Preserving PL/SQL's perks with plain SQL

Abstract

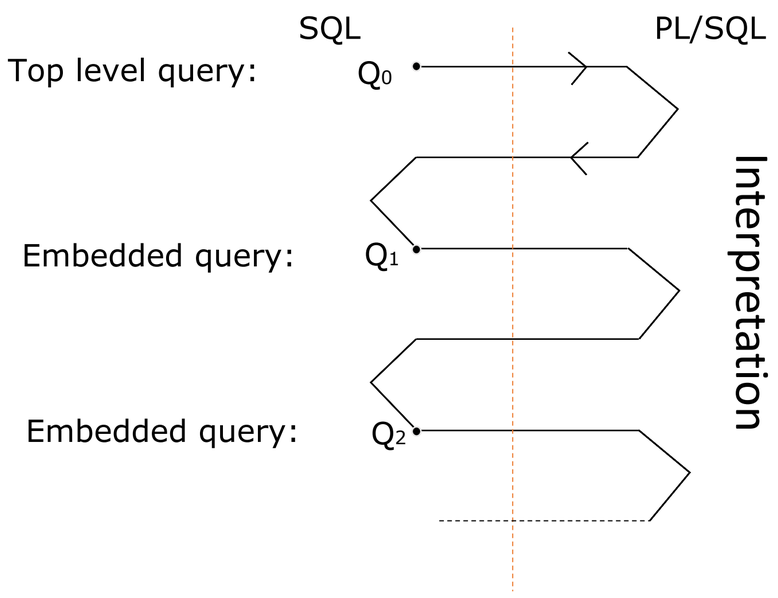
Procedural language for SQL is a procedural extension for SQL which facilitates data processing, but also has performance issues in combination with iteration. This poster aims to describe a source-to-source computation process that translates user defined PL/SQL functions with iterations into plain SQL which express the same functions with recursion to improve computation speed.

Premise

While working with large amounts of data in databases, there are several ways to manage data processing. Conventionally, the raw data is sent from the database to a programming language of choice to process it. Another way to approach this task is to bring the computation closer to the data and only require a chosen programming language to post-process the result.

PL/SQL is a language which utilizes the second approach. It contains elements of programming languages, as well as the use of embedded SQL statements. This enables users to declare variables containing SQL queries. Subsequently, using PL/SQL facilitates the implementation of arbitrary complex control flows containing SQL.

Attributable to the additional functionality of a programming language with SQL, PL/SQL is flawed in terms of its performance when utilizing loops. This is a direct consequence of the context switches between the PL/SQL interpreter and SQL interpreter when iterating.



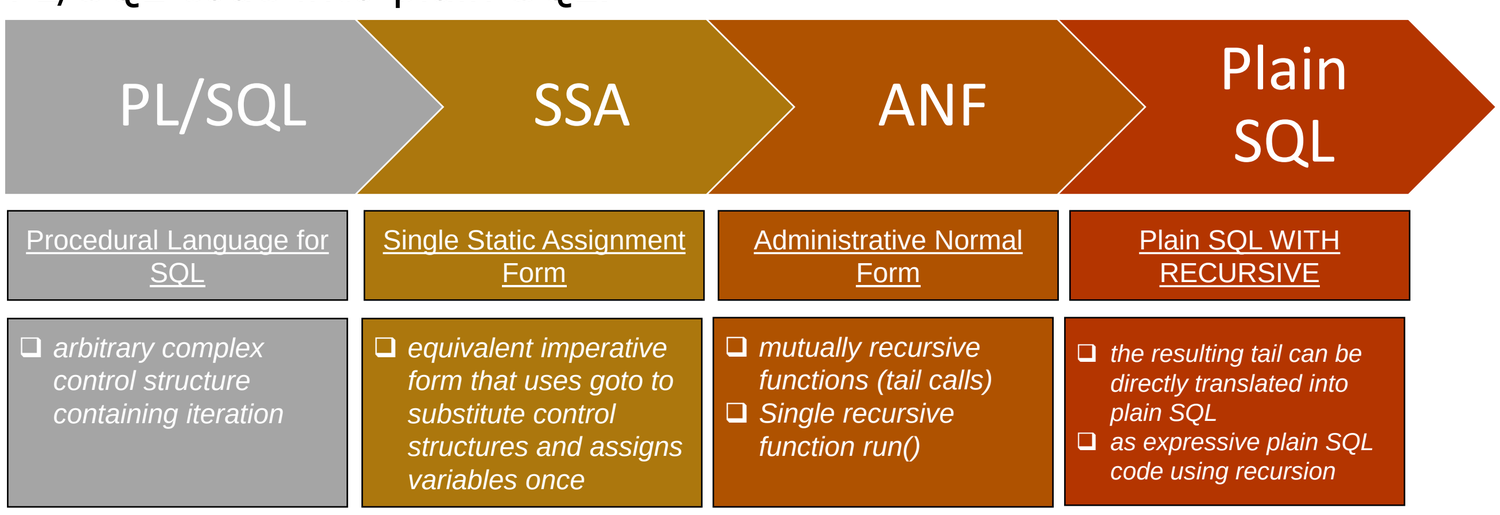
[1]Example of multiple context switches during an arbitrary loop which has embedded queries in PL/SQL, causing poor performance

Removing the context switches

Since the addition of recursion in SQL:1999, SQL is Turing complete. This enables the translation of PL/SQL with iteration into plain SQL utilizing recursion with a drastic reduction in computation due to the removal of context switches. WITH RECURSIVE is a common table expression which can express our desired recursion.

Procedure

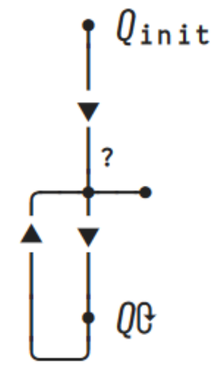
The following displays the computation chain from an arbitrary PL/SQL code into plain SQL.



In the first step, the arbitrary code containing iteration is translated into the Single Static Assignment Form(SSA). The SSA assigns variables once and transforms the input into an equivalent imperative form which utilizes the *goto* statement. After this form, the code taken from the SSA is changed into the Administrative Normal Form(ANF). The ANF has two steps. In the first step of two, the ANF rewrites the given functions to mutually recursively call each other. Following this, the ANF creates a single recursive function *run()* from the mutually recursive functions. The result is a function which can be mapped back to SQL using WITH RECUSIVE, due to the similar calling pattern. This concludes the source-to-source computation and results in equivalently expressive plain SQL code using recursion with a drastically decreased run time.

WITH RECURSIVE

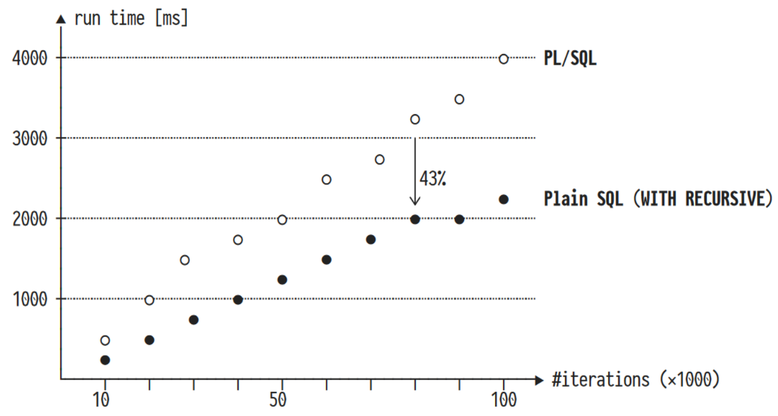
The control flow expressed by SQL's CTE WITH RECURSIVE is restricted. After the query Q_{init} is initialized, $Q_{\circlearrowright}$ is iterated until the iteration cedes no more rows and exits the computation.



[2]WITH RECURSIVE's control flow

SQL vs PL/SQL

SQL provides more efficient runtime with recursion the more complex the equivalent PL/SQL program with iterations is.



[3] PL/SQL vs SQL performance

Resources:

Based off Dr. Torsten Grust's lecture *PL/SQL and UDFs are Lousy - But we can do something about it* held on Dec. 07 2021 during the *Lecture Series on Database Research* at Hasso Plattner Institute

[1] based off the lecture

[2] from page 32 of mentioned lecture

[3] from page 40 of mentioned lecture

Reiner Stolle

Bachelor Computer Science Student at the TU Dresden

Hasso Plattner Institute, Potsdam, Germany

E-Mail: reiner.stolle@mailbox.tu-dresden.de

HPI Hasso Plattner Institut