Universidad de San Carlos de Guatemala
Facultad de Ingeniería Escuela de Ingeniería en Ciencias y Sistemas
Organización de Lenguajes y Compiladores 2
Escuela de Vacaciones Diciembre 2023
Catedrático: Ing. Luis Espino
Tutor académico: Ing. Juan Carlos Maeda

XSQL

Contenido

1. Competencias 3
1.1. Competencias Generales 3
1.2. Competencias Específicos 3
2. Descripción 3
3. Características del IDE 3
4. Editor de Texto 6
5. Salida de Datos 6
6. Descripción del lenguaje 8
7. Componentes del lenguaje XSQL 8
7.1 Bases de datos 9
7.2 Tablas 9
7.3 Procedimientos almacenados 9
7.4 Funciones 9
7.5 Llaves primarias 9
7.6 Llave foránea 10
7.7 Tipos de datos 10
7.8 Funciones definidas por el usuario 11
7.9 Funciones del sistema 11
7.10 Expresiones Aritmeticas 14
Precedencia de Operadores 15
7.11 Nombre de Variables 15

7.11.1 Expresiones Relacionales 15
7.12 Sentencias XSQL 16
7.12.1 Create 16
7.12.2 Create Data Base 17
7.12.3 Create Table 17
7.12.4 Create Procedure 18
7.12.5 Create Function 20
7.12.6 Llamado de funciones y Procedures 21
7.12.7 Alter 21
7.12.7.1 Alter Table 21
..... 21
7.12.7.2 Alter Function 22
7.12.7.3 Alter Procedure 22
8 Reportes 22

1. Competencias

1.1. Competencias Generales

Aplicar los conocimientos del curso de Organización de Lenguajes y Compiladores 2 en el desarrollo de una aplicación.

1.2. Competencias Específicos

- Utilizar herramientas para la generación de analizadores léxicos y sintácticos.
- El estudiante construye un intérprete para el lenguaje X-SQL mediante la traducción dirigida por la sintaxis.
- El estudiante utiliza la herramienta PLY o SLY de Python.
- El estudiante traduce mediante un analizador sintáctico ascendente.

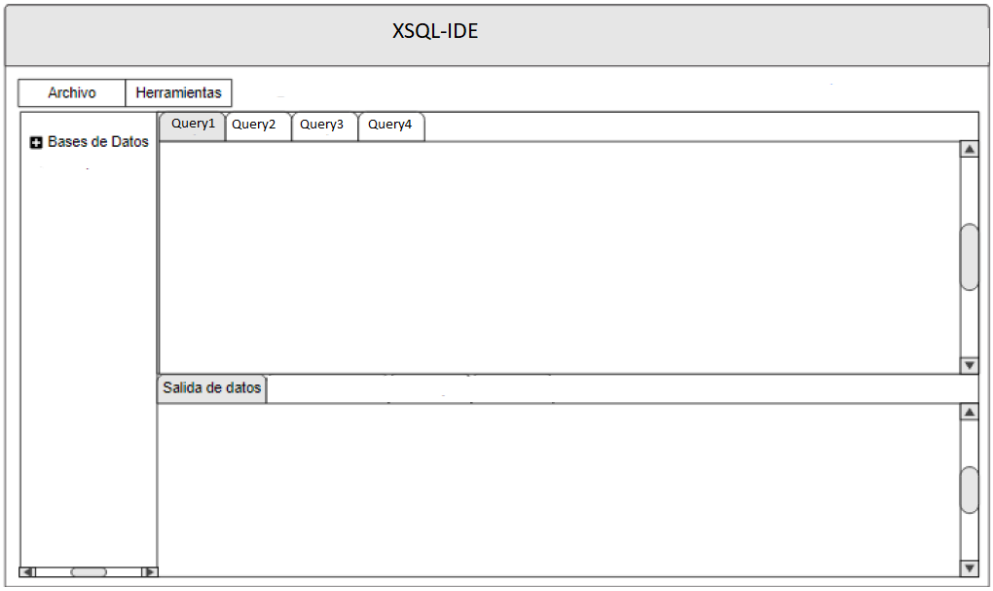
2. Descripción

Se solicita que el estudiante desarrolle un sistema administrador de base de datos, capaz de manejar las instrucciones básicas de un DBMS relacional convencional, dicho sistema recibirá el nombre de XSQL. El servidor de base de datos debe constar con un IDE con el que el usuario interactúa directamente, y tendrá un conjunto de herramientas básicas que permiten el uso fácil de la herramienta.

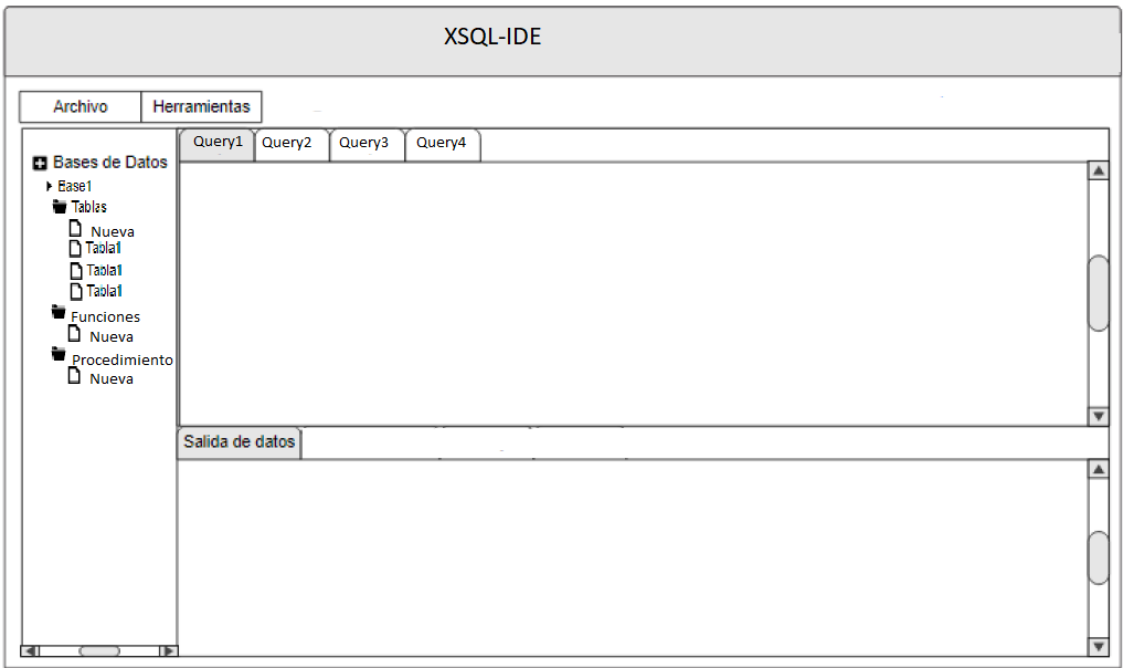
Para almacenar y manipular la información de las bases de datos se deberá manejar un sistema de archivos con formato XML, que contendrá la estructura de las bases de datos creadas, así como las tablas, objetos o procedimientos que contenga. Para la manipulación de la información el estudiante deberá de implementar un intérprete (una forma a discreción del estudiante de leer la información del contenida) XML para leer la información guardada en los archivos.

3. Características del IDE

El IDE debe contar con ciertas características que permitan facilitar al usuario el manejo del DBMS y sus componentes.



Se deberá tener un menú en forma de árbol en el que se debe de mostrar cada base de datos creada. Para cada base de datos se deben de tener los submenús que desplieguen las tablas, funciones, procedimientos con los que cuenta.

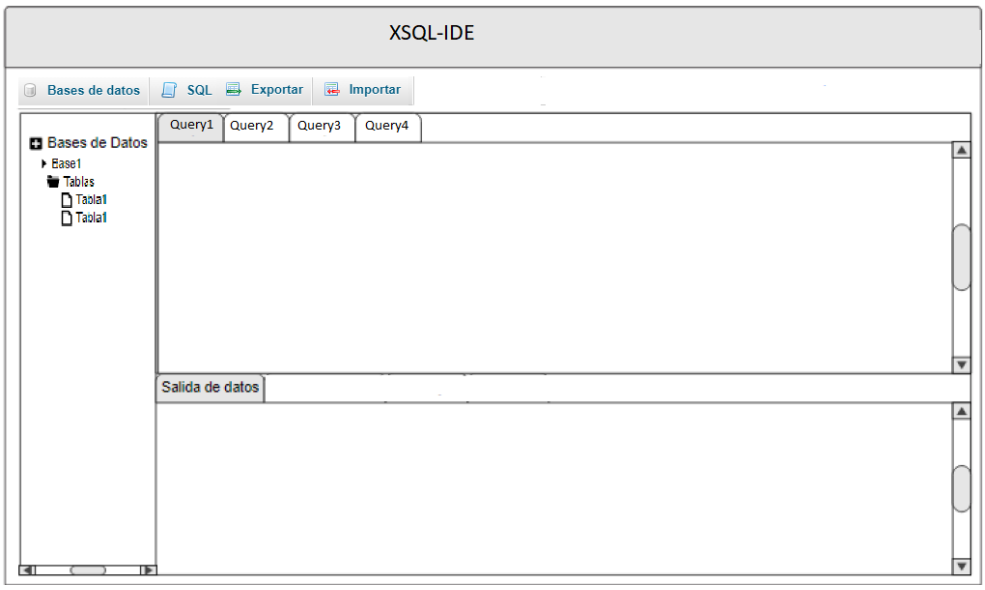


La opción de Archivo debe permitir:

- Nuevo

- Abrir
- Guardar
- Guardar como
- Cerrar
- Salir

En Herramientas debe contar con las siguientes opciones:



- Base de datos debe permitir:
  - Crear una nueva base de datos
  - Eliminar base de datos
  - Crear DUMP: Permite crear un script de la base de datos, crea un archivo con la creación de tablas, funciones y procedimientos. Solamente Estructura
  - Seleccionar Base de datos: Muestra un listado de las bases de datos en el servidor. Se puede seleccionar 1 y mostrar todas las tablas, procedimiento y funciones que contienen.
- SQL
  - Nuevo Query

- Ejecutar Query

- Exportar: Esta opción permite exportar el contenido de una tabla o varias tablas.
- Importar: Permite importar los datos de una o varias tablas a otra base de datos, ya debe existir la estructura.

4. Editor de Texto

La opción del editor permite ingresar el código XSQL, debe tener resaltado de sintaxis, es decir marcar palabras claves de un color, nombre de objetos de otro color.

Ejemplo:

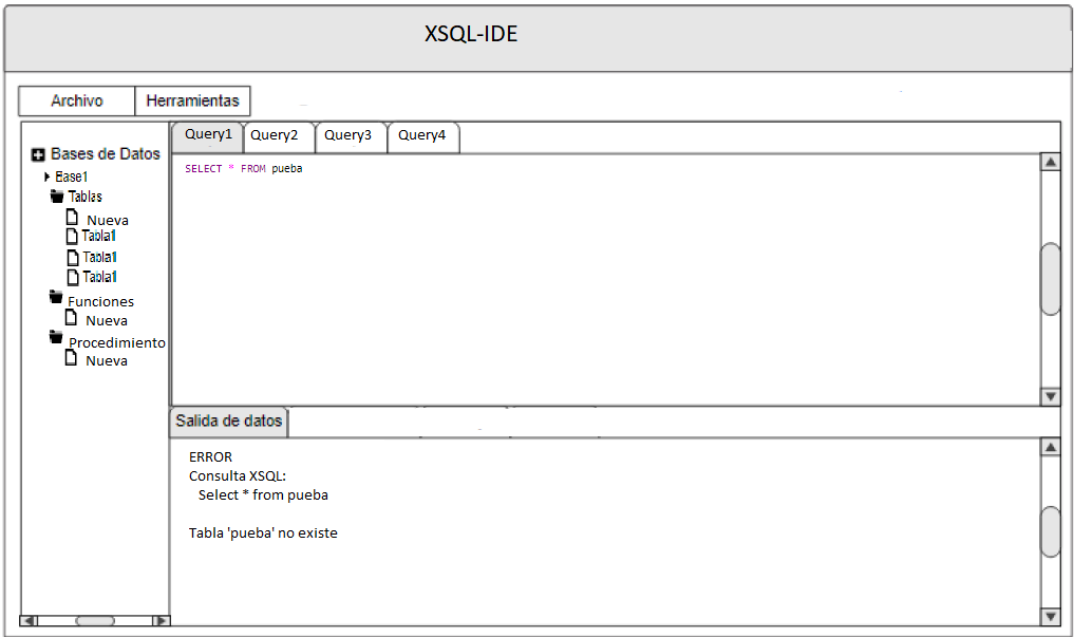
```
INSERT INTO `prueba`(`Id`, `Nombre`, `Direccion`) VALUES (2,'Nombre2','Direccion2')

SELECT * FROM `prueba` WHERE 1
```

Debe ser posible guardar los querys para ser utilizados nuevamente.
Al momento de ejecutar query se debe ejecutar el contenido que se encuentre en el Editor de Texto activo. Pueden existir varias pestañas, y cada una puede tener código, pero solo se debe ejecutar el código de la pestaña visible o activa.

5. Salida de Datos

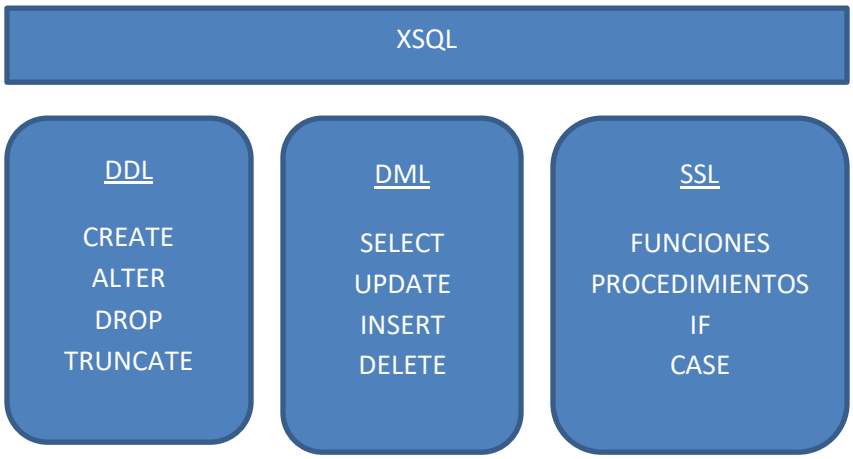
Corresponde al área con forma de tabla, en donde se muestran los datos recuperados, en el caso de no recuperarse ningún dato solamente se mostrará el encabezado de la tabla con el nombre de los campos que se esperaban recuperar.

En el caso de que ocurra algún tipo de error (léxico, sintáctico, semántico), se deberá reportar la posición exacta y palabra que sea la causante.



6. Descripción del lenguaje

El lenguaje XSQL es un lenguaje de bases de datos. Este lenguaje se divide en 4 partes: DML, DDL, DCL y SSL. La estructura del lenguaje se ve representado en el siguiente diagrama:



En donde:

DDL (Data Definition Language por sus siglas en inglés): conjunto de instrucciones de definición de datos. Este set de instrucciones se encarga de definir la estructura de las bases de datos, tablas, procedimientos y objetos.

DML (Data Manipulation Language por sus siglas en inglés): conjunto de instrucciones que manipulan los datos. Este set de instrucciones se encarga de manipular la información siguiendo la estructura definida por las instrucciones DDL.

SSL (System Statement Language por sus siglas en inglés): conjunto de sentencias propias del sistema XSQL, permite la escritura de sentencias de control y funciones propias utilizadas en el manejo de la información.

7. Componentes del lenguaje XSQL

En esta sección se presentan todos los elementos e instrucciones que soportará este lenguaje y que pueden ser ingresados para la administración y manipulación de la información.

7.1 Bases de datos

La base de datos es el conjunto de tablas, procedimientos y funciones diseñada para un acceso fácil y rápido.

7.2 Tablas

Es un conjunto de elementos definidos, los objetos XSQL Tabla deben de pertenecer a una base de datos, y su representación fija son archivos en formato XML.

7.3 Procedimientos almacenados

El objeto XSQL procedimientos almacenados es un conjunto de instrucciones que se guardara en una base de datos para ser utilizada posteriormente.

7.4 Funciones

Son objetos XSQL que retornar un valor escalar, hay funciones definidas por el usuario y funciones propias del sistema de Base de Datos.

7.5 Llaves primarias

Es un tipo de campo que debe ser usado en las tablas y utiliza para identificar los registros dentro de la tabla, un campo que es llave primaria tiene las propiedades de No Nulo (no puede ser vacío), y Único (no se puede repetir con otro campo de la misma tabla) de forma irrevocable.

7.6 Llave foránea

Es un tipo de campo que puede ser usado en las tablas XSQL y servirá para denotar que existe relación entre este campo y la llave primaria de otra tabla que debe de ser especificada. Un campo que es llave primaria tiene las propiedades de No Nulo (no puede ser vacío) de forma irrevocable.

7.7 Tipos de datos

Para las variables se permiten los tipos de datos listados en la siguiente tabla.

Grupo	Tipo de Datos	Descripción	Ejemplo
Numéricos exactos	Int	Tipo de dato que maneja valores numéricos enteros. De -2.147.483.648 a 2.147.483.647	50 -65 100 1251
	Bit	Tipo de datos entero que puede aceptar los valores 1, 0 o NULL	1 0
	Decimal	Tipo de dato que maneja valores numéricos decimales o enteros. Rango entre -2.147.483.648 a 2.147.483.647	17.9 -50.54 111.05 0.05
Fecha y hora	Date	Tipo de dato para almacenar fechas en formado dd-mm-yyyy Del 1 de enero de 1753 hasta el 31 de diciembre de 9999	12-07-2020

	Datetime	Tipo de dato para - almacenar la fecha y la - hora en formato dd-mm-yyyy hh:mm:ss	12-07-2020 23:54
Cadenas de Caracteres	Nchar(n)	Dato carácter de longitud fija, con n caracteres, n debe estar entre 1 y 4,000	“Hola” “Proyecto 1”
	Nvarchar(n)	Dato carácter de longitud variable, el valor máximo de n puede ser 2,000,000.	“Adiós”

7.8 Funciones definidas por el usuario

Las funciones definidas por el usuario en XSQL serán únicamente de los tipos Numéricos exactos, no permite otro tipo de dato para las funciones.

Ejemplo

```
CREATE FUNCTION suma
(
@val1 AS int,
@val2 AS int
)
RETURNS int
AS
BEGIN
        Declare @valor int
        Set @valor = @val1 + @val2

        RETURN @valor

END
```

7.9 Funciones del sistema

El sistema debe contar con funciones propias, ya definidas.

Nombre	Descripción	Parámetros	Valor que retorna
CONCATENA	Concatena dos valores tipo	(cadena,cadena) nchar, nchar ó nchar, nvarchar ó	Nvarchar

	cadena de caracteres	nvarchar, nvarchar	
SUBSTRAER	Extrae una porción de una cadena de caracteres	(texto, inicio, longitud) Nvarchar, int, int ó Nchar, int, int	nvarchar
HOY	Devuelve la fecha y hora del sistema		datetime
CONTAR	Devuelve el número de filas de una tabla que cumplen con cierto criterios		Int
SUMA	Devuelve la suma de una columna numérica	(nombre de la columna o número de la columna)	Decimal
CAS	Convierte una columna o variable a un valor definido	(Variable as Valor)	Según el valor seleccionado en e CAS

Ejemplo

CONCATENAR

```
SELECT CONCATENAR('HOLA','MUNDO')
```

Resultado

HOLAMUNDO

SUBSTRAER

```
SELECT SUBSTRAER('HOLAMUNDO',1,4)
```

Resultado

HOLA

HOY

```
SELECT HOY()
```

Resultado

13-07-2020 23:50

CONTAR

```
SELECT CONTAR(*) FROM TABLA WHERE CAMPO1 = 1
```

Resultado

100

SUMA

```
SELECT SUMA(IVA) FROM TABLA WHERE CAMPO1 = 1
```

Resultado

2152.15

CAST Ejemplo

SQL	Valor Salida
Select CAST(@BIT AS INT)	**1 o 0 (Entero)**
Select CAST(@NCHAR AS INT) o Select CAST(@NVARCHAR AS INT)	**Valor ASCII del texto si es un valor, y si es una cadena es la sumatoria de los ASCII de la cadena.**
Select CAST(@INT AS VARCHAR)	**Si el valor esta entre 1 y 255 imprimir el valor ASCII. Si es mayor a 255 devolver NULL**
Select CAST(@BIT A VARCHAR)	**Devuelve 0 y 1 según el valor**

7.10 Expresiones Aritmeticas

El lenguaje XSQL tendrá las operaciones aritméticas básicas: suma, resta, multiplicación, división.

	Bit	Int/Decimal	Date/Datetime	Nchar/Nvarchar
Suma (+)				
Bit	OR	Suma	Error	Concatenar
Int/Decimal	Suma	Suma	Error	Concatenar
Date/Datetime	Error	Error	Error	Error
Nchar/Nvarchar	Concatenar	Concatenar	Error	Concatenar
Resta (-)				
Bit	Error	Resta	Error	Error
Int/Decimal	Resta	Resta	Error	Error
Date/Datetime	Error	Error	Error	Error
Nchar/Nvarchar	Error	Error	Error	Error
Multiplicación (*)				
Bit	AND	Multiplicación	Error	Error

Int/Decimal	Multiplicación	Multiplicación	Error	Error
Date/Datetime	Error	Error	Error	Concatenación
Nchar/Nvarchar	Error	Error	Concatenación	Error
División (/)				
Bit	Error	División	Error	Error
Int/Decimal	División	División	Error	Error
Date/Datetime	Error	Error	Error	Concatenación
Nchar/Nvarchar	Error	Error	Concatenación	Error

Nota: No hay operaciones con la función SUMA.

Precedencia de Operadores

Símbolo	Precedencia
+ -	1
*** /**	2
== != <> <= >=	3
||	4
&&	5
()	9

7.11 Nombre de Variables

Los nombres de las variables están definidos como una sucesión de caracteres que inicia con una arroba (@), seguido de una letra que puede estar sucedida por cero o más caracteres como letras, dígitos. Se deben declarar precedidas por la palabra reservada DECLARE.

7.11.1 Expresiones Relacionales

Nombre	Símbolo	Descripción
Igual	**==**	Esta operación comprueba si dos expresiones tienen el mismo valor, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Diferente	**!=**	Esta operación comprueba si dos expresiones tienen distinto valor, de ser así

		retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor que	**<**	Esta operación comprueba si la primera expresión es menor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor que	**>**	Esta operación comprueba si la primera expresión es mayor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor o igual	**<=**	Esta operación comprueba si la primera expresión es menor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor o igual	**>=**	Esta operación comprueba si la primera expresión es mayor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).

7.11.2 Expresiones Logicas

Nombre	Símbolo	Descripción
AND	**&&**	**Este operador comprueba que tanto la primera expresión como la segunda expresión tengan valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).**
OR	**||**	**Este operador comprueba que la primera expresión o la segunda expresión tengan valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).**
NOT	**!**	**Este operador cambia el valor de la expresión de la que viene acompañada.**

7.12 Sentencias XSQL

En esta sección se detallan los comandos o sentencias que puede manejar el DBMS.

7.12.1 Create

Permite crear un objeto de XSQL, puede ser una base de datos, tabla, funciones, procedimientos.

7.12.2 Create Data Base

Sintaxis para crear una base de datos, una base de datos también se puede crear desde la consola con un botón, es una opción en el menú.

Sintaxis

```
CREATE DATA BASE Nombre_base_de_datos;
```

Crea una base de datos, es decir la estructura XML para manejar una base de datos. Se debe validar que la base de datos no exista ya dentro de su motor.

El nombre de la base de datos no debe contener caracteres especiales (+,-,*,/,%,$,%,(,),",!?), se puede utilizar _ en el nombre.

7.12.3 Create Table

Permite crear una tabla dentro de una base de datos, esto quiere decir que se debe crear dentro de un entorno de base de datos. La base de datos se puede seleccionar por defecto al crearla, o se puede usar la palabra reservada **USAR NombreBasedeDatos** para luego crear objetos sobre esa Base de datos.

Sintaxis

```
CREATE TABLE NombreTabla  (
    Columna1 tipodato,
    Columna2 tipodato,
    Columna3 tipodato,
   …..
)
```

Se debe validar que no exista una tabla con ese nombre dentro de la misma base de datos.

Ejemplo

```
CREATE TABLE Persona (
        ID int NOT NULL PRIMARY KEY,
        PrimerNombre varchar(150) NOT NULL,
        SegundoNombre varchar(150) NULL,
        FechaNacimiento datetime,
        Identificacion int,
                FOREING KEY (Identificacion) REFERENCE
                Identificaciones(Identificacion)
        );
```

El nombre de la tabla no debe contener caracteres especiales (+,-,*,/,%,$,%,(,),",!?), se puede utilizar _ en el nombre.

- NULL o Not Null: Este complemento se utiliza para especificar si se aceptara el valor nulo en este campo.
- Primary Key: Este complemento se utilizará para especificar si el campo será llave principal de la tabla.
- Foreing Key: Este complemento se utilizará para especificar si el campo será llave foránea. Seguido del nombre de la tabla a la que se hace referencia. Se deberá validar que exista la tabla y el atributo que se está referenciando, así como que el tipo de dato sea el correcto. Al momento de insertar un valor en la tabla se deberá validar que el valor de la llave foránea exista en la tabla referenciada.

7.12.4 Create Procedure

Un procedimiento es un objeto dentro del motor en el que se permiten ejecutar sentencias XSQL, consultas, actualizaciones, inserciones, eliminación, pero no retornan ningún valor, pueden recibir parámetros.

Sintaxis

```
CREATE PROCEDURE procedure_name

AS

XSql_Sentencias

;
```

Ejemplo

```
CREATE PROCEDURE inicializacomisiones (@Ciudad nvarchar(30),
@Departamento varchar(10))

AS

begin

UPDATE tbcomision set comision = 0 where ciudad = @Ciudad and
Departamento = @Departamento and mes = month(getdate());

TRUNCATE TABLE tbreportecomisiones;

END
```

La ejecución puede ser

```
EXEC inicializacomisiones @Ciudad =‘Guatemala’ ,@Departamento = ‘Guatemala’

o

EXEC inicializacomisiones ‘Guatemala’ , ‘Guatemala’
```

El nombre de los procedimientos no debe contener caracteres especiales (+,-,*,/,%,$,%,(,),”,!?), se puede utilizar _ en el nombre.

7.12.5 Create Function

Una Función es un Objeto dentro del Motor que permite ejecutar sentencias XSQL ejecutarlas y retornar un valor, pueden ser llamadas dentro de otras sentencias XSQL como por ejemplo dentro de un Select.

Sintaxis

```
CREATE FUNCTION function_name (@parametro tipodato)

RETURN tipodato

AS

BEGIN

      XSql Sentencias;

END;
```

Ejemplo

```
CREATE FUNCTION Retornasuma(@ProductID int)
RETURNS int
AS
-- Returns the stock level for the product.
BEGIN
    DECLARE @ret int;
    SELECT @ret = SUM(Cantidad)
    FROM inventario
    WHERE ProductoId = @ProductID

     IF (@ret == NULL)
        SET @ret = 0;
    RETURN @ret;
END;
```

El Return debe ser del tipo de dato declara en el objeto. Finaliza la función.

7.12.6 Llamado de funciones y Procedures

Las funciones de usuario o del sistema pueden ser llamadas así:

```
Select retornasuma(id) Suma
From producto
Where Pais = 'Guatemala'
```

```
Create Procedure Valores
Begin
        Declare @SumaProducto Decimal
        Set @SumaProducto = retornaSuma(1)
end
```

```
EXEC SP_CARGA_DATOS;
```

7.12.7 Alter

Esta sentencia permite hacer modificaciones a las estructuras de tablas, procedimientos y funciones.

7.12.7.1 Alter Table

Permite modificar la estructura de la tabla agregar o eliminar columnas.

Ejemplo

```
        ALTER TABLE NOMBRETABLA ADD
NOMBRECOLUMNA TIPO
        ALTER TABLE NOMBRETABLA
DROP NOMBRECOLUMNA
```

7.12.7.2 Alter Function

Permite modificar la estructura de una función, modificar el contenido de la función, cambio en parámetros, resultado.

7.12.7.3 Alter Procedure

Permite modificar la estructura de un procedimiento, modificar el contenido del procedimiento.

8 Reportes

Los reportes deben ser generados después de una ejecución, ya sea correcta o incorrecta. Estos reportes deben estar asociados al código que tiene el “focus” del IDE, si se cambia de ventana de código y ya se había ejecutado puede mostrar los reportes del otro código.

8.1 Reporte de errores

En IDE debe generar los siguientes reportes:

1. Errores léxicos
2. Errores sintácticos
3. Errores semánticos

Cada reporte debe contener como mínimo, el tipo, la descripción, y el número de línea.

8.2 Reporte de la tabla de símbolos

El IDE debe mostrar la tabla de símbolos después de la ejecución de un archivo, mostrando las variables, funciones y procedimientos con mínimo los siguientes

datos: identificador, tipo, dimensión, valor, declarada en (ámbito), y referencias (estructura interna).

8.3 Reporte del AST

El IDE, después de una ejecución, con el analizador sintáctico ascendente, en una pestaña nueva, el árbol sintáctico abstracto (definición 2.5.1 del libro de texto) utilizando Graphviz (opcional), puede ser una imagen o un documento.

9 Entregables

La entrega debe tener los siguientes productos:

1. Código fuente listo para su ejecución
2. Gramáticas utilizadas Documento detallando las gramáticas
3.-Reglas de optimización utilizadas

10 Consideraciones

Se debe tomar en consideración lo siguiente:

10.1. El proyecto es parejas y debe generar una aplicación de escritorio escrita en Python.

10.2. Se deben utilizar dos tipos de gramáticas para el analizador ascendente

10.3. Copias de proyectos obtendrán una nota 0, por lo que pierde automáticamente el laboratorio

10.4. Durante la calificación se verificará la autoría mediante preguntas, si no las responde se considera copia.

10.5 Debe hacer la generación en código a 3 direcciones, para esto deben mostrar una pantalla adicional en donde se pueda visualizar la generación de código a 3 direcciones y optimización de código.

El código de 3 direcciones se generara únicamente para **una clase especial** de Store Procedure que contendrá IF (estos pueden ser anidados) y Ciclos While

(Estos no serán anidados), CASE (estos no pueden ser anidados) esta clase de store procedures no tendrá SELECT, ni UPDATE, ni DELETE.

Ejemplo

```
CREATE PROCEDURE sp_conteo (@i AS integer)
AS
BEGIN
    DECLARE @contador AS integer
    if(@i != 0)
    begin
        WHILE @contador < @i
        BEGIN
            SELECT CONCAT(" EJECUCION No.",@i);
            SET @i= @i+1;
        END

    end
    ELSE
    begin
            SELECT "NO HAY NADA QUE HACER";
    end
END
```

11 Entrega

Los aspectos de la entrega son:

1. La entrega será virtual, se debe utilizar GitHub https://github.com/, deben colocar el repositorio de manera privada y dar permiso al auxiliar.

2. La fecha de entrega límite del proyecto es el lunes 25 de Diciembre a las 23:59. Se puede entregar antes para evitar contratiempos, ya que después de esa hora NO SE RECIBE NINGÚN PROYECTO.

3. La calificación se realizará de manera virtual (ya sea en meet o zoom) con las cámaras activadas, cada calificación será almacenada.

12 Anexo

Sintaxis SQL

Definición de datos

CREATE TABLE

```
CREATE TABLE products (
    product_no int PRIMARY KEY,
    name nvarchar(1000),
    price decimal
);


CREATE TABLE tbfactura (
       Idfactura int PRIMARY KEY,
       Fechafactura date not null,
       Nit nvarchar(20) not null,
       Nombrecliente nvarchar(50) not null,
       Referencia nvarchar(10)
);

CREATE TABLE tbdetallefactura (
    iddetallefac int PRIMARY KEY,
    idfactura int REFERENCE tbfactura(idfactura),
    product_no int REFERENCE products (product_no),
    price decimal NOT NULL,
    cantidad decimal NOT NULL

);
```

ALTER TABLE

```
Alter table products add column inventario decimal;
Alter table tbfactura add column formapago int;
Alter table tbfactura add column tipotarjeta int;
```

TRUNCATE TABLE

```
Truncate table tbfactura;
Truncate table tbdetallefactura;
```

DROP

```
Alter table tbfactura drop column tipotarjeta ;
DROP TABLE tbproducts;
```

MANIPULACION DE DATOS

SELECT

SELECT * from tbproducto;

SELECT * from tbfactura where fechafactura between ´2023-01-01’ and getdate();

SELECT * from tbdetallefactura,tbproducto where (cantidad*tbdetallefactura.price)/1.12 > 100

And tbproducto.idproducto = tbdetallefactura.tbproducto;

SELECT idfactura,(cantidad*price)/1.12 iva,(cantidad*price)*0.07 isr, nit

From tbdetallefactura where fechafactura between ´2023-01-01’ and getdate()

And idproducto = 65;

UPDATE

UPDATE tbproducto set nombreproducto=’Queso’,precio = 150,idunidad = 2 where idproducto = 65;

UPDATE tbproducto set nombreproducto=’Queso’,precio = 150,descuento = precio*0.10,idunidad = 2 where idproducto = 65;

INSERT

```
INSERT INTO tbvalores (id,nombre,bandera) VALUES(1,'JULIO LOPEZ',1);
INSERT INTO tbvalores (id,nombre,bandera) VALUES(2,'CARLOS JUAREZ',1);
INSERT INTO tbvalores (id,nombre,bandera) VALUES(3,'CARLA PEREZ',1);
```

DELETE

```
DELETE FROM tbvalores where id = 3;
```

IF Y CASE

```
select if(28>11,"Verdadero","falso");
```

```
select nombre,edad, if(edad >= 18,"Mayor de edad", "Menor de Edad") as estado
from tbmedico;

SELECT nombre,
        case
        when edad > 18 and edad <= 25
                then 'Adolecente'
         when edad > 25 and edad <= 35
                then 'Adulto joven'
         when edad >35 and edad <= 45
                then 'Adulto Maduro'
         else
                then 'Adulto Mayor'
        end clasificacion
 from tbpersona;

CREATE PROCEDURE sp_nuevoprocedimiento(@MONTO AS DECIMAL,@IDFACTURA INT)
AS
BEGIN
        DECLARE @IVA DECIMAL;
        DECLARE @ISR DECIMAL;

   SET @IVA = @MONTO - @MONTO/1.12;
   IF @MONTO > 2000 THEN
                SET @ISR = @MONTO *0.07;
        ELSE
                SET @ISR = @MONTO *0.10;
   END IF;
```

```
    UPDATE tbfactura set monto = @monto where idfactura = @idfactura;

END;
```