

# 1. INTRODUCTION

The main tool in porting applications is the compiler, as this is the means of converting your raw source code into an executable. When moving between compilers on different machines it is necessary to look at the behavior of one compiler compared to another, and to assess what changes may need to be made to Make files and compile lines in order to get optimal performance combined with the required accuracy.

Compilers bridge source programs in high-level languages with the underlying hardware.

A compiler requires:

- Determining the correctness of the syntax of programs
- Generating correct and efficient object code
- Run-time organization
- Formatting output according to assembler and/or linker conventions.

A compiler consists of three main parts: the frontend, the middle-end, and the backend.

## 1.1 PROBLEM DEFINITION

The main objective of this application is to develop a SQL Query processor using C. This compiler like any other compiler must be able to do following functions:

- The lexical phase tokenizes the given statement.
- Syntax analysis which checks whether the statement is syntactically correct.
- Validating the statements.
- Executing the statements.

We are aiming at a compiler such that any updating or changes required in the future can be made with minimal or no changes.

## 1.2 DOMAIN STUDY

The spread of dynamic websites on the World Wide Web today is largely due to the possibility for their content to be handled through database. Database management is a complicated process, which has been considerably rationalized by the sql programming language. As its full name (structured query language) implies, SQL is responsible for querying and getting information stored in a certain database management system. SQL offers great flexibility to users by supporting distributed database, i.e. database that can be run on several computer networks at a time.

The aim of this project is to develop a SQL query processor. As a database, it is software whose primary is to store and retrieve data as requested by the SQL statements. The SQL query processor is aimed at achieve following features.

- Maintaining the table schemas: The database schema is defined during the database only and it is not expected to change frequently. Maintaining the database schema includes create, drop and update the database definition. The processor need to maintain the schema using data definition languages (DDL).
- Maintaining table data: once the database is defined (schema), the next level is to maintain the content of the database. Maintaining database includes of inserting, retrieve, delete and modify the data in the database tables. The processor need to maintain the schema using data manipulation language (DML).
- Validates and process the queries: it is very important to validate the SQL query or statement to achieve the efficient executions. It includes of checking the syntactical error and logical errors after parsing the SQL statements.
- Optimize the queries: optimization is the process of achieving the efficient way to execute a SQL statement or query. The query processor has to find the efficient way before the execution. This is also used to optimize the execution of the statements that are executed later also.

### 1.2.1 PHASES OF A COMPILER

A compiler is a program that reads a program in one language, the source language and translates into an equivalent program in another language, the target language. The translation process should also report the presence of errors in the source program.

The compilation process consists of seven phases

i. Lexical Analysis

This phase emphasizes for recognition of tokens and creation of uniform symbol table. Token is a group of characters like keywords, identifiers and symbols etc. Lexical analyses scan the source program and construct basic elements of the language. Keyword table, literal table, Identifier table and uniform symbol table are getting generated at this phase. [1, pp. 231-242]

ii. Syntax Analysis

The syntax analysis basically concerned with generation of intermediate code and produce valid grammar. It receives valid tokens from scanner and check it syntax and produces valid syntactical constructor. This process is known as parsing.

iii. Interpretation

The interpretation phase is a collection of routines that are called only when construct is recognized in syntactic phase. This collection of routines produces an intermediate form of source program. A machine independent code optimization is applied over source code. This phase eliminate the redundant expressions and sub expressions, minimize of expression through boolean algebra, reduce the inclusion of redundant code segment and use a specified function for the said optimization technique.[1]

iv. Storage Assignment

The main task of storage assignment is to store variables and assign location to them. The purpose of storage assignment is to assign storage to

literals, to assign temporary locations for intermediate results and to assign storage to all referenced variable. [1, pp. 289-294]

v. Code Generation

The code generation phase produces optimized appropriate code for execution. The purpose of code generation phase is to refer identifier and literal table in order to generate appropriate code for execution and it uses macro definitions to define macro operators in matrix.[1, pp.260-271]

vi. Assembly and Output

The assembly and output phases resolve table references. It also formats the information for loader. The code generation phase generates symbolic machine instruction and labels. The purpose of assemble and output phase are to calculate the addresses and resolves the labels reform, generate the codes matches with grammar, and generate binary code and storage.

## 2. SYSTEM REQUIREMENTS

### 2.1 EXISTING SYSTEM

1. **MySQL** (query processing): MySQL is currently the most popular open source database server in existence. On top of that, it is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications. Execution of MySQL statement performing the following steps:
  - Parse statement: the process of dividing the SQL statements into different individual units (words) for validation of the statements. It checks the syntax and semantics of the statements in this stage.
  - Validate: here, it assure the tables and columns specified in the statements are existing in the specified database.
  - Optimize: it is the process of achieving the efficient way of executing an SQL statement.
  - Generate access plan:
  - Execute: execute the SQL statement.
2. **Oracle** (optimizing queries)
  - Statistics collection: Oracle maintains statistics of the processes that run within the database. From this it can get information about the time required to execute a query.
  - Cost based optimizer: SQL select statements can be achieved in many ways, especially in the case of complex statements involving joins. Oracle breaks each step into individual units and analyses the various steps based on the resources it uses.
  - Soft parsing (bind variable): Queries that are executed are manipulated to include the bind variable and saved across. Statements that are input are checked with these and if a match is found the statement is executed without parsing.

A compiler translates (or compiles) a program written in a high-level programming language that is suitable for human programmers into the low-level machine language that is required by computers[1]. During this process, the compiler will also attempt to spot and report obvious programmer mistakes. The compilation process is divided into series of sub processes called phases. We are doing the following phases for SQL.

- Lexical analysis
- Syntax analysis

The lexical analyzer or scanner separates the characters of the source language into groups that logically belong together, these groups are called tokens. The usual tokens are keywords such as CREATE, TABLE and ALTER, identifiers, and operator and punctuation symbols. The output of the lexical analyzer is a stream of tokens. The tokens in this stream can be represented by using integers. The syntax analyzer groups tokens together into syntactic structures. The syntactic structure can be regarded as tree whose leaves are the tokens.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not adhering to the relational model as described by Codd, it became the most widely used database language. Although SQL is often described as, and to a great extent is, a declarative language, it also includes procedural elements. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standards (ISO) in 1987. Since then, the standard has been enhanced several times with added features. However, issues of SQL code portability between major RDBMS products still exist due to lack of full compliance with, or different interpretations of, the standard.

## 2.2 PROPOSED SYSTEM

We have proposed a Compiler for SQL Query using C. It has the following functionalities:

- Divides the given program statements into tokens
- Identifies each token. Tokens can be an identifier, Literal or keyword.

- These Tokens are given as input for syntax analysis. The entered statement is checked if it is syntactically correct. The semantics can also be checked.
- If there is an error in the syntax phase it will displayed on the output screen along with the line numbers.
- The statement is also is check for validity. This involves checks for table or database existence. And to check for the data types etc.
- The statement is executed if it is syntactically correct. The changes are reflected on the txt files. The database is maintained in these files.

## 2.3 CONSTRAINTS AND ASSUMPTIONS

To develop the proposed system the following assumptions and constraints are made:

- Only DDL and DML statements are available in YeSQL, TCL and DCL statements are not supported in the proposed system.
- All the keywords available are not implemented. Refer appendix A to see the keywords supported by YeSQL.
- Data types allowed are only integer (int) and text (txt).
- DDL statements are included are create database, create table, drop database, drop table, alter table.
- DML statements cannot have complex conditions.
- The select statement does not support joins.
- YeSQL is a standalone application; it will not work over a network.

## 2.4 TOOL SURVEY

Programming in C is a tremendous asset in those areas where you may want to use Assembly Language, but would rather keep it a simple to write and easy to maintain program. The time saved in coding can be tremendous, making it the most desirable language for many programming chores. In addition, since most programs spend 90 percent of their operating time in only 10 percent or less of the code, it is possible to write a program in C, then rewrite a small portion of the code in Assembly Language and

approach the execution speed of the same program if it were written entirely in Assembly Language[3]. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. Therefore c was useful for many applications that had formerly been coded in assembly language, such as in system programming.

C language is the most suitable tool to develop the SQL query processor because of the following features.

1. Powerful programming language: C is very efficient and powerful programming language; it is best used for data structures and designing system software.
2. Better file handling: since the file manipulation is the one of the main functionality of SQL Query processor, it's very important to facilitate maintenance the file easily and efficiently.
3. Array manipulations: during the parsing stage of the SQL statements the Query processor required array to store the parsed individual units (words).c provides very efficient and easy manipulations of the arrays.
4. The c language requires only the basic requirements of the system to operate.
5. Portability: we can compile or execute C program in any operating system (UNIX, dos, windows).
6. Modularity: modularity is one of the important characteristics of C. we can split the C program into no. of modules instead of repeating the same logic statements (sequentially). It allows reusability of modules.



## 2.5 HARDWARE REQUIREMENTS

Hardware	Requirements
Processor	Pentium 60 MHz or higher processor
RAM	32MB RAM or higher
Available Hard Disk Space	10 MB hard-disk space free (not including swap space required for the Operating system.
Operating System	Windows 95 or higher

Table 2.1 Hardware Requirements

## 2.6 SOFTWARE REQUIREMENTS

Software requirement for developing this project is as follows:

- Dev C++
- Windows XP or higher version

## 3. DESIGN SPECIFICATION

### 3.1 MODULAR DESIGN

There are mainly five modules in this project

#### 3.1.1 LEXICAL ANALYSIS

In lexical analysis the SQL Queries are divided into tokens and stored in the uniform symbol table. So the input for this phase is the character array containing SQL Queries and the output is the uniform symbol table. There are various operations performed in this module:

Identify the keywords:

All the keywords are stored in an array. The tokens are compared with the contents of the keyword array. If they are present then they are labeled as “keyword” and store in the uniform symbol table or else we proceed to find the type of token.

Identify the Literals:

We are using the technique called “Finite automata” to perform the above function. Each character checked one by one and even if one goes wrong then the token is not a literal. We have considered even decimal values. All the conditions have been checked.

The rules for naming the variable are:

- Variables name must be begin with a letter followed by letter or digits or combination of both.
- No special characters allowed .
- Maximum length of variable length must not exceed 255 characters.
- SQL Query variables are not case sensitive.
- It must be unique with in the same scope.

### **3.1.2 SYNTAX ANALYSIS**

In syntax analysis uniform symbol table is taken as an input and then checking the syntax with predefined SQL Query syntax. Syntax analysis implemented using recursive parsing method. The main operation done here is syntax checking and displaying the error. The errors generated in these phase is stored in a structure along with the line number.

### **3.1.3 VALIDATION**

During validation it checks that the database name, table names, column names and their data types are exist as specified in the query statement. If there is any mismatch then the first encountered error will be displayed. If the statements are correct then the execution takes places.

### **3.1.4 OPTIMIZATION**

Optimization is the process of increasing the through put of execution. Yesql will be containing a text file called “optimization” to do this particular processs.

### **3.1.5 EXECUTION**

After the syntax analysis and validation process if the statements are syntactically and semantically correct, the output will be reflected in the corresponding external files. There will be 3 files related to a database.

## 3.2 DETAILS OF EACH MODULE

### 3.2.1 FLOW CHART

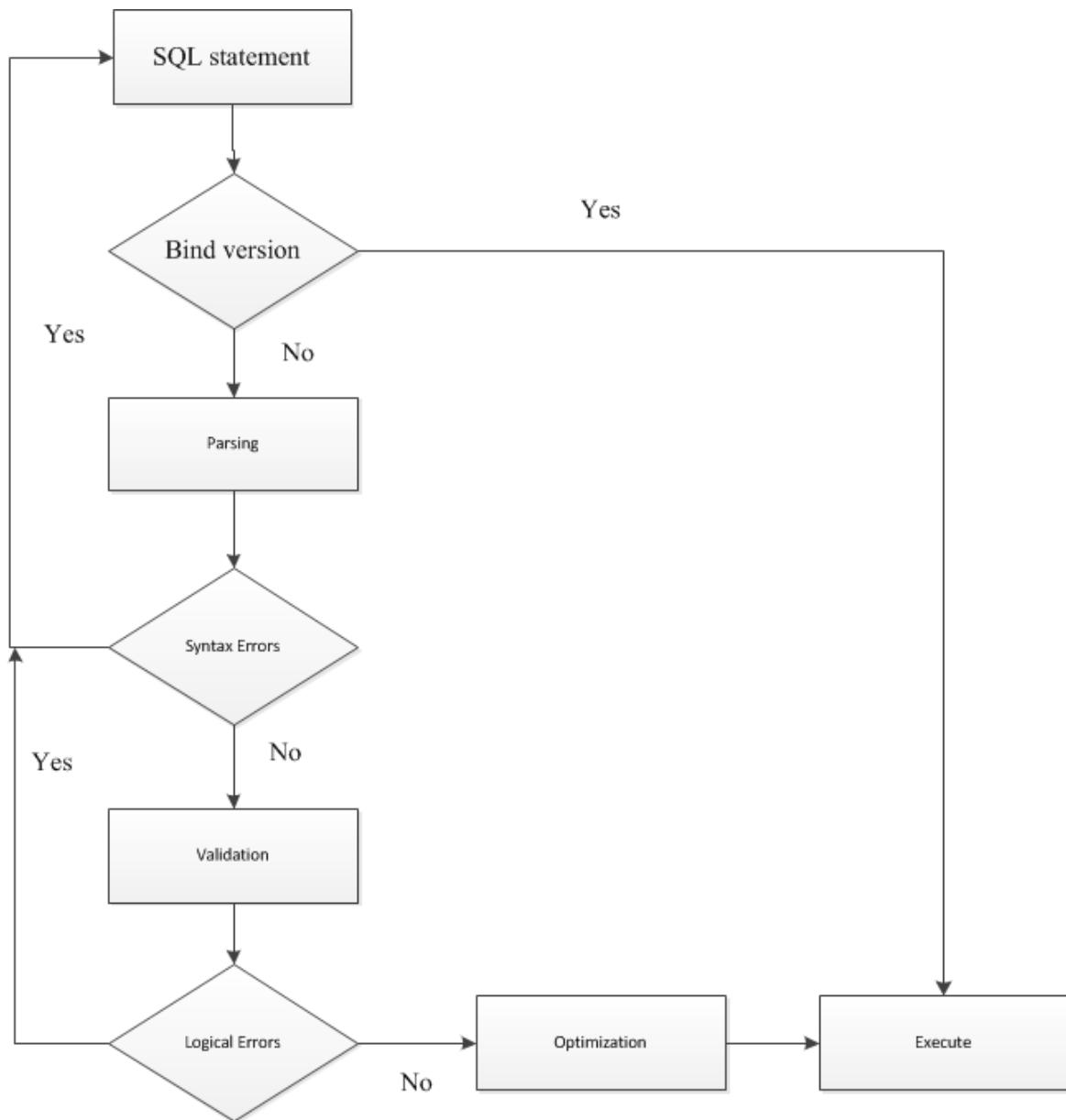


Fig. 3.1 Flow Chart

### 3.2.2 DATA FLOW DIAGRAM (DFD)

#### Level 0:

This initial level, describes the flow of processing at .Sql statements are optimized in the first step of query processing.

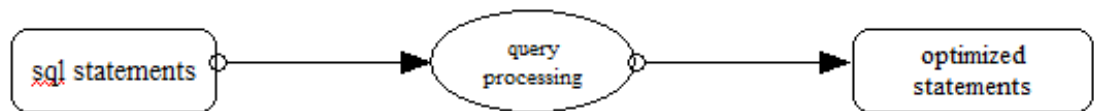


Fig 3.2 DFD level 0

**Level 1:**

Statements are parsed into different categories and stored into relative tables like keyword table , identifier table . These table are used for validating the statements, logically as well as syntactically. Then the statement is set for execution.

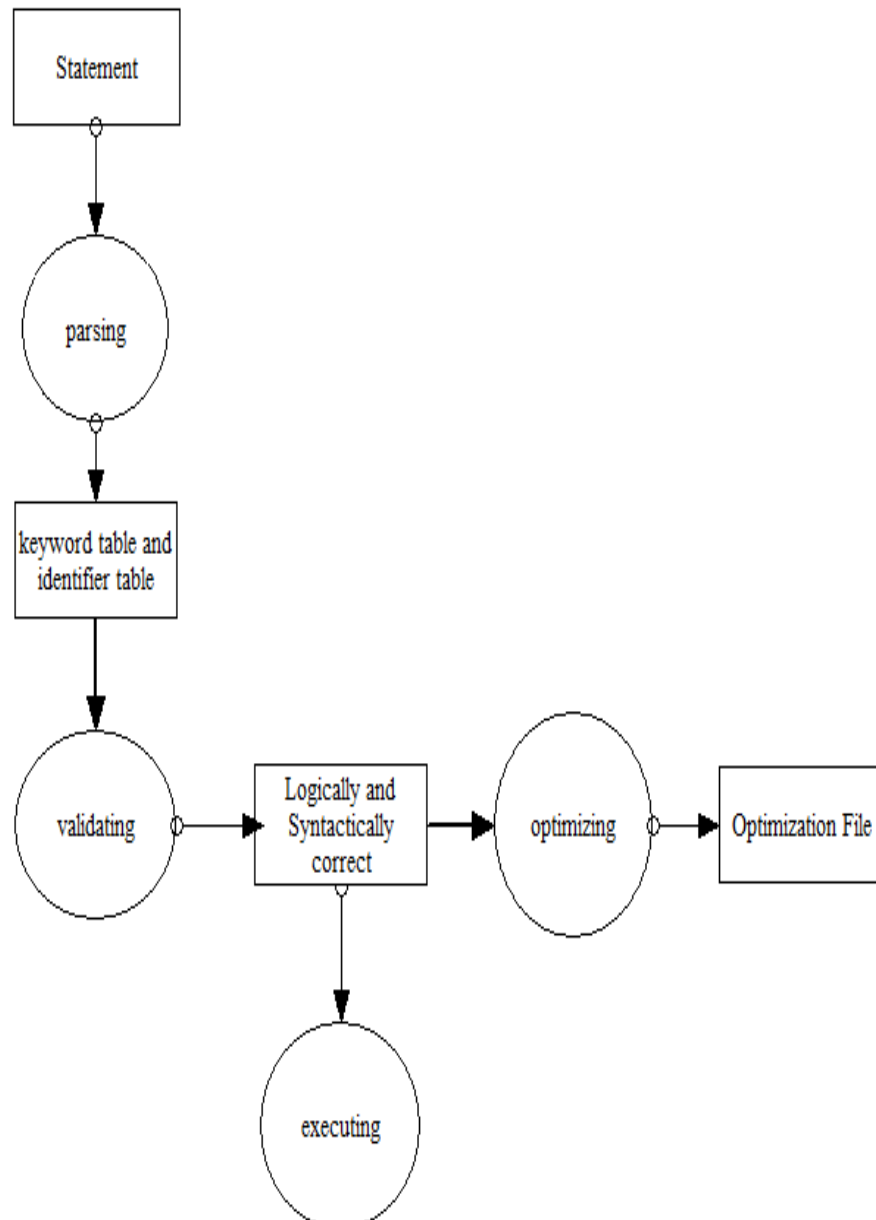


Fig 3.3 DFD level1

### 3.2.3 BLOCK DIAGRAM

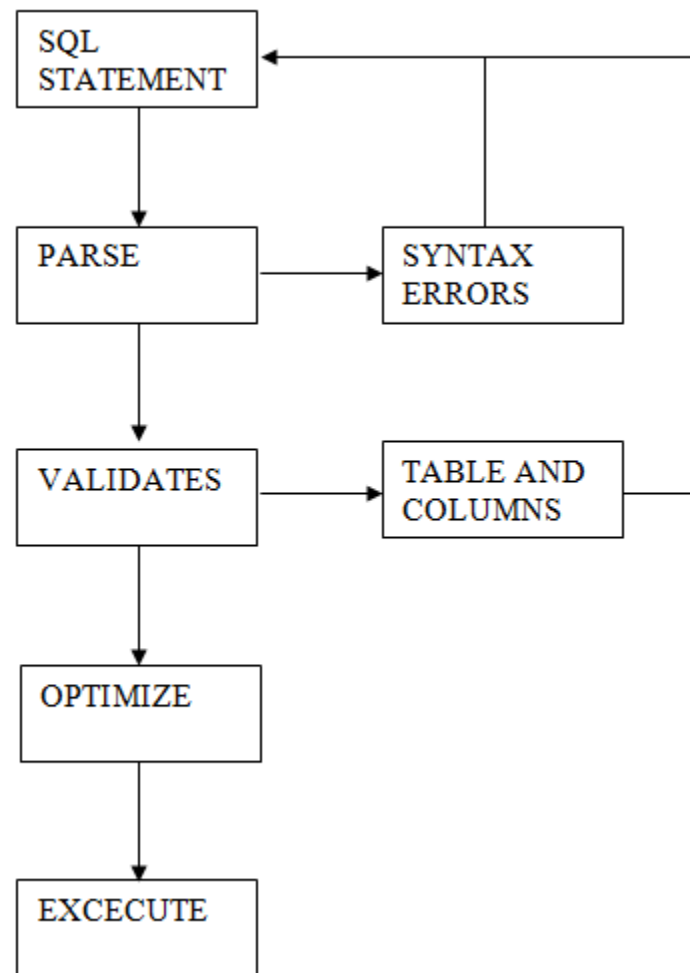


Fig. 3.4 Compiler Block Diagram

### 3.2.4 DATA STRUCTURE DESIGN

#### 3.2.4.1 Arrays

There are 2 arrays are required for sql query processor. It is used during the parsing of the sql statements. These arrays are named as identifier table and keyword table.

- **Keyword Table:** It is a static table that contains all the keywords that will be used in the SQL statement. An essential function of a compiler is to record the identifiers used in the source program and collect information about various attributes of each identifier. These attributes provide information about the storage allocated for an identifier, its type, its scope. This table exists throughout the compilation and run-time. It is one of the most important data structures in every compiler. The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly. When any keyword in the source program is detected in the lexical phase, the identifier is checked from the keyword table. It is handled as a 2 dimensional character array. It gets initialized when the application is started.
- **Identifier Table:** It is a dynamic table that contains the table names of all the tables in the database. It is one of the most important data structures in every compiler. This table allows us to find the record for each table quickly and to check whether the table exists or not. It helps in avoiding duplication of the table. It is handled as a 2 dimensional character array. It gets initialized when the database is started.

#### 3.2.4.2 Files:

Mainly there are three files required for SQL query processor. While creating a database, a folder will be created automatically along with required files. The folder will be having the database name; the files related to the database are stored in the created folder. The file structure for SQL query processor is given below.

- **Relationship file:** This file is used to store the relationship between different tables in a database. When the tables are created it will be added as an entry of this relation file.



- **Table Data file:** This file is used to store the content of each table in the database. Definition of the each file will be stored in the schema file and accordingly the data is stored in data file
- **Table Schema file:** This file is used to store the schema of the database when it's created.

**Basic File structure:**

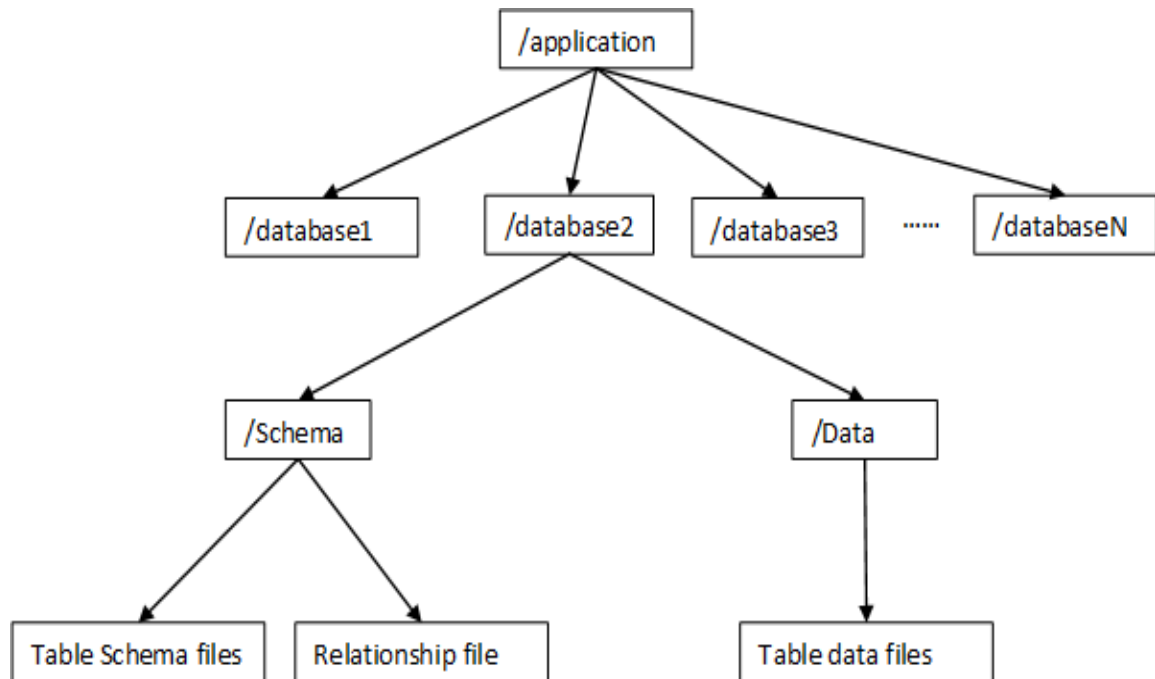


Fig 3.5 File Structure

In the above diagram it described the different levels of file hierarchy. It is shown that there are N numbers of databases inside an application. Under each database (represented as directory), it has two subdirectory named as schema and data. Schema file contains two files called as table schema file and relationship file. Both the files are used to maintain the schema of particular database. Inside the data sub-directory, there will be a file called table data file which is used to store the content of each table in the database.

- **Table Schema file:** This file is used to store the schema of the database when it's created. i.e., it will be containing the general information about a database which

includes name, access modes; size etc. content of schema file can be altered. And it is accessed and modified using the data definition language (DDL).

**Structure of table schema file:**

Column_name1	Data type	Length	Key
Column_name2	Data type	Length	Key
Column_name3	Data type	Length	Key
Column_name4	Data type	Length	Key

Table 3.1 Structure of table schema file

As the tables explain, schema file has four columns. The first column entry represents the column name. it is user defined (like name, place and an object etc). The second column will be containing the data type of particular entry. It can be either int or txt. Third column specifies the length of the data type. It is compulsory to specify the length during the creation. And the fourth column specifies the key entry for the particular data. It is not necessary to specify the key during the creation of tables; we can modify it later also.

- **Relationship file:** This file is used to store the relationship between different tables in a database. When the tables are created it will be added as an entry of this relation file. During the foreign key specification the table name along with the foreign key will be updated to relationship file as per the request. The contents of the relationship files are accessed and modified using Data Definition Language (DDL).

**Structure of relationship file:**

Table name1	Table name2	Key attribute
Table name1	Table name2	Key attribute
Table name1	Table name2	Key attribute
Table name1	Table name2	Key attribute

Table 3.2 Structure of relationship file

The relationship file will be containing three fields. First two fields are stands for the table names which are related. And the third field represents the foreign key. Each relation of a table will be specified separately. As explained above in the diagram “tablename1” and “tablename2” are the two related table with the primary key “key attribute”. The content of this file altered during the relation specification.

- **Table Data file:** This file is used to store the content of each table in the database. Definition of the each file will be stored in the schema file and accordingly the data is stored in data file. Content of this file is accessed and maintained using data manipulation language (DML).

**Structure of Table Data file:**

Datacolumn1	Datacolumn2	Datacolumn3	Datacolumn4	Datacolumn5
Datacolumn1	Datacolumn2	Datacolumn3	Datacolumn4	Datacolumn5
Datacolumn1	Datacolumn2	Datacolumn3	Datacolumn4	Datacolumn5
Datacolumn1	Datacolumn2	Datacolumn3	Datacolumn4	Datacolumn5

Table 3.3 Structure of Table Data file

---

## TOKEN CATEGORIES OF SQL QUERY

### VARIABLES

Variables are placeholders that are used to store values during the execution of the program. The data stored in the variable can be accessed by referring to variable name. The rules for naming the variable are :

- Variables name must be contains either characters or digits.
- No special characters are allowed.
- Maximum length of variable length must not exceed 255 characters.
- SQL query variables are not case sensitive.
- It must be unique with in the same scope.

### KEYWORDS

Keywords are also known as reserved words. Each keyword has unique meaning. Keyword cannot be used a variables[2]. Some of the available keywords in SQL query are:

- CREATE: - Either to create database or table.
- ALTER: - To alter the table.
- TABLE:- To specify table.
- DROP
- INTO
- TXT
- INT

### DATA TYPES AVAILABLE IN YESQL QUERY

- Int: - Numeric Data Type.
- Txt :-String Data type

A string variable can store text or a string of characters. String variable are of variable length by default. Fixed string length can be declared as follows:

### **OUTPUT OF SYNTAX ANALYSIS**

Here, The error which is encountered first will be displayed as error indication. If the statements are syntactically correct then the statements are passed for the validation process.

### **OUTPUT OF VALIDATION**

During validation it checks that the database name, table names, column names and their data types are exist as specified in the query statement. If there is any mismatch then the first encountered error will be displayed. If the statements are correct then the execution takes places.

### **OUTPUT OF EXECUTION**

If the statements are syntactically and semantically correct, the output will be reflected in the corresponding external files.

### 3.3 GRAMMAR DESIGN

The grammar specification of the statements that are dealt with in YeSQL is as shown below.

#### Create database statement

Only one form of this statement is available in YeSQL. The syntax of the statement is as follows:

```
CREATE DATABASE <database_name>;
```

The first and the second words are keywords, that is “CREATE” and “DATABASE” what follows is the two keywords is a user defined word; which is the name of the database. The name of the database cannot be duplicated.

#### Drop database statement

Only one form of this statement is available in YeSQL. The syntax of the statement is as follows:

```
DROP DATABASE <database_name>;
```

The first and the second words are keywords, that is “DROP” and “DATABASE” what follows is the two keywords is a user defined word; which is the name of the database. The name of the database has to exist within the system, else a validation error is thrown.

#### Create table statement

Two forms of this statement are available in YeSQL. The first syntax is as follows:

```
CREATE TABLE <table_name>;
```

The first and the second words are keywords, that is “CREATE” and “TABLE” what follows is the two keywords is a user defined word; which is the name of the table. The name of the table cannot be duplicated within the same database. With this form, the table attributes can be later added with alter table command.

The second syntax that is available has the following syntax

```
CREATE TABLE <table_name> ( <attribute_description>, <attribute_description>, ..... );
```

Where <attribute \_description> = (column\_name1 datatype (length) key)

In this syntax the table attributes are given with the create table command. The first 3 words has the same meaning as in the previous syntax. The attribute description can have

details about the attribute such as name of the attribute, datatype and length and they key. The key is not mandatory for all fields. The attribute description can have only these fields and the has to maintain the same order.

### **Drop table statement**

Drop table statement is used to delete the table in the database. It has the following syntax

```
DROP TABLE <tablename>;
```

The first and the second words are keywords, that is “DROP” and “TABLE” what follows is the two keywords is a user defined word; which is the name of the table. The name of the database has to exist within the database that we are working on, else a validation error is thrown.

### **Alter table statement**

Alter table statement is used to change the schema of the table. The table schema when changed the data in the corresponding table is cleared. The syntaxes that is allowed in the system is as follows:

```
Alter table <table_name> ADD <attribute_description>;
```

This statement is used to add a attribute to the schema of the table. The attribute name cannot be duplicated and the relationships cannot collide with the previous relationship. In the syntax, the first and the second words are keywords; and the third word is a user defined word, which is the table name. The next word is again a keyword which is ADD, which tells the system to add a attribute to the schema of the table.

```
Alter table <table_name> DROP COLUMN <attribute_name>;
```

This statement is used to drop an attribute to the schema of the table. The relationships cannot collide with the previous relationship. In the syntax, the first and the second words are keywords; and the third word is a user defined word, which is the table name. The next two words is again a keyword which are DROP and COLUMN, which tells the system to delete an attribute to the schema of the table, the name of that attribute is specified by the attribute name.

**Insert table statement**

Insert table is used to add data to the tables in the database. YeSQL supports two insert table syntaxes, they are as follows:

INSERT INTO <table\_name> Values <attribute\_values>

In the syntax we have to add all the data values of the specified table. In this statement the first and second words are keyword which is followed by the table name on which the operation is to take place. The next word is a keyword “VALUES”. The occurrence of this word after table name says that all the data values are to be inserted into the table. The values to be inserted are specified in the attribute values.

INSERT INTO <table\_name> <column\_name> Values <attribute\_values>

In the syntax we have to add few data values of the specified table. In this statement the first and second words are keyword which is followed by the table name on which the operation is to take place. The next words are user defined column names which are separated by commas. The next word is a keyword “VALUES”. The occurrence of this word after table name says that all the data values are to be inserted into the table. These values are the values that correspond to the column values specified in the <column\_names>

**Delete table statement**

Delete table is used to delete tuples from the table. The syntaxes supported in the system are as follows:

DELETE FROM <table\_name>;

This format is used to delete all the rows that are present in a table. In the syntax, the first and the second words are keywords “DELETE” and “FROM”. The next word that follows is a user defined word which is the name of the table on which the operation is to be performed.

DELETE FROM <table\_name> WHERE <condition>;

<condition> = <column\_name> = “some value”>

This syntax is used to delete only rows from a table that satisfies a particular condition. This condition is specified with a where clause. The clause is specified by a keyword “WHERE”. This comes after the word “table name” in the syntax. The condition cannot be complex and cannot have sub queries.



### Update table statement

Update table is used to update tuples from the table. The syntaxes supported in the system are as follows:

```
UPDATE <table_name> SET <column_name=value>;
```

This format is used to update all the rows that are present in a table. In the syntax, the first word is a keywords “UPDATE”. The next word that follows is a user defined word which is the name of the table on which the operation is to be performed. The values to modify are specified by the clause “SET”. The column names and their values are specified after this.

```
UPDATE <table_name> SET <column_name=value> WHERE <condition>;
```

```
<condition> = <column_name = “some value”>
```

This syntax is used to update only rows from a table that satisfies a particular condition. This condition is specified with a where clause. The clause is specified by a keyword “WHERE”. This comes after the word “table name” in the syntax. The condition cannot be complex and cannot have sub queries.

### Select statement

Select statement is used to select data from the tables in the database. The syntaxes supported in the system are as follows:

```
SELECT * FROM <table_name>;
```

This format is used to display all the data values in the specified table in the database. In the format the first three words are key words, the second word is a special character. The fourth word is the table name, on which the operation is to be performed.

```
SELECT FROM <table_name> WHERE <condition>;
```

```
<condition> = <column_name = “some value”>
```

This syntax is used to select only rows from a table that satisfies a particular condition. This condition is specified with a where clause. The clause is specified by a keyword “WHERE”. This comes after the word “table name” in the syntax. The condition cannot be complex and cannot hav

### 3.4 INPUT SPECIFICATION

The input SQL statement from the user is taken from the user. The first keywords of the statement are checked to find out the type of statement. The statement is then passed to the corresponding function to check the syntax. For some statements there is some manipulation that is done on the statement, such as removing spaces. This is done prior to syntax checking.

The output of the syntax analysis statement is sent to the corresponding validation function of the statement. Here we check for the presence of the table, data types and other various sorts of constraints. Once the verification of the validity is conformed, the statement is also executed in and the changes are reflected on the files, which maintain the database.

#### 3.4.1 List of valid statements

- **Create query:**

```
YeSql>create database collage;  
  
YeSql>create table bca;  
  
YeSql>create table mca where (number int(20), fees txt(10));
```

Fig 3.6 Create query

- **Use query:**

```
YeSql>use collage;
```

Fig 3.7 Use query

- **Drop query:**

```
YeSql>drop database collage;  
  
YeSql>drop table abc1;
```

Fig 3.8 Drop query

- **Alter Query:**

```
YeSql>alter table mca rename to bca;  
  
YeSql>alter table bca add column roll int(40);  
  
YeSql>alter table bca drop column roll;
```

Fig 3.9 Alter Query

- **Describe query:**

```
YeSql>describe bca;
```

Fig 3.10 describe query

- **Delete query:**

```
YeSql>delete from mca;  
  
YeSql>delete from bca where rollno=15;
```

Fig 3.11 Delete query

- **Update query:**

```
YeSql>update bca set name='first';  
YeSql>update bca set name='first' where rollno=12;
```

Fig 3.12 Update query

- **Select query:**

```
YeSql>select * from bca;  
YeSql>select * from bca where age=12;  
YeSql>select <name,age> from bca;  
YeSql>select <name,age> from bca where rollno=112373;
```

Fig 3.13 Select query

### 3.5 OUTPUT SPECIFICATION

Once the statement is determined to be syntactically correct, essential parts of the statement is extracted from the statement. This is done while the syntax checking.

The output of the validation is the set of values and column names for the execution. The output of the execution confirms the success of the statement or shows the error that occurred during the process.

#### 3.5.1 List of valid outputs

- **Create query:**

```
database 'collage' has been created  
'collage' database already exists.  
'mca' table created in 'collage' database
```

Fig 3.14 Create query

- Describe query:

FIELD NAME	DATA TYPE	LENGTH	PRIMARY KEY
age	int	11	PK
name	txt	21	
rollno	txt	15	

Table 3.4 Describe query

- Select query:

age	name	rollno
12	vinnas	1125953
12	vincnas	11225953
13	pramod	1125936
14	sonal	1125945
15	fazil	1125964
16	manu	1125929
17	sharon	1125943
18	nimisha	1125932
19	fazil	1125964
20	fazil	1125964

Table 3.5 Select query

- Alter query:

```

bca renamed to mca

bca table altered succeesfully!!!

column dropped

```

3.15 Alter query

### 3.6 USER INTERFACE DESIGN

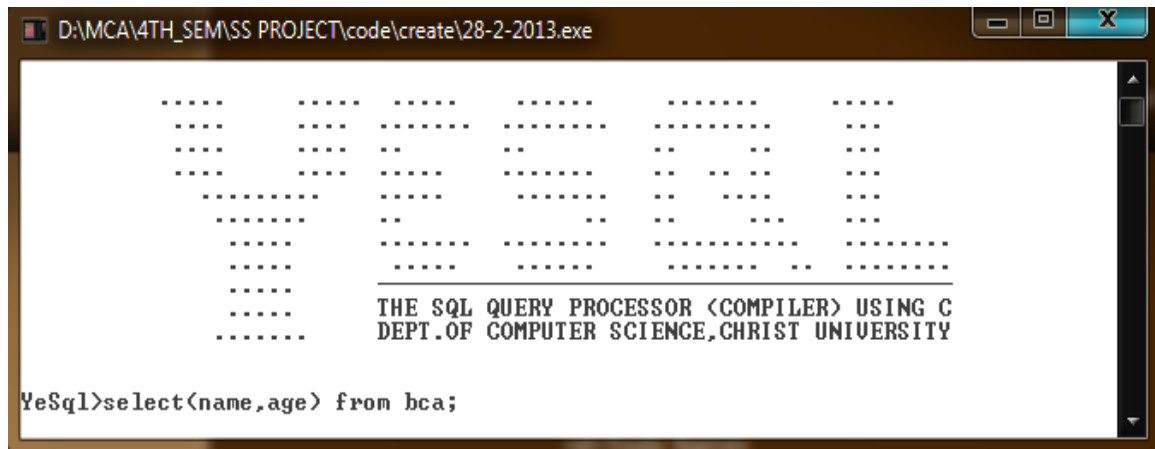


Fig 3.16 user interface

## 4. IMPLEMENTATION DETAILS

### 4.1 SCREEN SHOTS

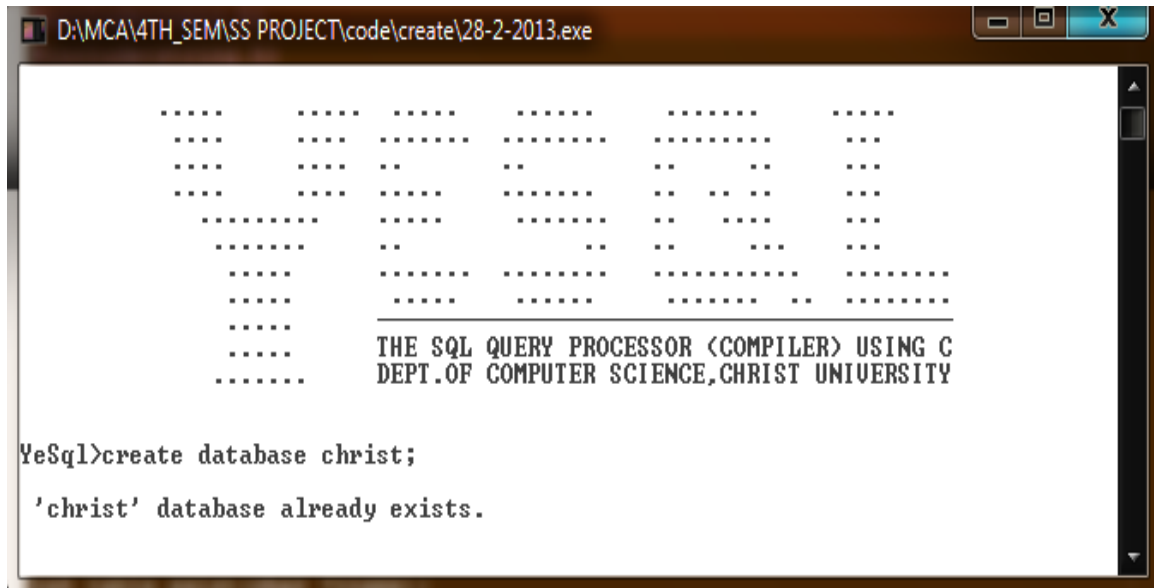


Fig 4.1 first page

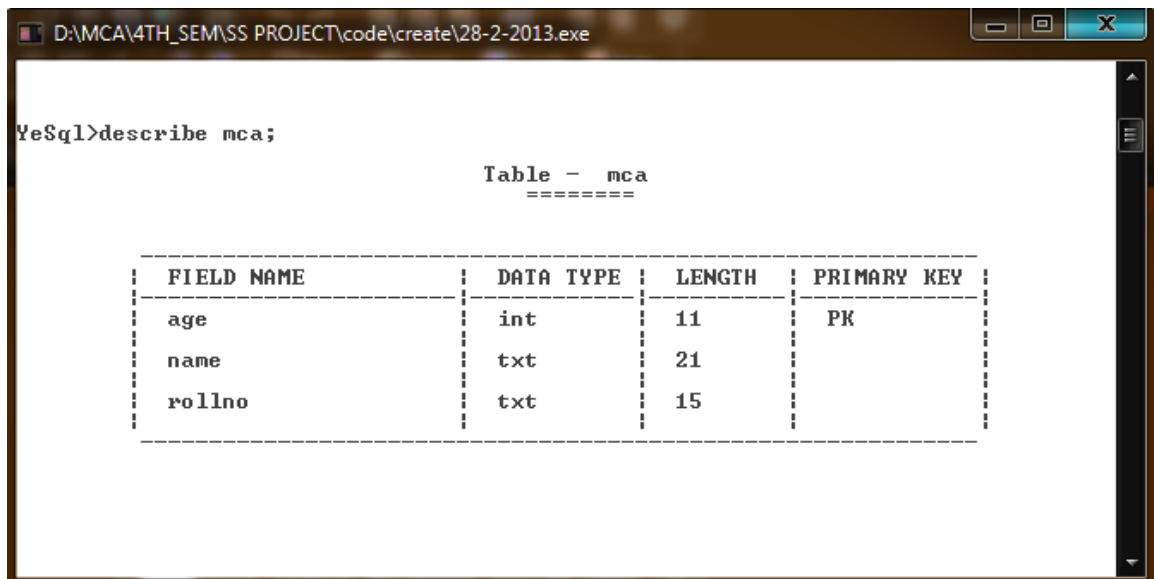
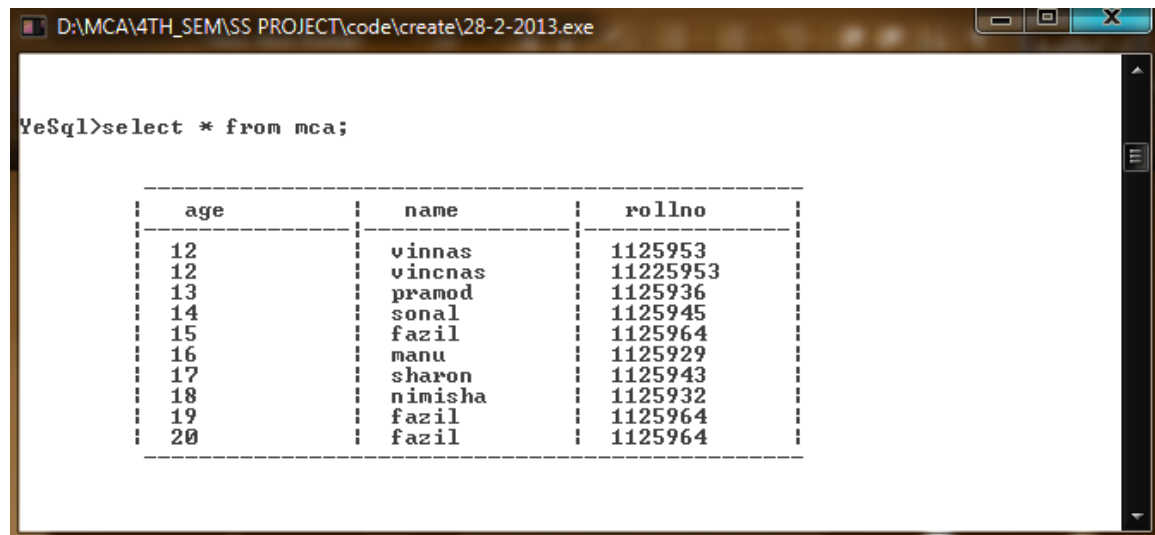


Fig 4.2 describe query



The screenshot shows a terminal window titled "D:\MCA\4TH\_SEM\SS PROJECT\code\create\28-2-2013.exe". The prompt is "YeSql>". The command entered is "select \* from mca;". The output is a table with three columns: age, name, and rollno. The data is as follows:

age	name	rollno
12	vinnas	1125953
12	vincnas	11225953
13	pramod	1125936
14	sonal	1125945
15	fazil	1125964
16	manu	1125929
17	sharon	1125943
18	nimisha	1125932
19	fazil	1125964
20	fazil	1125964

Fig 4.3 select query

## 4.2 SAMPLE CODE

```
#include<conio.h>
#include<string.h>
#include<stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <direct.h>
#include<dir.h>
char *pch;
char temp[100],temp1[100],temp2[100];
char parts[100][100];
int i,count,t;
char usedatabase[100];
char statement[100];
void create_database_syntax(char *a);
int database_exist(char *name);
int table_exist(char *tname);
void create_database_validate(char *databasename);
void create_table_syntax(char *abc);
void create_table_validate(char *tablename,char columns[100][100],int numbercolumns);
int name_duplicate_check(char a[100][100],char b[100],int c);
int name_keywords(char *b);
void use(char *b);
void drop(char *a);
void drop_database_validate(char *databasename);
void alter(char *a);
int name_val(char *a,int len);
int type_val(char *a);
```



```

void describe_validate(char *table);
void describe_syntax(char *statement);
void select_syntax(char *statement);
void delete_syntax(char *statement);
void update_syntax(char *statement);
void logo();
int bracesvalid(char *statement);
void removespaces(char *statement);
void insertsyntax(char *statement);
int main()
{

char statement[100],statement1[100];
char parts[100][100];
int count,type;
char ch;
//logo();
do
{
int i;
printf("\n\n\nYeSql>");
fflush(stdin);
gets(statement1);
//convert upper to lower
for(i=0;i<strlen(statement1);i++)
{
if(statement1[i]>=65&&statement1[i]<=90)
statement1[i]=statement1[i]+32;
}
i=0;
strcpy(statement,statement1);
pch = strtok (statement," ");
while (pch != NULL)
{
strcpy(parts[i],pch);
count=i;
i++;
pch = strtok (NULL," ||,");
}
if(strcmp(parts[0],"create")==0) type=1;
else if(strcmp(parts[0],"drop")==0) type=2;
else if(strcmp(parts[0],"alter")==0) type=3;
else if(strcmp(parts[0],"use")==0) type=4;
else if(strcmp(parts[0],"describe")==0) type=5;
else if(strcmp(parts[0],"delete")==0) type=6;
else if(strcmp(parts[0],"update")==0) type=7;
else if(strcmp(parts[0],"select")==0) type=8;
else if(strcmp(parts[0],"insert")==0) type=9;

```

```
else if(strcmp(parts[0],"quit")==0) type=10;
else type=99;
switch(type)
{
case 1: //create_database_syntax table
if(strcmp(parts[1],"database")==0)
{
create_database_syntax(statement1);
}
else if(strcmp(parts[1],"table")==0)
{
create_table_syntax(statement1);
}
else
{
printf("\n\t invalid attempt to creation of '%s' (only database/table)",parts[1]);
}
break;
case 2://drop
drop(statement1);
break;
case 3://alter
alter(statement1);
break;
case 4://use
use(statement1);
break;
case 5://describe
describe_syntax(statement1);
break;
case 6://delete
delete_syntax(statement1);
break;
case 7://delete
update_syntax(statement1);
break;
case 8://select
select_syntax(statement1);
break;
case 9://insert
if(bracesvalid(statement1)==1)
{
removespaces(statement1);
insertsyntax(statement1);
}
break;
case 10://quit
exit(0);
```

```
break;
default:
//name_duplicate_check(parts,temp,2);

printf("\ninvalid operation!!!");
break;

}
// printf("\n\ndo u want to continue(y/n) : ");
scanf("%c",&ch);
}while(ch!='n');

//DATABASE CREATE QUERY SYNTAX
void create_database_syntax(char *abc)
{
char db_name[100];
i=0;
count=0;
t=0;
strcpy(statement,abc);
pch = strtok (statement," ");
while (pch != NULL)
{
strcpy(parts[i],pch);
count=i;
i++;
pch = strtok (NULL, " ||,");
}
strcpy(temp,parts[count]);
t=strlen(temp);
if(temp[t-1]!=';')
{
if(i==3)
{
strcpy(temp2,parts[2]);
temp2[strlen(temp2)-1]='\0';
if(name_val(temp2,strlen(temp2))!=1)
{
strcpy(db_name,parts[2]);
db_name[strlen(db_name)-1]='\0';
create_database_validate(db_name);
}
}
else
{
printf("\nstatement syntax error !!!");
}
}
}
```

```

else
{
printf("\nstatement error- ';' missing");

}
}
//CREATE TABLE VALIDATE
void create_table_validate(char *tablename,char columns[100][100],int numbercolumns)
{
int i;
FILE *f1;
char dirname[100];
//strcpy(usedatabase,"db1");
strcpy(dirname,"c:/YeSQL/");
strcat(dirname,usedatabase);
strcat(dirname,"/Schemas/");
strcat(dirname,tablename);
strcat(dirname,".txt");
f1=fopen(dirname,"w");
for(i=0;i<numbercolumns;i++)
{
fprintf(f1,"%s\n",columns[i]);
}
fclose(f1);
}

//CREATE TABLE
void create_table_syntax(char *abc)
{

char tb_name[100];
char clm_name[100];
char table_content[100][100];
int num_columns=0;
char duplication[100][100];
int dc=0;
i=0;
count=0;
t=0;
int kc=0;
strcpy(statement,abc);
pch = strtok (statement," ");
while (pch != NULL)
{
strcpy(parts[i],pch);
count=i;
i++;
pch = strtok (NULL," ||,");

```

---

```

    }
    strcpy(temp,parts[count]);
    t=strlen(temp);
    if(temp[t-1]==';')
    {
        if(i<3)
        {
            printf("\nstatement incomplete");
        }
        else if(i==3)//simple creation of tables
        {
            strcpy(temp2,parts[2]);
            temp2[strlen(temp2)-1]='\0';
            if(name_val(temp2,strlen(temp2))!=1)
            {
                strcpy(temp,parts[2]);
                for(i=0;i<strlen(temp)-1;i++)
                {
                    tb_name[i]=temp[i];
                }
                tb_name[i]='\0';
                if(database_exist(usedatabase)!=1)
                {
                    printf("specify the database name using 'use' command");
                    return;
                }
                printf("\n'%s' table created in '%s' database\nadd data members using ALTER
command",tb_name,usedatabase);//table name printing
            }
        }
        else if(count>=4)//creation of tables with data values
        {

            strcpy(temp,parts[3]);
            if(temp[0]!='(')
            {
                printf("\nstatement error - ( missing");
                printf("\n\t\t'%s'",temp);
            }
            else
            {
                //datatype checking
                int j=4;
                int t1=0;
                int k;
                while(j<=count)
                {
                    //column name validation

```

---

```
if(j-1==3)//ignore first '('
{
char valid[10];
char valid1[10];
strcpy(valid,parts[j-1]);
for(i=0;i<strlen(valid);i++)
{
if(valid[i]!='(')
{
valid1[t1]=valid[i];
t1++;
}
}
valid1[t1]='\0';
strcpy(temp,valid1);
}
else
{
strcpy(temp,parts[j-1]);
}
strcpy(clm_name,temp);
if(name_duplicate_check(duplication,clm_name,dc)==1)
{
t1++;
break;
}
strcpy(duplication[dc],clm_name);
dc++;
if(name_val(temp,strlen(temp))==1)
{
t1++;
break;
}
//datatype validation
t1=0;
if(type_val(parts[j])==1)
{
t1++;
break;
}
else//print tokens
{
strcpy(temp,parts[j]);
for(i=0;i<3;i++)
{
temp1[i]=temp[i];
}
k=0;
```



```
}
else
{
if(t1==0)
{
strcpy(tb_name,parts[2]);
printf("\n\n\t\t\t%s TABLE CREATED",tb_name);
}
}
}
else
{
printf("\n\n\t\t\t error!!!! improper termination");
}
}
else
{
if(kc>1)
{
printf("\n\n\t\t\t error!!! more than one primary key present ");
}
}
else
{
if(t1==0)
{
if(database_exist(usedatabase)!=1)
{
printf("specify the database name using 'use' command");
return;
}
}
else
{
strcpy(tb_name,parts[2]);
printf("\n\n\t\t\t%s TABLE CREATED",tb_name);
}
}
}
}
}
}
else
{
printf("\n\n\t\t\t syntax error!!! (invalid columns)");
}
}
```



```
else
{
printf("\nstatement error- ';' missing");

}
create_table_validate(tb_name,table_content,num_columns+1);
for(i=0;i<=num_columns;i++)
{
printf("\n  %s ",table_content[i]);
}
}
//END OF QUERY

//DROP DATABASE VALIDATE
void drop_database_validate(char *databasename)
{
int dbe;
int check;
char dirname[100];
char dbname[100];
check=database_exist(databasename);
if(check==0)
{
printf("\n database name does not exist.");
return;
}
strcpy(dirname,"c:/YeSQL/");
strcat(dirname,databasename);
strcat(dirname,"/Schemas");
rmdir(dirname);
strcpy(dirname,"c:/YeSQL/");
strcat(dirname,databasename);
strcat(dirname,"/Data");
rmdir(dirname);
strcpy(dirname,"c:/YeSQL/");
strcat(dirname,databasename);
dbe=rmdir(dirname);
if(dbe!=0)
{
printf("\n Error: tables has to be deleted first.");
return;
}
else
{
printf("\n'%s' database has been removed",temp);
}
}
```

## 5. SYSTEM TESTING

### 5.1 TEST CASES

#### 5.1.2 LEXICAL ERRORS

Description	Expected Result	Actual Result
Database validation	Enter the Database name which does not exist for creating database and enter existing database for database manipulation. Database name should follow the naming constraints.	It show an error message as invalid database name if enter an invalid database name
Table validation	Table name must contain only alphabets, numbers. Enter the Table name which does not exist for creating database and which is existing for Table manipulation. Specify the database for table.	It show an error message as invalid table

Table 5.1 Lexical phase Test Cases

#### 5.1.2 SYNTAX ERRORS

Description	Expected Result	Actual Result
Syntax validation	The input statement should follow SQL format	If the input statement is not following the SQL format Show the error

Table 5.2 Syntax phase Test Cases

## 5.2 TEST REPORT

### 5.2.1 LEXICAL ERRORS

#### IDENTIFIER ERRORS:

```
YeSql>create database student@##;  
invalid name!!!! <special charecters not allowed>  
  
YeSql>create database student;  
database 'student' has been created
```

Fig. 5.1 Identifier Error Page for Syntax Analyzer

### 5.2.2 SYNTAX ERROR

#### KEYWORD ERRORS:

```
YeSql>create database create;  
invalid name !!!<'create' is a keyword in YeSQL>  
  
YeSql>create table sdf <table sdg<40>>;  
invalid name !!!<'table' is a keyword in YeSQL>  
  
YeSql>create table f1 <name int<40>>;  
  
f1 TABLE CREATED  
  
name-int-40
```

Fig. 5.2 Keyword Error Page for Syntax Analyzer

#### DATATYPE ERROR PAGE:

```
YeSql>create table sdf <name sad<20>,id int<30>>;  
invalid datatype sad<20> <only txt/int>  
  
YeSql>create table sdf <name txt<sc>,id int<30>>;  
errorr.....give degit as length  
  
YeSql>create table sdf <name txt<23>>;  
  
sdf TABLE CREATED  
  
| name-txt-23
```

Fig. 5.3 Data Type Error Page for Syntax Analyzer

## STATEMENT MISSING:

```
YeSql>create table f1 (name int<40>>)
statement error- ';' missing
-
YeSql>create table f1 (name);
syntax error!!! (invalid columns)
YeSql>create table f1 (name int<40>>);
f1 TABLE CREATED
name-int-40
```

Fig. 5.4 statement missing Page for Syntax Analyzer

## 6. CONCLUSIONS

YeSQL is a SQL compiler application developed to compile and process SQL statements. The database is maintained in the form of text files; both the schema and the data are available in the database files. The processing and validation of the statement is done in a single pass. The queries are also optimized to get maximum throughput from the system.

YeSQL, the query processor, compiles SQL statements and executes SQL statements. The inputs are directly taken from the user, one statement at a time. YeSQL handles both DDL and DML statements. This statement is syntactically checked and then it is validated. Once this is done, the query is executed. Here, the queries that are executed reflect on the files and directories. The files maintained are text files so they are easily accessible and can be viewed with basic operating system requirements. The optimization method is also implemented which allows queries to be directly executed without being syntactically checked and validated, this is done by keeping a copy of syntactically and logically valid statements in a file. When a query is executed, the query is first checked for in the optimization file if the same statement is found then the query is directly validated.

The system can be enhanced further to include all the statements that SQL supports. It can also be extended to handle complex queries. As a future enhancement it can also be enhanced to take and execute multiple statements from the user at a time. A better GUI can also be added to the project if it can be coupled with another platform. In the future enhanced systems can handle not only DDL and DML statements; it had also handle DCL and TCL statements. Also, the stand alone system can be enhanced to work over multiple networked systems.

---

## REFERENCES

- (1) (n.d.). Retrieved Dec 12, 2012, from <[http://msdn.microsoft.com/en-in/library/ms190623\(v=sql.105\).aspx](http://msdn.microsoft.com/en-in/library/ms190623(v=sql.105).aspx)>
- (2) (n.d.). Retrieved Dec 17, 2012, from <[http://docs.oracle.com/cd/A58617\\_01/server.804/a58234/sql\\_proc.htm](http://docs.oracle.com/cd/A58617_01/server.804/a58234/sql_proc.htm)>
- (3) (n.d.). Retrieved Dec 27, 2012, from <<http://www.cs.ucdavis.edu/~green/courses/ecs165a-w11/8-query.pdf>>
- (4) (n.d.). Retrieved NOV 28, 2012, from dev.mysql.com/doc/
- (5) R.Alapati, S. (2005). 5. In S. R.Alapati, *Expert Oracle Database 10g Administration* (p. 1276). APress.