



INGENIERÍA EN
SISTEMAS COMPUTACIONALES

.....



**Tecnológico Nacional de México
Campus Felipe Carrillo Puerto**

**Unidad Académica Chunhuhub
Ingeniería en Sistemas Computacionales**

Taller de base de datos
**Tema 2.- Lenguaje de manipulación
de datos**

Docente: José Torres Ek

2.1 Inserción, eliminación y modificación de registros



Insertar datos SQL

la instrucción INSERT permite agregar datos a las diferentes tablas en una base de datos. Su sintaxis es la siguiente:


```
INSERT INTO tabla (campo1, campo2) VALUES (valor1, valor2)
```

- El número de campos debe coincidir con el número de valores a insertar.
- Se debe proporcionar un valor por cada columna en la tabla excepto para las columnas que permiten valores nulos, que sean llaves primarias o que tienen un valor definido por defecto.
- Cada valor con un carácter del tipo de datos de cadena debe estar encerrado en comillas sencillas.




Insertar datos SQL

Tabla: tc_personal

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/>	1	id_persona 	tinyint(4)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/>	2	nombres	varchar(80)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	3	a_paterno	varchar(30)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	4	a_materno	varchar(30)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	5	activo	tinyint(4)		UNSIGNED	No	1	1=si, 0=no	
<input type="checkbox"/>	6	fecha_registro	timestamp			No	CURRENT_TIMESTAMP		

Insertamos un registro en la tabla:


Ejecutar la(s) consulta(s) SQL en la tabla `esquema_restricciones.tc_personal`: 

```
1 INSERT INTO tc_personal (nombres, a_paterno, a_materno) VALUES ('Juan Manuel', 'Solis', 'Valle')
```



Insertar datos SQL

Tabla: tc_personal

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/>	1	id_persona 	tinyint(4)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/>	2	nombres	varchar(80)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	3	a_paterno	varchar(30)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	4	a_materno	varchar(30)	latin1_swedish_ci		No	Ninguna		
<input type="checkbox"/>	5	activo	tinyint(4)		UNSIGNED	No	1	1=si, 0=no	
<input type="checkbox"/>	6	fecha_registro	timestamp			No	CURRENT_TIMESTAMP		

Insertamos un registro en la tabla:

Ejecutar la(s) consulta(s) SQL en la tabla `esquema_restricciones.tc_personal`: 

```
1 INSERT INTO tc_personal (nombres, a_paterno, a_materno, activo) VALUES ('Ana Patricia', 'Moreno', 'Salazar', 0)
```



Actualizar datos SQL

La instrucción UPDATE permite actualizar los datos en una base de datos. Con la instrucción UPDATE se pueden modificar datos en una o más filas para una o más columnas. La sintaxis para esta instrucción es la siguiente:

UPDATE *tabla* SET *campo* = *nuevo_valor* WHERE *condicion*

- En la cláusula SET se pueden especificar una o más expresiones separadas por comas.
- En la cláusula WHERE se debe especificar una o varias condiciones que servirán de filtro para los registros que serán actualizados. Únicamente los registros que cumplan la condición serán actualizados.



Actualizar datos SQL

Tabla: tc_personal

▼	id_persona	nombres	a_paterno	a_materno	activo 1=si, 0=no	fecha_registro
	1	Juan Manuel	Solis	Valle	1	2021-10-02 20:45:16
	2	Ana Patricia	Moreno	Salazar	0	2021-10-02 20:50:30

Actualizamos el campo activo para el id_persona=2:

Ejecutar la(s) consulta(s) SQL en la tabla esquema_restricciones.tc_personal: ⓘ

```
1 UPDATE tc_personal SET activo=1 WHERE id_persona=2
```



Actualizar datos SQL

Tabla: tc_personal

▼	id_persona	nombres	a_paterno	a_materno	activo 1=si, 0=no	fecha_registro
	1	Juan Manuel	Solis	Valle	1	2021-10-02 20:45:16
	2	Ana Patricia	Moreno	Salazar	0	2021-10-02 20:50:30

Actualizamos el campo activo para el registro con nombres = Ana Patricia y el a_paterno= Moreno:

Ejecutar la(s) consulta(s) SQL en la tabla `esquema_restricciones.tc_personal`:

```
1 UPDATE tc_personal SET activo=1 WHERE nombres='Ana Patricia' AND a_paterno= 'Moreno'
```



Actualizar datos SQL

Tabla: tc_personal

▼	id_persona	nombres	a_paterno	a_materno	activo 1=si, 0=no	fecha_registro
	1	Juan Manuel	Solis	Valle	1	2021-10-02 20:45:16
	2	Ana Patricia	Moreno	Salazar	0	2021-10-02 20:50:30

Actualizamos el campo nombres, a_paterno y a_materno para el registro con el id_persona=2:

Ejecutar la(s) consulta(s) SQL en la tabla esquema_restricciones.tc_personal: ?

```
1 UPDATE tc_personal SET nombres = 'Patricia', a_paterno='Salazar', a_materno='Moreno' WHERE id_persona=2
```



Eliminar datos SQL

La instrucción DELETE se utiliza para eliminar los datos en una base de datos. La sintaxis para la instrucción DELETE es la siguiente:

DELETE FROM *tabla* WHERE *condicion*

- En la cláusula WHERE se debe especificar una o varias condiciones que servirán de filtro para los registros que serán eliminados. Únicamente los registros que cumplan la condición serán eliminados.
- De no especificarse una condición todos los registros de la tabla serán eliminados.



Eliminar datos SQL

Tabla: tc_personal

▼	id_persona	nombres	a_paterno	a_materno	activo 1=si, 0=no	fecha_registro
	1	Juan Manuel	Solis	Valle	1	2021-10-02 20:45:16
	2	Ana Patricia	Moreno	Salazar	0	2021-10-02 20:50:30

Eliminamos el registro con el id_persona=2:

Ejecutar la(s) consulta(s) SQL en la tabla esquema_restricciones.tc_personal: ?

```
1 DELETE FROM tc_personal WHERE id_persona=2
```



2.2 Consultas



Consulta de datos SQL

La instrucción **SELECT** permite crear consultas que permiten recuperar información específica de la base de datos. La sintaxis para la instrucción **SELECT** puede mostrarse como sigue:



```
SELECT seleccion FROM tabla WHERE condicion
```

- En la cláusula **SELECT** se especifica uno, varios o todos los campos que se desean visualizar.
- En la cláusula **WHERE** se debe especificar una o varias condiciones que servirán de filtro para los registros que serán mostrados. Únicamente los registros que cumplan la condición serán mostrados.



Consulta de datos SQL

Tabla: tc_caja_movimientos

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/> 1	id_movimiento 	int(11)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/> 2	tipo_movimiento	tinyint(4)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 3	num_venta	int(11)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 4	num_tic	int(11)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 5	id_caja 	int(11)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 6	id_sucursal	tinyint(4)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 7	id_usuario	tinyint(4)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 8	cobro	float			No	Ninguna		
<input type="checkbox"/> 9	pagado	float			No	Ninguna		
<input type="checkbox"/> 10	cambio	float			No	Ninguna		
<input type="checkbox"/> 11	tipo_pago	tinyint(4)			No	Ninguna		
<input type="checkbox"/> 12	fecha_registro	timestamp			No	CURRENT_TIMESTAMP		



Consulta de datos SQL

Ejemplo: Seleccionamos todos los campos de la tabla tc_caja_movimientos de la sucursal con id_sucursal = 3

```
✓ Mostrando filas 0 - 24 (total de 20452, La consulta tardó 0,0008 segundos.)

SELECT * FROM `tc_caja_movimientos` WHERE id_sucursal=3

☐ Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]
```

Ejemplo: Seleccionamos todos los campos de la tabla tc_caja_movimientos de la sucursal con id_sucursal = 2 y que pertenezcan al id_usuario=11

```
✓ Mostrando filas 0 - 24 (total de 1604, La consulta tardó 0,0011 segundos.)

SELECT * FROM `tc_caja_movimientos` WHERE id_sucursal=3 AND id_usuario=11

☐ Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]
```



Consulta de datos SQL

Ejemplo: Seleccionamos los campos id_caja, cobro, pagado, fecha_registro de la tabla tc_caja_movimientos de la sucursal con id_sucursal = 3 y que pertenezcan al id_usuario=11

✓ Mostrando filas 0 - 24 (total de 1604, La consulta tardó 0,0009 segundos.)

```
SELECT id_caja, cobro, pagado, fecha_registro FROM `tc_caja_movimientos` WHERE id_sucursal=3 AND id_usuario=11
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

Ejemplo: Seleccionamos todos los campos de la tabla tc_caja_movimientos de la sucursal con id_sucursal = 3 y que el cobro sea mayor a 2000

✓ Mostrando filas 0 - 24 (total de 843, La consulta tardó 0,0037 segundos.)


```
SELECT * FROM `tc_caja_movimientos` WHERE id_sucursal=3 AND cobro > 2000
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]



Consulta de datos SQL

La palabra clave **DISTINCT** se utiliza si se desean eliminar filas duplicadas de los resultados de la consulta por ejemplo: Seleccionamos una lista (sin repeticiones) de los usuarios de la tabla `tc_caja_movimientos` de la sucursal con `id_sucursal = 3`.

 Mostrando filas 0 - 23 (total de 24, La consulta tardó 0,1603 segundos.)

```
SELECT DISTINCT id_usuario FROM `tc_caja_movimientos` WHERE id_sucursal=3
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]



2.3 Funciones, agrupamiento y ordenamiento



Funciones en consultas de datos SQL

La función COUNT cuenta el número de registros o el número de valores en una columna, según se especifique en la instrucción SELECT. Por ejemplo: Contamos el numero de registros de la tabla tc_caja_movimientos de la sucursal con id_sucursal = 3.

Su consulta se ejecutó con éxito.

```
SELECT COUNT(*) FROM `tc_caja_movimientos` WHERE id_sucursal=3
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

+ Opciones

COUNT(*)

20452

Su consulta se ejecutó con éxito.

```
SELECT COUNT(id_movimiento) FROM `tc_caja_movimientos` WHERE id_sucursal=3
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

+ Opciones

COUNT(id_movimiento)

20452



Funciones en consultas de datos SQL

La función MAX arroja el valor más alto para la columna especificada, y la función MIN arroja el valor más bajo. Ambas funciones requieren que se especifique un nombre de columna. Por ejemplo:

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,1129 segundos.)

```
SELECT MAX(cobro) FROM `tc_caja_movimientos`
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 Filtrar filas

+ Opciones

MAX(cobro)
19679

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,1662 segundos.)

```
SELECT MIN(cobro) FROM `tc_caja_movimientos`
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 Filtrar filas:

+ Opciones

MIN(cobro)
100



Funciones en consultas de datos SQL

La función SUM y AVG se utilizan para obtener la suma y el promedio de los valores de una columna respectivamente. Ambas funciones requieren que se especifique un nombre de columna. Por ejemplo:

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,1140 segundos.)

```
SELECT SUM(cobro) FROM `tc_caja_movimientos` WHERE id_sucursal=3 AND id_usuario=1
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 ▼

+ Opciones

SUM(cobro)
2620

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,1981 segundos.)

```
SELECT AVG(cobro) FROM `tc_caja_movimientos` WHERE id_sucursal=3 AND id_usuario=1
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 ▼ | Filtrar filas:

+ Opciones

AVG(cobro)
124.76190476190476



Agrupamiento en consultas de datos SQL

La cláusula GROUP BY se utiliza para agrupar tipos de información con el fin de resumir datos relacionados. Por ejemplo: Se quiere obtener el total de ventas por sucursal en la tabla tc_caja_movimientos.

✓ Mostrando filas 0 - 9 (total de 10, La consulta tardó 0,2089 segundos.)

```
SELECT id_sucursal, SUM(cobro) FROM `tc_caja_movimientos` GROUP BY id_sucursal
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

id_sucursal	SUM(cobro)
1	2272981
2	574814
3	9031569
4	16628656
5	22085
6	13619767
7	23163238
8	14621613
9	2977174
10	119331



Ordenamiento en consultas de datos SQL

La cláusula ORDER BY toma la salida de la cláusula SELECT y ordena los resultados de la consulta, en esta clausula se puede especificar si las filas se organizan en un orden ascendente (utilizando la palabra clave ASC) o en orden descendente (usando la palabra clave DESC).

✓ Mostrando filas 0 - 9 (total de 10, La consulta tardó 0,2716 segundos.)

```
SELECT id_sucursal, SUM(cobro) AS cobrado FROM  
'tc_caja_movimientos' GROUP BY id_sucursal ORDER BY cobrado ASC
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

✓ Mostrando filas 0 - 9 (total de 10, La consulta tardó 0,1973 segundos.)

```
SELECT id_sucursal, SUM(cobro) AS cobrado FROM  
'tc_caja_movimientos' GROUP BY id_sucursal ORDER BY cobrado DESC
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

id_sucursal	cobrado ▲ 1
5	22085
10	119331
2	574814
1	2272981
9	2977174
3	9031569
6	13619767
8	14621613
4	16628656
7	23163238

id_sucursal	cobrado ▼ 1
7	23163238
4	16628656
8	14621613
6	13619767
3	9031569
9	2977174
1	2272981
2	574814
10	119331
5	22085



2.4 Joins



Consultas a más de una tabla SQL

Las sentencias JOIN en SQL sirven para combinar filas de dos o más tablas basándose en un campo común entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.

El tipo más común de JOIN es el INNER JOIN el cual selecciona todas las filas de dos columnas siempre y cuando haya una coincidencia entre las columnas en ambas tablas.

```
SELECT seleccion FROM tabla1 INNER JOIN tabla2  
ON (campocomuntabla1 = campocomuntabla2) AND condicion
```





Consultas a más de una tabla SQL

Utilizaremos las siguientes tablas para ver un ejemplo del uso la sentencia INNER JOIN

tc_caja

Nombre	Tipo	Cotejamiento	Atributos
id_caja 	int(11)		UNSIGNED
id_sucursal	tinyint(4)		UNSIGNED
usuario_inicio	tinyint(4)		UNSIGNED
usuario_cierre	tinyint(4)		UNSIGNED
estado	tinyint(4)		UNSIGNED
cantidad_inicial	float		
cantidad_cierre	float		
observaciones	tinytext	latin1_swedish_ci	
fecha_inicio	timestamp		
fecha_cierre	timestamp		

tc_sucursal

Nombre	Tipo	Cotejamiento	Atributos
id_sucursal 	tinyint(4)		UNSIGNED
activo	tinyint(4)		UNSIGNED
nombre_sucursal 	varchar(60)	latin1_swedish_ci	
direccion	varchar(170)	latin1_swedish_ci	
colonia	varchar(120)	latin1_swedish_ci	
ciudad	varchar(100)	latin1_swedish_ci	
fecha_registro	timestamp		

Las tablas tienen en común el campo id_sucursal



Consultas a más de una tabla SQL

Ejemplo: Realizar una consulta que devuelva el nombre de la sucursal (tc_sucursales), el id de las sucursales (tc_caja), la cantidad con la que cerró la caja (tc_caja) y la fecha en la que cerró la caja (tc_caja) y que tengan una cantidad de cierre mayor a 35,000.

✓ MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0009 segundos.)

```
SELECT tc_sucursal.nombre_sucursal, tc_caja.id_sucursal, tc_caja.cantidad_cierre, tc_caja.fecha_cierre FROM  
tc_sucursal INNER JOIN tc_caja ON (tc_sucursal.id_sucursal=tc_caja.id_caja) AND tc_caja.cantidad_cierre >= 35000
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]



Consultas a más de una tabla SQL

En la figura se muestra parte del resultado, como puede verse, el resultado devuelve en una sola vista el campo nombre_sucursal de la tabla tc_sucursal y las columnas id_sucursal, cantidad_cierre y fecha_cierre de la tabla tc_caja.

nombre_sucursal	id_sucursal	cantidad_cierre	fecha_cierre
Morelos2	3	39240	2012-12-24 23:04:08
Carrillo2	4	36337.5	2013-12-24 23:08:12
Carrillo1	1	200634	2014-12-30 11:57:37
Morelos2	3	41450	2015-12-26 10:45:32
Tulum1	7	43993	2015-12-26 09:43:42
Tulum2	8	35054	2015-12-31 22:28:19
Tulum1	7	45700	2015-12-31 22:28:33
Tulum1	7	36181	2016-01-02 22:52:22
Tulum2	8	35640	2016-01-17 09:30:38
Tulum1	7	35043	2016-04-03 11:37:54
Tulum1	7	37220	2016-04-15 22:26:41
Tulum1	7	35749	2016-07-31 22:04:47
Tulum2	8	38205	2016-11-19 21:59:54
Tulum1	7	35852	2016-11-20 22:52:49
Tulum1	7	75270	2016-12-02 22:55:44



2.5 Subconsultas



Subconsultas utilizando IN

El predicado IN permite determinar si los valores en la columna especificada de una tabla están contenidos en una lista definida o contenidos dentro de otra tabla.

Se debe especificar el nombre de la columna, la palabra clave IN y una subconsulta, que haga referencia a la segunda tabla. Si el valor de la columna coincide con uno de los valores en la lista o en los resultados de la subconsulta, el predicado se evalúa como verdadero y la fila es arrojada en los resultados de la consulta.

```
SELECT seleccion FROM tabla1 WHERE campoComun IN (SELECT  
campoComun FROM tabla2 WHERE condicion)
```



Subconsultas utilizando IN

Cuando se incluye una subconsulta en un predicado IN, la cláusula SELECT de la subconsulta debe arrojar solamente una columna de datos. Si se especifica más de una columna en el conjunto de resultados o se especifica un asterisco, se recibirá un error.



Subconsultas utilizando IN

Utilizaremos las siguientes tablas para ver un ejemplo del uso la sentencia IN

tc_caja

Nombre	Tipo	Cotejamiento	Atributos
id_caja 🔑	int(11)		UNSIGNED
id_sucursal	tinyint(4)		UNSIGNED
usuario_inicio	tinyint(4)		UNSIGNED
usuario_cierre	tinyint(4)		UNSIGNED
estado	tinyint(4)		UNSIGNED
cantidad_inicial	float		
cantidad_cierre	float		
observaciones	tinytext	latin1_swedish_ci	
fecha_inicio	timestamp		
fecha_cierre	timestamp		

tc_sucursal

Nombre	Tipo	Cotejamiento	Atributos
id_sucursal 🔑	tinyint(4)		UNSIGNED
activo	tinyint(4)		UNSIGNED
nombre_sucursal 🔑	varchar(60)	latin1_swedish_ci	
direccion	varchar(170)	latin1_swedish_ci	
colonia	varchar(120)	latin1_swedish_ci	
ciudad	varchar(100)	latin1_swedish_ci	
fecha_registro	timestamp		

Las tablas tienen en común el campo id_sucursal



Subconsultas utilizando IN

Ejemplo: Sabemos que en la tabla de sucursales existe un registro marcado como activo igual a cero. Obtener la suma de la cantidad cierre de la sucursal que tenga un status activo igual a cero.

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,0067 segundos.)

```
SELECT SUM(cantidad_cierre) FROM tc_caja WHERE id_sucursal IN (SELECT id_sucursal FROM tc_sucursal WHERE activo=0)
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]



Subconsultas utilizando NOT IN

Al igual que muchos otros predicados, el predicado IN permite especificar el inverso de una condición al utilizar la palabra clave NOT. Por ejemplo: Obtendremos la suma de la caja cierre de todas aquellas sucursales que no tengan el estado activo marcado como cero.

✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,0033 segundos.)

```
SELECT SUM(cantidad_cierre) FROM tc_caja WHERE id_sucursal NOT IN (SELECT id_sucursal  
FROM tc_sucursal WHERE activo=0)
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]



Subconsultas utilizando EXISTS

A pesar de ser similar al predicado IN, el predicado EXISTS tiene un enfoque ligeramente diferente. Está dedicado únicamente a determinar si la subconsulta arroja alguna fila o no. Si ésta arroja una o más filas, el predicado se evalúa como verdadero; de otra manera, el predicado se evalúa como falso.

```
SELECT seleccion FROM tabla1 WHERE EXISTS (SELECT seleccion  
FROM tabla2 WHERE condición tabla1 y tabla2)
```

Para que la subconsulta sea un valor real (y el predicado EXISTS también), debe incluir un predicado que coincida con dos columnas en diferentes tablas.



Subconsultas utilizando EXISTS

Utilizaremos las siguientes tablas para ver un ejemplo del uso la sentencia EXISTS

tc_caja

Nombre	Tipo	Cotejamiento	Atributos
id_caja 🔑	int(11)		UNSIGNED
id_sucursal	tinyint(4)		UNSIGNED
usuario_inicio	tinyint(4)		UNSIGNED
usuario_cierre	tinyint(4)		UNSIGNED
estado	tinyint(4)		UNSIGNED
cantidad_inicial	float		
cantidad_cierre	float		
observaciones	tinytext	latin1_swedish_ci	
fecha_inicio	timestamp		
fecha_cierre	timestamp		

tc_sucursal

Nombre	Tipo	Cotejamiento	Atributos
id_sucursal 🔑	tinyint(4)		UNSIGNED
activo	tinyint(4)		UNSIGNED
nombre_sucursal 🔑	varchar(60)	latin1_swedish_ci	
direccion	varchar(170)	latin1_swedish_ci	
colonia	varchar(120)	latin1_swedish_ci	
ciudad	varchar(100)	latin1_swedish_ci	
fecha_registro	timestamp		

Las tablas tienen en común el campo id_sucursal



Subconsultas utilizando EXISTS

Ejemplo1: Mostrar una lista sin repeticiones, de los usuarios que iniciaron caja en aquellas sucursales que en la tabla sucursales tengan el status activo igual a cero.

✓ Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0056 segundos.)

```
SELECT DISTINCT `usuario_inicio` FROM `tc_caja` WHERE EXISTS (SELECT * FROM tc_sucursal  
WHERE tc_caja.id_sucursal=tc_sucursal.id_sucursal AND tc_sucursal.activo=0)
```

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Como si existen sucursales con estatus activo igual a cero en la tabla sucursales, la subconsulta es verdadera y entonces la consulta general devolverá el id de los usuarios que se solicita.



Subconsultas utilizando EXISTS

Ejemplo2: Mostrar una lista sin repeticiones, de los usuarios que iniciaron caja en aquellas sucursales que en la tabla sucursales tengan el status activo igual a dos.

✓ MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0057 segundos.)

```
SELECT DISTINCT `usuario_inicio` FROM `tc_caja` WHERE EXISTS (SELECT * FROM tc_sucursal  
WHERE tc_caja.id_sucursal=tc_sucursal.id_sucursal AND tc_sucursal.activo=2)
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

Como no existen sucursales con estatus activo igual a dos en la tabla sucursales, la subconsulta es falsa y entonces la consulta general devolverá una lista vacía.



Subconsultas utilizando NOT EXISTS

El predicado EXISTS, como se puede esperar, permite utilizar el inverso de la condición del predicado utilizando la palabra clave NOT:

```
SELECT seleccion FROM tabla1 WHERE NOT EXISTS (SELECT  
seleccion FROM tabla2 WHERE condición tabla1 y tabla2)
```

Para que la subconsulta sea un valor real (y el predicado EXISTS también), debe incluir un predicado que coincida con dos columnas en diferentes tablas.



Subconsultas utilizando NOT EXISTS

Ejemplo: Mostrar una lista sin repeticiones, de los usuarios que iniciaron caja en aquellas sucursales que en la tabla sucursales no tengan el status activo igual a uno.

✓ Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0043 segundos.)

```
SELECT DISTINCT `usuario_inicio` FROM `tc_caja` WHERE NOT EXISTS (SELECT * FROM  
tc_sucursal WHERE tc_caja.id_sucursal=tc_sucursal.id_sucursal AND tc_sucursal.activo=1)
```

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

La subconsulta es verdadera (si existen sucursales con status activo igual a uno). Pero al utilizar NOT EXISTS la consulta general ignorará a las sucursales con status activo igual a uno.



