Pregrado

Programa de Ingeniería de Sistemas

GESTIÓN DE DATOS E INFORMACIÓN II

Sesión 2

Tema:

Consultas avanzadas, subconsultas, funciones definidas por el usuario.

Resultado de aprendizaje

Implementa y optimiza sentencias SQL usando consultas avanzadas, funciones, vistas y estructura de índices en la construcción de una base de datos para una organización.

Evidencia de aprendizaje

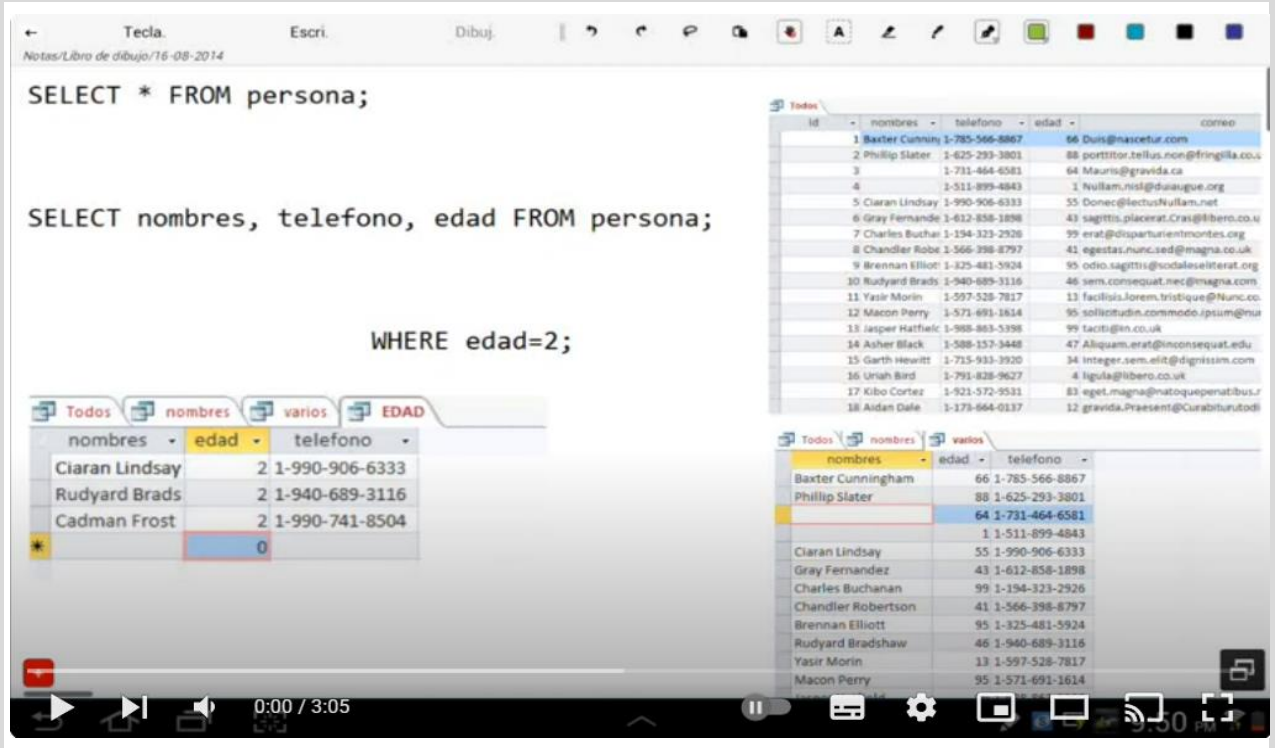
Los estudiantes demostrarán su comprensión y habilidades avanzadas en el diseño y ejecución de consultas en bases de datos

Contenido

Nombre del tema

- Consultas avanzadas
- Subconsultas
- Funciones definidas por el usuario

Revisa el siguiente video:



https://youtu.be/SeJDdwqZKnA?si=OAcY55-SA69vbOBU

Después de haber visualizado el video en la slide anterior, reflexionamos y respondemos las siguientes interrogantes:

01 ¿Para que sirven las sentencias **select**?

02 ¿Para que sirven las sentencias **from**?

03 ¿Para que sirven las sentencias **where**?

Tema

Consultas avanzadas

OVER

La sentencia **OVER** permite definir particiones y ordenamientos específicos dentro del conjunto de datos, lo que brinda un control granular sobre cómo se deben realizar los cálculos de agregación y cómo se deben asignar los resultados a cada fila. Esto es particularmente útil en situaciones en las que se necesita realizar cálculos basados en subconjuntos específicos de datos o cuando se quiere asignar rangos a filas basándose en ciertos criterios.

Clausula OVER y Row_Number

```
SELECT Row_Number() over (Order By Descripcion) As Item,Descripcion, Stockac,
Producto, Marca, Peso
FROM producto
WHERE Marca = 'M1'
```

Item	Descripcion	Stock	Producto	Marca	Peso
1	aceites	100	PR02	M1	1
2	FRIOL 1 LT	2764	PR01	M1	1
3	FRIOL 1/2 lt	2867	PR08	M1	0.5
4	FRIOL 5 LT	2139	PR13	M1	5

Clausula OVER , Row_Number() y Rank() Over

```
SELECT Row_Number() over (Order By Descripcion) As Item,Descripcion, Stockac,
Producto, Marca, Peso
FROM producto
WHERE Marca = 'M1'
```

Item	Descripcion	Stock	Producto	Marca	Peso
1	aceites	100	PR02	M1	1
2	FRIOL 1 LT	2764	PR01	M1	1
3	FRIOL 1/2 lt	2867	PR08	M1	0.5
4	FRIOL 5 LT	2139	PR13	M1	5

Clausula Dense_Rank() ,Rank() y Row_Number()

```
SELECT Row_Number() over (Order By Zona) As Item, Nombre, cliente, Zona, Saldo, Rank()
Over (Order By Zona) as Ranking,
Dense_Rank() OVER (ORDER BY Zona) AS DenseRank
FROM Cliente
WHERE Saldo BETWEEN 100 AND 5000
```

Item	Descripcion	Stock	Producto	Marca	Ranking	R_Densid
1	COMERCIAL PEPITA	CLI1	Z1	2500	1	1
2	SAN ANTONIO COM	CLI3	Z1	500	1	1
3	COQUITOS SAC	CLI4	Z2	2092	3	2
4	COMER SAN LORENZO	CLI2	Z2	4855	3	2
5	PREMIUN.NET	CLXY	Z3	1340	5	3

Uso de Common Table Expresions

```
with Tablita(zona, cliente, saldo) as
(
select zona, cliente, saldo
from cliente  )
select * from Tablita
```

Zona	Cliente	Saldo
Z2	CL10	0
Z1	CLI1	2500
Z2	CLI2	4855
Z1	CLI3	500
Z2	CLI4	2092
Z1	CLI5	0
Z1	CLI7	0
Z3	CLXY	1340

Uso de Pivote con tablas temporales

```
select personal, [z1], [z2]   from #datos
pivot (sum(pagado) FOR zona in ([Z1], [Z2])) as pvt
```

```
select personal, [z1], [z2]   from #datos
pivot (sum(pagado) FOR zona in ([Z1], [Z2])) as pvt
```

Personal	Z1	Z2
P1	21417	9275
P2	404	7534
P3	3930	1726
P4	1100	9860
P5	NULL	15603
P6	NULL	4438
ZZ	NULL	567

Uso de Pivote con CTE

```
WITH Tablita (Anual, Trimestral, Cobrado) as
( SELECT datepart(yy, fecha) , datepart(QQ, FECHA), pagado
  FROM documento)

SELECT * FROM Tablita
PIVOT (SUM(Cobrado) FOR Trimestral in ([1], [2],[3], [4])) as pvt
```

Anual	1	2	3	4
2000	5	1935	908	904
2002	338	2581	2	2439
2003	33259	6401	20652	6432

Uso de Pivote con CTE

```
WITH Tablita (Anual, Trimestral, Cobrado) as
( SELECT datepart(yy, fecha) , datepart(QQ, FECHA), pagado
  FROM documento)

SELECT * FROM Tablita
PIVOT (SUM(Cobrado) FOR Trimestral in ([1], [2],[3], [4])) as pvt
```

Anual	1	2	3	4
2000	5	1935	908	904
2002	338	2581	2	2439
2003	33259	6401	20652	6432

Tema

Subconsultas

Subconsultas

Una subconsulta en SQL es una consulta que se incorpora dentro de otra consulta principal. La subconsulta se utiliza dentro de una cláusula **WHERE**, **HAVING**, **FROM** o **SELECT** de la consulta principal para recuperar datos que se utilizarán en la evaluación de la condición o para proporcionar valores a ser seleccionados.

Las subconsultas pueden aparecer en diversas partes de una consulta SQL y se utilizan para realizar tareas como filtrar datos, realizar cálculos, o proporcionar valores para comparaciones.

Hay dos tipos principales de subconsultas: subconsultas correlacionadas y subconsultas no correlacionadas.

Subconsulta NO correlacionada

```
SELECT Nombre
FROM Empleados
WHERE DepartamentoID IN (SELECT DepartamentoID
                         FROM Departamentos
                         WHERE Ubicacion = 'Oficina Central');
```

En este **ejemplo**, la subconsulta no correlacionada **(SELECT DepartamentoID FROM Departamentos WHERE Ubicacion = 'Oficina Central')** se ejecuta una vez y luego se utiliza en la condición de la cláusula **WHERE** de la consulta principal para recuperar los nombres de los empleados que pertenecen a los departamentos ubicados en la 'Oficina Central'.

Subconsulta correlacionada

```
SELECT Nombre
FROM Productos
WHERE Precio > (SELECT AVG(Precio)
                FROM Productos
                WHERE Categoria = Productos.Categoria);
```

En este caso, la subconsulta correlacionada **(SELECT AVG(Precio) FROM Productos WHERE Categoria = Productos.Categoria)** se ejecuta para cada fila de la consulta principal, ya que depende de la columna Categoria de la tabla Productos. La subconsulta calcula el precio promedio para cada categoría y luego la consulta principal selecciona los productos cuyo precio es mayor que el promedio de su propia categoría.

Las subconsultas son una herramienta poderosa en SQL y permiten realizar consultas más complejas y flexibles al combinar datos de múltiples fuentes.

Tema

Funciones definidas por el usuario

Funciones escalares

Una función es una rutina almacenada que recibe unos parámetros escalares de entrada, los procesa según la definición de la función y finalmente retorna un resultado de un tipo específico que permitirá su utilización con un objetivo.

```
Create Function NombreFunción (@Parámetro1 tipo, @Parámetro 2 tipo...)
Returns Tipo As
Begin
    Sentencias
End
```

Funciones escalares

Convierte mayúsculas los campos de una tabla

```
CREATE FUNCTION EnMayusculas
(
@Nombre Varchar(50),
@Apellido Varchar(50)
)
RETURNS Varchar(100) AS
--Declarar Variables
BEGIN
 RETURN (UPPER(@Apellido) + ', ' + UPPER(@Nombre))
END
--Ejecutar
Print dbo.EnMayusculas('Geynen','Montenegro’)
go
--llamada a la función
Select dbo.EnMayusculas(Nombre,Apellidos) As Nombre
From Cliente
```

Funciones escalares

Ingresar el número del día de la semana y devuelve el nombre del día

```
Create function NombreDia(@Dia int) Returns Varchar(10) AS
Begin
        Declare @Var Varchar(10)
        Select @Var= Case @Dia
         When 1 Then 'Lunes'
         When 2 Then 'Martes'
         When 3 Then 'Miercoles'
         When 4 Then 'Jueves'
         When 5 Then 'Viernes'
         When 6 Then 'Sabado'
         When 7 Then 'Domingo'
        End
        Return @Var
End
--Ejecutar
Print dbo.NombreDia(5)
```

Funciones de tabla

Permiten retornar tablas en los resultados. Esta característica proporciona al programador facilidad a la hora de administrar sus bases de datos.

```
Create Function NombreFunción(Parámetro1 tipo, Parámetro2 tipo...)
Returns Table As
Return(Sentencias)
/
Sentencias
Return
```

Funciones de tabla

Ingresa el id del empleado y devuelve una tabla con los campos id, nombre ,apellidos y el número de pedidos realizados

```
Create Function Tabla(@IdEmpleado int)
Returns Table
AS
 Return (Select P.IdEmpleado, Nombre ,Apellidos, Count(P.IdEmpleado) as CantidadPedidos
 From Empleados E inner join Pedidos P On P.IdEmpleado=E.IdEmpleado
 Where E.IdEmpleado=@IdEmpleado
 Group By P.IdEmpleado, Nombre ,Apellidos)

--Ejecutar
Select * From dbo.Tabla(1)
```

Autoevaluación

Sesión 2

Pregunta 1

¿Cuál de las siguientes consultas utiliza correctamente la cláusula OVER para calcular la suma acumulativa de las ventas por producto, ordenadas por mes?

- SELECT Mes, Producto, SUM(Ventas) OVER (PARTITION BY Mes ORDER BY Producto) AS SumaAcumulativa FROM Ventas;
- SELECT Mes, Producto, SUM(Ventas) OVER (PARTITION BY Producto ORDER BY Mes) AS SumaAcumulativa FROM Ventas;
- SELECT Mes, Producto, SUM(Ventas) OVER (ORDER BY Mes, Producto) AS SumaAcumulativa FROM Ventas;
- SELECT Mes, Producto, SUM(Ventas) OVER (PARTITION BY Mes, Producto) AS SumaAcumulativa FROM Ventas;

Pregunta 2

¿Cuál de las siguientes afirmaciones describe correctamente una subconsulta?

- Una subconsulta es una consulta que se ejecuta automáticamente cuando se inicia la base de datos.
- Una subconsulta es una consulta anidada dentro de otra consulta principal.
- Una subconsulta es una consulta que solo puede contener una condición de igualdad.
- Una subconsulta es una consulta que siempre devuelve un único resultado.

Pregunta 3

En el contexto de SQL, ¿cuál es la principal ventaja de utilizar funciones definidas por el usuario?

- Facilitan la ejecución de consultas básicas sin necesidad de escribir código adicional.
- Mejoran la concurrencia de las transacciones en la base de datos.
- Proporcionan una forma estándar de realizar operaciones de agregación.
- Permiten a los usuarios personalizar y reutilizar lógica de manipulación de datos.

Autoevaluación

¡Vamos por más logros!

¡Felicitaciones!

Ha concluido la autoevaluación

Conclusiones

OVER en SQL es que proporciona una funcionalidad poderosa y flexible para realizar operaciones de análisis y agregación en conjuntos de datos. Al utilizar **OVER** junto con funciones de ventana, como **SUM(), AVG(), ROW_NUMBER(),** entre otras, se puede realizar un análisis detallado y avanzado de los datos sin la necesidad de realizar subconsultas complicadas o unirse a tablas adicionales.

El uso de **subconsultas** en SQL es que proporcionan una forma eficaz y flexible de realizar consultas complejas y anidadas, permitiendo a los desarrolladores y administradores de bases de datos acceder a información específica y realizar operaciones más avanzadas.

Las **funciones definidas por el usuario** ofrecen la capacidad de realizar operaciones complejas y personalizadas que van más allá de las funciones incorporadas de SQL, permitiendo a los usuarios crear funciones específicas para sus necesidades empresariales. Estas funciones pueden aceptar parámetros, devolver resultados y ser utilizadas en diversas partes del código SQL, lo que promueve la consistencia y la eficiencia en el desarrollo.

Aplicando lo aprendido:

Desarrollar la Guía de Laboratorio N°2

Referencias

CAPACHO, José y Wilson NIETO. Diseño de Bases de Datos [en línea]. Barranquilla: Universidad del Norte, 2017. ISBN 9789587418255. Disponible en: https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1690049&lang=es&site=ehost-live

WANUMEN Luis, RIVAS Edwin, Mosquera Darín. Bases de datos en SQL Server [en línea]. Bogotá: Ecoe Ediciones, 2017. ISBN 9789587715705. Disponible en: https://www.digitaliapublishing.com/a/66605

HUESO Luis. Bases de datos [en línea]. Madrid: Rama Editorial, 2014. ISBN 9788499641577.
Disponible en: https://www.digitaliapublishing.com/a/109943

Pregrado