that we have not covered a case where when the user deletes a value from the database table and want to refer back later. So we extended the wrapper class to handle that as well, but to store this value we will have to create a new database that stores all the deleted values.

[1] https://github.com/sgowdaks/Temporal_tabels_in_sqlite

6 Discussion

Even tough the database seems to handle the queries well, it does not handle the complex query like queries inside the query. The wrapper class was only added to CREATE, SELECT, UPDATE and DELETE and can only handle the basic queries well. The task of writing the wrapper class was harder than we thought, has we had to handle all possible scenarios. Moreover we have not analyzed the complexity of this method with the existing methods. One more limitation of this wrapper class is you can modify only one field, and the field that you choose to modify should be selected at the beginning of the creation of the table, and you have to mention that as the last field while creating, but it is very difficult to predict which values will be modified in the future.

For our future work we want to remove the limitation of predicting the value that will be modified in the future. Also, we want to compare its complexity with the existing methods. We have also not added the views for the modified field, which is to be handheld in our future work.

References

Daniel Abadi, Anastasia Ailamaki, David Andersen, Peter Bailis, Magdalena Balazinska, Philip A Bernstein, Peter Boncz, Surajit Chaudhuri, Alvin Cheung, Anhai Doan, et al. The seattle report on database research. *Communications of the ACM*, 65(8):72–79, 2022.

R Elmasri, Shamkant B Navathe, R Elmasri, and SB Navathe. *Fundamentals of Database Systems¡/Title*. Springer, 2000.

Qiang Huang and Lianzhong Liu. A logging scheme for database audit. In *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering - Volume 02*, IWCSE '09, page 390–393, USA, 2009. IEEE Computer Society. ISBN 9780769538815. doi: 10.1109/WCSE.2009.837. URL https://doi.org/10.1109/WCSE.2009.837.

Wentian Lu, Gerome Miklau, and Neil Immerman. Auditing a database under retention policies. *The VLDB Journal*, 22(2):203–228, apr 2013. ISSN 1066-8888. doi: 10.1007/s00778-012-0282-x. URL https://doi.org/10.1007/s00778-012-0282-x.

Olumuyiwa O Matthew and Carl Dudley. Critical assessment of auditing contributions to effective and efficient security in database systems. In *International Conference on Computer Science, Informatin Technology and Applications (CSITA-2015)*. AIRCC Publishing, 2015.

N.L. Sarda. Time-rollback using logs in historical databases. *Information and Software Technology*, 35(3):171–180, 1993. ISSN 0950-5849. doi: https://doi.org/10.1016/0950-5849(93)90054-7. URL https://www.sciencedirect.com/science/article/pii/0950584993900547.

Richard Snodgrass. Temporal databases. pages 22–64, 09 1992.