Tam Yuen
Columbia University

Programming Languages and Translators
COMS-W4115

Project: ty

Abstract

The legacy of SQL has presented a need to improve its verbose syntax. By offering simpler syntax, ty can reduce the amount of time in composing traditional SQL statements.

1. Brief Introduction to DBMS and SQL

A database management system (DBMS) is a piece of software that is designed to assist in maintaining and utilizing large collection of data stored in its database. To retrieve data stored in a DBMS, a query language is needed to retrieve records from its database. The structure query language, or SQL, was originally developed as the query language for IBM's System-R relational DBMS in the mid-1970s, and is now the de-facto standard query language for creating, manipulating, and querying relational DBMSs. SQL was standardized in the late 1980s, and the current standard, SQL 1999, was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).

Relational DBMSs store data in a table, which can be thought of as a set of records with rows and columns. In order to interact with the database, users issue SQL commands against the database. For example, commands such as "select empid from emp" and "select lastname, firstname from person where empid=123" query the database to find all the employee IDs from the emp table and find the last and first name of the person whose employee ID is 123, respectively.

The SQL language is structured in several components, but the two components that deal with creating, manipulating, and querying are the Data Definition Language (DDL) and the Data Manipulation Language (DML). The DDL supports the creation, deletion, and modification of definitions for tables; while the DML allows users to pose queries and to insert, delete, and modify rows.

2. SQL Dilemmas

SQL was invented several decades ago and because of its early and rapid adaptation, SQL was widely deployed to many DBMSs. Today, many enterprise DBMSs use SQL or some variants of SQL as their query language. Although SQL has been in usage for many years, its syntax has remained relatively unchanged since its inception.

Even though SQL is a powerful and versatile query language, however, it has several undesirable attributes due to its inherited legacy from the early 1970s. The first of such attribute is the verbosity of the SQL language; SQL queries can often be overwhelmingly verbose or lengthy. Common queries such as "select empid from emp" and "select lastname, firstname from person where empid=123" are often unnecessarily long or wordy.

For example, one of the most frequently used SQL commands is joining of tables. Presently, the join command in SQL is unnecessarily verbose. Because joining tables are some of the most widely used query commands in SQL, therefore, it is important to simplify the syntax of joins, particularly natural joins. Without simplification to SQL users need to spend more time on typing SQL commands rather executing some other activities.

SQL's syntax is also very lengthy in arithmetic. Arithmetic between columns in one or more tables can sometimes be unnecessarily verbose because names of columns need to be specified. As a result, typing mathematic SQL commands can also be very a time-consuming activity.

All of these undesirable attributes of the SQL query language can be mitigated by using ty. ty can be used to simplify frequently used SQL commands with shorter and simpler syntax.

3. Introduction to ty

ty, pronounced "tie" or just "t y", is named after its creator, Tam Yuen. ty can address some of SQL's dilemmas by simplifying frequently used DML and DDL query commands into shorter and easy-to-use commands. In contrast to SQL, ty's syntax is simpler, easy to remember and to use—yet also powerful and versatile. For example, SQL commands such as "select empid from emp" and "select lastname, firstname from person where empid=123" can be easily rewritten by more compact commands like "ty emp" and "ty emp(1, 2, empid=123)", respectively.

ty is not meant to replace traditional SQL, but rather compliment it by offering simplified syntax for frequently used SQL commands; in fact, queries composed using ty can be completely recomposed using SQL—but not vice versa. ty, in essence, can be thought of as a special component of SQL that helps to simplify certain types of query. Therefore, by using ty, users can experience less time on constructing traditional SQL commands and more time on other higher value-adding activities.

4. Comparing ty to SQL

The following examples highlight some of ty's syntax by comparing them to traditional SQL statements.

A. ENTITY TABLE:

– return all rows from emp table
ty: ty emp
SQL: select * from emp

– return row 1,2,3 from emp table
ty: ty emp(1,2,3)
SQL: select empid, fname, lname from employee

B. TWO ENTITY TABLES (Natural Join):

– return all rows from naturally joined tables emp and person
ty: ty emp, person
SQL: select * from emp, person where emp.empid=person.empid

– same as above, but with explicit primary key column number
ty: ty emp(1), person(1)
SQL: select * from emp, person where emp.empid=person.empid

C. RELATIONAL TABLES

– return all rows from tables emp with empid as primary key
ty: ty emp(empid), person(empid), salary(empid)
SQL: select * from emp, person, salary where empid.empid=person.empid and person.empid=salary.empid

– return column1 of emp table, column4 of person table, and column2 of salary table with empid as the primary key
ty: ty emp(1), person(4), salary(2)
SQL: select emp.column1, person.column4, salary.column2 where emp.empid=person.empid and person.empid=salary.empid

D. CONCATENATION:

– returns two columns, with the first being the concatenation of column1 and column2
ty: ty emp(1||2,3)
SQL: select column1 || column2, column3 from emp

E. ADDITION

– addition of 2 columns; default, formal decl with add command
ty: ty emp(1) + emp(4)
SQL: select emp.column1 + emp.column4 from emp

– addition of 2 columns from 2 different tables
ty: ty emp(1) + person(3)
SQL: select emp.column1, person.column3 from emp, person

F. SUBTRACTION

ty: ty emp(1) - emp(4)
SQL: select emp.column1 - emp.column4 from emp

ty: ty emp(1) – person(3)
SQL: select emp.column1 – person.column2 from emp, person

G. MULTIPLICAITON

ty: ty emp(1) * emp(4)
SQL: select emp.column1 * emp.column4 from emp

ty: ty emp(1) * person(3)
SQL: select emp.column1 * person.column2 from emp, person

H. DIVISION

ty: ty emp(1) / emp(4)
SQL: select emp.column1 / emp.column4 from emp

ty: ty emp(1) / person(3)
SQL: select emp.column1 / person.column2 from emp, person

I. AVERAGE

ty: ty avg emp(1) -- average for column 1
SQL: select avg(emp.column1) from emp

5. Future Improvement and Upgrade

Due to time constraints, only some SQL commands have been simplified. These initial improvements should serve as a springboard for future enhancement and upgrade. Possible enhancements and upgrades would include extensive implementation of other SQL DDL and DML commands.

6. Summary

Although SQL is a powerful and versatile query language, ty can compliment SQL by offering simpler syntax for frequently used SQL DDL and DML commands.