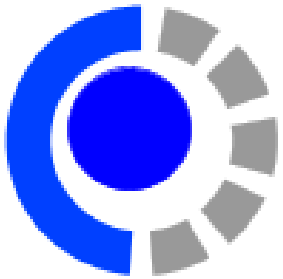
CONCEPTOS DE BASES DE DATOS

SQL - Structured Query Language

UNIDAD IV -VI

Departamento Ingeniería de Sistemas
Facultad de Informática
Universidad Nacional del Comahue



Recordemos…

Un **Sistema Gestor de Base de Datos(SGBD)** consiste de una *colección organizada de datos interrelacionados* y un conjunto de programas para almacenar, acceder y modificar a dichos elementos de datos

Recordemos…

- Los SGBD proporciona dos tipos lenguajes:
  - **Lenguaje de definición de datos (LDD):** se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema físico. Lo utilizan los diseñadores y los administradores de la BD.
  - **Lenguaje de manipulación de datos (LMD):** se utilizan para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones.

SQL - Structured Query Language

SQL - Structured Query Language

- Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.
- Proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
- Incluye lenguajes de consultas que permite manipular los datos.
- Incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.

SQL - Structured Query Language

- Incluye comandos para definir las vistas.
- Tiene comandos para especificar el comienzo y el final de una transacción.
- Incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

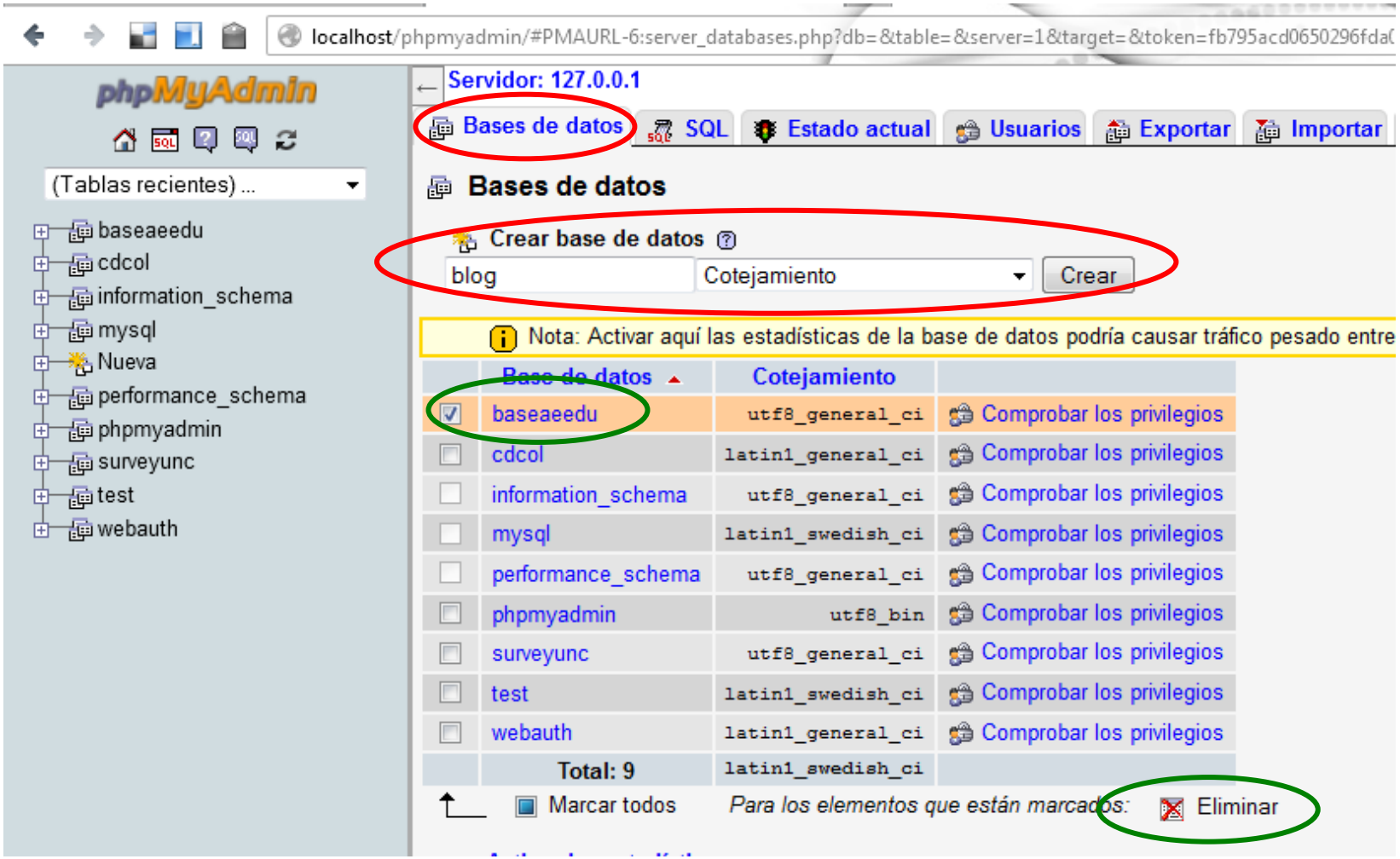
Herramienta - PhpMyAdmin

- Para administrar la base de datos, vamos a utilizar la interfaz Web phpMyAdmin.
- Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web.
- Con ella se puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL.
- Además se puede administrar claves en campos, administrar privilegios, exportar datos en varios formatos.
- Se encuentra disponible bajo la licencia GPL.

SQL - DDL

- El comando **CREATE DATABASE** crea una base de datos con el nombre dado. Para usar CREATE DATABASE, necesita el permiso CREATE en la base de datos.
- El comando **DROP DATABASE** permite borrar todas las tablas en la base de datos y borrar la base de datos. Sea *muy* cuidadoso con este comando! Para usar DROP DATABASE, necesita el permiso DROP en la base de datos.
- Ejemplo:
  - CREATE DATABASE blog;
  - DROP DATABASE blog;

SQL - DDL


localhost/phpmyadmin/#PMAURL-6:server_databases.php?db=&table=&server=1&target=&token=fb795acd0650296fda
phpMyAdmin
(Tablas recientes) ...
baseaeedu
cdcol
information_schema
mysql
Nueva
performance_schema
phpmyadmin
surveyunc
test
webauth
Servidor: 127.0.0.1
Bases de datos
SQL
Estado actual
Usuarios
Exportar
Importar
Bases de datos
Crear base de datos
blog
Cotejamiento
Crear
Nota: Activar aquí las estadísticas de la base de datos podría causar tráfico pesado entre
Base de datos
Cotejamiento
baseaeedu utf8_general_ci Comprobar los privilegios
cdcol latin1_general_ci Comprobar los privilegios
information_schema utf8_general_ci Comprobar los privilegios
mysql latin1_swedish_ci Comprobar los privilegios
performance_schema utf8_general_ci Comprobar los privilegios
phpmyadmin utf8_bin Comprobar los privilegios
surveyunc utf8_general_ci Comprobar los privilegios
test latin1_swedish_ci Comprobar los privilegios
webauth latin1_general_ci Comprobar los privilegios
Total: 9 latin1_swedish_ci
Marcar todos
Para los elementos que están marcados:
Eliminar

SQL - DDL

- Las bases de datos están compuestas por tablas.
- Las **tablas** son la estructura básica donde se almacena la información en la base de datos.
- Cada fila de una tabla se denomina **registro** y esta compuesto por campos o atributos.
- Cada **campo** tiene un nombre único para la tabla de la cual forma parte, además es de un tipo (naturaleza) determinado.

SQL – DDL: CREATE TABLE

- Para crear una tabla en una base de datos usaremos la sentencia CREATE TABLE

```
CREATE TABLE entradas (
  idEntrada int(11) NOT NULL AUTO_INCREMENT,
  idCategoria int(11) NOT NULL,
  Titulo varchar(200) NOT NULL,
  Resumen varchar(1000) NOT NULL,
  Contenido text NOT NULL,
  PRIMARY KEY (idEntrada)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
   AUTO_INCREMENT=1;
```

SQL – DDL: CLAVE PRIMARIA

- Se puede definir una clave primaria sobre una columna, usando la palabra clave *KEY* o ***PRIMARY KEY***, o especificarla a continuación de los atributos que componen la tablas
- Tenemos la posibilidad de crear una columna autoincrementada, aunque esta columna sólo puede ser de tipo entero. Al momento de insertar un nuevo registro, toma el valor más alto de esa columna y suma una unidad.

Opción ENGINE de Mysql

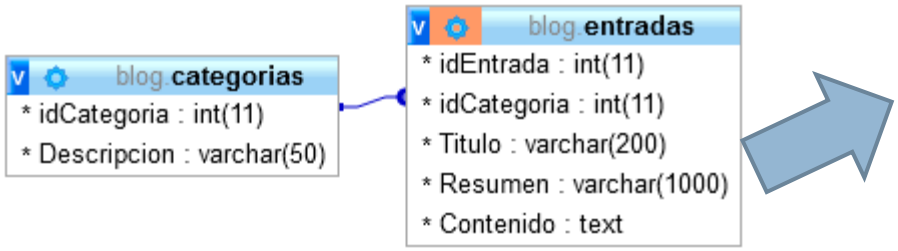
- Hay varios motores (ENGINE) de almacenamiento disponibles. Algunos de ellos serán de uso obligatorio si queremos tener ciertas opciones disponibles. Por ejemplo, el soporte para claves foráneas sólo está disponible para el motor InnoDB.
- Algunos motores conocidos son:
  - **InnoDB**: tablas de transacción segura con bloqueo de fila y claves foráneas.
  - **MyISAM**: trata tablas no transaccionales.
  - **MEMORY**: proporciona tablas en memoria.
  - **ARCHIVE:** se usa para guardar grandes cantidades de datos sin índices con una huella relativamente pequeña.
- Nosotros usaremos tablas InnoDB.

SQL – Tipos de Datos

- Los campos de las tablas MySQL nos dan la posibilidad de elegir entre tres grandes tipos de contenidos:
  - Datos **numéricos**,
  - Datos para guardar cadenas de caracteres **(alfanuméricos)**,
  - Datos para almacenar **fechas y horas**.
- Material sobre tipo de datos: Apunte_Unidad_4_mySQL_tipos_de_datos.pdf

SQL - DDL

Clave Foránea



Donde *idcategoria* en la tabla **entradas** es una *clave foránea* que hace referencia a la tabla **categorias**

```
CREATE TABLE entradas (
  idEntrada int(11) NOT NULL AUTO_INCREMENT,
  idCategoria int(11) NOT NULL,
  Titulo varchar(200) NOT NULL,
  Resumen varchar(1000) NOT NULL,
  Contenido text NOT NULL,
  PRIMARY KEY (idEntrada),
  FOREIGN KEY (`idCategoria`) REFERENCES `categorias`
(`idCategoria`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1;
```

SQL – DML

- Consultas de Selección:

  SELECT lista de campos

  FROM Tabla

  WHERE condición

- Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros.
- La lista de campos puede ser reemplaza por el carácter comodín ***** indicando así que se desea recuperar TODOS los CAMPOS de la tabla.
- El **WHERE** puede no aparecer en la consulta, indicando así que se desea recuperar TODOS los REGISTROS de la tabla.

SQL - DML

- **Preguntas para la Construcción de la consulta:**
  - Para construir una consulta SQL debemos hacernos como mínimo tres preguntas:
    - Primero hemos de preguntarnos: **¿qué datos nos están pidiendo?**
    - Lo siguiente que nos preguntamos es: **¿dónde están esos datos?**
    - Y por último: **¿qué requisitos deben cumplir los registros?**

SQL - DML

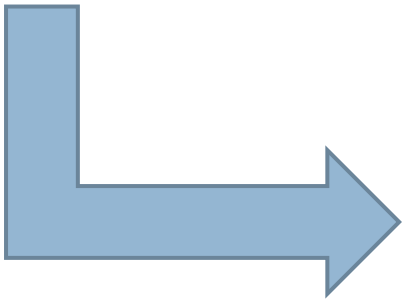
- Ejemplo: Dada la tabla Clientes:

idCliente	NroDNI	FechaAlta	Nombre	FechaUltimaCompra	idProvincia	Localidad	Domicilio	TEFijo	FAX	TEMovil	Deuda	VIP
1	43535633	2010-03-06	Martin Jose Barrueto Bastidas	2012-04-14	15	Neuquén	PRINGLES 1500	2257359165	2847183555	1581961819	59077.5	0
2	53243455	2010-08-21	Praxedes Leal	2012-06-08	5	Comodoro Rivadavia	El tostado 286	263520686	1415280617	4333367791	41376.4	1
3	32423444	2003-11-02	Maria Antonia Montesinos	2012-04-06	23	Ushuaia	Esquiu 269	6677326654	1909758758	442087835	56281.7	0
4	43432432	2008-04-29	Natalia Del Carmen Godoy	2012-09-20	17	Salta	DOCTOR RAMON 4822	2350781331	1201119981	868396925	10667.8	0
5	54324222	2011-04-08	Norma Elvia Muñoz Repol	2012-07-12	9	Formosa	Ministro gonzalez 54	6077040926	3788285221	1534259982	96464.1	1

- Necesito saber ¿DNI y Nombre de clientes, con una deuda superior a 60000 pesos?
  - **¿qué datos nos están pidiendo?** ➡ DNI y Nombre
  - **¿dónde están esos datos?** ➡ Clientes
  - **¿qué requisitos deben cumplir los registros?** ➡ Deuda mayor a 60000

SQL - DML


```
SELECT NroDNI, NOMBRE
FROM Clientes
WHERE Deuda > 60000
```



Resultado:

NroDNI	Nombre
54324222	Norma Elvia Muñoz Repol

SQL - DML

- ***Operadores de comparación***:
  - Es posible incluir expresiones en la cláusula WHERE comparando valores usando los seis operadores de Comparación: **= , <>, < , >, <= , >=** y **LIKE**.
  - Los valores que pueden ser comparados incluyen constantes y atributos de la tabla/s mencionada/s luego de la cláusula FROM.
  - La clausula **LIKE** se usa para el reconocimiento de patrones. Se usa junto con el **%** como comodin.

SQL - DML

- ***Operadores lógicos:***
  - Es posible incluir también operadores lógicos en las expresiones en la cláusula WHERE.
  - Son los siguientes:
    - **AND** significa "y",
    - **OR** significa "o",
    - **NOT** significa "no", invierte el resultado

SQL – DML: Ejemplos

- Necesito saber DNI y Nombre de clientes, con una deuda superior a 60000 pesos y que son de la localidad de Neuquén

```
SELECT NroDNI, NOMBRE
 FROM Clientes
WHERE Deuda > 60000 AND Localidad = 'Neuquén'
```

- Necesito saber DNI de los clientes cuyo Nombre comience con "Ma"

```
SELECT NroDNI FROM Clientes
WHERE NOMBRE LIKE 'Ma%'
```

SQL – DML: INSERT INTO

- Para agregar registros a una tabla de la base de datos:
  - **INSERT INTO** *nombre_tabla* (lista de campos separados por comas) **VALUES** (lista de datos separados por comas)
- Ejemplo:
  - INSERT INTO categorias (idCategoria, Descripcion) VALUES (1, 'Software');
  - Si idCategoria es un campo autoincremental se puede obviar:

    INSERT INTO categorias (Descripcion) VALUES ('Software');

OBS: los valores para campos de tipo cadena y de tipo fecha van encerrados entre comillas simples, los valores numéricos no llevan comillas.

SQL – DML: UPDATE

- Para Actualizar registros de una tabla de la base de datos:

  UPDATE *nombre_tabla*
  SET campo1 = valor1, … , campoN = valorM
  WHERE condiciones

- Ejemplo:

```
UPDATE  entradas Set Titulo = 'Nuevo Titulo', idCategoria = 3
    WHERE idEntrada = 1
```

¿Que pasa si no ponemos la clausula WHERE?

Se actualiza TODA LA TABLA

SQL – DML: DELETE

- Para Eliminar registros de una tabla de la base de datos:

  DELETE FROM *nombre_tabla*

  WHERE condiciones

- Ejemplo:

  DELETE FROM entradas WHERE idEntrada = 1

¿Que pasa si no ponemos la clausula WHERE?

Integridad de los Datos: Nulos

- Cuando en un registro un atributo es desconocido, se dice que es **nulo** y se lo trata de forma diferente.
- Un **nulo** no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado.
- El **nulo** implica ausencia de información:
  - porque al insertar el registro se desconocía el valor del atributo, ó
  - porque para dicho registro el atributo no tiene sentido.

Integridad de los Datos

- Para las **claves primarias** de las tablas, ninguno de los atributos que componen la clave primaria puede ser nulo…..

Por qué?

¿Es posible insertar una tupla con algún atributo clave con valor nulo?

Integridad Referencial

- Se aplica a las **claves foráneas** de las tablas
- Si en una relación hay alguna clave foránea:
  - Sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien,
  - Deben ser completamente nulos.

Integridad Referencial

Por lo tanto, para cada **clave foránea:**

- ***Regla de los nulos:*** *¿Tiene sentido que la clave foránea* acepte nulos?
- ***Regla de borrado:*** *¿Qué ocurre si se intenta borrar la tupla* referenciada por la clave foránea?
- ***Regla de modificación:*** *¿Qué ocurre si se intenta modificar el* valor de la clave primaria de la tupla referenciada por la clave foránea?
- ***Acciones que se pueden tomar: No Permitir, Propagar, Colocar Nulos***

Funciones de agregación

- **Count(*)**: retorna la cantidad de registros encontrados.
  - Ej: SELECT COUNT(*) FROM entradas;

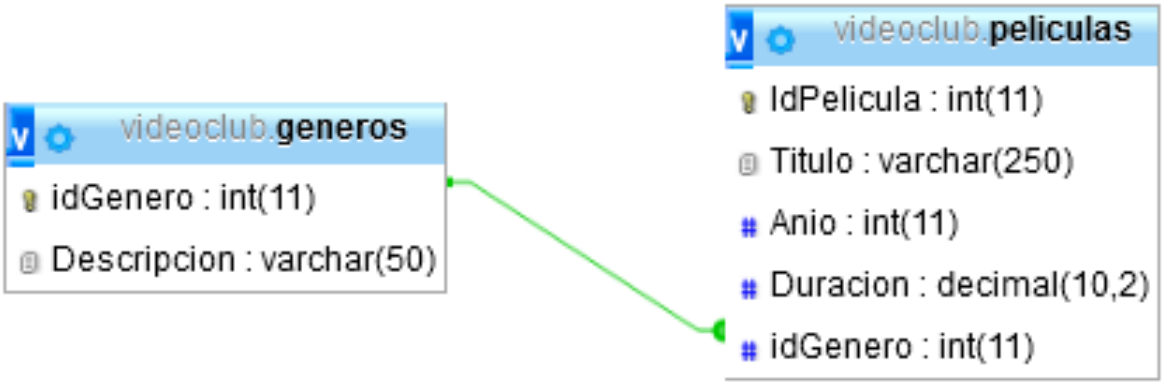
    SELECT COUNT(*) FROM entradas WHERE idCategoria = 1;
- **Sum(*col*):** Suma de los valores de la columna *col* de los registros seleccionados.
  - Ej: SELECT SUM(Deuda) FROM Clientes WHERE Localidad = ‘Neuquén’

Funciones de agregación

- **AVG(*col*):** Promedio de todos los valores de la columna *col* de los registros seleccionados.
  - Ej: SELECT AVG(Deuda) FROM Clientes WHERE Localidad = ‘Neuquén’
- **MAX(*col*):** Valor máximo de todos los valores de la columna *col* de los registros seleccionados.
  - Ej: SELECT MAX(Deuda) FROM Clientes WHERE Localidad = ‘Neuquén’
- **MIN(*col*):** Calcula el valor mínimo de todos los valores de la columna *col* de los registros seleccionados.
  - Ej: SELECT MIN(Deuda) FROM Clientes WHERE Localidad = ‘Neuquén’

Ejercicio

- Dado el siguiente modelo relacional, crear las tablas en su base de datos con phpMyAdmin:



Ejercicio

- Inserte en la tabla “generos” los siguientes datos:

idGenero	Descripcion
1	Drama
2	Terror
3	Romance

- Inserte en la tabla “peliculas” los siguientes datos:

idPelicula	Titulo	Anio	Duracion	idGenero
1	El sexto sentido	2000	1.47	1
2	Cuento de invierno	2014	1.52	3
3	Identidad secreta	2011	2.05	1
4	Un Angel Enamorado	1998	1.45	3
5	Oculus	2013	2.13	2

Ejercicio

- Liste todas las películas.
- Liste solo las películas que se hayan estrenado después del 2010.
- Cuantas películas de Drama hay cargadas?
- Actualice la duración a 1.59 de la película cuyo id es 3.

Material

- http://pedco.fi.uncoma.edu.ar/file.php/1548/2014/Apuntes/Apunte_Unidad_4.pdf
- Manual Mysql
- https://dev.mysql.com/doc/refman/5.0/es/