SE 3XA3: Software Requirements Specification
GoDBMS

Team #7, Databased
Eesha Qureshi, qureshe
Faiq Ahmed, ahmedf46
Kevin Kannammalil, kannammk

April 12, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 17	1.0	Started assignment
March 17	1.1	Created Modules
March 18	1.2	Created Modules Heirarchy Diagram
March 18	1.3	Finished Document
April 12	1.4	Updated module secrets and added new module

1 Introduction

GoDBMS is a database management software based on the project SimpleDB. The aim of GoDBMS is to develop a software that maintains the basic functionality of SimpleDB but has a lighter weight, optimizes memory usage, has a simpler code base and is developed using GoLang. This project will be properly documented and iteratively developed following the Software Development Life Cycle.

1.1 Purpose

The purpose of this document is to provide insight into the modular decomposition of the system. This will allow for developers and other intended audience to understand each component in the system and how it connects in the system architecture, the module hierarchy, and information hiding between modules.

1.2 Scope

This document provides detail about the modular decomposition and relational hierarchy of the modules that comprise the system. In contrast, the MIS will provide specifications about the behaviour of each module and component.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Revision History**

Term	Abbreviation	Definition
Database Management System	DBMS	Software used to manage database transactions and operations
Structured Query Language	**SQL**	Language used to write database transaction instructions
Command Line Interface	**CLI**	Point of contact between user and computer instructions
Catalog	N/A	A list storing the names of all the tables in the database. This list can be encoded and decoded from a file to allow persistent storage.
Record	N/A	A single row of data in a database
Heap File	N/A	A basic data structure that uses memory blocks and stores new files at the end
Encoder	N/A	Resolves the mapping between items
Lock	N/A	Data variables that synchronize concurrent transactions
HTTP Server	N/A	Information transfer protocol for the internet

1.4 Overview of Document

This document will highlight anticipated changes to the system, as well as unlikely changes. Then it will delve into the module hierarchy, followed by comparisons between the requirements and design. Next it will present the modular decomposition of the system in great detail. Finally, this document will provide a traceability matrix and summarize the use hierarchy between modules.

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2.

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The directory where the **DBMS** is installed

AC2: The method to store the **record**s

AC3: The method of interfacing with the **DBMS**

AC4: The styling of the command line interface and displayed messages

AC5: The supported **SQL** commands in the **DBMS**

AC6: The supported **SQL** data types in the **DBMS**

AC7: The structure of the **catalog**.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices since interacting with a database only requires the use of a keyboard.

UC2: The encoding and decoding of **DBMS** persistent data.

UC3: The **DBMS** will always take in strings as input queries.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 3. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: CLI

M2: Controller

M3: Parser

M4: Parse Modify **Record**

M5: Parse Insert **Record**

M6: Parse Create Table

M7: Parse Delete Table

M8: Parse Delete **Record**

M9: Parse Select

M10: Process **SQL** Statements

M11: Storage Lock

M12: Storage Encoder

M13: Heap File

M14: **Catalog**

M15: **Catalog** Encoder

M16: File Management

M17: Parser Structs

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 4.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas. The Secrets field in a module decomposition is a brief statement of the design decision hidden by the module. The Services field specifies what the module will do without documenting how to do it. For each module, a suggestion for the implementing software is given under the Implemented By title. If the entry is OS, this means that the module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	**CLI** Parser Storage Encoder **Catalog** Encoder File Management
Software Decision Module	Controller Parse Modify **Record** Parse Insert **Record** Parse Create Table Parse Delete Table Parse Delete **Record** Parse Select Process **SQL** Statements Storage Lock Heap File **Catalog** Parser Structs

Table 3: Module Hierarchy

5.1 Hardware Hiding Modules

N/A

5.2 Behaviour-Hiding Module

5.2.1 CLI (M1)

Secrets: Method to get user input and print back output to the user.

Services: Gets the user input from a the command line and sends it to the **DBMS** using a http request. Prints back the output of the request back to the user.

Implemented By: Go**DBMS**

5.2.2 Parser (M3)

Secrets: Method to retrieve appropriate query fields for each statement from a string input.

Services: Takes a string input and parses the necessary information from the string that is specific to the **SQL** statement.

Implemented By: GoDBMS

5.2.3 Storage Encoder (M12)

Secrets: Method to convert a **heap file** into a byte array and convert a byte array into a **heap file**.

Services: Loads or saves a Heap File.

Implemented By: GoDBMS

5.2.4 Catalog Encoder (M15)

Secrets: Method to convert a **catalog** array into a byte array and convert a byte array into a **catalog** array.

Services: Loads or saves the **Catalog**.

Implemented By: GoDBMS

5.2.5 File Management (M16)

Secrets: Path of directory for save files, methods to load and save files.

Services: Is able to save a byte array into a data file in the save files directory and load a byte array from a data file in the save files directory.

Implemented By: GoDBMS

5.3 Software Decision Module

5.3.1 Controller (M2)

Secrets: Method to get the type of **SQL** command from string input and the connection between the parser and processing **SQL** statements.

Services: Gets a string value from the **HTTP Server** and gets the correct command type before sending it to the appropriate parser function. Takes the return value of the parser function and sends the relevant information to be processed. This module also returns all outputs and errors back to the **HTTP Server**.

Implemented By: GoDBMS

5.3.2 Parse Modify Record (M4)

Secrets: Method to convert modify **record** query string into a struct ~~Modify **Record** Statement Information~~

Services: This module has a constructor for the modify **record** structure which contains all the relevant information about the modify **record** statement.

Implemented By: GoDBMS

5.3.3 Parse Insert Record (M5)

Secrets: Method to convert insert **record** query string into a struct ~~Modify Insert Statement Information~~

Services: This module has a constructor for the insert **record** structure which contains all the relevant information about the insert **record** statement.

Implemented By: GoDBMS

5.3.4 Parse Create Table (M6)

Secrets: Method to convert create table query string into a struct ~~Create Table Statement Information~~

Services: This module has a constructor for the create table structure which contains all the relevant information about the create table statement.

Implemented By: GoDBMS

5.3.5 Parse Delete Table (M7)

Secrets: Method to convert delete table query string into a struct ~~Delete Table Statement Information~~

Services: This module has a constructor for the delete table structure which contains all the relevant information about the delete table statement.

Implemented By: GoDBMS

5.3.6 Parse Delete Record (M8)

Secrets: Method to convert delete **record** query string into a struct ~~Delete **Record** Statement Information~~

Services: This module has a constructor for the delete **record** structure which contains all the relevant information about the delete **record** statement.

Implemented By: GoDBMS

5.3.7 Parse Select (M9)

Secrets: Method to convert select query string into a struct ~~Select Statement Information~~

Services: This module has a constructor for the select structure which contains all the relevant information about the select statement.

Implemented By: GoDBMS

5.3.8 Process SQL Statements (M10)

Secrets: The logic to modify the storage of the database appropriately according to each **SQL** statement.

Services: Modifies the storage of the database and the **catalog** according to which function in this module is called and what parameters are passed in.

Implemented By: GoDBMS

5.3.9 Storage Lock (M11)

Secrets: The semaphore data structures **lock**s for each table.

Services: Allows a **lock** for a table to be acquired or released.

Implemented By: GoDBMS

5.3.10 Heap File (M13)

Secrets: Data structure and logic for all **record**s in the table String ~~**record**s that are in the current table~~

Services: Saves new **record**s, removes **record**s, modifies **record**s, or allows searching for **record**s.

Implemented By: GoDBMS

5.3.11 Catalog (M14)

Secrets: Data structure and logic for all tables in the database ~~List of all tables in the database and how they are stored.~~

Services: Saves new tables, deletes tables, and allows searching for table information.

Implemented By: GoDBMS

5.3.12 Parser Structs (M17)

Secrets: Data structure to hold all parsed query inputs

Services: Structs with appropriate fields holding information parsed from input queries

Implemented By: GoDBMS

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	**Modules**
BE1FR1, BE2FR1, BE3FR1, BE4FR1, BE5FR1, BE6FR1, BE7FR1, BE8FR1	M1, M2
BE1FR2, BE2FR2, BE3FR2, BE4FR2, BE5FR2, BE6FR2, BE7FR2	M3 M17
BE1FR3	M6
BE2FR3	M7
BE3FR3	M2
BE4FR3	M5
BE5FR3	M4
BE6FR3	M8
BE7FR3	M9
BE1FR4, BE1FR5, BE2FR4, BE4FR4, BE4FR5, BE5FR4, BE5FR5, BE5FR6, BE6FR4, BE7FR4	M10, M13, M12, M14, M15, M16
BE1FR6, BE1FR8, BE2FR5, BE2FR7, BE3FR4, BE3FR5, BE4FR6, BE4FR8, BE5FR7, BE5FR9, BE6FR5, BE6FR7, BE7FR5, BE7FR6	M1, M2, M10
BE1FR7, BE2FR6, BE4FR7, BE5FR8, BE6FR6	M12, M13, M14, M15, M16
BE8FR2, BE8FR3, BE8FR4, BE8FR5	M11

Table 4: Trace Between Requirements and Modules

AC	Modules
AC1	M16
AC2	M13
AC3	M1
AC4	M1
AC5	M3, M10, M17
AC6	M13
AC7	M14

Table 5: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas said of two programs A and B that A uses B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A uses B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 2 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

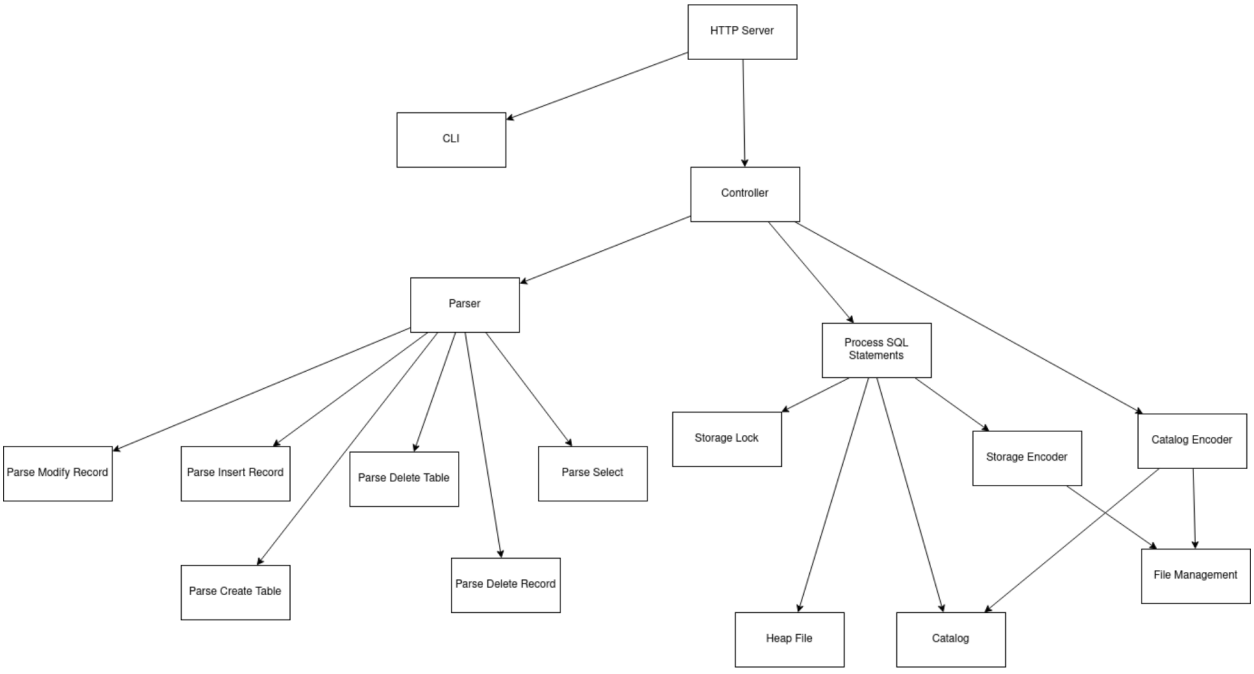

Figure 1: ~~Use hierarchy among modules: Old~~

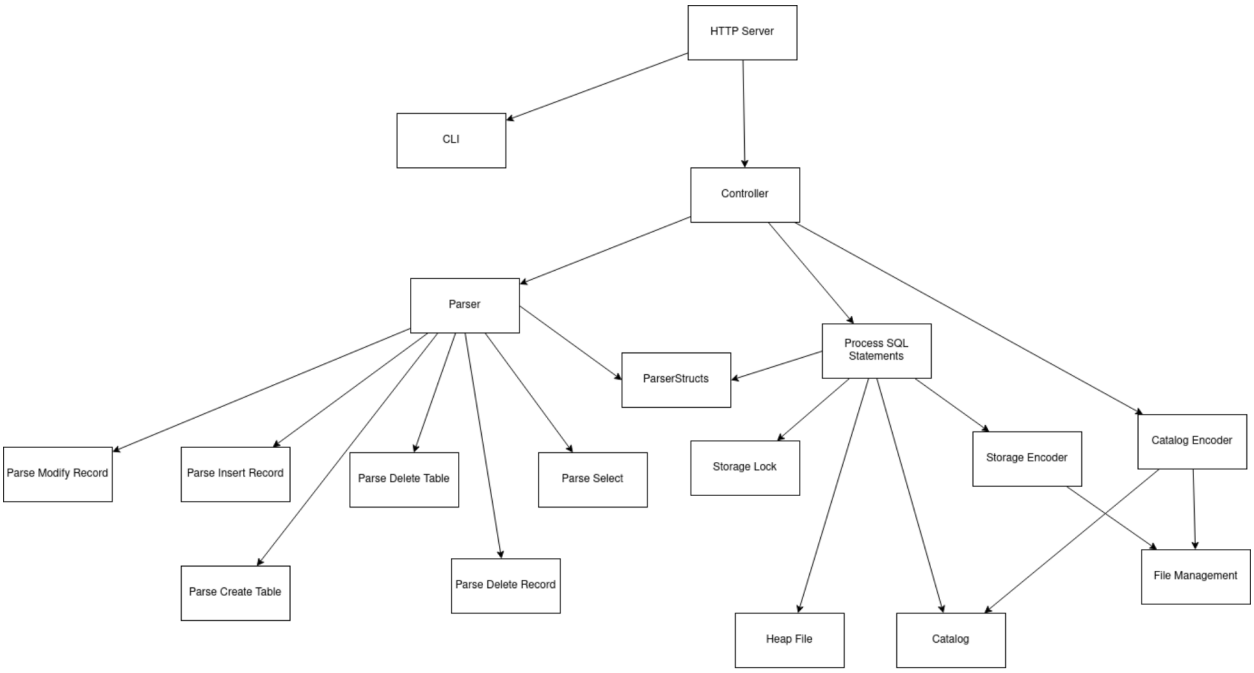

Figure 2: Use hierarchy among modules