Project Progress Report

Weisheng Wang ww2609

Overview

For my project, I'm working on my midterm paper's topic about "database-backed web application performance anti-pattern." In particular, I focus on SQL-based APs and try to extend the original SQLCheck tool by enhancing its AP detection correctness (i.e. reducing its false-positive rate). For the first stage, I will focus on logical design APs such as Multi-Valued Attribute AP and No Foreign Key AP. The overall workflow is to first run SQLCheck on a database schema and collect AP results. Then, run my SQLCheck+ with the results(containing SQL statement and detected APs) and raw data to get the final AP results. Ideally, I will experiment with 5~10 Kaggle database schemas(50~150 CREATE statements), annotate the SQL statements manually, and compare SQLCheck's and SQLCheck+'s AP detection performance over the schemas. For the second stage, if time permits, I will also create query statements for 1~2 datasets from the Kaggle datasets, evaluate SQLCheck's performance on query APs(in my midterm paper, I found that SQLCheck might detect false-positive AP in query statements as well), and extend SQLCheck+ to reduce such false-positive detections. In addition, since SQLCheck provides a regression test suite, I will also use it to compare the two tools' performance.

Value to User Community

Data storage has become one of the most common features for web applications. While software developers design database schemas and program query statements to retrieve data, they may inevitably introduce database-access performance anti-patterns (AP) which not only violate basic design principles and best practices but also bring down the overall system performance. For instance, APs may cause redundant storage and increase applications' response time. Besides, identifying APs and implementing fixes require much developers' effort and are time-consuming. Therefore, many studies aim to tackle the database-access performance AP problem by identifying APs and suggesting fixes.

For my project, I focus on SQL-based APs and intend to enhance SQLCheck's performance. The major problem with the open-sourced SQLCheck tool is that it has a relatively high false-positive rate. After examining the false-positive cases, I find it solvable by implementing additional rule-based approaches and analyzing raw data's contents (data analysis is proposed in SQLCheck, but the open-sourced SQLCheck tool does not support inputting raw data. Besides, SQLCheck uses data analysis to help detect APs. Here, I will modify the SQLCheck's data analysis algorithm and perform it over SQLCheck's results to reduce the false-positive rate.) Therefore, my project aims to develop SQLCheck+, a SQL-based AP detection tool with lower false-positive rates than SQLCheck.

Research Questions

RQ1: What are the AP distributions in the selected Kaggle database schemas?

In the SQL-based AP domain, there has not been a published well-annotated benchmark dataset. So, I'd like to collect the SQL statements, annotate them manually, analyze the AP distribution, and share the dataset for future studies. Analyzing AP distributions is important

because it can help developers to realize what the common APs are and pay more attention to avoid them.

RQ2: How much false-positive rates have dropped using SQLCheck+?
Since my project extends the original SQLCheck tool, I will compare it with my tool and evaluate the performance.

Demo Plan
1 min elevator pitch: introduce the topic and overview of my project
5 min live demo: I will present SQLCheck+ and run it using the cmd line. Then, I will give an example of a false positive case detected by SQLCheck+(with an explanation about why it is a false positive).

Code Delivery
I will be using a public GitHub repository to store the source code and example data. If the datasets are too large, I will use zenodo to store the datasets.

https://github.com/BotMichael/SQLCheck-Plus