IV. CLAUSULAS JOIN

CONSULTAS DE UNIÓN INTERNAS

Consultas de Combinación entre tablas

Las vinculaciones entre tablas se realizan mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común. Su sintaxis es:

- **SELECT** campos
  FROM tb1

  INNER JOIN tb2 **ON** tb1.campo1 comp tb2.campo2

En donde:

tb1, tb2	Son los nombres de las tablas desde las que se combinan los registros.
campo1, campo2	Son los nombres de los campos que se combinan. Si no son numéricos, los campos deben ser del mismo tipo de datos y contener el mismo tipo de datos, pero no tienen que tener el mismo nombre.
comp	Es cualquier operador de comparación relacional: =, <,<>, <=, =>, ó >.

Se puede utilizar una operación INNER JOIN en cualquier cláusula FROM. Esto crea una combinación por equivalencia, conocida también como unión interna. Las combinaciones equivalentes son las más comunes; éstas combinan los registros de dos tablas siempre que haya concordancia de valores en un campo común a ambas tablas. Se puede utilizar INNER JOIN con las tablas Departamentos y Empleados para seleccionar todos los empleados de cada departamento. Por el contrario, para seleccionar todos los departamentos (incluso si alguno de ellos no tiene ningún empleado asignado) se emplea LEFT JOIN o todos los empleados (incluso si alguno no está asignado a ningún departamento), en este caso RIGHT JOIN.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

También se pueden enlazar varias cláusulas ON en una instrucción JOIN, utilizando la sintaxis siguiente:

- **SELECT** campos **FROM** tabla1
  INNER JOIN tabla2 **ON** (tb1.campo1 comp tb2.campo1

  AND tb1.campo2 comp tb2.campo2)

  OR tb1.campo3 comp tb2.campo3

Si empleamos la cláusula INNER en la consulta se seleccionarán sólo aquellos registros de la tabla de la que hayamos escrito a la izquierda de INNER JOIN que contengan al menos un registro de la tabla que hayamos escrito a la derecha. Para solucionar esto tenemos dos cláusulas que sustituyen a la palabra clave INNER, estas cláusulas son LEFT y RIGHT. LEFT toma todos los registros de la tabla ubicada en FROM aunque no tengan ningún registro en la tabla del JOIN. RIGHT realiza la misma operación pero al contrario, toma todos los registros de la tabla del JOIN aunque no tenga ningún registro en la tabla del FROM.

Consultas de Auto combinación

La auto combinación se utiliza para unir una tabla consigo misma, comparando valores de dos columnas con el mismo tipo de datos. La sintaxis en la siguiente:

- **SELECT** alias1.columna, alias2.columna, ...
  FROM tabla1 as alias1, tabla2 as alias2

  WHERE alias1.columna = alias2.columna

  AND otras condiciones

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Por ejemplo, para visualizar el número, nombre y puesto de cada empleado, junto con el número, nombre y puesto del supervisor de cada uno de ellos se utilizaría la siguiente sentencia:

- **SELECT** t.num_emp, t.nombre, t.puesto, t.num_sup,s.nombre, s.puesto
  FROM empleados AS t, empleados AS s

  WHERE t.num_sup = s.num_emp

CONSULTAS DE UNIÓN EXTERNAS

Se utiliza la operación UNION para crear una consulta de unión, combinando los resultados de dos o más consultas o tablas independientes. Su sintaxis es:

- consulta1 UNION consulta2 UNION consulta N

En donde:

Consulta1, consulta2, consulta n	Son instrucciones SELECT

Puede combinar los resultados de dos o más consultas, instrucciones SELECT, en cualquier orden, en una única operación UNION. Todas las consultas en una operación UNION deben pedir el mismo número de campos, no obstante los campos no tienen porqué tener el mismo tamaño o el mismo tipo de datos.

Se puede utilizar una cláusula GROUP BY y/o HAVING en cada argumento consulta para agrupar los datos devueltos. Puede utilizar una cláusula ORDER BY al final del último argumento consulta para visualizar los datos devueltos en un orden específico.

CRITERIOS DE SELECCIÓN

Antes de comenzar el desarrollo de este apartado hay que recalcar tres detalles de vital importancia. El primero de ellos es que cada vez que se desee establecer una condición referida a un campo de texto la condición de búsqueda debe ir encerrada entre comillas simples; la segunda hace referencia a las fechas se hace necesario particularizarlas según el banco de datos.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Referente a los valores lógicos True o False; en estos sistemas se utilizan los campos BIT que permiten almacenar valores de 0 ó 1, 0 para los valores FALSE, y 1 para los valores TRUE, se puede utilizar la sintaxis siguiente que funciona en todos los casos: si se desea saber si el campo es falso "... CAMPO = 0" y para saber los verdaderos "CAMPO <> 0" o “CAMPO = 1”.

Operadores Lógicos

Los operadores lógicos soportados por SQL son: AND, OR, NOT. A excepción del último todos poseen la siguiente sintaxis:

- <expresión1> **operador** <expresión2>

En donde expresión1 y expresión2 son las condiciones a evaluar, el resultado de la operación varía en función del operador lógico. La tabla adjunta muestra los diferentes posibles resultados:

<expresión1>	Operador	<expresión2>	Resultado
Verdad	AND	Falso	Falso
Verdad	AND	Verdad	Verdad
Falso	AND	Verdad	Falso
Falso	AND	Falso	Falso
Verdad	OR	Falso	Verdad
Verdad	OR	Verdad	Verdad
Falso	OR	Verdad	Verdad
Falso	OR	Falso	Falso

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Si a cualquiera de las anteriores condiciones le anteponemos el operador NOT el resultado de la operación será el contrario al devuelto sin el operador NOT.

Ejemplos:

- **SELECT * FROM** Empleados
  WHERE Edad > 25 AND Edad < 50

- **SELECT * FROM** Empleados
  WHERE (Edad > 25 AND Edad < 50)

  OR Sueldo = 100

- **SELECT * FROM** Empleados
  WHERE NOT Estado = 'Soltero'

- **SELECT * FROM** Empleados
  WHERE (Sueldo > 100 AND Sueldo < 500)

  OR (Municipio = 'Soyapango' AND Estado = 'Casado')

Valores Nulos

En muchas ocasiones es necesario emplear como criterio de selección valores nulos en los campos. Podemos emplear el operador IS NULL para realizar esta operación. Por ejemplo:

- **SELECT * FROM** Empleados
  WHERE DUI IS NULL

Intervalos de Valores

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between cuya sintaxis es:

- campo [Not] Between valor1 And valor2 (la condición Not es opcional)

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.

- SELECT * FROM Empleados
  WHERE Codigo Between 1154 And 1524

El Operador Like

Se utiliza para comparar un campo con un modelo en una expresión SQL. Su sintaxis es:

- campo Like modelo

En donde campo se compara con el modelo expresión. Se puede utilizar el operador Like para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana María), o se puede utilizar una cadena de caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (An%,).

El operador Like se puede utilizar en una expresión para comparar un valor de un campo con una expresión de cadena. Por ejemplo, si introduce Like 'C*' en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen por la letra C. En una consulta con parámetros, puede hacer que el usuario escriba el modelo que se va a utilizar.

La cláusula WHERE

La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Después de escribir esta cláusula se deben especificar las condiciones expuestas en los apartados anteriores. Si no se emplea esta cláusula, la consulta devolverá todas las filas de la tabla. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM o INNER, LEFT RIGHT JOIN.

- SELECT Apellidos, Salario FROM Empleados
  WHERE Salario = 21000

- SELECT IdProducto, Existencias FROM Productos
  WHERE Existencias <= NuevoPedido

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

- **SELECT** * **FROM** Pedidos
  WHERE FechaEnvio = ‘19943005’

- **SELECT** Apellidos, Nombre **FROM** Empleados
  WHERE Apellidos = 'King'

- **SELECT** Apellidos, Nombre **FROM** Empleados
  WHERE Apellidos Like 'S%'

- **SELECT** Apellidos, Salario **FROM** Empleados
  WHERE Salario Between 200 And 300

- **SELECT** Apellidos, Salario **FROM** Empleados
  WHERE Apellidos Between 'Lon' And 'Tol'

- **SELECT** IdPedido, FechaPedido **FROM** Pedidos
  WHERE FechaPedido Between ‘19940101’ And ‘19941231’

- **SELECT** Apellidos, Nombre, Ciudad **FROM** Empleados
  WHERE Ciudad In ('Sevilla', 'Los Angeles', 'Barcelona')

AGRUPAMIENTO DE REGISTROS

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

GROUP BY

Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT. Su sintaxis es:

- **SELECT** campos **FROM** tabla
  WHERE criterio

  GROUP BY campos del grupo

GROUP BY es opcional. Los valores de resumen se omiten si no existe una función SQL agregada en la instrucción SELECT. Los valores Null en los campos GROUP BY se agrupan y no se omiten. No obstante, los valores Null no se evalúan en ninguna de las funciones SQL agregadas.

Se utiliza la cláusula WHERE para excluir aquellas filas que no desea agrupar, y la cláusula HAVING para filtrar los registros una vez agrupados.

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función SQL agregada.

- SELECT IdFamilia, Sum(Stock) AS StockActual FROM Productos
  GROUP BY IdFamilia

Una vez que GROUP BY ha combinado los registros, HAVING muestra cualquier registro agrupado por la cláusula GROUP BY que satisfaga las condiciones de la cláusula HAVING.

HAVING es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuales de ellos se van a mostrar.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

- **SELECT** IdFamilia, Sum(Stock) AS StockActual **FROM** Productos
  GROUP BY IdFamilia

  HAVING StockActual > 100 AND NombreProducto Like 'BOS%'

AVG

Calcula el promedio de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es la siguiente

- Avg (expr)

En donde expr representa el campo que contiene los datos numéricos para los que se desea calcular el promedio o una expresión que realiza un cálculo utilizando los datos de dicho campo. El promedio para Avg se calcula asi: la suma de los valores dividido por el número de valores). La función Avg no incluye ningún campo Null en el cálculo.

- **SELECT** Avg(Gastos) AS Promedio **FROM** Pedidos
  WHERE Gastos > 100

Count

Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente

- Count(expr)

En donde expr contiene el nombre del campo que desea contar. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL). Puede contar cualquier tipo de datos incluso texto.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Aunque expr puede realizar un cálculo sobre un campo, Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros. La función Count no cuenta los registros que tienen campos null a menos que expr sea el carácter comodín asterisco (*). Si utiliza un asterisco, Count calcula el número total de registros, incluyendo aquellos que contienen campos null. Count(*) es considerablemente más rápida que Count(Campo). No se debe poner el asterisco entre dobles comillas ('*').

- SELECT Count(*) AS Total FROM Pedidos

Max, Min

Devuelven el máximo o el mínimo de un conjunto de valores contenidos en un campo especifico de una consulta. Su sintaxis es:

- Min(expr)
- Max(expr)

En donde expr es el campo sobre el que se desea realizar el cálculo. Expr puede incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

- SELECT Min(Gastos) AS ElMin FROM Pedidos
  WHERE Pais = 'España'

- SELECT Max(Gastos) AS ElMax FROM Pedidos
  WHERE Pais = 'España'

Sum

Devuelve la suma del conjunto de valores contenido en un campo específico de una consulta. Su sintaxis es:

- Sum(expr)

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

En donde expr representa el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

- `SELECT Sum(PrecioUnidad * Cantidad) AS Total FROM DetallePedido`

SUBCONSULTAS

Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE o dentro de otra subconsulta. Puede utilizar dos formas de sintaxis para crear una subconsulta:

expresión [NOT] IN (instrucción sql)

[NOT] EXISTS (instrucción sql)

En donde:

Expression	Es una expresión por la que se busca el conjunto resultante de la subconsulta.
instrucción SQL	Es una instrucción SELECT, que sigue el mismo formato y reglas que cualquier otra instrucción SELECT. Debe ir entre paréntesis.

Se puede utilizar una subconsulta en lugar de una expresión en la lista de campos de una instrucción SELECT o en una cláusula WHERE o HAVING. En una subconsulta, se utiliza una instrucción SELECT para proporcionar un conjunto de uno o más valores especificados para evaluar en la expresión de la cláusula WHERE o HAVING.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

El predicado IN se emplea para recuperar únicamente aquellos registros de la consulta principal para los que algunos registros de la subconsulta contienen un valor igual. El ejemplo siguiente devuelve todos los productos vendidos con un descuento igual o mayor al 25 por ciento:

- **SELECT * FROM** Productos
  WHERE IDProducto IN

  (SELECT Improducto FROM DetallePedido

  WHERE Descuento = 0.25)

Inversamente se puede utilizar NOT IN para recuperar únicamente aquellos registros de la consulta principal para los que no hay ningún registro de la subconsulta que contenga un valor igual.

El predicado EXISTS (con la palabra reservada NOT opcional) se utiliza en comparaciones de verdad/falso para determinar si la subconsulta devuelve algún registro. Supongamos que deseamos recuperar todos aquellos clientes que hayan realizado al menos un pedido:

- **SELECT** Clientes.Compañía, Clientes.Teléfono **FROM** Clientes
  WHERE EXISTS (**SELECT * FROM** Pedidos

  WHERE Pedidos.IdPedido = Clientes.IdCliente **)**

Esta consulta es equivalente a esta otra:

- **SELECT** Clientes.Compañía, Clientes.Teléfono **FROM** Clientes
  WHERE IdClientes IN (SELECT Pedidos.IdCliente FROM Pedidos)

Se puede utilizar también alias del nombre de la tabla en una subconsulta para referirse a tablas listadas en la cláusula FROM fuera de la subconsulta. El ejemplo siguiente devuelve los nombres de los empleados cuyo salario es igual o mayor que el salario medio de todos los empleados con el mismo título. A la tabla Empleados se le ha dado el alias T1:

- **SELECT** Apellido, Nombre, Titulo, Salario **FROM** Empleados AS T1

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

WHERE Salario = (SELECT Avg(Salario) FROM Empleados

WHERE T1.Titulo = Empleados.Titulo)

ORDER BY Titulo

En el ejemplo anterior, la palabra reservada AS es opcional.

- **SELECT** Apellidos, Nombre, Cargo, Salario **FROM** Empleados
  WHERE Cargo LIKE 'Agente Ven%'

  AND Salario > (SELECT Salario FROM Empleados

  WHERE Cargo LIKE '%Jefe%'

  OR Cargo LIKE '%Director%')

(Obtiene una lista con el nombre, cargo y salario de todos los agentes de ventas cuyo salario es mayor que el de todos los jefes y directores.)

- **SELECT DISTINCT** NombreProducto, Precio_Unidad **FROM** Productos
  WHERE PrecioUnidad = (**SELECT** PrecioUnidad **FROM** Productos

  WHERE NombreProducto = 'Almíbar anisado')

(Obtiene una lista con el nombre y el precio unitario de todos los productos con el mismo precio que el almíbar anisado.)

- **SELECT DISTINCT** NombreContacto, NombreCompania, CargoContacto, Telefono **FROM** Clientes
  WHERE IdCliente IN (SELECT DISTINCT IdCliente FROM Pedidos

  WHERE FechaPedido < ’19930701’)

(Obtiene una lista de las compañías y los contactos de todos los clientes que han realizado un pedido en el segundo trimestre de 1993.)

- **SELECT** Nombre, Apellidos **FROM** Empleados AS E

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

WHERE EXISTS (SELECT * FROM Pedidos AS O

WHERE O.IdEmpleado = E.IdEmpleado **)**

(Selecciona el nombre de todos los empleados que han reservado al menos un pedido.)

- **SELECT DISTINCT** Pedidos.Id_Producto, Pedidos.Cantidad,
  (SELECT Productos.Nombre FROM Productos

  WHERE Productos.IdProducto = Pedidos.IdProducto) AS ElProducto

  FROM Pedidos

  WHERE Pedidos.Cantidad = 150

  ORDER BY Pedidos.Id_Producto

(Recupera el Código del Producto y la Cantidad pedida de la tabla pedidos, extrayendo el nombre del producto de la tabla de productos.)

- **SELECT** NumVuelo, Plazas **FROM** Vuelos
  WHERE Origen = 'Madrid'

  AND Exists (SELECT T1.NumVuelo FROM Vuelos AS T1

  WHERE T1.PlazasLibres > 0 AND T1.NumVuelo=Vuelos.NumVuelo)

(Recupera números de vuelo y capacidades de aquellos vuelos con destino Madrid y plazas libres

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)