Manual Práctico de SQL

ORIENTADO A SQL 7.0

Preparado por:
Alvaro E. García
alvaroegarcia@ubbi.com

ÍNDICE

INTRODUCCIÓN.

SQL (Lenguaje de Consulta Estructurado)

El SQL (Structure Query Language), es un lenguaje de consulta estructurado establecido claramente como el lenguaje de alto nivel estándar para sistemas de base de datos relacionales. Los responsables de publicar este lenguaje como estándar, fueron precisamente los encargados de publicar estándar, la ANSI (Instituto Americano de Normalización) y la ISO (organismo Internacional de Normalización). Es por lo anterior que este lenguaje lo vas a encontrar en cualquiera de los DBMS relacionales que existen en la actualidad, por ejemplo, ORACLE, SYBASES, SQL SERVER por mencionar algunos.

El SQL agrupa tres tipos de sentencias con objetivos particulares, en los siguientes lenguajes:

- ✓ Lenguaje de Definición de Datos (DDL, Data Definiton Language)
- ✓ Lenguaje de Manipulación de Datos (DML, Data Management Language)
- ✓ Lenguaje de Control de Datos (DCL, Data Control Language)

A continuación se describen cada uno de los lenguajes:

Lenguaje de Definición de Datos (DDL, Data Definiton Language)

Grupo de sentencias del SQL que soportan la definición y declaración de los objetos de la base de datos. Objetos tales como: la base de datos misma(DATABASE), las tablas(TABLE), las Vistas (VIEW), los índices (INDEX), los procedimientos almacenados (PROCEDURE), los disparadores (TRIGGER),

Reglas (RULE), Dominios (Domain) y Valores por defecto (DEFAULT).

CREATE,

ALTER y

DROP

Lenguaje de Manipulación de Datos (DML, Data Management Language)

Grupo de sentencias del SQL para manipular los datos que están almacenados en las bases de datos, a nivel de filas (tuplas) y/o columnas (atributos). Ya sea que se requiera que los datos sean modificados, eliminados, consultados o que se agregaren nuevas filas a las tablas de las base de datos.

INSERT,

UPDATE,

DELETE y

SELECT

Lenguaje de Control de Datos (DCL, Data Control Language)

Grupo de sentencias del SQL para controlar las funciones de administración que realiza el DBMS, tales como la atomicidad y seguridad.

COMMIT TRANSACTION,

ROLLBACK TRANSACTION,

GRANT

REVOKE

4. Definir una llave primaria para una tabla. La columna(s) a la cual se le dará esta responsabilidad debe contener previamente valores únicos por fila.

Ejemplo: ALTER TABLE T1 ADD PRIMARY KEY (a1,a2)

5. Definir una nueva llave foránea para una tabla. La columna a definir como llave foránea debe contener previamente valores que corresponden a la llave primaria de otra tabla. Es similar al descrito en llave primaria, solo que la palabra clave reservada es ADD FOREIGN KEY .

6. Se puede habilitar o inhabilitar los disparadores (trigger) en una tabla.

ALTER TABLE nombre_de_la_tabla ENABLE TRIGGER nombre_de_trigger

ALTER TABLE nombre_de_la_tabla DISABLE TRIGGER nombre_de_trigger

La persona que puede realizar esta operación de alterar la tabla puede ser: el DBA o el creador de la tabla, u otra persona que el creador o DBA haya autorizado. Para la ultima situación en que a una persona se le otorga el permiso en la operación, el usuario, debe calificar el nombre de la tabla con el nombre de su creador por delante y alterar la definición de la tabla de otro usuario, por ejemplo:

ALTER TABLE nombredueño.nombretabla

Consultas Simples

El corazón o poder del Lenguaje SQL es el poder hacer consultas de cualquier tipo a la base de datos en forma no procedural. La sentencia SELECT es muy poderosa y ampliamente rica en sus cláusulas y variantes permitiendo la capacidad de atender en poco tiempo a consultas complejas sobre la base de datos. Está en el especialista desarrollador de aplicaciones conocerlo a profundidad para explotar las bondades y virtudes.

Gracias a esta sentencia es que se pueden realizar todas las operaciones del Algebra Relacional, tratadas en la sección 3.2 y 3.3 del modulo III. En esta sección veremos la sintaxis de como se utiliza.

La sentencia SELECT, obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas.

La sintaxis completa de la instrucción SELECT es compleja, pero la voy resumir como sigue (los corchetes cuadrados indican que la cláusula no es obligatoria):

SELECT nombres de las columnas

[INTO nueva Tabla destino para resultados del select_]

FROM origenTabla

[WHERE condición de Búsqueda]

[GROUP BY nombres de columnas por la cual Agrupar]

[HAVING condiciónBúsqueda para Group By]

[ORDER BY nombre de columnas [ASC | DESC]]

También se puede unir estas sentencias con otras por el operador UNION entre consultas para combinar sus resultados en un sola tabla de resultados sin nombre.

Veamos cales son las funciones de cada una de las cláusulas de la sentencia SELECT.

La cláusula SELECT: Se usa para listar las columnas de las tablas que se desean ver en el resultado de una consulta. Además de las columnas se pueden listas columnas a calcular por el SQL cuando actué la sentencia. Esta cláusula no puede omitirse.

La cláusula FROM: Lista las tablas que deben ser analizadas en la evaluación de la expresión de la cláusula WHERE y de donde se listarán las columnas enunciadas en el SELECT. Esta cláusula no puede omitirse.

La cláusula WHERE: establece criterios de selección de ciertas filas en el resultado de la consulta gracias a las condiciones de búsqueda. Si no se requiere condiciones de búsqueda puede omitirse y el resultado de la consulta serán todas las filas de las tablas enunciadas en el FROM.

La cláusula GROUP BY: especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la base de datos, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria de resultados para cada grupo de los nombres de columnas enunciado en esta cláusula. En otras palabras, esta cláusula permitirá agrupar un conjunto de columnas con valores repetidos y utilizar las funciones de agregación sobre las columnas con valores no repetidas. Esta cláusula puede omitirse.

La cláusula HAVING: le dice al SQL que incluya sólo ciertos grupos producidos por la cláusula GROUP BY en los resultados de la consulta. Al igual que la cláusula WHERE, utiliza una condición de búsqueda para especificar los grupos deseados. La cláusula HAVING es la encargada de condicionar la selección de los grupos en base a los valores resultantes en las funciones agregadas utilizadas debidas que la cláusula WHERE condiciona solo para la selección de filas individuales. Esta cláusula puede omitirse.

La cláusula ORDER BY: permitirá establecer la columna o columnas sobre las cuales las filas resultantes de la consulta deberán ser ordenadas. Esta cláusula puede

Sentencia SELECT-FROM

El utilizar la sentencia SELECT, con estas dos cláusulas SELECT - FROM, muestra como resultado a todas las filas existentes en las tablas especificadas en el FROM.

Ejemplo # 1: Seleccionar todas las columnas y filas de la tabla CLIENTE

```
SELECT * FROM CLIENTE
```

Da como resultado un total de 8 filas con las 7 columnas que posee la tabla CLIENTE.

Ejemplo # 2: Seleccionar columnas: nomcli y e_mail de la tabla CLIENTE

```
SELECT NUMCLI, NOMCLI, E-MAIL FROM CLIENTE
```

Da como resultado un total de 8 filas con solo 3 columnas.

Ejemplo # 3: Este ejemplo selecciona las columnas y las muestra con el titulo especificado en AS, es decir con un alias. A NUMCLI lo muestra como NUMERO DE CLIENTE y NOMCLI lo muestra con el nombre especificado en el AS como NOMBRE DEL CLIENTE. Esto permite mostrar una columna con encabezados mas familiares a los usuarios finales.

```
SELECT NUMCLI AS 'NUMERO DE CLIENTE', NOMCLI AS
'NOMBRE DE CLIENTE' FROM CLIENTE
```

Da como resultado un total de 8 filas.

La cláusula AS puede omitirse y el resultado es el mismo.

```
SELECT NUMCLI 'NUMERO DE CLIENTE', NOMCLI 'NOMBRE DE
CLIENTE' FROM CLIENTE
```

Da como resultado un total de 8 filas.

SENTENCIA DE FILAS DUPLICADAS (DISTINCT)

Si una consulta incluye la llave primaria (pk) de una tabla en su lista de selección, entonces cada fila de resultados será única (ya que la llave primaria (pk) tiene un valor diferente en cada fila). Si no se incluye la llave primaria en los resultados, pueden producirse filas duplicadas. Veamos el siguiente ejemplo,

Ejemplo # 5: Seleccionar el código de artículos que han sido pedidos. Sin usar la palabra reservada DISTINCT.

El resultado tendría 18 filas y con códigos de productos repetidos.

```
SELECT     NUMART
FROM     DETALLE_PED
ORDER BY     NUMART
```

Ejemplo # 6: Seleccionar el código de artículos que han sido pedidos. Utilizando la palabra reservada DISTINCT.

El resultado contiene menos filas, 9 filas y con códigos de productos únicos, es decir no se repiten por las veces que fueron comprados como en el ejemplo #5.

```
SELECT    DISTINCT    NUMART
FROM     DETALLE_PED
ORDER BY     NUMART
```

Columnas Calculadas

Además de las columnas cuyos valores serán introducidos a la base de datos a través de la sentencia INSERT, una consulta SQL puede incluir en su cláusula SELECT columnas calculadas cuyo valor se calculan a partir de los valores de las otras columnas almacenadas en las tablas. Estas columnas, son especie de una columna virtual pues no existen físicamente en la tabla y sus valores calculados corresponden a los valores por fila.

Esta es una ventaja del SQL que evita que se tengan que diseñar columnas calculadas físicas en la base de datos trayendo perdida en almacenamiento físico y inconsistencia por falta de mantenimiento de la misma.

Ejemplo # 7: Determinar a los clientes con saldo.

Recuerde que el saldo se encuentra almacenado en tres columnas diferentes: SALD_0_30 + SALD_31_60 + SALD_61_90.

```
SELECT NUMCLI, NOMCLI,
(SALD_0_30 +SALD_31_60 + SALD_61_90) as "Saldo del Cliente"
FROM CLIENTE
WHERE (SALD_0_30 +SALD_31_60 + SALD_61_90) <> 0
```

Ejemplo # 8: Para determinar el monto de dinero que llevan acumulados en ventas por arriba de sus cuotas los vendedores, la consulta SELECT con columnas calculadas por vendedor es,

```
SELECT CODVEND, NOMVEND, APELLVEND, (VENTAS - CUOTA)
AS 'Monto por Arriba de Coutas'
FROM VENDEDOR
WHERE (VENTAS - CUOTA) > 0
```

Condiciones de Búsqueda (=, <>, >, <, >=, <=, BETWEEN, IN, LIKE, IS NULL, compuestas (AND, OR, NOT))

SQL usa las conectivas lógicas AND, OR y NOT en la cláusula WHERE.
Los operandos de las conectivas lógicas pueden ser expresiones que contengan los operadores de comparación <,<=,>, > =, = y <>. SQL permite usar los operadores de comparación para comparar cadenas y expresiones aritméticas, así como tipos especiales, tales como el tipo fecha.

Condiciones de Búsqueda (=, <>, >, <, >=, <=):

Ejemplo # 9: Seleccionar a los clientes cuyo número de cliente está contenido entre el rango 1309 y 1950.

```
SELECT NUMCLI AS 'NUMERO DE CLIENTE', NOMCLI AS
'NOMBRE DE CLIENTE'

FROM CLIENTE
WHERE NUMCLI >= 1309 AND NUMCLI <= 1950
```

Se obtiene el mismo resultado utilizando la palabra reservada BETWEEN. Este especifica que un valor sea menor o igual que un valor y mayor o igual que otro valor.

Condiciones de Búsqueda con BETWEEN:

Ejemplo # 10: Para seleccionar a los clientes cuyo número de cliente está contenido entre el rango 1309 y 1950, es:

```
SELECT NUMCLI , NOMCLI FROM CLIENTE WHERE NUMCLI
BETWEEN 1309 AND 1950
```

Ejemplo # 11: Para seleccionar los clientes con nombre entre H Y P, la sentencia SELECT es:

```
SELECT NUMCLI AS 'NUMERO DE CLIENTE', NOMCLI AS
'NOMBRE DE CLIENTE' FROM CLIENTE
WHERE (NOMCLI BETWEEN 'H' AND 'P')
```

Condiciones de Búsqueda Compuestas con AND, OR y NOT:

Ejemplo # 12: Se puede combinar las conectivas lógicas AND, OR or y NOT y filtrar mayor la selección en la consulta. Por ejemplo al seleccionar a los clientes cuyo número de cliente está contenido entre el rango 1309 y 1950 y que han comprado o incrementado su saldo dentro del mes en curso, es decir su saldo en mes corriente no es cero.

```
SELECT NUMCLI AS 'NUMERO DE CLIENTE', NOMCLI AS
'NOMBRE DE CLIENTE', SALD_0_30

FROM CLIENTE
WHERE NUMCLI BETWEEN 1309 AND 1950 AND NOT
SALD_0_30 = 0
```

Ejemplo # 13: Utilizando el operador OR podemos buscar un cliente cuyo código dudamos si es 824 o 842.

```
SELECT NOMCLI, NUMCLI FROM CLIENTE WHERE NUMCLI =
842 OR NUMCLI = 824
```

Ejemplo # 14: Para seleccionar los vendedores cuyas ventas es mayor que la couta que se espera deben vender y que sus ventas son superiores a B/8000.00.

```
SELECT CODVEND, NOMVEND, APELLVEND, VENTAS, CUOTA
FROM VENDEDOR
WHERE VENTAS > CUOTA AND VENTAS > 8000
```

Condiciones de Búsqueda con LIKE:

Ejemplo # 15: Seleccionar a todos los clientes que contengan la letra "H" dentro de su nombre.

```
SELECT NUMCLI AS 'NUMERO DE CLIENTE', NOMCLI AS
'NOMBRE DE CLIENTE'  FROM CLIENTE  WHERE  NOMCLI
LIKE    '%H%'
```

Condiciones de Búsqueda con la función SOUNDEX:

Se emplea en situaciones donde se tiene idea del sonido de un valor de columna pero que no se conoce su correcta escritura (es una función nueva incorporada en el SELECT).

Ejemplo # 16: Seleccionar a todos los clientes cuyo nombre suene parecido al especificado en la función SOUNDEX ( ).

```
SELECT NUMCLI, NOMCLI FROM CLIENTE WHERE
SOUNDEX (NOMCLI) = SOUNDEX ( 'DUIT' )
```

Condiciones de Búsqueda con IS NULL:

Los valores de NULL crean una lógica trivaluada para las condiciones de búsqueda en SQL. Para una fila determinada, el resultado de una condición de busqueda puede ser TRUE o FALSE, o puede ser NULL debido a que una de las columnas utilizadas en la evaluación de la condición de búsqueda contenga un valor NULL.

A veces es útil comprobar explícitamente los valores NULL en una condición de búsqueda y manejarlas directamente.

Ejemplo # 17: Consultar que filas de la tabla ARTICULO tienen valor NULL en algunas de sus columnas: DESCRIPCION, EXISTENCIA Y CATEGORIA_ART.

```
SELECT *
FROM ARTICULO
WHERE DESCRIPCION IS NULL OR EXISTENCIA IS NULL OR
CATEGORIA_ART IS NULL
```

Ordenación de resultados de Consulta (cláusula ORDER BY)

Al igual que las filas de una tabla en la base de datos las filas de los resultados de una consulta no están dispuestas en ningún orden particular. Existen situaciones en la que es necesario ver la información en un orden en especial, como en orden alfabetico (ASC, ascendente) u ver a las cifras de dinero listadas de mayor monto a menor (DESC, descendente). Se puede pedir a SQL que ordene los resultados de una consulta incluyendo la cláusula ORDER BY en la sentencia SELECT.

Ejemplo # 18: Para buscar la información de los vendedores por orden de su apellido o nombre, la sentencia select con la cláusula ORDER BY seria la siguiente:

```
SELECT *
FROM VENDEDOR
ORDER BY APELLVEND
```

Ejemplo # 19: Para determinar en una lista resultante del ejemplo #8, quien el nombre del vendedor con mayor el monto de dinero es conveniente ordenar por esta que llevan acumulados en ventas por arriba de sus cuotas los vendedores, la consulta SELECT con columnas calculadas por vendedor es,

```
SELECT CODVEND, NOMVEND, APELLVEND, (VENTAS - CUOTA)
AS 'Monto por Arriba de Coutas'
FROM VENDEDOR
WHERE (VENTAS - CUOTA) > 0

ORDER BY (VENTAS - CUOTA)  DESC
```

Consultas a Múltiples Tablas

Generalmente el poder de la sentencia SELECT se basa en su capacidad de poder en una sola sentencia consultar múltiples tablas simultáneamente. Esta operación también se le llama JOIN y es lo que en particular llamo un "pegue de tablas en forma horizontal" y ocurre gracias a que existen columnas de conexión ósea atributos de asociación comunes en las tablas que se unen en esta forma, vea figura:

Para que se puedan realizar consultas a múltiples tablas el requisito principal es que las tablas a reunirse en una consulta tengan columnas con valores o dominios comunes, es decir columna de conexión. Si reunimos a las tablas de la figura 4.4.a, de tal forma que la Tabla1 se reúna con la Tabla2 y la Tabla2 se reuniera con la Tabla3, el requisito indispensable sería: la Tabla1 debe tener una columna común en la Tabla2 y Tabla2 debe tener una columna de conexión en la Tabla3. Por lo general estas columnas comunes son: en una tabla la columna es la llave primaria y en la otra tabla la columna asociada es la llave foránea que referencia a la primaria. Para este tipo de consultas a múltiples tablas la cláusula FROM es la responsable de indicar cual será la(s) tabla(s) fuentes.

Existen dos formas de sintaxis permitidas para la escritura de la sentencia SELECT para la reunión de tablas, basada en la figura 4.4.a, estas formas son las siguientes:

FORMA 1:

```
SELECT  a1, a2, a5,  b1,  b2, b3, c1, c2, c3
```

FROM Tabla1, Tabla2, Tabla3

WHERE Tabla1.a1 = Tabla2.b2 AND Tabla2. b1 = Tabla3.c1

FORMA 2:

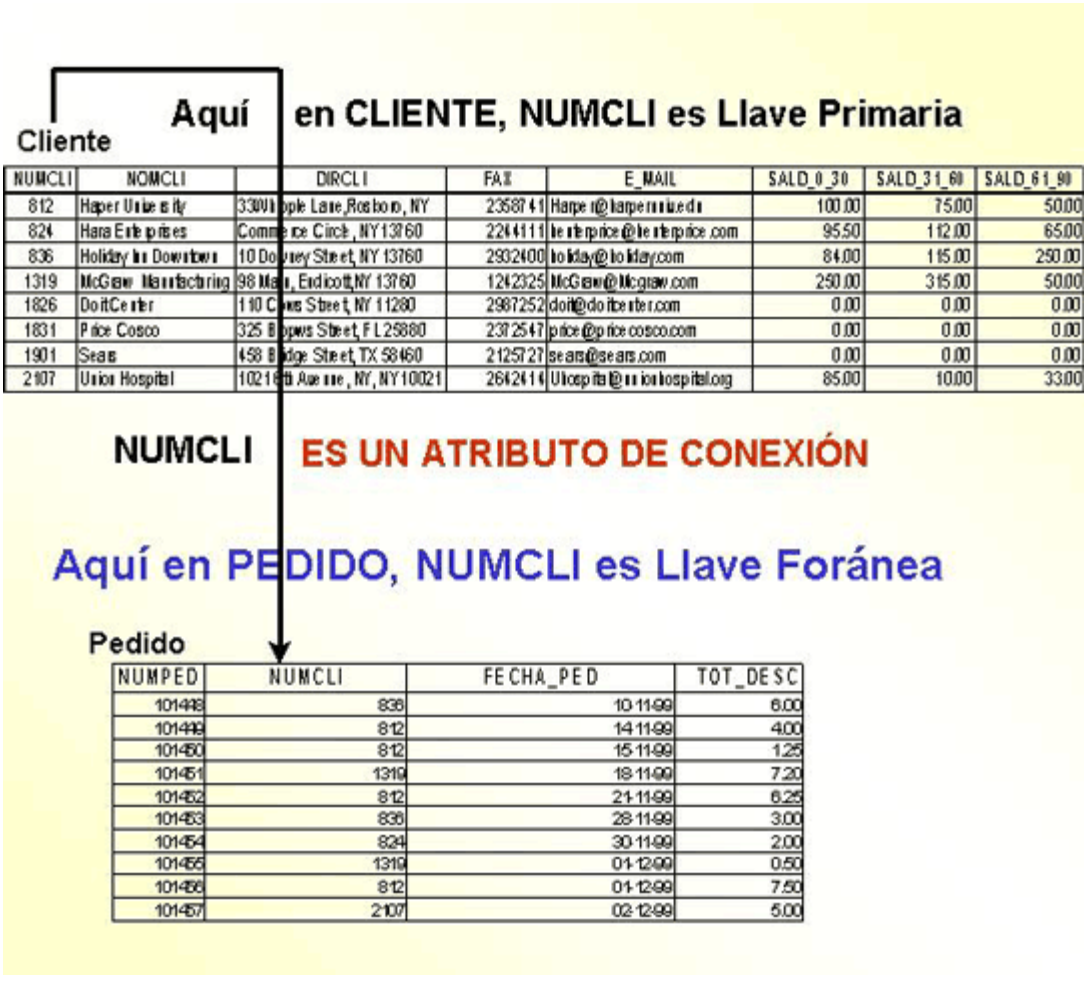
SELECT a1, a2, a5, b1, b2, b3, c1, c2, c3

FROM Tabla1 INNER JOIN Tabla2 ON Tabla1.a1 = Tabla2.b2

INNER JOIN Tabla3 Tabla2. b1 = Tabla3.c1

Note que las columnas comunes en ambas tablas han de calificarse con el nombre de la tabla que pertenecen para evitar errores de ambigüedad.

La Forma 2, tiene la ventaja de liberar a la cláusula WHERE y dejar esta para filtros específicos sobre las filas resultantes de la reunión de tablas.



NUMCLI	NOMCLI	DIRCLI	FAX	E_MAIL	SALD_0_30	SALD_31_60	SALD_61_90
812	Haper University	3300 Apple Lane,Roslono, NY	2358741	Harper@harperuniv.edu	100.00	75.00	50.00
824	Hara Enterprises	Commerce Circle, NY13760	2244111	hrenterprice@hrenterprice.com	95.50	112.00	65.00
836	Holiday Inn Downtown	10 Downey Street, NY 13760	2932400	holiday@holiday.com	84.00	115.00	250.00
1319	McGraw Manufacturing	98 Main, Endicott,NY 13760	1242325	McGraw@Mcgraw.com	250.00	315.00	50.00
1826	DoitCenter	110 Cows Street, NY 11280	2987252	doit@doitcenter.com	0.00	0.00	0.00
1831	Price Cosco	325 Bypws Street, FL25880	2372547	price@pricecosco.com	0.00	0.00	0.00
1901	Sears	458 Bridge Street, TX 58460	2125727	sears@sears.com	0.00	0.00	0.00
2107	Union Hospital	102 Fifth Avenue, NY, NY10021	2642414	Uhospital@unionhospital.org	85.00	10.00	33.00

NUMPED	NUMCLI	FECHA_PED	TOT_DESC
101448	836	10-11-99	6.00
101449	812	14-11-99	4.00
101450	812	15-11-99	1.25
101451	1319	18-11-99	7.20
101452	812	21-11-99	6.25
101453	836	28-11-99	3.00
101454	824	30-11-99	2.00
101455	1319	01-12-99	0.50
101456	812	01-12-99	7.50
101457	2107	02-12-99	5.00

Ejemplo #21: Si se desea unir la tabla de CLIENTES con PEDIDO (ver figura 4.4.b.), para ambas tablas, la columna de conexión es NUMCLI, la consulta SELECT que realiza esta reunión es la siguiente:

FORMA 1:

```
SELECT CLIENTE.NUMCLI, NOMCLI, NUMPED, FECHA_PED, TOT_DESC
FROM CLIENTE, PEDIDO
WHERE CLIENTE.NUMCLI = PEDIDO.NUMCLI
```

FORMA 2:

```
SELECT CLIENTE.NUMCLI, NOMCLI, NUMPED, FECHA_PED, TOT_DESC
FROM CLIENTE INNER JOIN PEDIDO
ON  CLIENTE.NUMCLI = PEDIDO.NUMCLI
```

Ejemplo # 24: Seleccionar todos los pedidos con su información completa, esto incluye: datos del cliente, datos del pedido, datos del detalle de la compra, datos del articulo de la compra y datos den vendedor que atendió el pedido. Esta consulta requiere que se realice la unión (join) de las cinco tablas: Cliente, Pedido, Detalle_Ped , Artículo y Vendedor, que son todas las tablas de la base de datos del **VENTAS.**

```
SELECT PEDIDO.NUMPED, CLIENTE.NUMCLI,NOMCLI,DIRCLI,
APELLVEND AS VENDEDOR, FECHA_PED,ARTICULO.NUMART,
DESCRIPCION,PRECIO,CANTIDAD,(PRECIO*CANTIDAD) AS TOTAL
FROM CLIENTE INNER JOIN PEDIDO ON CLIENTE.NUMCLI=PEDIDO.NUMCLI
INNER JOIN DETALLE_PED ON PEDIDO.NUMPED = DETALLE_PED.NUMPED
INNER JOIN ARTICULO ON DETALLE_PED.NUMART=ARTICULO.NUMART
INNER JOIN VENDEDOR ON PEDIDO.CODVEND = VENDEDOR.CODVEND
ORDER BY PEDIDO.NUMPED
```

RESULTADO:

NUMPED	NUMCLI	NOMCLI	DIRCLI	VENDEDOR	FECHA_PED	NUMART	DESCRIPCIO
101448	836	Holiday Inn Downtown	10 Downey Stre	Torres	10-Nov-99	A38	Velas Aromati
101448	836	Holiday Inn Downtown	10 Downey Stre	Torres	10-Nov-99	F251	Raqueta de Te
101448	836	Holiday Inn Downtown	10 Downey Stre	Torres	10-Nov-99	S247	Tornillos 3/4
101449	812	Haper University	33Whipple Lane	Ramírez	14-Nov-99	B14	Toallas
101450	812	Haper University	33Whipple Lane	Medina	15-Nov-99	C118	Sabanas
101451	1319	McGraw Manufacturing	98 Main, Endicot	Pinzón	18-Nov-99	C118	Sabanas
101451	1319	McGraw Manufacturing	98 Main, Endicot	Pinzón	18-Nov-99	D117	Tacos 3/4
101452	812	Haper University	33Whipple Lane	Martínez	21-Nov-99	A38	Velas Aromati
101452	812	Haper University	33Whipple Lane	Martínez	21-Nov-99	B16	Cobija
101452	812	Haper University	33Whipple Lane	Martínez	21-Nov-99	F251	Raqueta de Te
101453	836	Holiday Inn Downtown	10 Downey Stre	Prado	28-Nov-99	N38	Servilletas
101453	836	Holiday Inn Downtown	10 Downey Stre	Prado	28-Nov-99	T101	Manteles
101454	824	Hara Enterprises	Commerce Circl	Jímenez	30-Nov-99	B14	Toallas
101456	812	Haper University	33Whipple Lane	Prado	01-Dic-99	B14	Toallas
101456	812	Haper University	33Whipple Lane	Prado	01-Dic-99	B16	Cobija
101456	812	Haper University	33Whipple Lane	Prado	01-Dic-99	C118	Sabanas
101456	812	Haper University	33Whipple Lane	Prado	01-Dic-99	T101	Manteles
101457	2107	Union Hospital	1021 6th Avenu	Pinzón	02-Dic-99	C118	Sabanas

Consultas Resumen

Estas consultas resultan ventajosas para hacer búsquedas a nivel de los valores de las columnas a nivel de todas las filas. Por ejemplo si se desea saber en una tabla de 10,000 filas: el mayor saldo adeudado, la cuota mínima o máxima asignada a los vendedores, el tamaño promedio de pedidos, el total de clientes o vendedores (es decir total de filas).

Ejemplo # 27: Determinar en una misma consulta lo siguiente:

- Total de Vendedores. COUNT (NOMVEND)
- Monto Total de COUTAS que deben alcanzarse entre todos los vendedores. SUM(CUOTA).
- El monto promedio de cuotas asignado a un vendedor. AVG(COUTA)
- El valor de la cuota máxima asignado al un vendedor. MAX(COUTA)
- El valor de la cuota mínima asignado a un vendedor. MIN (COUTA)

SENTENCIA SQL:

```
USE VENTAS
SELECT COUNT (NOMVEND)AS "TOTAL VENDEDORES",
SUM(CUOTA)"MONTO DE CUOTAS",AVG(CUOTA) AS "CUOTA
PROMEDIO", MAX(CUOTA) "CUOTA MAX.", MIN(CUOTA)"CUOTA MIN."
FROM VENDEDOR
```

El resultado de esta consulta es una fila con los siguientes valores:

```
TOTAL VENDEDORES   MONTO DE CUOTAS   CUOTA PROMEDIO     CUOTA MAX.
CUOTA MIN.
---------------------------    ----------------------------    ----------------------------  --------- ----
-------   ---------------
                10                25341.00                          2534.10
8705.00                 200.00
```

Otros ejemplos:

Sentencia SUM:

```
SELECT SUM (cantidad) as 'Suma de las Cantidades'   FROM detalle_ped
```

Sentencia AVG:

```
SELECT AVG (cantidad) as 'Cantidad Promedio Pedida'  FROM detalle_ped
```

Sentencia MIN:

SELECT MIN (cantidad) as 'Cantidad Mínimo Pedida' FROM detalle_ped

Sentencia MAX:

SELECT MAX (cantidad) as 'Cantidad Máxima Pedida'
FROM detalle_ped

Sentencia COUNT

SELECT COUNT (*) as 'Total de Clientes' FROM CLIENTE

Sentencia DISTINCT

SELECT COUNT(DISTINCT numped) as 'Total de Pedidos Solicitados'
FROM detalle_ped

En esta ultima consulta hace que los números de pedidos en DETALLE_PED , que son repetidos, la cláusula DISTINCT cuente solo números de uno de los que se repiten.

CONCLUSIÓN

Este manual fue hecho con el propósito de ayudar a los estudiantes universitarios que tienen un deseo de aprender, más fuerte cada día.

También para ayudarme a mí mismo en la codificación de SQL.

Cualquier comentario o sugerencia (constructivas claro), me lo pueden hacer a mi correo electrónico:

alvaroegarcia@ubbi.com
alvaroegarcia@iespana.es

FECHA DE CREACIÓN:
Lunes 24 de noviembre de 2003

BIBLIOGRAFÍA

- ✓ Módulos 3 y 4 del curso de Sistemas de Información II Virtual de la **Universidad Tecnológica de Panamá,** (2003)

  Nota: tales módulos fueron creados por la ingeniera Tamara Batista