TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN
UTN-FRC

TUP

UTN
Facultad Regional Córdoba

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

LABORATORIO DE COMPUTACION II

Unidad Temática 1:
Sentencias de Manipulación de Datos

Teórico

Curso: 1 er Año. 2 ndo Cuatrimestre

2019

Índice

Sentencias de Manipulación de Datos

SQL

- **Lenguaje de Consulta Estructurado** (StructuredQueryLanguage) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

SQL Server

- Es un **Sistema de Gestión de Bases de Datos Relacionales** (SGBDR o RDBMS Relational Data Base Management System), diseñado para trabajar con grandes cantidades de información. Ofrece el soporte de información para aplicaciones Cliente/Servidor, las cuales mediante una interfaz y una red LAN los clientes acceden a los datos

TRANSACT -SQL

- Contiene los comandos y sentencias necesarios para administrar una instancia de SQL Server, crear y administrar sus objetos, y consultar, insertar, modificar, y eliminar datos de las tablas

Tipos de Sentencias

Las distintas sentencias que se pueden ejecutar son:

- De Definición de Datos (DDL Data Definition Language): se utilizan para crear, modificar y eliminar objetos de una base de datos. Algunas son: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX etc.
- De Manipulación de Datos (DML Data Manipulation Language): se utilizan para recuperar, agregar, modificar o eliminar datos de una base de datos. Por ejemplo: SELECT, INSERT, UPDATE, DELETE.
- De Control de Acceso (DCL Data Control Language): conceden o suprimen privilegios a usuarios. Por ejemplo: GRANT y REVOKE.
- De control de Transacciones (TCL Transaction Contro Languaje): finalizan o abortan la transacción actual: COMMIT, ROLLBACK.
- De Programación: DECLARE, OPEN, EXECUTE, DESCRIBE, etc.

En esta unidad vamos a ver la Sentencia de Manipulación de Datos de datos que son las siguienes:

SELECT

Para recuperar los datos de una o más tablas.

INSERT

Para insertar o cargar nuevos datos.

UPDATE

Para modificar datos existentes.

DELETE

Para eliminar datos existentes.

La sentencia SELECT es una sentencia de recuperación de datos ya que trae desde su almacenamiento los datos requeridos y arma una tabla de resultados en memoria, no realiza modificación alguna ni el los datos, como las otras tres, ni en su estructura como en el caso de las sentencias de definición de datos.

Insert, Update y Delete se llaman de actualización de datos ya que modifican los datos almacenados en el base de datos. Para hacer efectivos los cambios ejecutados con estas tres sentencias, es necesario ejecutar seguidamente, una sentencia de Control de Transacciones: COMMIT. Esto hace que el resto de los usuarios puedan percibir las modificaciones realizada por el usuario.

Actualización de Datos

Sentencia Insert

Cuya sintaxis es:

INSERT INTO Nombre_Tabla (lista_columnas) VALUES (lista_valores)

En donde:

Lista_columnas: son columnas de la tabla, en la que se quiere insertar datos.

lista_valores: son los valores que se van a insertar en las columnas antes especificadas. Los valores deben coincidir con el tipo de dato de la columna correspondiente, y debe existir tantos valores como columnas se especifiquen en la lista. Si una columna no tiene valor puede introducir un valor Null (si la definición de la columna lo permite) Los datos de tipo fecha y texto (varchar por ejemplo) van entre comillas simples y el separador de decimales es el punto. La coma se utiliza como separador de listas.

Ahora sí, ingresaremos a la base de datos los artículos que están en esta lista:

También es posible agregar múltiples filas a través del siguiente formato:

```
INSERT  INTO Nombre_Tabla1
SELECT lista_campos FROM Tabla2
```

De esta forma se pueden insertar registros en una tabla, obteniédolos de otra.

Sentencia Update

La sintaxis es:

```
UPDATE      Nombre_Tabla
SET         columna   = Nuevo Valor,
            columna2 = Nuevo_Valor2
[WHERE      condición]
```

En la cláusula SET se especifican todas las columnas a las que se les quiere cambiar su valor, separadas por comas.

En la cláusula WHERE se puede especificar un filtro para modificar solo aquellos registros que cumplan con la condición establecida.

Tener en cuenta que si la actualización de una fila no cumple con una restricción o regla, infringe la configuración de valores NULL o si el nuevo valor es de un tipo de datos incompatible, se cancela la instrucción, se devuelve un error y no se actualiza ningún registro.

Sentencia Delete

La sintaxis de DELETE es:

```
DELETE Nombre_tabla
 [WHERE Condición]
```

Tener en cuenta que si no se especifica la cláusula WHERE, se borran todas las filas de una tabla.

Recuperación de Datos

La Sentencia SELECT

Esta sentencia, que se utiliza para expresar consultas, es la más potente y compleja de las sentencias de SQL. La sentencia SELECT recupera datos de una base de datos y los devuelve en forma de tablas que no quedan guardadas en la base. La sintaxis completa es la siguiente:

```
SELECT [DISTINCT|ALL] lista_columnas
[INTO nueva_tabla]        Propio de SQL Server
FROM lista_tablas
[WHERE condición]
[GROUP BY columna, ...]
[HAVING condición]
[ORDER BY columna, ... [ASC|DESC]]
```

En donde:

- Las cláusulas que se encuentran entre [ ] son opcionales.
- SELECT lista_columnas: se especifica el nombre de la/s columna/s de las cuales se quiere mostrar la información. Esta lista recupera y muestra las columnas en el orden especificado. La lista de columnas separadas por comas se la suele denominar en la bibliografía de SQL como lista de selección.
- Los nombres de columnas se separan con comas.
- Se usa un asterisco en lugar de la lista de select para recuperar todas las columnas de la tabla.
- Se pueden combinar columnas utilizando el signo + (signo más)
- Se pueden renombrar las columnas, utilizando la palabra reservada AS, teniendo en cuenta que si el alias está compuesto de dos palabras debe encerrarse entre comillas simples.
- DISTINCT|ALL elimina filas duplicadas en el resultado de la consulta. Por defecto está definida la cláusula ALL
- INTO nueva_tabla Esta cláusula es propia de SQL Server. Define la creación de una nueva tabla a partir de la respuesta a la consulta especificada.
- FROM lista_tablas: se especifica el nombre de la/s tabla/s de las cuales se quiere obtener la información. Si hay más de una tabla se separan con comas.
- WHERE condición: Con frecuencia suele ser necesario aplicar algún tipo de condición que restringa el número de registros devuelto por una consulta
- GROUP BY lista_campos: Establece la lista de columna por las cuales se agrupará la información.
- HAVING condición: permite filtrar los grupos generados por GROUP BY.
- ORDER BY lista_campos: indica la lista de campos separados por comas, si hay más de uno, que debe utilizarse como criterio para ordenar los registros. La palabra reservada DESC indica que el orden debe invertirse. Así, por ejemplo, si tratamos con valores numéricos, el orden por defecto que se aplica es de menor a mayor. Si se utiliza la palabra reservada DESC entonces se invierte dicho orden, colocándose los registros de mayor a menor.

Alias

Si prestamos atención al encabezado de cada columna de los ejemplos anteriores vemos que aparece en nombre del campo origen del dato o bien columna sin nombre.

Podemos agregarle un alias a los campos que aparecerán como encabezados de columnas:

```
select cod_cliente Código, ape_cliente 'Apellido del cliente', nro_tel Teléfono
from clientes
```

El alias va entre comillas salvo que no contenga caracteres especiales (como espacios, -, *, /, +, ?, =, etc.)

Columnas Calculadas

Además de las columnas cuyos valores provienen directamente de los campos de las tablas de la base de datos, se pueden incluir columnas calculadas cuyos valores surgen a partir de resolver expresiones.

La siguiente consulta calcula el subtotal de las ventas:

```
select nro_factura, cod_articulo, pre_unitario, cantidad, pre_unitario*cantidad Subtotal
from detalle_facturas
```

Si se quiere mostrar los datos de los clientes donde aparezca el apellido y nombre en la misma columna:

```
select cod_cliente,ape_cliente+' '+nom_cliente Cliente
from clientes
```

Distinct

En algunos casos las consultas pueden dar como resultados filas duplicadas en el resultado de la consulta es decir dos o más filas exactamente iguales.

Para eliminar las filas duplicadas utilizamos el predicado DISTINCT con lo que cada código de artículo con su precio de venta aparecería una sola vez, entonces la consulta quedaría:

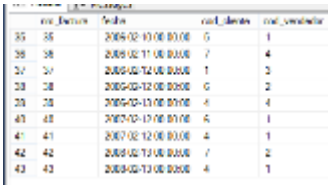
```
select distinct cod_articulo,pre_unitario

from detalle_facturas
```

Cláusula Where

Se utiliza si solo se quiere seleccionar solo una parte de las filas de una tabla, las que cumplan con la condición especificada en esta cláusula. Conceptualmente SQL recorre cada una de las filas de la tabla y aplica la condición. Esta fila se incluye en el resultado de la consulta si la condición es true (verdadero) de lo contrario se excluye. Se pueden resumir las condiciones de búsqueda en cinco diferentes:

Test de comparación:

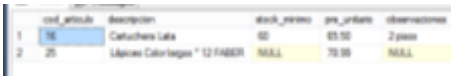
=, < >, <, >, <=, >= SQL calcula y compara los valores de dos expresiones por cada fila de datos como puede ser un campo con una constante o expresiones más complejas. Por ejemplo listar las facturas emitidas antes del 10/7/2008



```
Select *

from facturas

Where fecha < '10/07/2008'
```

Test de rango: BETWEEN .. AND ..

Comprueba si un valor de dato se encuentra entre dos valores especificados. Implica el uso de tres expresiones SQL. La primera define el valor a comprobar; la segunda y tercera definen los extremos del rango. Los tipos de datos de las tres expresiones deben ser comparables. Un ejemplo podría ser un listado de artículos cuyo precio esté entre 50 y 100 pesos:



```
Select *

from articulos

Where pre_unitario between 50 and 100
```

Test de pertenencia al conjunto: IN

Examina si un valor de dato coincide con uno de una lista de valores objetivos. Por ejemplo listar los clientes cuyo código sean los siguientes: 1,3, 7, 8 y 12

```
Select *

from clientes

Where cod_cliente in (1,3,7,8,12)
```

Test de correspondencia con patrón: LIKE

Comprueba si el valor de dato de una columna se ajusta a un patrón especificado. El patrón es una cadena que puede incluir uno o más caracteres comodines:

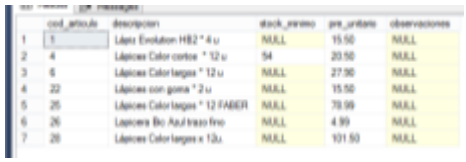
% : reemplaza a uno o más caracteres.

_ (guión bajo): reemplaza a un solo caracter.

[ ]: reemplaza a cualquier caracter que se encuentre entre el rango [a-f] o bien en el conjunto [abcdef].

^ : reemplaza a cualquier caracter que NO se encuentre entre el rango [a-f] o bien en el conjunto [abcdef].

Si se quiere listar todos los artículos que comiencen con “L”



```
Select *
from articulos
Where descripcion like 'L%'
```

La cláusula where de todos los que terminen con N sería: …where descripcion like ‘%N’

La cláusula where de todos los que contengan lápiz sería: …where descripcion like ‘%lapiz%’

De los que comiencen con letras que van de la l a la m: where descripcion like ‘[l-m]%’

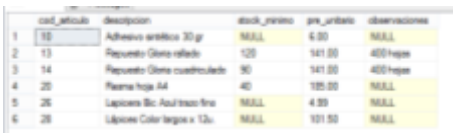
Test de valor nulo: IS NULL

Comprueba la existencia de valores Null en una condición de búsqueda. Por ejemplo listar los artículos para los que no existan observaciones

```
Select *
from articulos
Where observaciones is null
```

NOT

se puede utilizar para seleccionar filas en donde la condición de búsqueda es falsa. Por ejemplo listar los artículos cuyo precio no esté entre 10 y 100



```
Select *
from articulos
Where pre_unitario not between 10 and 100
```

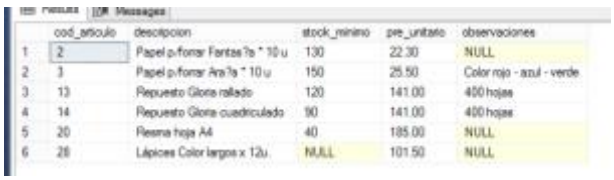
Condiciones de búsqueda compuestas

Utilizando las reglas de la lógica, se pueden combinar estas condiciones de búsqueda simples para formar otras más complejas utilizando los operadores lógicos OR y AND

OR

Se utiliza para combinar dos condiciones de búsqueda cuando una o la otra o ambas deban ser ciertas:

```
Select *
from articulos
Where stock_minimo > 100 or pre_unitario > 80
```



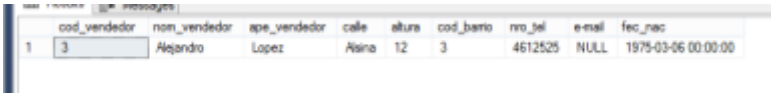
	cod_articulo	descripcion	stock_minimo	pre_unitario	observaciones
1	2	Papel p/forrar Fantas?a * 10 u	130	22.30	NULL
2	3	Papel p/forrar Ara?a * 10 u	150	25.50	Color rojo - azul - verde
3	13	Repuesto Gloria rallado	120	141.00	400 hojas
4	14	Repuesto Gloria cuadriculado	90	141.00	400 hojas
5	20	Resma hoja A4	40	185.00	NULL
6	28	Lápices Color largos x 12u	NULL	101.50	NULL

En el ejemplo se listarán los registros cuyo stock minimo sea mayor a 100 o cuyo precio mayor a 80. Preste atención al 1er. registro, cumple con la condición del stock mínimo mayor a 100 pero no cumple con la del precio unitario mayor a 80; el registro 3 cumple con ambas condiciones y el registro cuatro solo cumple la condición del precio unitario.

AND

Se utiliza para combinar dos condiciones de búsqueda que deban ser ciertas simultáneamente, como es el caso del listado de vendedores cuyo nombre comience con A y nacidos antes de 1980:

```
Select *
from vendedores
Where nom_vendedor like 'A%' and fec_nac < '1/1/1980'
```



	cod_vendedor	nom_vendedor	ape_vendedor	calle	altura	cod_barrio	nro_tel	e-mail	fec_nac
1	3	Alejandro	Lopez	Alsina	12	3	4612525	NULL	1975-03-06 00:00:00

En este caso solo se muestra el registro que cumple con ambas condiciones

Cláusula ORDER BY

Las filas de los resultados de una consulta no están dispuestas en ningún orden particular. Se puede pedir a SQL que ordene los resultados de una consulta incluyendo la cláusula Order By en la sentencia Select. Por ejemplo si se quiere listar los artículos ordenados por precio y descripción:

	cod_articulo	descripcion	stock_minimo	pre_unitario	observaciones
1	20	Resma hoja A4	40	185.00	NULL
2	13	Repuesto Gloria rallado	120	141.00	400 hojas
3	14	Repuesto Gloria cuadriculado	50	141.00	400 hojas
4	28	Lápices Color largos x 12u.	NULL	101.50	NULL
5	25	Lápices Color largos * 12 FABER	NULL	78.99	NULL
6	16	Cartuchera Lata	60	65.50	2 piece
7	21	Portaminas	NULL	48.90	NULL
8	15	Conjunto Geométrico	NULL	35.50	Regla - escuadra - transportador
9	27	Cuaderno tapa dura rayado	20	32.99	NULL
10	17	Cartuchera de Tela	55	32.60	NULL
11	18	Correctores Bic Lápiz * 1u	NULL	31.40	NULL
12	24	Goma para lapiceras * 10u	NULL	29.50	NULL
13	6	Lápices Color largos * 12 u.	NULL	27.80	NULL

Por omisión SQL ordena los datos en secuencia ascendente. Para ordenar en secuencia descendente se incluye la palabra DESC en la especificación de la ordenación a la derecha de cada columna que se quiera ordenar en forma descendente como ordenar los artículos en por precio en forma descendente

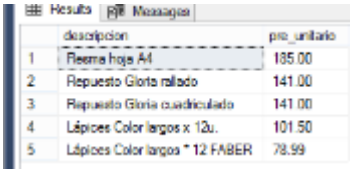
```
Select *
from articulos
order by pre_unitario desc
```

La cláusula Order by admite que en lugar de usar el nombre de columna se pueda utilizar también el número de columna o el alias de la misma.

Select top

Si se quiere limitar la cantidad de filas devueltas por la sentencia select, se puede utilizar combinada con Top seguido por el número de filas. Por ejemplo: se quiere listar los 5 artículos más caros, para ello emitimos el listado de los articulos ordenados por precio en forma descendente (de mayor a menor) y mostramos los 5 de más arriba:

```
select top 5 descripcion, pre_unitario
from articulos
order by 2 desc
```



Results Messages

	descripcion	pre_unitario
1	Resma hoja A4	185.00
2	Repuesto Gloria rallado	141.00
3	Repuesto Gloria cuadriculado	141.00
4	Lápices Color largos x 12u.	101.50
5	Lápices Color largos * 12 FABER	78.99

Comentarios

Para aclarar algunas instrucciones, en ocasiones, necesitamos agregar comentarios. Es posible ingresar comentarios en la línea de comandos, es decir, un texto que no se ejecuta; para ello se emplean dos guiones medios (--) al comienzo de la línea. Todo lo que está luego de los guiones (hacia la derecha) no se ejecuta.

```
select * from articulos --mostramos los registros de articulos;
```

Para agregar varias líneas de comentarios, se coloca una barra seguida de un asterisco (/*) al comienzo del bloque de comentario y al finalizarlo, un asterisco seguido de una barra (*/); todo lo que está entre los símbolos "/*" y "*/" no se ejecuta.

```
select nom_vendedor, ape_vendedor
 /*mostramos nombre de vendedores y
 su apellido
*/
 from vendedores;
```

Funciones Incorporadas

Funciones para el manejo de cadenas

Microsoft SQL Server tiene algunas funciones para trabajar con cadenas de caracteres. Estas son algunas:

- substring(cadena,inicio,longitud): devuelve una parte de la cadena especificada como primer argumento, empezando desde la posición especificada por el segundo argumento y de tantos caracteres de longitud como indica el tercer argumento. Ejemplo:

```
select substring('Buenas tardes',8,6) --retorna "tardes".
```

- str(numero,longitud,cantidaddecimales): convierte números a caracteres; el primer parámetro indica el valor numérico a convertir, el segundo la longitud del resultado y el tercero, la cantidad de decimales. Ejemplo: se convierte el valor numérico "123.456" a cadena, 7 de longitud y 3 decimales:

```
select str(123.456,7,3);
```

- stuff(cadena1,inicio,cantidad,cadena2): inserta la cadena enviada como cuarto argumento, en la posición indicada en el segundo argumento, reemplazando la cantidad de caracteres indicada por el tercer argumento en la cadena que es primer parámetro. Ejemplo:

```
select stuff('abcde',3,2,'opqrs') --retorna "abopqrse".
```

- len(cadena): retorna la longitud de la cadena enviada como argumento.

```
select len('abcde') --retorna 5.
```

- char(x): retorna un caracter en código ASCII del entero enviado como argumento.

- left(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la izquierda, primer caracter. Ejemplo:

```
select left('buenos dias',8)  --retorna "buenos d".
```

- right(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la derecha, último caracter. Ejemplo:

```
select right('buenos dias',8); retorna "nos dias".
```

- lower(cadena): retornan la cadena con todos los caracteres en minúsculas. lower significa reducir en inglés. Ejemplo:

```
select lower('HOLA ESTUDIAnte'); retorna "hola estudiante".
```

- upper(cadena): retornan la cadena con todos los caracteres en mayúsculas. Ejemplo:

```
select upper('HOLA ESTUDIAnte'); retorna "HOLA ESTUDIANTE"
```

- ltrim(cadena): retorna la cadena con los espacios de la izquierda eliminados. Ejemplo:

select ltrim(' Hola ')); retorna "Hola ".

- rtrim(cadena): retorna la cadena con los espacios de la derecha eliminados. Ejemplo:

select rtrim(' Hola '); retorna " Hola".

- replace(cadena,cadenareemplazo,cadenareemplazar): retorna la cadena con todas las ocurrencias de la subcadena reemplazo por la subcadena a reemplazar. Ejemplo:

select replace('xxx.esmiweb.com','x','w'); retorna www.esmiweb.com

- reverse(cadena): devuelve la cadena invirtiendo el orden de los caracteres.

- patindex(patron,cadena): devuelve la posición de comienzo (de la primera ocurrencia) del patrón especificado en la cadena enviada como segundo argumento. Si no la encuentra retorna 0. Ejemplos:

select patindex('%Luis%', 'Jorge Luis Borges'); retorna 7.

- charindex(subcadena,cadena,inicio): devuelve la posición donde comienza la subcadena en la cadena, comenzando la búsqueda desde la posición indicada por "inicio". Si no la encuentra, retorna 0. Ejemplo:

select charindex('or','Jorge Luis Borges',5); retorna 13.

- replicate(cadena,cantidad): repite una cadena la cantidad de veces especificada.

- space(cantidad): retorna una cadena de espacios de longitud indicada por "cantidad", que debe ser un valor positivo.

Por ejemplo se quiere mostrar el nombre del vendedor seguido por el apellido en mayúscula separados por 10 espacios todo en una misma columna

select nom_vendedor + SPACE(10) + UPPER(ape_vendedor)

from vendedores


Results Messages

	(No column name)
1	Martín CARRIZO
2	Mariela LEDESMA
3	Alejandro LOPEZ
4	Marcelo MIRANDA
5	Gabriel MONTI
6	Susana JUAREZ
7	Ana ORTEGA
8	Juan MONTI
9	Ana ORTEGA

Funciones matemáticas

Las funciones matemáticas realizan operaciones con expresiones numéricas y retornan un resultado, operan con tipos de datos numéricos.

- abs(x): retorna el valor absoluto del argumento "x

- ceiling(x): redondea hacia arriba el argumento "x". Ejemplo:
select ceiling(12.34); retorna 13.

- floor(x): redondea hacia abajo el argumento "x". Ejemplo:

select floor(12.34); retorna 12.

- %: %: devuelve el resto de una división.

- power(x,y): retorna el valor de "x" elevado a la "y" potencia.

- round(numero,longitud): retorna un número redondeado a la longitud especificada. "longitud" debe ser tinyint, smallint o int. Ejemplo:

select round(123.456,1); retorna "123.400", es decir, redondea desde el primer decimal.

- sign(x): si el argumento es un valor positivo devuelve 1;-1 si es negativo y si es 0, 0.
- square(x): retorna el cuadrado del argumento.
- srqt(x): devuelve la raíz cuadrada del valor enviado como argumento.

Funciones para el uso de fechas y horas

Microsoft SQL Server ofrece algunas funciones para trabajar con fechas y horas. Estas son algunas:

- getdate(): retorna la fecha y hora actuales.
- datepart(partedefecha,fecha): retorna la parte específica de una fecha, el año, trimestre, día, hora, etc. Los valores para "partedefecha" pueden ser: year (año), quarter (cuarto), month (mes), day (dia), week (semana), hour (hora), minute (minuto), second (segundo) y millisecond (milisegundo).
- datename(partedefecha,fecha): retorna el nombre de una parte específica de una fecha. Los valores para "partedefecha" pueden ser los mismos que se explicaron anteriormente.
- dateadd(partedelafecha,numero,fecha): agrega un intervalo a la fecha especificada, es decir, retorna una fecha adicionando a la fecha enviada como tercer argumento, el intervalo de tiempo indicado por el primer parámetro, tantas veces como lo indica el segundo parámetro. Los valores para el primer argumento pueden ser: year (año), quarter (cuarto), month (mes), day (dia), week (semana), hour (hora), minute (minuto), second (segundo) y millisecond (milisegundo). Ejemplo:

select dateadd(day,3,'1980/11/02'); retorna "1980/11/05", agrega 3 días.

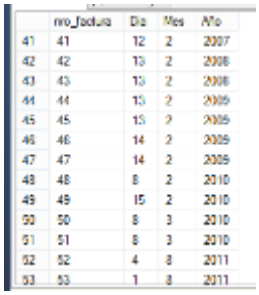
- datediff(partedelafecha,fecha1,fecha2): calcula el intervalo de tiempo, según el primer argumento entre las 2 fechas. Ejemplo:

select datediff (day,'2005/10/28','2006/10/28'); retorna 365 (días).

- day(fecha): retorna el día de la fecha especificada.
- month(fecha): retorna el mes de la fecha especificada.
- year(fecha): retorna el año de la fecha especificada.

Por ejemplo: Listar el día, mes y año en columnas separadas de todas las facturas

```
select nro_factura,day(fecha) Dia, month(fecha) Mes,year(fecha) Año
 from facturas
```



	nro_factura	Dia	Mes	Año
41	41	12	2	2007
42	42	13	2	2008
43	43	13	2	2008
44	44	13	2	2009
45	45	13	2	2009
46	46	14	2	2009
47	47	14	2	2009
48	48	8	2	2010
49	49	15	2	2010
50	50	8	3	2010
51	51	8	3	2010
52	52	4	8	2011
53	53	1	8	2011

Composiciones Simples

El proceso de formar parejas de filas haciendo coincidir los contenidos de las columnas relacionadas se denomina componer (joining) las tablas. La tabla resultante (que contiene datos de las dos tablas originales) se denomina una composición entre las dos tablas. Una composición basada en una coincidencia exacta entre dos columnas se denomina más precisamente una equicomposición.

Para realizar en SQL composiciones multitabla se puede utilizar la sentencia SELECT con una condición de búsqueda que especifique la comparación de columnas de tablas diferentes; previamente la cláusula FROM lista las dos tablas intervinientes.

Entonces la sentencia SELECT, sería la siguiente:

```
select ape_vendedor+' '+nom_vendedor Vendedor, barrio
from vendedores, barrios
where vendedores.cod_barrio=barrios.cod_barrio
```

Results | Messages

	Vendedor	barrio
1	Carrizo Martín	ALTO ALBERDI
2	Ledesma Mariela	GENERAL PAZ
3	Lopez Alejandro	OBSERVATORIO
4	Miranda Marcelo	CENTRO
5	Monti Gabriel	JARDIN
6	Juarez Susana	SAN VICENTE
7	Ortega Ana	ALTO ALBERDI
8	Monti Juan	GENERAL PAZ

Genera resultados solo para los pares de filas en los que el número de barrio (cod_barrio) de la tabla barrios coincide con el número de barrio de la tabla vendedores.

Entre la tabla barrios y vendedores existe una relación de uno a varios entonces se ha especificado en la condición de búsqueda que compare la clave foránea y la clave primaria de ambas tablas.

Hay que tener en cuenta que si un campo tiene el mismo nombre en dos tablas diferentes se debe anteponer al mismo el nombre de la tabla a la cual pertenece separado por un punto.

Podemos darle alias a las tablas en el FROM para evitar escribir tantas veces el nombre de la misma teniendo en cuenta que cada alias en cada consulta debe ser único. En el ejemplo anterior le vamos a dar un aliar a vendedores “v” y un alias a barrios “b”

```
select ape_vendedor+' '+nom_vendedor Vendedor, barrio
from vendedores v, barrios b
where v.cod_barrio=b.cod_barrio
```

SQL puede combinar tres o más tablas utilizando las mismas técnicas básicas utilizadas para las consultas de dos tablas. La siguiente consulta lista los artículos facturados con sus respectivas facturas

```
Select f.nro_factura,fecha,descripcion,cantidad,d.pre_unitario,
       cantidad*d.pre_unitario Subtotal
  from  facturas f, detalle_facturas d, articulos a
  where f.nro_factura = d.nro_factura
    and a.cod_articulo = d.cod_articulo
```

Composiciones con criterios de selección de fila.

La condición de búsqueda que especifica las columnas de emparejamiento en una consulta multitabla puede combinarse con otras condiciones de búsqueda para restringir aún más los contenidos de los resultados. Por ejemplo si se quiere listar las facturas del mes de mayo de 2015 con sus clientes:

```
Select f.nro_factura,fecha,ape_cliente+' '+nom_cliente Cliente
  from facturas f, clientes c
  where f.cod_cliente=c.cod_cliente
    and month(fecha)=5 and year(fecha)=2015
```

100 % | Results | Messages

	nro_factura	fecha	Cliente
1	149	2015-05-12 00:00:00	Abarca Héctor
2	150	2015-05-15 00:00:00	Abarca Héctor
3	151	2015-05-26 00:00:00	Paez Roque
4	181	2015-05-10 00:00:00	Morales Santiago

Multiplicación de tablas.

Una composición es un caso especial de una combinación más general de datos procedentes de dos tablas, conocida como el producto cartesiano de dos tablas. El producto de dos tablas es otra tabla que consta de todos los pares posibles de filas de las

dos tablas. Las columnas de la tabla producto son todas las columnas de la primera tabla, seguidas de todas las columnas de la segunda tabla. Si se especifica una consulta de dos tablas sin una cláusula WHERE, SQL produce el producto de las dos tablas como resultado. El siguiente ejemplo muestra todas las posibles de vendedores y barrios.

```
select * from vendedores,barrios
```

100 %

Results | Messages

	cod_vendedor	nom_vendedor	ape_vendedor	calle	altura	cod_barrio	nro_tel	e-mail	fec_nac	cod_barrio	barrio
1	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	1	CENTRO
2	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	2	ALTO ALBERDI
3	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	3	OBSERVATORIO
4	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	4	JARDIN
5	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	5	GENERAL PAZ
6	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	6	PUEYRREDON
7	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	7	PARQUE HORIZON...
8	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	8	SAN MARTIN
9	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	9	SAN VICENTE
10	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	10	JUNIOR
11	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	11	MAIPU
12	1	Martín	Carrizo	San Lorenzo	369	2	NULL	mcarrizo@latinmail....	NULL	12	PANAMERICANO
13	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	1	CENTRO
14	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	2	ALTO ALBERDI
15	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	3	OBSERVATORIO
16	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	4	JARDIN
17	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	5	GENERAL PAZ
18	2	Mariela	Ledesma	Chachapo...	1560	5	4526060	NULL	1979-02-22 ...	6	PUEYRREDON

Bibliografía

Groff, J. & Weinberg, P. (1991). Aplique SQL. Madrid. Editorial Mc Graw Hill

Microsoft (2019) Queries. Disponible en: https://docs.microsoft.com/es-es/sql/t-sql/queries/queries?view=sql-server-2017

Opel, A. & Sheldon, R. (2010). Fundamentos de SQL. Madrid. Editorial Mc Graw Hill

Tutoriales Ya. (2015) Sql Server Ya. Recuperado de: www.sqlserverya.com.ar