



Consultas multitaslas

En este capítulo veremos el diseño de consultas que leen más de una tabla: las consultas correlacionadas, combinaciones ó joins, y las subconsultas. Además, trataremos algunas consultas especiales como el uso de DISTINCT y UNION.

Esta página se ha dejado en blanco intencionalmente.

Capítulo 10

Consultas multitas

Contenido

- ❑ Consultas correlacionadas
- ❑ Inner join
 - ✓ **Ejercicio 88:** Uso de INNER JOIN
 - Uso de alias como referencia a tablas
 - ✓ **Ejercicio 89:** Catálogo de Productos
 - ✓ **Ejercicio 90:** Monto de la Guía de Remisión X
 - ✓ **Ejercicio 91:** Join de tres tablas
 - ✓ **Ejercicio 92:** Monto total enviado a cada local
 - ✓ **Ejercicio 93:** Unidades totales despachadas al mes del producto X
 - ✓ **Ejercicio 94:** Unidades mensuales despachadas de cada producto
- ❑ Outer join
 - ✓ **Ejercicio 95:** Uso de OUTER JOIN
 - ✓ **Ejercicio 96:** Reporte de unidades despachadas de cada producto
- ❑ Cross join
 - ✓ **Ejercicio 97:** Uso de CROSS JOIN
- ❑ El operador UNION
 - ✓ **Ejercicio 98:** Uso del operador UNION
- ❑ La instrucción SELECT...INTO
 - ✓ **Ejercicio 99:** Uso de SELECT...INTO
 - ✓ **Ejercicio 100:** Creación de una tabla temporal
- ❑ Consulta autojoin
 - ✓ **Ejercicio 101:** Consulta autojoin

- ❑ *Subconsultas*
 - ✓ *Subconsulta que entrega un solo valor (1 fila, 1 columna)*
 - ✓ **Ejercicio 102:** *Subconsulta definida en la lista de columnas del SELECT externo*
 - ✓ **Ejercicio 103:** *Porcentaje despachado de cada producto respecto al total despachado para la categoría X*
 - ✓ **Ejercicio 104:** *Subconsulta definida en el WHERE del SELECT externo*
 - ✓ *Subconsulta que entrega un conjunto de valores (varias filas, 1 columna)*
 - ✓ **Ejercicio 105:** *Test de pertenencia*
 - ✓ *Subconsulta correlacionada*
 - ✓ **Ejercicio 106:** *Test de existencia – Uso de EXISTS*
- ❑ *Inserción de filas con datos leídos por SELECT*
 - ✓ **Ejercicio 107:** *Inserción de filas con subconsulta*
- ❑ *El operador PIVOT*
 - ✓ **Ejercicio 108:** *Uso del operador PIVOT*
- ❑ *Common Table Expression (CTE)*
 - ✓ **Ejercicio 109:** *Uso de una Common Table Expression*
 - *Filtrando una CTE*
 - *Uso de agregación en una CTE*
 - ✓ **Ejercicio 110:** *Consulta recursiva en una CTE*

Consultas multitaslas

En la mayoría de los casos, la recuperación de los datos que los usuarios necesitan para trabajar implica la lectura de muchas tablas para que la información así obtenida sea de utilidad para ellos. En este capítulo veremos el diseño de las instrucciones SELECT que nos permiten recuperar datos de varias tablas en un solo conjunto de resultados.

Consultas correlacionadas

Un **join**, **combinación** ó **consulta correlacionada** es la consulta que selecciona columnas de dos tablas ó conjuntos de filas, y las entrega en un único conjunto de resultados. Las filas de las tablas ó conjuntos de filas se combinan relacionando valores comunes, típicamente valores de clave primaria y clave foránea.

Sintáxis general

```
SELECT lista_columnas  
FROM tabla1  
tipo_join JOIN tabla2 ON condición_del_join
```

- **lista_columnas** es la lista de columnas a mostrar en el resultado de la consulta. Se recomienda que cada columna sea calificada con el alias de la tabla a la cual pertenece.
- **tipo_join** indica si el join es interior (INNER), exterior (OUTER) ó irrestricto (CROSS).
- **condición_del_join** es una expresión que indica en base a qué columnas de cada una de las tablas se establece la relación entre ellas.

Una combinación (join) puede ser de cualquiera de los siguientes tipos:

- inner join
- outer join
 - left outer join
 - right outer join
 - full outer join
- cross join

Antes de explicar cada uno de los tipos de combinaciones, veremos algunos detalles de la base de datos **MarketPERU** que nos permitirán entender cómo se ejecutan cada uno de los tipos de join.

1. Ejecute las siguientes instrucciones para ver dichos detalles:

```
SELECT COUNT(*) FROM Producto
go
```

El resultado arroja 138. Esto quiere decir que la base de datos tiene registrados los datos (idProducto, nombre, etc.) de **138 productos** (valores de idProducto del 1 al 138).

2. Los productos, cuando salen del almacén, lo hacen con el documento Guía de Remisión. Ejecute la consulta que le permite determinar cuántos de los 138 productos, registran por lo menos una salida del almacén.

```
SELECT idProducto FROM Guia_detalle
go
```

El resultado tiene 1177 filas, lo que significa que la base de datos registra 1177 detalles de guía. Muchos productos aparecen más de una vez en el listado, lo que indica que ellos tienen registrada más de una salida.

3. Ahora, elimine las filas duplicadas del resultado de la consulta para determinar cuántos productos registran salida del almacén.

```
SELECT DISTINCT idProducto
FROM Guia_detalle
ORDER BY idProducto
go
```

Observe que el resultado ahora tiene solo 66 filas. Esto indica que hay **66 productos que registran salida del almacén**. Por lo tanto, si tenemos **138 productos registrados**, entonces $138 - 66 = 72$ **productos que no registran salida del almacén**.

4. También pudo obtener la cuenta de productos que no registran salida del almacén ejecutando la siguiente consulta:

```
SELECT COUNT(DISTINCT idProducto)
FROM Guia_detalle
go
```

Tenga presente estos datos durante la explicación de cada uno de los tipos de combinaciones ó joins.

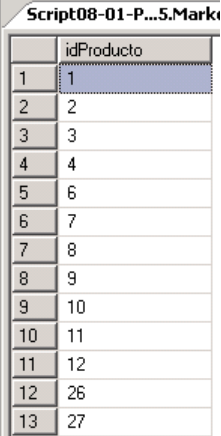
Inner join

Un **inner join** es la consulta correlacionada que combina todas las filas que están relacionadas de las dos tablas ó conjuntos de filas.

Ejercicio 88: Uso de inner join

Se desea obtener una lista de los productos que registran salida del almacén. Los productos que registran salida del almacén son aquellos cuyo **idProducto** figura en la tabla **Guia_detalle**.

```
SELECT DISTINCT idProducto
FROM Guia_detalle
ORDER BY idProducto
go
```



	idProducto
1	1
2	2
3	3
4	4
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	26
13	27

El resultado de la consulta muestra los **idProducto** requeridos, pero para un usuario sería muy difícil identificar al producto usando solo su **idProducto**.

Modifique la consulta para que también muestre el nombre del producto. El nombre del producto figura en la tabla **Producto**, por lo que será necesario leer también esta tabla para satisfacer este requerimiento. La consulta debe encontrar todos los **idProducto** que se encuentran en la tabla **Producto**, y que también se encuentran en la tabla **Guia_detalle**. Por lo tanto, el tipo de join tiene que ser INNER JOIN.


```

SELECT Guia_detalle.idProducto, Producto.nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
ORDER BY Guia_detalle.idProducto
go

```

Script08-01-P...S.MarketPERU*			Summary
	idProducto	nombre	
1	1	CARAMELOS BASTON VIENA ARCOR	
2	1	CARAMELOS BASTON VIENA ARCOR	
3	1	CARAMELOS BASTON VIENA ARCOR	
4	1	CARAMELOS BASTON VIENA ARCOR	
5	1	CARAMELOS BASTON VIENA ARCOR	
6	1	CARAMELOS BASTON VIENA ARCOR	
7	1	CARAMELOS BASTON VIENA ARCOR	
8	1	CARAMELOS BASTON VIENA ARCOR	
9	1	CARAMELOS BASTON VIENA ARCOR	
10	1	CARAMELOS BASTON VIENA ARCOR	
11	1	CARAMELOS BASTON VIENA ARCOR	
12	1	CARAMELOS BASTON VIENA ARCOR	
13	1	CARAMELOS BASTON VIENA ARCOR	

Note que el producto **1** aparece varias veces. Esto indica que dicho producto aparece en varias Guías de Remisión, es decir, registra varias salidas del almacén.

Modifique la consulta para eliminar las filas duplicadas del resultado de la consulta.

```

SELECT DISTINCT Guia_detalle.idProducto, Producto.nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
ORDER BY Guia_detalle.idProducto
go

```

Script08-01-P...5.MarketPERU*			Summary
	idProducto	nombre	
1	1	CARAMELOS BASTON VIENA ARCOR	
2	2	CARAMELOS SURTIDO DE FRUTAS	
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	
4	4	CARAMELOS FRUTAS MASTICABLES	
5	6	FRUNA SURTIDA DONOFRIO	
6	7	CHOCOLATE DOÑA PEPA FIELD	
7	8	CHOCOLATE CUA CUA FIELD	
8	9	MELLOWS FAMILIAR FIELD	
9	10	WAFER CHOCOLATE FIELD	
10	11	CHOCOLATE BARRA REGULAR	
11	12	CHOCOLATE MOSTRO FIELD	
12	26	JAMONADA LAIVE	
13	27	JAMONADA ESPECIAL LA SEGOVIANA	

El resultado muestra 66 productos que registran salida del almacén.

Uso de alias como referencia a tablas

En la siguiente consulta

```
SELECT DISTINCT Guia_detalle.idProducto, Producto.nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
ORDER BY Guia_detalle.idProducto
go
```

cada columna a la que se hace referencia va calificada con el nombre de la tabla de la que se lee la columna.

Si bien calificar las columnas hace que la consulta sea más fácil de entender para el usuario, hacerlo no es obligatorio, salvo en los casos en los que no hacerlo puede conducir a ambigüedad en la lectura de la instrucción.

Por ejemplo,

```
SELECT DISTINCT idProducto, nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
ORDER BY idProducto
go
-- Error 209: nombre de columna ambiguo: idProducto
```

la instrucción anterior genera el error 209 debido a que la columna **idProducto** mencionada en SELECT existe tanto en la tabla **Guia_detalle** como en la tabla **Producto**, y no se sabe de cuál de las tablas hay que leerla. No ocurre lo mismo con la columna **nombre** ya que ésta solo existe en la tabla **Producto**.

En conclusión, si un nombre de columna al que se hace referencia existe en las dos tablas, entonces es obligatorio calificar la columna para que no se produzca el error 209.

Para evitar que se genere el error 209, podemos escribir la consulta así:

```
SELECT DISTINCT Guia_detalle.idProducto, nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
ORDER BY Guia_detalle.idProducto
go
```

También es posible simplificar la digitación de los nombres de tabla definiendo un **alias** para cada tabla. El alias es una cadena corta que podemos definir para hacer referencia a la tabla con un nombre corto.

```
SELECT DISTINCT gd.idProducto, p.nombre
FROM Guia_detalle gd INNER JOIN Producto p
    ON gd.idProducto = p.idProducto
ORDER BY gd.idProducto
go
```

Ya sea que utilice ó no los alias para tablas, la recomendación al escribir una consulta JOIN es que califique todas las columnas ya que así es más fácil de entender para el usuario que no ha escrito la consulta.

Ejercicio 89: Catálogo de Productos

Escriba una consulta que lista el Catálogo de Productos de la empresa.

```
SELECT Categoria.categoria, Producto.idProducto,
       Producto.nombre, Producto.unidadMedida,
       Producto.precioProveedor
FROM Producto INNER JOIN Categoria
  ON Producto.idCategoria = Categoria.idCategoria
ORDER BY Categoria.categoria, Producto.idProducto
go
```

Script08-02-U...05.MarketPERU*					
Summary					
	categoria	idProducto	nombre	unidadMedida	precioProveedor
1	EMBUTIDOS	26	JAMONADA LAIVE	KILOGRAMO	12.50
2	EMBUTIDOS	27	JAMONADA ESPECIAL LA SEGOVIANA	KILOGRAMO	10.50
3	EMBUTIDOS	28	JAMONADA POLACA OTTO KUNZ	KILOGRAMO	12.50
4	EMBUTIDOS	29	JAMONADA DE POLLO SAN FERNANDO	KILOGRAMO	10.00
5	EMBUTIDOS	30	JAMONADA ESPECIAL OTTO KUNZ	KILOGRAMO	17.00
6	EMBUTIDOS	31	JAMON INGLES SAN FERNANDO	KILOGRAMO	12.50
7	EMBUTIDOS	32	JAMON INGLES LAIVE	KILOGRAMO	20.50
8	EMBUTIDOS	33	JAMON LIGHT BRAEDT	KILOGRAMO	20.50
9	EMBUTIDOS	34	JAMON YORK BRAEDT	KILOGRAMO	22.50
10	EMBUTIDOS	35	JAMON INGLES LA SEGOVIANA	KILOGRAMO	11.50
11	EMBUTIDOS	36	JAMON YORK SALCHICHERIA ALEMANA	KILOGRAMO	21.50
12	EMBUTIDOS	37	HOT DOG LAIVE PELADO	KILOGRAMO	5.50
13	EMBUTIDOS	38	HOT DOG LA SEGOVIANA	KILOGRAMO	6.80
14	EMBUTIDOS	39	HOT DOG AMERICANO OTTO KUNZ	KILOGRAMO	7.50
15	EMBUTIDOS	40	HOT DOG CERDEÑA	KILOGRAMO	8.00
16	EMBUTIDOS	41	HOTDOG AMERICANO BRAEDT	KILOGRAMO	9.50
17	EMBUTIDOS	42	SALCHICHA DE HUACHO	KILOGRAMO	10.50
18	EMBUTIDOS	43	HOT DOG EXTRA SAN FERNANDO	KILOGRAMO	9.50
19	EMBUTIDOS	44	CHORIZO PARRILLERO LAIVE	KILOGRAMO	11.50

Ejercicio 90: Monto de la Guía de Remisión X

Escriba una consulta que muestre los datos de la cabecera de la guía de remisión número **27**, y además su monto total.

```
SELECT Guia.idGuia, Guia.idLocal, Guia.fechaSalida,
       Monto =
       SUM(Guia_detalle.precioVenta * Guia_detalle.cantidad)
FROM Guia INNER JOIN Guia_detalle
  ON Guia.idGuia = Guia_detalle.idGuia
GROUP BY Guia.idGuia, Guia.idLocal, Guia.fechaSalida
HAVING Guia.idGuia = 27
go
```

Script08-02-U...05.MarketPERU*				
				Summary
	idGuia	idLocal	fechaSalida	Monto
1	27	2	2005-01-23 11:38:54.357	10515.00

Ejercicio 91: Join de tres tablas

Modifique la consulta del ejercicio anterior para que muestre la dirección del local al que se enviaron los productos registrados en la guía de remisión número **27**.

```
SELECT Guia.idGuia, Local.direccion, Guia.fechaSalida,
       Monto = SUM(Guia_detalle.precioVenta *
Guia_detalle.cantidad)
FROM Guia INNER JOIN Guia_detalle
  ON Guia.idGuia = Guia_detalle.idGuia
INNER JOIN Local
  ON Guia.idLocal = Local.idLocal
GROUP BY Guia.idGuia, Local.direccion, Guia.fechaSalida
HAVING Guia.idGuia = 27
go
```

Script08-02-U...05.MarketPERU*				
				Summary
	idGuia	direccion	fechaSalida	Monto
1	27	AV. BOLIVAR 1789	2005-01-23 11:38:54.357	10515.00

Ejercicio 92: Monto total enviado a cada local

Escriba una consulta que muestre el monto total despachado a cada local.

```
SELECT Local.direccion, Monto =
    SUM(Guia_detalle.precioVenta * Guia_detalle.cantidad)
FROM Local INNER JOIN Guia
    ON Local.idLocal = Guia.idLocal
INNER JOIN Guia_detalle
    ON Guia.idGuia = Guia_detalle.idGuia
GROUP BY Local.direccion
go
```

	direccion	Monto
1	AV. BOLIVAR 1789	76363.75
2	AV. ESPAÑA 775	75122.50
3	AV. LA PAZ 659	76363.75
4	AV. SAENZ PEÑA 590	75122.50
5	PANAMERICANA NORTE KM. 17.5	75122.50

Ejercicio 93: Unidades totales despachadas al mes del producto X

Escriba una consulta que muestre el total de unidades despachadas por mes del producto **27**.

```
SELECT YEAR(Guia.fechaSalida) AS Año,
    MONTH(Guia.fechaSalida) AS Mes,
    SUM(Guia_detalle.cantidad) As Unidades
FROM Guia INNER JOIN Guia_detalle
    ON Guia.idGuia = Guia_detalle.idGuia
WHERE Guia_detalle.idProducto = 27
GROUP BY YEAR(Guia.fechaSalida), MONTH(Guia.fechaSalida)
ORDER BY Año, Mes
go
```

Script08-02-U...05.MarketPERU*				Summary
	Año	Mes	Unidades	
1	2005	1	225	
2	2005	2	75	

Ejercicio 94: Unidades mensuales despachadas de cada producto

Escriba una consulta que muestre el total de unidades mensuales despachadas de cada producto. La consulta debe mostrar el nombre del producto.

```
SELECT Producto.idProducto, Producto.nombre,
       YEAR(Guia.fechaSalida) AS Año,
       MONTH(Guia.fechaSalida) AS Mes,
       SUM(Guia_detalle.cantidad) As Unidades
FROM Guia INNER JOIN Guia_detalle
      ON Guia.idGuia = Guia_detalle.idGuia
INNER JOIN Producto
      ON Guia_detalle.idProducto = Producto.idProducto
GROUP BY Producto.idProducto, Producto.nombre,
         YEAR(Guia.fechaSalida), MONTH(Guia.fechaSalida)
ORDER BY Producto.idProducto, Año, Mes
go
```

Script08-02-U...05.MarketPERU*						Summary
	idProducto	nombre	Año	Mes	Unidades	
1	1	CARAMELOS BASTON VIENA ARCOR	2005	1	300	
2	1	CARAMELOS BASTON VIENA ARCOR	2005	2	100	
3	2	CARAMELOS SURTIDO DE FRUTAS	2005	1	300	
4	2	CARAMELOS SURTIDO DE FRUTAS	2005	2	100	
5	3	CARAMELOS FRUTAS SURTIDA ARCOR	2005	1	300	
6	3	CARAMELOS FRUTAS SURTIDA ARCOR	2005	2	100	
7	4	CARAMELOS FRUTAS MASTICABLES	2005	1	300	
8	4	CARAMELOS FRUTAS MASTICABLES	2005	2	100	

Outer join

Un **outer join** es la consulta correlacionada que entrega todas las filas que están relacionadas, y además:

- las filas no relacionadas de la tabla izquierda (LEFT OUTER JOIN), ó
- las filas no relacionadas de la tabla derecha (RIGHT OUTER JOIN), ó
- las filas no relacionadas de ambas tablas (FULL OUTER JOIN)

Se considera como la tabla izquierda, a aquella que se menciona primero en la cláusula FROM.

Ejercicio 95: Uso de OUTER JOIN

Se desea obtener una lista de los productos que NO registran salida del almacén. Los productos que registran salida del almacén son aquellos cuyo **idProducto** figura en la tabla **Guia_detalle**. Por lo tanto, la consulta a diseñar debe buscar productos cuyo **idProducto** no se encuentra en la tabla **Guia_detalle**; es decir, aquellos productos registrados en la tabla **Producto**, y que no tienen relación con la tabla **Guia_detalle**.

```
SELECT Producto.idProducto, Producto.nombre
FROM Producto LEFT OUTER JOIN Guia_detalle
    ON Producto.idProducto = Guia_detalle.idProducto
ORDER BY Producto.idProducto
go
```

Script08-02-U...05.MarketPERU*		Script08-03-U...05.M...
	idProducto	nombre
1	1	CARAMELOS BASTON VIENA ARCOR
2	1	CARAMELOS BASTON VIENA ARCOR
3	1	CARAMELOS BASTON VIENA ARCOR
4	1	CARAMELOS BASTON VIENA ARCOR
5	1	CARAMELOS BASTON VIENA ARCOR
6	1	CARAMELOS BASTON VIENA ARCOR
7	1	CARAMELOS BASTON VIENA ARCOR

Observe que el producto **1** aparece varias veces en el resultado. Esto indica que del producto **1** se tienen varias salidas registradas en la tabla **Guia_detalle**. Recuerde que una consulta OUTER JOIN entrega filas relacionadas y filas no relacionadas.

La consulta es LEFT OUTER JOIN porque estamos buscando valores de **idProducto** que se encuentran en la tabla **Producto**, pero que no están en la tabla **Guia_detalle**.

Añada a la consulta anterior la columna **cantidad** de la tabla **Guia_detalle**.

```
SELECT Producto.idProducto, Producto.nombre,
       Guia_detalle.cantidad
FROM Producto LEFT OUTER JOIN Guia_detalle
      ON Producto.idProducto = Guia_detalle.idProducto
ORDER BY Producto.idProducto
go
```

Script08-02-U...05.MarketPERU*		Script08-03-U...05.MarketPERU	
	idProducto	nombre	cantidad
77	4	CARAMELOS FRUTAS MASTICABLES	20
78	4	CARAMELOS FRUTAS MASTICABLES	20
79	4	CARAMELOS FRUTAS MASTICABLES	20
80	4	CARAMELOS FRUTAS MASTICABLES	20
81	5	CHUPETES LOLY AMBROSOLI	NULL
82	6	FRUNA SURTIDA DONOFRIO	30
83	6	FRUNA SURTIDA DONOFRIO	30
84	6	FRUNA SURTIDA DONOFRIO	30
85	6	FRUNA SURTIDA DONOFRIO	30
86	6	FRUNA SURTIDA DONOFRIO	30
87	6	FRUNA SURTIDA DONOFRIO	30
88	6	FRUNA SURTIDA DONOFRIO	30
89	6	FRUNA SURTIDA DONOFRIO	30
90	6	FRUNA SURTIDA DONOFRIO	30
91	6	FRUNA SURTIDA DONOFRIO	30

Note que para el producto **5**, el valor en **cantidad** es NULL. Este producto es uno de los productos que NO registra salida del almacén.

Modifique la consulta anterior para que muestre solo los productos que NO registran salida del almacén.

```
SELECT Producto.idProducto, Producto.nombre
FROM Producto LEFT OUTER JOIN Guia_detalle
    ON Producto.idProducto = Guia_detalle.idProducto
WHERE Guia_detalle.cantidad IS NULL
ORDER BY Producto.idProducto
go
```

Script08-02-U...05.MarketPERU*		Script08-03-U...05.Marke
	idProducto	nombre
1	5	CHUPETES LOLY AMBROSOLI
2	13	CHOCOLATE BARRA MILKY WAY
3	14	SNICKERS BAR KING SIZE
4	15	CHOCOLATE BARRA MILK DOVE
5	16	CHOCOLATE BARRA DARK DOVE
6	17	MILKY WAY BAR KING SIZE
7	18	GALLETAS CHIPS AHOY
8	19	GALLETAS TUAREG COSTA
9	20	GALLETAS VAINILLA COSTA
10	21	GALLETAS SURTIDAS BUTTER COOKIES
11	22	CHOCOLATE LOVER CHIPS DELUXE
12	23	FUDGE SHOPPE DELUXE GRAHAMS
13	24	FUDGE SHOPPE STICKS KEEB
14	25	GALLETAS DELICE
15	31	JAMON INGLES SAN FERNANDO
16	32	JAMON INGLES LAIVE
17	33	JAMON LIGHT BRAEDT
18	34	JAMON YORK BRAEDT

Hay 72 productos que NO tienen registrada salida del almacén, que sumados a los 66 productos que tienen salida registrada, nos da un total de 138 productos.

Ejercicio 96: Reporte de unidades despachadas de cada producto

Diseñe una consulta que muestre cuántas unidades se han despachado en total para cada uno de los productos. El resultado debe mostrar dicho valor para todos los productos.

```
SELECT Producto.idProducto, Producto.nombre,
       SUM(Guia_detalle.cantidad) AS Unidades
FROM Producto INNER JOIN Guia_detalle
       ON Producto.idProducto = Guia_detalle.idProducto
GROUP BY Producto.idProducto, Producto.nombre
ORDER BY Producto.idProducto
go
```

Script08-04-U...05.MarketPERU*		Script08-03-U...05.MarketPERU	
	idProducto	nombre	Unidades
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	6	FRUNA SURTIDA DONOFRIO	600
6	7	CHOCOLATE DOÑA PEPA FIELD	500
7	8	CHOCOLATE CUA CUA FIELD	500
8	9	MELLOWS FAMILIAR FIELD	400
9	10	WAFER CHOCOLATE FIELD	400
10	11	CHOCOLATE BARRA REGULAR	1000
11	12	CHOCOLATE MOSTRO FIELD	1000
12	26	JAMONADA LAIVE	300
13	27	JAMONADA ESPECIAL LA SEGOVIANA	300
14	28	JAMONADA POLACA OTTO KUNZ	300
15	29	JAMONADA DE POLLO SAN FERNANDO	400

Note que el resultado no muestra todos los productos. Para que se muestren todos los productos convierta la consulta INNER JOIN en una consulta OUTER JOIN, y para los productos en los que **cantidad** es NULL que se muestre **0** (cero) como el valor en **cantidad**.

```
SELECT Producto.idProducto, Producto.nombre,  
       ISNULL(SUM(Guia_detalle.cantidad), 0) AS Unidades  
FROM Producto LEFT OUTER JOIN Guia_detalle  
       ON Producto.idProducto = Guia_detalle.idProducto  
GROUP BY Producto.idProducto, Producto.nombre  
ORDER BY Producto.idProducto  
go
```

Script08-04-U...05.MarketPERU*		Script08-03-U...05.MarketPERU	
	idProducto	nombre	Unidades
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	5	CHUPETES LOLY AMBROSOLI	0
6	6	FRUNA SURTIDA DONOFRIO	600
7	7	CHOCOLATE DOÑA PEPA FIELD	500
8	8	CHOCOLATE CUA CUA FIELD	500
9	9	MELLOWS FAMILIAR FIELD	400
10	10	WAFER CHOCOLATE FIELD	400
11	11	CHOCOLATE BARRA REGULAR	1000
12	12	CHOCOLATE MOSTRO FIELD	1000
13	13	CHOCOLATE BARRA MILKY WAY	0
14	14	SNICKERS BAR KING SIZE	0
15	15	CHOCOLATE BARRA MILK DOVE	0
16	16	CHOCOLATE BARRA DARK DOVE	0
17	17	MILKY WAY BAR KING SIZE	0
18	18	GALLETAS CHIPS AHOY	0

Cross join

Un **cross join** es la consulta correlacionada que combina cada una de las filas de una de las tablas con todas las filas de la otra tabla.

No es necesario que exista una columna común para ejecutar cross join. Esta consulta también se conoce como el **producto cartesiano** de dos tablas.

Ejercicio 97: Uso de CROSS JOIN

```
SELECT Categoria.idCategoria, Categoria.categoria,
       Producto.idProducto, Producto.nombre
FROM Categoria CROSS JOIN Producto
ORDER BY Categoria.idCategoria, Producto.idProducto
go
```

	idCategoria	categoria	idProducto	nombre
18	1	GOLOSINAS	18	GALLETAS CHIPS AHOY
19	1	GOLOSINAS	19	GALLETAS TUAREG COSTA
20	1	GOLOSINAS	20	GALLETAS VAINILLA COSTA
21	1	GOLOSINAS	21	GALLETAS SURTIDAS BUTTER COOKI...
22	1	GOLOSINAS	22	CHOCOLATE LOVER CHIPS DELUXE
23	1	GOLOSINAS	23	FUDGE SHOPPE DELUXE GRAHAMS
24	1	GOLOSINAS	24	FUDGE SHOPPE STICKS KEEB
25	1	GOLOSINAS	25	GALLETAS DELICE
26	1	GOLOSINAS	26	JAMONADA LAIVE
27	1	GOLOSINAS	27	JAMONADA ESPECIAL LA SEGOVIANA
28	1	GOLOSINAS	28	JAMONADA POLACA OTTO KUNZ
29	1	GOLOSINAS	29	JAMONADA DE POLLO SAN FERNANDO
30	1	GOLOSINAS	30	JAMONADA ESPECIAL OTTO KUNZ
31	1	GOLOSINAS	31	JAMON INGLES SAN FERNANDO
32	1	GOLOSINAS	32	JAMON INGLES LAIVE
33	1	GOLOSINAS	33	JAMON LIGHT BRAEDT
34	1	GOLOSINAS	34	JAMON YORK BRAEDT
35	1	GOLOSINAS	35	JAMON INGLES LA SEGOVIANA

La tabla **Categoria** tiene 6 categorías:

```
SELECT COUNT(*) FROM Categoria  
go
```

La tabla **Producto** tiene 138 productos:

```
SELECT COUNT(*) FROM Producto  
go
```

El resultado de la consulta CROSS JOIN entrega $6 \times 138 = 828$ filas.

El operador UNION

Sintaxis

```
sentencia_SELECT_1  
  
UNION [ ALL ]  
  
sentencia_SELECT_2  
  
UNION [ ALL ]  
  
sentencia_SELECT_3, ...
```

El operador UNION une los resultados de dos ó más instrucciones SELECT en un solo conjunto de resultados.

Use el operador UNION cuando los datos que desea recuperar residen en diferentes localizaciones y no puede acceder a ellos con una sola consulta. Cuando use el operador UNION considere lo siguiente:

- SQL Server requiere que las consultas a las tablas referenciadas tengan el mismo número de columnas, los mismos tipos de datos, y que las columnas se encuentren en el mismo orden en la lista de cada uno de los SELECT.
- SQL Server elimina las filas duplicadas en el resultado. Sin embargo, si usa la opción ALL, todas las filas (incluso las duplicadas) son incluidas en el resultado.
- Debe especificar los nombres de las columnas en la primera instrucción SELECT. Por consiguiente, si quiere definir los nuevos títulos de las columnas para el resultado, debe crear los seudónimos de las columnas en la primera instrucción SELECT.
- Si quiere que el resultado completo sea devuelto en un orden específico, debe especificar el orden e incluir la cláusula ORDER BY dentro de la sentencia UNION.

Ejercicio 98: Uso del operador UNION

Genere un reporte que muestre todos los documentos registrados (órdenes de compra y guías de remisión) en la base de datos **MarketPERU**. El reporte debe ordenar los documentos por su fecha de emisión e indicar para cada uno, cuál es el tipo de documento.

```
SELECT fechaOrden AS 'Fecha emisión',
       idOrden AS 'Número documento',
       'Orden de Compra' AS 'Tipo documento'
FROM Orden
UNION
SELECT fechaSalida, idGuia, 'Guía de Remisión'
FROM Guia
ORDER BY 'Fecha emisión'
go
```

Script08-07-U...05.MarketPERU*			
Summary			
	Fecha emisión	Número documento	Tipo documento
20	2005-01-23 11:38:54.340	20	Guía de Remisión
21	2005-01-23 11:38:54.340	21	Guía de Remisión
22	2005-01-23 11:38:54.340	22	Guía de Remisión
23	2005-01-23 11:38:54.340	23	Guía de Remisión
24	2005-01-23 11:38:54.340	24	Guía de Remisión
25	2005-01-23 11:38:54.357	25	Guía de Remisión
26	2005-01-23 11:38:54.357	26	Guía de Remisión
27	2005-01-23 11:38:54.357	27	Guía de Remisión
28	2005-01-23 11:38:54.357	28	Guía de Remisión
29	2005-01-23 11:38:54.357	29	Guía de Remisión
30	2005-01-23 11:38:54.373	30	Guía de Remisión
31	2005-01-23 11:38:54.683	1	Orden de Compra
32	2005-01-23 11:38:54.683	2	Orden de Compra
33	2005-01-23 11:38:54.700	3	Orden de Compra
34	2005-01-23 11:38:54.700	4	Orden de Compra
35	2005-01-24 11:38:54.373	31	Guía de Remisión
36	2005-01-24 11:38:54.373	32	Guía de Remisión

La instrucción SELECT...INTO

Puede colocar el resultado de cualquier consulta en una nueva tabla usando la sentencia SELECT...INTO.

Use la sentencia SELECT...INTO para crear nuevas tablas en la base de datos. También puede usar la sentencia SELECT INTO para solucionar problemas en los que requiere leer datos desde varias fuentes.

Sintaxis

```
SELECT lista_columnas
INTO nombre_nueva_tabla
FROM tabla
WHERE condición_filas
```

- SELECT...INTO siempre crea la tabla destino. Si el nombre especificado en **nombre_nueva_tabla** ya existe se produce un error.

Ejercicio 99: Uso de SELECT...INTO

Crear una tabla de nombre **Monto_guias** que registre el monto de cada una de las guías de remisión.

```
SELECT Guia.idGuia, Guia.fechaSalida,
       SUM(Guia_detalle.precioVenta * Guia_detalle.cantidad)
AS Monto
INTO Monto_guias
FROM Guia INNER JOIN Guia_detalle
ON Guia.idGuia = Guia_detalle.idGuia
GROUP BY Guia.idGuia, Guia.fechaSalida
ORDER BY Guia.idGuia
go
```

Revise la tabla resultante.

```
SELECT * FROM Monto_guias
go
```

Script08-08-U...05.MarketPERU*				Summary
	idGuia	fechaSalida	Monto	
1	1	2005-01-23 11:38:54.263	609.00	
2	2	2005-01-23 11:38:54.263	609.00	
3	3	2005-01-23 11:38:54.263	609.00	
4	4	2005-01-23 11:38:54.280	609.00	
5	5	2005-01-23 11:38:54.280	609.00	
6	6	2005-01-23 11:38:54.280	3039.00	
7	7	2005-01-23 11:38:54.280	3039.00	

Ejercicio 100: Creación de una tabla temporal

Cuando en una instrucción que crea una tabla se especifica el nombre de la tabla precedido del símbolo #, el objeto se crea como una tabla temporal.

```
SELECT Orden.idOrden, Orden.fechaOrden,
       SUM(Orden_detalle.precioCompra *
           Orden_detalle.cantidadRecibida)
       AS Monto
INTO #Monto_ordenes
FROM Orden INNER JOIN Orden_detalle
      ON Orden.idOrden = Orden_detalle.idOrden
GROUP BY Orden.idOrden, Orden.fechaOrden
ORDER BY Orden.idOrden
go
```

Para consultar la tabla temporal, ejecute:

```
SELECT * FROM #Monto_ordenes
go
```

Consulta autojoin

Es una consulta correlacionada en la que una tabla se combina consigo misma para generar un nuevo conjunto de resultados.

Ejercicio 101: Consulta autojoin

Para ilustrar este tipo de consulta crearemos una tabla que tenga una autorelación. Esta tabla contiene una clave foránea que apunta a la clave primaria en la misma tabla.

Ejecute las siguientes instrucciones para crear y cargar los datos en la tabla.

```
USE MarketPERU
go

-- Consulta autojoin, creación de la tabla con autorelación
CREATE TABLE Trabajador(
    idTrabajador int PRIMARY KEY,
    Apellidos varchar(30) not null,
    Jefe int null )
go

ALTER TABLE Trabajador
    ADD CONSTRAINT fk_Trabajador_Trabajador
    FOREIGN KEY(Jefe)
    REFERENCES Trabajador
go

INSERT INTO Trabajador VALUES(102, 'Ardiles Soto', NULL)
INSERT INTO Trabajador VALUES(101, 'Camacho Saravia', 102)
INSERT INTO Trabajador VALUES(105, 'Vilchez Santos', 102)
INSERT INTO Trabajador VALUES(103, 'Sánchez Aliaga', 101)
INSERT INTO Trabajador VALUES(104, 'Castro Avila', 101)
INSERT INTO Trabajador VALUES(107, 'Urrunaga Tapia', 101)
INSERT INTO Trabajador VALUES(106, 'Juárez Pinto', 105)
go

SELECT * FROM Trabajador
go
```

SQLQuery2.sql...5.MarketPERU*			
Summary			
	idTrabajador	Apellidos	Jefe
1	101	Camacho Saravia	102
2	102	Ardiles Soto	NULL
3	103	Sánchez Aliaga	101
4	104	Castro Ávila	101
5	105	Vilchez Santos	102
6	106	Juárez Pinto	105
7	107	Urrunaga Tapia	101

La columna **jefe** de la tabla **Trabajador** registra el código del jefe de un trabajador. Por ejemplo, el trabajador **101 (Camacho Saravia)** es el jefe de los trabajadores **103, 104 y 107**.

Se desea crear una consulta que muestre una lista de trabajadores. La lista debe mostrar los **apellidos** del **jefe** de cada trabajador.

```
SELECT T1.idTrabajador, T1.apellidos, T2.apellidos AS Jefe
FROM Trabajador T1 INNER JOIN Trabajador T2
    ON T1.jefe = T2.idTrabajador
go
```

SQLQuery2.sql...5.MarketPERU*			
Summary			
	idTrabajador	apellidos	Jefe
1	101	Camacho Saravia	Ardiles Soto
2	103	Sánchez Aliaga	Camacho Saravia
3	104	Castro Ávila	Camacho Saravia
4	105	Vilchez Santos	Ardiles Soto
5	106	Juárez Pinto	Vilchez Santos
6	107	Urrunaga Tapia	Camacho Saravia

Note que el resultado muestra a todos los trabajadores con su respectivo jefe, pero el trabajador **102 (Ardiles Soto)** no aparece en la lista porque él no tiene **jefe**.

Modifique la consulta para que también se muestre al trabajador **102**.

```
SELECT T1.idTrabajador, T1.apellidos, T2.apellidos AS Jefe
FROM Trabajador T1 LEFT OUTER JOIN Trabajador T2
    ON T1.jefe = T2.idTrabajador
go
```

SQLQuery2.sql...5.MarketPERU*			
Summary			
	idTrabajador	apellidos	Jefe
1	101	Camacho Saravia	Ardiles Soto
2	102	Ardiles Soto	NULL
3	103	Sánchez Aliaga	Camacho Saravia
4	104	Castro Ávila	Camacho Saravia
5	105	Vilchez Santos	Ardiles Soto
6	106	Juárez Pinto	Vilchez Santos
7	107	Urrunaga Tapia	Camacho Saravia

Subconsultas

Un subconsulta es una declaración SELECT anidada dentro una sentencia SELECT, INSERT, UPDATE o DELETE o dentro de otra subconsulta.

Si la respuesta a un requerimiento de datos requiere la ejecución de una serie de pasos lógicos, utilice subconsultas para tratar de resolver el requerimiento con una sola sentencia.

Las subconsultas son de los tipos siguientes:

- Subconsulta que entrega un solo valor (1 fila, 1 columna)
- Subconsulta que entrega un conjunto de valores (varias filas, 1 columna)

Una subconsulta se especifica entre paréntesis, y se puede especificar en cualquier donde la sintáxis permite una expresión.

Subconsulta que entrega un solo valor (1 fila, 1 columna)

Cuando la subconsulta se especifica:

- en la lista de columnas del SELECT externo, ó
- en la cláusula WHERE del SELECT externo usando un operador relacional (test de comparación),

la subconsulta debe ser una que entregue un solo valor.

Ejercicio 102: Subconsulta definida en la lista de columnas del SELECT externo

Genere una consulta que entregue la lista de precios de todos los productos, especificando en una columna adicional la diferencia entre el precio de cada producto y el precio promedio de todos los productos.

Primero, especifique la consulta que entrega el precio promedio de todos los productos.

```
SELECT AVG(precioProveedor) FROM Producto
go
```

SQLQuery1.sql...5.Mark	
	[No column name]
1	5.7468

Ahora, escriba la consulta que entrega la lista de precios solicitada teniendo en cuenta la fórmula que determina la diferencia entre el precio de cada producto y el precio promedio de todos los productos.

```
SELECT idProducto, nombre, precioProveedor,
       Diferencia = precioProveedor -
       (SELECT AVG(precioProveedor) FROM Producto)
FROM Producto
go
```

SQLQuery1.sql...5.MarketPERU*		Summary		
	idProducto	nombre	precioProveedor	Diferencia
1	1	CARAMELOS BASTON VIENA ARCOR	1.50	-4.2468
2	2	CARAMELOS SURTIDO DE FRUTAS	1.00	-4.7468
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1.50	-4.2468
4	4	CARAMELOS FRUTAS MASTICABLES	1.30	-4.4468
5	5	CHUPETES LOLY AMBROSOLI	1.20	-4.5468
6	6	FRUNA SURTIDA DONOFRIO	1.80	-3.9468
7	7	CHOCOLATE DOÑA PEPA FIELD	2.20	-3.5468
8	8	CHOCOLATE CUA CUA FIELD	1.60	-4.1468
9	9	MELLOWS FAMILIAR FIELD	2.10	-3.6468
10	10	WAFER CHOCOLATE FIELD	0.70	-5.0468
11	11	CHOCOLATE BARRA REGULAR	0.40	-5.3468
12	12	CHOCOLATE MOSTRO FIELD	1.50	-4.2468
13	13	CHOCOLATE BARRA MILKY WAY	0.80	-4.9468

Ejercicio 103: Porcentaje despachado de cada producto respecto al total despachado para la categoría X

Escriba una consulta que determine el porcentaje de unidades despachadas de cada producto de la categoría 4 respecto al total despachado de la categoría.

Primero, escriba la consulta que calcula el total despachado para la categoría 4.

```
SELECT SUM(cantidad)
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto =
        Producto.idProducto
WHERE Producto.idCategoría = 4
go
```

SQLQuery1.sql...5.Mark	
	(No column name)
1	10625

Ahora, escriba la consulta que, utilizando la consulta anterior, presente el listado requerido.

```
SELECT Producto.idProducto, Producto.nombre,
    Despachado = ISNULL(SUM(Guia_detalle.cantidad), 0),
    Porcentaje =
    CONVERT(float, ISNULL(SUM(Guia_detalle.cantidad), 0)) /
    (SELECT SUM(cantidad)
        FROM Guia_detalle INNER JOIN Producto
            ON Guia_detalle.idProducto = Producto.idProducto
        WHERE Producto.idCategoría = 4) * 100
FROM Producto LEFT OUTER JOIN Guia_detalle
    ON Producto.idProducto = Guia_detalle.idProducto
WHERE Producto.idCategoría = 4
GROUP BY Producto.idProducto, Producto.nombre
go
```

SQLQuery1.sql...5.MarketPERU* Summary				
	idProducto	nombre	Despachado	Porcentaje
1	92	CREMA DE LECHE LAIVE	425	4
2	93	CREMA DE LECHE DUPRE	425	4
3	94	CREMA DE LECHE NESTLE	0	0
4	95	YOGURT GLORIA FRESA	850	8
5	96	YOGURT YOLEIT FRESA	0	0

Ejercicio 104: Subconsulta definida en el WHERE del SELECT externo

Escriba una consulta que entregue una lista de los productos que se despacharon en la fecha que se despachó la última salida del almacén. Tenga en cuenta que en dicha fecha se puede haber registrado más de una salida.

Primero, obtenga la fecha de la última salida

```
SELECT MAX(fechaSalida) FROM Guia
go
```

Script08-11-5...05.MarketPER	
	(No column name)
1	2005-02-02 11:38:54.683

Ahora, utilizando adecuadamente la consulta anterior, escriba la consulta que responde al requerimiento especificado.

```
SELECT DISTINCT Guia_detalle.idProducto, Producto.nombre
FROM Guia_detalle INNER JOIN Producto
    ON Guia_detalle.idProducto = Producto.idProducto
INNER JOIN Guia
    ON Guia_detalle.idGuia = Guia.idGuia
WHERE CONVERT(char(10), Guia.fechaSalida, 103) =
    (SELECT CONVERT(char(10), MAX(fechaSalida), 103)
     FROM Guia)
go
```

Script08-11-5...05.MarketPERU* Summary		
	idProducto	nombre
1	1	CARAMELOS BASTON VIENA ARCOR
2	2	CARAMELOS SURTIDO DE FRUTAS
3	3	CARAMELOS FRUTAS SURTIDA ARCOR
4	4	CARAMELOS FRUTAS MASTICABLES
5	6	FRUNA SURTIDA DONOFRIO
6	7	CHOCOLATE DOÑA PEPA FIELD
7	8	CHOCOLATE CUA CUA FIELD
8	9	MELLOWS FAMILIAR FIELD
9	10	WAFER CHOCOLATE FIELD
10	11	CHOCOLATE BARRA REGULAR
11	12	CHOCOLATE MOSTRO FIELD
12	26	JAMONADA LAIVE

Subconsulta que entrega un conjunto de valores (varias filas, 1 columna)

Cuando la subconsulta se define en la cláusula WHERE del SELECT externo utilizando el operador IN (test de pertenencia), puede ser una subconsulta que entrega un conjunto de valores.

Ejercicio 105: Test de pertenencia

Escriba una consulta que entregue una lista de los productos que no registran salida del almacén. Recuerde que este requerimiento fue resuelto líneas arriba utilizando una consulta OUTER JOIN.

```
SELECT idProducto, nombre
FROM Producto
WHERE idProducto NOT IN
    (SELECT idProducto FROM Guia_detalle)
ORDER BY idProducto
go
```

Script08-12-T...5.MarketPERU*			Summary
	idProducto	nombre	
1	5	CHUPETES LOLY AMBROSOLI	
2	13	CHOCOLATE BARRA MILKY WAY	
3	14	SNICKERS BAR KING SIZE	
4	15	CHOCOLATE BARRA MILK DOVE	
5	16	CHOCOLATE BARRA DARK DOVE	
6	17	MILKY WAY BAR KING SIZE	
7	18	GALLETAS CHIPS AHOY	
8	19	GALLETAS TUAREG COSTA	
9	20	GALLETAS VAINILLA COSTA	
10	21	GALLETAS SURTIDAS BUTTER COOKIES	
11	22	CHOCOLATE LOVER CHIPS DELUXE	
12	23	FUDGE SHOPPE DELUXE GRAHAMS	
13	24	FUDGE SHOPPE STICKS KEEB	
14	25	GALLETAS DELICE	
15	31	JAMON INGLES SAN FERNANDO	
16	32	JAMON INGLES LAIVE	
17	33	JAMON LIGHT BRAEDT	

Subconsulta correlacionada

Se presenta cuando la consulta externa debe entregar datos a la consulta interna para que se pueda ejecutar.

- La consulta interna se evalúa repetidamente, una vez por cada fila de la consulta externa.
- Se puede definir en la cláusula WHERE de la consulta externa usando el operador EXISTS (Test de existencia).

Ejercicio 106: Test de existencia – Uso de EXISTS

Genere la lista de productos que registran salida del almacén.

```
SELECT Producto.idProducto, Producto.nombre
FROM Producto
WHERE EXISTS
    (SELECT * FROM Guia_detalle
     WHERE Producto.idProducto = Guia_detalle.idProducto)
ORDER BY Producto.idProducto
go
```

Script08-13-T...05.MarketPERU*			Summary
	idProducto	nombre	
1	1	CARAMELOS BASTON VIENA ARCOR	
2	2	CARAMELOS SURTIDO DE FRUTAS	
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	
4	4	CARAMELOS FRUTAS MASTICABLES	
5	6	FRUNA SURTIDA DONOFRIO	
6	7	CHOCOLATE DOÑA PEPA FIELD	
7	8	CHOCOLATE CUA CUA FIELD	
8	9	MELLOWS FAMILIAR FIELD	
9	10	WAFER CHOCOLATE FIELD	
10	11	CHOCOLATE BARRA REGULAR	
11	12	CHOCOLATE MOSTRO FIELD	
12	26	JAMONADA LAIVE	
13	27	JAMONADA ESPECIAL LA SEGOVIANA	
14	28	JAMONADA POLACA OTTO KUNZ	
15	29	JAMONADA DE POLLO SAN FERNANDO	
16	30	JAMONADA ESPECIAL OTTO KUNZ	
17	37	HOT DOG LAIVE PELADO	

Inserción de filas con datos leídos por SELECT

Sintáxis

```
INSERT [ INTO ] tabla_destino
    SELECT lista_columnas
    FROM tabla_origen
    [ WHERE condición_filas_tabla_origen ]
```

Ejercicio 107: Inserción de filas con subconsulta

Se desea generar una lista de correos electrónicos a partir de la columna **representante** de la tabla **Proveedor**. La lista se almacenará en la tabla **Correo**.

Primero, cree la tabla **Correo**.

```
USE MarketPERU
go

CREATE TABLE Correo(
    Nombre varchar(50),
    Email varchar(35) )
go
```

Ahora, diseñe la consulta con la que se llenará la tabla **Correo**. El formato de la dirección será **npaterno@marketperu.com**, donde **n** es la inicial del nombre del representante, y **paterno** es su apellido paterno.

```
INSERT INTO Correo
    SELECT representante,
        LOWER(SUBSTRING(representante,
            PATINDEX('%', '%', representante)+2, 1) +
            SUBSTRING(representante, 1,
                PATINDEX('% %', representante)-1)) +
            '@marketperu.com'
    FROM Proveedor
go
```

A continuación, revise el contenido de la tabla **Correo**.

```
SELECT * FROM Correo  
go
```

Script08-14-I...5.MarketPERU*			Summary
	Nombre	Email	
1	AREVALO SANCHIZ, WALTER	warevalo@marketperu.com	
2	SCHULTZ SORIA, JACOBO	jschultz@marketperu.com	
3	GORDILLO BARRIGA, SANCHO	sgordillo@marketperu.com	
4	ALVARADO VERTIZ, FERNANDO	falvarado@marketperu.com	
5	ALEGRE PINTADO, ALICIA	aalegre@marketperu.com	
6	SORIANO OLAECHEA, VICTOR	vsoriano@marketperu.com	
7	QUIROGA QUIROZ, ESTHER	equiroga@marketperu.com	
8	SANTOS VELA, ARTURO	asantos@marketperu.com	
9	DEL PINO ALARCON, URSULA	udel@marketperu.com	
10	MARTICORENA MEJIA, JUAN	jmarticorena@marketperu.com	
11	CHAVEZ VICTORINO, VICTOR	vchavez@marketperu.com	
12	VICENTE ALIAGA, VICTORINO	vvicente@marketperu.com	
13	MALCA UBIDIA, JOSEFINA	jmalca@marketperu.com	
14	ZAVALA ZEGARRA, CLAUDIA	czavala@marketperu.com	
15	FELICES ARSENIO, JOSE	jfelices@marketperu.com	

El operador PIVOT

Con mucha frecuencia necesitamos ver los datos en base a múltiples dimensiones (ó entidades). En las versiones anteriores de SQL Server disponemos de los operadores ROLLUP y CUBE que se utilizan con la cláusula GROUP BY, y permiten ver data resumida en base a distintas dimensiones.

SQL Server 2005 incorpora el operador PIVOT que es más fácil de entender e implementar que los operadores ROLLUP y CUBE. Por ejemplo, con el operador PIVOT podemos rotar filas y mostrarlas como columnas para obtener una vista diferente de los datos. El resultado de PIVOT es una tabla de doble entrada.

Sintáxis

```
SELECT...  
...  
PIVOT( función_agregación( columna_numérica )  
      FOR columna_dimensión  
      IN ( lista_valores_columna_dimensión ) )
```

- **columna_numérica** es la columna cuyos valores se desea mostrar en una tabla de doble entrada.
- **columna_dimensión** es la columna cuyos valores en una consulta normal se ven como filas, y que en la consulta PIVOT se desea ver como columnas.
- **lista_valores_columna_dimensión** son los valores de columna_dimensión a mostrar como columnas en la tabla de doble entrada.

Ejercicio 108: Uso del operador PIVOT

Se desea un reporte que muestre el monto mensual enviado a cada local durante el año 2005.

Primero, diseñe una consulta que guarde en una tabla el monto mensual enviado a cada local en el año 2005.

```
SELECT Guia.idLocal,
       DATEPART(month, Guia.fechaSalida) AS Mes,
       SUM(Guia_detalle.cantidad * Guia_detalle.precioVenta)
AS Monto
INTO DespachadoMensualPorLocal_2005
FROM Guia INNER JOIN Local
      ON Guia.idLocal = Local.idLocal
INNER JOIN Guia_detalle
      ON Guia.idGuia = Guia_detalle.idGuia
WHERE DATEPART(year, Guia.fechaSalida) = 2005
GROUP BY Guia.idLocal, DATEPART(month, Guia.fechaSalida)
go
```

```
SELECT * FROM DespachadoMensualPorLocal_2005
go
```

SQL Query1.sql...5.MarketPERU*			
			Summary
	idLocal	Mes	Monto
1	1	1	67607.25
2	2	1	67607.25
3	3	1	67607.25
4	4	1	67607.25
5	5	1	67607.25
6	1	2	8756.50
7	2	2	8756.50
8	3	2	7515.25
9	4	2	7515.25
10	5	2	7515.25

Ahora, consulte la tabla **DespachadoMensualPorLocal_2005** "pivoteando" la columna **mes** para que los meses se muestren como columnas.

```
SELECT * FROM DespachadoMensualPorLocal_2005
PIVOT (SUM(monto) FOR mes IN ( [1], [2], [3], [4] )) AS a
go
```

SQLQuery1.sql...5.MarketPERU*					
Summary					
	idLocal	1	2	3	4
1	1	67607.25	8756.50	NULL	NULL
2	2	67607.25	8756.50	NULL	NULL
3	3	67607.25	7515.25	NULL	NULL
4	4	67607.25	7515.25	NULL	NULL
5	5	67607.25	7515.25	NULL	NULL

Finalmente, pivotee nuevamente la consulta para que ahora sean los locales los que se muestren como columnas.

```
SELECT * FROM DespachadoMensualPorLocal_2005
PIVOT (SUM(monto) FOR idLocal IN ( [1], [2], [3], [4], [5]
)) AS a
go
```

SQLQuery1.sql...5.MarketPERU*						
Summary						
	Mes	1	2	3	4	5
1	1	67607.25	67607.25	67607.25	67607.25	67607.25
2	2	8756.50	8756.50	7515.25	7515.25	7515.25

Common Table Expression (CTE)

Un CTE es un conjunto de resultados temporal derivado de una consulta que puede ser utilizado como una tabla derivada. Su comportamiento es muy similar al de una vista. Una CTE puede contener referencias a ella misma, por lo que permite diseñar consultas recursivas.

Sintáxis

```
WITH nombre_CTE( lista_columnas )
AS
(
    SELECT_que_puebla_la_CTE
)
SELECT_que_muestra_la_CTE
```

- **SELECT_que_puebla_la_CTE** es la consulta que carga los datos en la CTE.
- **SELECT_que_muestra_la_CTE** es la consulta que muestra la data cargada en la CTE.

Ejercicio 109: Uso de una Common Table Expression

Cree una CTE que muestre el número de veces que fue despachado cada producto.

```
WITH DespachosCTE( IdProducto, Despachos )
AS
(
    SELECT idProducto, COUNT(idGuia)
    FROM Guia_detalle
    GROUP BY idProducto
)
SELECT * FROM DespachosCTE
```

	IdProducto	Despachos
1	92	17
2	43	20
3	26	20
4	135	15
5	9	20
6	118	15
7	129	15
8	3	20
9	112	15
10	95	17
11	63	20
12	29	20
13	12	20

Filtrando una CTE

```

WITH DespachosCTE( IdProducto, Despachos )
AS
(
    SELECT idProducto, COUNT(idGuia)
    FROM Guia_detalle
    GROUP BY idProducto
)
SELECT * FROM DespachosCTE
WHERE Despachos > 15
go

```

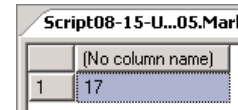
	IdProducto	Despachos
1	92	17
2	43	20
3	26	20
4	9	20
5	3	20
6	95	17
7	63	20

Uso de agregación en una CTE

```

WITH DespachosCTE( IdProducto, Despachos )
AS
(
    SELECT idProducto, COUNT(idGuia)
    FROM Guia_detalle
    GROUP BY idProducto
)
SELECT AVG(Despachos)
FROM DespachosCTE
go

```



Script08-15-U...05.Mar	
	(No column name)
1	17

Ejercicio 110: Consulta recursiva en una CTE

Las CTEs permiten el diseño de consultas recursivas. Este ejercicio muestra la estructura de una consulta recursiva.

Primero, proceda a crear y cargar la tabla **PartesAuto**:

```

USE MarketPERU
go

CREATE TABLE PartesAuto
(
    IdAuto int NOT NULL,
    Parte varchar(15),
    SubParte varchar(15),
    Cantidad int
)
go

INSERT PartesAuto
VALUES (1, 'Cuerpo', 'Puerta', 4)
INSERT PartesAuto
VALUES (1, 'Cuerpo', 'Puerta maletera', 1)

```

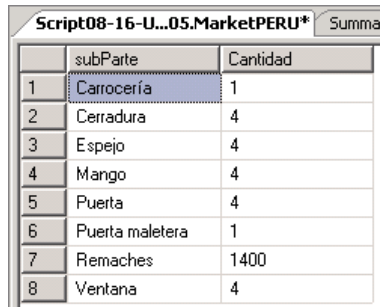
```
INSERT PartesAuto
    VALUES (1, 'Cuerpo', 'Carrocería', 1)
INSERT PartesAuto
    VALUES (1, 'Puerta', 'Mango', 1)
INSERT PartesAuto
    VALUES (1, 'Puerta', 'Cerradura', 1)
INSERT PartesAuto
    VALUES (1, 'Puerta', 'Ventana', 1)
INSERT PartesAuto
    VALUES (1, 'Cuerpo', 'Remaches', 1000)
INSERT PartesAuto
    VALUES (1, 'Puerta', 'Remaches', 100)
INSERT PartesAuto
    VALUES (1, 'Puerta', 'Espejo', 1)
go
```

Note que los datos de la tabla forman una lista de materiales para ensamblar el cuerpo de un auto. Esta lista de materiales tiene una estructura jerárquica.

Ahora, digite y ejecute la siguiente consulta:

```
WITH PartesAutoCTE(SubParte, Cantidad)
AS
(
    -- Miembro ancla - Anchor Member (AM):
    -- Consulta no recursiva
    SELECT subParte, cantidad
    FROM PartesAuto
    WHERE parte = 'Cuerpo'
    UNION ALL
    -- Miembro recursivo - Recursive Member (RM):
    -- Consulta recursiva
    SELECT PartesAuto.subParte, PartesAutoCTE.cantidad *
PartesAuto.cantidad
    FROM PartesAutoCTE INNER JOIN PartesAuto
        ON PartesAutoCTE.subParte = PartesAuto.parte
    WHERE PartesAuto.idAuto = 1
)
```

```
-- Consulta externa
SELECT subParte, SUM(cantidad) AS Cantidad
FROM PartesAutoCTE
GROUP BY subParte
go
```



	subParte	Cantidad
1	Carrocería	1
2	Cerradura	4
3	Espejo	4
4	Mango	4
5	Puerta	4
6	Puerta maletera	1
7	Remaches	1400
8	Ventana	4

La consulta muestra un listado que totaliza la cantidad de partes necesarias para ensamblar el cuerpo de un auto.

Una CTE recursiva es construida desde al menos dos consultas. Una, es una consulta no recursiva conocida como el miembro ancla. La segunda, es la consulta recursiva conocida como el miembro recursivo. Estas consultas van separadas por el operador UNION ALL.

La CTE es inicialmente cargada con el resultado de la consulta del miembro ancla. A continuación, el operador UNION ALL combina el resultado de la primera consulta con el resultado de la consulta recursiva.

Esta página se ha dejado en blanco intencionalmente.