CMPSC380
Principles of Database Systems, Fall 2014

Final Project: Advanced Topics in Data Management

Purpose: DeeBee: Implementing A Database Management System

Pledge:

Hand in: Wednesday, November 19th, 2014

Abstract: DeeBee is a small database management system implemented for educational purposes. It will implement a minimal subset of the structured query language; enough to support simple database operations. DeeBee will be designed in a modular fashion to facilitate the iterative construction of a complete system.

1 Introduction

In order to learn more about how a database management system (DBMS) functions, I intend to embark into the implementation of a small database, called 'DeeBee'. While the implementation of a production-quality DBMS is a significant undertaking requiring a great deal of time and the concerted efforts of many programmers, the implementation of a minimal DBMS that supports a subset of the structured query language should be an accomplishable task. DeeBee will be designed in a modular fashion, in order to support the implementation of various database features in an iterative fashion.

2 Architecture

DeeBee will follow an architecture similar to the one presented in Figure 1.5 in *Database System Concepts* [1, p. 24], with several significant simplifications. DeeBee will not support any form of query optimization, so parsed SQL queries will be interpreted directly, rather than compiled, organized, and optimized. Furthermore, authorization and transactions will not be supported, with the intent of allowing access by only a single user for demonstration and testing purposes.

Additionally, DeeBee will initially store data on disk using a comma-separated values file, to make the implementation of the storage management system less difficult, and to facilitate the simple testing of the persisted files. While I am interested in pursuing the implementation of a B+ tree for managing storage on disk in the manner used by many popular DBMSs, the implementation of a B+ tree is a significant task on its' own. I may choose to undertake this task at a later point, so DeeBee's storage manager will be designed in a modular fashion to permit the swapping out of storage backends.

3 Implementation

DeeBee will be implemented using the Scala programming language. I have already configured a Git repository (`https://github.com/hawkw/deebee`), Travis continuous integration system (`https://travis-ci.org/hawkw/deebee`, and build toolchain, using the `sbt` build system, to support my development efforts.

3.1 Testing

DeeBee's performance will be assessed by test suites written using the ScalaTest testing framework. Coverage data will be collected using the Scoverage tool. These tests will be written before a majority of implementation takes place, according to the principles of test-driven development, and will act as the specification or requirements document for DeeBee.

3.2 Implementation Plan

I intend to begin by defining the core interfaces of DeeBee, defining the internal API that will allow the system's components to communicate. Once I have defined these interfaces, I will use their method definitions to write a test suite for DeeBee's core behaviour. This test suite will initially fail, as all of the methods will be unimplemented. Following the principles of test-driven development, I will then use the test suite as a 'road map' to implement the core DeeBee features.

I intend to begin by developing a simple storage manager that persists data using a CSV file. I will then begin implementing a query-processing system capable of parsing and executing a subset of SQL queries. The initial supported subset will likely consist of the `SELECT` (probably excluding joins), `INSERT`, and `DELETE` statements, as well as a simple `CREATE TABLE` statement.

Once all the tests pass, if there is time remaining, I will perform another iteration by writing failing tests for additional functionality, and then begin implementing those features. I may begin working on a B+ tree implementation at this point, or add support for additional SQL features.

References

[1] A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill Education, 2010.