



**Argentina
programa
4.0**



Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

***primero
la gente***

Clase 25: Bases de datos Relacionales

Data Definition Language

Agenda de hoy

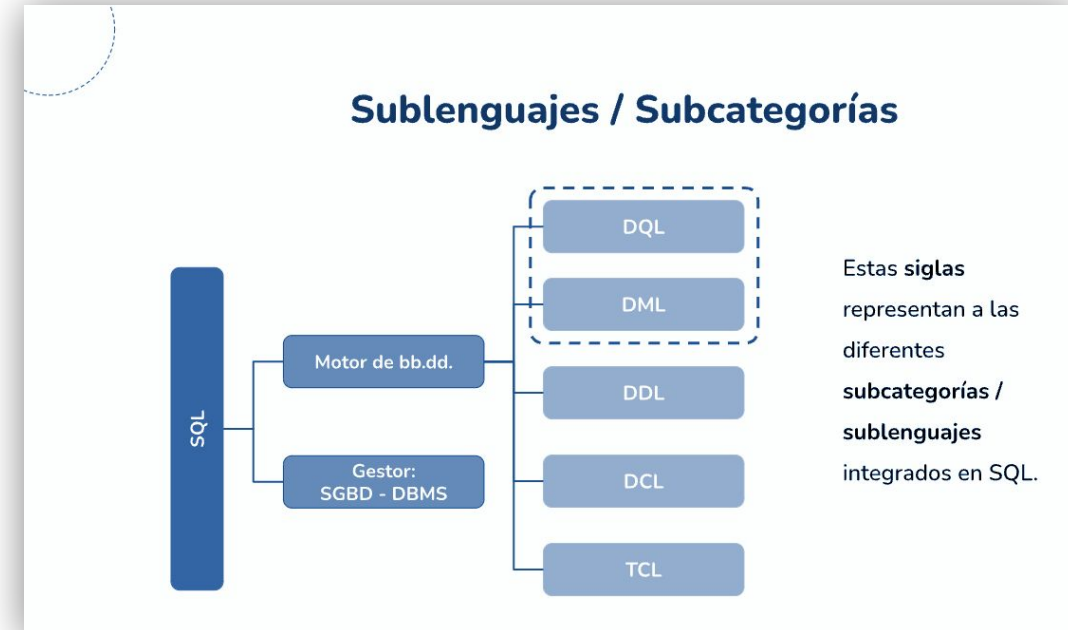
- A. Data Definition Language
- B. Crear tabla con el lenguaje SQL
 - a. CREATE TABLE
 - b. ALTER TABLE
 - c. Add
 - d. Modify
 - e. Drop
- C. Limitaciones al modificar una tabla
- D. Crear múltiples tablas utilizando DDL



Data Definition Language

Data Definition Language

En nuestros primeros encuentros enfocados a bases de datos SQL, hablamos de los sublenguajes que este posee, que son como categorías que definen cómo hacer determinadas operaciones sobre esta base de datos.



Data Definition Language



Hoy profundizaremos sobre la categoría **DATA DEFINITION LANGUAGE**, para comenzar a sacar partido de todos sus beneficios.

Data Definition Language

DQL

DML

DDL

DCL

TCL

DATA DEFINITION LANGUAGE permite modificar la estructura de objetos de una base de datos SQL.

Está conformado por algunas sentencias que nos permiten crear, modificar, borrar o definir un cambio en la estructura de tablas, vistas y otros objetos que se almacenan en la base de datos.

Data Definition Language

Las sentencias disponibles a través de DDL, son:

- CREATE
- ALTER
- DROP
- TRUNCATE

Con ellas creamos, modificamos, y eliminamos objetos o información extendida contenida en estos.



CREATE

Create

La sentencia **CREATE** cumple la función de crear nuevos objetos en la base de datos.

Los tipos de objetos a crear pueden ser: tablas, índices, stored procedures y hasta nuevas bases de datos, además de usuarios específicos.



Create Database / Schema

Create Database / Schema

A continuación tenemos un ejemplo de cómo crear una base de datos utilizando la cláusula SQL CREATE.

CREATE SCHEMA o **CREATE DATABASE**, seguido del nombre de la base de datos, es la cláusula que necesitamos para crear una base de datos nueva dentro de nuestro motor MySQL. Además, para soportar todos los juegos de caracteres posibles, sumamos la definición del tipo de caracteres **UTF-8**

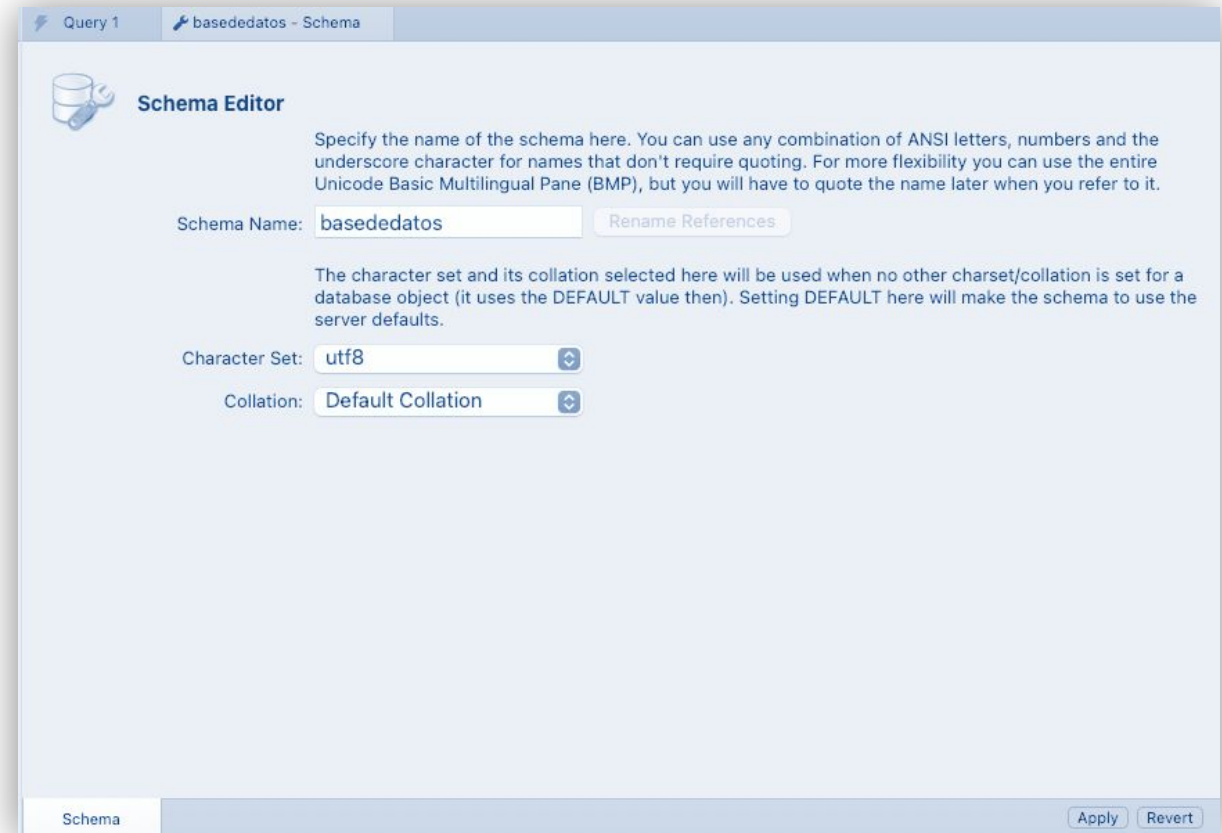
```
DDL

CREATE SCHEMA `basededatos`
DEFAULT CHARACTER SET utf8;
```

Create Database / Schema

Es lo mismo que vimos oportunamente, cuando creamos una base de datos nueva utilizando MySQL Workbench.

Cualquiera de las dos vías, es totalmente válida, aunque en determinados escenarios de trabajo podemos no llegar a contar con un RDBMS y debamos recurrir a la Consola de Administración de MySQL.

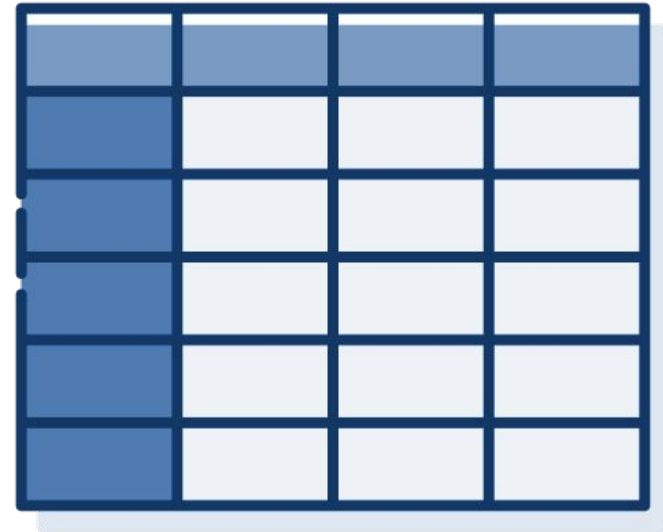


Create Table

Create table

Para definir los campos o columnas al momento de crear una tabla, debemos indicar el nombre que queremos para ésta, el tipo de dato que contendrá (*con o sin límite en cantidad de caracteres*), y si acepta o no valores nulos.

De forma similar a cuando vimos la creación de una tabla de forma manual, pero integrando el proceso junto a la cláusula **CREATE**.



Create table

Aquí tenemos un ejemplo de la estructura de código que debemos aplicar utilizando la cláusula CREATE TABLE.

Analizando el mismo, vemos que respeta una estructura bastante similar a la herramienta gráfica de MySQL Workbench para crear tablas, definir campos, tipos, longitud, y parámetros adicionales de configuración.

```
SQL CREATE tabla

CREATE TABLE amigos (
  id INT NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(30) NOT NULL,
  apellido VARCHAR(30) NOT NULL,
  email VARCHAR(50) NOT NULL,
  ([parámetros de la tabla]));
```


Create table

Tomemos como ejemplo el campo **id**:
Al definirlo, le indicamos una serie de parámetros que demarcan su tipo de dato además del comportamiento que se espera.

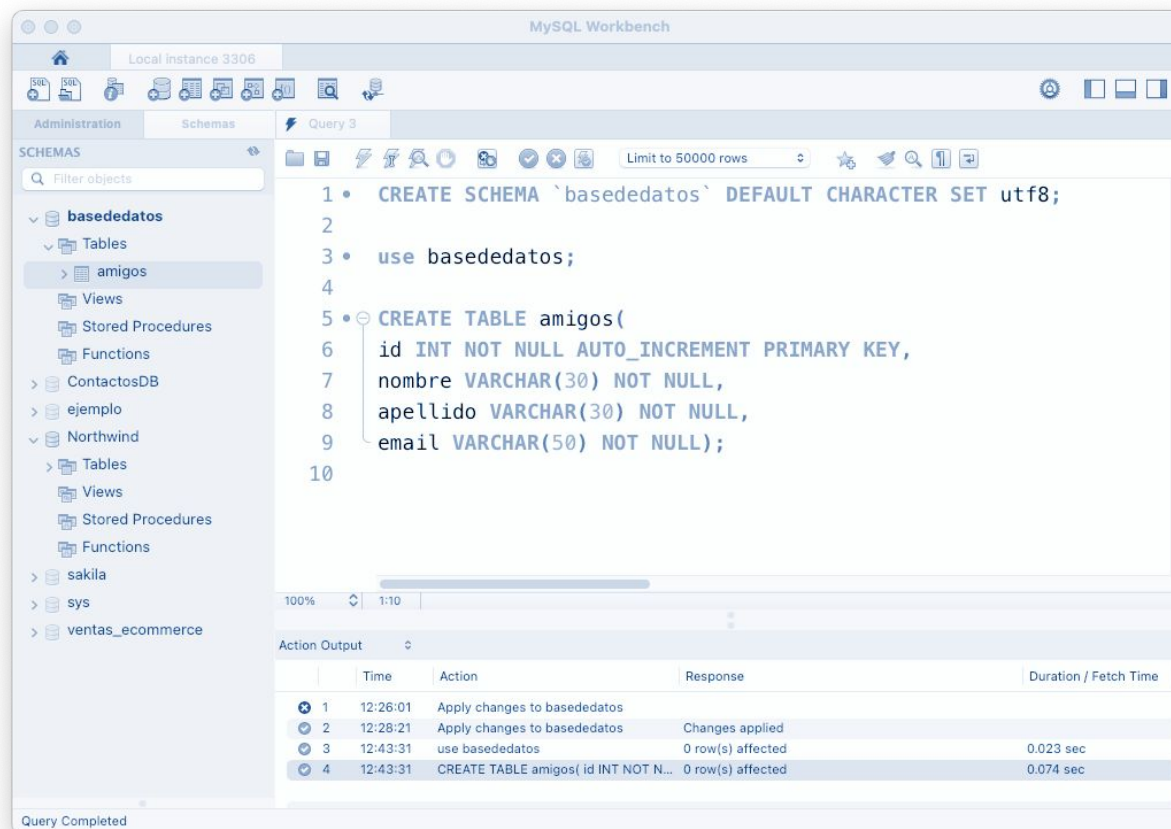
Por ejemplo, la información que contendrá es numérica, definida por el tipo de dato entero (INT), no acepta valores nulos o vacíos **NOT NULL**, y su valor será automática y autoincremental **AUTO_INCREMENT**.

```
SQL CREATE tabla

CREATE TABLE amigos (
  id INT NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL,
  apellido VARCHAR(30) NOT NULL,
  email VARCHAR(50) NOT NULL,
  ([parámetros de la tabla]));
```

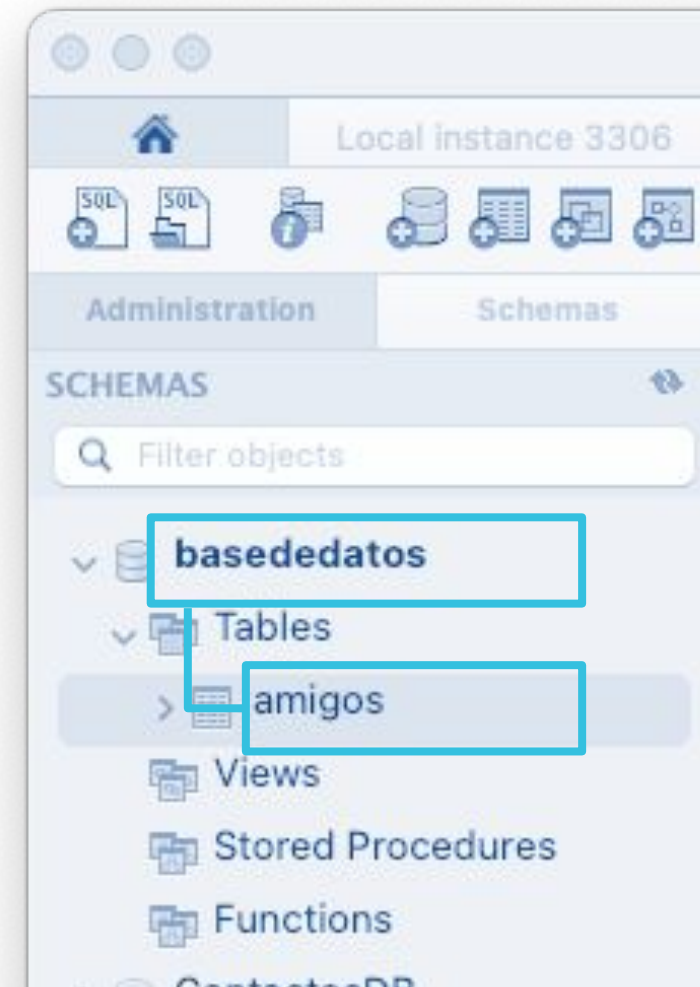
Create table

Veamos un ejemplo de crear una **base de datos**, posicionarnos en ella, y crear a continuación una tabla denominada **amigos**.



Create table

Si todo fue bien, debemos ver en el **panel Schemas** de MySQL Workbench, la base de datos (*homónima*) creada, y dentro del apartado **Tables**, la tabla **amigos**, también creada.



ALTER

Alter table

La sentencia **ALTER** , del inglés “*alterar*”, permite modificar la estructura de una tabla u objeto de base de datos.

Con ella podemos *agregar* o *quitar* campos, *modificar* el tipo de datos de un campo y *agregar* o *quitar* índices.



Alter table

Si debemos agregar un campo a una tabla, combinamos el uso de **ALTER TABLE** seguido de la cláusula **ADD**.

SQL ALTER

```
ALTER TABLE [nombre de la tabla]  
ADD [definiciones de columnas];
```

Alter table

[nombre de la tabla]: definimos el nombre distintivo de la tabla a modificar.

ADD: es la acción que realizaremos sobre la tabla; en este caso, agregar un **nuevo campo**.

[definición de columna]: Definimos la o las nuevas columnas, tal como hicimos con la sentencia **CREATE**.

SQL ALTER table

```
ALTER TABLE amigos  
ADD telefono VARCHAR(30);
```

Alter table

En el caso de tener que agregar más de un campo en la misma operación, agregamos uno del otro separado por una coma. Cada uno debe llevar su definición correcta.

```
SQL ALTER table  
  
ALTER TABLE amigos  
ADD telefono VARCHAR(30),  
ADD comentarios VARCHAR(200);
```


modify

Alter table

Si deseamos modificar los valores para un campo existente, reemplazamos la instrucción **ADD** por **MODIFY**.

Seguido a ello, definimos el campo y el cambio que deseamos aplicar en su estructura.



Alter table

En el siguiente ejemplo vemos una mínima modificación en la longitud de el campo **telefono**, agregado anteriormente sobre la tabla **amigos**.

```
SQL ALTER modify  
  
ALTER TABLE amigos  
MODIFY telefono VARCHAR(50) NOT NULL;
```

Alter table

También, podemos hacer el mismo procedimiento que con la cláusula ADD, agregando una cláusula **MODIFY** por cada uno de los campos que deseemos modificar en un mismo proceso.

```
SQL ALTER table  
  
ALTER TABLE amigos  
MODIFY telefono VARCHAR(50) NOT NULL,  
MODIFY comentarios VARCHAR(220);
```

Otras modificaciones

Otras modificaciones

CHANGE COLUMN: podemos cambiar el nombre de una columna previamente definida.

RENAME TO: podemos cambiar el nombre inicial de una tabla, por uno nuevo.

DROP COLUMN: podemos eliminar una columna o campo.



Change column name

Cambiamos el nombre de la columna **comentarios** por el nombre **sugerencias**.



CHANGE field

```
ALTER TABLE amigos  
CHANGE comentarios sugerencias VARCHAR(500);
```

Change table name

Modificar el nombre de la tabla
amigos por el nombre **contactos**.

● ● ● RENAME table

```
ALTER TABLE amigos  
RENAME contactos;
```


Drop column



RENAME table

```
ALTER TABLE contactos  
DROP sugerencias;
```

Limitaciones al modificar una tabla

Limitaciones al modificar una tabla

Cuando tuvimos nuestro primer contacto con MySQL Workbench para crear una tabla en una base de datos, verificamos diferentes condiciones que siempre se deben tener en cuenta una vez creada la tabla en cuestión.

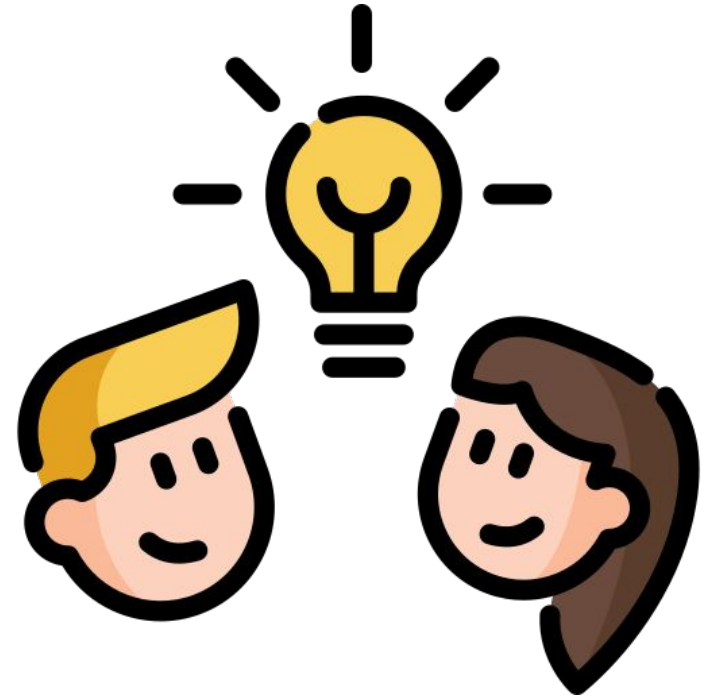
Veamos a continuación un refresh de dichas condiciones dado que estamos llevando a cabo el mismo procedimiento, pero esta vez mediante código SQL.



Limitaciones al modificar una tabla

Existen restricciones al momento de renombrar una tabla, de igual forma que con la necesidad de cambiar el tipo de datos de un campo.

Tengamos presente que, estas cosas sí se pueden realizar en MySQL Workbench y mientras estamos aprendiendo pero, **sobre bases de datos en producción, estas actividades No Se Deben Realizar**, o se realizarán luego de un exhaustivo análisis sobre los posibles problemas que puedan surgir.

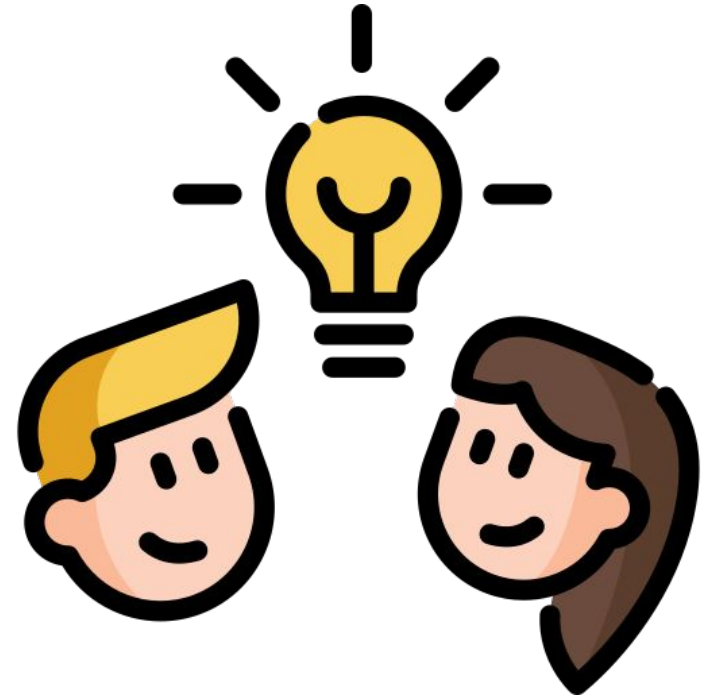


Limitaciones al modificar una tabla

Los roles de **Arquitectos de software, DBA's, Data Engineer, o Líderes técnicos**, son generalmente quienes delinean la estructura principal de una base de datos.

Este proceso tiene un exhaustivo análisis y cuando llega el momento de crear tablas y campos, es porque dicho análisis será el definitivo.

Muy rara vez puede darse la situación de que se creen campos y luego se busquen eliminar o modificar su estructura original.



**Crear múltiples
tablas mediante
DDL**

Crear múltiples tablas mediante DDL

Veamos a continuación un ejemplo de cómo podemos pensar la creación de múltiples tablas SQL a partir de la cláusula **CREATE TABLE**, teniendo en cuenta que estas tablas deben estar relacionadas entre sí.

Dicho ejemplo estará basado en un modelo de **Estudiantes, Profesores y Aulas**, relacionando estas tablas entre sí.



Crear múltiples tablas mediante DDL

La tabla estudiantes posee información simple relacionada a los estudiantes de una institución.

Su campo **idAula** será el nexa con la tabla **Aulas**, para saber en qué ubicación les toca cursar.

Estudiantes
idEstudiante
nombreCompleto
idAula
fechaNacimiento
domicilio

Crear múltiples tablas mediante DDL

La tabla **Profesores** tiene información de los profesores en cuestión, pero no posee relación con ninguna de las otras dos tablas que componen este modelo.

Profesores
idProfesor
nombreCompleto
materia
fechaContratacion
telefono

Crear múltiples tablas mediante DDL

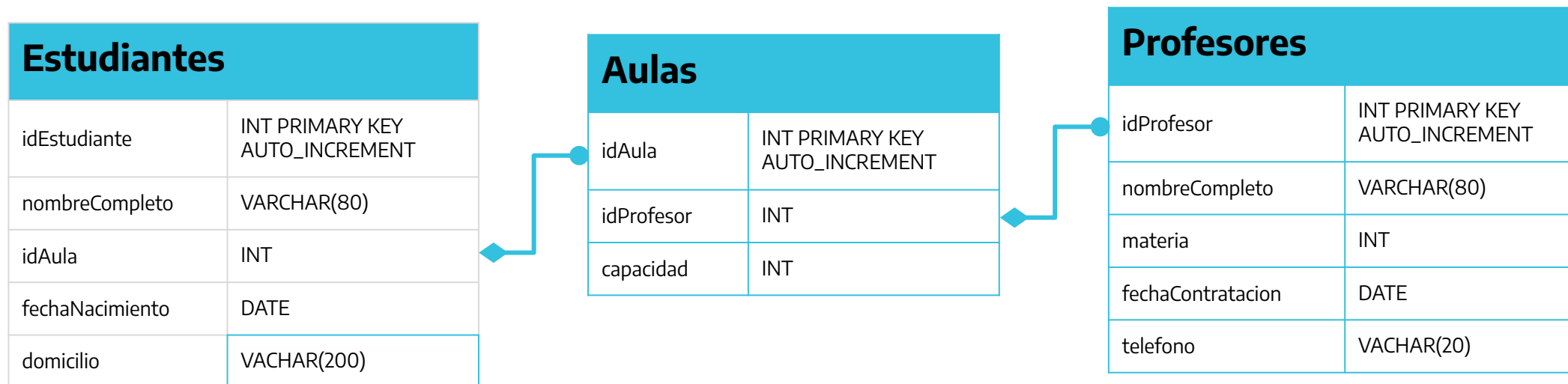
La tabla **Aulas** se relaciona con la tabla profesores, teniendo un campo **idProfesor** que oficia de clave foránea.

A su vez, **idAula** será la clave foránea de la tabla **Estudiantes**.

Aulas
idAula
idProfesor
capacidad

Crear múltiples tablas mediante DDL

Veamos la estructura de datos de estas tres tablas, para realizar luego las cláusulas DDL que permitan crearlas, estableciendo desde DDL el tipo de relación entre ellas.



Crear múltiples tablas mediante DDL

La tabla **Profesores** será la primera que creamos. Al no tener una dependencia foránea en ninguno de sus campos, podremos crearla en primer lugar sin problema alguno.

```
CREATE TABLE

CREATE TABLE Profesores (
  idProfesor INT PRIMARY KEY AUTO_INCREMENT,
  nombreCompleto VARCHAR(100),
  materia VARCHAR(50),
  fechaContratacion DATE,
  telefono VARCHAR(20)
);
```

Profesores

idProfesor

nombreCompleto

materia

fechaContratacion

telefono

Crear múltiples tablas mediante DDL

Luego creamos la tabla **Aulas**. Esta tabla utilizará el campo Id de la tabla Profesores, como clave foránea.

```
CREATE TABLE

CREATE TABLE Aulas (
  idAula INT PRIMARY KEY AUTO_INCREMENT,
  idProfesor INT,
  FOREIGN KEY (idProfesor) REFERENCES
  Profesores(idProfesor),
  capacidad INT
);
```

Aulas

idAula

idProfesor

capacidad

Crear múltiples tablas mediante DDL

Finalmente creamos la tabla **Estudiantes**. Esta tabla utiliza un campo con clave foránea a la columna **id** de la tabla **Aulas**.

```
CREATE TABLE

CREATE TABLE Estudiantes (
  idEstudiante INT PRIMARY KEY AUTO_INCREMENT,
  nombreCompleto VARCHAR(100),
  idAula INT,
  fechaNacimiento DATE,
  direccion VARCHAR(200),
  FOREIGN KEY (idAula) REFERENCES Aulas(idAula));
```

Estudiantes

idEstudiante

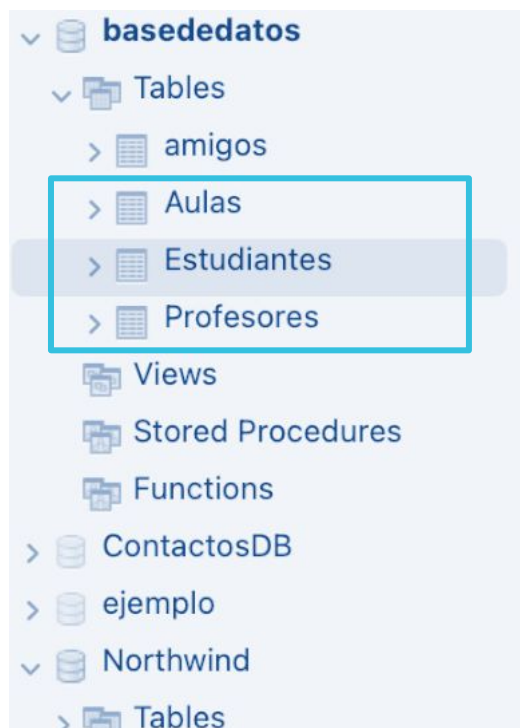
nombreCompleto

idAula

fechaNacimiento

domicilio

Crear múltiples tablas mediante DDL



Aquí tenemos el resultado de la creación de las tres tablas en cuestión, más el agregado de algunos registros en su interior.

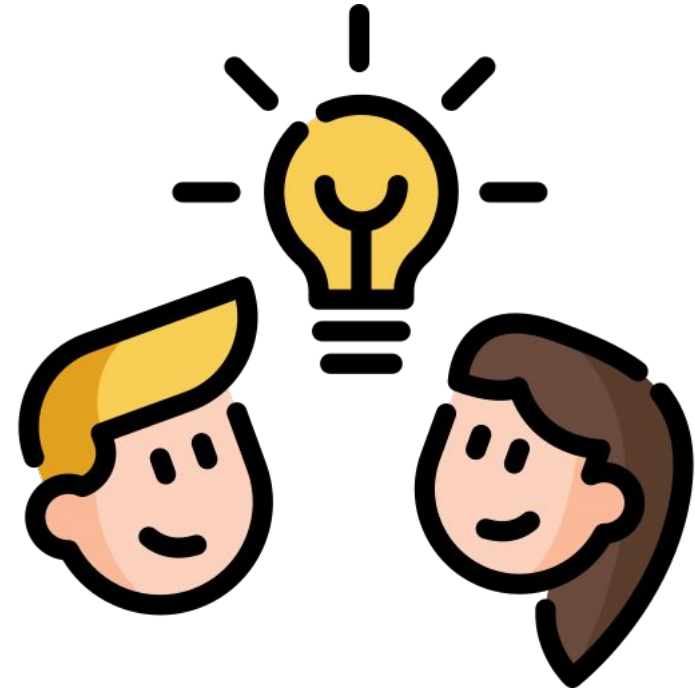
idEstudiante	nombreCompleto	idAula	fechaNacimiento	direccion
1	Juan Pérez	1	1999-05-10	Calle Principal 123
2	María López	1	2000-02-15	Avenida Central 456
3	Carlos Ramírez	2	1998-09-20	Calle Secundaria 789

Crear múltiples tablas mediante DDL

Partiendo de este último ejemplo abordado en la creación de múltiples tablas relacionales, utilizando la referencia a campos foráneos.

¿Se animan a adivinar a qué Forma Normal pertenece este modelo de tablas creado?

Hagan memoria sobre nuestra clase titulada “*El modelo Relacional*” donde utilizamos un ejemplo similar al aquí abordado.



Sección práctica

Desarrollaremos a continuación un ejercicio de similares características a este último.

Veamos la consigna para luego comenzar a trabajar en SQL.



Prácticas

A partir del siguiente modelo de Tablas, deberás establecer cuál es la relación entre las mismas, y crear las cláusulas **SQL DDL** correspondientes para luego insertar datos.

Equipos
idEquipo
nombreEquipo
especialidad
idCliente

Clientes
idCliente
nombreEmpresa
rubroEmpresa

Empleados
idEmpleado
nombreEmpleado
puestoEmpresa
idEquipo

Prácticas

Previo al proceso de crear las tablas, define:

1. El tipo de datos más apropiado para cada columna
1. El tipo de relación entre una tabla con otras tablas de este modelo
1. Crea a continuación las cláusulas SQL DDL en el orden que corresponde crearlas
 - a. Define en las cláusulas correspondientes, el uso de FOREIGN KEY
 - b. No olvides agregar PRIMARY KEY y AUTO_INCREMENT en los campos principales



Muchas gracias.



Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

*primero
la gente*