* ***Escriba la sintaxis, en formato BNF, de las sentencias del lenguaje SQLXXX\_β (donde XXX es su nombre), que cumpla con las siguientes consignas:***

**Sentencia: select**

**select** **[all | distinct]** ListaColumnas **from** ListaTablas **where** Condición **order by** ListaColumnasOrd

donde:

* [all | distinct] Los corchetes indican que son opcionales. El pipe indica que en la sentencia va **all** o **distinct**, una de las dos opciones.
* ListaColumnas: es una lista de una o más nombres de columnas de tablas separadas por comas.
* ListaTablas: es una lista de una o más nombres de tablas, separadas por coma.
* Condición especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and** y **or**.
* *ListaColumnasOrd* es una lista de uno o más nombres de columnas con su ordenamiento. Debe especificarse para cada nombre de la lista de columnas el ordenamiento **asc** o **desc**: Ejemplo: col1 asc, col2, col3 desc
* Las palabras reservadas **select, from, where, order by, all, distinct, and, or, asc y desc** son terminales.
* Los nombres de columnas y tablas serán terminales **id.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Sentencia: update**

**update** Tabla **set** Columna1=valor1, Columna2=valor2,.. Columnan=valorn [**where** Condición]

donde:

* Columnai: es el nombre de una columna. Puede ser una columna o más de una, cada una con la asignación de su valor.
* Valori: es una constante numérica o alfanumérica.
* Tabla: es el nombre de una tabla.
* [**where** Condición] especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and** y **or.** Los corchetes indican que la condición es opcional.
* Las palabras reservadas **set, and or y where** son terminales.
* Los nombres de columnas y tablas serán terminales **id.**
* Los valores constantes de las columnas serán terminales **cte.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Sentencia: delete**

**delete from**  Tabla [**where** Condición]

* Tabla: es el nombre de una tabla.
* [where Condición] especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and y or**. Los corchetes indican que la condición es opcional.
* Las palabras reservadas **delete, from, where, and y or** son terminales.
* Los nombres de tablas serán terminales **id.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Sentencia: insert**

**insert into** Tabla (Campo1, Campo2, …Campon) **values** (Valor1, Valor2, ..Valorn)

donde

* Campoi: es el nombre del campo de una tabla. Puede ser uno o más campos.
* Valori: es una constante numérica o alfanumérica. La cantidad de valores debe ser igual a la cantidad de campos.
* Tabla: es el nombre de una tabla.
* Las palabras reservadas **insert, into, values** son terminales.
* Los nombres de campos y tablas serán terminales **id.**
* Los valores de los campos serán terminales **cte.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

Definir las reglas gramaticales adicionales necesarias para poder escribir un script de sentencias que incluya las sentencias definidas, donde pueda haber una o más sentencias separadas por punto y coma.

* ***Script de sentencias del lenguaje SQLXXX\_β*** : escriba un script (cadena) que cumpla con las reglas de sintaxis de su lenguaje de programación, y arme el árbol de derivación de la misma.
* ***Implementación Jflap***: Diseñar el Parser LL o LR de la gramática de alguna de las sentencias del lenguaje de programación, en el aplicativo mencionado. Tenga en cuenta que Jflap no acepta notación BNF. Modifique la notación para que pueda implementar el parser.
* ***Entregables:*** Documento digital que contenga:
* Las reglas de la gramática del lenguaje de programación,
* El script con las sentencias del lenguaje,
* El árbol de derivación de dicho script,
* La gramática de la sentencia elegida para armar el parser LL o LR,
* Parser LL o LR de la gramática de la sentencia elegida y el análisis sintáctico de una cadena que acepte y una que no acepte el parser, junto con el archivo de extensión .flap del parser diseñado.

Respuesta:

**<select>** 🡪 **select <listacol> from <listatab> where <listacond> orderby <listacolord>**

**<sselect>** 🡪 **select <resul> <listacol> from <listatab> where<listacond> order by <listacolord>**

**<resul>** 🡪 **all | distinct**

**<listacol>**🡪 **id | <listacol>,id**

**<listatab>**🡪 **id | <listatab>, id**

**<listacond>** 🡪 **<condbinaria> | <listacond> and <condbinaria> | <listacond> or <condbinaria>**

**<condbinaria>** 🡪 **id <opbinario> id**

**<opbinario>** 🡪 **>=, <=, == >, < ,** !=

**<listacolord>**🡪 **id <ord> | id | <listacol>, id <ord> | <listacol>, id**

**<ord>** 🡪 **asc | desc**

**<update> 🡪 update id set <listacolval> where <listacond>**

**<listacolval> 🡪 id=cte | <listacolval>, id=cte**

**<delete> 🡪 delete from id where <listacond>**

**<insert> 🡪 insert into id <camposvalores>**

**<camposvalores> 🡪 (id, <camposvalores1> , cte)**

**<camposvalores1> 🡪 id, <camposvalores1> , cte | id ) values (cte**

**<script> 🡪 <sentencias>**

**<sentencias> 🡪 <sentencia>; | <sentencias>**

**Script: select distinct id,id from id where id==id;**

**update id set id=cte where id==id;**

**Para parser: Gramática: D 🡪 delete from id where L**

**L 🡪 B | L and B | L or B**

**B 🡪 id O id**

**O 🡪 == | > | < | >=**

Falta hacer parser y análisis sintáctico de cadenas.