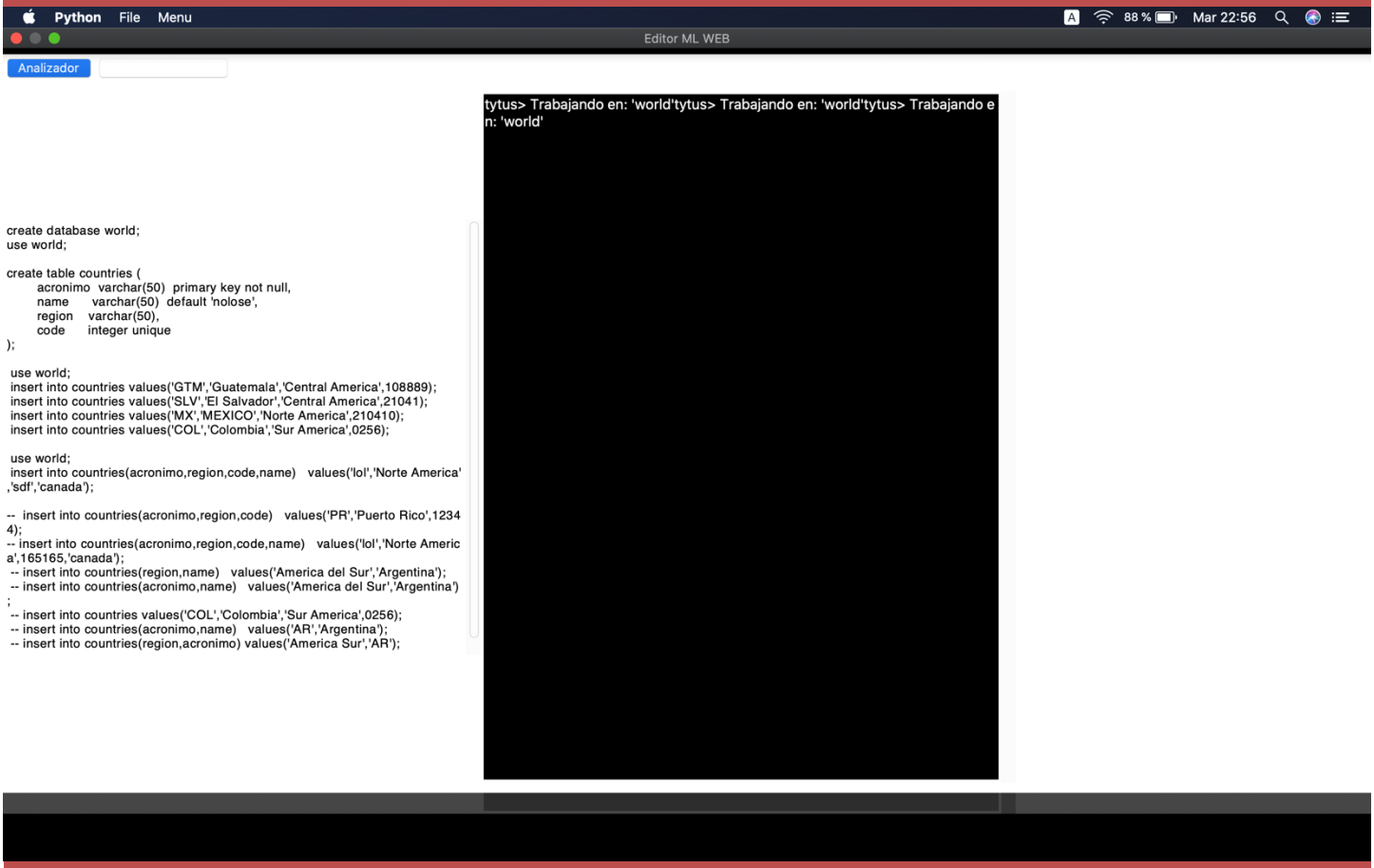
TytusDB

Manual de Usuario
OLC2 fase 1

Universidad San Carlos de Guatemala

Grupo 17

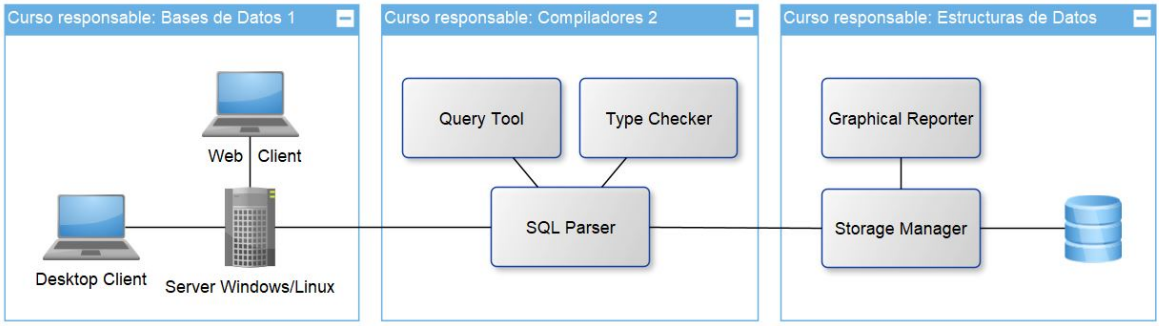
Pablo Rodrigo Barillas 201602503
Christopher Jhoanis Soto 201602569
Edgar Jonathan Arrecis 201602633
Nery Eduardo Herrera 201602870



Interfaz General

La interfaz cuenta con una entrada de texto donde el usuario podra colocar la sintaxis del codigo que desea compilar, y cuenta tambien con una terminal de salida donde podremos ir viendo los resultados del codigo que escribimos, cuenta tambien con dos botones uno de analizar que corre todo el codigo y otro que es analizar seccion que analiza la porcion de codigo que este seleccionado y el diseno cuenta tambien con dos menus desplegables File y Menu

FLUJO DEL PROGRAMA



El proyecto esta dividido en varias etapas este manual se enfocara en el parser de dicho programa es decir la parte de compilacion y de analisis del texto de entrada, a continuacion les explicaremos la sintaxis de dicho codigo.

SINTAXIS DE CODIGO

El codigo se basa en la sintaxis de sql

Numeric Types

Name Storage Size	Description Range
smallint 2 bytes	small-range integer -32768 to +32767
integer 4 bytes	typical choice for integer -2147483648 to +2147483647
bigint 8 bytes	large-range -9223372036854775808 to integer +9223372036854775807
decimal variable	user-specified precision, exact up to 131072 digits before the decimal point; up to 16383 digits after the decimal point

numeric variable	user-specified precision, exact up to 131072 digits before the decimal point; up to 16383 digits
Name Storage Size	Description Range
	after the decimal point
real 4 bytes	variable- precision, 6 decimal digits precision inexact
double precision 8 bytes	variable- precision, 15 decimal digits precision inexact
money 8 bytes	currency -92233720368547758.08 to amount +92233720368547758.07

Character types

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text variable unlimited length	

Comments

Para realizar un comentario utilizaremos la siguiente notacion

```
-- This is a standard SQL comment /* multiline comment

 * with nesting: /* nested block comment */
 */
```

Creacion de tabla

Para crear una tabla se utilziara la siguiente sintaxis, colocandole el nombre deseado

```
CREATE TABLE my_first_table (

column1 type [PRIMARY KEY]
[, column2 type [REFERENCES table]] [, column3...]

);
```

Eliminacion de tabla

Para eliminar una tabla se utilziara la siguiente sintaxis, colocandole el nombre de la tabla que se desea eliminar

```
DROP TABLE my_first_table;
```

Alterar o modificar la tabla

Para agregar modificar y quitar campos de una tabla se utilziara la siguiente sintaxis, colocandole el nombre de la tabla que se desea eliminar y la accion deseada

```
ALTER TABLE table ADD COLUMN column type;

ALTER TABLE products DROP COLUMN description;

ALTER TABLE table DROP CONSTRAINT some_name;
```

Manipulacion de datos

Para inserter una tupla de la tabla utilizaremos la siguiente sintaxis

```
INSERT INTO products VALUES (1, 'Cheese', 9.99);
```

Para modificar una tupla de la tabla utilizaremos la siguiente sintaxis

```
UPDATE products SET price = 10,costo = 9 WHERE price between 5 and 8;
```

Para eliminar una tupla de la tabla utilizaremos la siguiente sintaxis

```
DELETE FROM products WHERE price = 10;
```

Trigonometric Functions

ACOS	`SELECT ACOS(0) AS "Acos (0)";`	La función se usa para devolver el coseno inverso de un argumento dado.	Únicamente en el Select
ACOSD	`SELECT acosd(0.5);`	La Función matemática que devuelve el coseno inverso de la expresión especificada, medido en grados.	Únicamente en el Select, update
ASIN	`SELECT ASIN(0) AS "Asin (0)";`	Se usa para devolver el seno inverso de un argumento dado.	Únicamente en el Select, update
ASIND	`SELECT ASIN(0) AS "AsinD (0)";`	Se usa para devolver el seno inverso de un argumento dado, especificada mediante grados	Únicamente en el Select
ATAN	`SELECT ATAN(0) AS "Atan (0)";`	Tangente inversa	Únicamente en el Select
ATAND	`SELECT ATAND(0) AS "Atan (0)";`	Trangente inversa especificada mediante grados	Únicamente en el select
ATAN2	`SELECT ATAN2(0,1) AS "Atan2 (0,1)";`	Se usa para devolver la tangente inversa de una división dada en el argumento	Únicamente en el select
atan2d	`SELECT ATAN2D(0,1) AS "Atan2 (0,1)";`	Se usa para devolver la tangente inversa de una división dada en el argumento, expresada en grados	Únicamente en el select
COS	`SELECT COS(0) AS "Cos (0)";`	Retorna el coseno de un argumento.	Únicamente en el select
COSD	`SELECT COSD(0) AS "Cosd (0)";`	Retorna el coseno de un argumento,	Únicamente en el select

Para realizar comparaciones en las condiciones utilizaremos los siguientes

LIKE

NOT LIKE

Como por ejemplo

```
string LIKE pattern [ESCAPE escape-character] string NOT LIKE pattern [ESCAPE
escape-character] substring(string, pattern, escape-character)
```

Subqueries

Esto quiere decir que podremos enlazar un query con otro en alguna parte del codigo como podemos ver en el ejemplo

```
FROM table_reference [AS] alias
```

```
FROM (SELECT * FROM table1) AS alias_name
```

Limit and offset

Estas condiciones nos permiten limitar el numero de tuplas que devolvera la consulta

```
SELECT select_list
FROM table_expression
[ ORDER BY ... ]
[ LIMIT { number | ALL } ] [ OFFSET number ]
```

REPORTES

REPORTES DE ERRORES
El siguiente reporte le mostrara al usuario los errores que tienen su código y donde encontrarlos, asi este podrá ubicarlos y mejorarlos.

TIPO	FILA	COLUMNA	DESCRIPCION	AMBIENTE
SINTACTICO	2	20	SE ESPERABA ;	PRINCIPAL
SEMANTICO	14	5	NO SE PUEDE DIVIDIR ENTRE 0	CLASE DIVISION

REPORTE DE TABLA DE SIMBOLOS
El siguiente reporte le mostrara al usuario las variables que se encuentran almacenadas en nuestra tabla de símbolos

TIPO	IDENTIFICADOR	REFERENCIA
TABLA	PAISES	Tabla
TABLA	PERSONAS	Tabla

REPORTE DE GRAMATICA

Dicho reporte desplegara al usuario la gramatica utilizada en el proyecto.

```
ROOT -> SETinstrucciones

SETinstrucciones -> SETinstrucciones SETinstrucciones_paso
            |   SETinstrucciones_paso

SETinstrucciones_paso -> INSTRUCCION
            | INSTRUCCION ';'

INSTRUCCION -> PRINTC
          |   SELECT
          |   UPDATE
          |   ...

// ------------------------------------------------

SELECT ->       'select' 'distinct' CUERPO_SELECT
          |   'select' CUERPO_SELECT

CUERPO_SELECT -> grouping_column_reference 'from' table_expression condition

condition ->  WHERE_condition condition
          | GROUP BY grouping_column_reference condition
```

```
        | , grouping_column_reference condition
        | GROUP BY grouping_column_reference
        | , grouping_column_reference
        | WHERE_condition

WHERE_condition -> 'WHERE' search_condition

grouping_column_reference ->    * , grouping_column_reference
                        | IDen , grouping_column_reference
                        | OBJETO , grouping_column_reference
                        | AGREGACION , grouping_column_reference
                        | *
                        | IDen
                        | OBJETO
                        | AGREGACION

IDen ->     ID
      |   ID as IDen
      |   cadena

search_condition ->     search_condition '=' search_condition
                |   search_condition '!=' search_condition
                |   search_condition '>' search_condition
                |   search_condition '<' search_condition
                |   search_condition '>=' search_condition
                |   search_condition '<=' search_condition
                |   search_condition 'or' search_condition
                |   search_condition 'and' search_condition
                |   search_condition 'like' search_condition
                |   IDen
                |   number
                |   boolean
                |   '(' search_condition ')'


// -------- UPDATE

UPDATE -> 'UPDATE' IDen set L_search_condition WHERE_condition

L_search_condition ->       search_condition L_search_condition
                  |   ',' search_condition L_search_condition
                  |   search_condition

// -------- INSERT

INSERT ->   'insert' 'into' IDen '(' grouping_column_reference ')' INSERT_CUERPO
      |   'insert' 'into' IDen  INSERT_CUERPO

INSERT_CUERPO -> 'values' '(' grouping_column_reference ')'

// ---- DELETE

DELETE -> 'DELETE' 'from' IDen WHERE_condition

// ------------------------------------------------
```

REPORTE DE ARBOL

Mostrara el árbol que se ira formando con forme la entrada del usuario sea recorrida y compilada.