CSCE 315 – 503

Project I: Design Document

**Team**

Wesley Tang

Sidian Wu

Grace Coffman

**Objective**

Build a database manager that stores data that can be then be accessed by an application.

Phase I - Build a database management language parser. Design a database manager to manage databases operations after parsing occurs.

Phase II - Build a user interface to take in user inputs and display results returned from the database manager once the parser passes on the command and the operation is executed.

**High Level Design**

Relation

Database Manager

Parser

Application

**Phase I**

*Database Manager*

1. Usage

The database manager is will take and act on the commands after the parse has parsed the user command. The manager will keep track of all databases once they are created as well as all sub-relations created by the user during SELECT calls. The manager is responsible for interacting with the database that will store the actual data. The parser will call the database manager to execute specific functions once a command has been seen to be valid and the command has been parsed.

1. Model and Interaction

DB Manager

Set relationNames

Realtion realtions = new Relation[]

ReadDB()

CloseDB()

CheckCommand()

ParseCommand()

ExecuteCommand()

Relation

int numberOfAttributes

String nameOfRelation

Attribute attributeList = new Attribute [numberOfAttributes]

Set keys

Relation()

Operations

* Open()
* Close()
* Write ()
* Exit()
* Show()
* Update(Entity e)
* Insert(Entity e)
* Delete(Entity e)

Attribute

Attribute(String name, bool key, int dataType)

Entity

String data = new String [numberOfAttributes]

Entity()

setData()

**Phase II**

*Application*

1. Usage

To test and show the validity of the database management language created in Phase I. This specific application will reflect a database representing information on zoos and their animals.

1. ER Model

Name

Name

Zoo Name

Type

Animals

Owner

Zoos

Location

1. Interaction

Many

Animals

Zoo 1

Many

Animals

Zoo 2

**Benefits**

1. Easy distinction between classes
2. Clear definition of data structures
3. Understanding of how to enforce uniqueness by using sets
4. Using integer values to save the type of data being stored
5. Concrete application idea to test the DML

**Risks**

1. Error checking does not catch a mistake in the DML syntax
2. Confusion over what data type is represented by a specific integer value
3. Not verifying uniqueness
4. Enforcing data integrity

**Assumptions**

1. There is a document for the developers to understand what integer values relate to what data type and such a document is used
2. Only valid DML commands are allowed to be executed
3. Valid DML commands are parsed and then handled with respect to the type of command they are