



Tecnicatura Universitaria
en Programación

LABORATORIO DE COMPUTACIÓN I

Unidad Temática N°2:
Sentencias de Manipulación de Datos

Material Teórico
1° Año – 1° Cuatrimestre



Índice

SENTENCIAS DE ACTUALIZACIÓN DE DATOS.....	2
Sentencia INSERT.....	2
Sentencia UPDATE	3
Sentencia DELETE.....	4
RECUPERACIÓN DE DATOS	5
Consultas.....	5
Sentencia SELECT	5
La lista de selección: Cláusula SELECT.....	6
Ordenamiento de los resultados de una consulta.....	1
Condiciones de búsqueda en el WHERE	3
Condiciones de búsqueda compuestas	6
Consultas compuestas	7
Combinación de Tablas	7
FUNCIONES INCORPORADAS	10
Funciones para el manejo de cadenas	10
Funciones matemáticas	12
Funciones para el uso de fechas y horas	13
OTRAS FORMAS DE COMPOSICIÓN: INNER, LEFT Y RIGHT JOIN	15
Combinación externa izquierda: left join	16
Combinación externa derecha: right join	17
BIBLIOGRAFÍA	19

SENTENCIAS DE ACTUALIZACIÓN DE DATOS

Con las Sentencias de Manipulación de Datos se puede recuperar datos de las tablas de una base de datos, insertar nuevos registros, modificar o eliminar registros existentes.

Las sentencias son:

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.

Para hacer efectivos los cambios ejecutados con las sentencias Insert, Update y Delete, 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.

Sentencia INSERT

Se utiliza la sentencia INSERT para agregar nuevos registros de datos a una tabla de una base de datos.

```
INSERT INTO Nombre_Tabla (lista_columnas)
VALUES (lista_valores)
```

En donde:

- **Nombre_tabla:** es la tabla de la base de datos donde se quieren insertar nuevos registros
- **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 deben 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 alfanumérico (varchar por ejemplo) van entre comillas simples y el separador de decimales es el punto. La coma se utiliza como separador de listas.

Por ejemplo: Dadas las tablas del diagrama y teniendo previamente registros en la tabla tipos_dni, insertar un nuevo registro en la tabla personas

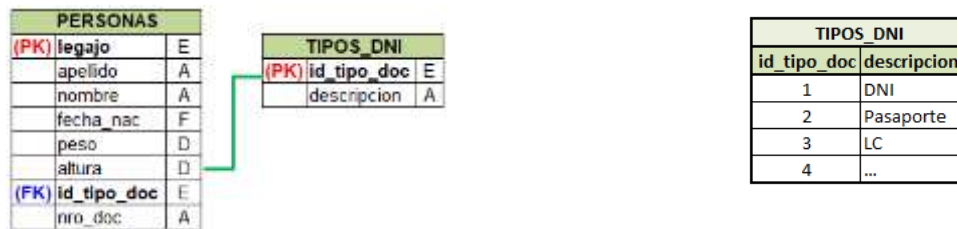


Ilustración 1 y 2: Elaboración propia.

```
insert into personas (legajo,apellido,peso,id_tipo_doc,fecha_nac)
values (2154, 'López',89.5,1, '10/05/1997')
```

Se observa que la lista de columnas o campos y la lista de valores se escribe separadas por comas. Cada valor se corresponde en el orden a cada campo.

Es obligatorio ingresar datos en legajo ya que es clave primaria que debe ser no nulo y un valor no repetido y en el apellido ya que en la unidad 1 se vio en la creación de la tabla personas y que se definió como “not null”.

Además el valor del id_tipo_doc que corresponde a la 4ta. Columna, debe estar ingresado antes en la tabla tipos_dni.

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, obteniéndolos de otra.

Tener en cuenta que, si el valor que intenta agregar a una de las columnas no cumple con alguno de las restricciones establecidas para el campo, la operación no se ejecutará.

Sentencia UPDATE

Esta sentencia nos permite modificar los datos de las columnas de las tablas. La sintaxis es:

```
UPDATE Nombre_Tabla
SET     columna1 = Nuevo_Valor1 [,
        columna2 = Nuevo_Valor2]
[WHERE condición]
```

En donde:

- **Nombre_tabla:** es la tabla a la que se quieren actualizar los datos
- **SET** se especifican todas las columnas a las que se les quiere cambiar su valor y el nuevo valor , separadas por comas.
- **WHERE** se puede especificar un filtro para modificar solo aquellos registros que cumplan con la condición establecida.

Por ejemplo: actualizar el peso y la altura de la persona ingresada en el ejemplo anterior con cuyo legajo es 2154

```
update libros
set    peso = 85.5,
        altura = 1.78
where legajo = 2154
```

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 instrucción DELETE elimina una o más filas de una tabla. 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.

Por ejemplo: eliminar el registro de la tabla personas cuyo legajo es 3210

```
delete personas
where legajo = 3210
```

RECUPERACIÓN DE DATOS

Consultas

Una Consulta es un conjunto de instrucciones, que permiten ver o interactuar con los datos contenidos en las tablas de una base de datos. Se recurre a ellas cuando, por ejemplo, se quiere obtener salidas de información de tablas de una base de datos obteniendo un subconjunto de registros que pertenecen a una o varias tablas o bien, se quiere ver, modificar o eliminar los datos de una tabla (o varias) que cumplan con determinadas condiciones, de tal forma que se pueda previsualizar y comprobar si realmente son éstas las que se quiere operar.

Para trabajar con los datos de una base de datos, se debe utilizar un conjunto de comandos e instrucciones (lenguaje) definidos por el software del DBMS. En las bases de datos relacionales se pueden utilizar varios lenguajes, de los que SQL es el más común. El American National Standards Institute (ANSI) y la International Standards Organization (ISO) definen estándares de software, incluidos los estándares para el lenguaje SQL.

Probablemente consultar los datos, es una de las tareas que se llevan a cabo con mayor frecuencia. Cuando existen varias tablas y las relaciones entre ellas son complejas, las consultas pueden llegar a ser francamente complicadas, tardando en ejecutarse una cantidad de tiempo nada despreciable.

Esta es precisamente otra de las cuestiones que deben afrontarse, ya que no se trata únicamente de obtener los datos deseados, sino de hacerlo además en un tiempo razonable, optimizando el tiempo de respuesta. El lenguaje SQL es muy flexible y permite realizar una misma consulta de muy diversas formas.

Sentencia SELECT

La sentencia SELECT se utiliza para obtener un conjunto de registros que puede proceder de una o más tablas.

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:

La sintaxis de la sentencia completa es:

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

Se explicará en detalle cada una de las cláusulas de la sentencia SELECT a continuación

La lista de selección: Cláusula SELECT

lista_columnas: se especifica el nombre de la/s columna/s de las cuales se quiere mostrar la información separadas por comas. Esta lista recupera y muestra las columnas en el orden especificado.

Por ejemplo: Teniendo las tablas del personas y tipos_dni, se quiere listar legajo, apellido, nombre y fecha de nacimiento desde la tabla personas



Ilustración 1: Elaboración propia.

```
select legajo, apellido, nombre, fecha_nac
from personas
```

La lista de columnas en la lista de selección va separada por comas, las columnas son los nombres de los campos que pertenecen a la tabla enunciada en el from. El resultado de esta consulta va a ser un listado con cuatro columnas.

legajo	apellido	nombre	fecha_nac
100	Rodríguez	Juan	1/5/1996
200	Pérez	Pedro	3/8/1984
300	Pérez	Ana	12/10/2010
400	Álvarez	Juana	3/9/1977
500	Sanchez	Juan	5/8/2002

Ilustración 2: Elaboración propia.

Uso del *: Para emitir un listado de todos los campos de la/s tabla/s del from se incluye el * en la lista de selección. En el ejemplo se mostrarían todos los campos de la tabla personal

```
select * from personas
```

Se pueden crear columnas calculadas creando expresiones con operaciones matemáticas (+;-;*/ para sumar, restar, multiplicar y dividir) y funciones. También concatenar caracteres utilizando el signo más: '+'

Si se quiere el listado anterior con el nombre y apellido en una sola columna se puede usar el signo más para concatenar caracteres

```
select legajo, apellido + ', ' + nombre, fecha_nac  
from personas
```

Lo que se va a lograr, en este caso, es ver en la 2da. columna el apellido del alumno sumado a una coma y un espacio y luego el nombre; la tabla de resultado va a contener tres columnas.

legajo	(no column name)	fecha_nac
100	Rodríguez, Juan	1/5/1996
200	Pérez, Pedro	3/8/1984
300	Pérez, Ana	12/10/2010
400	Álvarez, Juana	3/9/1977
500	Sanchez, Juan	5/8/2002

Ilustración 3: Elaboración propia.

Alias: Si se presta atención al encabezado de cada columna de los ejemplos anteriores se verá que aparece en nombre del campo origen del dato o bien columna sin nombre.

Se puede agregar un alias a los campos que aparecerán como encabezados de columnas:

```
select legajo, apellido + ', ' + nombre as 'Nombre Completo',  
       fecha_nac  
from personas
```


legajo	Nombre Completo	fecha_nac
100	Rodríguez, Juan	1/5/1996
200	Pérez, Pedro	3/8/1984
300	Pérez, Ana	12/10/2010
400	Álvarez, Juana	3/9/1977
500	Sanchez, Juan	5/8/2002

Ilustración 4: Elaboración propia.

El alias va entre comillas salvo que no contenga caracteres especiales, y en el Transact-SQL, actualmente, no es necesaria la palabra reservada AS

```
select legajo, apellido + ', ' + nombre 'Nombre Completo',
       fecha_nac
from personas
```

Aquí la tercer columna también tiene alias que está sin comillas ya que pueden omitirse cuando el alias no tiene caracteres especiales.

DISTINCT|ALL. Distinct elimina filas duplicadas en el resultado de la consulta. Por defecto está definida la cláusula ALL es decir muestra todo, incluido si existen filas con exactamente el mismo contenido.

Se puede ver ejecutando estas dos consultas y observando detenidamente la diferencia entre ambos resultados

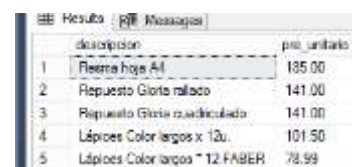
```
select peso, altura
from personas
```

```
select distinct peso, altura
from personas
```

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
```



	descripcion	pre_unitario
1	Pluma bote All	135.00
2	Repuesto Glorix mallado	141.00
3	Repuesto Glorix cuadrado	141.00
4	Lápices Color largos x 12u.	101.50
5	Lápices Color largos * 12 FABER	78.99

Otras cláusulas de la sentencia select

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 y habrá que prestar atención a la forma en que las tablas deben relacionarse o combinarse en la sentencia select.

WHERE condición: Con frecuencia suele ser necesario aplicar algún tipo de condición que restrinja el número de registros devuelto por una consulta. Estas son condiciones de búsqueda o bien, en alguna bibliografía se lo suele encontrar como test del where que se verá más adelante.

GROUP BY: Establece la lista de columnas por las cuales se agrupará la información.

HAVING: Condición que permite filtrar los grupos generados por GROUP BY.

Ordenamiento de los resultados de una consulta

ORDER BY: lista de columnas que debe utilizarse como criterio para ordenar los registros. La palabra reservada ASC indica que el orden ascendente es decir de mayor a menor para columnas numéricas; en order alfabético para columnas alfanuméricas según el código ASCII, y de la fecha más antigua a las más reciente si la columna es de tipo fecha. Si se indica DESC la columna se ordena en forma inversa a la anterior

Si la lista está compuesta por más de una columna, se separan por comas, y cada una con su respectivo ASC y DESC según corresponda

Se verá esta cláusula en un ejemplo: listar legajo apellido, nombre y fecha de nacimiento de personas ordenado por apellido y nombre

```
select legajo, apellido, nombre, fecha_nac  
from personas  
order by apellido, nombre
```

legajo	apellido	nombre	fecha_nac
400	Álvarez	Juana	3/9/1977
300	Pérez	Ana	12/10/2010
200	Pérez	Pedro	3/8/1984
100	Rodríguez	Juan	1/5/1996
500	Sanchez	Juan	5/8/2002

Ilustración 5: Elaboración propia.

Se puede observar cómo en primer lugar se ordena el listado por el apellido de las personas y en segundo lugar (cuando el apellido, el primer criterio es repetido) se ordena por el nombre.

Otro ejemplo: ordenar el listado por la segunda columna :

```
select legajo, apellido + ', ' + nombre 'Nombre Completo',
       fecha_nac
from personas
order by 2
```

o bien,

```
select legajo, apellido + ', ' + nombre 'Nombre Completo',
       fecha_nac
from personas
order by apellido + ', ' + nombre
```

o bien,

```
select legajo, apellido + ', ' + nombre 'Nombre Completo',
       fecha_nac
from personas
order by 'Nombre Completo'
```

Si se quisiera ordenar por apellido en forma descendente (de la Z a la A) y en nombre en forma ascendente:

```
select legajo, apellido, nombre, fecha_nac
from personas
order by apellido desc, nombre
```

o bien,

```
select legajo, apellido, nombre, fecha_nac
from personas
order by 2 desc, 3
```

Condiciones de búsqueda en el 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.

Los ejemplos de aquí en más se van a basar en la base de datos librería cuyo diagrama es el que se muestra a continuación:



Ilustración 6: Elaboración propia

Condiciones de comparación o 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. Se utilizan los operadores de comparación. Por ejemplo, si se quiere mostrar los libros cuyo precio sea menor a \$ 150 podemos utilizar un operador de comparación en el WHERE.

Por ejemplo, listar las facturas emitidas antes del 10/7/2008

```

select *
from facturas
where fecha < '10/07/2008'
  
```

Condición de búsqueda (o Test) de correspondencia con un patrón de búsqueda

Comprueba si el valor de dato de una columna se ajusta a un patrón especificado. Se utiliza el operador **LIKE** y 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].

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

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



	cod_articulo	descripcion	stock_minimo	pre_unitario	observaciones
1	1	Lápiz Evolution HB2 * 4 u	NULL	50.00	NULL
2	4	Lápices Color cortos * 12 u	54	120.50	NULL
3	6	Lápices Color largos * 12 u	NULL	127.90	NULL
4	22	Lápices con goma * 2 u	NULL	45.50	NULL
5	25	Lápices Color largos * 12 FABER	NULL	178.99	NULL
6	26	Lapicera Bic Azul trazo fino	NULL	24.99	NULL

Ilustración 7: Elaboración propia

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]%'
```

Condición de búsqueda (o Test) de rango

Se utiliza el operador BETWEEN para verificar si el valor de una columna se encuentra entre dos valores devueltos por dos expresiones, 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
```

	cod_articulo	descripcion	stock_minimo	pre_unitario	observaciones
1	1	Lápiz Evolution HB2 * 4 u	NULL	50.00	NULL
2	7	Separadores tamaño rivadavia * 6 u	NULL	85.70	Motivos de Disney
3	11	Cuademo Tamaño rivadavia rayado	80	52.70	NULL
4	12	Cuademo Tamaño rivadavia cuadriculado	65	52.70	NULL
5	15	Conjunto Geométrico	NULL	85.90	Regla - escuadra - transportador
6	17	Cartuchera de Tela	55	62.60	NULL
7	18	Correctores Bic Lápiz * 1 u	NULL	71.40	NULL
8	19	Rótulos * 18 u	NULL	55.90	NULL
9	23	Goma para Lápiz * 10 u	NULL	87.50	NULL
10	24	Goma para lapicera * 10 u	NULL	99.50	NULL
11	27	Cuademo tapa dura rayado	20	92.99	NULL

Ilustración 8: Elaboración propia

Condición de búsqueda (o Test) de pertenencia a un conjunto

Se utiliza el operador IN para verificar si el valor de una columna se encuentra entre una lista de valores. 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)
```

	cod_cliente	nom_cliente	ape_cliente	calle	altura	cod_barrio	nro_tel	e-mail
1	1	Rodolfo	Perez	San Martin	120	1	NULL	NULL
2	3	Héctor	Abarca	Luis Gongora	160	12	4701314	habarca@hotmail.com
3	7	Roque	Paez	Humberto Primo	79	1	4262630	NULL
4	8	Elvira Josefa	Luque	Mariano Usandivaras	360	3	4502829	NULL
5	12	Adriana	Gonzalez	San Jerónimo	763	1	NULL	NULL

Ilustración 9: Elaboración propia

Condición de búsqueda (o Test) de valor nulo

Se utiliza el operador NULL para verificar si el valor de una columna contiene o no un valor nulo, por ejemplo, Por ejemplo listar los artículos para los que no existan observaciones

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


Operador 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

	cod_articulo	descripcion	stock_minimo	pre_unitario	observaciones
1	2	Papel p/forrar Fantasia * 10 u	130	220.00	NULL
2	3	Papel p/forrar Araña * 10 u	150	250.00	Color rojo - azul - verde
3	4	Lápices Color cortos * 12 u	54	120.50	NULL
4	5	Fibras cortas * 6	40	125.30	NULL
5	6	Lápices Color largos * 12 u	NULL	127.90	NULL
6	8	Carpeta fibra negra - Tamaño rivadavia	NULL	102.90	3 anillos
7	9	Carpeta Fantasia - Tamaño Rivadavia	60	150.90	3 anillos

Ilustración 10: Elaboración propia

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

Operador 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 Fantasia * 10 u	130	22.30	NULL
2	3	Papel p/forrar Araña * 10 u	150	25.50	Color rojo - azul - verde
3	13	Repuesto Glove rallado	120	141.00	400 hojas
4	14	Repuesto Glove cuadrado	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

Ilustración 11: Elaboración propia

En el ejemplo se listarán los registros cuyo stock mínimo 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.

Operador 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	Alma	12	3	4612525	NULL	1975-03-06 00:00:00

Ilustración 12: Elaboración propia

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

Consultas compuestas

Por lo general, se suelen combinar registros procedentes de dos o más tablas a partir de valores coincidentes en un campo común. Otras veces es preciso hacer subconsultas, o lo que es lo mismo, crear mediante sentencias SELECT tablas temporales de datos que forman parte a su vez de la cláusula FROM de otra sentencia SELECT. En todos estos casos, donde las posibles combinaciones de elementos pueden ser muchas, no hay que olvidar el factor eficiencia.

Combinación de Tablas

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.

Hay varias formas de realizar una combinación de tablas; en esta oportunidad se realizará igualando los campos claves en la cláusula WHERE. Es lo que algunas bibliografías llaman “*Join Implícito*” y el resultado obtenido es una composición “*Interna*”, es decir que se mostrarán los registros de la primer tabla que tengan registro relacionados en la segunda tabla y los registros de la segunda que tengan

registros relacionados con la primera, si existe algún registro de una tabla que no tiene relación con la otra no se mostrará.

Por ejemplo, se quiere listar los vendedores y el barrio donde viven.

Listando los vendedores y los barrios por separado se puede observar que el vender código 1 vive en el barrio con código 2, y el vendedor 2 vive en el barrio 5

cod_vendedor	nombre	cod_barrio
1	Camizo Mart?n	2
2	Ledesma Mariela	5
3	Lopez Alejandro	3
4	Miranda Marcelo	1
5	Monti Gabriel	4
6	Juarez Susana	9

cod_barrio	barrio
1	CENTRO
2	ALTO ALBERDI
3	OBSERVATORIO
4	JARDÍN
5	GENERAL PAZ
6	PUEYRREDÓN
7	PARQUE HORIZONTE
8	SAN MARTÍN
9	SAN VICENTE
10	NUEVA CÓRDOBA
11	MAIPÚ
12	PANAMERICANO

Ilustración 13: Elaboración propia

Entonces, si se realiza la composición de las tablas a través del campo que tienen en común cod_barrio, la sentencia SELECT, sería la siguiente:

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

Ilustración 14: Elaboración propia

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.

Results		Messages
Vendedor	barrio	
1 Camizo 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	AL TO ALBERDI	

Ilustración 15: Elaboración propia

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.

Se puede dar un 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 alias 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
```

Ilustración 16: Elaboración propia

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
```

Ilustración 17: Elaboración propia

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

Ilustración 18: Elaboración propia

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.

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
```



	(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

Ilustración 19: Elaboración propia

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 "123.4", 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.
- **sqrt(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 (día), week (semana), hour (hora), minute (minuto), second (segundo) y millisecond (milisegundo).
- **datetimeame(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 (día), 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	2007
42	42	10	2008
43	43	10	2008
44	44	10	2009
45	45	10	2009
46	46	14	2009
47	47	14	2009
48	48	8	2010
49	49	15	2010
50	50	8	2010
51	51	8	2010
52	52	4	2011
53	53	1	2011

Ilustración 20: Elaboración propia

OTRAS FORMAS DE COMPOSICIÓN: INNER, LEFT Y RIGHT JOIN

Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

Hay tres tipos de combinaciones:

- combinaciones internas (inner join o join),
- combinaciones externas y
- combinaciones cruzadas.

Composición interna Inner Join

INNER JOIN (se conoce también como Join Explícito) selecciona registros que tienen valores coincidentes en ambas tablas.

La combinación interna emplea "join", que es la forma abreviada de "inner join". Se emplea para obtener información de dos tablas y combinar dicha información en una salida. La sintaxis básica es la siguiente:

```
SELECT campos
FROM tabla1
JOIN tabla2
ON condicion de combinacion;
```

Por ejemplo si se quieren listar los datos de los clientes con sus facturas: (cada factura con su respectivo cliente)

```
select ape_cliente + ' ' + nom_cliente CLIENTE, nro_factura FACTURA, fecha FECHA
from facturas f join clientes c on f.cod_cliente=c.cod_cliente
```

En la cláusula From se especifica el nombre de la primer tabla (facturas), luego combinamos join con la segunda tabla (clientes) y luego de on especificamos a través de qué campos se combinan.

Si se quiere realizar una consulta con más de dos tablas, por ejemplo listar los datos de los clientes, con sus facturas y vendedores, sería:

```
select ape_cliente+' '+nom_cliente CLIENTE,nro_factura FACTURA,FORMAT (fecha,'dd/mm/yyyy') FECHA,ape_vendedor+' '+nom_vendedor VENDEDOR
from facturas f join clientes c on f.cod_cliente=c.cod_cliente
join vendedores v on f.cod_vendedor=v.cod_vendedor
```

	CLIENTE	FACTURA	FECHA	VENDEDOR
510	Morales Santiago	510	30/00/2018	Miranda Marcelo
511	Castillo Marta Analía	511	01/00/2018	Miranda Marcelo
512	Morales Santiago	512	01/00/2018	Monti Gabriel
513	Morales Pilar	513	02/00/2018	Monti Gabriel
514	Morales Santiago	514	10/00/2018	Lopez Alejandro
515	Morales Santiago	515	11/00/2018	Miranda Marcelo
516	Perez Rodolfo	516	12/00/2018	Monti Gabriel
517	Castillo Marta Analía	517	14/00/2018	Miranda Marcelo

Ilustración 21: Elaboración propia

Usar “join” o “inner join” da el mismo resultado que realizar la composición como una condición de búsqueda en el where, solo se mostrarán los resultados de aquellos registros donde coincidan los valores de sus claves primarias y foráneas respectivas es decir no se mostrarán los registros donde el cliente no tenga facturas o el vendedor no tenga registradas facturas o facturas sin clientes ni facturas sin vendedores.

Combinación externa izquierda: left join

Una combinación interna (join) encuentra registros de la primera tabla que se correspondan con los registros de la segunda, y si un valor de la primera tabla no se encuentra en la segunda tabla, el registro no aparece.

```
select a.cod_articulo,descripcion,nro_factura,cantidad
from articulos a left join detalle_facturas d on a.cod_articulo = d.cod_articulo
```

	cod_articulo	descripcion	nro_factura	cantidad
939	23	Goma para lápiz * 10 u	267	20
940	23	Goma para lápiz * 10 u	276	15
941	24	Goma para lapicera * 10 u	48	25
942	25	Lápices Color largos * 12 FABER	59	2
943	25	Lápices Color largos * 12 FABER	138	7
944	25	Lápices Color largos * 12 FABER	247	1
945	25	Lápices Color largos * 12 FABER	308	1
946	25	Lápices Color largos * 12 FABER	188	1
947	25	Lápices Color largos * 12 FABER	380	1
948	26	Lapicera Bic Azul trazo fino	NULL	NULL
949	27	Cuademo tapa dura rayado	NULL	NULL
950	28	Lápices Color largos x 12u.	NULL	NULL
951	29	Conjunto geométrico Maped	NULL	NULL

Query executed successfully.

Ilustración 22: Elaboración propia

Se emplea una combinación externa izquierda para mostrar todos los registros de la tabla de la izquierda. Si no encuentra coincidencia con la tabla de la derecha, el registro muestra los campos de la segunda tabla seteados a "null". En el siguiente ejemplo solicitamos la descripción del artículo y cantidad vendida en cada factura:

El resultado mostrará las descripciones de los artículos y las cantidades vendidas; los artículos que no han sido vendidos aparecen en el resultado, pero con el valor "null" en los campos "cant" y "nro_factu".

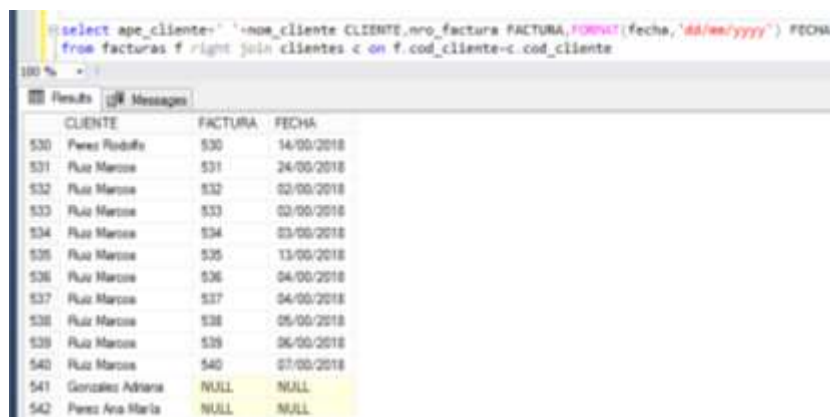
Es importante la posición en que se colocan las tablas en un "left join", la tabla de la izquierda es la que se usa para localizar registros en la tabla de la derecha.

Entonces, un "left join" se usa para hacer coincidir registros en una tabla (izquierda) con otra tabla (derecha); si un valor de la tabla de la izquierda no encuentra coincidencia en la tabla de la derecha, se genera una fila extra (una por cada valor no encontrado) con todos los campos correspondientes a la tabla derecha seteados a "null".

Combinación externa derecha: right join

Vimos que una combinación externa izquierda (left join) encuentra registros de la tabla izquierda que se correspondan con los registros de la tabla derecha y si un valor de la tabla izquierda no se encuentra en la tabla derecha, el registro muestra los campos correspondientes a la tabla de la derecha con "null".

Una combinación externa derecha ("right outer join" o "right join") opera del mismo modo sólo que la tabla derecha es la que localiza los registros en la tabla izquierda. En el siguiente ejemplo solicitamos el nombre del cliente (aunque no haya facturas de ese cliente) y las facturas de dicho cliente:



```
select ape_cliente= ' '||nom_cliente CLIENTE,nro_factura FACTURA,FORMAT(fecha,'dd/mm/yyyy') FECHA
from facturas f right join clientes c on f.cod_cliente=c.cod_cliente
```

	CLIENTE	FACTURA	FECHA
530	Perez Rodolfo	530	14/05/2018
531	Perez Rodolfo	531	24/05/2018
532	Perez Rodolfo	532	02/06/2018
533	Perez Rodolfo	533	02/06/2018
534	Perez Rodolfo	534	03/06/2018
535	Perez Rodolfo	535	13/06/2018
536	Perez Rodolfo	536	04/06/2018
537	Perez Rodolfo	537	04/06/2018
538	Perez Rodolfo	538	05/06/2018
539	Perez Rodolfo	539	06/06/2018
540	Perez Rodolfo	540	07/06/2018
541	Gonzalez Adriana	NULL	NULL
542	Perez Ana Maria	NULL	NULL

Ilustración 23: Elaboración propia

Es FUNDAMENTAL tener en cuenta la posición en que se colocan las tablas en los "outer join". En un "left join" la primera tabla (izquierda) es la que busca coincidencias en la segunda tabla (derecha); en el "right join" la segunda tabla (derecha) es la que busca coincidencias en la primera tabla (izquierda).

Combinación externa completa: full join

Una combinación externa completa ("full outer join" o "full join") retorna todos los registros de ambas tablas. Si un registro de una tabla izquierda no encuentra coincidencia en la tabla derecha, las columnas correspondientes a campos de la tabla derecha aparecen seteadas a "null", y si la tabla de la derecha no encuentra correspondencia en la tabla izquierda, los campos de esta última aparecen conteniendo "null". Veamos un ejemplo:

```
select ape_cliente+' '+nom_cliente CLIENTE,nro_factura FACTURA  
from facturas f full join clientes c on f.cod_cliente=c.cod_cliente
```

La salida del "full join" precedente muestra todos los registros de ambas tablas, incluyendo los clientes que no tengan facturas y las facturas que no tengan clientes.

Combinaciones cruzadas: cross join

Las combinaciones cruzadas (cross join) muestran todas las combinaciones de todos los registros de las tablas combinadas. Para este tipo de join no se incluye una condición de enlace. Se genera el producto cartesiano en el que el número de filas del resultado es igual al número de registros de la primera tabla multiplicado por el número de registros de la segunda tabla, es decir, si hay 5 registros en una tabla y 6 en la otra, retorna 30 filas.

Combinar varios tipos de join en una misma sentencia.

Es posible realizar varias combinaciones para obtener información de varias tablas. Las tablas deben tener claves externas relacionadas con las tablas a combinar. En consultas en las cuales empleamos varios "join" es importante tener en cuenta el orden de las tablas y los tipos de "join"; recuerde que la tabla resultado del primer join es la que se combina con el segundo join, no la segunda tabla nombrada.

BIBLIOGRAFÍA

Gorman K., Hirt A., Noderer D., Rowland-Jones J., Sirpal A., Ryan D. & Woody B (2019) Introducing Microsoft SQL Server 2019. Reliability, scalability, and security both on premises and in the cloud. Packt Publishing Ltd. Birmingham UK

Microsoft. SQL Server 2016. Disponible en: <https://www.microsoft.com/es-es/sql-server/sql-server-2016>

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

Varga S., Cherry D., D'Antoni J. (2016). Introducing Microsoft SQL Server 2016 Mission-Critical Applications, Deeper Insights, Hyperscale Cloud. Washington. Microsoft Press



Atribución-No Comercial-Sin Derivadas

Se permite descargar esta obra y compartirla, siempre y cuando no sea modificado y/o alterado su contenido, ni se comercialice. Referenciarlo de la siguiente manera: Universidad Tecnológica Nacional Facultad Regional Córdoba (S/D). Material para la Tecnicatura Universitaria en Programación, modalidad virtual, Córdoba, Argentina.