Supporting Procedural Constructs in SQL Compilers

Nelson Mattos

IBM Santa Teresa Laboratory, 555 Bailey Avenue, San Jose, CA 95141
mattos@vnet.ibm.com

Abstract - **In this presentation, the author will describe an integrated approach that extends existing query compilers to support SQL/PSM-like procedural extensions. This work was a joint development at the IBM Santa Teresa Lab. with Gene Fuh, Jyh-Herng Chow, and Brian Tran who also co-author a more complete publication of this work [FCMT95]. In the presentation, it will be shown how the existing SQL compiler infrastructure can be generalized to accommodate the new procedural constructs and described how one can implement this approach as part of an existing DBMS.**

The draft of the SQL/PSM standard has defined a procedural extension to the existing SQL92 language. An essential part of this extension is the support of procedural constructs such as `BEGIN/END` blocks, local variables, assignment statements, conditional statements, different forms of loops, etc.

Such an extension introduces new challenges to existing SQL compilers. Most (if not all) SQL compilers exiting in the marketplace today were built based on the declarativeness of SQL. The question is how these procedural extensions can be best implemented in a relational DBMS without losing the power of existing query optimization mechanisms. So far, most implementations of the SQL procedural extensions rely on the use of a separate interpreter to handle the procedural statements so that the existing SQL compiler can be left untouched. Although this approach is quite easy to implement (as it follows the paradigm currently used between SQL and host languages), it does not provide the best possible performance.

To achieve the required performance, we have followed a different approach to support the execution of SQL procedural statements. Instead of relying on a separate interpreter, we have extended an existing SQL compiler to handle the procedural extensions in an integrated fashion.

We have observed that SQL/PSM statements are "skeletons" which ultimately specify a well defined execution sequence over a set of SQL92 statements. To be more specific, for instance, the SQL/PSM `while` statement defines a conditional repetition of statements which are encapsulated in the loop body: the loop construct is basically a control skeleton gluing the other statements. A natural consequence following this observation is that every SQL/PSM statement can be handled by an extended SQL compiler in the following way: each SQL non-procedural statement can be extracted and compiled into an execution plan by the existing SQL query compiler with minimum enhancements (except for extensions to deal with the nesting of scopes of local variables and the support of global optimizations); the control skeleton is translated into an abstract representation by a new compiler component; finally, the abstract representation of the control skeleton is used by the code generator to synthesize the plan of the control statement with those of the non-procedural statements and produce a single plan. Local variables can be treated as internal host variables that do not involve any data movements or type conversions and have scopes associated with them.

The proposed approach has the following advantages. First, it minimizes the impact on the existing query compiler. Second, it results in efficient execution plans. Since all statements are compiled into a single plan, there is no communication overhead between the SQL interpreter and the interpreter for the procedural language, no unnecessary data movement between the two interpreters, data conversions, and/or bind-in/bind-out operation taking place during the execution. Third, it facilitates the use of modern optimization technologies from traditional programming languages, and provides a general framework for global optimization across query and control boundaries.

References

[FCMT95] Gene Fuh, Jyh-Herng Chow, Nelson Mattos and Brian Tran. *Extending SQL Query Compiler for Control Statements.* IBM Technical Report TR03.626. Santa Teresa Lab, August 1995.

Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996