Clase 6: Material Complementario

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nelson Brian Avila Solano
Curso:	Curso de Backend Developer - Turno Noche	Día:	Tuesday, 18 de November de 2025, 19:29
Libro:	Clase 6: Material Complementario		

Tabla de contenidos

1. Introducción a las bases de datos

Temario:

- ¿Qué son las bases de datos?
- Tablas.
- Columnas.
- Registros.
- Tipos de datos.
- Relaciones.

1.1. Introducción a las bases de datos

¿Qué son las bases de datos?

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente. Se pueden clasificar según la variabilidad de los datos almacenados:

● **Bases de datos estáticas:** son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos

● **Bases de datos dinámicas:** son bases de datos donde la información almacenada se modifica con el tiempo, permite operaciones como actualización, adición , consulta y eliminación de datos.

Bases de datos relacionales

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

En este modelo, el lugar y la forma en que se almacenan los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red).

Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos.

Características

● Una base de datos relacional se compone de varias tablas o relaciones.

● No pueden existir dos tablas con el mismo nombre.

● Cada tabla es a su vez un conjunto de registros o filas.

● Cada registro representa un objeto del mundo real.

● No pueden existir dos campos o columnas con el mismo nombre en la misma tabla.

● Los valores almacenados en una columna deben ser del mismo tipo de dato.

● Todas las filas de una misma tabla poseen el mismo número de campos.

● No se considera el orden en que se almacenan los registros en las tablas.

● No se considera el orden en que se almacenan las tablas en la base de datos.

● La información puede ser recuperada o almacenada mediante "consultas"que ofrecen una amplia flexibilidad y poder para administrar la información.

Tablas

Tabla en las bases de datos, se refiere al tipo de modelado de datos donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de tablas

Las tablas se componen de dos estructuras:

● **Campo o columnas:** Corresponde al nombre de la columna. Debe ser único y además tener un tipo de dato asociado.

● **Registro:** Corresponde a cada fila que compone la tabla. Allí se componen los datos y los registros. Eventualmente pueden ser nulos en su almacenamientos. Son las casillas horizontales dentro de una tabla

Columnas

En el contexto de una tabla de base de datos relacional , una columna, campo o atributo es cada uno de los valores únicos (datos) que proporcionan la estructura según la cual se descomponen las filas (registros o tuplas) de una tabla.

Los datos de cada campo pueden ser de diferentes tipos , pero solo uno por columna: numéricos, alfanuméricos, lógicos (verdadero o falso), de texto, multimedia, binarios, etc.

Para evitar la inconsistencia de una estructura de campos, al diseñarla, se deben seguir las formas normales , sobre todo en bases de datos de gran tamaño.

Tabla: alumnos

id	nombre	apellido	edad	mail	celular
numérico	alfanumérico	alfanumérico	numérico	alfanumérico	alfanumérico

Registros

Un registro (también llamado fila) representa un objeto único de datos implícitamente estructurados en una tabla. En términos simples, una tabla de una base de datos puede imaginarse formada de filas y columnas (campos o atributos).

Cada fila de una tabla representa un conjunto de datos relacionados, y todas las filas de la misma tabla tienen la misma estructura. No puede haber un registro duplicado, los datos deben ser diferentes en al menos uno de los campos.

Tipos de datos

Una de las características principales de los campos, además de su nombre, es el tipo de dato que va a guardar. Esto define los posibles valores que se permitirán almacenar, la forma de guardarlos y hasta el espacio requerido para cada columna.

Tipos de datos con formato string

char: Son textos cuya longitud siempre será fija. Si un campo nombre es de longitud 50 y por ejemplo guardamos Flavia (6 caracteres) , los otros 44 no se ocupan.

varchar: Son textos cuya longitud es variable. Siguiendo el ejemplo anterior, si guardamos el nombre Flavia (6 caracteres) , los otros 44 no se desperdiciaron.

text: Este tipo de dato permite guardar textos muy extensos, hasta 2gb, son capaces de contener hasta un libro completo.

Tipos de datos numéricos

int (integer): Solo permiten valores positivos o negativos sin punto decimal.

decimal (numeric): Guarda valores con decimales.

float: número de coma flotante, este tipo de datos se suele utilizar para los valores en notación científica.

Fechas

date: Válido para almacenar una fecha con año, mes y día, su rango oscila entre ‘1000-01-01’ y ‘9999-12-31’.

datetime: Almacena una fecha (año-mes-día) y una hora (horasminutos-segundos), su rango oscila entre ‘1000-01-01 00:00:00’ y ‘9999-12-31 23:59:59’.

Relaciones

Claves Primarias

Son campos llave que identifican unívocamente registros que están dentro de la tabla. Sólo puede existir una clave primaria por tabla y ningún campo de dicha clave puede contener valores NULL.

Claves Foráneas

Son campos llave que permiten referenciar unívocamente a un registro que está dentro de otra tabla. Es una referencia a una clave en otra tabla. Necesitan no ser claves únicas.

Ejemplo

nombre	apellido	dirección	teléfono
Pablo	Martinez	Rivadavia 2564	4567-9852
Diego	Luca	Cuba 1234	4785-2369
Alfredo	Gomez	Medrano 200	4712-9865
Silvia	Conte	Corrientes 1452	2358-8965
Adelina	Romero	Belgrano 5698	4589-5623
Pablo	Martinez	San Pedrito 111	4578-4253
Diego	Lopez	Gascón 1239	4158-7423
Jorge Darío	Gassi	Uruguay 1478	4896-3265
Nicolás	Molda	Cucha cucha 1258	4289-6547

Si alguien nos preguntara si tenemos el teléfono del cliente Pablo Martínez, porque ocurrió un problema en la entrega de un producto en su domicilio y hay que contactarse con él, le preguntaremos a la base de datos:

- ¿Cuál es el teléfono del cliente con nombre Pablo y apellido Martínez?

La respuesta sería una tabla con los siguientes datos:

nombre	apellido	dirección	teléfono
Pablo	Martinez	Rivadavia 2564	4567-9852
Pablo	Martinez	San Pedrito 111	4578-4253

La respuesta que nos devolvió la BD es totalmente válida, los dos registros cumplen con la condición que le pedimos: el cliente Pablo Martínez. Para arreglar esto, tendríamos que preguntar cuál es la dirección del Pablo Martínez del que se necesita el teléfono y volver a generar la consulta, especificando un poco más:

- ¿Cuál es el teléfono del cliente con nombre Pablo, apellido Martínez y dirección Rivadavia 2564?

Ahora la respuesta sería una tabla con los siguientes datos:

nombre	apellido	dirección	teléfono
Pablo	Martinez	Rivadavia 2564	4567-9852

Logramos conseguir sólo el dato que nos hacía falta, pero tuvimos que usar muchas condiciones para lograr que la BD nos devolviera un registro único.

Para evitar este tipo de situaciones se usan los campos **Llave o clave** : son campos (columnas) cuyo contenido va a ser único a lo largo de toda la tabla.

A fin de evitar que los valores que se vayan a almacenar en este tipo de campo se dupliquen, por lo general se usan campos numéricos que la base de datos maneja cuidándose siempre de asignar un número no usado anteriormente. Un claro ejemplo de esto en la vida real es el DNI, el número de la tarjeta de crédito, las cuentas bancarias, etc.

¿Cómo aplicamos esto a nuestra tabla de clientes? Lo que podemos hacer es agregar un campo llave denominado "nro cliente" que represente un Número Único de Cliente, el cual utilizaremos para referirnos unívocamente a cada uno de ellos.

Entonces la tabla quedaría así:

Tabla: PEDIDOS

nro_cliente	nombre	apellido	dirección	teléfono
1	Pablo	Martinez	Rivadavia 2564	4567-9852
2	Diego	Luca	Cuba 1234	4785-2369
3	Alfredo	Gomez	Medrano 200	4712-9865
4	Silvia	Conte	Corrientes 1452	2358-8965
5	Adelina	Romero	Belgrano 5698	4589-5623
6	Pablo	Martinez	San Pedrito 111	4578-4253
7	Diego	Lopez	Gascón 1239	4158-7423
8	Jorge Dario	Gassi	Uruguay 1478	4896-3265
9	Nicolás	Molda	Cucha cucha 1258	4289-6547

En las Bases de Datos Relacionales, como su nombre lo indica, las relaciones entre las diversas tablas que la componen tienen un rol importante. Para indicar estas relaciones de una tabla a la otra, se utilizan los campos llave.

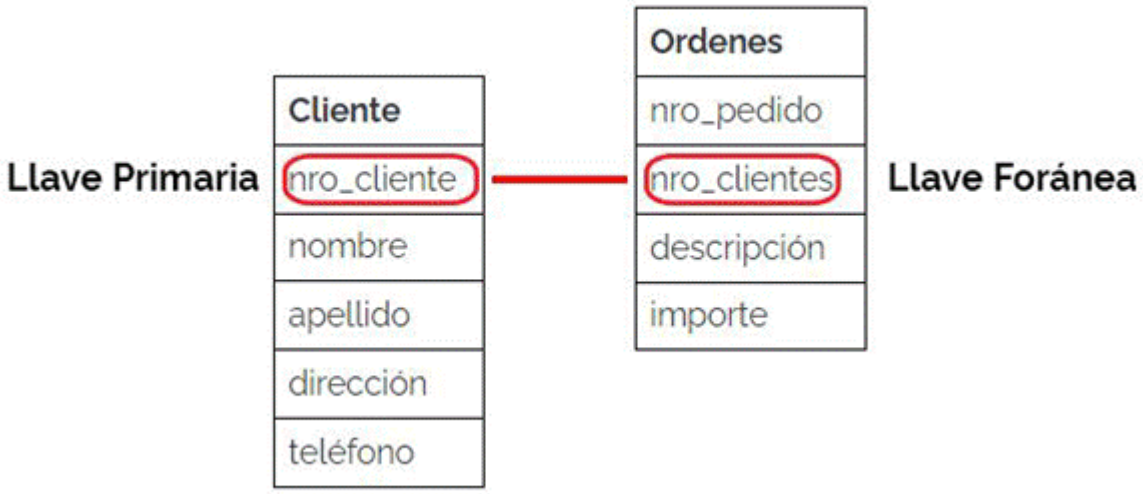
Tabla: Ordenes

nro_pedido	nro_cliente	descripción	importe
1	5	10 Resmas A4	3000
2	1	100 lapiceras	1000
3	6	22 Cuadernos Oficio	2420

Si analizamos la estructura de esta tabla, nos encontramos con dos campos llave: nro pedido y nro cliente.

Con “ nro pedido ” lo que logramos es identificar unívocamente al pedido realizado, y con “ nro cliente” , podemos ubicar al cliente que realizó la compra.

Si queremos saber quién compró “100 Resmas A4”, sólo hace falta ver el Número de Cliente que está en la columna “nro cliente ” y luego ir a la tabla Clientes para ver a quien le corresponde ese número, que en este caso es “Adelina Caraibo”.



Bibliografía:

- SQL Commands Disponible desde la URL:

http://www.postgresql.org/docs/9.1/static/sql-commands.html

- Tutorial de SQL Disponible desde la URL:

http://www.unalmed.edu.co/~mstabare/Sql.pdf

2. Introducción al lenguaje SQL

Temario:

- ¿Qué es?
- Consultas
- Agregar, Modificar o Eliminar registros

3. Introducción al lenguaje SQL

¿Qué es?

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Características generales de SQL

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros —y no a registros individuales— permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros.

SQL también tiene las siguientes características:

● **Lenguaje de definición de datos**: El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.

● **Lenguaje interactivo de manipulación de datos:** El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.

● **SQL incorporado y dinámico:** Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, PHP, COBOL, Pascal y Fortran.

● **Autorización:** El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

Tipos de datos

● **Varchar:** Recibe cadena de palabras compuestas de letras, números y caracteres especiales.

● **Int** es el principal tipo de datos de valores enteros de SQL Server. Con números enteros con o sin signo

● **Date:** una fecha de calendario que contiene el año (de cuatro cifras), el mes y el día.

● **Time:** La hora del día en horas minutos segundos (el valor predeterminado es 0).

Consultas

Es el lenguaje de consulta universal para bases de datos. Los mandatos de SQL se dividen en tres grandes grupos diferenciados.

● **DDL** (Data Definition Language): es el encargado de la definición de Bases de Datos, tablas, vistas e índices entre otros.

● **DML** (Data Manipulation Language): cuya misión es la manipulación de datos. A través de él podemos seleccionar, actualizar insertar, eliminar datos.

● **DCL** (Data Control Language): encargado de la seguridad de la base de datos, en todo lo referente al control de accesos y privilegios entre los usuarios.

Lenguaje de definición de datos de mysql

Para crear una base de datos es la consulta es la siguiente: CREATE DATABASE, se puede escribir en mayúscula o minúscula.

```
CREATE DATABASE NombreDeBaseDeDatos;
```

Para crear una tabla:

```
CREATE TABLE 'alumnos' ('nombre' VARCHAR(30), 'apellido'
VARCHAR(30), 'FechaNac' DATETIME, 'email' VARCHAR(30));
```

En primer momento colocamos el nombre de la tabla en este caso alumnos y entre paréntesis se debe especificar el nombre y el tipo de dato de cada columna de la tabla.

Para seleccionar datos de una tabla y devolver un resultado, que es una matriz de datos (filas y columnas).

```
SELECT nombre_de_columna(s)
FROM nombre_de_tabla
```

Para seleccionar los datos de todas las columnas de una tabla, se usa un asterisco en lugar de los nombres de las columnas:

```
SELECT *
FROM nombre_de_la_tabla
```

Para seleccionar datos de las tablas en donde sólo queramos recuperar registros en donde una columna tenga un valor en particular, usamos la cláusula WHERE.

```
SELECT nombre_de_columna(s)
FROM nombre_de_tabla
WHERE nombre_de_columna operador valor
```

Si se quisiera recuperar registros que cumplan con más de una condición, se utiliza AND u OR entre las condiciones del WHERE:

```
SELECT nombre_de_columna(s)
FROM nombre_de_tabla
WHERE nombre_de_columna operador valor
AND|OR nombre_de_columna operador valor
```

Ejemplos:

```
SELECT Nombre, Apellido
FROM alumnos
```

Nombre	Apellido
Laura	Gomez
Victoria	Sanchez

```
SELECT sueldo
FROM empleados
WHERE Puesto='Puesto1'
```

Sueldo
20000

Operadores que se pueden usarse con WHERE:

Operador	Descripción
=	Igual a
<>	Diferente de
>	Mayor que
<	Menor que
>=	Mayor que o igual a
>=	Menor que o igual a
BETWEEN	Entre cierto rango
LIKE	Busca cierto patrón de caracteres en los datos. Al patrón buscado se lo precede o sucede con **"%"** según se desee: **LIKE '%ma%':** Busca campos que contengan la sílaba "ma". **LIKE '%ma':** Busca campos que terminen con la sílaba "ma". **LIKE 'ma%':** Busca campos que empiecen con la sílaba "ma".

Consultas auxiliares

ORDER BY

Para lograr un ordenamiento específico de la respuesta podemos utilizar la palabra clave "ORDER BY"

```
ORDER BY nombre_de_columna ASC | DESC
```

Para dar dos ordenamientos sucesivos (por edad y por apellido, por ejemplo), vamos agregando las instrucciones una detrás de la otra en el orden en que queramos que se apliquen:

```
SELECT Nombre, Apellido, Edad
FROM Empleados
ORDER BY Edad ASC, Apellido ASC
```

LIMIT

Se utiliza para cuando sólo queremos una cantidad limitada de datos

```
SELECT nombre_de_columna(s)
FROM nombre_de_tabla
LIMIT numero_fila_inicial, numero_de_filas
```

número_fila_inicial es desde qué fila de las que nos devolvería queremos empezar

número_de_filas es el es la cantidad de filas que queremos recibir

```
SELECT ID_Empleado, Nombre, Apellido
FROM Empleados
LIMIT 2,1
```

ID_Empleado	Nombre	Apellido
2	Laura	Conti
3	Juana	Diaz

Agregar, Modificar o Eliminar registros

El lenguaje de manipulación de datos consiste en 3 consultas para manipular los datos.

INSERT

Inserta nuevos registros en una tabla

```
INSERT INTO nombre_de_tabla
VALUES (valor1, valor2,....)
```

```
INSERT INTO Empleados VALUES (03, 'Carlos', 'García');
```

ID_Empleado	Nombre	Apellido
1	Laura	Conti
2	Juana	Diaz
3	Carlos	García

O también se pueden especificar las columnas a las que se les agrega datos:

```
INSERT INTO nombre_de_tabla (columna1, columna2,...)
VALUES (valor1, valor2,....);
```

```
INSERT INTO Empleados (ID_Empleado, Nombre, Apellido) VALUES (03,
'Carlos', 'García');
```

DELETE

Sirve para eliminar registros de una tabla.

```
DELETE FROM nombre_de_tabla
WHERE nombre_de_columna operador valor
```

```
DELETE FROM Empleados
WHERE ID_Empleado=2
```

ID_Empleado	Nombre	Apellido
1	Laura	Conti
3	Carlos	García

También se pueden eliminar todas las filas de una tabla:

```
DELETE *
FROM nombre_de_tabla
```

UPDATE

Sirve para modificar datos de una tabla

```
UPDATE nombre_de_tabla
SET nombre_de_columna=valor
WHERE nombre_de_columna operador valor
```

```
UPDATE Empleados
SET nombre=Lara
WHERE ID_Empleado= 1;
```

ID_Empleado	Nombre	Apellido
1	Lara	Conti
3	Carlos	García

Instalación de MySQL y otras herramientas

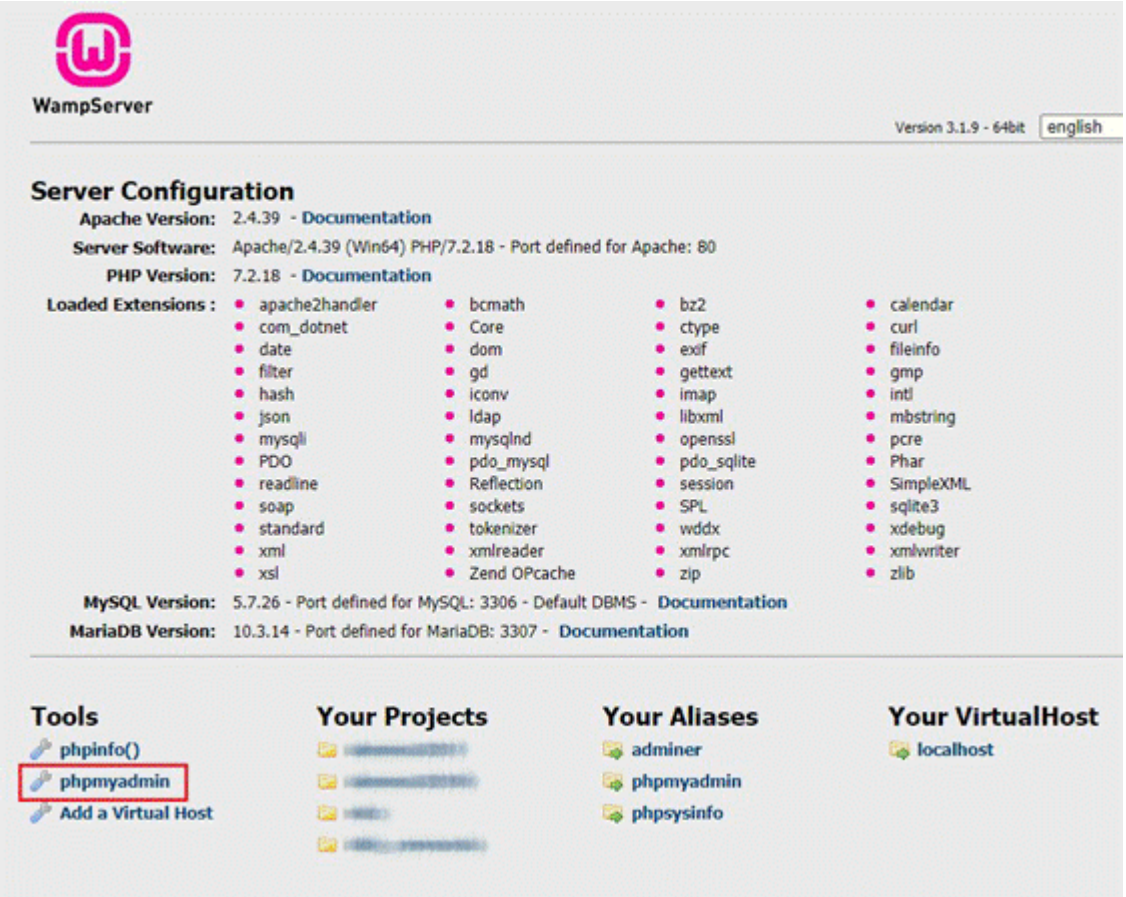
Para poder probar de forma local las consultas y operaciones que acabamos de ver necesitamos tener instalado MySQL y algún cliente o gestor que nos permita tanto administrar las bases de datos como las tablas. Para esto utilizaremos un paquete de aplicaciones llamado Wamp Server que incluye MySQL como motor de base de datos, a la vez que instala y configura de forma automática el servidor Apache y

el intérprete del lenguaje PHP. Tanto PHP como Apache nos serán de utilidad para ejecutar una herramienta llamada phpMyAdmin . Esta aplicación nos permite administrar bases de datos MySQL de forma muy sencilla.

WampServer puede ser descargado para windows de forma gratuita desde https://www.wampserver.com/en/. En caso de utilizar la plataforma de Mac, su equivalente es MAMP y puede ser descargado desde https://www.mamp.info/.

Cualquiera sea el que instalemos deberemos seguir las instrucciones de instalación provistas en ambos sitios.

Una vez instalado WampServer, nos aseguramos de que los servicios están ejecutándose y abriremos la dirección http://localhost . Una vez ahí, en la sección de Tools o herramientas buscaremos el link a phpMyAdmin.



En caso de que nos pida las credenciales de acceso al servidor de MySQL, y no las hayamos modificado, ingresamos root como usuario y dejaremos vacío el campo de contraseña (MAMP generalmente usa también root como contraseña).

Una vez que ingresamos al servidor podemos ejecutar las consultas en la pestaña SQL.

Bibliografía:

- SQL Commands Disponible desde la URL:

http://www.postgresql.org/docs/9.1/static/sql-commands.html

- Tutorial de SQL Disponible desde la URL:

http://www.unalmed.edu.co/~mstabare/Sql.pdf

- Wampserver Disponible desde la URL: http://www.wampserver.es/