**TEMA: CONSULTAS AVANZADAS CON SQL SERVER**

**OBJETIVO**:

Realizar consultas avanzadas (Multitablas, Combinadas, Subconsultas, etc) y funciones de agregado del lenguaje SQL utilizando SQL Server

## PARTE 1. SENTENCIAS SQL PARA CONSULTA DE DATOS

A continuación encontrara un listado de las principales sentencias SQL utilizadas para realizar diferentes tipos de consulta sobre una base de datos:

1. **SELECT;** La palabra clave SELECT nos permite tomar toda la información de una o varias columnas de una tabla

   *Sintaxis:* SELECT columna1,columna2,…ColumnaN FROM nombre_tabla

2. **DISTINCT:** La palabra clave DISTINCT Permite seleccionar valores de una tabla mostrándolos una sola vez, inclusive aquellos que estén repetidos

   *Sintaxis*: SELECT DISTINCT columna1,columna2,…ColumnaN FROM nombre_tabla

3. **WHERE:** La palabra clave WHERE se utiliza para seleccionar condicionalmente los datos de una tabla

   *Sintaxis*: SELECT columna1,columna2,…ColumnaN FROM nombre_tabla WHERE Condición

4. **AND/OR:** La palabra clave WHERE selecciona datos condicionalmente desde una tabla. Esta condición puede ser una condición simple o puede ser una condición compuesta. Las condiciones compuestas están formadas por múltiples condiciones simples conectadas por AND u OR. No hay límites en el número de condiciones simples que pueden presentarse en una sola instrucción SQL.

   *Sintaxis:* SELECT columna1,columna2,…ColumnaN FROM nombre_tabla
   WHERE "condición simple" {[AND|OR] "condición simple"}…

5. **IN:** La palabra clave IN Permite seleccionar los registros que contengan unos valores (varios) en un mismo campo o columna

   *Sintaxis:* SELECT columna1,columna2,…ColumnaN FROM nombre_tabla WHERE
   nombre_columna IN ('valor1', 'valor2', ...)

6. **BETWEEN:** La palabra clave BETWEEN permite la selección de registros que de un rango de valores

   Sintaxis: SELECT columna1,columna2,…ColumnaN FROM nombre_tabla WHERE
   nombre_columna BETWEEN 'valor1' AND 'valor2'

7. **LIKE:** La palabra clave LIKE se utiliza en la cláusula WHERE. Básicamente, LIKE permite hacer una búsqueda basada en un patrón en vez de especificar exactamente lo que se desea

   Sintaxis: SELECT columna1,columna2,…ColumnaN FROM nombre_tabla
   WHERE nombre_columna LiKE {patrón}

   El {patrón} que acompaña la cláusula LIKE consiste en comodines tales como:

   'ABC%': Selecciona todos los datos que comienzan con 'ABC'. Por ejemplo, 'ABCD' y 'ABCABC' ambas deberían satisfacer la condición.
   '%XYZ': Selecciona todos los datos que terminan con 'XYZ'. Por ejemplo, 'WXYZ' y 'ZZXYZ' ambas deberían satisfacer la condición.
   '%AN%': Selecciona todos los datos que contienen el patrón 'AN' en cualquier lado. Por ejemplo, 'LOS **AN**GELES' y 'SAN FR**AN**CISCO' ambos deberían satisfacer la condición.

8. **ORDER BY:** La palabra clave ORDER BY se utiliza cuando se necesita enumerar el resultado en un orden particular. Esto podría ser en orden ascendente, en orden descendente, o podría basarse en valores de texto

   Sintaxis: SELECT columna1,columna2,…ColumnaN FROM nombre_tabla WHERE
   Condición ORDER BY nombre_columna [ASC/DESC]

9. **COUNT:** Es una función de SQL Server que permite contar el número de filas en una consulta determinada.

   Sintaxis: SELECT COUNT(nombre_columna) FROM nombre_tabla

10. **MAX:** Es una función de SQL Server que permite encontrar el número mayor en una columna determinada.

    Sintaxis: SELECT MAX(nombre_columna) FROM nombre_tabla

11. **MIN:** Es una función de SQL Server que permite encontrar el número menor en una columna determinada.

    Sintaxis: SELECT MIN(nombre_columna) FROM nombre_tabla

12. **SUM:** Es una función de SQL Server que permite realizar la suma de todos los valores almacenados en una determinada columna

    Sintaxis: SELECT SUM(nombre_columna) FROM nombre_tabla

13. **AVG:** Es una función de SQL Server que permite calcular el Promedio de todos los valores almacenados en una determinada columna

    Sintaxis: SELECT AVG(nombre_columna) FROM nombre_tabla

14. **GROUP BY:** La palabra clave GROUP BY se utiliza cuando se selecciona columnas múltiples desde una tabla (o tablas) y aparece al menos un operador aritmético en la instrucción SELECT.

Cuando esto sucede, se puede agrupar todas las otras columnas seleccionadas, *es decir*, todas las columnas excepto aquella(s) que se operan por un operador aritmético.

Sintaxis: SELECT columna1, SUM(Columna2) FROM nombre_tabla GROUP BY columna1

15. **HAVING:** La palabra clave HAVING se utiliza para limitar el resultado de una consulta según una determinada condición aplicando generalmente una función de agregado como por ejemplo SUM, AVG, etc

Sintaxis: SELECT columna1, SUM(Columna2) FROM nombre_tabla GROUP BY columna1 HAVING (condición de función aritmética)

* El uso del GROUP BY es Opcional

** Condición de función aritmética ejemplo:

SUM(Columna) > 1000 ó AVG(Columna) < 5000

16. **AS:** La palabra Clave AS se utiliza para proporcionarle un Alias o un nombre alternativo a cada una de los valores consultados sean columnas o sea el resultado de una función Aritmética

**Sintaxis:** SELECT columna1 AS OtroNombre1, SUM(Columna2) As Total , AVG(Columna3) AS Promedio FROM nombre_tabla

Para ampliar información y ver ejemplos de cómo aplicar estas sentencias de consulta sql puede ingresar a los siguientes enlaces web

http://deletesql.com/

http://deletesql.com/viewforum.php?f=5&sid=e67b23df988b4c5c151ad6d1b3c94717

## PARTE 2. PRODUCTO CARTESIANO, CONSULTAS COMBINADAS Y SUBCONSULTAS SQL

### a) PRODUCTO CARTESIANO SQL

En SQL un producto cartesiano entre dos tablas es una consulta que obtiene los datos combinando todos los elementos o registros de la TABLA1 con todos los registros de la TABLA2, de manera que cada fila de resultado es una de las combinaciones posibles. Por tanto en un producto cartesiano el número de filas resultantes será igual al número de registros de la TABLA1 multiplicado por el número de registros de la TABLA2.

La estructura general de un producto cartesiano SQL es:

***SELECT*** *Campos* ***FROM*** *TABLA1 , TABLA2 , TABLA3 , …TABLA N*

El producto cartesiano SQL se puede aplicar para combinar de datos de N tablas y se le puede agregar filtros usando WHERE al final de la sentencia SQL.

### b) CONSULTAS COMBINADAS SQL

Habitualmente cuando se requiere recuperar la información de una base de datos en muchos casos dicha información se encuentra repartida en varias tablas, referenciadas a través de varios códigos o id's. Sin embargo está forma de almacenar la información no resulta muy útil a la hora de consultar los datos. Es por esta razón que SQL proporciona alternativas de consulta que facilitan mostrar la información que se encuentra repartida entre varias tablas, estas son las denominadas consultas combinadas o JOINS.

Las consultas combinadas pueden ser de tres tipos:

Combinación interna
Combinación externa
Uniones

Para aplicar Consultas Combinas se emplea la sentencia JOIN, la cual permite combinar registros de dos o más tablas en una base de datos relacional.

La combinación interna o "INNER JOIN" es la sentencia JOIN por defecto, y consiste en combinar cada fila de una tabla con cada fila de la otra tabla, seleccionado aquellas filas que cumplan una determinada condición. Con esta operación se calcula el producto cruzado de todos los registros; así cada registro en la tabla A es combinado con cada registro de la tabla B; pero sólo permanecen aquellos registros en la tabla combinada que satisfacen las condiciones que se especifiquen. Este es el tipo de JOIN más utilizado, por lo que es considerado el tipo de combinación predeterminado.

La sintaxis genérica de un INNER JOIN en el lenguaje SQL es:

***SELECT*** *Campos* ***FROM*** *TABLA1*
***INNER JOIN*** *TABLA2* ***ON*** *TABLA1.CampoPK = TABLA2.CampoFK*

Las combinaciones externas tales como LEFT OUTER JOIN y RIGHT OUTER JOIN, son operaciones de consulta donde no se requiere que cada registro en las tablas a tratar tenga un registro equivalente en la otra tabla. El registro es mantenido en la tabla combinada si no existe otro registro que le corresponda; este tipo de operación se subdivide dependiendo de la tabla a la cual se le admitirán los registros que no tienen correspondencia, ya sean de tabla izquierda (LEFT), de tabla derecha (RIGHT) o combinación completa.(FULL). Para aplicarlas empleando el lenguaje SQL se hace usando la misma sintaxis del INNER JOIN pero usando la palabra clave correspondiente en lugar de la palabra "INNER"

### c) SUBCONSULTAS SQL

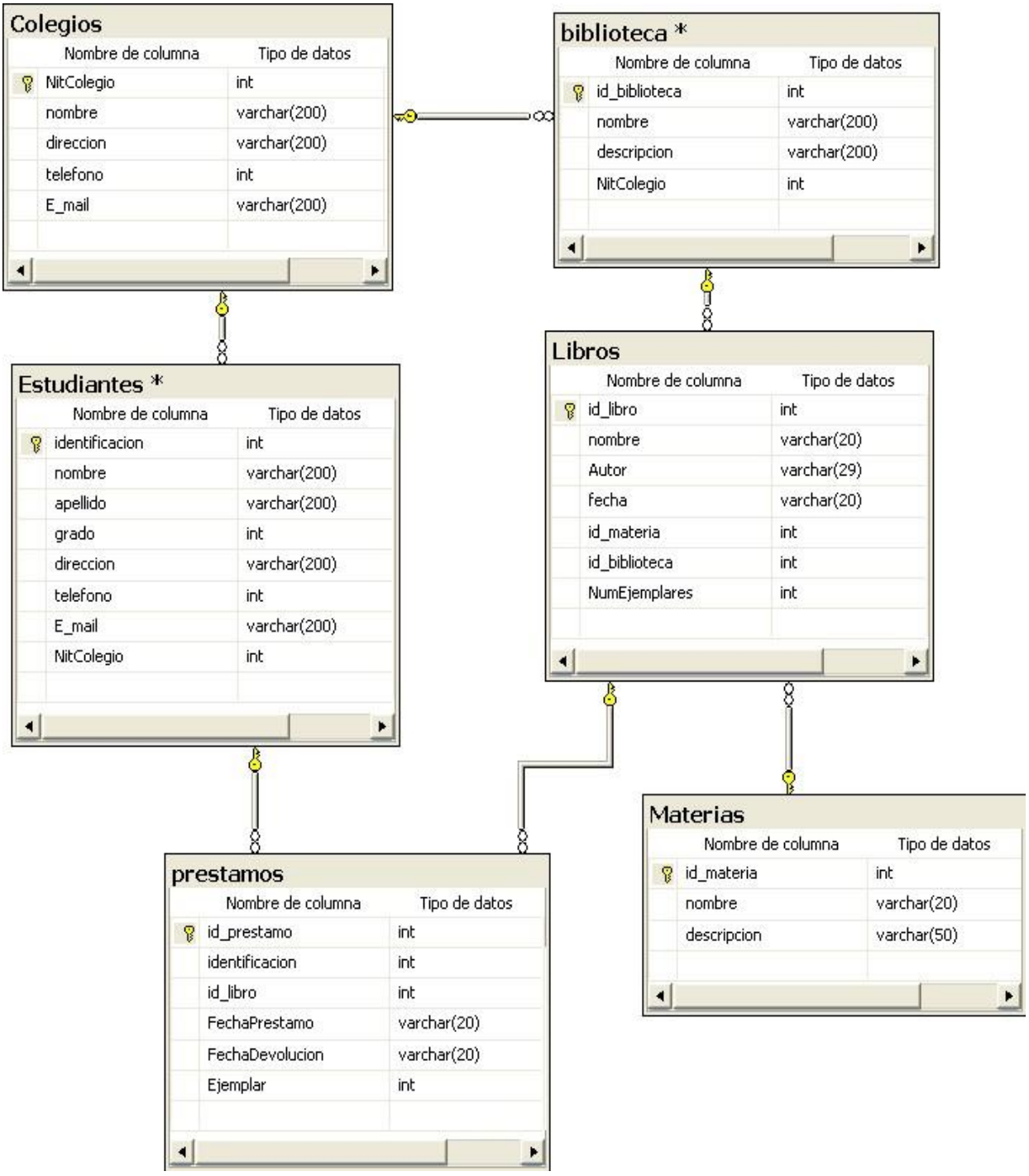
Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, INSERT INTO, DELETE, o UPDATE o dentro de otra subconsulta. Esta consulta anidada es algo así como una consulta dentro de otra consulta y se suele añadir en la cláusula WHERE de la instrucción SQL utilizando paréntesis. Un ejemplo básico de una subconsulta seria:

***SELECT*** *campos* ***FROM*** *TABLA1* ***WHERE*** *columna1 = (****SELECT*** *columna1* ***FROM*** *TABLA2);*

## EJERCICIO DE CONSULTAS AVANZADAS EN SQL SERVER

A continuación encontrará ejemplos de cómo realizar consultas avanzadas haciendo uso de algunas palabras claves mostradas en esta guía.

1. Implemente en Sql Server la base de datos del siguiente Modelo Relacional:




2. Ingrese información en las diferentes tablas de tal forma que las siguientes consultas arrojen varios resultados (minimo 3 registros)

**Consulta N°1:** Consulta de información desde dos tablas (Estudiantes y Colegios) usando producto cartesiano donde los estudiantes sean del grado 1101 y estudien en el colegio 'Colegio Enrique Olaya Herrera'.

---

```
Select
Estudiantes.identificacion,
```

```
Estudiantes.Nombre,
Estudiantes.Apellido,
Estudiantes.grado,
Colegios.Nombre
FROM
Estudiantes,Colegios

Where Estudiantes.NitColegio = Colegios.NitColegio
AND Estudiantes.Grado = 1101
AND Colegios.Nombre = 'Colegio Enrique Olaya Herrera'
```

---

***Nota:*** Notese que en un producto cartesiano para consultar dos tablas al mismo tiempo se debe colocar la combinación de llave Primaria/Foranea que estable la relación entre las tablas. En este caso es el campo NitColegio y se debe agregar despues del Where: “Estudiantes.NitColegio = Colegios.NitColegio”

**Consulta N°2:** Consulta de información desde dos tablas (Estudiantes y Colegios) usando producto cartesiano en la que se cuenta el número de estudiantes donde los estudiantes sean de grado 1101 y estudien en el colegio 'Colegio Enrique Olaya Herrera'.

---

```
Select
Count(*) As Numero_De_Estudiantes
FROM
Estudiantes,Colegios
Where Estudiantes.NitColegio = Colegios.NitColegio
AND Estudiantes.Grado = 1101
AND Colegios.Nombre = 'Colegio Enrique Olaya Herrera'
```

---

**Consulta N°3:** Consulta de información desde dos tablas (Estudiantes y Colegios) usando “INNER JOIN” donde el Apellido del Estudiante inicie con la letra “A”

---

```
Select
Estudiantes.identificacion,
Estudiantes.Nombre,
Estudiantes.Apellido,
Estudiantes.grado,
Colegios.Nombre
FROM
Estudiantes
INNER JOIN Colegios ON Estudiantes.NitColegio = Colegios.NitColegio
WHERE Estudiantes.Apellido Like 'A%'
```

---

**Consulta N°4:** Consulta de información desde Tres tablas (Libros, Materias y Biblioteca) usando "INNER JOIN"

```
Select
*
FROM
Libros
INNER JOIN Materias ON Libros.Id_materia = Materias.Id_Materia
INNER JOIN Biblioteca ON Libros.Id_Biblioteca = Biblioteca.Id_Biblioteca
```

**Consulta N°5:** Consulta de información desde Tres tablas (Libros, Materias y Biblioteca) usando "INNER JOIN", llamando columnas específicas y asignándoles un alias por medio de la palabra AS

```
Select
Libros.nombre As Nombre_Libro,
Libros.autor,
Libros.NumEjemplares,
Materias.nombre As Nombre_Materia,
Biblioteca.nombre AS Nombre_Biblioteca
FROM
Libros
INNER JOIN Materias ON Libros.Id_materia = Materias.Id_Materia
INNER JOIN Biblioteca ON Libros.Id_Biblioteca = Biblioteca.Id_Biblioteca
```

**Consulta N°6:** Consulta de información desde tres tablas (Libros, Materias y Biblioteca) usando "INNER JOIN", realizando diferentes Operaciones Aritméticas por medio de las funciones Count, Max, Min, Sum y Avg y asignándoles un alias por medio de la palabra AS

```
Select
Count(*) As Cantidad_Libros,
Max(Libros.Numejemplares) AS Mayor_Numero_Ejemplares,
Min(Libros.Numejemplares) AS Menor_Numero_Ejemplares,
Sum(Libros.Numejemplares) AS Total_Ejemplares, Avg(Libros.Numejemplares)
AS Promedio_Ejemplares
FROM
Libros
INNER JOIN Materias ON Libros.Id_materia = Materias.Id_Materia
INNER JOIN Biblioteca ON Libros.Id_Biblioteca = Biblioteca.Id_Biblioteca
```

**INFORMACIÓN FINAL:**

**COMO DIGITAR CÓDIGO DE CONSULTAS AVANZADAS EN LENGUAJE C#**

Un ejemplo de cómo escribir una consulta avanzada dentro de un Método de una “Clase” en Lenguaje C# seria:

```
string cadenaSQLConsultar =
   "Select " +
   "Estudiantes.identificacion, " +
   "Estudiantes.Nombre, " +
   "Estudiantes.Apellido, " +
   "Estudiantes.grado, " +
   "Colegios.Nombre " +
   "FROM Estudiantes " +
   "INNER JOIN Colegios ON Estudiantes.NitColegio = Colegios.NitColegio”
```