3XA3 Module Interface Specification
GoDBMS

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

March 18, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 16, 2022	1.0	Initial Document
March 17, 2022	1.1	Wrote MIS for half the modules
March 18, 2022	1.2	Finished MIS for all modules
April 12, 2022	1.3	Modified MIS and added new modules

CLI Module

Module

CLI

Uses

Http Server

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
~~StartCLI~~ main			

Semantics

State Variables

N/A

Environment Variables

Keyboard
Monitor screen

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

~~StartCLI()~~ main(): Take user input from the keyboard directly from a command line continuously and convert the string to lowercase. If this string is "quit" then immediately exit the loop, otherwise send this string to the http server with a post request. The returned string response from the http server is then printed our to the monitor screen.

Local Functions/Constants

N/A

Controller Module

Module

Controller

Uses

Http Server, Parser, Process SQL Statements, Catalog Encoder

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
InitializeCatalog StartDBMS			

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

The CLI has been started to be able to send inputs to the controller through the http server

Access Routine Semantics

startDBMS(): Receives input from the http server as a string. If the string starts with "update", send the string to Parser.ParseModifyRecord(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessModifyRecord(). Otherwise if the string starts with "insert", send the string to Parser.ParseInsertRecord(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessInsertRecord(). Otherwise if the string starts with "create", send the string to Parser.ParseCreateTable(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessCreateTable(). Otherwise if the string starts with "drop table", send the string to Parser.ParseDeleteTable(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessDeleteTable(). Otherwise if the string starts with "delete", send the string to Parser.ParseDeleteRecord(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessDeleteRecord(). Otherwise if the string starts with "select", send the string to Parser.ParseSelect(), if an error is not returned, send the values of the returned output struct to ProcessSQLStatements.ProcessSelect(). Otherwise if the string is "shut down", call the CatalogEncoder.EncodeCatalog() function and return. Otherwise return an error stating that the query was invalid. If an error is returned by Parser or ProcessSQLStatement at any point, immediately send the error back to the http server as a response and return. If no error is returned, send a success message or the output of the ProcessSQLStatement function back to the http server.

InitializeCatalog(): Call the CatalogEncoder.DecodeCatalog() function.

Local Functions/Constants

N/A

Process SQL Statements

Module

Process SQL Statements

Uses

Storage Lock, Storage Encoder, Heap File, Catalog, Parser Structs.

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out
ProcessCreateTable	ParserStructs.CreateTableStatement struct	error
ProcessInsertRecord	ParserStructs.InsertTupleStatement struct	error
ProcessSelect	ParserStructs.SelectStatement struct	String, error
ProcessDeleteTable	ParserStructs.DeleteTableStatement struct	error
ProcessDeleteRecord	ParserStructs.DeleteTupleStatement struct	error
ProcessModifyRecord	ParserStructs.ModifyTableStatement	error
ProcessModifyRecord	ParserStructs.ModifyTupleStatement	error
ListAllTables		String

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ProcessCreateTable(struct): Receives the fields of the statement from the parameter and does the error handling. It checks if the table name already exists, if so it returns an error. After the error handling, passes the fields into the InsertTable function in the Catalog module. It then gets a heap pointer when it creates a heap for the table using the Heap File module and uses the heap pointer to encode the heap into a file. The function returns nil to indicate there are no errors.

ProcessInsertRecord(struct): Receives the fields that represents the insert statement to store a record into the database. It validates the query with multiple checks to ensure it follows the constraints. It decodes the table from the heap using Storage Encoder. It does error handling by checking if the table it is inserting into exists, verifies the primary key validity and duplicate record. Then it extracts the values from the insert statement and passes it to the Heap File to create the record and store in the storage. Finally it encodes the heap again once it's done with the process.

ProcessSelect(struct): Receives the table name and a set of conditions to retrieve the records in that table. It decodes the heap to access the records, filters the records to what is desired and returns it.

ProcessDeleteTable(struct): Receives the table name to delete from the database. It sends the table name to the Storage Encoder module to delete the file and then erases the table in Catalog and the lock in Storage Lock.

ProcessDeleteRecord(struct): Receives the name to retrieve the heap, then it uses the primaryKeyIndex to delete the record from the heap. It then encodes the heap and stores it back into a file.

ProcessModifyTable(struct): Uses the struct to get table and columns to modify before making the appropriate modification in the heap of that table and the catalog.

ProcessModifyRecord(struct): Uses the table name and primary key index to find the record using Catalog and then modify the value of the column in that record.

ListAllTables(): Retrieves the table map from Catalog to and returns the existing tables as an array

Local Functions/Constants

N/A

Storage Encoder Module

Module

Storage Encoder

Uses

File Management

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
EncodeHeap	HeapStruct	error	
DecodeHeap	name	HeapStruct	
~~DeleteHeap~~	~~name~~	~~error~~	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

EncodeHeap(heap): Receives the heap pointer to write it into the file. It returns any errors that it encounters when writing to the file.

DecodeHeap(name): Receives the table name and checks if the file exists for that table name, if not it returns an error. It reads the file and saves the data onto a heap which it loads into the memory. It returns the heap pointer to be used in other modules.

~~DeleteHeap(name): Receives the table name to retrieve the heap and delete the file using the File Management module.~~

Local Functions/Constants

N/A

Storage Lock

Module

Storage Lock

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
IntializeLocks			
AcquireTableLock	name		
ReleaseTableLock	name		
~~DeleteLock~~	~~name~~		
AcquireCatalogLock	name		
ReleaseCatalogLock	name		

Semantics

State Variables

TableLocks: map of locks for their corresponding table
CatalogLock : lock for the catalog

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

InitializeLocks(): Initializes the catalog and take locks so they can be used

AcquireTableLock(name): Creates a lock for a specific table using the name and stores it into *TableLocks*, so it can be accessed by other modules.

ReleaseTableLock(name): Retrieves the lock from the TableLocks map based on the table name and releases the lock.

~~DeleteLock(name): Receives the name to remove the lock from the TableLocks map which would be deleting it.~~

AcquireCatalogLock(): Acquires the lock for the catalog so no other transaction can modify the catalog at the same time

ReleaseCatalogLock(): Releases the lock for the catalog

Local Functions/Constants

N/A

CatalogEncoder

Module

CatalogEncoder

Uses

Catalog, File Management

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
EncodeCatalog DecodeCatalog			

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

EncodeCatalog(): Calls function Catalog.GetTablesMap() to get a reference to the catalog which is then converted into a byte array before being saved passing in the byte array to FileManagement.WriteByteFile().

DecodeCatalog(): Passes in the string "catalog" to FileManagement.WriteByteFile() get a byte array which is then converted to a catalog reference before being passed into Catalog.StoreTableMap() to load the catalog into memory.

Local Functions/Constants

N/A

File Management

Module

File Management

Uses

Syntax

Exported Constants

directory string

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
FileExists	string	bool	
ReadByteFile	string	seq of byte, error	
WriteByteFile	string, seq of byte	error	
DeleteFile	string	error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

FileExists(name): Looks too see if file by the specified input name exists in the data directory. If directory does not exist or file does not exist, return false. Otherwise return true.

ReadByteFile(name): Checks to see if the data directory exists, if it does not exist, it creates it using the initializeDirectory() local function. The function then looks for a file by the specified input name, if it is able to find the file it reads the bytes from the file and returns the byte array.

ReadByteFile(name, bytes): Checks to see if the data directory exists, if it does not exist, it creates it using the initializeDirectory() local function. The function then create a new file with the specified input name and adds the specified input bytes to it before saving the file.

DeleteFile(name): Looks to see if file by the specified input name exists in the data directory. If the directory and file exist, then it deletes the file with the specified input name.

Local Functions/Constants

initializeDirectory(): Create a new directory in location specified by the directory constant.

Catalog

Module

Catalog

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
LoadTablesMap	map of structs		
GetTablesMap		map of structs	
TableExists	string	bool	
InsertTable	struct		
GetTable	string	struct	
DeleteTable	string		

Semantics

State Variables

catalog: map of structs

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

LoadTablesMap(newCatalog): This methods sets the catalog state variable to newCatalog.

GetTablesMap(): This methods returns the catalog state variable.

TableExists(name): This method checks if the catalog map state variable has a key with the specified name in it. If it does it return true, else it returns false.

GetTable(name): This method gets the struct stored at key name in the catalog state variable and returns it.

Delete(name): This method removes the entry by with the specified key name from the catalog state variable.

Local Functions/Constants

N/A

Heap File

Module

Heap File

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

HeapStruct

Exported Access Programs

Routine name	In	Out	Exceptions
InitializeHeap		this instance of HeapStruct	
InsertTuple	Tuple struct		
TupleExists	any type, integer	bool	
GetTuple	any type, integer	Tuple struct	
~~ModifyRecord~~	~~integer, any type, seq of any type~~		
DeleteTuple	any type, integer	error	
GetHeap		seq of Tuple struct	

Semantics

State Variables

heap: seq of seq of any type

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

InitializeHeap(): Set heap to an empty array and return the HeapStruct.

InsertTuple(tuple): Setter method on the HeapStruct that adds tuple to current heap in current instance.

TupleExists(value, keyIndex): This method checks if the specified value exists in the keyIndex index of any nested array in the heap of the current instance. If it does, return true, else return false.

GetTuple(value, keyIndex): This method checks if the specified value exists in the keyIndex index of any nested array in the heap of the current instance. If it does, then return that nested array.

~~ModifyRecord(keyIndex, value, newValues): This method checks if the specified value exists in the keyIndex index of any nested array in the heap of the current instance. If it does, then it overwrites that nest array with newValues in the heap array.~~

DeleteTuple(value, keyIndex): This method checks if the specified value exists in the keyIndex index of any nested array in the heap of the current instance. If it does, then remove that nested array from the heap array.

GetHeap(): This function is used to return a sequence of all the tuples in the heap.

Local Functions/Constants

N/A

Parser

Module

Uses

Parse Modify Record, Parse Create Table, Parse Insert Record, Parse Select, Parse Delete Table, Parse Delete Record, Parse Modify Table, Parser Structs.

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseCreateTable	String	struct, error	
ParseInsertTuple	String	struct, error	
ParseSelect	String	struct, error	
ParseDeleteTable	String	struct, error	
ParseDeleteTuple	String	struct, error	
ParseModifyTuple	String	struct, error	
ParseModifyTable	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseCreateTable(input): ~~Reads the String input from standard input then parses and splits it to return string representing table name, string representing primary key index, and 2D String array representing columns and sends this info to InitParseCreateTable in the Parse Create Table module. The output along with any errors is returned.~~ Interface to pass input to Parse Create Table module function and return its output

ParseInsertTuple(input): ~~Reads the String input from standard input then parses and splits it to return String representing table name and 2D String array representing columns and sends this info to InitParseInsertRecord in the Parse Insert Record module. The output along with any errors is returned.~~ Interface to pass input to Parse Insert Record module function and return its output

ParseSelect(input): ~~Reads the String input from standard input then parses and splits it to return String representing table name and sends this info to InitParseSelect in the Parse Select module. The output along with any errors is returned.~~ Interface to pass input to Parse Select module function and return its output

ParseDeleteTable(input): ~~Reads the String input from standard input then parses and splits it to return String representing table name and sends this info to InitParseDeleteTable in the Parse Delete Table module. The output along with any errors is returned.~~ Interface to pass input to Parse Delete Table module function and return its output

ParseDeleteTuple(input): ~~Reads the String input from standard input then parses and splits it to return String representing table name, string representing primary key index, and a string representing the target value and sends this info to is sent to InitParseDeleteRecord in the Parse Delete Record module. The output along with any errors is returned.~~ Interface to pass input to Parse Delete Record module function and return its output

ParseModifyTuple(input): ~~Reads the String input from standard input the parses and splits it to return strings for the table name, primaryKeyIndex, and columns and sends this info to is sent to initParseModifyRecord in the Parse Modify Record module. The output along with any errors is returned.~~ Interface to pass input to Parse Modify Record module function and return its output

ParseModifyTable(input): Interface to pass input to Parse Modify Table module function and return its output

Local Functions/Constants

N/A

Parse Modify Record

Module

Parse Modify Record

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseModifyTuple	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseModifyTuple(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.ModifyTupleStatement struct.

Local Functions/Constants

N/A

Parse Insert Record

Module

Parse Insert Record

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseInsertTuple	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseInsertTuple(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.InsertTupleStatement struct.

Local Functions/Constants

None

Parse Create Table

Module

Parse Create Table

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out
ParseCreateTable	String	struct, error

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseCreateTable(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.CreateTableStatement struct.

Local Functions/Constants

N/A

Parse Delete Table

Module

Parse Delete Table

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseDelete	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseDeleteTable(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.DeleteTableStatement struct.

Local Functions/Constants

N/A

Parse Select

Module

Parse Select

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseSelect	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseSelect(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.SelectStatement struct.

Local Functions/Constants

N/A

Parse Delete Record

Module

Parse Delete Record

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseDeleteTuple	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseDeleteTuple(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.DeleteTupleStatement struct.

Local Functions/Constants

N/A

Parse Modify Table

Module

Parse Modify Table

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ParseModifyTable	String	struct, error	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ParseModifyTable(input): Takes the input string and parses it to get the appropriate information to store in the ParserStructs.ModifyTableStatement struct.

Local Functions/Constants

N/A

Parser Structs

Module

Parser Structs

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

CreateTableStatement: struct
InsertTupleStatement: struct
SelectStatement: struct
DeleteTableStatement: struct
DeleteTupleStatement: struct
ModifyTableStatement: struct
ModiftTupleStatement: struct

Exported Access Programs

N/A

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

N/A

Local Functions/Constants

N/A