Database Management System Project

Subject: Object-Oriented Programming

Student: OM AMAR (BT23CSE106)

ASHISH TUKARAM PAKHLE (BT23CSE107)

DEBASISH MONDAL (BT23CSE108)

TANAY SUNIL UMRE (BT23CSE109)

Instructor: DR MILIND PENURKAR

Introduction

This document provides an overview of a Database Management System (DBMS) project implemented in C++ using object-oriented programming principles. This project demonstrates how OOP concepts such as encapsulation, inheritance, and polymorphism can be applied to create a simple DBMS that can handle basic SQL-like queries, including CREATE, INSERT, SELECT, UPDATE, DELETE, and COPY. The code manages data using CSV files and operates via a command-line interface.

Project Goals

The primary goal of this project is to simulate a lightweight DBMS to execute simple SQL commands using object-oriented programming in C++. The DBMS is designed to store data in CSV files and enable basic data manipulation commands.

Object-Oriented Programming Concepts Used

This project employs several OOP principles:

1. Encapsulation: Data and functions are bundled within classes to protect and organize the code.
2. Inheritance: The DBMS structure doesn't explicitly use inheritance but leverages modular classes.
3. Abstraction: Complex database operations are abstracted into high-level methods.
4. Polymorphism: Not directly implemented but demonstrated through different query types in a single parser.

Class Descriptions

1. Table Class

The `Table` class represents a database table, which stores column definitions and data in rows. Each table is saved as a CSV file, making data persistent.

Key Attributes:

- name: Name of the table.
- columns: List of column names.
- columnTypes: Data types for each column.
- rows: 2D vector holding row data.

Key Methods:

- writeToFile(): Saves table data to a CSV file.
- addRow(): Adds a new row to the table.
- deleteRows(): Deletes rows matching a condition.
- selectRows(): Selects rows based on a condition.
- updateRows(): Updates rows based on a condition.

2. Database Class

The `Database` class is a container for tables, allowing the creation, deletion, and access of tables.

Key Attributes:

- tables: Unordered map of table names to `Table` objects.
- count: Static variable tracking the total number of tables.

Key Methods:

- createTable(): Creates a new table with specified columns.
- loadExistingTables(): Loads tables from existing CSV files.
- deletetable(): Deletes a table from the database.
- print(): Displays a list of available tables.

3. ParsedQuery Struct

`ParsedQuery` is a struct used to store parsed command data, including command type, table name, columns, values, and conditions.

4. QueryParser Class

The `QueryParser` class interprets commands, converts them into `ParsedQuery` objects, and executes operations on the database.

Key Methods:

- parse(): Analyzes a command string, identifies the operation, and stores it in `ParsedQuery`.
- execute(): Uses a `ParsedQuery` to perform the corresponding operation on the database.

Functionality

The DBMS supports multiple commands similar to SQL operations:
- CREATE: Creates a new table.
- INSERT: Inserts a row into an existing table.
- SELECT: Queries data based on specific conditions.
- UPDATE: Modifies rows matching a condition.
- DELETE: Removes rows matching a condition or deletes a table.
- COPY: Duplicates an existing table.

Code Walkthrough

The `main()` function initializes the database and enters an interactive loop where the user can input commands. Commands are parsed by the `QueryParser`, which then executes the desired operation on the database.

Error Handling and Edge Cases

The code includes exception handling for common errors such as invalid commands, missing tables, or column mismatches. The program displays error messages for user-friendly feedback.

Conclusion

This project demonstrates the use of object-oriented programming to create a simple DBMS that can execute basic SQL-like commands. By organizing database components into distinct classes, we achieve modularity, making the code easier to extend and maintain.