SQL Compiler:

# Overview:

SQL compiler seeks to bridge the gap between database managers and traditional programmers. It does not just help to validate the syntax for queries, it also provides the power of decision making and looping when writing SQL commands. It also allows to create functions that group certain SQL statements together. The main philosophy with which the compiler is made is brevity, elementary decision making, and reusability.

The compiler is sufficient enough to allow for simple programs to run. The project appropriately uses all the phases in compiler design.

# Program Structure:

The program will follow a basic Java syntax combined with the ability to perform SQL data manipulation. The DML will be enclosed between java-like code including if-else statements and/or for loops.

A simple function in SQL Compiler has a name, a return type, arguments, and a body.

The name of a function needs to be unique so that each function is identified by the compiler. Every functions must have a specific return type i.e. the data type of the value that the function produces. Data types are described later on.

The function will receive inputs through arguments listed. Each argument will have a specific data type mentioned. These arguments will be separated by commas.

The body will consist of the code and may include conditionals, loops, functions, and SQL commands. The body will stop the execution of the function at each curly bracket.

# Data Types:

Basic SQL compiler again follow the Java structure data types are int, double, string, boolean, and void. Values of types int, double and boolean are represented by literals. ‘void’ has no values and no literals.

# Primitive functions:

The language will consist of primitive functions, though the user has complete power over creating his custom functions as well. Each function corresponds to a statement in the DML. Some primitive functions are as follows:

InsertVal(String table, String[] columns, String[] value) { //columns and value arrays

int count = 0

while(count < columns.length()) { //loop

// Data Manipulation using SQL

INSERT INTO table (column[i])   
VALUES (value[i]);

count++;

}

}

Select(String table, String[] columns, boolean all, String condition) {

if (all == True) {

SELECT \* FROM table;

}

else {

if (condition == null) {

int count = 0;

while(count < columns.length() {

SELECT columns[i] FROM table;

count++;

}

}

else {

int count = 0;

while(count < columns.length() {

SELECT columns[i] FROM table

WHERE condition;

count++;

}

}

UpdateVal(String table, String[] columns, String[] value, String condition) {

int count = 0;

while(count < columns.length() {

UPDATE table  
SET column[i] = value[i]  
WHERE condition;

count++;

}

}