

Uso del insert para cargar datos

Con lo visto hasta ahora podemos crear todas las tablas necesarias en nuestra base de datos. Ahora, debemos entender cómo incorporar datos a las tablas y algunos temas que debemos tener en cuenta al momento de hacerlo.

Incorporando registros:

Para poder incorporar registros en una tabla debemos tener claro cómo está conformada nuestra tabla, desde el nombre de las columnas y los tipos de datos.

La sentencia que vamos a utilizar es INSERT. Por definición, es bastante sencilla de usar.

INSERT [tabla] (campo1, campo2, campo3, ..., campoN)

VALUES (valor1, valor2, valor3, ..., valorN);

[tabla]: corresponde al nombre de la tabla donde vamos a ingresar el registro.

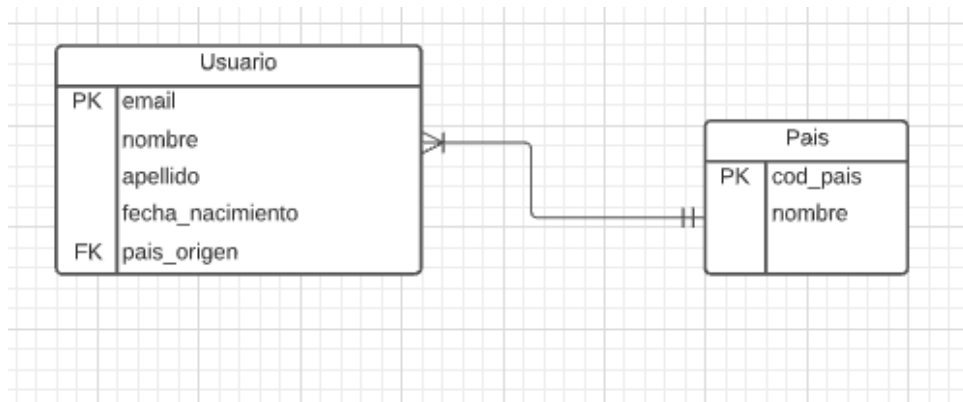
(campo1, campo2, campo3,...): Corresponde a los campos de la tabla, solo vamos a incorporar los campos que vamos a cargar con valores, aquellos que no vayamos a cargar no es necesario incorporarlos. Y se listan separándolos por “comas”. No olvidarse de aquellos campos que hayamos definido como obligatorios (NOT NULL al momento de definir el campo).

(valor1, valor2, valor3, ...): Luego de la palabra reservada VALUES y encerrado en paréntesis vamos a agregar los valores que corresponden a cada columna en el mismo orden que hayamos puesto las columnas previo a VALUES.

Es importante conocer la definición de nuestras tablas, qué tipo de datos se definieron para cada columna, ya que los valores alfanuméricos deben guardarse entre comillas. Los valores numéricos sin comillas y los valores con tipo fecha, dependerá del motor. Por ejemplo, en SQL Server se guardan entre comillas simples.

Veamos un ejemplo sencillo con nuestra tabla PAIS:

Previamente, veamos cómo estaba el DER de nuestra base.



Ahora que tenemos claro cómo es la estructura de nuestras tablas veamos la definición de País.

Column	Type	Default Value	Nullable
◆ cod_pais	int		NO
◆ nombre	varchar(20)		NO

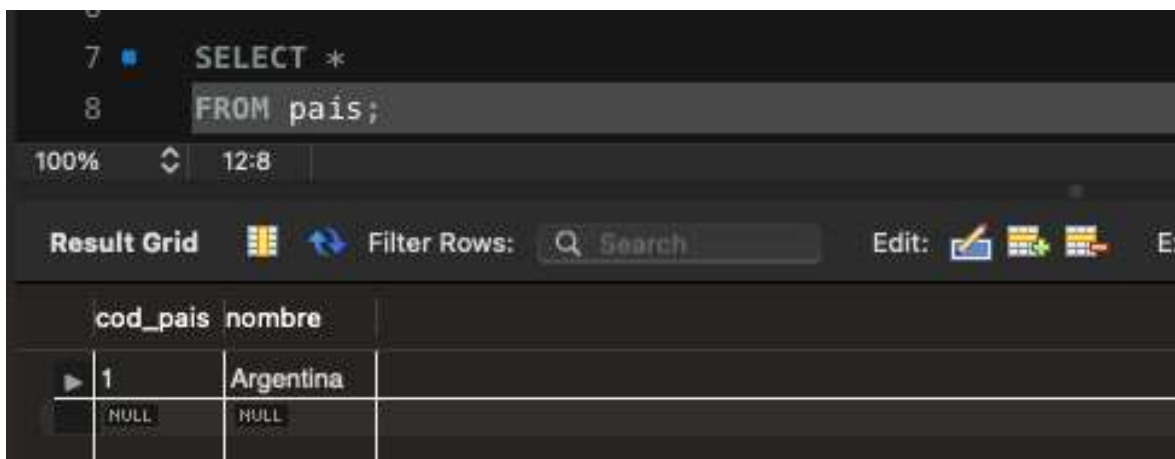
Como vemos en la definición que nos brinda el cliente de SQL (MySQL Workbench), tenemos dos columnas: cod_país que tiene como tipo de dato INT (del grupo de los números) representa enteros, y nombre que tiene el tipo de dato VARCHAR (20) por lo que solo vamos a poder usar 20 caracteres como máximo. En ambos casos no pueden estar vacíos, ya que en la columna Nullable figura un NO.

Armamos entonces el INSERT que corresponde para agregar el país ARGENTINA con código 1 (es mi primer registro).

```

MiBaseDeDatos.pais  Query 1
Limit to 1000 rows
1  INSERT INTO pais (cod_pais, nombre)
2  VALUES (1, 'Argentina');
3
  
```

Luego de ejecutar la consulta podremos validar el listado de datos de nuestra tabla que dará como resultado.



The screenshot shows a database query editor with a SQL query: `SELECT * FROM país;`. Below the query, there is a toolbar with options like 'Result Grid', 'Filter Rows', and 'Search'. The results are displayed in a grid with two columns: 'cod_pais' and 'nombre'.

	cod_pais	nombre
▶	1	Argentina
▶	NULL	NULL

Veamos otro ejemplo, pero utilizando algunos tipos de datos distintos. Ya pudimos ver que en el caso de caracteres va a ser necesario el uso de comillas simples, pero no en el caso de los enteros.

Ahora ingresamos los datos del primer usuario, pero antes validamos como está definida la tabla USUARIO.

Column	Type	Default Value	Nullable
◆ apellido	varchar(45)		YES
◆ email	varchar(250)		NO
◆ fecha_nac	date		YES
◆ nombre	varchar(45)		YES
◆ pais_origen	int		NO

Vemos que tiene el campo apellido con tipo de dato varchar (45) por lo cual no podremos ingresar un apellido que supere los 45 caracteres. Algo similar nos ocurre con nombre. Para

el caso de email, el máximo permitido será de 250 caracteres, pero a diferencia de apellido y nombre, no será posible no definirlo por estar en NO dentro de la columna Nullable, y algo similar ocurre con pais_origen (la clave foránea nunca debe estar en NULL, sin definición).

Veamos un ejemplo erróneo para ver qué ocurre:

```
10 INSERT INTO usuario (email, nombre, apellido, pais_origen)
11 VALUES ('mailprueba@mailprueba.com', NULL, NULL, NULL);
```

Probamos qué ocurre si decido poner en NULL un campo definido como NO Nullable.

```
* 7 20:22:58 INSERT INTO usuario (email, nombre, apellido, pais_o... Error Code: 1048. Column 'pais_origen' cannot be null 0.044 sec
```

El motor protege de incorporar un registro que no cumple con la definición regresando un error dando aviso que la columna pais_origen no puede ser NULL.

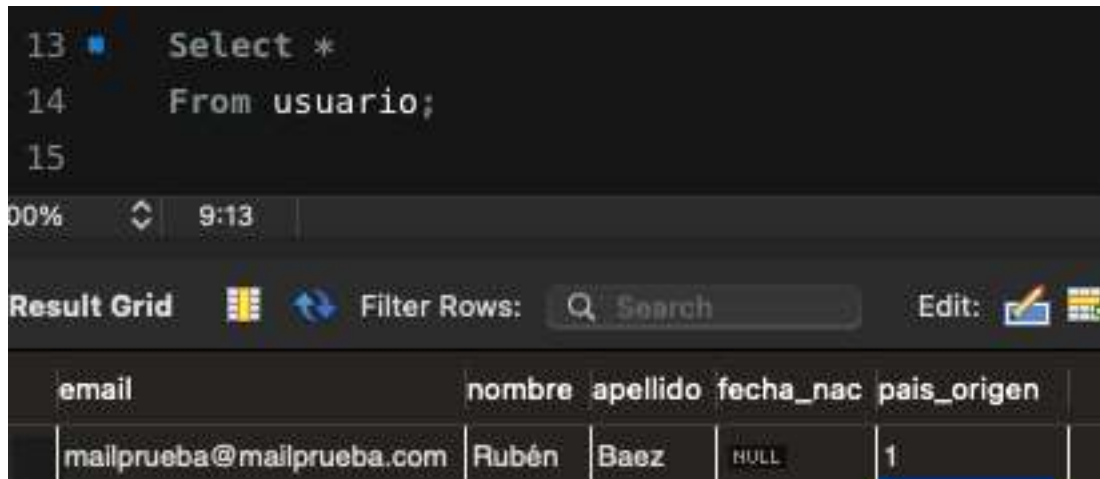
Ahora modifiquemos el INSERT para que pueda ingresar nuestro primer registro en usuario de forma correcta.

```
10 INSERT INTO usuario (email, nombre, apellido, pais_origen)
11 VALUES ('mailprueba@mailprueba.com', 'Rubén', 'Baez', 1);
```

Completamos el nombre y apellido. Adicionalmente, incorporamos el código 1 para país de origen. Sabemos que corresponde a Argentina ya que lo ingresamos previamente. En caso de que no supiéramos el código deberemos realizar una consulta para obtener los códigos disponibles (en el próximo módulo arrancamos con consultas).

```
✓ 8 20:28:28 INSERT INTO usuario (email, nombre, apellido, pais_o... 1 row(s) affected
```

El motor nos regresa el siguiente mensaje, confirmando que logró ejecutar correctamente nuestra sentencia, y podemos validar que se haya insertado el registro en la tabla.



The screenshot shows a SQL client window with a dark theme. The top pane contains a SQL query: `Select * From usuario;`. The bottom pane displays the 'Result Grid' with a table of results. The table has five columns: email, nombre, apellido, fecha_nac, and pais_origen. The first row contains the values: mailprueba@mailprueba.com, Rubén, Baez, NULL, and 1.

email	nombre	apellido	fecha_nac	pais_origen
mailprueba@mailprueba.com	Rubén	Baez	NULL	1


Cabe destacar que no ingresamos la fecha de nacimiento por eso figura como NULL.

Ahora probemos un ejemplo incorporando la fecha de nacimiento.

```
16 • INSERT INTO usuario (email, nombre, apellido, fecha_nac, pais_origen)
17   VALUES ('pedro.perez@mail.com', 'Pedro', 'Perez', '1981-09-16', 1);
```

En este ejemplo vamos a ingresar al usuario con nombre Pedro, apellido Perez, email pedro.perez@mail.com, país Argentina (código 1) y con fecha de nacimiento 16 de septiembre de 1981. Antes de seguir aclaro algo sobre las fechas, es necesario saber cómo están configuradas en el motor que estemos usando. Podés referirte al manual de tu motor para saber cómo viene por defecto. En mi caso, MySQL, viene en formato USA 'AAAA-mm-dd' (año-mes-día).

Ejecutamos y obtenemos la respuesta satisfactoria:



The screenshot shows the status bar of a SQL client. It displays a green checkmark, the number 10, the time 10:57:03, the command `INSERT INTO usuario (email, nombre, apellido, fecha_...`, and the message `1 row(s) affected`.

Quedando el registro correctamente insertado en nuestra tabla.



The screenshot shows a SQL client interface with a query editor and a result grid. The query editor contains the following SQL statement:

```
13 Select *
14 From usuario;
```

The result grid displays the following data:

	email	nombre	apellido	fecha_nac	pais_origen
▶	mailprueba@mailprueba.com	Rubén	Baez	NULL	1
	pedro.perez@mail.com	Pedro	Perez	1981-09-16	1

Insertando registros sin necesidad de aclarar las columnas

Hay una forma que nos permite SQL de ingresar registros sin aclarar, previo al VALUES, cuáles son las columnas que vamos a necesitar.

Directamente escribimos de la siguiente forma:

```
INSERT [tabla] VALUES (valor1, valor2, valor3, ..., valorN);
```

Pero para realizar esto, debemos conocer la definición de nuestra tabla, y con esto nos referimos a en qué orden el motor incorporó las columnas.

Aprovechamos para mostrarles cómo vemos esto por línea de comando en MySQL sin necesidad de usar un programa.

Nos conectamos a la base y luego nos posicionamos sobre la base de datos donde vamos a realizar las sentencias.

```

[mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| MiBaseDeDatos |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.04 sec)

[mysql> use MiBaseDeDatos;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>

```

Podemos ver las tablas que tenemos creadas con el comando *show tables*.

```

[mysql> show tables;
+-----+
| Tables_in_mibasededatos |
+-----+
| pais |
| usuario |
+-----+
2 rows in set (0.01 sec)

```

Y luego, para ver la definición de una tabla, utilizamos el comando *desc [tabla]* y completamos tabla con aquella en la cual requerimos inspeccionar la definición. Probemos con usuario.

```

[mysql> desc usuario;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| email | varchar(250) | NO | PRI | NULL | |
| nombre | varchar(45) | YES | | NULL | |
| apellido | varchar(45) | YES | | NULL | |
| fecha_nac | date | YES | | NULL | |
| pais_origen | int | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

```

Sabiendo que la definición nos indica que el orden es email, nombre, apellido, fecha_nac y pais_origen, podemos directamente incorporar un nuevo registro con ese orden. En caso que no queramos completar alguno de los campos (que se permite como NULL), incorporamos la palabra reservada NULL en su lugar.

Vamos a incorporar un usuario del cual no tenemos su primer nombre, pero sí su apellido y demás datos necesarios.

```
[mysql> INSERT INTO usuario VALUES ('garcia@mail.com', NULL, 'García', '1977-05-21', 1);  
Query OK, 1 row affected (0.02 sec)
```

Para comprobar si está bien ingresado, ingresamos el query SELECT * FROM usuario.

```
[mysql> SELECT * FROM usuario;  
+-----+-----+-----+-----+-----+  
| email                | nombre | apellido | fecha_nac | pais_origen |  
+-----+-----+-----+-----+-----+  
| garcia@mail.com      | NULL   | García   | 1977-05-21 | 1           |  
| mailprueba@mailprueba.com | Rubén  | Baez     | NULL       | 1           |  
| pedro.perez@mail.com  | Pedro  | Perez     | 1981-09-16 | 1           |  
+-----+-----+-----+-----+-----+  
3 rows in set (0.01 sec)
```

Ingresando varios registros al mismo tiempo

Ingresar de a un registro puede ser algo tedioso, por eso la sentencia INSERT nos permite realizar la incorporación de varios registros en una sola operación.

El cambio en la sentencia es bastante sencillo. Basta con agregar, luego del primer listado de valores, aquellos que necesitamos incorporando una coma entre cada conjunto de valores.

INSERT [tabla] (campo1, campo2, campo3, ..., campoN)

VALUES (valorA1, valorA2, valorA3, ..., valorAN), (valorB1, valorB2, valorB3, ..., valorBN),
(valorC1, valorC2, valorC3, ..., valorCN);

Hacemos una prueba y vemos cómo responde el motor:

```
19 INSERT INTO usuario (email, nombre, apellido, fecha_nac, pais_origen)
20 VALUES ('juan.garcia@mail.com', 'Juan', 'García', '1980-10-21', 1),
21 ('ester.rodriguez@mail.com', 'Ester', 'Rodríguez', '1990-12-27', 1),
22 ('laura.field@mail.com', 'Laura', 'Field', '1995-04-22', 1);
```

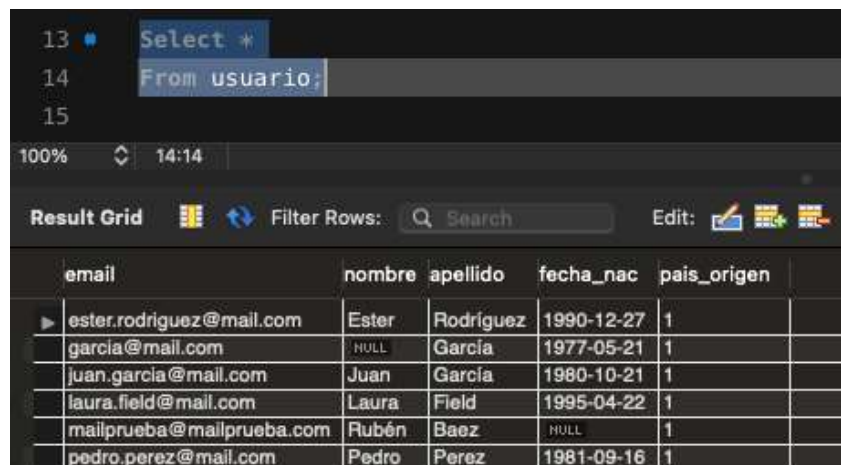
En el ejemplo propuesto se realiza la incorporación de 3 registros en única sentencia. En este caso, todos los datos se completaron, pero podría haberse combinado con algunos datos en NULL siempre que el campo estuviera definido así.

```
✓ 12 11:55:02 INSERT INTO usuario (email, nombre, apellido, fecha_... 3 row(s) affected Records: 3 Duplicates: 0 Warnings:...
```

Esta es la respuesta que nos regresa el motor, y como vemos, incorpora algo más de información donde nos permite saber la cantidad de registros, los duplicados y cuántos warnings se han detectado. A continuación, el extracto completo:

```
11:55:02      INSERT INTO usuario (email, nombre, apellido, fecha_nac, pais_origen)
VALUES      ('juan.garcia@mail.com',      'Juan',      'García',      '1980-10-21',      1),
('ester.rodriguez@mail.com', 'Ester', 'Rodríguez', '1990-12-27', 1), ('laura.field@mail.com',
'Laura', 'Field', '1995-04-22', 1)      3 row(s) affected Records: 3 Duplicates: 0 Warnings:
0      0.032 sec
```

Nos resta conocer como quedaron en la tabla.



The screenshot shows a database client interface. At the top, a SQL query is entered: `Select *` and `From usuario;`. Below the query, the 'Result Grid' is displayed, showing the results of the query. The grid has columns for email, nombre, apellido, fecha_nac, and pais_origen. The results are as follows:

email	nombre	apellido	fecha_nac	pais_origen
ester.rodriguez@mail.com	Ester	Rodríguez	1990-12-27	1
garcia@mail.com	NULL	García	1977-05-21	1
juan.garcia@mail.com	Juan	García	1980-10-21	1
laura.field@mail.com	Laura	Field	1995-04-22	1
mailprueba@mailprueba.com	Rubén	Baez	NULL	1
pedro.perez@mail.com	Pedro	Perez	1981-09-16	1

Existe otra forma de insertar registros de forma múltiple combinando Crear y Leer, pero es conveniente que primero veamos más en detalle acerca de leer los datos de la tabla, para poder realizar dicha combinación.