

Bloque C

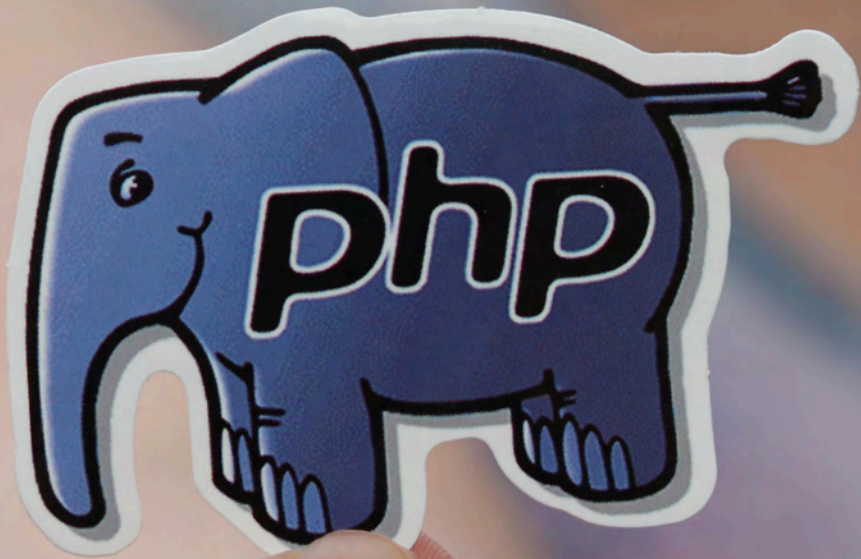
Sitios web basados en bases de datos

C11. Structured Query Language (SQL)



Contenidos

1. Introducción
2. Obteniendo datos de una base de datos (READ)
3. Añadiendo datos (CREATE)
4. Actualizando datos (UPDATE)
5. Borrando datos (DELETE)
6. Restricciones
7. Resumen



1. Introducción

Introducción

El lenguaje de consulta estructurado (**SQL**) es un lenguaje que se utiliza para **comunicarse con las bases de datos**. Se utiliza para **solicitar datos, añadir datos nuevos, editar datos existentes y eliminar datos**.

Introducción

En este capítulo, aprenderás a utilizar SQL para realizar las siguientes tareas:

- **Seleccionar datos** de una base de datos
- **Crear nuevas filas** en una tabla de base de datos
- **Actualizar datos ya almacenados** en una base de datos
- **Eliminar filas** de las tablas de una base de datos

Introducción

Una instrucción para obtener o cambiar datos almacenados en la base de datos se conoce como **sentencia SQL (SQL statement)**. Una sentencia SQL que sólo pide información también puede denominarse **consulta SQL (SQL query)**, ya que está pidiendo datos a la base de datos. Aprenderás a escribir consultas SQL antes de aprender a crear, actualizar o borrar datos de una base de datos.

Introducción

Para aprender el lenguaje SQL, utilizarás *phpMyAdmin*. Una vez que hayas aprendido cómo funciona SQL en esta unidad, las siguientes dos unidades te mostrarán cómo una página PHP utiliza *PDO* (PHP Data Objects) para **ejecutar sentencias SQL que obtienen o actualizan los datos de la base de datos.**

Introducción

Algunos de los ejemplos de esta unidad actualizan los datos que se almacenan en la base de datos que configuramos en la introducción a este bloque, y que forman la base del ejemplo de sitio web que vamos a desarrollar a lo largo del módulo, por lo que **deben ejecutarse una vez, en el orden en que aparecen en las diapositivas.**

Si no se ejecutan en ese orden, es posible que los ejemplos posteriores no funcionen. Si esto ocurre, o si quieres ejecutarlos de nuevo, borra la base de datos y configúrala de nuevo utilizando las instrucciones de la introducción a esta sección.



2. Obteniendo datos de una base de datos (READ)

Obteniendo datos de una base de datos

Para solicitar datos a la base de datos, utiliza el comando `SELECT` de SQL y especifica los datos que deseas obtener. La base de datos creará un **conjunto de resultados (result set)** con los datos solicitados.

Obteniendo datos de una base de datos

El comando `SELECT` indica que se desea obtener datos de la base de datos. Va seguido de los **nombres de las columnas que contienen los datos deseados**. Cada nombre de columna debe ir separado por una coma.

La cláusula `FROM` va seguida del **nombre de la tabla de la que desea obtener los datos**. Una sentencia SQL debe terminar con punto y coma (aunque en determinadas circunstancias se puede omitir).



The diagram illustrates the structure of an SQL query. On the left, two labels are shown: 'COLUMNS TO SELECT' and 'TABLE COLUMNS ARE IN'. Arrows point from these labels to the corresponding parts of the SQL query: 'SELECT column1, column2' and 'FROM table;'. The SQL keywords 'SELECT' and 'FROM' are highlighted in orange, while the column and table names are in a dark red font.

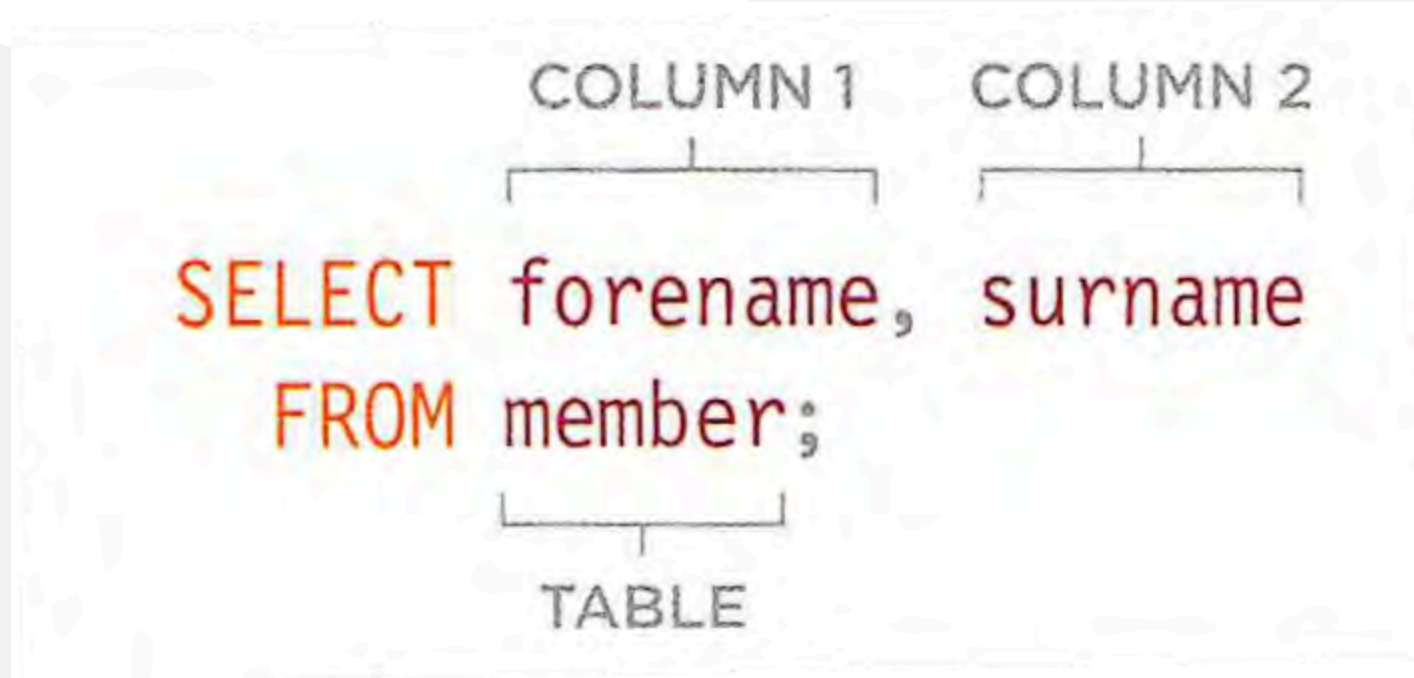
```
SELECT column1, column2  
FROM table;
```

Obteniendo datos de una base de datos

La sentencia SQL que se muestra a continuación **solicita los datos de las columnas *forename* y *surname* de la tabla de *members***. Devuelve los datos de cada fila de la tabla.

Se puede leer literalmente; dice lo siguiente

- SELECCIONAR las columnas nombre y apellidos
- DE la tabla de miembros



Obteniendo datos de una base de datos

Los comandos SQL pueden escribirse en mayúsculas o minúsculas. En los ejemplos que veréis en estas unidades, están en mayúsculas para distinguirlos de los nombres de tablas y columnas. **Los nombres de tablas y columnas deben utilizar las mismas mayúsculas y minúsculas que la base de datos.**

Cuando se ejecuta una consulta SQL, la base de datos obtiene los datos solicitados y los coloca en un conjunto de resultados. Las columnas se añaden al conjunto de resultados **en el mismo orden en que se nombran en la consulta**. Para controlar el orden en que se añaden las filas al conjunto de resultados, utiliza la cláusula `ORDER BY` (la veremos más adelante).

Obteniendo datos de una base de datos

result set	
forename	surname
Ivy	Stone
Luke	Wood
Emiko	Ito

Obteniendo datos de una base de datos

A continuación se muestra cómo introducir la consulta SQL en *phpMyAdmin* y cómo se muestra el conjunto de resultados que genera la consulta.

1. Abre phpMyAdmin y selecciona la base de datos `phpbook-1` . Si no la ha creado, consulte la introducción al bloque C.
2. Selecciona la pestaña *SQL*.
3. Escribe la consulta SQL que hemos visto previamente en el área de texto.
4. Haz clic en *Ir*.

Obteniendo datos de una base de datos

The screenshot displays the phpMyAdmin web interface. On the left, the database structure tree shows 'phpbook-1' selected, with its tables 'article', 'category', 'image', and 'member' listed. The main panel is in the 'SQL' tab, showing a query: `1 SELECT forename, surname` and `2 FROM member;`. The interface includes buttons for 'Clear', 'Format', and 'Get auto-saved query'. At the bottom, there are checkboxes for 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks'. A 'Go' button is at the bottom right. Red numbered circles highlight key steps: 1 points to the selected database, 2 points to the SQL tab, 3 points to the query text, and 4 points to the 'Go' button.

phpMyAdmin

Recent Favorites

- New
- information_schema
- mysql
- performance_schema
- phpbook-1
- sys

Server: localhost:8889 » Database: phpbook-1

Structure SQL Search Query Export More

Run SQL query/queries on database phpbook-1:

```
1 SELECT forename, surname
2 FROM member;
```

Clear Format Get auto-saved query

Bind parameters

[Delimiter ;] ☒ Show this query here again ☐ Retain query box

Rollback when finished ☒ Enable foreign key checks

Go

Obteniendo datos de una base de datos

5. Cuando se pulsa *Ir*, se ejecuta la consulta SQL. A continuación, MySQL devuelve el conjunto de resultados a *phpMyAdmin*, y *phpMyAdmin* muestra el conjunto de resultados en una tabla.

Obteniendo datos de una base de datos

The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of databases: 'New', 'information_schema', 'mysql', 'performance_schema', 'phpbook-1', and 'sys'. The 'phpbook-1' database is selected, showing its tables: 'New', 'article', 'category', 'image', 'member', and 'sys'. The 'member' table is highlighted.

The main panel shows the 'Table: member' view. At the top, a navigation bar includes 'Browse', 'Structure', 'SQL', 'Search', 'Insert', and 'More'. Below this, a 'Show query box' section displays a successful query: `SELECT forename, surname FROM member`. A yellow status bar indicates 'Showing rows 0 - 2 (3 total, Query took 0.0001 seconds.)'. Below the query, there are links for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'.

Below the query box, there are controls for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box labeled 'Search this table'.

The '+ Options' section shows a table of data with columns 'forename' and 'surname'. The table is highlighted with a red box. To the right of the table is a red circle with the number '5'. Each row has 'Edit', 'Copy', and 'Delete' icons.

	forename	surname
	Ivy	Stone
	Luke	Wood
	Emiko	Ito

At the bottom, there are links for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'.

Obteniendo datos de una base de datos

Para el resto de esta unidad, en lugar de mostrar capturas de pantalla de *phpMyAdmin*, se mostrarán las consultas SQL y sus conjuntos de resultados como hemos visto previamente.

Devolver filas específicas de una tabla

Para obtener datos de filas específicas de una tabla (en lugar de todas ellas), añade la cláusula `WHERE` seguida de una **condición de búsqueda**.

Devolver filas específicas de una tabla

Una vez que hayas especificado qué columnas de datos obtener de una tabla, puedes añadir una condición de búsqueda para controlar qué filas de esa tabla deben añadirse al conjunto de resultados. En la condición de búsqueda, **nombra una columna de la tabla, y si su valor debe ser igual a `=` , mayor que `>` , o menor que `<` un valor que especifiques.**

A medida que la base de datos recorre las filas de la tabla, **si la condición resulta verdadera, añade una nueva fila al conjunto de resultados** y copia los datos de las columnas nombradas tras el comando `SELECT` en el conjunto de resultados.

Devolver filas específicas de una tabla

COLUMNS TO SELECT	→	SELECT <i>column(s)</i>
TABLE COLUMNS ARE IN	→	FROM <i>table</i>
ROWS TO ADD TO RESULT SET	→	WHERE <i>column = value;</i>
		OPERATOR

Devolver filas específicas de una tabla

Si el valor que se especifica en la condición es **texto**, escribe el texto **entre comillas simples**.

Si el valor que se especifica en la condición es un **número**, **no se debe poner el número entre comillas**.

MySQL **no tiene un tipo de dato booleano**, pero se puede utilizar un tipo de dato `tinyint` para **representar un booleano con un valor de 1 para verdadero y 0 para falso**. Debido a que estos valores son números, no deben colocarse entre comillas.

Devolver filas específicas de una tabla

Puedes combinar varias condiciones de búsqueda utilizando los tres **operadores lógicos** que se muestran en la tabla siguiente.

OPERATOR DESCRIPTION	
AND	All conditions must return true.
OR	Any one condition can return true.
NOT	Reverses a condition; checks it is not true.

Funcionan como los operadores lógicos de PHP mostrados en la unidad 1.

Cada condición de búsqueda se coloca entre paréntesis para garantizar que se ejecuta por
sí sola.

Devolver filas específicas de una tabla

COLUMNS TO SELECT → **SELECT** *column(s)*
TABLE COLUMNS ARE IN → **FROM** *table*
ROWS TO ADD TO RESULT SET → **WHERE** *(column < value) AND (column > value);*

CONDITION 1 LOGICAL OPERATOR CONDITION 2

Ejemplo: usando operadores de comparación en SQL

Este ejemplo selecciona la dirección de correo electrónico de todos los miembros que tienen el valor *Ivy* en la columna *forename* de la tabla de *members*.

```
SELECT email  
  FROM member  
 WHERE forename = 'Ivy';
```

result set
email
ivy@eg.link

Ejemplo: usando operadores de comparación en SQL

Este otro ejemplo busca las direcciones de correo electrónico de los miembros que tienen un valor inferior a 3 en la columna *id* de la tabla de miembros.

```
SELECT email  
FROM member  
WHERE id < 3;
```

result set
email
ivy@eg.link
luke@eg.link

Ejemplo: usando operadores de comparación en SQL

Este ejemplo utiliza el operador AND para encontrar los resultados en los que la dirección de correo electrónico de un miembro cumple dos condiciones. Empieza por:

1. Una letra mayor que *E*
2. Una letra menor que *L*

```
SELECT email  
  FROM member  
 WHERE (email > 'E') AND (email < 'L');
```

result set
email
emi@eg.link
ivy@eg.link

Actividad C11.1. Usando operadores de comparación en SQL

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Encuentra la dirección de correo electrónico de cualquier miembro cuyo nombre sea `Luke` .
- Encontrar las direcciones de correo electrónico de los miembros cuyo id es menor o igual a `3` .
- Busca las direcciones de correo electrónico de los miembros cuya dirección de correo electrónico empiece por las letras `G-L` .

Búsqueda con Like y comodines

El operador `LIKE` puede utilizarse en una condición de búsqueda para encontrar filas de datos en las que **el valor de una columna específica empiece por, termine en o contenga caracteres específicos.**

Búsqueda con Like y comodines

El operador `LIKE` encuentra filas de datos en las que una columna contiene caracteres que coinciden con un **patrón** especificado. Por ejemplo, el patrón puede utilizarse para buscar filas en las que el valor de una columna especificada:

- Empieza por una letra determinada
- Termina con un número determinado
- Contiene una palabra o un conjunto de caracteres específicos (se suele utilizar para crear funcionalidades de búsqueda en los sitios web).

Un patrón puede utilizar **símbolos comodín** para especificar dónde podrían estar otros caracteres (como se muestra en la tabla siguiente):

- `%` indica cero o más caracteres
- `_` indica un carácter individual

Búsqueda con Like y comodines

VALUE	MATCHES COLUMNS WHOSE VALUE
To%	Starts with To
%day	Ends with day
%to%	Contains to at any point
h_11	Has a single character in place of the underscore (e.g., hall, hell, hill, hull)
%h_11%	Contains h, then any character, then two 11s (e.g., hall, hell, chill, hilly, shellac, chilled, hallmark, hullabaloo)
1%	Starts with 1
%!	Ends with !

El valor no distingue entre mayúsculas y minúsculas, lo que significa que si se busca el nombre /vy también se encontrarán los valores /VY e ivy.

Búsqueda con Like y comodines

COLUMNS TO SELECT	→	SELECT	column(s)	
TABLE COLUMNS ARE IN		FROM	table	
ROWS TO ADD TO RESULT SET	→	WHERE	column	LIKE '%value%';
			└──┬──┘	
			LIKE OPERATOR	WILDCARD SYMBOLS

Ejemplo: buscando valores

Este ejemplo busca las direcciones de correo electrónico de todos los socios cuyo nombre empiece por la letra I (mayúscula o minúscula).

```
SELECT email  
FROM member  
WHERE forename LIKE 'I%';
```

result set
email
ivy@eg.link

Ejemplo: buscando valores

Este ejemplo obtiene los correos electrónicos de todos los miembros cuyo nombre de pila:

- Empieza por la letra **E**
- Seguido de otro carácter
- Después la letra **I**
- A continuación, cualquier otro carácter o secuencia de caracteres (devolvería nombres como Eli, Elias, Elijah, Elisha, Emi, Emiko, Emil, Emilio, Emily, Eoin y Eric).

```
SELECT email  
FROM member  
WHERE forename LIKE 'E_I%';
```

result set
email
emi@eg.link

Ejemplo: buscando valores

Este ejemplo busca la dirección de correo electrónico de cualquier persona cuyo nombre sea Luke. Como no hay caracteres comodín, sólo devuelve coincidencias exactas.

```
SELECT email  
  FROM member  
 WHERE forename LIKE 'Luke';
```

result set
email
luke@eg.link

Actividad C11.2. Buscando valores

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Encuentra miembros cuyo nombre empiece por la letra `E`.
- Buscar socios cuyo nombre coincida con `L_K%`.
- Buscar miembros que se llamen `Ivy`.

Controlando el orden de las filas de un result set

Para **controlar el orden en que se añaden las filas a un conjunto de resultados**, se utiliza la cláusula `ORDER BY`, seguida del nombre de una columna cuyos valores ordenarán los resultados y, a continuación, `ASC` para ascendente o `DESC` para descendente.

Controlando el orden de las filas de un result set

`ORDER BY` puede añadirse al final de una consulta para controlar el orden en que las filas se añaden al conjunto de resultados. Los valores de la columna especificada se utilizan para **controlar el orden de los resultados**.

Esto debe ir seguido de una de dos palabras clave: `ASC` para **ascendente** o `DESC` para **descendente**. (Si no se especifica ninguna, ordena en orden **ascendente (por defecto)**, pero tu SQL será más fácil de leer si especificas `ASC` o `DESC`).

Controlando el orden de las filas de un result set

COLUMNS TO SELECT → **SELECT** *column(s)*
TABLE COLUMNS ARE IN → **FROM** *table*
CONTROL ORDER OF RESULTS → **ORDER BY** *column* **ASC**;
 └──┬──┘ └──┬──┘ └──┬──┘
 ORDER BY CLAUSE COLUMN DIRECTION

Controlando el orden de las filas de un result set

Es posible organizar el orden en que las filas se añaden al conjunto de resultados **utilizando los valores de varias columnas**. Cada nombre de columna estará separado por una coma. **Si la primera columna utilizada para ordenar los valores contiene valores idénticos, se remitirá a la segunda columna de la lista.**

Por ejemplo, si se ordena a los miembros por su nombre, y más de un miembro comparte el mismo nombre (almacenado en la columna *forename*), se podría ordenar por su apellido (almacenado en la columna *surname*).

Controlando el orden de las filas de un result set

COLUMNS TO SELECT → **SELECT** *column(s)*

TABLE COLUMNS ARE IN → **FROM** *table*

CONTROL ORDER OF RESULTS → **ORDER BY** *column1* **ASC**, *column2* **DESC**;

ORDER BY CLAUSE COLUMN 1 DIRECTION COLUMN 2 DIRECTION

Ejemplo: ordenando resultados

Este ejemplo obtiene todas las direcciones de correo electrónico y las ordena en orden descendente.

```
SELECT email  
FROM member  
ORDER BY email DESC;
```

result set
email
luke@eg.link
ivy@eg.link
emi@eg.link

Ejemplo: ordenando resultados

Este ejemplo obtiene valores de las columnas title y category_id de la tabla article y los ordena por category_id primero en orden ascendente, y luego por el título en orden alfabético.

El conjunto de resultados contiene más filas de las que se muestran en la imagen (una por cada artículo), pero no hay espacio suficiente para mostrarlas todas aquí.

```
SELECT title, category_id  
FROM article  
ORDER BY category_id ASC, title ASC;
```

result set (showing first 10 rows of 24)	
title	category_id
Chimney Business Cards	1
Milk Beach Album Cover	1
Polite Society Posters	1
Systemic Brochure	1
The Ice Palace	1
Travel Guide	1
Chimney Press Website	2
Floral Website	2
Milk Beach Website	2
Polite Society Website	2

Actividad C11.3. Ordenando resultados

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

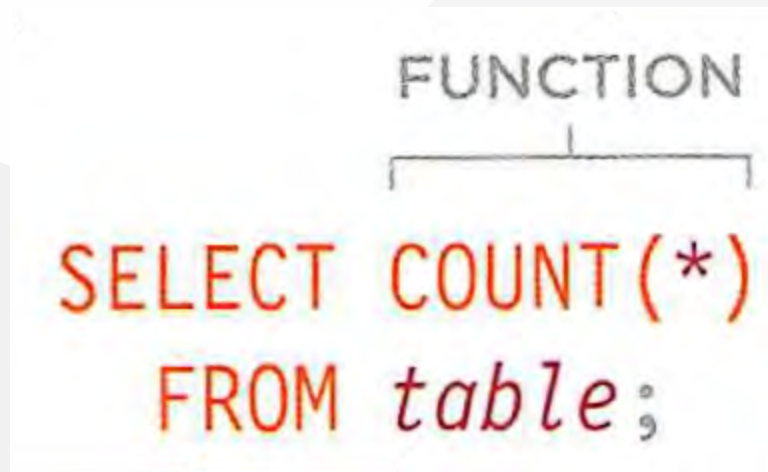
- Invierte el orden de los resultados cambiando DESC por ASC.
- Selecciona las columnas `title` y `member_id` de la tabla de artículos y ordena los resultados primero por el id de miembro y luego por el título en orden alfabético.

Recuento y agrupación de resultados

Cuando se utiliza la función `COUNT()` de SQL después del comando `SELECT`, se añade al conjunto de resultados el **número total de filas que coinciden con la consulta**. La **agrupación de resultados** permite contar cuántas filas contienen el mismo valor.

Recuento y agrupación de resultados

Para contar el número de filas de una tabla, se llama a `COUNT()` después del comando `SELECT` y se especifica la tabla. Utiliza un asterisco (conocido como **comodín**) como argumento de la función `COUNT()`.



FUNCTION

```
SELECT COUNT(*)  
FROM table;
```


Recuento y agrupación de resultados

Si añades una condición de búsqueda a la consulta, la función COUNT() devuelve el número de filas que coinciden con la consulta con la condición de búsqueda.

```
SELECT COUNT(*)  
FROM table  
WHERE column LIKE '%value%';
```

Recuento y agrupación de resultados

Si especificas un nombre de columna como argumento para la función `COUNT()`, ésta cuenta el número de filas en las que el valor de la columna especificada no es *NULL*.

```
SELECT COUNT(column)  
FROM table;
```

Recuento y agrupación de resultados

La cláusula `GROUP BY` puede utilizarse con la función `COUNT()` de SQL para determinar cuántas filas tienen el mismo valor en una columna. Por ejemplo, se puede contar cuántos artículos ha escrito un miembro, o cuántos artículos hay en una categoría.

Para ello, en la sentencia `SELECT` :

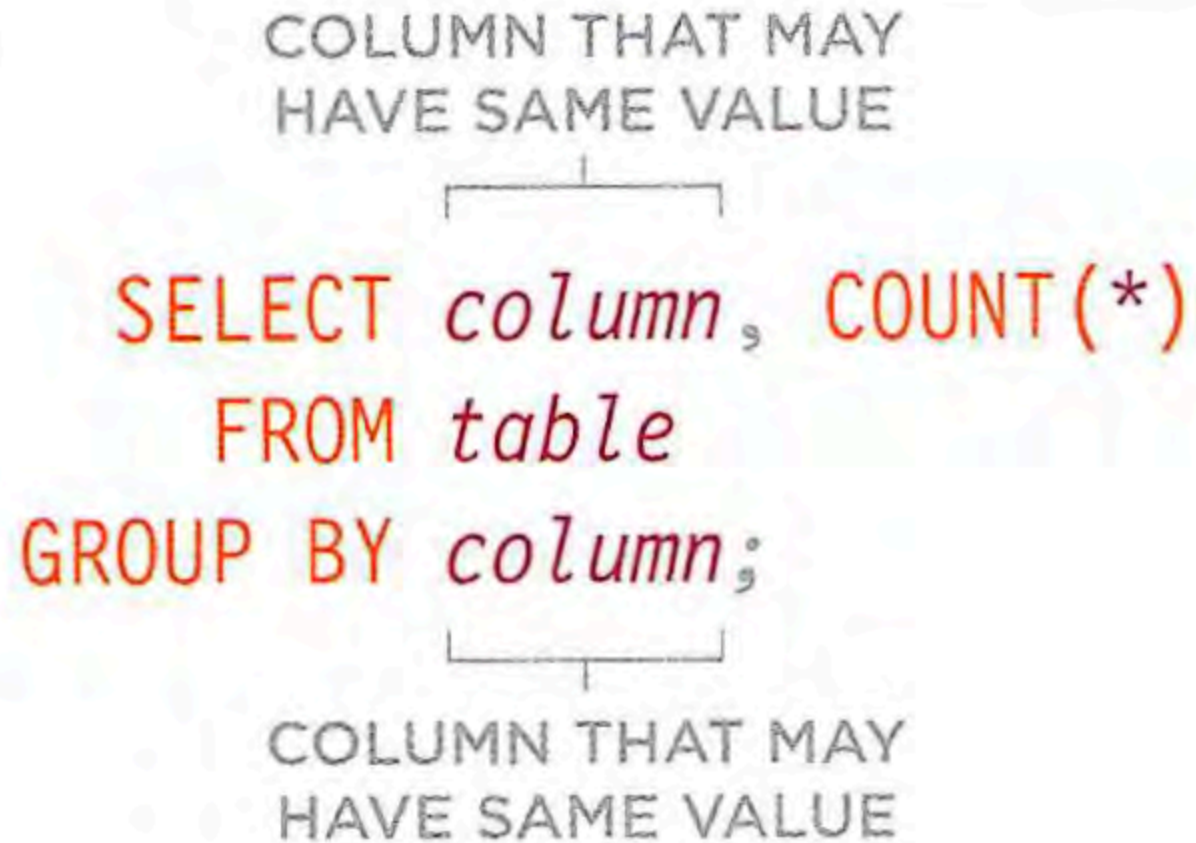
1. Selecciona un nombre de columna que pueda contener valores idénticos (por ejemplo, `member_id` o `category_id`).
2. Utiliza `COUNT(*)` para contar el número de filas.
3. Especifica la tabla en la que se encuentra la columna.
4. Utiliza la cláusula `GROUP BY` , seguida del nombre de la columna que puede contener valores idénticos, para que **agrupe el número de filas que comparten el mismo valor en esta columna y las cuente.**

Recuento y agrupación de resultados

COLUMN THAT MAY
HAVE SAME VALUE

SELECT *column*, COUNT(*)
FROM *table*
GROUP BY *column*;

COLUMN THAT MAY
HAVE SAME VALUE



Ejemplo: recuento del número de resultados coincidentes

Este ejemplo utiliza la función `COUNT()` de SQL para obtener el número de miembros que han proporcionado una imagen de perfil. Si el valor de la columna de imagen es NULL, no se contarán.

```
SELECT COUNT(picture)  
FROM member;
```

result set
COUNT(picture)
2

Ejemplo: recuento del número de resultados coincidentes

Este ejemplo utiliza la función `COUNT()` de SQL para obtener el número de artículos en los que las columnas de *title* o *content* contienen el término *design*.

```
SELECT COUNT(*)  
  FROM article  
 WHERE title LIKE '%design%' OR content LIKE '%design%';
```

result set
COUNT(*)
9

Ejemplo: recuento del número de resultados coincidentes

La siguiente consulta cuenta el número de artículos de cada miembro. La sentencia `SELECT` obtiene la columna de identificación del miembro y la función `COUNT()` cuenta el número de filas coincidentes. La cláusula `FROM` indica que se busca en la tabla `article`. La cláusula `GROUP BY` agrupa los valores de la columna `member_id` para que puedas ver el id del miembro y el número de artículos que escribió.

Ejemplo: recuento del número de resultados coincidentes

```
SELECT member_id, COUNT(*)  
FROM article  
GROUP BY member_id;
```

result set

member_id	COUNT(*)
1	10
2	8
3	6

Actividad C11.4. Recuento del número de resultados coincidentes

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Contar el número de miembros con una dirección de correo electrónico (que ese atributo no sea nulo).
- Encontrar el número de artículos que contienen el término `photo`.
- Calcular el número de artículos de cada categoría.

Limitar y omitir resultados

LIMIT restringe el número de resultados que se añaden a un conjunto de resultados.

OFFSET indica a la base de datos que omita un número determinado de registros y añada los siguientes al conjunto de resultados.

Limitar y omitir resultados

Para restringir el número total de filas que se añaden al conjunto de resultados, utiliza la cláusula `LIMIT`.

De este modo, sólo se añadirían al conjunto de resultados los n primeros resultados que coincidan con la consulta, siendo n el límite indicado en la cláusula `LIMIT`.



Limitar y omitir resultados

La cláusula `OFFSET` puede utilizarse después de la cláusula `LIMIT` para **omitir las primeras coincidencias** que, de otro modo, se habrían añadido al conjunto de resultados.

El siguiente ejemplo omitiría los seis primeros resultados que coinciden con la consulta y añadiría los tres siguientes al conjunto de resultados.

COLUMNS TO SELECT
TABLE COLUMNS ARE IN
LIMIT AND SKIP ROWS

```
SELECT column(s)
FROM table
LIMIT 3 OFFSET 6;
```

OFFSET CLAUSE RESULTS TO SKIP

Limitar y omitir resultados

`LIMIT` y `OFFSET` se utilizan a menudo cuando una consulta genera muchos resultados. Los resultados se dividen en páginas separadas utilizando una técnica llamada **paginación**. Los resultados de Google son un ejemplo muy conocido.

Después de la primera página de resultados, hay enlaces a páginas adicionales que muestran más resultados que coinciden con la misma consulta. En la próxima unidad aprenderás a utilizar estos comandos para añadir paginación.

Ejemplo: restringir el número de resultados coincidentes

Este ejemplo pide los títulos de los artículos, ordenados por el valor de la columna id. Utiliza la cláusula `LIMIT` para añadir sólo la primera coincidencia al conjunto de resultados.

```
SELECT title  
  FROM article  
 ORDER BY id  
 LIMIT 1;
```

result set
title
Systemic Brochure

Ejemplo: restringir el número de resultados coincidentes

Este ejemplo solicita los títulos de los artículos, ordenados por el valor de la columna `id`. Utiliza la cláusula `OFFSET` para omitir los nueve primeros resultados que coinciden con la consulta y, a continuación, utiliza la cláusula `LIMIT` para añadir las tres coincidencias siguientes al conjunto de resultados.

```
SELECT title  
  FROM article  
 ORDER BY id  
 LIMIT 3 OFFSET 9;
```

result set
title
Polite Society Mural
Stargazer Website and App
The Ice Palace

Actividad C11.5. Restringir el número de resultados coincidentes

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Consigue los cinco primeros artículos de la categoría `print`.
- Omitir las seis primeras coincidencias y devolver las seis siguientes.

Utilizar uniones para obtener datos de dos tablas

Una **unión (join)** permite **solicitar datos de más de una tabla**. Los datos de ambas tablas se añaden a una única fila del conjunto de resultados.

Utilizar uniones para obtener datos de dos tablas

Al diseñar una base de datos, conviene crear **una tabla para cada concepto que represente el sitio y evitar la duplicación de datos** en más de una tabla.

En el sitio web de ejemplo, los datos sobre *artículos*, *categorías*, *miembros* e *imágenes* viven en tablas diferentes. La primera columna de estas tablas contiene un valor que se utiliza para identificar cada fila de la tabla. Por ejemplo, el valor de la columna *id* de la tabla de categorías puede identificar cada categoría. Este valor se denomina **clave primaria**.

La tabla de artículos necesita almacenar en qué categoría se encuentra cada artículo. En lugar de duplicar los nombres de las categorías para cada artículo, tiene una columna llamada `category_id`. El valor de esta columna se conoce como **clave foránea** y se corresponderá con la clave primaria de la categoría en la que se encuentra.

Utilizar uniones para obtener datos de dos tablas

Las claves primaria y foránea describen cómo los datos de una fila de una tabla pueden **relacionarse** con los datos de una fila de otra tabla. A continuación se muestra la relación entre el segundo artículo y la categoría en la que se encuentra.

Cuando se escribe una consulta SQL para recopilar información sobre un artículo y se desea incluir información de otra tabla (como el nombre de la categoría en la que se encuentra), el artículo es el objeto primario de la consulta. Por lo tanto, la tabla de artículos se conoce como **tabla izquierda**.

Cuando se obtienen datos adicionales sobre el artículo de una segunda tabla (como la tabla de categorías), la segunda tabla se conoce como **tabla derecha**.

Una cláusula `JOIN` describe cómo se relacionan los valores.

Utilizar uniones para obtener datos de dos tablas

article								
id	title	summary	content	created	category_id	member_id	image_id	published
1	Systemic Bro...	Brochure...	This bro...	2021-01-26	1	2	1	1
2	Forecast	Handbag...	This dra...	2021-01-29	3	2	2	1
3	Swimming Pool	Architec...	This pho...	2021-02-02	4	1	3	1

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1

Utilizar uniones para obtener datos de dos tablas

Hasta ahora, las consultas han recogido datos de una tabla de la base de datos cada vez. Cuando se utiliza un `JOIN` para obtener datos de más de una tabla, **se especifica el nombre de la tabla en la que se encuentra una columna, así como el nombre de la columna**. Para ello se utiliza:

- El nombre de la tabla en la que están los datos
- Seguido de un punto (`.`)
- Seguido del nombre de la columna

Utilizar uniones para obtener datos de dos tablas

La siguiente consulta selecciona todos los títulos y resúmenes de artículos de la tabla de artículos, y también obtiene el nombre de la categoría a la que pertenece cada artículo de la tabla de categorías.

1. El comando `SELECT` va seguido de los nombres de las columnas de las que deben devolverse los datos.
2. El comando `FROM` va seguido del nombre de la tabla de la izquierda (el tema principal de la consulta). En este caso, se trata de la tabla *article*.
3. La cláusula `JOIN` va seguida del nombre de la tabla derecha (la que contiene la información adicional). En este caso, es la tabla *category*.

Utilizar uniones para obtener datos de dos tablas

A continuación, la unión indica a la base de datos el nombre de una columna de la tabla izquierda y de la tabla derecha cuyos valores coincidirán.

Para ello, utiliza la palabra clave `ON` , seguida de:

- Columna de la tabla izquierda que contiene su clave externa
- Símbolo igual
- Columna de la tabla derecha que contiene su clave primaria

Utilizar uniones para obtener datos de dos tablas

```
SELECT article.title, article.summary, category.name  
FROM article  
JOIN category ON article.category_id = category.id;
```

FOREIGN KEY PRIMARY KEY

Utilizar uniones para obtener datos de dos tablas

El siguiente conjunto de resultados muestra las tres primeras filas de datos que se añadirían al conjunto de resultados (el conjunto de resultados completo contendría todos los artículos).

NOTA: Los nombres de las columnas del conjunto de resultados no utilizan nombres de tablas porque los datos se han tomado de esas tablas y se han combinado en un único conjunto de resultados.

result set (showing first 3 rows of 24)		
title	summary	name
Milk Beach Website	Website for music series	Digital
Wellness App	App for health facility	Digital
Stargazer Website and App	Website and app for music festival	Digital

Utilizar uniones para obtener datos de dos tablas

Para seleccionar una fila individual, o un subconjunto de filas, se puede añadir una condición de búsqueda después del JOIN. Por ejemplo, la siguiente consulta sólo devolvería los detalles de los artículos que se encuentran en la categoría *print*.

La condición de búsqueda también puede ir seguida de cláusulas que ordenen, limiten y omitan resultados cuando se añadan al conjunto de resultados (como se muestra en ejemplos anteriores de esta unidad).

```
SELECT article.title, article.summary, category.name  
FROM article  
JOIN category ON article.category_id = category.id  
WHERE category.id = 1;
```

Cómo funcionan las uniones si faltan datos

Cuando la base de datos intenta realizar un `JOIN` en una fila de datos, pero faltan algunos de los datos, se puede especificar si **añadir los datos disponibles al conjunto de resultados** u **omitir esa fila y no añadirla** al conjunto de resultados.

Cómo funcionan las uniones si faltan datos

Imagina que quieres obtener los datos de cada imagen que se ha subido a un artículo. Se podría utilizar un `JOIN` para obtener los datos de la imagen (como el `JOIN` que obtuvo el título del artículo y el nombre de la categoría en la que se encuentra en la página anterior).

La columna `image_id` de la tabla `article` es una clave externa porque su valor es la clave primaria de la tabla `image` que contiene la imagen del artículo.

Sin embargo, hay una diferencia clave. Mientras que cada artículo debe pertenecer a una categoría (la base de datos lo impone utilizando algo llamado restricción que conocerás más adelante), **un artículo no necesita tener obligatoriamente una imagen.**

Cómo funcionan las uniones si faltan datos

Si no se carga una imagen para un artículo, la columna image de la tabla de artículos tiene el valor `NULL`.

En la siguiente tabla de artículos, la columna `image_id` de uno de los artículos tiene el valor `NULL` porque **no hay ninguna imagen para el artículo**. Esto significa que el `JOIN` no encontraría ningún dato de imagen correspondiente en la tabla de imágenes.

A continuación, se muestran dos tipos de `JOIN` que pueden utilizarse para especificar **si la consulta debe añadir el resto de los datos que puede encontrar al conjunto de resultados, o si debe omitir toda la fila de datos porque no se ha podido encontrar la imagen correspondiente**.

Cómo funcionan las uniones si faltan datos

article								
id	title	summary	content	created	category_id	member_id	image_id	published
4	Walking Birds	Artwork ...	The brie...	2021-02-12	3	3	4	1
5	Sisters	Editoria...	The arti...	2021-02-27	3	3	NULL	1
6	Micro-Dunes	Photogra...	This pho...	2021-03-03	4	1	6	1

image		
id	file	alt
4	birds.jpg	Collage of two birds
6	micro-dunes.jpg	Photograph of tiny sand dunes

Cómo funcionan las uniones si faltan datos

INNER JOIN

Una **unión interna (inner join)** añade datos al conjunto de resultados si la base de datos dispone de todos los datos para realizar la unión. Para crear una unión interna, se utiliza la cláusula `JOIN` o `INNER JOIN`.

Si esta consulta se ejecutara en las tablas de la diapositiva anterior, **el artículo con el id 5 no se añadiría al conjunto de resultados porque su columna `image_id` tiene el valor `NULL`** (por lo que no se puede crear la unión).

Cómo funcionan las uniones si faltan datos

INNER JOIN

```
SELECT article.id, article.title, image.file  
FROM article  
JOIN image ON article.image_id = image.id;
```

result set (showing first 5 rows of 23)

id	title	file
1	Systemic Brochure	systemic-brochure.jpg
2	Forecast	forecast.jpg
3	Swimming Pool	swimming-pool.jpg
4	Walking Birds	birds.jpg
6	Micro-Dunes	micro-dunes.jpg

Cómo funcionan las uniones si faltan datos

LEFT OUTER JOIN

Una **unión externa izquierda** (**left outer join**) añade todos los datos solicitados de la tabla izquierda al conjunto de resultados. A continuación, utiliza *NULL* para los valores que no puede obtener de la tabla derecha. Para crear una unión externa izquierda, se utiliza la cláusula `LEFT JOIN` o `LEFT OUTER JOIN`.

Cómo funcionan las uniones si faltan datos

LEFT OUTER JOIN

Si se ejecuta esta consulta en las tablas anteriores, **el título del artículo que tiene el id 5 se añade al conjunto de resultados, pero el valor de la columna archivo recibe el valor NULL** porque no se encuentran datos correspondientes para la imagen.

Cómo funcionan las uniones si faltan datos

LEFT OUTER JOIN

```
SELECT article.id, article.title, image.file  
FROM article  
LEFT JOIN image ON article.image_id = image.id;
```

result set (showing first 5 rows of 24)

id	title	file
1	Systemic Brochure	systemic-brochure.jpg
2	Forecast	forecast.jpg
3	Swimming Pool	swimming-pool.jpg
4	Walking Birds	birds.jpg
5	Sisters	NULL

Obtención de datos de varias tablas

Un comando `SELECT` puede ir seguido de varias cláusulas `JOIN` para **recopilar datos de más de dos tablas**.

Obtención de datos de varias tablas

Para recopilar datos de varias tablas, se puede añadir más de una cláusula `JOIN` :

- Después de la sentencia `SELECT` , indica los nombres de todas las columnas de las que deseas obtener datos (utilizando el nombre de la tabla, un punto y, a continuación, el nombre de la columna).
- Utiliza una cláusula join para establecer la relación entre los datos de cada una de las tablas.

La siguiente consulta recopila datos sobre un artículo a partir de tres tablas: article (artículo), category (categoría) e image (imagen).

Obtención de datos de varias tablas

```
SELECT article.title, article.summary,  
       category.name,  
       image.file, image.alt  
FROM article  
JOIN category  ON article.category_id = category.id  
LEFT JOIN image ON article.image_id   = image.id  
ORDER BY article.id ASC;
```

result set (showing first 3 rows of 24)

title	summary	name	file	alt
Systemic Brochure	Brochure design for...	Print	systemic-brochure.jpg	Brochure...
Forecast	Handbag illustration...	Illustration	forecast.jpg	Illustrati...
Swimming Pool	Architecture magazine...	Photography	swimming-pool.jpg	Photograph...

Obtención de datos de varias tablas

La tabla de artículos es la tabla de la izquierda. Contiene el título y el resumen de los artículos.

La tabla categoría proporciona el nombre de la categoría a la que pertenece cada artículo. **Como cada artículo debe estar en una categoría**, se utiliza la cláusula `JOIN` .

La tabla de imágenes proporciona el nombre de archivo y el texto alternativo de la imagen que se utiliza con cada artículo. **Como no es necesario que cada artículo tenga una imagen**, se utiliza una cláusula `LEFT JOIN` para **garantizar que todos los datos disponibles se añaden al conjunto de resultados independientemente de que tengan una imagen asociada o no**.

Ejemplo: usando múltiples join

La consulta que se muestra a continuación recoge datos sobre varios artículos que pertenecen a una categoría determinada.

1. La sentencia `SELECT` va seguida de los nombres de las columnas que deben añadirse al conjunto de resultados. Los datos se recogen de las tablas de artículos, categorías e imágenes.
2. La cláusula `FROM` indica que la tabla de la izquierda es la tabla de artículos.
3. La primera unión indica que los datos de la tabla categoría deben proceder de la fila cuya columna `id` tiene el mismo valor que la columna `category_id` de la tabla artículo.

Ejemplo: usando múltiples join

4. La segunda unión indica que los datos de la tabla de imágenes deben seleccionarse de la fila cuya columna `id` tiene el mismo valor que la columna `image_id` de la tabla de artículos. Se trata de un `LEFT JOIN`, por lo que se utiliza `NULL` para cualquier dato que falte.
5. La cláusula `WHERE` restringe los resultados a las filas de datos en las que la tabla `article` tiene un valor de `3` en la columna `category_id` y `1` en la columna `published`.
6. La cláusula `ORDER BY` controla el orden en que los resultados se añaden al conjunto de resultados, utilizando los ids de los artículos en orden descendente.

Ejemplo: usando múltiples join

```
① [ SELECT article.id, article.title,  
    category.name,  
    image.file, image.alt  
② FROM article  
③ JOIN category ON article.category_id = category.id  
④ LEFT JOIN image ON article.image_id = image.id  
⑤ [ WHERE article.category_id = 3  
    AND article.published = 1  
⑥ ORDER BY article.id DESC;
```

result set

id	title	name	file	alt
21	Stargazer	Illustration	stargazer-masc...	Illustrat...
17	Snow Search	Illustration	snow-search.jpg	Illustrat...
10	Polite Society...	Illustration	polite-society...	Mural for...
5	Sisters	Illustration	NULL	NULL
4	Walking Birds	Illustration	birds.jpg	Collage...
2	Forecast	Illustration	forecast.jpg	Illustrat...

Actividad C11.6. Usando múltiples join

Realiza las siguientes consultas sobre la base de datos de ejemplo, teniendo en cuenta el ejemplo anterior:

- Obtener los mismos datos de los artículos que están en la categoría que tiene un id de 2.
- Añadir el nombre y apellidos del autor desde la tabla de miembros.

Alias

Los **alias de tabla** facilitan la lectura de las consultas que utilizan uniones. Los **alias de columna** especifican los nombres de las columnas del conjunto de resultados.

Alias

En las consultas SQL complejas en las que las uniones seleccionan datos de varias tablas, puedes dar a cada nombre de tabla un alias.

Un alias de tabla es como **una abreviatura del nombre de una tabla y reduce la cantidad de texto de la consulta.**

Un alias de tabla se crea después de los comandos `FROM` o `JOIN`. Tras el nombre de la tabla, utiliza el comando `AS` y, a continuación, especifica un alias para esa tabla. A continuación, en el resto de la consulta, utiliza el alias en lugar del nombre completo de la tabla.

Alias

```
SELECT t1.column1, t1.column2, t2.column3
FROM table1 AS t1
JOIN table2 AS t2 ON t1.column4 = t2.column1;
```

CREATE ALIAS ALIAS FOR TABLE 1 ALIAS FOR TABLE 2

Alias

Los nombres de las columnas del conjunto de resultados suelen tomarse de los nombres de las columnas de las tablas de las que se recogieron los datos.

Los alias de columna permiten cambiar el nombre de una columna del conjunto de resultados. También pueden dar un nombre a una columna que contenga el resultado de una función `COUNT ()`.

Alias

Para crear un alias de columna, después de especificar el nombre de una columna de la que se desea obtener datos, utiliza el comando `AS` y el nombre que debe utilizar la columna en el conjunto de resultados.

O bien, después de la función `COUNT()` , utiliza el comando `AS` y un nombre de columna para el recuento en el conjunto de resultados.

Alias

DATABASE COLUMN ALIAS FOR RESULT SET

SELECT *column1* AS *newname1*
FROM *table*;

COUNT FUNCTION ALIAS FOR RESULT SET

SELECT COUNT(*) AS members
FROM members;

Ejemplo: Usando alias

Esta consulta obtiene los mismos datos que el ejemplo anterior, pero especifica alias para los nombres de las tablas después de los comandos `FROM` y `JOIN` :

- `a` para la tabla de artículos
- `c` para la tabla de categorías
- `i` para la tabla de imágenes

Los alias se utilizan en lugar de los nombres de tabla completos tras el comando `SELECT` y tras las cláusulas `WHERE` , `AND` y `ORDER BY` .

Ejemplo: Usando alias

```
SELECT a.id, a.title,  
       c.name,  
       i.file, i.alt  
  
FROM article AS a  
JOIN category AS c ON a.category_id = c.id  
LEFT JOIN image AS i ON a.image_id = i.id  
  
WHERE a.category_id = 3  
      AND a.published = 1  
ORDER BY a.id DESC;
```

result set

id	title	name	file	alt
21	Stargazer	Illustration	stargazer-masc...	Illustrat...
17	Snow Search	Illustration	snow-search.jpg	Illustrat...
10	Polite Society...	Illustration	polite-society...	Mural for...
5	Sisters	Illustration	NULL	NULL
4	Walking Birds	Illustration	birds.jpg	Collage...
2	Forecast	Illustration	forecast.jpg	Illustrat...

Ejemplo: Usando alias

Este otro ejemplo selecciona los valores de las columnas nombre y apellidos de la tabla de miembros y utiliza alias para darles nuevos nombres de columna en el conjunto de resultados.

Ejemplo: Usando alias

```
SELECT forename AS firstname, surname AS lastname  
FROM member;
```

result set	
firstname	lastname
Ivy	Stone
Luke	Wood
Emiko	Ito

Actividad C11.7. Usando alias

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Cambia los alias a: `art` para la tabla de artículos `cat` para la tabla de categorías `img` para la tabla de imágenes.
- Cuenta el número de artículos en la tabla de artículos y utiliza un alias para llamar a la columna `artículos`.

Combinar columnas y alternativas a nulos

`CONCAT ()` añade datos de dos columnas en una columna del conjunto de resultados.

`COALESCE ()` especifica un valor a utilizar si una columna contiene *NULL*.

La función `CONCAT ()` de SQL se utiliza para **tomar valores de dos o más columnas y unirlos** (o concatenarlos) **en una sola columna del conjunto de resultados**.

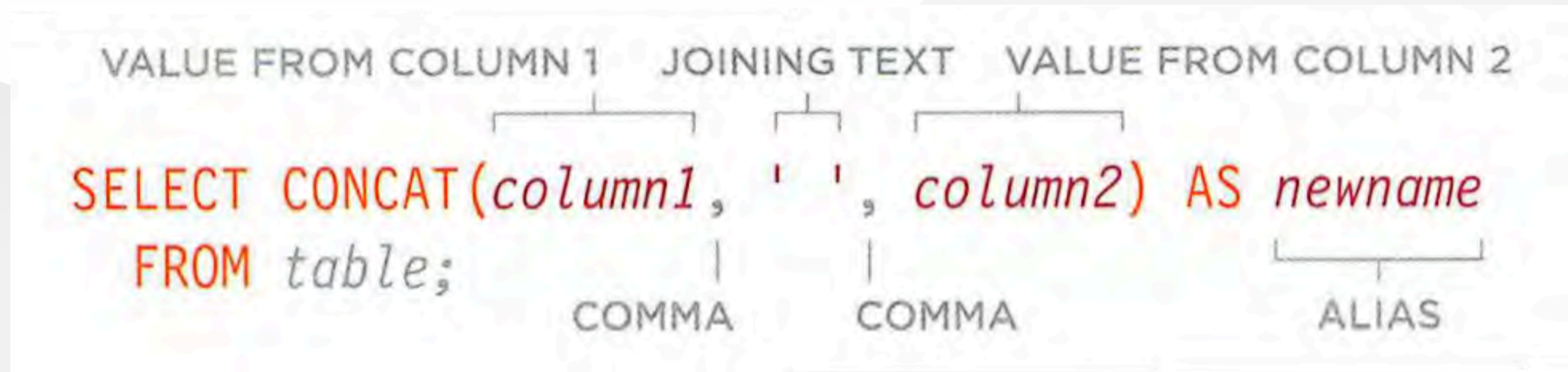
A menudo, se inserta una cadena entre los valores de las dos columnas para separarlos. Cuando se unen valores de dos columnas, **se utiliza un alias para especificar el nombre de esa columna** en el conjunto de resultados.

Combinar columnas y alternativas a nulos

Como en cualquier función, una coma separa los argumentos que la función `CONCAT()` une para crear el nuevo valor. A continuación, se unen los valores de dos columnas y se utiliza un espacio para separar los datos de las dos columnas.

Si el valor de una columna es `NULL`, los valores de las otras columnas también se tratarán como `NULL`.

Combinar columnas y alternativas a nulos



Combinar columnas y alternativas a nulos

Si sabes que un valor de una columna puede ser nulo (NULL), puedes utilizar la función `COALESCE()` de SQL para indicar:

- El nombre de otra columna cuyo valor puede intentar utilizar en su lugar (si este valor no es NULL, se utilizará en su lugar).
- Un valor por defecto a utilizar si todas las columnas especificadas tienen un valor NULL

Combinar columnas y alternativas a nulos

Cuando se añade una fila al conjunto de resultados, si el valor de la primera opción de columnas es NULL, se comprobará el valor de la segunda opción de columnas. Si los valores de todas las columnas alternativas también son NULL, utilizará el valor por defecto.

Cuando se utiliza la función `COALESCE()`, se debe proporcionar un alias para el nombre de la columna en el conjunto de resultados, ya que los datos podrían proceder de varias columnas.

Combinar columnas y alternativas a nulos

```
SELECT COALESCE(column1, column2, default) AS newname
FROM table;
```

1ST CHOICE OF DATA 2ND CHOICE OF DATA DEFAULT VALUE ALIAS

Ejemplo: concat() y coalesce()

El siguiente ejemplo utiliza la función `CONCAT()` para **unir los valores de las columnas nombre y apellidos en una columna llamada autor** en el conjunto de resultados.

Se ha añadido un espacio entre los valores de las dos columnas para garantizar que aparezca un espacio entre el nombre y los apellidos.

```
SELECT CONCAT(forename, ' ', surname) AS author  
FROM member;
```

result set
author
Ivy Stone
Luke Wood
Emiko Ito

Ejemplo: concat() y coalesce()

En este ejemplo se utiliza la función `COALESCE()` de SQL para especificar valores alternativos que pueden utilizarse si el valor de la columna de perfil de la tabla de miembros es NULL (porque el usuario no ha cargado una imagen de perfil). En este caso, la sentencia `SELECT` busca

- Un valor en la columna de imagen de la tabla de miembros.
- Si el valor es `NULL`, se utilizará el valor en forename
- Si es `NULL`, se mostrará en su lugar el texto friend.

Un alias especifica el nombre que debe utilizar esta columna en el conjunto de resultados. En este caso, se denomina `profile`.

Ejemplo: concat() y coalesce()

```
SELECT COALESCE(picture, forename, 'friend') AS profile  
FROM member;
```

result set
profile
ivy.jpg
Luke
emi.jpg

Actividad C11.8. Concat() y Coalesce()

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Añade la dirección de correo electrónico a los valores seleccionados y llama al alias `author_details`.
- Si el usuario no ha proporcionado una imagen, utiliza en su lugar el valor predeterminado `placeholder.png`.

Ejemplo: Consultas de artículos para el CMS de ejemplo

El CMS utiliza consultas SQL para **mostrar información sobre un artículo individual, y los resúmenes de un conjunto de artículos**. Estas consultas reúnen las técnicas que has visto a lo largo de este capítulo.

Ejemplo: Consultas de artículos para el CMS de ejemplo

1. La siguiente consulta SQL **obtiene datos sobre un artículo individual** de las cuatro tablas de la base de datos. La sentencia `SELECT` va seguida de los nombres de las columnas de las que se recogerán los datos.
2. Los **alias de columna** se utilizan para dar al nombre de la categoría y al nombre de archivo y texto alternativo de la imagen nuevos nombres de columna en el conjunto de resultados.

Ejemplo: Consultas de artículos para el CMS de ejemplo

3. La función `CONCAT()` une el nombre y el apellido del miembro que escribió el artículo, y un alias de columna dice que el conjunto de resultados debe almacenar el nombre en una columna llamada `author`.
4. La **tabla de la izquierda** es la tabla de artículos. Tres uniones muestran las relaciones con los datos de otras tablas.

Ejemplo: Consultas de artículos para el CMS de ejemplo

Después de los comandos `FROM` y `JOIN`, cada nombre de tabla recibe un **alias de tabla**. Cada vez que la consulta especifica una columna de datos, se utilizan estos alias en lugar de los nombres completos de las tablas.

5. La cláusula `WHERE` especifica el id del artículo a recoger. Sólo se devuelve si la columna `published` tiene el valor 1.

Ejemplo: Consultas de artículos para el CMS de ejemplo

```
① SELECT a.title, a.summary, a.content, a.created, a.category_id, a.member_id,  
②      c.name      AS category,  
③      CONCAT(m.forename, ' ', m.surname) AS author,  
②      i.file      AS image_file,  
②      i.alt       AS image_alt  
  
④ { FROM article      AS a  
   { JOIN category    AS c  ON a.category_id = c.id  
   { JOIN member      AS m  ON a.member_id   = m.id  
   { LEFT JOIN image   AS i  ON a.image_id    = i.id  
⑤ { WHERE a.id        = 22  
   { AND a.published  = 1;
```

result set

title	summary	content	created	category_id	member_id	category	author	image_file	image_alt
Polite... Poster...	These...	2021-0...	1	1	Print	Ivy St...	polite-so...	Photogra...	

Ejemplo: Consultas de artículos para el CMS de ejemplo

1. La siguiente consulta SQL **obtiene información resumida sobre todos los artículos de una categoría especificada**, utilizando datos de las cuatro tablas de la base de datos.

La sentencia `SELECT` va seguida de los nombres de las columnas de las que se recogerán los datos.

Ejemplo: Consultas de artículos para el CMS de ejemplo

2. Como en el ejemplo de la izquierda, los alias dan al nombre de la categoría, y al archivo de imagen y al texto alternativo de la imagen nuevos nombres de columna en el conjunto de resultados.

Se llama de nuevo a la función `CONCAT()` para unir el nombre y los apellidos del autor.

Ejemplo: Consultas de artículos para el CMS de ejemplo

3. Las uniones son idénticas a las del ejemplo de la izquierda.
4. La cláusula `WHERE` especifica el **id de la categoría en la que deben estar los artículos** y que el **artículo debe estar publicado**. Los resultados se ordenan por el id del artículo en orden descendente.

Ejemplo: Consultas de artículos para el CMS de ejemplo

```
① SELECT a.id, a.title, a.summary, a.category_id, a.member_id,  
②     c.name      AS category,  
     CONCAT(m.forename, ' ', m.surname) AS author,  
     i.file       AS image_file,  
     i.alt        AS image_alt  
  
③     FROM article    AS a  
     JOIN category    AS c  ON a.category_id = c.id  
     JOIN member      AS m  ON a.member_id   = m.id  
     LEFT JOIN image  AS i  ON a.image_id    = i.id  
  
④     WHERE a.category_id = 1  
     AND a.published      = 1  
     ORDER BY a.id DESC;
```

Ejemplo: Consultas de artículos para el CMS de ejemplo

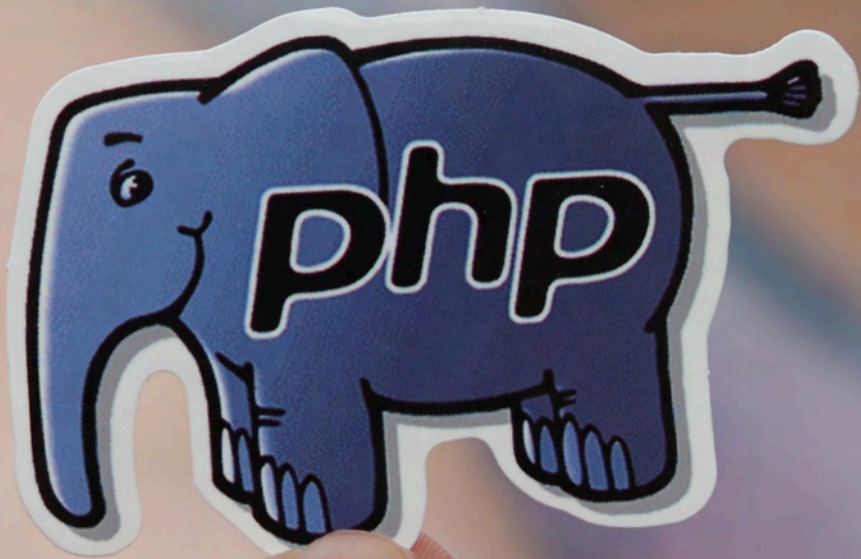
result set

id	title	summary	category_id	member_id	category	author	image_file	image_alt
24	Travel Guide	Book de...	1	1	Print	Ivy Stone	feathervi...	Two page...
22	Polite Societ...	Poster...	1	1	Print	Ivy Stone	polite-so...	Photogra...
20	Chimney Busin...	Station...	1	2	Print	Luke Wood	chimney-c...	Business...
14	Milk Beach Al...	Packagi...	1	1	Print	Ivy Stone	milk-beac...	Vinyl LP...
12	The Ice Palace	Book co...	1	2	Print	Luke Wood	the-ice-p...	The Ice...
1	Systemic Broc...	Brochure...	1	2	Print	Luke Wood	systemic-...	Brochure...

Actividad C11.9. Consultas de artículos para el CMS de ejemplo

Realiza las siguientes consultas con SQL sobre la base de datos de ejemplo:

- Selecciona los mismos datos sobre cada artículo, pero utiliza una cláusula `WHERE` diferente para seleccionar los artículos escritos por un miembro individual del sitio.
- Selecciona los mismos datos sobre cada artículo, pero utiliza la cláusula `LIMIT` para obtener los seis artículos publicados más recientemente en la base de datos (en cualquier categoría).
- Selecciona los mismos datos de cada artículo, pero utiliza la cláusula `LIKE` de SQL para obtener los datos de los artículos cuyo título contenga el término `design`.



3. Añadiendo datos (CREATE)

Añadiendo datos a la base de datos

El comando `INSERT INTO` de SQL añade una nueva fila de datos a una tabla. Sólo se pueden añadir datos a una tabla cada vez.

Añadiendo datos a la base de datos

El comando INSERT INTO indica a la base de datos que inserte datos en una única tabla.
Va seguido de:

- El nombre de la **tabla** a la que desea añadir los datos
- Paréntesis que contienen los nombres de las **columnas** a las que se desea añadir los datos

Añadiendo datos a la base de datos

El comando `VALUES` va seguido de paréntesis que contienen los **valores que se desean añadir a las columnas**.

Los valores deben aparecer en el mismo orden en que se especificaron las columnas. Las cadenas deben ir entre comillas; los números no.

Diagram illustrating the SQL `INSERT INTO` statement structure with annotations:

```
INSERT INTO table (column1, column2)
VALUES ('value1', 'value2');
```

Annotations:

- WHERE THE DATA GOES** points to `table`.
- THE VALUES TO ADD** points to `VALUES`.
- TABLE** is labeled above `table`.
- COLUMNS** is labeled above `(column1, column2)`.
- NEW VALUES** is labeled below `('value1', 'value2')`.

Añadiendo datos a la base de datos

En la base de datos de ejemplo, la columna `id` de cada tabla es la clave primaria de esa tabla, por lo que **cada fila necesita un valor único en la columna id.**

Para asegurar que el valor de la columna `id` es único, se crea utilizando una característica de MySQL llamada **autoincremento**. **Genera un número para la columna y asegura que es único incrementando ese número en 1 cada vez que se añade una nueva fila a la tabla.**

Debido a que este valor es creado por la base de datos, **cuando se añade una nueva fila a una tabla, no se especifica el nombre de la columna id o un valor para esta columna.**

Añadiendo datos a la base de datos

Otras cuatro columnas tienen valores por defecto, lo que significa que no es necesario especificar un valor para ellas al añadir una fila de datos. El valor por defecto de...

- la columna `created` de la tabla `article` es la **fecha y hora en que se añadió la fila a la base de datos**.
- La columna `image_id` de la tabla `article` es `NULL` **Si no se proporciona ninguna imagen, esta columna contendrá NULL**.
- La columna `published` de la tabla `article` es `0` **. Si esta columna no es 1, el artículo no está publicado**.
- La columna `joined` de la tabla `member` es la **fecha y hora en que la fila se añadió a la base de datos**.

Añadiendo datos a la base de datos

El SQL que se muestra a continuación **añade una categoría** llamada `News` a la tabla de categorías.

El SQL debe especificar un valor para las columnas `name`, `description` y `navigation`.
No debe proporcionar un valor para la columna `id` porque la base de datos la añade utilizando la función de autoincremento.

La nueva fila aparece resaltada en la tabla de categorías que se muestra junto a la sentencia SQL.

Añadiendo datos a la base de datos

```
INSERT INTO category (name, description, navigation)
VALUES ('News', 'Latest news from Creative Folk', 0);
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	Latest news from Creative Folk	0

Añadiendo datos a la base de datos

El siguiente ejemplo **añade tres filas a la tabla de imágenes**, cada una con detalles de una imagen diferente.

Para cada imagen, el SQL debe especificar el **nombre** del archivo y el **texto alternativo**. **No debe proporcionar un valor para la columna id**, ya que la base de datos la añade utilizando la función de autoincremento.

Los valores de cada fila se colocan entre paréntesis, igual que cuando se añade una fila a la base de datos.

Cada paréntesis está separado por una coma. Hay un punto y coma (sin coma) después de la última fila de datos.

Las nuevas filas se han resaltado en la tabla `image`.

Añadiendo datos a la base de datos

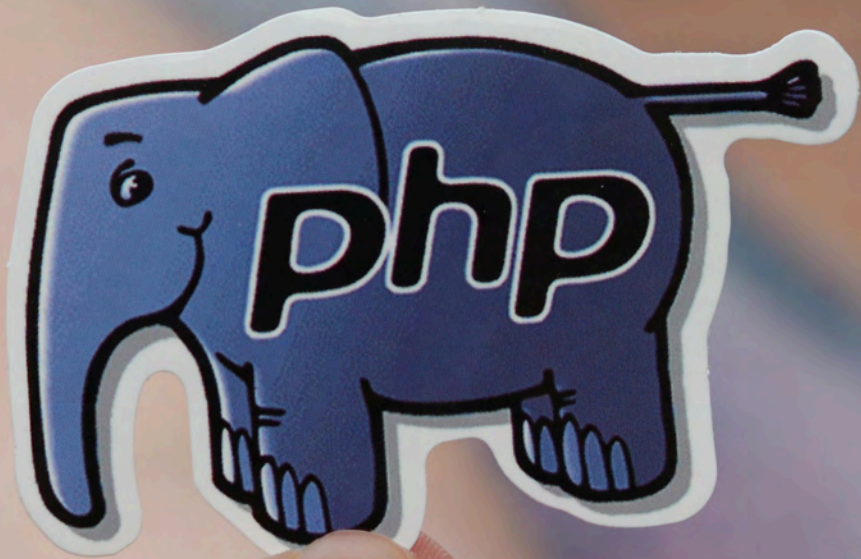
```
INSERT INTO image (file, alt)
VALUES ('bicycle.jpg', 'Photo of bicycle'),
       ('ghost.png', 'Illustration of ghost'),
       ('stamp.jpg', 'Polite Society stamp');
```

image		
id	file	alt
22	polite-society-posters.jpg	Photograph of three posters...
23	golden-brown.jpg	Photograph of the interior...
24	featherview.jpg	Two pages from a travel boo...
25	bicycle.jpg	Photo of bicycle
26	ghost.png	Illustration of ghost
27	stamp.jpg	Polite Society stamp

Añadiendo datos a la base de datos

Los datos adicionales que estos ejemplos añaden a la base de datos se eliminarán en los ejemplos restantes de esta unidad.

Es necesario que ejecutes todos los ejemplos en el mismo orden en que aparecen en las diapositivas. Si no lo haces, te encontrarás con errores más adelante.



4. Actualizando datos (UPDATE)

Actualizando datos en la base de datos

El comando UPDATE de SQL permite actualizar los datos de la base de datos. El comando SET indica las columnas a actualizar y sus nuevos valores. La cláusula WHERE controla qué fila(s) de la tabla debe(n) actualizarse.

Actualizando datos en la base de datos

El comando UPDATE indica a la base de datos que desea actualizar los datos de la base de datos. Va **seguido del nombre de la(s) tabla(s) que se desea actualizar**.

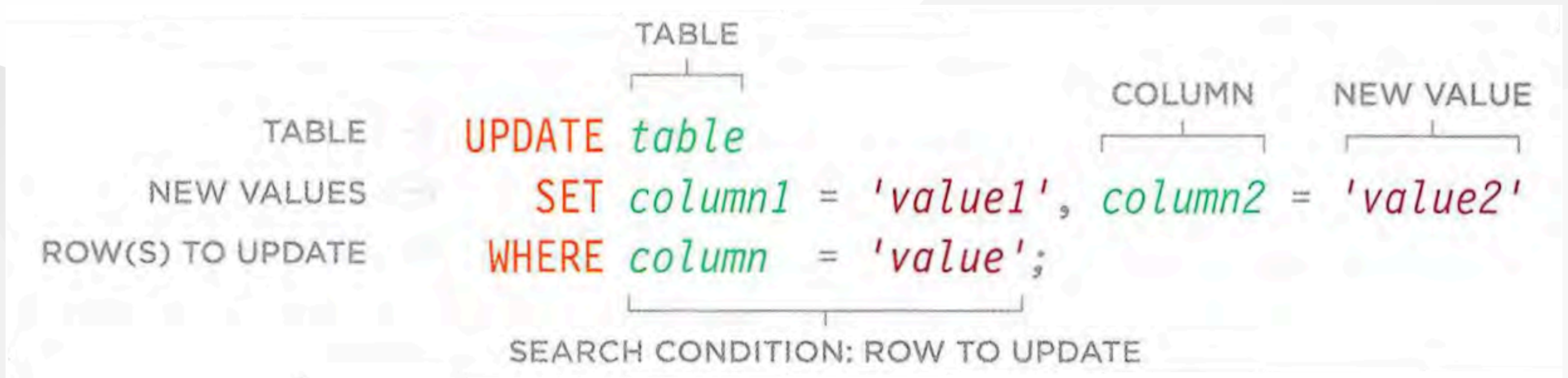
A continuación, se utiliza el comando `SET` para **especificar las columnas que se desean actualizar y sus nuevos valores**. Sólo es necesario indicar los nombres y valores de las columnas que se desean actualizar (las demás columnas conservarán los valores que ya tienen).

Actualizando datos en la base de datos

La cláusula `WHERE` se utiliza para **especificar qué filas deben actualizarse**, del mismo modo que se utiliza cuando se solicitan filas específicas de datos a la base de datos. (Si no se utiliza, se actualizarán todas las filas de la tabla).

Si la condición de búsqueda coincide con más de una fila, todas las filas con las que coincide se actualizan con los mismos valores. **Para actualizar datos en varias tablas al mismo tiempo**, la sentencia SQL utilizaría un comando `JOIN`.

Actualizando datos en la base de datos



Actualizando datos en la base de datos

A menudo, sólo se desea actualizar una fila cada vez. En estos casos, la cláusula `WHERE` utilizaría la clave primaria de la tabla para especificar qué fila debe actualizarse. En la base de datos de ejemplo, la clave primaria de cada tabla es el valor de la columna `id`.

Actualizando datos en la base de datos

Para **actualizar varias filas de una tabla**, utiliza una condición de búsqueda que seleccione más de una fila. Por ejemplo, **para ocultar todos los artículos de un autor**, el valor de la columna publicado se actualizaría a 0, y la condición de búsqueda especificaría el `id` del autor.

Actualizando datos en la base de datos

COPIA DE SEGURIDAD DE UNA BASE DE DATOS

Dado que una sentencia SQL puede actualizar muchas filas de la base de datos, es una buena idea crear una copia de seguridad de la base de datos antes de ejecutar nuevas consultas.

Si la consulta SQL afecta accidentalmente a más datos de los que debía, se puede utilizar la copia de seguridad para restaurar los datos originales antes de ejecutar la consulta.

Actualizando datos en la base de datos

COPIA DE SEGURIDAD DE UNA BASE DE DATOS

Para crear una copia de seguridad de la base de datos en *phpMyAdmin*:

1. Selecciona la base de datos.
2. Haz clic en la pestaña *Exportar*.
3. Utiliza las opciones mostradas por defecto y pulsa *Ir*. Se generará un código SQL, que deberás guardar en un archivo de texto. Este archivo es como el que utilizaste para crear la base de datos.

Actualizando datos en la base de datos

El SQL que se muestra a continuación **actualiza la fila que se añadió a la tabla de categorías en el ejemplo anterior**. Sólo funciona con esta fila porque la cláusula `WHERE` especifica que la categoría debe tener un id de `5`.

Cambia el valor de la columna `name` a *Blog* y el valor de la columna `navigation` a `1`.

Puedes ver que la fila actualizada está resaltada en la tabla de categorías que acompaña al SQL.

Actualizando datos en la base de datos

```
UPDATE category  
  SET name = 'Blog', navigation = 1  
 WHERE id = 5;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	Blog	Latest news from Creative Folk	1

Actualizando datos en la base de datos

El siguiente ejemplo SQL **actualiza todas las filas de la tabla de categorías en las que la columna de navegación tiene el valor 1** (porque la cláusula `WHERE` especifica cualquier fila en la que la navegación = 1).

Actualiza el valor de la columna de navegación a `0`. Esto evita que todas las categorías se muestren en la barra de navegación.

Hay ocasiones en las que querrás ofrecer a los usuarios la posibilidad de afectar a varias filas de una tabla, pero este ejemplo también destaca la importancia de asegurarte de que tu SQL sólo actualiza las filas que quieres en una tabla.

Actualizando datos en la base de datos

```
UPDATE category  
  SET navigation = 0  
 WHERE navigation = 1;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	0
2	Digital	Powerful pixels	0
3	Illustration	Hand-drawn visual storytelling	0
4	Photography	Capturing the moment	0
5	Blog	Latest news from Creative Folk	0

Actividad C11.10

Utiliza un comando SQL en phpMyAdmin para volver a mostrar todas las categorías.

NOTA: Es necesario volver a activar las categorías en la base de datos para poder verlas en las unidades siguientes.



5. Borrar datos (DELETE)

Borrar datos de la base de datos

El comando `DELETE` de SQL elimina una o varias filas de una tabla. El comando `FROM` indica la tabla de la que se eliminan los datos. La cláusula `WHERE` indica qué fila(s) de la tabla debe(n) eliminarse.

Borrar datos de la base de datos

Puedes borrar una o varias filas de una tabla al mismo tiempo. En primer lugar, utiliza el comando `DELETE FROM`, seguido del nombre de la tabla de la que quieres borrar datos.

A continuación, utiliza una **condición de búsqueda** para especificar qué filas eliminar (si no lo haces, se eliminarán todas las filas de datos de la tabla). Para elegir una fila, la condición puede especificar la columna con una clave primaria.

The diagram illustrates the structure of the SQL `DELETE` command. It shows the command `DELETE FROM table WHERE column = 'value';` with color-coded keywords: `DELETE` and `FROM` are in orange, `table` is in green, `WHERE` is in orange, `column` is in green, and `'value'` is in red. Annotations include: a bracket above `table` labeled "TABLE"; an arrow pointing from the text "TABLE TO REMOVE DATA FROM" to `DELETE FROM`; an arrow pointing from the text "ROW(S) TO REMOVE" to `WHERE`; and a bracket below `column = 'value';` labeled "ROW TO UPDATE".

```
TABLE TO REMOVE DATA FROM → DELETE FROM table
ROW(S) TO REMOVE → WHERE column = 'value';
                                ROW TO UPDATE
```

Borrar datos de la base de datos

Si la cláusula `WHERE` coincide con más de una fila de datos de la tabla, cada una de las filas se elimina de la tabla.

No se pueden eliminar valores de columnas individuales de la base de datos utilizando el comando `DELETE`. En su lugar, utiliza `UPDATE` y establece el valor de la columna en `NULL`.

Borrar datos de la base de datos

El siguiente ejemplo SQL elimina la fila de la tabla de categorías, donde la columna `id` tiene el valor 5, que es la categoría `Blog` añadida en los ejemplos anteriores.

La fila resaltada se elimina de la tabla.

Si la cláusula `WHERE` coincide con más de una fila de datos, **elimina todas las filas de datos coincidentes** de la tabla.

Borrar datos de la base de datos

```
DELETE FROM category  
WHERE id = 5;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	Blog	Latest news from Creative Folk	1

Borrar datos de la base de datos

ATENCIÓN ! NO EJECUTES ESTE EJEMPLO !!

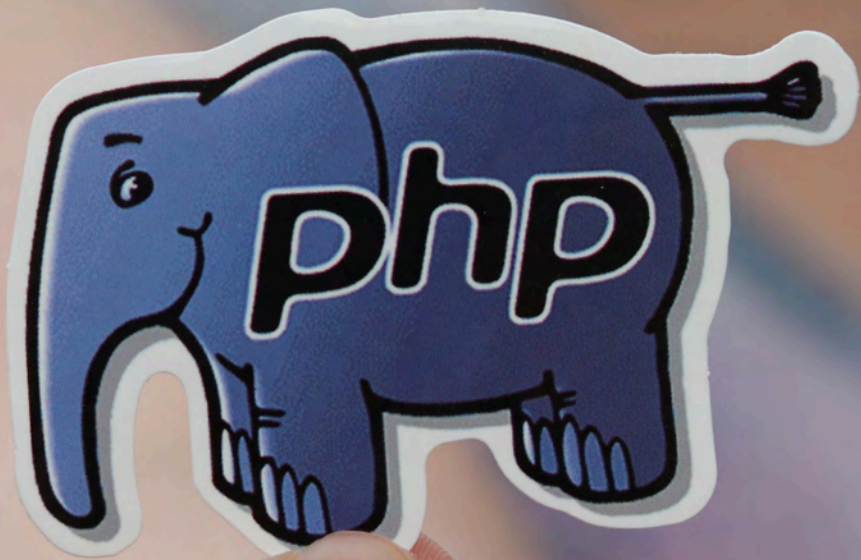
Es importante tener cuidado de no borrar más datos de los que se desea con un comando `DELETE` . Por ejemplo, la consulta SQL que se muestra en la siguiente diapositiva **borra todas las categorías que se utilizan en la navegación del sitio.**

Hacer una copia de seguridad de la base de datos antes de ejecutar una nueva sentencia SQL que borre datos de la base de datos **es una buena práctica** que garantiza que se disponga de una copia de los datos en caso de que la consulta no realice la acción prevista.

Borrar datos de la base de datos

```
DELETE FROM category  
WHERE navigation = 1;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1



6. Restricciones

Restricciones de unicidad

Los valores de algunas columnas deben ser únicos. Por ejemplo, no debe haber dos artículos con el mismo título, ni dos categorías con el mismo nombre, ni dos miembros con la misma dirección de correo electrónico.

Restricciones de unicidad

Si el valor de una columna se supone que es único, pero dos filas tienen el mismo valor en esa columna, se denomina **entrada duplicada**.

Para evitar que las filas de la base de datos tengan el mismo valor en una columna, se puede indicar a MySQL que aplique una **restricción de unicidad** (se llama así porque restringe los valores permitidos en esa columna para garantizar que sean únicos).

Las restricciones de unicidad sólo deben utilizarse cuando sean necesarias, ya que cada vez que se añaden datos, o se actualizan datos existentes, la base de datos tiene que comprobar todas las demás filas de esa columna para asegurarse de que el valor no existe ya. Esto requiere más capacidad de procesamiento y ralentiza la base de datos.

Restricciones de unicidad

A continuación, puedes ver cómo añadir una restricción de unicidad para los nombres de categoría en *phpMyAdmin* para asegurarte de que no hay dos categorías con el mismo nombre.

1. Selecciona la base de datos `phpbook-1` y luego la tabla de categorías en el menú de la izquierda.
2. Selecciona la pestaña *Structure*.
3. Cada fila de la tabla que aparece debajo representa una columna de la base de datos. En la fila que representa la columna `name`, haz clic en el menú desplegable *more*.
4. Haz clic en el enlace que dice *Unique*.

Ahora, si una sentencia SQL intentara añadir o actualizar una categoría utilizando el mismo nombre que otra ya existente, **la base de datos emitiría un error**. Aprenderás cómo manejar estas situaciones más adelante.

Restricciones de unicidad

The screenshot shows the phpMyAdmin interface. On the left, the database 'phpbook-1' is selected, indicated by a red circle with the number 1. The top navigation bar shows 'Server: localhost' (annotated with a red circle and number 2), 'Database: phpbook-1', and 'Table: category'. The 'Table structure' tab is active. The table structure is displayed in a table with the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(24)	utf8mb4_unicode_ci		No	None			Change Drop More
3	description	varchar(254)	utf8mb4_unicode_ci		No	None			Change Primary
4	navigation	tinyint(1)			No	None			Change Unique

At the bottom of the table structure view, there are buttons for 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Fulltext', 'Spatial', 'Index', 'Fulltext', and 'Distinct values'. The 'Unique' button is highlighted with a red circle and the number 4. The 'Drop' button is highlighted with a red circle and the number 3. The 'Primary' button is highlighted with a red circle and the number 2. The 'Unique' button is highlighted with a red circle and the number 4.

Restricciones de unicidad

En la base de datos de ejemplo, hay tres tablas que contienen cada una una columna que requiere una restricción de unicidad para garantizar que los valores que contiene son todos diferentes.

Se tratan de la columna de `title` de la tabla de artículos, la columna de `name` de la tabla de categorías y la columna `email` de la tabla de miembros.

Restricciones de clave foránea

Cuando existe una relación entre dos tablas, una **restricción de clave foránea** comprueba que el valor de una clave foránea sea una clave primaria válida en otra tabla.

Restricciones de clave foránea

Tres columnas de la tabla de artículos utilizan claves externas:

- `category_id` : el id de la categoría en la que está el artículo
- `member_id` : el id del miembro que lo escribió
- `image_id` : el id de la imagen del artículo

La tabla de artículos utiliza claves foráneas para:

- **Asegurarse de que cualquier valor añadido a estas columnas es una clave primaria en la tabla correspondiente** (si no lo es, la base de datos genera un error).
- **Impedir que se elimine una categoría, un miembro o una imagen si su clave principal se utiliza como clave externa en la tabla de artículos.**

Restricciones de clave foránea

Para **añadir una restricción de clave externa a una columna de una tabla**:

1. Selecciona la tabla que tiene la clave foránea.
2. Marca la casilla de la columna con la clave foránea.
3. Haz clic en *More* y luego en *Index* para añadir un índice a la columna. (Un índice es una copia de columnas seleccionadas en la tabla que acelera la búsqueda de datos en la tabla. Pero deben utilizarse con cuidado ya que pueden ocupar espacio extra y ralentizar la base de datos).
4. Selecciona la vista *Relación*.
5. Añade un nombre para la restricción.
6. Selecciona la columna con la clave foránea.
7. Selecciona la tabla y la columna que contienen la clave primaria. A continuación, haz clic en *Save* (no se ve este botón en la imagen mostrada a continuación).

Restricciones de clave foránea

phpMyAdmin

Recent Favorites

- New
- information_schema
- mysql
- performance_schema
- phpbook-1
 - New
 - article
 - category
 - image
 - member
- sys

Server: localhost3389 - Database: phpbook-1 - Table: article

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	title	varchar(80)	utf8mb4_unicode_ci		No	None			Change Drop More
3	summary	varchar(254)	utf8mb4_unicode_ci		No	None			Change Drop More
4	content	text	utf8mb4_unicode_ci		No				Change Drop More
5	created	timestamp			No	CURRENT_TIMESTAMP			Change Drop More
6	category_id	int(11)			No	None			Change Drop More
7	member_id	int(11)			No	None			Change Primary
8	image_id	int(11)			Yes	NULL			Change Unique
9	published	tinyint(4)			No	0			Change Index

Check all With selected: Browse Change Drop Primary Unique Index Fulltext Spatial Fulltext Distinct values

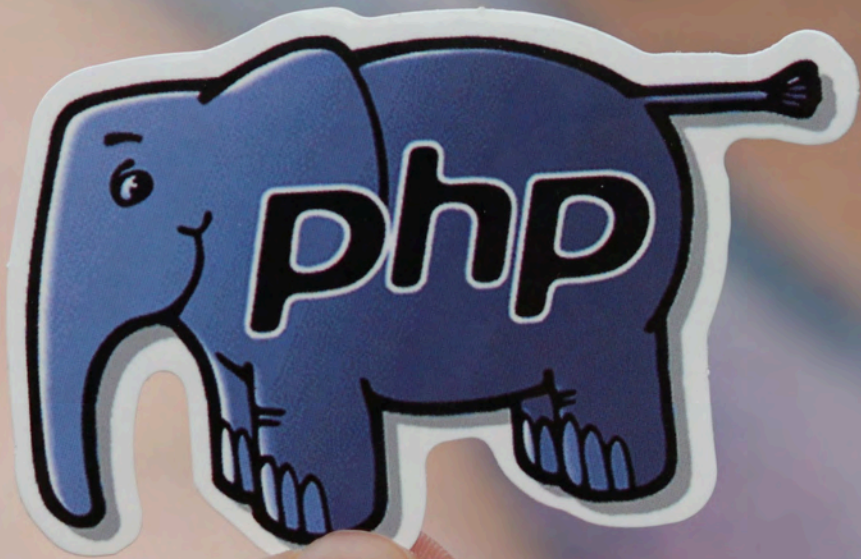
Print Propose table structure Move columns Normalize

Add 1 column(s) after published Go

Foreign key constraints

Actions Constraint properties

Drop	category_exists	Column	Foreign key constraint (INNODB)		
		Database	Table	Column	
		category_id	phpbook-1	category	id
		+ Add column			
ON DELETE	RESTRICT				
ON UPDATE	RESTRICT				



7. Resumen

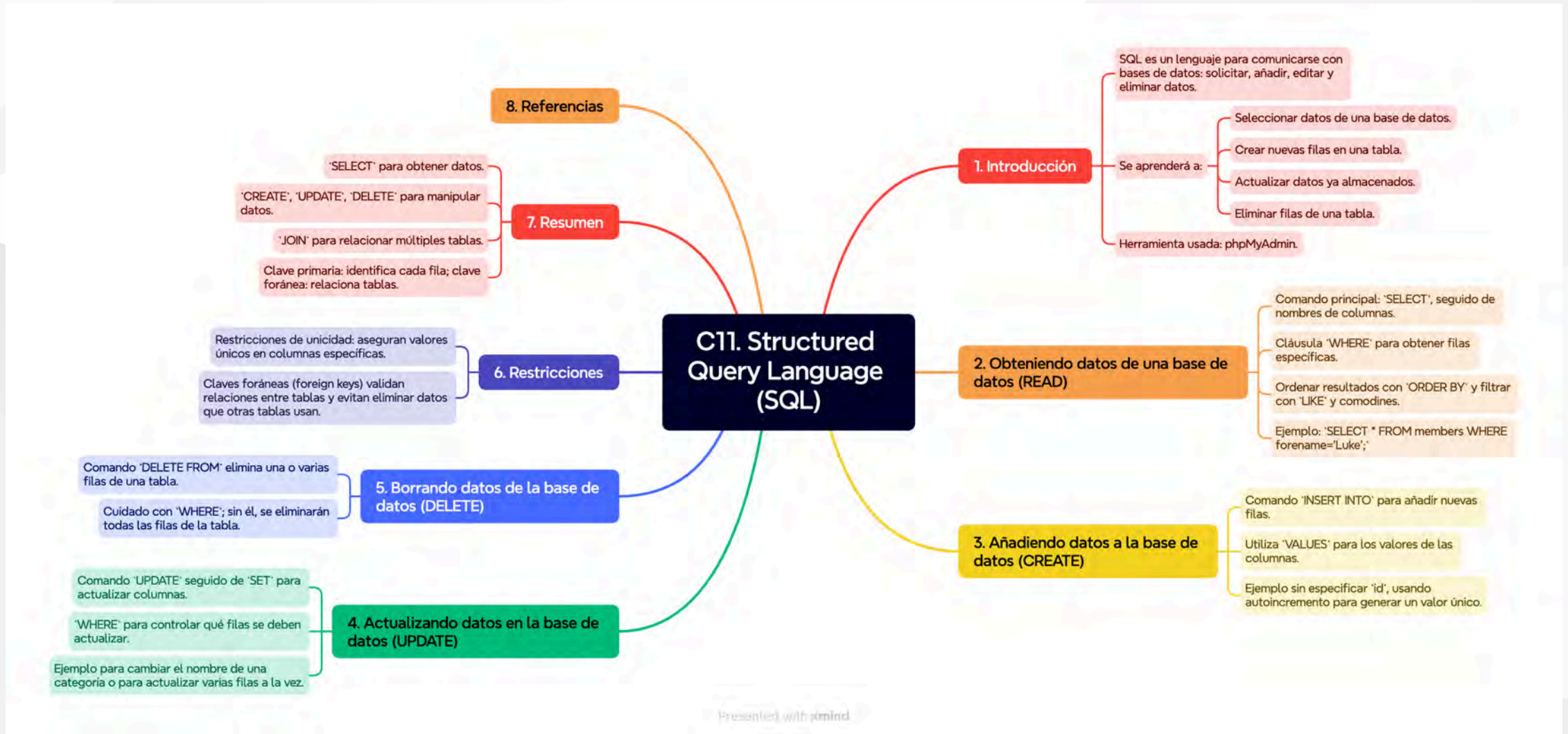
Resumen

- SQL se utiliza para la comunicación con las bases de datos.
- SELECT especifica las columnas de datos que deben recogerse de una base de datos. A continuación, los datos se añaden a un conjunto de resultados.
- Los comandos CREATE, UPDATE y DELETE se utilizan para crear, actualizar o eliminar filas de datos.
- FROM especifica una tabla con la que trabajar.

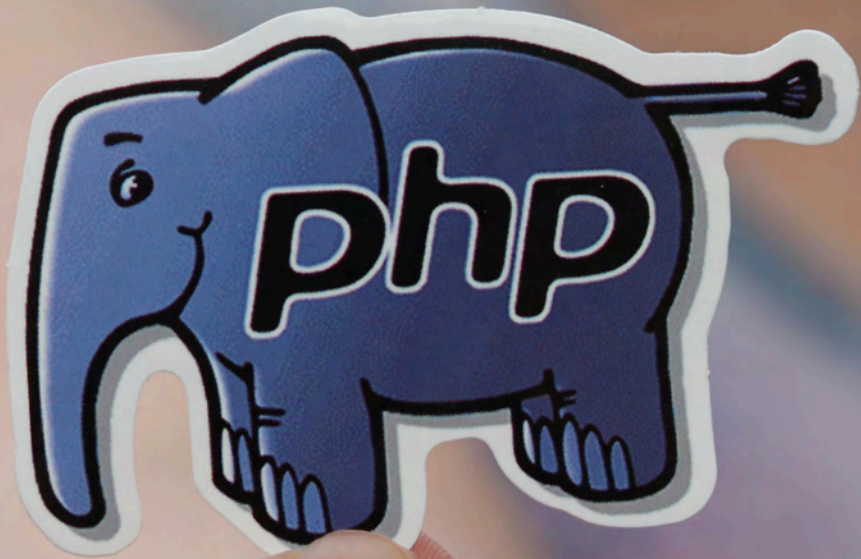
Resumen

- WHERE especifica las filas de datos con las que se va a trabajar.
- Los JOINS describen relaciones entre múltiples tablas.
- Una clave primaria es una columna que tiene un valor único para identificar cada fila. El valor puede crearse utilizando la función de autoincremento de MySQL.
- Una clave externa es una columna que almacena la clave primaria de otra tabla y describe su relación.
- Las restricciones evitan las entradas duplicadas y garantizan que una clave externa coincida con una clave primaria de otra tabla.

Resumen



Presented with pomind



8. Referencias

Referencias

1. W3Schools SQL Tutorial

W3Schools ofrece un tutorial completo sobre SQL, desde consultas básicas hasta temas avanzados.

[W3Schools SQL](#)

2. SQLZoo Interactive SQL Tutorial

Plataforma interactiva para practicar consultas SQL con ejercicios en tiempo real.

[SQLZoo](#)

3. Introduction to Databases and SQL - University of Michigan, Coursera

Curso introductorio que explora conceptos de bases de datos y SQL, con vídeos y ejercicios prácticos.

[Coursera Course](#)

Bloque C

Sitios web basados en bases de datos

C11. Structured Query Language (SQL)

