

## Ejemplos de modificación y borrado de datos

Hasta este módulo hemos abarcado de las operaciones básicas de CRUD (Create, Read, Update, Delete/Destroy), el primer gran grupo Crear y Leer. Debemos comprender los siguientes que nos permitirán avanzar en el uso de las operaciones de la persistencia de datos en las nuestras bases. Arrancamos por ver UPDATE o Actualización que nos permitirá actualizar datos de nuestras bases.

### Actualizando datos con la sentencia Update:

La sentencia UPDATE es una de las más sencillas en su lógica, pero al igual que DELETE (la veremos en el próximo módulo) conllevan un riesgo si no las utilizamos correctamente o tomamos recaudos suficientes.

La sentencia nos permite actualizar los valores que se encuentran guardados en las columnas (atributos) de una tabla. Puede actualizar uno, muchos o incluso todos los registros (filas) de la tabla va a depender de como la utilicemos y las condiciones que se incluyan.

- Analicemos cuál es la estructura de la sentencia:

**UPDATE** [tabla](1)

**SET** campo1 = X1, campo2 = x2, ..... campoN = xn. (2)

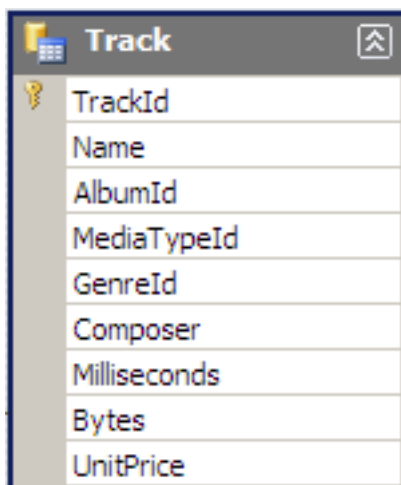
**WHERE** [condiciones] ; (3)

1. En **[tabla]** irá el nombre de la tabla que deseamos actualizar.
2. Luego de la cláusula SET ingresamos los campos (columnas/atributos) que deseamos modificar sus valores de la tabla elegida. A continuación de cada campo poniendo un igual "=" incorporamos el valor nuevo. Se permite ingresar operaciones aritméticas e incluso hacer referencia al mismo campo u otros campos de la misma tabla (Ej.: campo1 = campo1 + 1)
3. A continuación de WHERE incorporamos todas las condiciones necesarias para limitar a que el UPDATE se realice sobre los registros elegidos. Podemos afectar toda una tabla al no ingresar condiciones ni la cláusula WHERE, es decir que la cláusula es optativa

Veamos algunos ejemplos de cómo usar la sentencia.

#### Ejemplo 1

Supongamos que tenemos nuestra tabla de canciones que tiene este diseño:

El diagrama muestra la estructura de la tabla 'Track'. El título 'Track' está en la parte superior. A la izquierda de la lista de campos hay un icono de llave, indicando que 'TrackId' es la clave primaria. Los campos listados son: TrackId, Name, AlbumId, MediaTypeId, GenreId, Composer, Milliseconds, Bytes y UnitPrice.

Track	
TrackId	
Name	
AlbumId	
MediaTypeId	
GenreId	
Composer	
Milliseconds	
Bytes	
UnitPrice	

Requerimos realizar un incremento del 10% en el precio (UnitPrice) de todas las canciones (Track)

Reemplazamos los datos en el ejemplo de sentencia:

1. Incorporamos la tabla a modificar en el UPDATE

**UPDATE** Track

2. Agregamos los campos (atributos) a modificar, separando por comas en caso de que fuera más de uno, con los valores nuevos y los cálculos que fueran necesarios.  
En este caso para aumentar el 10% de un valor podemos realizarlo multiplicando por el valor por 1,10.

**UPDATE** Track

**SET** UnitPrice = UnitPrice \* 1.10

Podemos hacer referencia al campo, el motor realiza el cálculo con el valor anterior y luego lo guarda con el nuevo valor.

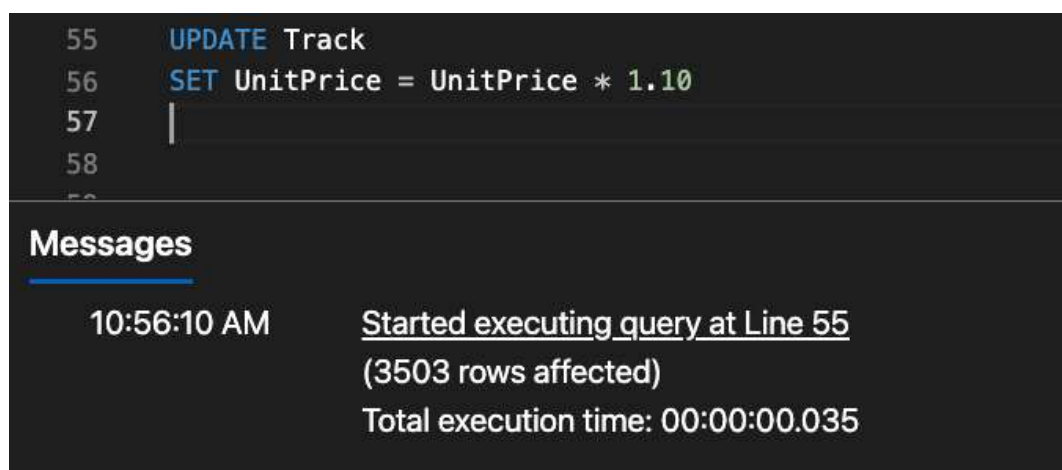
3. Debemos analizar si es necesario incorporar condiciones, en este caso se aclara que se debe modificar todas las canciones, es decir todos los registros de la tabla, dado esto no será necesario incorporar la cláusula WHERE.

Y así queda nuestra sentencia de modificación:

**UPDATE** Track

**SET** UnitPrice = UnitPrice \* 1.10

Cuando la lanzamos (ejecutamos) en el motor nos va regresar la cantidad de registros afectados junto con el tiempo que le tomó ejecutar la sentencia, siempre que esté bien escrita la sentencia y no exista ningún error o restricción.



The screenshot displays a SQL query window with the following text:

```
55  UPDATE Track
56  SET UnitPrice = UnitPrice * 1.10
57  |
58
59
```

Below the query window, the 'Messages' pane shows the execution results:

**Messages**

10:56:10 AM	<u>Started executing query at Line 55</u> (3503 rows affected) Total execution time: 00:00:00.035
-------------	---

### Ejecución controlada de Updates:

No parece existir grandes complicaciones en ejecutar esta sentencia, pero ¿qué ocurre si tuvimos un error? o ¿cómo validamos que afectó correctamente los cambios que deseábamos hacer?

Para responder a ese interrogante voy a aconsejar que siempre realicemos una sentencia de SELECT previa y algunos pasos para poder guardarnos los datos que vamos a modificar.

Ponemos en práctica esta técnica en el Ejemplo 1:

La consulta quedaría de la siguiente forma, considerando que voy a modificar todos los registros.

**SELECT \***

**FROM** Track

	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	1	For Those About To Rock (Live)	1	1	1	Angus Young, Malcolm Young	343719	11178334	1.09
2	2	Balls to the Wall	2	2	1	NULL	342562	5518424	1.09
3	3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U...	238619	3998994	1.09
4	4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Die...	252051	4331779	1.09
5	5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	6290521	1.09
6	6	Put The Finger On You	1	1	1	Angus Young, Malcolm Young	285662	5713451	1.09
7	7	Let's Get It Up	1	1	1	Angus Young, Malcolm Young	233926	7636561	1.09
8	8	Inject The Venom	1	1	1	Angus Young, Malcolm Young	218834	6852868	1.09
9	9	Snowballed	1	1	1	Angus Young, Malcolm Young	283182	6599424	1.09
10	10	Evil Walks	1	1	1	Angus Young, Malcolm Young	283497	8611245	1.09
11	11	C.O.D.	1	1	1	Angus Young, Malcolm Young	199836	6566314	1.09
12	12	Breaking The Rules	1	1	1	Angus Young, Malcolm Young	353288	8568868	1.09

Y para guardarnos la información, la mayoría de los clientes de SQL permiten que nos guardemos la información en varios formatos como CSV, Excel y algunos incluso la guardan en formato de creación (INSERT). En mi caso tengo un listado de opciones disponibles:



Guardar como CSV (Valores separados por comas)

Guardar como Excel

Guardar como JSON

Guardar como XML

De esta forma podemos guardar la información previa a la ejecución de un UPDATE o un DELETE permitiendo que en caso de error involuntario o necesidad de volver atrás nos sea un poco más fácil.

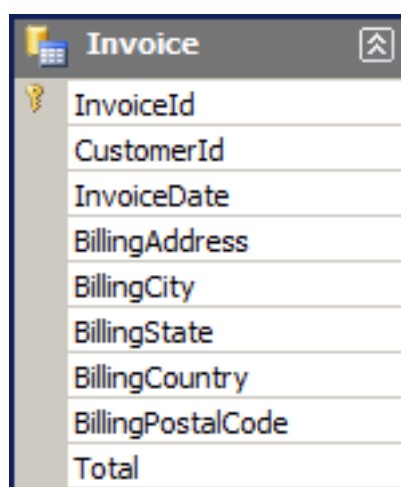
Vamos a utilizar esta técnica en todos los próximos ejemplos que veamos.

Actualizaciones utilizando condiciones dentro de la misma tabla:

Veamos ejemplos donde incorporamos condiciones a nivel de registro sobre la misma tabla que se va a actualizar, la intención es afectar con los cambios a menos registros.

### Ejemplo 2

Suponemos la existencia de una tabla facturas (Invoice):



Invoice	
?	InvoiceId
	CustomerId
	InvoiceDate
	BillingAddress
	BillingCity
	BillingState
	BillingCountry
	BillingPostalCode
	Total

Deseamos realizar una modificación en las facturas (invoice) emitidas el 28 de enero del 2021 (InvoiceDate), restándole 1,10 del total a cada una.

Vamos a trabajar primero en la consulta para saber cuáles son las facturas que se van a actualizar.

1. Definimos el dominio de mi consulta agregando en el FROM las tablas involucradas (con los Joins que fueran necesarios):

**FROM** Invoice

2. Verificamos si debo agregar condiciones a nivel de registro en el WHERE (si no tengo no lo agrego). En este caso nos solicitan las facturas emitidas en el 28 de enero del 2021, es decir que la fecha (InvoiceDate) sea igual a 28/01/2021.

WHERE InvoiceDate = '2021-01-28'

3. Por último, elegimos que columnas vamos a mostrar para agregarlas en el SELECT. En nuestro caso y dado el consejo dado nos conviene mostrar todos los datos, así que nos valemos del asterisco que nos va a permitir esto.

SELECT \*

Quedando nuestra consulta de esta forma:

SELECT \*

FROM Invoice

WHERE InvoiceDate = '2021-01-28'

Y estos serían los registros que van a ser modificados:

Results	Messages	InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
1		336	54	2021-01-28 00:00:00.000	110 Raeburn Pl	Edinburgh	NULL	United Kingdom	EH4 1HH	1.98
2		337	56	2021-01-28 00:00:00.000	387 Macacha Güemes	Buenos Aires	NULL	Argentina	1106	1.98

Ahora podemos trabajar con el UPDATE pero ya podemos trasladar algunas de las cosas encontradas en el SELECT facilitándonos la creación de la sentencia.

1. Incorporamos la tabla a modificar en el UPDATE. Ya sabemos que es Invoice al igual que en el SELECT.

UPDATE Invoice

2. Agregamos los campos (atributos) a modificar, separando por comas en caso de que fuera más de uno, con los valores nuevos y los cálculos que fueran necesarios. En este caso sería Total restándole 1,10, quedando:

UPDATE Invoice

SET Total = Total - 1.10

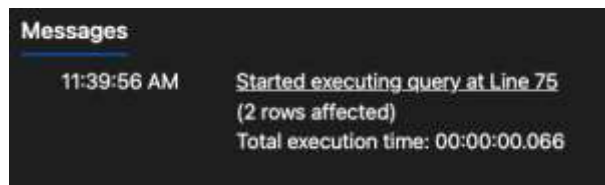
3. Debemos analizar si es necesario incorporar condiciones. Y trasladamos exactamente las mismas que utilizamos en el SELECT.

UPDATE Invoice

SET Total = Total - 1.10

WHERE InvoiceDate = '2021-01-28'

Y el resultado de ejecutar nuestra sentencia es:



Podemos ejecutar nuevamente el SELECT para verificar si el cambio se realizó correctamente y comprobar que la modificación sobre el campo **Invoice.Total**

	InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
1	336	54	2021-01-28 00:00:00.000	110 Raeburn Pl	Edinburgh	NULL	United Kingdom	EH4 1HH	0.88
2	337	56	2021-01-28 00:00:00.000	307 Macacha Gómez	Buenos Aires	NULL	Argentina	1106	0.88

Ahora veremos un ejemplo donde debemos modificar más de un campo y agregar más de una condición.

### Ejemplo 3

Dada la siguiente tabla donde se guarda el detalle de lo facturado (InvoiceLine):

InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

Buscamos modificar el detalle registrado para la factura con id (InvoiceId) 336 para la canción con id (TrackID) 626, se debe poner en cantidad 3 y el precio unitario incrementarlo en 1 (sumarle 1).

Arrancamos por la consulta (SELECT) como hicimos en el anterior ejemplo:

1. Definimos el dominio de mi consulta agregando en el FROM las tablas involucradas (con los Joins que fueran necesarios). En este caso sería InvoiceLine:

**FROM** InvoiceLine

2. Verificamos si debo agregar condiciones a nivel de registro en el WHERE (si no tengo no lo agrego). En este caso tenemos 2 condiciones por un lado el id de factura y por otro el id de canción, las combinamos por medio de un AND.

**WHERE** InvoiceId = 336

**AND** TrackID = 626

3. Por último, elegimos que columnas vamos a mostrar para agregarlas en el SELECT. Al igual que el ejemplo anterior mostramos todos los datos.

**SELECT** \*

Quedando nuestra consulta de esta forma:

**SELECT** \*

**FROM** InvoiceLine

**WHERE** InvoiceId = 336

**AND** TrackID = 626

Este sería el resultado de la consulta ejecutada:

```
86 SELECT *
87 FROM InvoiceLine
88 WHERE InvoiceId = 336
89 AND TrackID = 626
90
```

Results		Messages			
	InvoiceLineId	InvoiceId	TrackId	UnitPrice	Quantity
1	1823	336	626	0.99	1



Con los datos obtenidos del SELECT armamos el UPDATE.

1. Incorporamos la tabla a modificar en el UPDATE.

**UPDATE** InvoiceLine

2. Agregamos los campos (atributos) a modificar, separando por comas en caso de que fuera más de uno, con los valores nuevos y los cálculos que fueran necesarios. En este caso tenemos 2 campos por modificar, el precio (UnitPrice) y la cantidad (Quantity):

**UPDATE** InvoiceLine

**SET** UnitPrice = UnitPrice + 1, Quantity = 3

3. Debemos analizar si es necesario incorporar condiciones. Al igual que el ejemplo anterior nos traemos las mismas condiciones que realizamos en el WHERE.

**UPDATE** InvoiceLine

**SET** UnitPrice = UnitPrice + 1, Quantity = 3

**WHERE** InvoiceId = 336

**AND** TrackID = 626

Y el resultado de ejecutar nuestra sentencia es:

```
4:25:05 PM      Started executing query at Line 91
                  (1 row affected)
                  Total execution time: 00:00:00.052
```

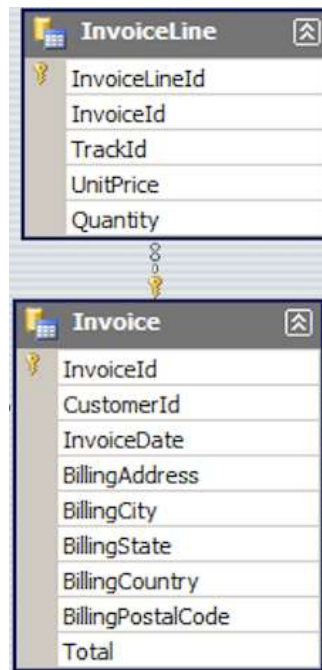
Y cuando volvemos a ejecutar el SELECT podemos comprobar que el cambio se realizó correctamente.

	InvoiceLineId ▾	InvoiceId ▾	TrackId ▾	UnitPrice ▾	Quantity ▾
1	1823	336	626	1.99	3

#### Ejemplo 4

Dado el ejemplo anterior y un poco en combinación con el otro vamos a realizar un UPDATE pero utilizando una subconsulta (subquery).

Dada la tabla Factura (Invoice) y Detalle de facturación (InvoiceLine):



Se desea que basado en el ejemplo 2, se actualice de la tabla factura el Total facturado (Total), sumando los detalles facturados. Recordamos que el id de factura es 336, comenzamos por las consultas:

Averiguamos cual es el total facturado, multiplicando el precio (UnitPrice) por la cantidad (Quantity):

```
SELECT UnitPrice * Quantity
```

```
FROM InvoiceLine
```

```
WHERE InvoiceId = 336
```

Con esta consulta obtenemos el total por línea de la factura:

```

96  Select UnitPrice * Quantity
97  From InvoiceLine
98  Where InvoiceId = 336
99

```

Results		Messages
	(No column name)	▼
1	5.97	
2	0.99	

Para poder sumarlo nos vamos a valer de una función de agregación, SUM, que nos permitirá obtener el sumario de las columnas de la siguiente forma:

**SELECT SUM**(UnitPrice \* Quantity)

**FROM** InvoiceLine

**WHERE** InvoiceId = 336

```

96  Select sum(UnitPrice * Quantity)
97  From InvoiceLine
98  Where InvoiceId = 336
99
100

```

Results		Messages
	(No column name)	▼
1	6.96	

El siguiente paso será crear la consulta sobre el registro donde vamos a realizar la modificación

Siguiendo los pasos que aprendimos, donde buscamos el dominio, las condiciones y que mostrar obtenemos:

**SELECT \***

**FROM** Invoice

**WHERE** InvoiceId = 336

InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
336	54	2021-01-28 00:00:00.000	110 Raeburn Pl	Edinburgh	NULL	United Kingdom	EH4 1HH	0.88

Y ahora nos enfocamos en el UPDATE con los datos de la anterior consulta y utilizando la consulta previa donde encontramos el sumario.

UPDATE Invoice

SET Total = (SELECT SUM(UnitPrice \* Quantity)

FROM InvoiceLine

WHERE InvoiceId = 336)

WHERE InvoiceId = 336

Como se puede ver podemos incorporar el valor de otra tabla incorporando entre paréntesis la consulta, la única condición es que retorne un único valor.

Como paso siguiente ejecutamos la consulta sobre la factura para verificar que se realizó correctamente:

SELECT \*

FROM Invoice

WHERE InvoiceId = 336

Y el resultado es:

InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
336	54	2021-01-28 00:00:00.000	110 Raeburn Pl	Edinburgh	NULL	United Kingdom	EH4 1HH	6.96

Con esto hemos visto lo básico para poder realizar diferentes tipos de modificaciones a los datos que tengamos guardados en nuestra base. Es importante recordar que las actualizaciones haciendo uso del UPDATE pueden ser algo riesgosas, pero el riesgo disminuye considerablemente si tomas los recaudos necesarios procurando guardar siempre una “foto” de cómo estaban mis registros previos ejecutando una consulta (SELECT).