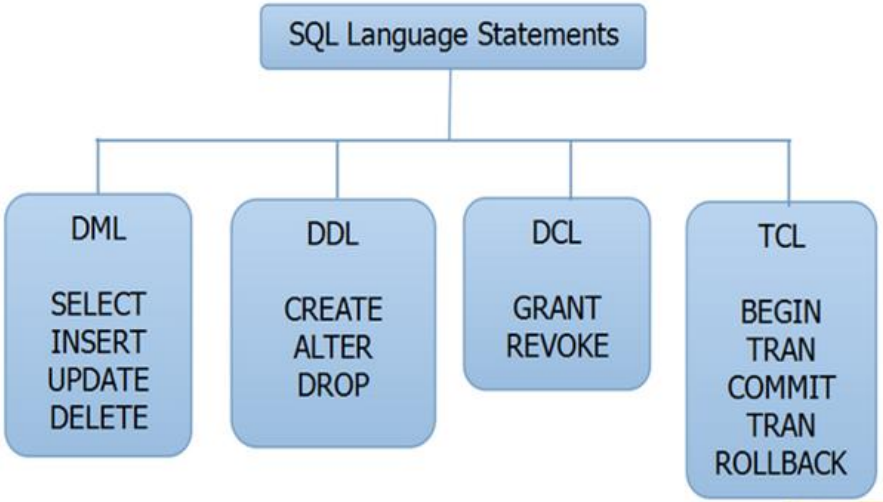
Tema III. Lenguaje SQL

- Introducción.
- Sentencias SQL de consulta.
- Otras sentencias de interés.
- Sentencias SQL de creación de tablas.
- Sentencias SQL para añadir registros.
- Sentencias SQL para borrar registros.
- Sentencias SQL para actualizar registros.



Fuente: https://csharpcorner-mindcrackerinc.netdna-ssl.com/UploadFile/BlogImages/01032017000551AM/image002.png

Introducción

El lenguaje de consulta estructurado **SQL** (Structured Query Language) es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos.

El científico Edgar Frank Codd (1923–2003) fue quien propuso un modelo relacional para las bases de datos y creó un sublenguaje para acceder a los datos a partir del cálculo de predicados. En base al trabajo de Codd, IBM (International Business Machines) definió el lenguaje conocido como Structured English Query Language (SEQUEL). El SEQUEL se considera el antecesor de SQL, un lenguaje de cuarta generación que se estandarizó en 1986.

Fuente: https://sequel.jeremyevans.net/images/ruby-sequel.png/

Sentencias SQL de consulta

- **SELECT:** La sentencia SELECT selecciona una serie de atributos de una o varias tablas, que cumplen unas determinadas condiciones. La sentencia SELECT permite operaciones avanzadas como ordenar el resultado. La sintaxis general es la siguiente:

SELECT atributos **FROM** tablas **WHERE** condiciones

Tabla city de la base de datos world:

Field	Schema	Table	Type	Character Set	Display Size	Precision
ID	world	city	INT	binary	11	4
Name	world	city	CHAR	utf8mb4	35	33
CountryCode	world	city	CHAR	utf8mb4	3	3
District	world	city	CHAR	utf8mb4	20	22
Population	world	city	INT	binary	11	7

Sentencias SQL de consulta

- Ejemplo 1: "SELECT * FROM world.city"

Selecciona todos los atributos de la tabla **city**, de la base de datos **world**. El * indica que devuelve todos los atributos de las tablas implicadas.

# ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701

Se muestran un total de once resultados de los 4079 registros disponibles

Sentencias SQL de consulta

- **Ejemplo 11:** "**SELECT** ID, city.Name **FROM** world.city"

Selecciona el atributo ***ID*** y el atributo ***Name*** (obsérvese que en este caso, nos referimos a él anteponiendo el nombre de la tabla. Es útil cuando dos tablas tienen atributos con el mismo nombre) de la tabla ***city,*** de la base de datos ***world***.

# ID	Name
1	Kabul
2	Qandahar
3	Herat
4	Mazar-e-Sharif
5	Amsterdam
6	Rotterdam
7	Haag
8	Utrecht
9	Eindhoven
10	Tilburg
11	Groningen

Se muestran un total de once resultados de los 4079 registros disponibles

Sentencias SQL de consulta

- Ejemplo III: "**SELECT** ID, Name **FROM** world.city **LIMIT** 5"

Selecciona el atributo ***ID*** y el atributo ***Name***, de la tabla ***city*** y de la base de datos ***world***. En este caso, limitamos el resultado a 5 registros. Esto es muy interesante cuando el número de registros devueltos es muy grande o nos interesa sólo el primer registro que cumple una condición.

# ID	Name
1	Kabul
2	Qandahar
3	Herat
4	Mazar-e-Sharif
5	Amsterdam

Se muestran un total de 5 registros

Sentencias SQL de consulta

- Ejemplo IV: "**SELECT** * **FROM** world.city **WHERE** CountryCOde = 'ESP' **LIMIT** 10"

Selecciona todos los atributos, de la tabla ***city*** y de la base de datos ***world***. En este caso, limitamos el resultado a 10 registros y mediante la cláusula **WHERE** indicamos que sólo nos interesan aquellos cuyo código de país (***CountryCOde***) es "ESP".

# ID	Name	CountryCode	District	Population
653	Madrid	ESP	Madrid	2879052
654	Barcelona	ESP	Katalonia	1503451
655	Valencia	ESP	Valencia	739412
656	Sevilla	ESP	Andalusia	701927
657	Zaragoza	ESP	Aragonia	603367
658	MÃ¡laga	ESP	Andalusia	530553
659	Bilbao	ESP	Baskimaa	357589
660	Las Palmas de Gran Canaria	ESP	Canary Islands	354757
661	Murcia	ESP	Murcia	353504
662	Palma de Mallorca	ESP	Balears	326993

Se muestran un total de 10 registros

Sentencias SQL de consulta

- **Ejemplo V:** "**SELECT** * **FROM** world.city **WHERE** CountryCOde = 'ESP' **AND** population > 500000 **LIMIT** 10"

Selecciona todos los atributos, de la tabla ***city*** y de la base de datos ***world***. En este caso, limitamos el resultado a 10 registros y mediante la cláusula **WHERE** indicamos que sólo nos interesan aquellos cuyo código de país (***CountryCOde***) es "ESP" y además cuya población (population) es mayor a 500.000 (cláusula ***AND***).

# ID	Name	CountryCode	District	Population
653	Madrid	ESP	Madrid	2879052
654	Barcelona	ESP	Katalonia	1503451
655	Valencia	ESP	Valencia	739412
656	Sevilla	ESP	Andalusia	701927
657	Zaragoza	ESP	Aragonia	603367
658	MÃ¡laga	ESP	Andalusia	530553

Sentencias SQL de consulta

- **Ejemplo VI:** "**SELECT** * **FROM** world.city **WHERE** CountryCOde = 'ESP' **AND** population > 500000 **ORDER BY** Name **LIMIT** 10"

Selecciona todos los atributos, de la tabla ***city*** y de la base de datos ***world***. En este caso, limitamos el resultado a 10 registros y mediante la cláusula **WHERE** indicamos que sólo nos interesan aquellos cuyo código de país (***CountryCOde***) es "ESP" y además cuya población (population) es mayor a 500.000 (cláusula ***AND***). Por último, ordenamos por el atributo ***Name*** mediante la cláusula ***ORDER BY***.

# ID	Name	CountryCode	District	Population
654	Barcelona	ESP	Katalonia	1503451
653	Madrid	ESP	Madrid	2879052
658	MÃ¡laga	ESP	Andalusia	530553
656	Sevilla	ESP	Andalusia	701927
655	Valencia	ESP	Valencia	739412
657	Zaragoza	ESP	Aragonia	603367

Sentencias SQL de consulta

- **Ejemplo VII:** "**SELECT** * **FROM** world.city **WHERE** CountryCOde = 'ESP' **AND** population > 500000 **ORDER BY** Name **DESC LIMIT** 10"

Selecciona todos los atributos, de la tabla ***city*** y de la base de datos ***world***. En este caso, limitamos el resultado a 10 registros y mediante la cláusula **WHERE** indicamos que sólo nos interesan aquellos cuyo código de país (***CountryCOde***) es "ESP" y además cuya población (population) es mayor a 500.000 (cláusula ***AND***). Por último, ordenamos por el atributo ***Name*** al revés (***DESC***) mediante la cláusula ***ORDER BY***.

# ID	Name	CountryCode	District	Population
657	Zaragoza	ESP	Aragonia	603367
655	Valencia	ESP	Valencia	739412
656	Sevilla	ESP	Andalusia	701927
658	MÃ¡laga	ESP	Andalusia	530553
653	Madrid	ESP	Madrid	2879052
654	Barcelona	ESP	Katalonia	1503451

Sentencias SQL de consulta

- Ejemplo VIII: "SELECT COUNT(*) FROM world.city"

Cuenta todos los registros de la tabla ***city***, de la base de datos ***world***.

count(*)
4079

- Ejemplo IX: "SELECT COUNT(*) AS total **FROM** world.city"

Cuenta todos los registros de la tabla ***city***, de la base de datos ***world*** y almacena el resultado en un atributo nuevo llamado ***total***, mediante la cláusula ***AS***.

total
4079

Sentencias SQL de consulta

- Ejemplo X: "**SELECT COUNT(DISTINCT**(name)**) AS** total **FROM** world.city"

Cuenta todos los registros que tengan el campo ***name*** diferente, de la tabla ***city***, de la base de datos ***world,*** y almacena el resultado en un nuevo atributo ***total***.

total
4001

- Ejemplo XI: "**SELECT DISTINCT**(name) **AS** nombres_diferentes **FROM** world.city **ORDER BY** name **LIMIT** 10"

Devuelve todos los registros que tengan el campo ***name*** diferente, de la tabla ***city***, de la base de datos ***world,*** y almacena el resultado en un nuevo atributo ***nombres_diferentes***. Limita el número de registros devueltos a 10. Ordena por ***nombre***.

Sentencias SQL de consulta

nombres_diferentes
A CoruÃ±a (La CoruÃ±a)
Aachen
Aalborg
Aba
Abadan
Abaetetuba
Abakan
Abbotsford
Abeokuta
Aberdeen

- **Ejemplo XII:** "**SELECT** city.ID, city.Name, country.code, country.Name **FROM** world.city, world.country **WHERE** country.code = city.CountryCode **ORDER BY** country.Name, city.Name **LIMIT** 10"

Devuelve los atributos ***ID y Name*** de la tabla ***city*** y los atributos ***code y Name*** de la tabla ***country***. Enlaza ambas tablas a través de los atributos ***country.code*** y ***city.CountryCode***. Limita el resultado a 10 registros y lo ordena por ***country.Name*** y ***city.Name***.

Sentencias SQL de consulta

# ID	Name	code	Name
3	Herat	AFG	Afghanistan
1	Kabul	AFG	Afghanistan
4	Mazar-e-Sharif	AFG	Afghanistan
2	Qandahar	AFG	Afghanistan
34	Tirana	ALB	Albania
35	Alger	DZA	Algeria
38	Annaba	DZA	Algeria
39	Batna	DZA	Algeria
49	BÃ©char	DZA	Algeria
45	BÃ©jaÃ¯a	DZA	Algeria

- **Ejemplo XIII:** "**SELECT** city.ID, city.Name, country.code, country.Name **FROM** world.city **INNER JOIN** world.country **ON** country.code = city.CountryCode **ORDER BY** country.Name, city.Name **LIMIT** 10"

En este caso, el resultado es el mismo pero incluimos la cláusula **INNER JOIN…ON** para relacionar ambas tablas.

Sentencias SQL de consulta

- **Ejemplo XIV:** "**SELECT** count(city.ID) **AS** numero_ciudades, country.code, country.Name **FROM** world.city **INNER JOIN** world.country **ON** country.code = city.CountryCode **GROUP BY** country.code **ORDER BY** country.Name **DESC,** city.Name **LIMIT 10**".

En este caso, la consulta o "query" nos devuelve el número de poblaciones en el nuevo atributo ***numero_ciudades***. Esto se consigue con la cláusula ***GROUP BY*** que agrupa todos los registros que tienen el atributo indicado en la cláusula, en este caso ***country.code***.

# numero_ciudades	code	Name
6	ZWE	Zimbabwe
7	ZMB	Zambia
8	YUG	Yugoslavia
6	YEM	Yemen
1	ESH	Western Sahara
1	WLF	Wallis and Futuna
1	VIR	Virgin Islands U.S.
1	VGB	Virgin Islands British
22	VNM	Vietnam
41	VEN	Venezuela

Sentencias SQL de consulta

- **Ejemplo XV:** "**SELECT** Name, Population **FROM** world.city **WHERE** ID **IN** (5, 23, 432, 2021)".

En este caso, la consulta o "query" nos devuelve los atributos ***Name*** y ***Population***, de la tabla ***city***, de la base de datos ***world*** cuyo ***ID*** sea 5, 23, 432 o 2021.

# Name	Population
Amsterdam	731200
Dordrecht	119811
Eunápolis	96610
Jining	265248

- **Ejemplo XVI:** "**SELECT** Name, Population **FROM** world.city **WHERE** Population **BETWEEN** 670000 **AND** 700000".

En este caso, la consulta o "query" nos devuelve los atributos ***Name*** y ***Population***, de la tabla ***city***, de la base de datos ***world*** cuyo atributo ***Population*** está entre los valores 670000 y 700000

Sentencias SQL de consulta

# Name	Population
Teresina	691942
Natal	688955
Bandar Lampung	680332
Gwalior	690765
Kermanshah	692986
Palermo	683794
Toronto	688275
Huainan	700000
Jixi	683885
Antananarivo	675669
Chihuahua	670208
Kano	674100
Tunis	690600

- **Ejemplo XVII:** "**SELECT SUM**(city.Population) **AS** total_poblacion **FROM** world.city **WHERE** CountryCode='ESP'".

En este caso, la consulta o "query" nos devuelve el numero total de habitantes de las ciudades de España. Esto se consigue mediante la cláusula ***SUM*** después del ***SELECT***. El resultado se guarda en un nuevo atributo llamado ***total_poblacion***.

total_poblacion
16669189

Sentencias SQL de consulta

- **Ejemplo XVIII:** "**SELECT * FROM** world.city **WHERE** Name **LIKE** '%MA%' **ORDER BY** name **LIMIT** 10"**.**

En este caso, la consulta o "query" nos devuelve 10 registros de la tabla ***city***, cuyo ***name*** contiene la secuencia de caracteres ***MA***. Esto se consigue mediante la cláusula ***LIKE '%MA%'*** después del ***SELECT***, siendo % un comodín. El resultado se ordena por ***Name***. Se muestran todos los atributos.

# ID	Name	CountryCode	District	Population
3392	Adiyaman	TUR	Adiyaman	141529
1168	Ahmadnagar	IND	Maharashtra	181339
2874	Ahmadpur East	PAK	Punjab	96000
1717	Aizuwakamatsu	JPN	Fukushima	119287
68	Ajman	ARE	Ajman	114395
1741	Akishima	JPN	Tokyo-to	106914
1375	al-Amara	IRQ	Maysan	208797
3177	al-Dammam	SAU	al-Sharqiya	482300
614	al-Mahallat al-Kubra	EGY	al-Gharbiya	395402
149	al-Manama	BHR	al-Manama	148000

Otras sentencias de interés

- Ejemplo XIX: "SHOW tables **IN** world**".**

Se muestran las tablas que existen en una determinada base de datos. En el ejemplo dado, se muestran las tablas que existen en la base de datos ***world***.

Tables_in_world
city
country
countrylanguage

- Ejemplo XX: "DESCRIBE world.city**".**

Se muestran los atributos de una tabla con sus características. En este ejemplo, se muestran los atributos de la tabla ***city*** de la base de datos ***world***.

# Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
Name	char(35)	NO			
CountryCode	char(3)	NO	MUL		
District	char(20)	NO			
Population	int(11)	NO		0	

Sentencias SQL de creación de tablas

- **CREATE:** La sentencia CREATE nos permite crear, en su forma más general, una tabla dándole un nombre y añadiendo una serie de atributos. Su forma habitual es:

CREATE TABLE nombre_tabla (atributo_1 tipo_1, atrinbuto_2 tipo_2, atributo_3 tipo_3, …, atributo_n tipo_n**);**

Ejemplo:

```
CREATE TABLE tasks
                    (task_id INT AUTO_INCREMENT,
                     title VARCHAR(255) NOT NULL,
                     start_date DATE,
                     due_date DATE,
                     status TINYINT NOT NULL,
                     priority TINYINT NOT NULL,
                     description TEXT,
                     PRIMARY KEY (task_id))
```

Sentencias SQL de creación de tablas

- En este ejemplo, se crea la tabla "***tasks***", con los atributos "***task_id***", "***title***", ***start_date***", "***due_date***", "***status***", "***priority***" y "***description***". El modificador "**AUTO_INCREMENT**" indica que se trata de un atributo que se autoincrementará de forma automática, cada vez que se añada un registro a la tabla. Es una forma de que no nos tengamos que preocupar de la clave principal. Mediante el modificador "**PRIMARY KEY**" indicamos el atributo que será clave principal. El modificador "**NOT NULL**" indica que no se permiten valores nulos en el atributo correspondiente. Cuando un atributo tiene el modificador "**NOT NULL**" y tratamos de introducir un valor nulo en ese atributo, el sistema nos devolverá un error. Lo mismo ocurrirá si tratamos de introducir un valor que ya existe en la clave principal.

Sentencias SQL de creación de tablas

- **ALTER TABLE:** La sentencia ALTER TABLE nos permite modificar, en su forma más general, una tabla que ya hemos creado previamente. Su forma habitual es:

```
ALTER TABLE nombre_tabla ACCIONES;
```

Ejemplo:

```
ALTER TABLE tasks
ADD COLUMN complete DECIMAL(2,1) NULL AFTER description;
```

En este ejemplo, modificamos la tabla "***tasks***", insertando un nuevo atributo llamado "***complete***", de tipo real (en MySql, el dos significa el total de dígitos que tiene el número real, sin decimales y el 1 significa el total de decimales. Es decir, en este caso 9,9 es válido, 10,3 es válido pero 100,4 no sería válido ni 27,43 tampoco). Este campo admite valores nulos (modificador "**NULL**") y se añade a continuación del atributo "***description***".

Sentencias SQL de creación de tablas

Ejemplos (borrado de un atributo, renombrar un atributo y añadir una clave primaria):

```
ALTER TABLE tasks DROP COLUMN description;
```

En este ejemplo, modificamos la tabla "***tasks***", eliminando el atributo "***description***" mediante "**DROP COLUMN**".

```
ALTER TABLE tasks RENAME TO work_items;
```

En este ejemplo, modificamos la tabla "***tasks***", renombrando su nombre a "**work_items**" mediante "**RENAME TO**".

```
ALTER TABLE t1 ADD PRIMARY KEY(id);
```

En este ejemplo, modificamos la tabla "***t1***", añadiendo una clave primaria mediante "**ADD PRIMARY KEY**". En este caso, el atributo "***id***" pasa a ser clave primaria.

Sentencias SQL de creación de tablas

Ejemplos (añadir un índice, añadir una clave foránea)**:**

```
ALTER TABLE tasks ADD INDEX indice_1 (start_date ASC, title DESC) ;
```

En este ejemplo, modificamos la tabla "***tasks***", añadiendo un índice nuevo "***indice_1***", mediante el modificador "**ADD INDEX**". Este índice estará compuesto por dos atributos, "***start_date***" y "***title***". En el caso de "***start_date***" el orden será ascendente y en el caso de "***title***" el orden será descendente.

```
ALTER TABLE products ADD FOREIGN KEY fk_vendor(vdr_id)
REFERENCES vendors(vdr_id)
```

En este ejemplo, modificamos la tabla "***products***", añadiendo una clave foránea mediante "**ADD FOREIGN KEY**". Esta clave foránea se llama "***fk_vendo***r" y afecta al atributo "***vdr_id***", que está en la tabla "***products***". Además, se crea una relación con el atributo "***vdr_id***" de la tabla "***vendors***" a través del modificador "**REFERENCES**".

Sentencias SQL de creación de tablas

Ejemplos (borrar claves primarias, claves foráneas, índices)**:**

```
ALTER TABLE products DROP FOREIGN KEY fk_vendor;
```

En este ejemplo, modificamos la tabla "***products***", eliminando la clave foránea "***fk_vendor***", mediante "**DROP FOREIGN KEY**". Importante saber que se borra la relación pero ni se borra el atributo.

```
ALTER TABLE products DROP PRIMARY KEY;
```

En este ejemplo, modificamos la tabla "***products***", eliminando la clave primaria mediante el modificador "**DROP PRIMARY KEY**". Igual que en el caso anterior, no se borran los atributos.

```
ALTER TABLE tasks DROP INDEX indice_1;
```

En este ejemplo, modificamos la tabla "***tasks***", eliminando el índice "***indice_1***" mediante el modificador "**DROP INDEX**". Tampoco se borran los atributos.

Sentencias SQL para añadir registros

- **INSERT:** La sentencia INSERT nos permite introducir un registro en una tabla. Su forma habitual es:

INSERT INTO nombre_table(atributo_1, atributo_2, …, atributo_n) **VALUES**(numero_1, 'cadena_1', …, numero_n);

Ejemplo:

INSERT INTO suppliers(name, phone, address)
VALUES('XYZ Corporation', '408-908-2476','4001 North 1st Street, San Jose, CA, USA')**;**

En este ejemplo, en los atributos "***name***","***phone***" y "***address***" de la tabla "***suppliers***", añadimos los valores "***XYZ Corporation***","***408-908-2476***" y "***'4001 North 1st Street, San Jose, CA, USA***" respectivamente. Los campos "**AUTO_INCREMENT**" que son claves principales, no es necesario añadirlos.

Sentencias SQL para añadir registros

Ejemplo:

```
INSERT INTO country(Code,Name) VALUES ('PPP','Prueba');
```

En este ejemplo, en los atributos "***Code***" y "***Name***" de la tabla "***country***", añadimos los valores "***PPP***" y "***"Prueba***" respectivamente.

```
INSERT INTO products VALUES
        (NULL, 'PEN', 'Pen Blue',  8000, 1.25),
        (NULL, 'PEN', 'Pen Red',  2000, 1.25),
        (NULL, 'PEN', 'Pen Yellow',  7000, 2.25),
        (NULL, 'PEN', 'Pen Black', 2500, 1.80);
```

En este ejemplo, introducimos varios registros en la tabla "***products***". Como se puede ver, los registros a introducir van entre paréntesis y se separan por la coma. En total, cuatro registros. Además en este caso, no se especifican los atributos en los que se insertarán los registros, con lo que afecta a todos los atributos.

Sentencias SQL para borrar registros

- **DELETE:** La sentencia DELETE nos permite borrar registros de una tabla. Su forma habitual es:

```
DELETE FROM tabla WHERE condicion;
```

Ejemplo:

```
DELETE FROM products WHERE name LIKE 'Pencil%'
```

En este ejemplo, se borran todos los registros de la tabla "***products***", cuyo atributo nombre empiece por los caracteres "***Pencil***". En resultado pueden ser uno o varios registros.

```
DELETE FROM products;
```

En este caso, se borrarán TODOS los registros de la tabla "***products***". Hay que tener mucho cuidado a la hora de usar "**DELETE**" sin la cláusula "**WHERE**", ya que se borran todos los registros de la tabla indicada.

Sentencias SQL para actualizar registros

- **UPDATE:** La sentencia UPDATE nos permite actualizar registros de una tabla. Su forma habitual es:

```
UPDATE tabla SET atributo = 'nuevo_valor' WHERE condicion;
```

Ejemplo:

```
UPDATE products SET quantity = 200 WHERE name = 'Pen Red';
```

En este ejemplo, se actualizan todos los registros de la tabla "***products***" cuyo atributo "***name***" sea "***Pen Red***". El atributo que se actualiza es "***quantity***" y el nuevo valor pasa a ser 200.

```
UPDATE products SET price = price * 1.1;
```

En este ejemplo, se actualizan todos los registros de la tabla "***products***" (al no tener cláusula **WHERE** se actualizan todos). El atributo que se actualiza es "***price***" y el nuevo valor pasa a ser lo que valía multiplicado por 1,1.

EJEMPLOS

EJEMPLO I: CREATE DATABASE southwind;

- Creamos una nueva base de datos que se va a llamar "***southwind***". En MySql las bases de datos se denominan "**SCHEMAS**".

EJEMPLO II: SHOW DATABASES;

En este caso, se muestran las bases de datos existentes en nuestro sistema.

Database
'information_schema'
'mysql'
'performance_schema'
'pruebas'
'sakila'
'southwind'
'sys'
'world'

EJEMPLOS

EJEMPLO III: USE southwind;

- Con la sentencia USE, seleccionamos la base de datos por defecto con la que vamos a trabajar. De esta forma, no tendremos que referirnos a ella de forma implícita.

EJEMPLO IV: CREATE TABLE IF NOT EXISTS

```
products (
        productID    INT UNSIGNED  NOT NULL AUTO_INCREMENT,
        productCode  CHAR(3)       NOT NULL DEFAULT '',
        name         VARCHAR(30)   NOT NULL DEFAULT '',
        quantity     INT UNSIGNED  NOT NULL DEFAULT 0,
        price        DECIMAL(7,2)  NOT NULL DEFAULT 99999.99,
        PRIMARY KEY  (productID)
       );
```

Creamos la tabla "***products***", siempre que no exista previamente. Se crean los atributos "***productID***", "***productCode***","***name***","***quantity***", y "***price***". La clave principal será el atributo "***productID***".

EJEMPLOS

EJEMPLO V: DESCRIBE products;

- Mostramos el diseño de la tabla "***products***" que acabamos de crear.

# Field	Type	Null	Key	Default	Extra
'productID'	'int(10) unsigned'	'NO'	'PRI'	NULL	'auto_increment'
'productCode'	'char(3)'	'NO'	''	''	''
'name'	'varchar(30)'	'NO'	''	''	''
'quantity'	'int(10) unsigned'	'NO'	''	'0'	''
'price'	'decimal(7,2)'	'NO'	''	'99999.99'	''

EJEMPLO VI: INSERT INTO products **VALUES** (1001, 'PEN', 'Pen Red', 5000, 1.23);

Insertamos un primer registro en la tabla "***products***". En este caso, no hemos especificado el nombre de los atributos con lo que se tiene que poner el valor de todos ellos. Con esta instrucción, nuestra tabla "***products***" cuenta ya con un registro.

EJEMPLOS

EJEMPLO VII: INSERT INTO products (productCode, name, quantity, price) **VALUES**

('PEC', 'Pencil 2B', 10000, 0.48),
('PEC', 'Pencil 2H', 8000, 0.49);

Ahora, introducimos dos registros más a la vez especificando los atributos y los valores que vamos a insertar. En este caso, al no especificar el atributo "***productID***" que es la clave principal y lleva el modificador "**AUTO_INCREMENT**", el motor crea el los registros de forma automática y asigna un valor al atributo "***productID***", sin tener que preocuparnos de que se repita.

EJEMPLO VIII: SELECT * FROM products;

# productID	productCode	name	quantity	price
'1001'	'PEN'	'Pen Red'	'5000'	'1.23'
'1002'	'PEC'	'Pencil 2B'	'10000'	'0.48'
'1003'	'PEC'	'Pencil 2H'	'8000'	'0.49'

EJEMPLOS

EJEMPLO IX: DELETE FROM products **WHERE** productID = '1002';

Borramos el registro cuyo atributo "***productID***" vale '***1002***'. Si después mostramos el contenido de la tabla con **SELECT** * **FROM** products, nos queda:

# productID	productCode	name	quantity	price
'1001'	'PEN'	'Pen Red'	'5000'	'1.23'
'1003'	'PEC'	'Pencil 2H'	'8000'	'0.49'

EJEMPLO X: UPDATE products **SET** price = price * 1.1 **WHERE** productID = '1001';

Actualizamos todos los registros de la tabla "***products***". El valor del atributo "***price***" se multiplica por 1,1, para el registro cuyo "***productID***" vale '***1001***'. El contenido de la tabla queda:

# productID	productCode	name	quantity	price
'1001'	'PEN'	'Pen Red'	'5000'	'1.35'
'1003'	'PEC'	'Pencil 2H'	'8000'	'0.49'

EJEMPLOS

EJEMPLO XI: CREATE TABLE suppliers (
supplierID INT UNSIGNED NOT NULL AUTO_INCREMENT,
name VARCHAR(30) NOT NULL DEFAULT '',
phone CHAR(8) NOT NULL DEFAULT '',
PRIMARY KEY (supplierID)
);

Creamos una nueva tabla "***suppliers***" cuya clave principal es "***supplierID***".

EJEMPLO XII: INSERT INTO suppliers **VALUE**
(501, 'ABC Traders', '88881111'),
(502, 'XYZ Company', '88882222'),
(503, 'QQ Corp', '88883333');

Introducimos tres registros en la tabla "suppliers". El contenido de la tabla queda:

# supplierID	name	phone
'501'	'ABC Traders'	'88881111'
'502'	'XYZ Company'	'88882222'
'503'	'QQ Corp'	'88883333'

EJEMPLOS

EJEMPLO XIII: ALTER TABLE products
ADD COLUMN supplierID **INT UNSIGNED NOT NULL;**

Modificamos la tabla "***products***" añadiendo el atributo "***supplierID***" que usaremos para relacionar ambas tablas (relación 1:N). Es decir, un producto es de un proveedor y un proveedor puede tener uno o varios productos.

EJEMPLO XIV: UPDATE products **SET** supplierID = 501**;**

Antes de crear la relación, actualizamos los valores del atributo "***supplierID***" que acabamos de crear en la tabla "***products***". ¿Por qué hacemos esto? El resultado queda:

# productID	productCode	name	quantity	price	supplierID
'1001'	'PEN'	'Pen Red'	'5000'	'1.35'	'501'
'1003'	'PEC'	'Pencil 2H'	'8000'	'0.49'	'501'

EJEMPLOS

EJEMPLO XV: ALTER TABLE products
ADD FOREIGN KEY (supplierID) **REFERENCES** suppliers (supplierID);

Añadimos una relación 1:N entre la tabla "***products***" y la tabla "***suppliers***". Se crea, como calve foránea, el atributo "***supplierID***" de la tabla "***products***". Si ahora hacemos un "**DESCRIBE products**" veremos que en la columna "***key***", aparece la palabra "***MUL***", que indica que es una clave foránea.

# Field	Type	Null	Key	Default	Extra
'productID'	'int(10) unsigned'	'NO'	'PRI'	NULL	'auto_increment'
'productCode'	'char(3)'	'NO'	''	''	''
'name'	'varchar(30)'	'NO'	''	''	''
'quantity'	'int(10) unsigned'	'NO'	''	'0'	''
'price'	'decimal(7,2)'	'NO'	''	'99999.99'	''
'supplierID'	'int(10) unsigned'	'NO'	'MUL'	NULL	''

EJEMPLOS

```
EJEMPLO XVI: CREATE TABLE dept_emp (
        emp_no     INT UNSIGNED  NOT NULL,
        dept_no    CHAR(4)       NOT NULL,
        from_date  DATE          NOT NULL,
        to_date    DATE          NOT NULL,
        INDEX      (emp_no),
        INDEX      (dept_no),
        FOREIGN KEY (emp_no)  REFERENCES employees (emp_no)
          ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (dept_no) REFERENCES departments (dept_no)
          ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY (emp_no, dept_no)
    );
```

Creamos la tabla "***dept_emp***" en la que la clave principal, es la suma de dos atributos "***emp_no***" y "***dept_no***". Además, creamos dos índices con nombre automático, uno para el atributo "***emp_no***" y otro para el atributo "***dept_no***". También creamos dos claves foráneas.

Bibliografía, fuentes y referencias

- https://svo.cab.inta-csic.es/docs/files/svo/Public/Meetings/SVO_thematic_network_First_School/sql_basico-061127.pdf.
- https://definicion.de/sql/.
- https://www.w3schools.com/sql.
- http://www.mysqltutorial.org/mysql-create-table/.
- http://www.ntu.edu.sg/home/ehchua/programming/sql/mysql_beginner.html.