Insert, Update, Delete

DigitalHouse >
Coding School

Insertando datos en todas las columnas

Si estamos insertando datos en todas las columnas, no hace falta aclarar los nombres de cada columna. Sin embargo, el orden en el que insertemos los valores, deberá ser el mismo orden que tengan asignadas las columnas en la tabla.

SQL
```sql
INSERT INTO table_name (columna_1, columna_2, columna_3, ...)
VALUES (valor_1, valor_2, valor_3, ...);
```

SQL
```sql
INSERT INTO usuarios (id, nombre, apellido)
VALUES (DEFAULT, 'Max', 'Power');
```

SQL
```sql
INSERT INTO usuarios
VALUES (DEFAULT, 'Max', 'Power');
```

Insertando datos en columnas específicas

Para insertar datos en una columna en específico, aclaramos la tabla y luego escribimos el nombre de la o las columnas entre los paréntesis.

SQL

```sql
INSERT INTO usuarios (nombre)
VALUES ('Santi');
```

SQL

```sql
INSERT INTO peliculas (duracion, titulo)
VALUES (112, 'kill Bill');
```

Update

UPDATE modificará los registros existentes de una tabla. Es importante recordar utilizar siempre el **WHERE** en la sentencia para agregar la condición de cuáles son las filas que queremos actualizar.

SQL

```sql
UPDATE nombre_tabla
SET columna_1 = valor_1, columna_2 = valor_2, ...
WHERE condición;
```

SQL

```sql
UPDATE usuarios
SET nombre = 'Cosme', apellido = 'Fulanito'
WHERE id = 1;
```

Delete

Con **DELETE** podemos borrar información de una tabla. Al igual que con UPDATE, es importante no olvidar el **WHERE** cuando escribimos la sentencia, aclarando la condición. Si no escribimos el **WHERE**, estaríamos borrando **toda** la **tabla** y no un registro en particular.

SQL
```sql
DELETE FROM nombre_tabla WHERE condición;
```

SQL
```sql
DELETE FROM usuarios WHERE usuario_id = 4;
```

Select

DigitalHouse >
Coding School

Cómo usarlo

Toda consulta a la base de datos va a empezar con la palabra **SELECT**. Su funcionalidad es la de realizar consultas sobre **una** o **varias columnas** de una tabla.

Para especificar sobre qué tabla queremos realizar esa consulta usamos la palabra **FROM** seguida del nombre de la tabla.

```sql
SELECT nombre_columna, nombre_columna, ...
FROM nombre_tabla;
```

Where y Order by

DigitalHouse >
Coding School

Where

La funcionalidad del **WHERE** es la de condicionar y filtrar las consultas **SELECT** que se realizan a una base de datos.

```SQL
SELECT nombre_columna_1, nombre_columna_2, ...
FROM nombre_tabla
WHERE condicion;
```

Teniendo una tabla **usuarios**, podríamos consultar nombre y edad, filtrando con un **WHERE** solo los usuarios **mayores de 17 años** de la siguiente manera:

```SQL
SELECT nombre, edad
FROM usuarios
WHERE edad > 17;
```

Operadores

=	Igual a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	Diferente a
!=	Diferente a

Operadores

IS NULL	Es nulo
BETWEEN	Entre dos valores
IN	Lista de valores
LIKE	Se ajusta a...

Order By

ORDER BY se utiliza para ordenar los resultados de una consulta **según el valor de la columna especificada**. Por defecto, se ordena de forma ascendente (ASC) según los valores de la columna. También se puede ordenar de manera descendente (DESC) aclarándolo en la consulta.

SQL

```sql
SELECT nombre_columna1, nombre_columna2
FROM tabla
WHERE condicion
ORDER BY nombre_columna1;
```

Between y Like

DigitalHouse >
Coding School

Between

Cuando necesitamos obtener valores **dentro de un rango**, usamos el operador BETWEEN.

- BETWEEN **incluye** los **extremos**.
- BETWEEN funciona con **números**, **textos** y **fechas**.
- Se usa como un filtro de un WHERE.

Por ejemplo, coloquialmente:

- Dados los números: 4, 7, 2, 9, 1

Si hiciéramos un BETWEEN entre 2 y 7, devolvería 4, 7, 2 (excluye el 9 y el 1, e incluye el 2).

Like

Cuando hacemos un filtro con un **WHERE**, podemos especificar un patrón de búsqueda que nos permita especificar algo concreto que queremos encontrar en los registros. Eso lo logramos utilizando **comodines** *(wildcards)*.

Por ejemplo, podríamos querer buscar:

- ➔ Los nombres que tengan la letra “a” como segundo caracter.
- ➔ Las direcciones postales que incluyan la calle “Monroe”.
- ➔ Los clientes que empiecen con “Los” y terminen con “s”.

COMODÍN %

Es un sustituto que representa **cero**, **uno**, o **varios** caracteres.

COMODÍN _

Es un sustituto para **un sólo** carácter.

Alias

DigitalHouse >
Coding School

Alias

Los **ALIAS** se usan para darle un nombre temporal y más amigable a las **tablas**, **columnas** y **funciones**. Los **alias** se definen durante una consulta y persisten **solo** durante esa consulta.

Para definir un alias usamos las iniciales **AS** precediendo a la columna que estamos queriendo asignarle ese alias.

```sql
SELECT nombre_columna1 AS alias_nombre_columna1
FROM nombre_tabla;
```

Funciones MySQL

DigitalHouse >
Coding School

Concat

Usamos **CONCAT** para **concatenar** dos o más expresiones:

SQL
```sql
SELECT CONCAT('Hola ', 'a ', 'todos.');
```

```
> 'Hola a todos.'
```

SQL
```sql
SELECT CONCAT('La respuesta es: ', 24, '.');
```

```
> 'La respuesta es 24.'
```

SQL
```sql
SELECT CONCAT('Nombre: ', first_name, ' ', last_name)
FROM actors;
```

```
> 'Nombre: Emilia Clarke'
```

Coalesce

Usamos **COALESCE** para obtener la **primera expresión** que **no sea NULL**:

```sql
SELECT COALESCE(NULL, 1, 20, 'Digital House');
```

```
> 1
```

```sql
SELECT COALESCE(NULL, NULL, 'Digital House');
```

```
> 'Digital House'
```

Datediff

Usamos **DATEDIFF** para devolver la **diferencia** entre dos fechas, tomando como granularidad el intervalo especificado.

```sql
SELECT DATEDIFF(hour, '2017/08/25 07:00', '2017/08/25 12:45');
```

```
> 5
```

Devuelve 5 porque es la cantidad de horas de diferencia entre las 7 y las 12:45. Esta información da un resultado aproximado.

```sql
SELECT DATEDIFF(minute, '2017/08/25 07:00', '2017/08/25 12:45');
```

```
> 345
```

Devuelve 345 porque es la cantidad de minutos que van desde las 7 hasta las 12:45 (300min + 45min).

Extract

Usamos **EXTRACT** para **extraer** partes de una fecha:

```sql
SELECT EXTRACT(SECOND FROM '2014-02-13 08:44:21');
```

```
> 21
```

```sql
SELECT EXTRACT(MINUTE FROM '2014-02-13 08:44:21');
```

```
> 44
```

```sql
SELECT EXTRACT(HOUR FROM '2014-02-13 08:44:21');
```

```
> 8
```

```sql
SELECT EXTRACT(DAY FROM '2014-02-13 08:44:21');
```

```
> 13
```

Extract

```sql
SELECT EXTRACT(WEEK FROM '2014-02-13 08:44:21');
```

```
> 6
```

```sql
SELECT EXTRACT(MONTH FROM '2014-02-13 08:44:21');
```

```
> 2
```

```sql
SELECT EXTRACT(QUARTER FROM '2014-02-13 08:44:21');
```

```
> 1
```

```sql
SELECT EXTRACT(YEAR FROM '2014-02-13 08:44:21');
```

```
> 2014
```

Replace

Usamos **REPLACE** para reemplazar una secuencia de caracteres por otra en un string.

```sql
SELECT REPLACE('abc abc', 'a', 'B');
```

```
> Bbc Bbc
```

```sql
SELECT REPLACE('abc abc', 'A', 'B');
```

```
> abc abc
-- no se encuentran coincidencias para reemplazar
```

```sql
SELECT REPLACE('123 123', '2', '5');
```

```
> 153 153
```

Date format

Usamos **DATE_FORMAT** para que dada una fecha determinada se pueda formatear la misma según deseemos.

```sql
SELECT DATE_FORMAT('2017-06-15', '%Y');
> '2017'
```

```sql
SELECT DATE_FORMAT('2017-06-15', '%W %M %e %Y');
> 'Thursday June 15 2017'
```

Descubre todas las posibilidades de formato que posee esta función.

Case

Usamos **CASE** para **evaluar condiciones** y devolver la primera que se cumpla. En este ejemplo, la tabla resultante tendrá 4 columnas: *id*, *title*, *rating* y *rating_categories*. Esta última mostrará 'Mala', 'Regular', etc., según el **rating** de la película.

SQL

```sql
SELECT id, title, rating
  CASE
    WHEN rating < 4 THEN 'Mala'
    WHEN rating < 6 THEN 'Regular'
    WHEN rating < 8 THEN 'Buena'
    WHEN rating < 9.5 THEN 'Muy buena'
    ELSE 'Excelente'
  END AS rating_categories
FROM movies
ORDER BY rating
```


DigitalHouse>
Coding School