



Tecnicatura Universitaria  
en Programación

## LABORATORIO DE COMPUTACIÓN I

Unidad Temática N°2:  
Sentencias de Manipulación de Datos

Guía de Estudio  
1° Año – 1° Cuatrimestre



## Índice

SENTENCIAS DE ACTUALIZACIÓN DE DATOS.....	2
Problema 1 .....	2
Problema 2 .....	4
PROBLEMA INTEGRADOR – PARTE III.....	8
SENTENCIAS DE RECUPERACIÓN DE DATOS.....	8
Problema 3 .....	8
Problema 4 .....	11
Problema 5 .....	16
PROBLEMA INTEGRADOR – PARTE IV .....	18
BIBLIOGRAFÍA .....	19

## SENTENCIAS DE ACTUALIZACIÓN DE DATOS

### Problema 1

En una institución educativa necesitan llevar los datos de los alumnos que asisten a dicha institución para ello relate los siguientes puntos:

1. Cree una base de datos de nombre Administracion\_Alumnos y en ella cree las siguientes tablas:

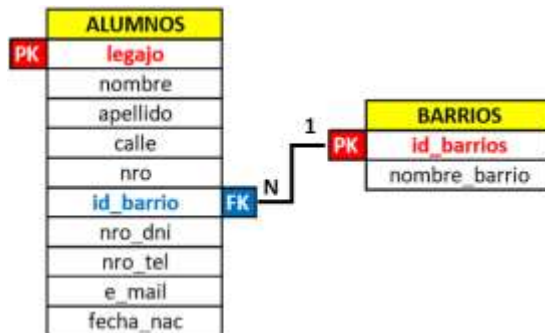


Ilustración 1: Elaboración propia

Al crear las tablas tenga en cuenta lo siguiente:

Tabla Barrios: Clave principal: Id\_barrio (Identity)

Tabla Alumnos: Clave principal: Legajo Clave foránea: Id\_barrio

2. Inserte al menos 10 registros en cada tabla

Ejemplo tabla barrios:

```
insert into barrios (nombre_barrio)
values ('San Andrés')
```

Se debe tener en cuenta que el campo id\_barrio no se incluye en la lista de campos porque es Identity.

Después de cada insert se puede consultar, por ejemplo:

```
select * from barrios
```

Ejemplo tabla alumnos:

```
insert into alumnos (Legajo, nombre, apellido, calle, nro,
id_barrio, nro_dni, nro_tel, e_mail, fecha_nac)
```

```
values (106743, 'Martín', 'Pedraza', 'Los Olmos', 3422, 1,  
       '31234567', '', Null, '12/03/1988')
```

```
select * from alumnos
```

Tenga en cuenta que:

- Los campos de tipo char, varchar y datetime se ingresan entre comillas simples.
- El campo legajo si no es identity se incluye en la lista de campos porque de lo contrario da error.
- Todos los campos incluidos en la lista de campos deben tener un dato en la lista values o bien 0 (cero) para numéricos, "" (cadena de longitud cero) para char o varchar o bien NULL.
- Si el número de teléfono es int y si se incluye código de área, la cantidad de dígitos no alcanza.

Inserte al menos 9 registros más en cada tabla, siga los ejemplos anteriores y cada vez que agregue un registro utilice:

```
select * from barrios
```

para comprobar los cambios en la tabla Barrios y

```
select * from alumnos
```

para comprobar los cambios en la tabla Alumnos

3. Realice las siguientes actualizaciones de datos; después de cada actualización consultar para ver el cambio

a) Agregue el nro. de teléfono 5945566 al alumno cuyo legajo es 106743.

Ejemplo:

```
update alumnos  
set nro_tel = '5945566'  
where legajo = 106743
```

```
select * from alumnos
```

- Agregue una dirección de e-mail al mismo alumno.
  - El mismo alumno del ejercicio anterior cambió su dirección por Caminoti 7689
  - Cambiar los datos de al menos 5 registros.
4. Elimine al menos dos registros de cada tabla.

Ejemplo:

```
delete alumnos where legajo = 106743
```

## Problema 2

Una librería mayorista cuenta con una base de datos llamada **LIBRERIA** que tiene la siguiente estructura:



Ilustración 2: Elaboración propia

El encargado de la librería necesita realizar una serie de actualizaciones,

1. Ingresar los artículos que están en la siguiente lista:

Descripción del artículo	Stock Mínimo	Precio unitario	Observaciones
Lápices Color largos * 12 u. Bic		101,50	
Conjunto Geométrico Maped		20,50	Regla, escuadra y transportador
Repuesto Gloria rallado	120	326,30	200 hojas
Repuesto Rivadavia		465,90	260 hojas, margen reforzado

**Tabla 1:** Elaboración propia

Para insertar nuevas filas a una tabla utilizamos la sentencia **Insert**,

Por ejemplo, para agregar a la tabla de artículos el primero de la lista anterior, deberemos escribir esta sentencia:

```

Insert into articulos (descripcion, pre_unitario)
values ('Lápices Color largos x 12u. Bic' 101.5)

--Una vez ejecutada ésta, podremos consultar todos los datos de la tabla artículos de esta manera
select * from articulos

--o bien, podremos consultar todos los datos de la tabla artículos cuya descripción comience con "Lápices"
select * from articulos
where descripción like 'Lápices%'

```

de esta forma se puede ir verificando lo realizado. De la misma manera continúe con el resto de los artículos, y tenga en cuenta lo siguiente:

- El campo cod\_articulo no se incluye en la lista de campos porque es Identity, lo mismo que cod\_vendedor y cod\_cliente.
- Si un campo clave principal no es identity se debe incluir en la lista de campos porque de lo contrario da error.
- Cuando ingrese campos de tipo char, varchar y datetime se ingresan entre comillas simples.
- Todos los campos incluidos en la lista de campos deben tener un dato en la lista values o bien 0 (cero) para numéricos, " (cadena de longitud cero) para char o varchar o bien NULL
- Si el número de teléfono es int y si se incluye código de área, la cantidad de dígitos no alcanza.

- Si el valor que intenta agregar a una de las columnas no cumple con alguno de los constraints establecidos la operación abortará inmediatamente.

2. La librería además necesita que se agreguen los siguientes vendedores:

Apellido	Nombre	Calle	Altura	Tel.	Mail	Fecha Nac.	Barrio
Monti	Juan	Altoaguirre	1245	4522122			5
Sena	Rosa	Av. Velez Sarsfield	25		rsena@hotmail.com	15/5/1968	1

Tabla 2: Elaboración propia

Agregue usted tres clientes.

3. Días después se encontraron algunos errores en la carga de datos y será necesario modificarlos.

La modificación pedida fue que para el artículo cuya descripción es “Conjunto Geométrico” hay que cambiarla por “Conjunto Geométrico de Plástico”. Primeramente, vamos a verificar el cod\_articulo que el mismo posee:

```
SELECT *  
  
FROM articulos  
  
WHERE descripcion LIKE 'conjunto%'
```

El código de artículo a modificar es el 29 entonces se modificará con la sentencia update dicho artículo. Se debe verificar en su base de datos cuál es el código que corresponda, ya que el mismo es identity y es la base de datos la que le otorga el número y puede variar de una a otra.

```
UPDATE articulos  
  
SET descripcion = 'Conjunto geométrico de plástico'  
  
WHERE cod_arti = 29
```

Se verifica si se logró realizar la actualización correctamente:

Luego se nos pidieron las siguientes modificaciones:



- a. Para el artículo cuya descripción es “Lápices Color largos \* 12 u” Bic; cambie el stock mínimo por 100, las observaciones por “Caja con motivos de Disney” y al precio por 17.20.
- b. Para el artículo cuya descripción es “Repuesto Rivadavia”, cambie la descripción por “Repuesto Rivadavia cuadriculado” y las observaciones por “48 hojas”.
- c. Al vendedor Monti Juan cambiar la fecha por 10/10/1970 y agregarle una dirección de e-mail.
- d. Al vendedor Sena Rosa, cambiar el teléfono por 4522221 y la dirección por Av. Vélez Sarsfield 125 - Centro.
- e. Aumentar el precio unitario en un 15% pero de aquellos artículos cuyo precio unitario sea inferior a 20. (tener en cuenta además que la descripción comience con su legajo)
- f. Aumentar el precio unitario en un 10% pero de aquellos artículos cuyo precio unitario esté entre a 20 y 30. (tener en cuenta además que la descripción comience con su legajo)

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.

4. Por errores en la carga de datos de algunos registros habrá que eliminarlos y para ello utilizaremos la sentencia Delete cuya sintaxis 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.

Los registros a borrar son los siguientes:

- a. Al vendedor cuyo apellido y nombre es “Sena Rosa”.
- b. Al artículo cuya descripción es “Repuesto Gloria rallado”.



- c. Elimine dos registros de los que ud. haya ingresado en los puntos anteriores.

Eliminar el vendedor “Sena Rosa” pero primero se verifica su código de vendedor (la clave principal que es única) para asegurarnos de que sea el correcto, y para ello se va a consultar los vendedores:

```
SELECT *  
FROM vendedores
```

Utilizar la sentencia delete para eliminar el registro.

```
DELETE vendedores  
WHERE cod_vendedor = 9
```

Volver a consultar los vendedores:

```
SELECT *  
FROM vendedores
```

Proceder del mismo modo para los otros dos registros.

## PROBLEMA INTEGRADOR – PARTE III

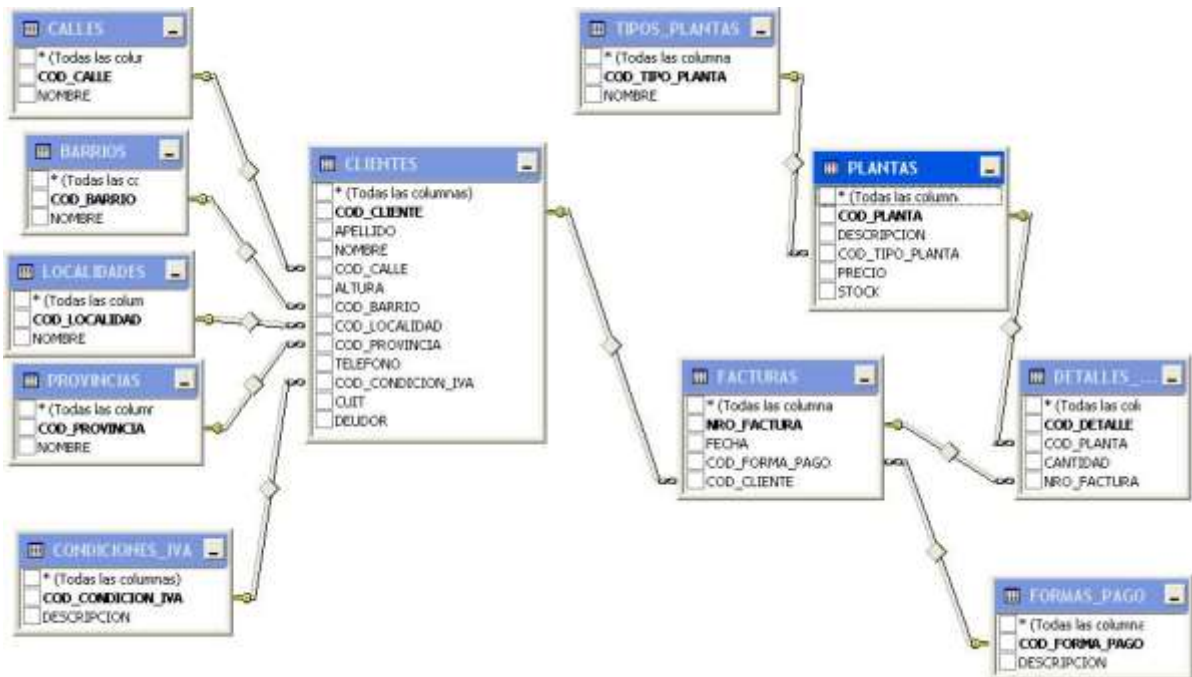
Utilizando sentencias de actualización de datos ingresar registros a las tablas de la base de datos designada al grupo de trabajo.

## SENTENCIAS DE RECUPERACIÓN DE DATOS

### Problema 3

Un vivero de la ciudad cuenta con una base de datos cuyo diagrama es el que se muestra a continuación.

El administrador del vivero necesita obtener información para la toma de decisiones.



**Ilustración 3:** Elaboración propia

Para ello abra la base de datos VIVERO\_FENIX y resuelva las consultas que se piden a continuación.

### La lista de selección y ordenación del resultado: **Select y Order by**

1. Listado de los clientes de la empresa, mostrando solo el código, apellido y nombre, ordenado por código.
2. Ídem al anterior pero rotule los campos como Código de Cliente, Apellido, Nombre. Ordenar por apellido y nombre en forma ascendente.
3. Listado de las plantas que se venden, mostrando el código, descripción y precio, ordenadas por código en forma descendente.
4. Mostrar el stock de cada planta, ordenadas por la descripción de las plantas. Rotular: Nombre Planta, Stock Actual.
5. Listado de las localidades que compran a la empresa, ordenados por orden alfabético.
6. Mostrar apellido y nombre del cliente en una sola columna, además de su código.
7. Mostrar todos los datos de las plantas y además calcular el precio con un descuento del 10%.
8. Mostrar el precio total de stock de cada planta.

### Test de Comparación

9. Mostrar todos los clientes deudores (deudor='S'), solo su código, nombre, apellido.
10. Mostrar todas las plantas cuyo stock sea mayor a 20, ordenadas por stock.
11. Mostrar todas las plantas cuyo stock no sea 30, ordenados por código.
12. Mostrar todas las facturas cuyas fechas sean mayores al 01/06/09.
13. Listar todas las plantas cuyo stock es menor a 10 unidades.

### Test de Rango: Between

14. Ídem a la consulta anterior pero las fechas oscilen entre 01/06/08 y 01/03/10.
15. Listar los datos de las plantas cuyo precio esté entre 20 y 70.
16. Mostrar la descripción, stock y precio de las plantas cuyo stock oscile entre 5 y 10.

### Test de correspondencia con patrón de búsqueda: Like

17. Todos los clientes cuyo apellido comience con "f".
18. Todos los clientes cuyo nombre contenga una "u".
19. Todos los clientes cuyo apellido no comience con letras que van de la "a" a la "c".
20. Todos los clientes cuyo apellido termine con "ez".
21. Listar todos los clientes cuyo apellido no comience con "p".
22. Ídem al anterior pero que no comience ni con "p" ni con "z".

### Test de valor nulo: is Null

23. Mostrar todos los clientes que no tengan teléfono.
24. Listar apellido, nombre, e-mail de los clientes cuyos direcciones de mail sea conocidas.
25. Listar apellido, nombre, teléfono de los clientes cuyos teléfono sea conocido.

### Varias condiciones de búsqueda

26. Listar todas las plantas que comiencen con R y precio superior a \$7.
27. Mostrar todas las plantas cuyo precio sean menor a \$ 50 o mayor a \$ 100, Mostrar también su precio con un aumento del 10%
28. Listar apellido, nombre, teléfono e e-mail de los clientes cuyos direcciones de mail o teléfono sean conocidos.

29. Listar los datos de los clientes cuyos nombres comiencen no comiencen ni con "m" ni con "n" ni con "p" o su precio sea mayor a 50 con una valoración de stock menor a \$ 1.000
30. Mostrar todas las plantas cuyo stock sea mayor a 5 y menor a 50 o bien su precio aumentado en un 15% esté entre 50 y 100. ordenr por descripción.

#### Consultas compuestas

31. Listado de clientes mostrando apellido y nombre, calle y número.
32. Listado de clientes mostrando apellido, nombre y localidad a la que pertenece.
33. Listado de clientes mostrando apellido, nombre y barrio al que pertenece.
34. Todas las facturas abonadas en efectivo, ordenada por código de cliente.
35. Listar todas plantas donde el tipo de planta sea Flores.
36. Todos los clientes cuya localidad sea AREQUITO.
37. Todos los registros completos de clientes cuyo IVA sea igual a Monotributo, ordenado por localidad.
38. Listar todos los datos de los clientes que viven en la localidad de Córdoba o su apellido comience con la letra 'A'.
39. Listar clientes que viven en la localidad de Carlos Paz, Alta Gracia y Cba Capital.
40. Mostrar todos los clientes que no viven en las localidades de AREQUITO y MERLO.
41. Mostrar nombre, apellido y barrio de los clientes de Cba. Capital o de aquellos cuyos apellidos no comiencen con letras de la 'D'a la 'P'. Ordenar por barrio.
42. Listar el detalle de la factura calculando la cantidad por el precio.
43. Listar todos los clientes que vivan en provincias que comiencen con 'San' y que sea Monotributista.
44. Listar todos los clientes que viven en barrio Alberdi, Alta Córdoba y que además posean teléfono.

#### Problema 4

1. Se quiere tener un listado de los vendedores y los barrios donde viven y se quieren tener incluidos los barrios, aunque no tengamos vendedores viviendo en ellos.

Si se realiza este listado como se venía trabajando hasta ahora solo se obtendrá los registros donde se correspondan vendedores con un barrio, pero no los vendedores que no tengan asignados barrios ni barrios que no tengan vendedores..

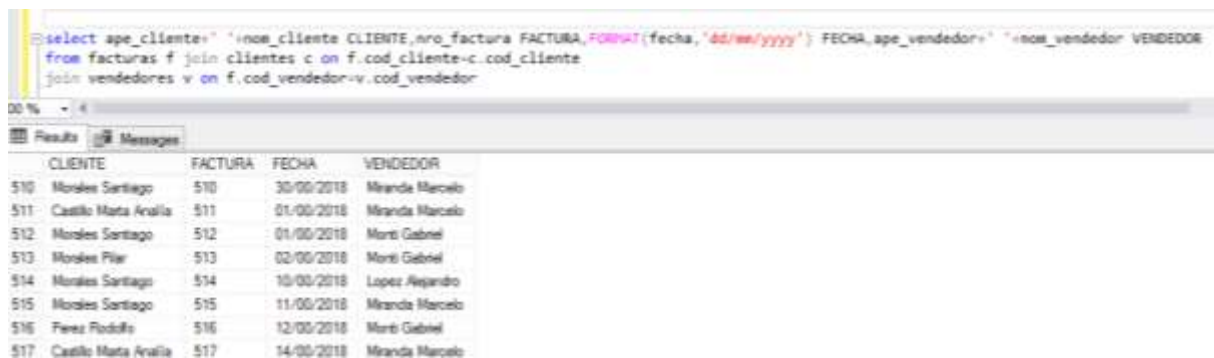
Esto es lo que se conoce como combinación interna, para lograr otro resultado como registros de una tabla que no tengan su correspondiente en la otra deberemos ver otra forma de realizar una composición de tablas.

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.

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
```

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	30/06/2018	Miranda Marcelo
511	Castillo Marta Aralia	511	01/06/2018	Miranda Marcelo
512	Morales Santiago	512	01/06/2018	Monti Gabriel
513	Morales Pilar	513	02/06/2018	Monti Gabriel
514	Morales Santiago	514	10/06/2018	Lopez Alejandro
515	Morales Santiago	515	11/06/2018	Miranda Marcelo
516	Perez Rodolfo	516	12/06/2018	Monti Gabriel
517	Castillo Marta Aralia	517	14/06/2018	Miranda Marcelo

Usar "**JOIN**" o "**INNER JOIN**" da el mismo resultado que realizar la composición como una condición de búsqueda en el **WHERE**.

### Combinación externa izquierda: **LEFT JOIN**

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 se solicita la descripción del artículo aunque no haya sido vendido y cantidad vendida en cada factura en caso que sí se haya vendido:

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 "cantidad" y "nro\_factura".

```
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.

### Combinación externa derecha: RIGHT JOIN

Se vio que una combinación externa izquierda (**LEFT JOIN**) encuentra registros de la tabla izquierda la que se muestra completa que se correspondan con los registros de la tabla derecha que solo se muestra lo que coincide.

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:

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).



```
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	14/00/2018
531	Ruiz Marcos	531	24/00/2018
532	Ruiz Marcos	532	02/00/2018
533	Ruiz Marcos	533	02/00/2018
534	Ruiz Marcos	534	03/00/2018
535	Ruiz Marcos	535	13/00/2018
536	Ruiz Marcos	536	04/00/2018
537	Ruiz Marcos	537	04/00/2018
538	Ruiz Marcos	538	05/00/2018
539	Ruiz Marcos	539	06/00/2018
540	Ruiz Marcos	540	07/00/2018
541	Gonzalez Adriana	NULL	NULL
542	Perez Ana María	NULL	NULL

### 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**". Por 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.

**Problema N° 4:** Resuelva utilizando **JOIN** los siguientes reportes

2. Genere un reporte con los datos de la facturación (datos de las facturas incluidos los del vendedor y cliente) de los años 2006, 2007, 2009 y 2012.
3. Liste los datos de la facturación, de los artículos y de la venta de las facturas correspondientes al mes pasado.
4. Emita un listado con los datos del vendedor y las ventas que ha realizado en lo que va del año. Muestre los vendedores aun así no tengan ventas registradas en el año solicitado.
5. Liste descripción, cantidad e importe; aun para aquellos artículos que no registran ventas.
6. Genere un reporte con los datos de la facturación (datos de las facturas incluidos los del vendedor y cliente) y de la venta (incluido el importe); para las ventas de febrero y marzo de los años 2006 y 2007 y siempre que el artículo empiece con letras que van de la "a" a la "m". Ordene por fecha, cliente y artículo.
7. Liste código de cliente, nombre, fecha y factura para las ventas del año 2007. Muestre los clientes hayan comprado o no en ese año.
8. Se quiere saber los artículos que compro la cliente Elvira López en lo que va del año. Liste artículo, observaciones e importe.
9. Se quiere saber los artículos que compraron los clientes cuyos apellidos empiezan con "p". Liste cliente, articulo, cantidad e importe. Ordene por cliente y artículo, este en forma descendente. Rotule como CLIENTE, ARTICULO, CANTIDAD, IMPORTE.

## Problema 5

Los usuarios finales de la librería mayorista del problema 2, necesitan obtener información para el funcionamiento del negocio y la toma de decisiones. Se vuelve a incluir la estructura de la base de datos:



10. Se necesita mostrar el código, nombre, apellido (todo el apellido en mayúsculas) y dirección (calle y altura en una sola columna; para la altura utilice una función de conversión) de todos los clientes cuyo nombre comience con "C" y cuyo apellido termine con "Z". Rotule como CÓDIGO DE CLIENTE, NOMBRE, DIRECCIÓN.
11. Ídem al anterior pero el apellido comience con letras que van de la "D" a la "L" y cuyo nombre no comience con letras que van de la "A" a la "G".
12. Muestre los datos de los vendedores (apellido todo en mayúsculas y en la misma columna que el nombre) cuyo nombre no contenga "Z", haya nacido en la década del 70 y que haya realizado ventas el mes pasado.
13. Mostrar las facturas realizadas entre el 1/1/2007 y el 1/5/2009 y cuyos códigos de vendedor sean 1, 3 y 4 o bien entre el 1/1/2010 y el 1/5/2011 y cuyos códigos de vendedor sean 2 y 4. Mostrar la fecha en formato Día, Mes, y Año (en ese orden y sin la hora)

14. Se quiere saber el subtotal de todos los artículos vendidos, para ello liste el artículo y multiplique la cantidad por el precio unitario de venta; mostrar el subtotal redondeado a dos decimales (o buscar la forma de dale formato apropiado). Ordene por alfabéticamente por artículo y cuyo subtotal mayor aparezca primero. No muestre datos duplicados.
15. Muestre las ventas (tabla detalle\_facturas) de los artículos cuyo precio unitario actual sea mayor o igual a 50 o cuyos códigos de artículos no sea uno de los siguientes: 2,5, 6, 8, 10. En ambos casos los precios unitarios a los que fueron vendidos oscilen entre 10 y 100.
16. Listar todos los datos de los artículos cuyo stock mínimo sea superior a 10 o cuyo precio sea inferior a 20. En ambos casos su descripción no debe comenzar con las letras "p" ni la letra "r".
17. Listar los datos de los vendedores nacidos en febrero, abril, mayo o septiembre.
18. Liste número de factura, fecha de venta y vendedor (apellido y nombre), para los casos en que los códigos del cliente van del 2 al 6. Ordene por vendedor y fecha, ambas en forma descendente.
19. Emitir un reporte con los datos de la factura del cliente y del vendedor de aquellas facturas confeccionadas entre el primero de febrero del 2008 y el primero de marzo del 2010 y que el apellido del cliente no contenga "C".
20. Listar los datos de la factura, los del artículo y el importe (precio por cantidad); para las facturas emitidas en el 2010, 2015 y 2017 y la descripción no comience con "R". Ordene por número de factura e importe, este en forma descendente. Rotule.
21. Se quiere saber qué artículos se vendieron, siempre que el precio unitario sin iva al que fue vendido no esté entre \$10 y \$50. Rotule.
22. Liste todos los datos de la factura (vendedor, cliente, artículo, incluidos los datos de la venta: cantidad, precio y subtotal); emitidas a clientes con teléfonos o direcciones de e-mail conocidas de aquellas facturas cuyo importe haya sido superior a \$250. Agregue rótulos y ordene el listado para darle mejor presentación.
23. Se quiere saber a qué cliente, de qué barrio, vendedor y en qué fecha se les vendió con los siguientes nros. de factura: 12, 18, 1, 3, 35, 26 y 29. ¿El barrio del cliente es el mismo que el barrio del vendedor que les vendió?

24. Emitir un reporte para informar qué artículos se vendieron, en las facturas cuyos números no esté entre 17 y 136. Liste la descripción, cantidad e importe ( $\text{Importe} = \text{cantidad} * \text{pre\_unitario}$ ). Ordene por descripción y cantidad. No muestre las filas con valores duplicados
25. Listar los datos de las facturas (cliente, artículo, incluidos los datos de la venta incluido el importe) emitidas a los clientes cuyos apellidos comiencen con letras que van de la "l" a "s" o los artículos vendidos que tengan descripciones que comiencen con las mismas letras. Ordenar el listado.
26. Realizar un reporte de los artículos que se vendieron en lo que va del año. (Muestre los datos que sean significativos para el usuario del sistema usando rótulos para que sea más legible y que los artículos no se muestren repetidos).
27. Se quiere saber a qué clientes se les vendió el año pasado, qué vendedor le realizó la venta, y qué artículos compró, siempre que el vendedor que les vendió sea menor de 35 años.
28. El usuario de este sistema necesita ver el listado de facturas, de aquellos artículos cuyos precios unitarios a los que fueron vendidos estén entre 50 y 100 y de aquellos vendedores cuyo apellido no comience con letras que van de la "l" a la "m". Ordenado por vendedor, fecha e importe.
29. Se desea emitir un listado de clientes que compraron en enero, además saber qué compraron cuánto gastaron (mostrar los datos en forma conveniente)
30. Emitir un reporte de artículos vendidos en el 2010 a qué precios se vendieron y qué precio tienen hoy. Mostrar el porcentaje de incremento.
31. Listar los vendedores que hace 10 años les vendieron a clientes cuyos nombres o apellidos comienzan con "C".
32. El encargado de la librería necesita tener información sobre los artículos que se vendían a menos de \$ 20 antes del 2015. Mostrar los datos que se consideren relevantes para el encargado, formatear, rotular y ordenar.

## PROBLEMA INTEGRADOR – PARTE IV

Crear consultas que obtengan información útil para la toma de decisiones a los usuarios finales de la base de datos designada al grupo de trabajo.

## 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.