Curso FullStack Python

Codo a Codo 4.0

MySQL

Parte 1

Bases de Datos

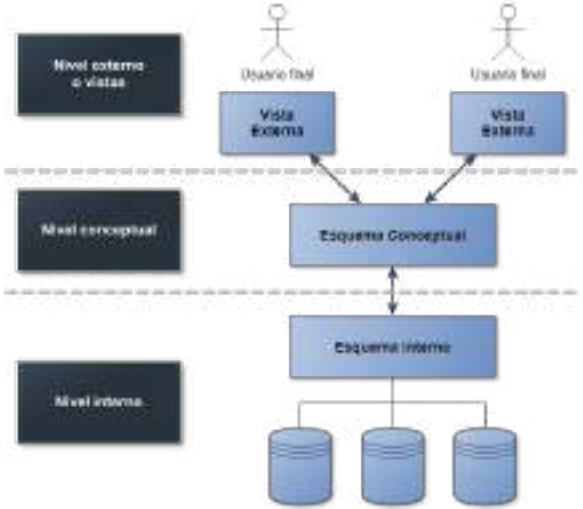
- Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Es una forma de almacenar información en forma más eficiente de lo que sería un archivo de texto.
- Las bases de datos se crean y mantienen a través de una colección de programas (**DBMS o motor de base de datos**). Este sistema de software de propósito general facilita la definición, construcción, manipulación y compartición de bases de datos entre usuarios y aplicaciones.
- Pueden ser datos cuyo contenido o temática difieren entre si, pero que poseen relaciones en común.

Por ejemplo: Una lista de alumnos no tiene nada que ver con una lista de libros. Pero si un alumno desea retirar los libros de una biblioteca, existe una relación.

*Las **bases de datos relacionales** buscan resolver una problemática relacionada con la **eficiencia en el almacenamiento de información** que podría tener, por ejemplo, un archivo de texto donde la lectura la haríamos en forma secuencial (línea por línea) o cargarla toda en memoria. Esa forma de almacenamiento es **ineficiente** ya que uno de los problemas es que no puedo acceder y guardar información al mismo tiempo. Además la consulta de información en un archivo de texto de 1 millón de registros me obligaría, en principio, a leer secuencialmente hasta llegar, por ejemplo, al último registro.*

Bases de Datos

Las bases de datos relacionales permiten **gestionar el acceso a los datos**, así como también su **almacenamiento**, **modificación**, **eliminación**, **consulta** y el **múltiple acceso** a ellos, que podrá ser desde distintas aplicaciones y usuarios. Inclusive podrán gestionar permisos para que una parte de los datos estén disponibles para ciertos usuarios y no para otros. Todo esto es resuelto por un motor de base de datos, generando una *independencia* entre la base de datos y la aplicación que la consulte..



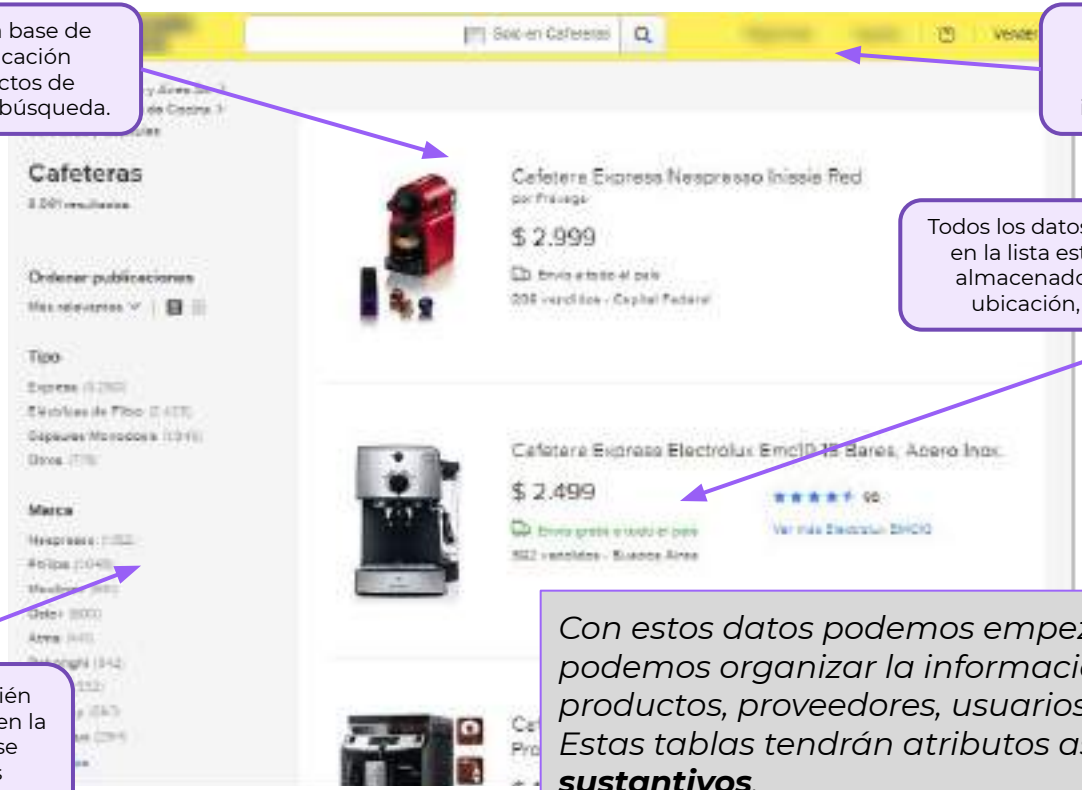
Supongamos que tenemos una empresa y le pido a un proveedor de software un sistema para registrar ventas, empleados, sueldos, etc. Voy a tener que contarle un poco de qué se trata el negocio, qué datos voy a guardar, etc.
Esos datos van a estar guardados en una base de datos con tablas para registrar esa información, pero una cosa son los ***datos*** *y otra la* ***visualización*** *de ellos, que voy a poder hacerlo desde un cliente de BD o incluso a través de un sistema desarrollado.*
El ***esquema conceptual*** *se refiere a las tablas y relaciones entre ellas.*

¿Por qué necesitamos una base de datos?
https://vimeo.com/225581128

Ejemplo de uso de base de datos

Los productos existen en la base de datos; al buscarlos, la aplicación consulta la lista de productos de acuerdo a la condición de la búsqueda.

Para ingresar, los usuarios deben haberse dado de alta primero. Al darse de alta, son ingresados a la base de datos.



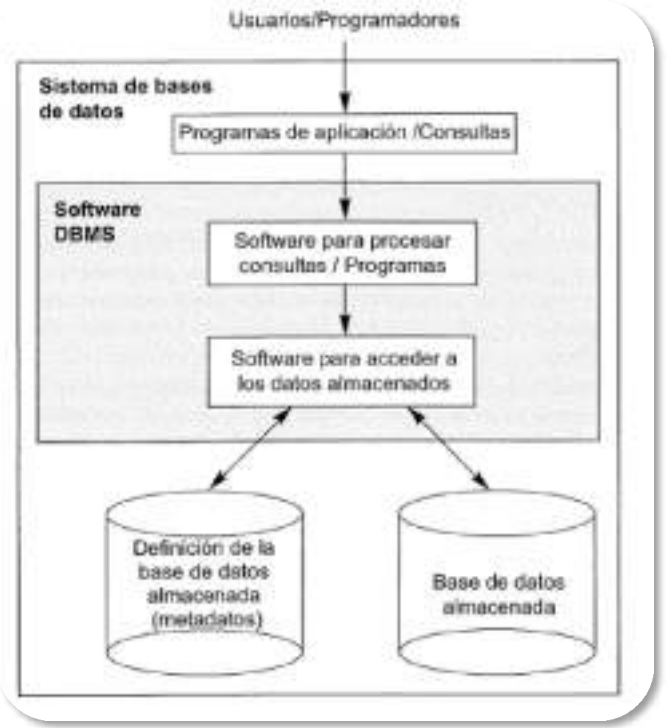
Todos los datos de los productos que aparecen en la lista están dentro de la base de datos almacenados para cada producto (precio, ubicación, vendedor, descripción, etc) .

Las categorías son también datos que se encuentran en la base de datos, por eso se puede contar cuántos productos hay en cada una

Con estos datos podemos empezar a pensar que podemos organizar la información en tablas: productos, proveedores, usuarios, categorías. Estas tablas tendrán atributos asociados que serán ***sustantivos***.

Ubicación de las Bases de Datos

- Las bases de datos se encuentran es el nivel más bajo dentro de un entorno de un sistema de bases de datos.
- Generalmente se los considera como parte "física", ya que, aunque sean un contenido lógico, se encuentran almacenadas o creadas en un dispositivo físico. Por ejemplo: un servidor.
- Para que un usuario pueda acceder a los datos en una Base de Datos, necesita de un software especial conocido como **SGBD (Sistema Gestor de Base de Datos)** o **DBMS (Data Base Managment System)**.



¿Cómo empezamos a pensar en una BD?

Tenemos que pensar qué datos son los que se van a almacenar, eso se empieza a descubrir de la entrevista con el cliente y el **relevamiento de datos**. El relato del cliente es clave ya que nos permite identificar qué información va a necesitar, por ejemplo el registro de los empleados, que es candidato a ser una **tabla** que va a tener atributos asociados, que van a ser sustantivos.
Para empezar a pensar la estructura debemos pensar en las **entidades (tablas)** de mi sistema. Siguiendo con el ejemplo de venta online tendría productos, marcas y distintos conceptos que debemos definir si son tablas o atributos. Nos damos cuenta a través de palabras claves que son candidatas a ser tablas. Si tienen mucha información relacionada a lo que nos cuenta el cliente probablemente sea una tabla. Por ejemplo: tabla Empleados con **atributos (columnas)**: nombre, apellido, DNI, fecha de nacimiento, etc..

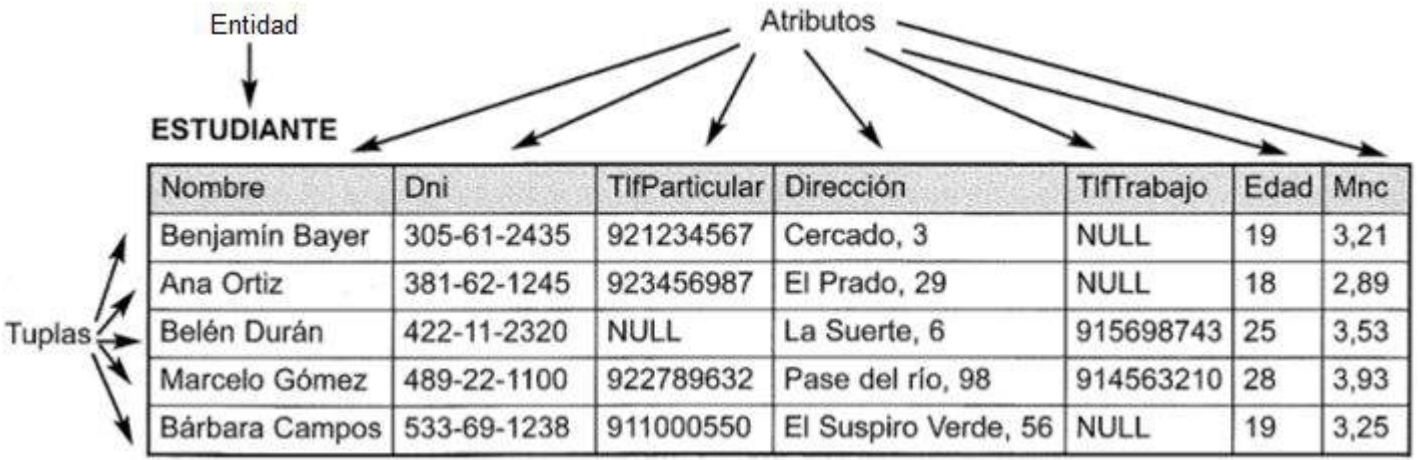
Entidad (tabla)

Empleados			
Campos/Atributos (columnas)			
DNI	**Apellido**	**Nombre**	**Fecha Nac.**
12.345.678	Gómez	Juan	25/09/1945
23.456.789	Fernández	Ana	12/07/1973

Filas/Registros

Modelo Relacional

- En la terminología formal del modelo relacional, una fila recibe el nombre de **tupla**, una cabecera de columna es un **atributo** y el nombre de la tabla se denomina **entidad**.
- El tipo de dato que describe los valores que pueden aparecer en cada columna está representado por un **dominio** de posibles valores.



Nombre	Dni	TlfParticular	Dirección	TlfTrabajo	Edad	Mnc
Benjamín Bayer	305-61-2435	921234567	Cercado, 3	NULL	19	3,21
Ana Ortiz	381-62-1245	923456987	El Prado, 29	NULL	18	2,89
Belén Durán	422-11-2320	NULL	La Suerte, 6	915698743	25	3,53
Marcelo Gómez	489-22-1100	922789632	Pase del río, 98	914563210	28	3,93
Bárbara Campos	533-69-1238	911000550	El Suspiro Verde, 56	NULL	19	3,25

SGBD más conocidos

- Dependiendo si la base de datos a tratar es RELACIONAL o NO RELACIONAL, existen diferentes Sistemas de Gestión de Base de Datos. Entre ellos se pueden mencionar:
  - **RELACIONALES**
    - MySQL
    - MariaDB
    - PostgreSQL
    - Ms. Access
    - entre otros...
  - **NO RELACIONALES**
    - MongoDB

Más información:
https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve

Base de Datos SQL vs NoSQL

- Como su propio nombre indica, las bases de datos no relacionales son las que, a diferencia de las relacionales, **no tienen un identificador que sirva de relación entre un conjunto de datos y otros**.
- La información se organiza normalmente mediante documentos y es muy útil cuando no tenemos un esquema exacto de lo que se va a almacenar.
- La indiscutible reina del reciente éxito de las bases de datos no relacionales es **MongoDB** seguida por Redis, Elasticsearch y Cassandra.

Base de Datos SQL vs NoSQL

FORMATO

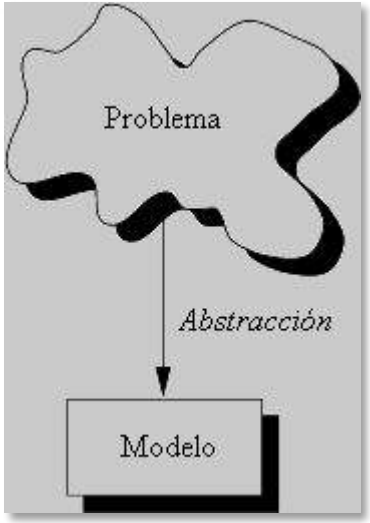
- La información puede organizarse en tablas o en documentos. Lo habitual es que las bases de datos basadas en **tablas** sean **bases de datos relacionales** y las basadas en **documentos** sean **no relacionales**, pero esto no tiene que ser siempre así. Solo es una cuestión de visualización.
- Esto explica por qué las bases de datos relacionales suelen servirse de **tablas** y las no relacionales de **documentos JSON**.
- Las bases de datos más competitivas suelen permitir operaciones de los dos tipos.

DISEÑO

- La diferencia entre el éxito y el fracaso recae en el **diseño del modelo**.
- De nada sirve elegir la base de datos más apropiada para nuestro sistema, si luego no se hace un **buen diseño**.

Abstracción y Modelado de datos

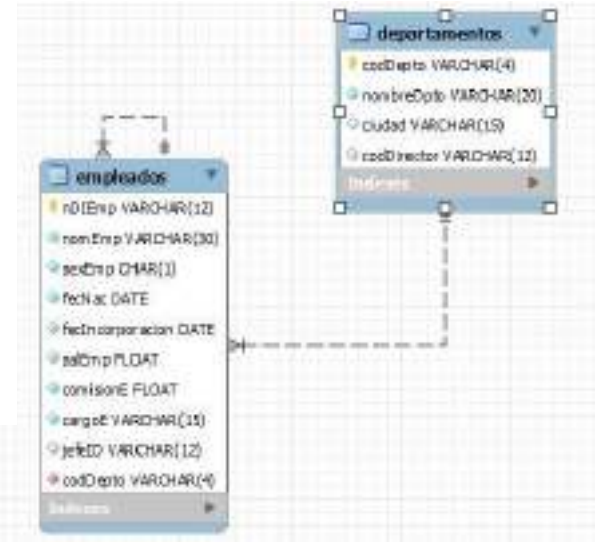
- La **abstracción de datos** es una técnica o metodología que permite diseñar estructuras de datos. La abstracción consiste en representar bajo ciertos lineamientos de formato las características esenciales de una estructura de datos.
- El **modelado de datos** permite describir:
  - **Las estructuras de datos de la base:** El tipo de los datos que hay en la base y la forma en que se relacionan.
  - **Las restricciones de integridad:** Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada.
  - **Operaciones de manipulación de los datos:** típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.



Modelo Entidad-Relación

- Es un método para diseñar Bases de Datos que posteriormente serán implementadas a través de un SGBD. Este modelo se representa a través de diagramas y está formado por varios elementos.

- El tipo de Diagrama utilizado para realizar el modelado Entidad-Relación es el **DER (Diagrama de Entidad-Relación)**, el cual pertenece al **Lenguaje de Modelado Unificado** (*UML, Unified Modeling Language*). Este diagrama representa entidades (tablas) y las relaciones lógicas entre ellas. *Por ejemplo una tabla de empleados y departamentos donde trabajan. Un departamento puede tener más de un empleado asociado.*



Más información:

- Diagrama Entidad-Relación
- Lenguaje de Modelado unificado

DER Componentes

ENTIDADES

- Las entidades representan cosas u objetos (ya sean reales o abstractos). Se representan en los diagramas como **rectángulos**. Se suelen colocar en *plural*.
- Por ejemplo:

Alumnos

Libros

Autos

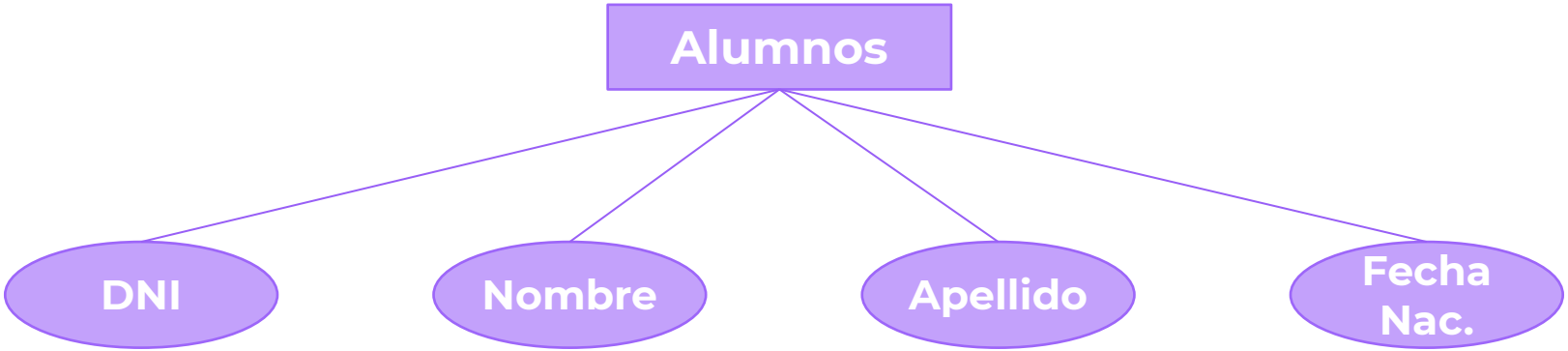
Empleados

Materias

DER Componentes

ATRIBUTOS

- Los atributos definen o identifican las características propias y por lo general únicas de una entidad.
- Cada entidad contiene distintos atributos, que dan información sobre ella misma.
- Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha, etc.), y se representan por medio de un **óvalo o elipse**.
- Por ejemplo:



DER Componentes

RELACIONES

- Se representan con **rombos** y tienen una característica conocida como "cardinalidad", la cual indica el sentido y la cantidad de "relaciones" existentes entre una entidad y otra.
- Los tipos de relaciones pueden ser:
  - **1 a N (uno a muchos)**: Por ejemplo: **una** persona puede tener **muchos** autos y viceversa, **muchos** autos pueden ser de **una** persona.

  - **1 a 1 (uno a uno):** Por ejemplo: a un alumno le pertenece únicamente una libreta y viceversa, una libreta pertenece únicamente a un alumno.

DER Componentes

RELACIONES

- Los tipos de relaciones pueden ser:
  - **N a N (muchos a muchos)**: Por ejemplo: **muchos** alumnos pueden tener **muchas** materias y viceversa, **muchas** materias pueden contener a **muchos** alumnos.

*En el rombo se coloca un **verbo** y la lectura puede ser "Muchos alumnos **tienen** muchas materias" o "Un alumno **tiene** muchas materias o una materia **tiene** muchos alumnos".*

DER - Ejemplo

- Supongamos que una empresa de venta de electrodomésticos tiene:
  - Clientes
  - Pedidos
  - Productos
- Se desea modelar a través de un DER, la forma en que se implementaría la Base de Datos.

1. Detectamos las **entidades**. Como sabemos, serán 3:

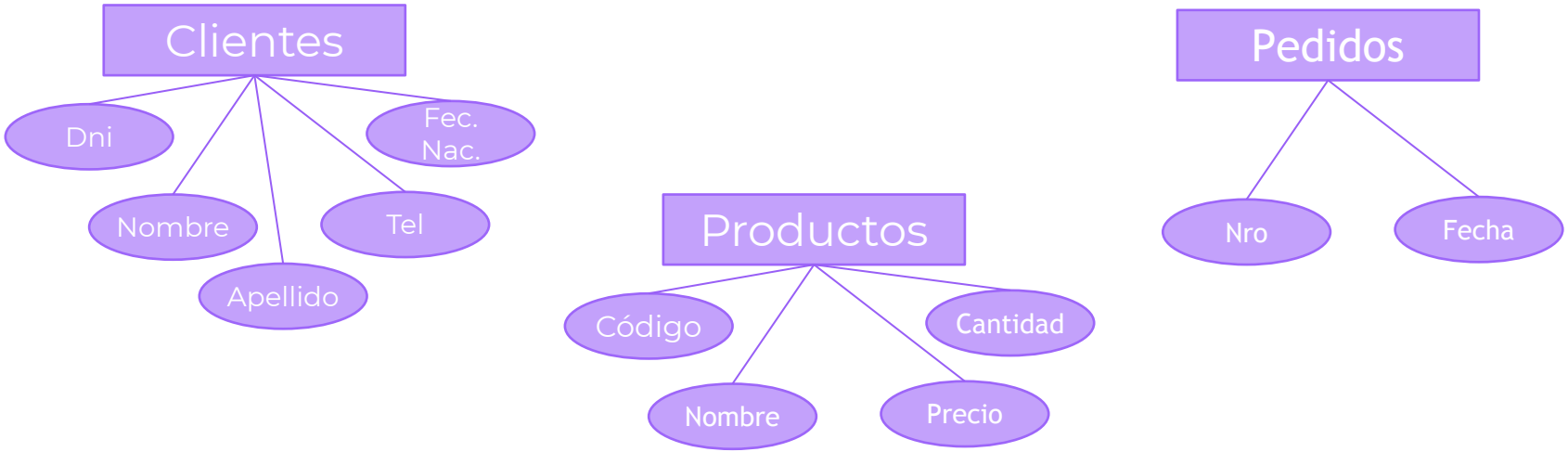
Clientes

Pedidos

Productos

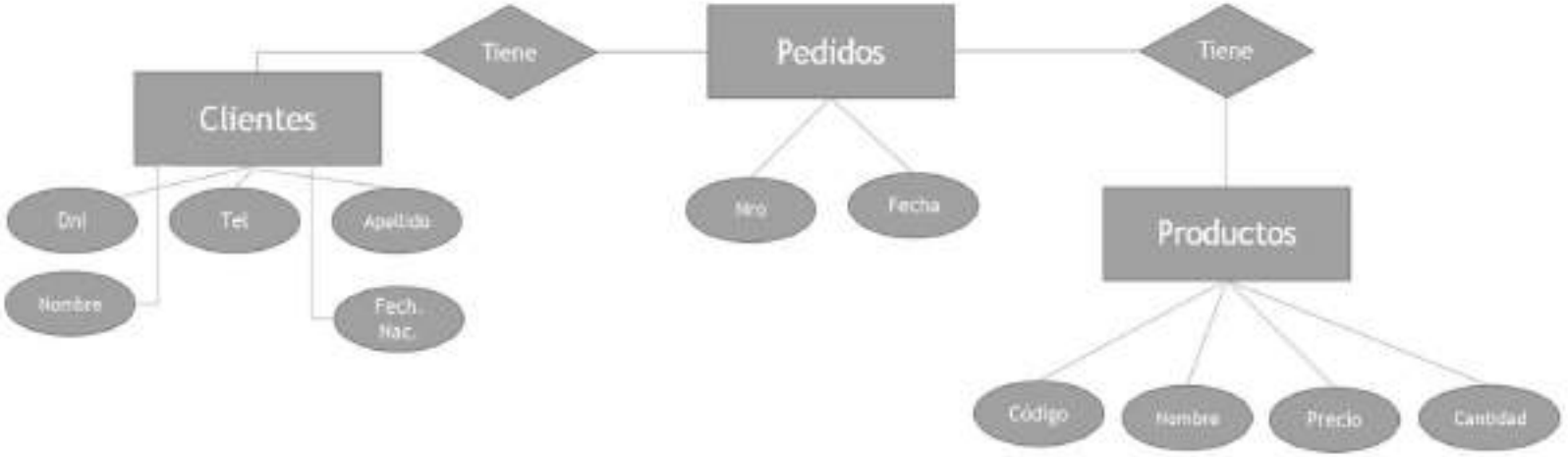
DER - Ejemplo

2. Ahora detectamos qué **atributos** tienen cada una de ellas:



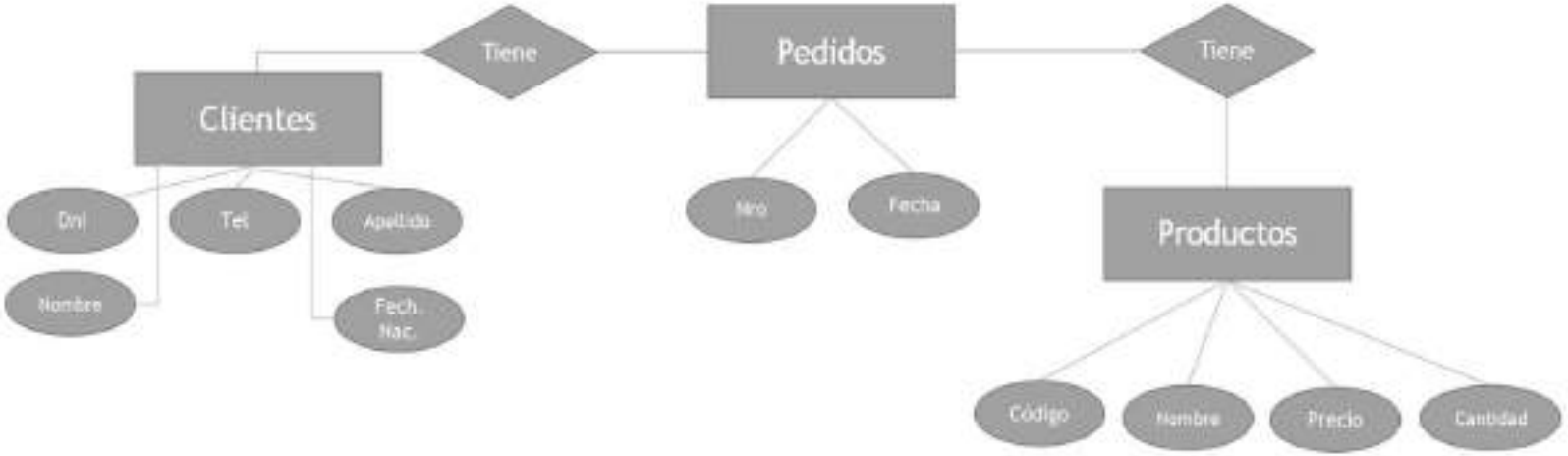
DER - Ejemplo

3. Una vez que conocemos cuáles son las **entidades** y sus **atributos**, podemos pasar a establecer las **relaciones** existentes entre sí:



DER - Ejemplo

- Como podemos ver, las relaciones que se encontraron fueron que, un cliente puede realizar varios pedidos (ya que en cada compra que realice, se efectuará un nuevo pedido) y que un pedido puede tener varios productos (ya que una misma compra/pedido pueden haber más de un artículo encargado).



DER - Ejemplo

- Una vez que tenemos el DER lo pasamos a forma de TABLA:

Clientes				
DNI	**Nombre**	**Apellido**	**Tel**	**Fec. Nac.**

Pedidos	
Nro	**Fecha**

Productos			
Código	**Nombre**	**Precio**	**Cantidad**

Tipos de Datos

Los **atributos de las entidades** deben cumplir o pueden ser únicamente de ciertos tipos de datos. Entre ellos, los más importantes/utilizados son:

NUMÉRICOS

- Se utiliza para representar valores/atributos de carácter numéricos tanto enteros, como decimales.

TEXTO (VARCHAR)

- Se utiliza para representar valores de texto, como ser cadenas de caracteres.

DATE (FECHA)

- Se utiliza para representar fechas, horas, minutos, segundos, etc.

BOOLEAN (LÓGICO)

- Se utiliza para representar valores verdaderos o falsos (true or false).

Ejemplo

- Tipos de datos:

Clientes				
DNI	**Nombre**	**Apellido**	**Tel**	**Fec. Nac.**
INT	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	DATE

Productos			
Código	**Nombre**	**Precio**	**Cantidad**
INT	VARCHAR(20)	DOUBLE	INT

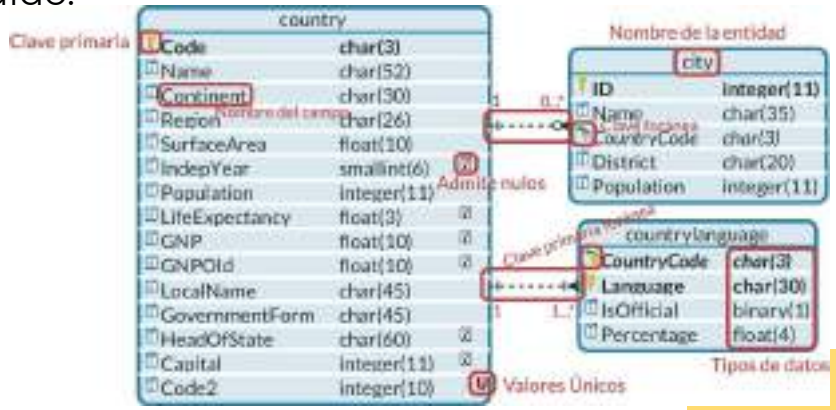
Pedidos	
Nro	**Fecha**
INT	DATE

Primary key y Foreign Key

- Las **claves primarias (Primary Keys)** son valores que identifican de manera única a cada fila o registro de una tabla, esto quiere decir que **no se puede repetir**. Por ejemplo: un DNI, un código de producto, etc.
- Una **clave foránea (Foreign Key)** es un campo de una tabla “X” que sirve para enlazar o relacionar entre sí con otra tabla “Y” en la cual el campo de esta tabla es una llave primaria (Primary Key). Para que sea una clave foránea un campo, esta tiene que ser una llave primaria en otra tabla.

Por ejemplo: en la tabla **“clientes”** el **DNI** es una *primary key*, pero en una tabla **“pedidos”** representa a quién pertenece ese determinado pedido.

*En este ejemplo, **CountryCode** en las tablas **City** y **CountryLanguage** son claves foráneas de la clave primaria **Code** de la tabla **Country** y, a su vez, **CountryCode** es clave primaria en la tabla **CountryLanguage** .*



Ejemplo

Clientes				
DNI	**Nombre**	**Apellido**	**Tel**	**Fec. Nac.**
INT	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	DATE

Productos			
Código	**Nombre**	**Precio**	**Cantidad**
INT	VARCHAR(20)	DOUBLE	INT

Pedidos	
Nro	**Fecha**
INT	DATE

Instalación del motor de base de datos + herramientas

Para trabajar con bases de datos debemos instalar:

- El **motor de base de datos**, para trabajar en forma local (MySQL Server)
- El **gestor de base de datos, servidor** (XAMPP)
- El **cliente**, la aplicación con la cual nos vamos a conectar con nuestro motor de Base de Datos:
  - MySQL Workbench
  - PHPMYADMIN
  - VISUAL STUDIO CODE (extensiones)

Importante: Para la instalación se recomienda ver los tutoriales que aparecen al final de esta presentación y utilizar los archivos que están en el Aula Virtual y la carpeta de Drive compartida.

Instalación MySQL Server

1) Descargar el instalador de https://dev.mysql.com/downloads/installer/
2) Ejecutar el instalador y seleccionar Server Only Execute.
3) Darle a next hasta llegar a la pantalla Authentication Method: Seleccionar Use Legacy Authentication Method.
4) En la siguiente pantalla setear contraseña para el usuario root en MySQL Root Password.
5) Darle next y al llegar a Apply Configuration apretar Execute.

Instalador: ver archivo **mysql-installer-web-community-8.0.22.0.msi**

Herramientas para manejo de Base de Datos

MYSQL WORKBENCH

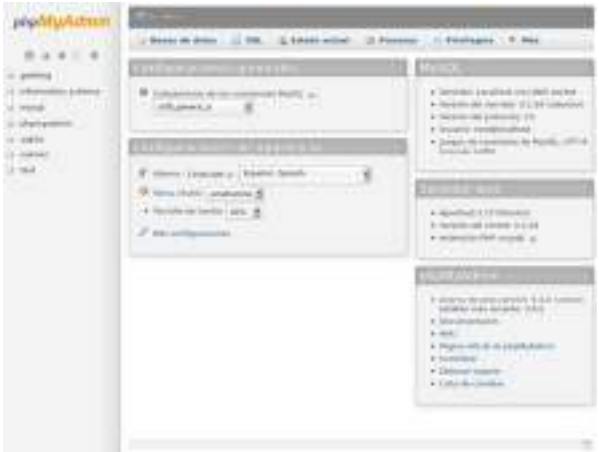
- **MySQL Workbench** es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

Se puede descargar desde: https://dev.mysql.com/downloads/workbench/

Herramientas para manejo de Base de Datos

PHPMYADMIN

- **phpMyAdmin** es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet.
- Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos.

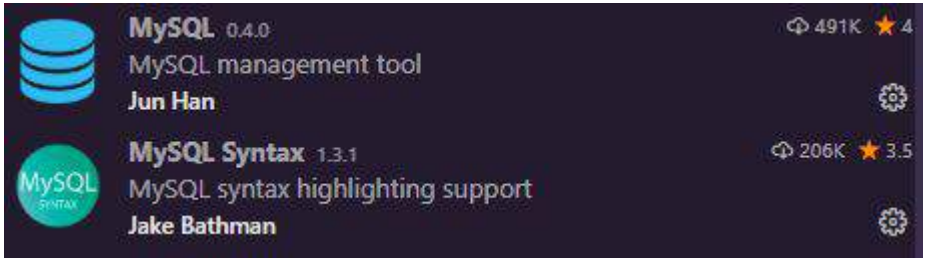


Se puede encontrar en: https://www.phpmyadmin.net/

Herramientas para manejo de Base de Datos

VISUAL STUDIO CODE

1) Descargar las siguientes extensiones en VSCode:



2) Cerrar y volver a abrir Visual Studio Code.

Herramientas para manejo de Base de Datos

VISUAL STUDIO CODE

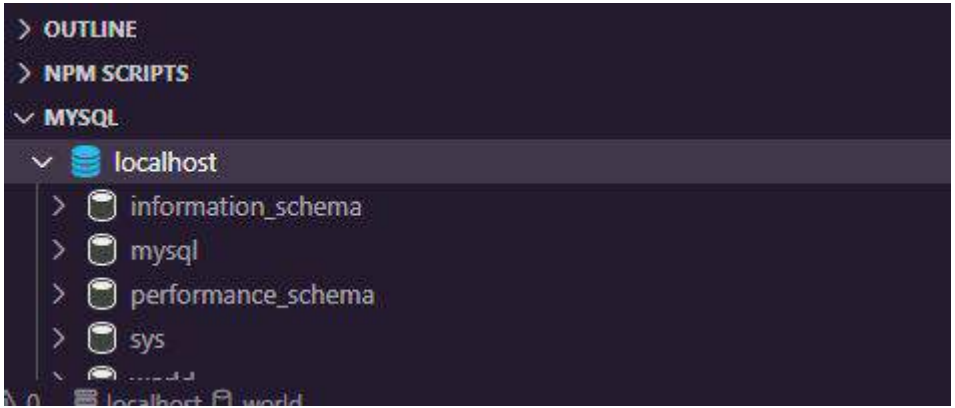
3) Apretar el símbolo + en el apartado MySQL. Al ser la primera vez que se configura no aparecerá ninguna base de datos:

Herramientas para manejo de Base de Datos

VISUAL STUDIO CODE

4) Rellenar usuario y contraseña, a los demás datos (puerto y SSL) apretar ENTER sin modificar nada.

5) Deberá aparecer localhost.

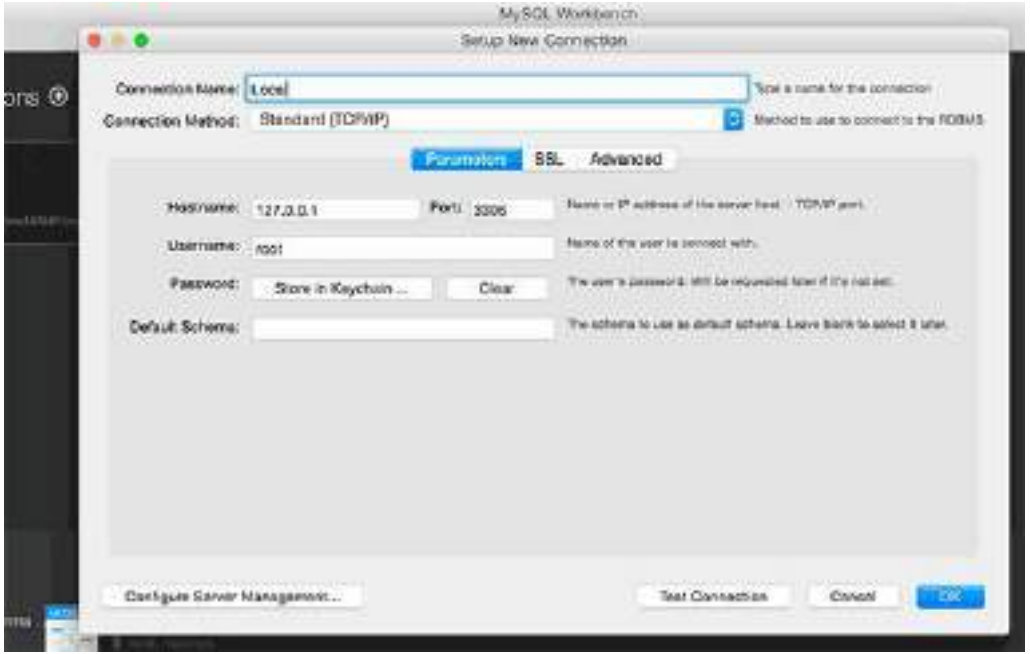


Conectarse al servidor MySQL

- Para que un programa cliente (MySQL Workbench, phpMyAdmin, VSCode, etc.) se conecte al servidor MySQL, debés utilizar los parámetros de conexión adecuados, como el nombre del host donde se ejecuta el servidor y el nombre de usuario y contraseña de tu cuenta MySQL.
- Cada parámetro de conexión tiene un valor predeterminado, pero puede anular los valores predeterminados según sea necesario utilizando las opciones del programa especificadas en la línea de comandos o en un archivo de opciones.

Conectarse al servidor MySQL

- **Ejemplo en MySQL Workbench:** En MySQL Connections deben establecer una nueva conexión con el signo + y poner los datos de la nueva conexión.



Crear una base de datos de prueba

WORLD.SQL

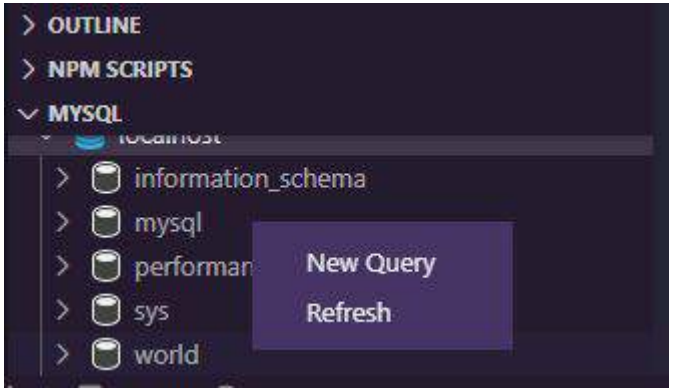
- Los pasos se detallan para **VSCode** pero para MySQL Workbench y phpMyAdmin resultan similares.
  1) Descargar **world.sql** del Aula Virtual y abrir con Visual Studio Code.
  2) Apretar botón derecho -> Run SQL Query.

Crear una base de datos de prueba

WORLD.SQL

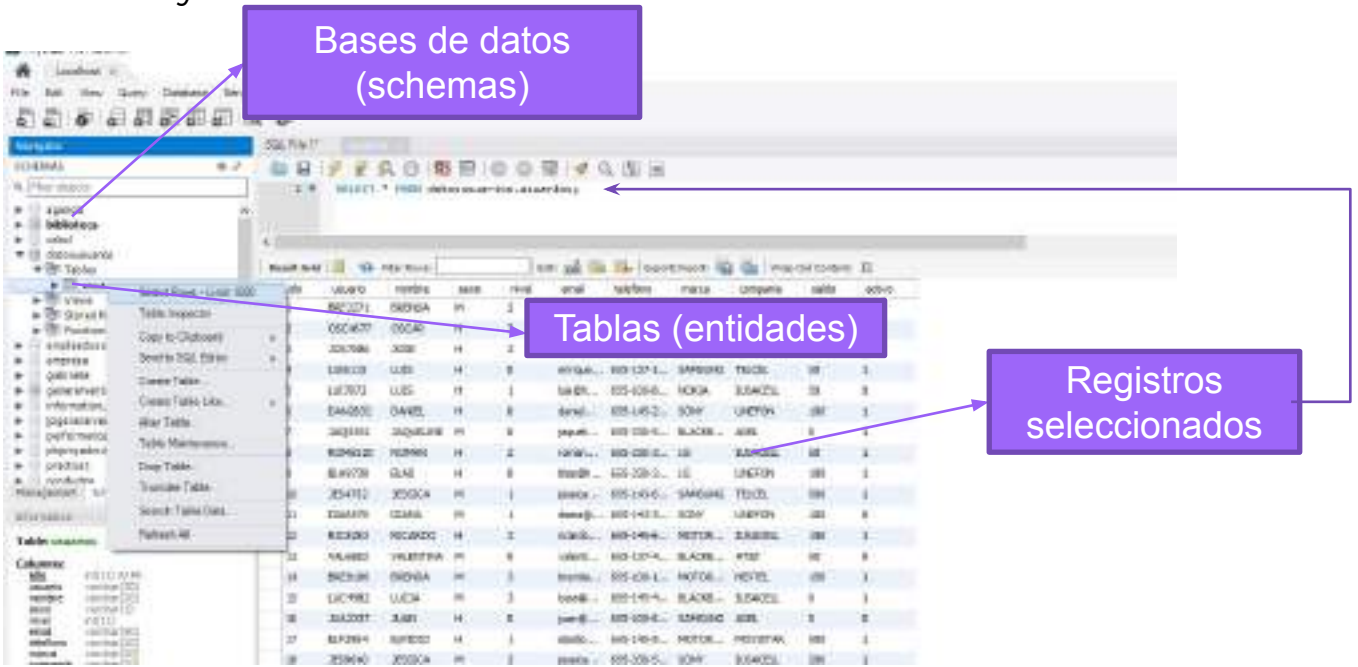
- Los pasos se detallan para **VSCode** pero para MySQL Workbench y phpMyAdmin resultan similares.

3) Apretar botón derecho -> Run SQL Query.



MySQL Workbench: ver una BD y acceder a sus tablas

Una vez que nos conectamos al LocalHost, previa conexión con XAMPP, podremos acceder a ver las bases de datos y sus tablas:



Sentencias DDL: CREATE, ALTER y DROP

- El **lenguaje de definición de datos** (***Data*** *Definition Language*, o DDL), es el que se encarga de la modificación de la **estructura** de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos.
- El **lenguaje DDL** o de **definición de datos**, contiene sentencias que permiten crear, modificar o eliminar objetos en el esquema interno de la base de datos en base al esquema conceptual.
- Para ello se utilizan tres sentencias:
  - CREATE
  - ALTER
  - DROP

Más información:
https://sites.google.com/site/sqlismysin/home/lenguaje-de-definicion-de-datos-ddl

Creando una Base de Datos

- CREATE DATABASE crea una base de datos con el nombre de indicado en esa orden.
- Para utilizar esta declaración, se necesita el privilegio o permiso del sistema de la base de datos. CREATE SCHEMA es sinónimo de CREATE DATABASE.
- Se produce un error si la base de datos ya existe y no le especificaste **IF NOT EXISTS**.
- La misma puede ser creada de la siguiente manera:

  CREATE DATABASE databasename;

Ver las tablas de una Base de Datos

- Para ver las tablas existentes en una base de datos tipeamos:

  SHOW TABLES;

Crear una Tabla

- Creamos una tabla llamada “alumnos” tipeando:

```
CREATE TABLE alumnos (
  dni int(11),
  nombre varchar(30),
  apellido varchar(30),
  fecha_nac date
);
```

- Si intentamos crear una tabla con un nombre ya existente (existe otra tabla con ese nombre), mostrará un mensaje de error indicando que la acción no se realizó porque ya existe una tabla con el mismo nombre.

Ver la estructura de una Tabla

- Para ver la estructura de una tabla usamos el comando "describe" junto al nombre de la tabla:

  DESCRIBE alumnos;

- Aparecerá lo siguiente:

	Field	Type	Null
	dni	int(11)	YES
	nombre	varchar(30)	YES
	apellido	varchar(30)	YES
	fecha_nac	date	YES

Eliminar una Tabla

- Para eliminar una tabla usamos "drop table". Tipeamos:

  DROP TABLE alumnos;

- Si tipeamos nuevamente:

  DROP TABLE alumnos;

- Aparece un mensaje de error, indicando que no existe, ya que intentamos borrar una tabla inexistente. Para evitar este mensaje podemos tipear:

  DROP TABLE IF EXISTS alumnos;

Modificar una Tabla

- ¿Qué sucede si quisiéramos cambiar algo en la tabla que creamos?
- ¿Cómo haríamos si quisiéramos agregar o eliminar una columna?
- Para agregar, eliminar o modificar una columna utilizamos la sentencia **ALTER**.
  - Para agregar una columna:

    ALTER TABLE nombre_de_tabla

    ADD nombre_de_columna tipo de dato;
  - Para eliminar una columna:

    ALTER TABLE nombre_de_tabla

    DROP COLUMN nombre_de_columna;

Creando nuestra primer BD

Crearemos nuestra primer BD llamada **empleados_departamentos**. Utilizaremos el archivo **bd_empleados_departamentos.sql** para ejecutar la sentencia SQL que la crea. Para ello seguiremos los siguientes pasos:

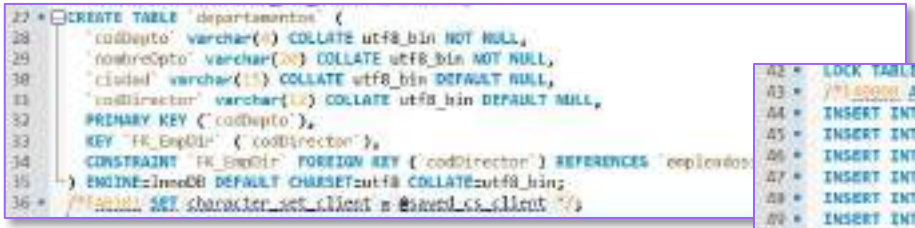
1) Abrir el archivo que contiene la sentencia SQL.
2) Pegar todo el texto dentro de una nueva consulta 

3) Ejecutar desde el ícono del ray 

Quedará creada la Base de Datos con dos tablas: departamentos y empleados

Creando nuestra primer BD

Para **crear una tabla** utilizamos **CREATE TABLE** e indicamos cuáles son las columnas (atributos/campos) que conformarán nuestra tabla:



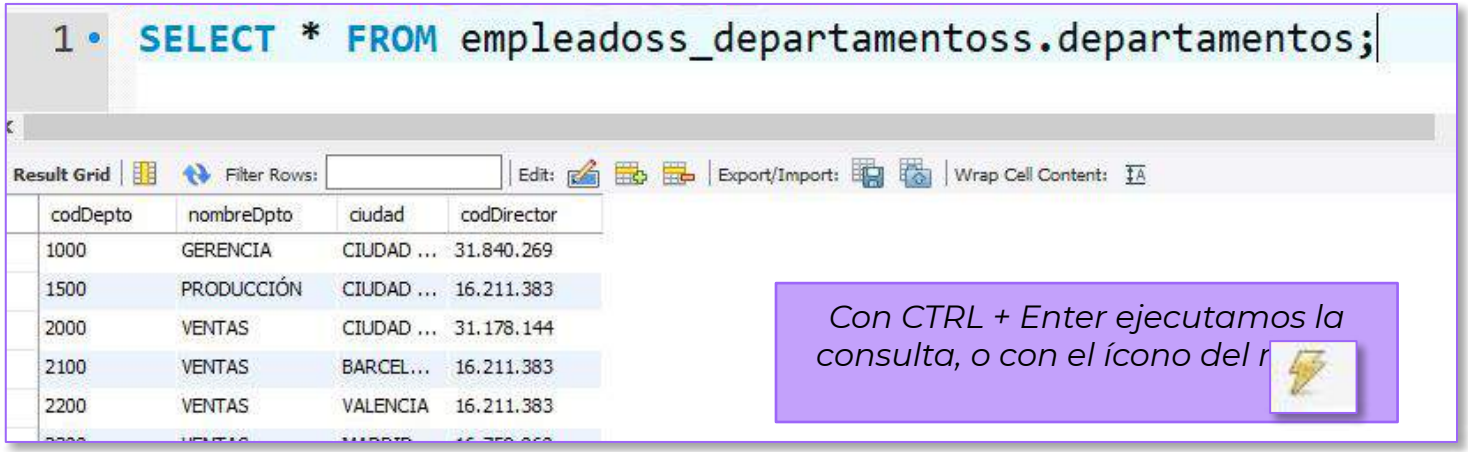
Para agregar registros utilizamos **INSERT INTO nombredelatabla VALUES** y estos datos van separados por comas en el mismo orden en que fueron incorporados los campos.

codDepto	nombreDpto	ciudad	codDirector
1000	GERENCIA	CIUDAD ...	31.840.269
1500	PRODUCCIÓN	CIUDAD ...	16.211.383
2000	VENTAS	CIUDAD ...	31.178.144
2100	VENTAS	BARCEL...	16.211.383
2200	VENTAS	VALENCIA	16.211.383
2300	VENTAS	MADRID	16.799.060
3000	INVESTIGACIÓN	CIUDAD ...	16.799.060
3500	MERCADEO	CIUDAD ...	22.222.222

Registros insertados!

¿Cómo se ven los datos de nuestras tablas?

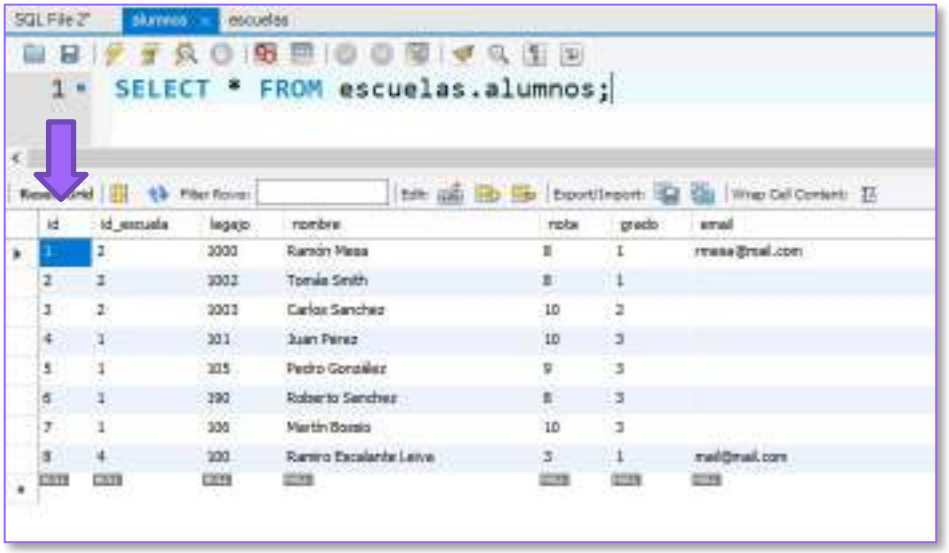
Haciendo clic con el botón derecho en nuestra tabla y seleccionando **Select Rows – Limit 1000** veremos los resultados de nuestra primer consulta SQL:



¿Cómo traducimos esto? *Estamos diciendo que seleccione todos los registros de la tabla departamentos del schema (base de datos) empleados_departamentos.*

Las claves principales

Utilizaremos la BD **escuelas**. Las tablas van a tener una **clave principal**, que es un identificador único para cada registro de la tabla. Para definirla tenemos que analizar las ***claves candidatas,*** aquellas que podrían ser claves principales, un valor propio de ese registro que identifique de forma única esa instancia del dato.
Cada registro debería tener un identificador único, para evitar duplicados:



***IMPORTANTE**: Los strings no suelen ser buenos candidatos para clave primaria.*

Agregar un registro

Para agregar un registro podemos escribir directamente en la tabla y al escribir el último dato debemos hacer clic en aplicar (se puede hacer luego de agregar varios registros). Se nos generará la siguiente instrucción SQL:

```
INSERT INTO `escuelas`.`alumnos` (`id_escuela`, `legajo`, `nombre`, `nota`, `grado`, `email`)
VALUES ('2', '200', 'Juan Pablo Nardone', '8', '2', 'mail@gmail.com');
```

Eliminar un registro

Para eliminar un registro podemos hacerlo desde el botón derecho a la izquierda del registro y luego hacer clic en **aplicar**. Debemos confirmar la siguiente instrucción SQL:

```
DELETE FROM `escuelas`.`alumnos` WHERE `id`='8';
```

En este caso estamos eliminando el registro del alumno cuyo id es el número 8.

Material complementario (sitios y videos)

- **Editor SQL On-Line “SQL Fiddle”:** permite probar scripts sql en los motores más populares (MySQL, Oracle, PostgreSQL, SQLite, SQL Server), este sistema en la nube impulsa al aprendizaje sin complicaciones de instalación o alguna conexión que se tenga que configurar. http://sqlfiddle.com/
- **Instalar XAMPP y phpMyAdmin:** ver video **Instalar XAMPP y phpMyAdmin** en el Aula Virtual.
- **Instalar XAMPP y MySQL Workbench:** https://youtu.be/wFZtb5UYRjM
- **Instalar MySQL y MySQL Workbench:** ver video **Instalar MySQL y MySQL Workbench** en el Aula Virtual.
- **Primer encuentro con una Base de Datos MySQL (MySQL Workbench):** ver video **Primer encuentro con una Base de Datos MySQL (MySQL Workbench)** en el Aula Virtual.
- **Primer encuentro con una Base de Datos MySQL (XAMPP y phpMyAdmin):** ver video **Primer encuentro con una Base de Datos MySQL (XAMPP y phpMyAdmin)** en el Aula Virtual.
- **Píldoras informáticas:** https://www.pildorasinformaticas.es/course/curso-sql/curriculum/