



Agencia de
Aprendizaje
a lo largo
de la vida

BIG DATA Data Analytics

Bases de Datos Relacionales - SQL

Les damos la bienvenida

Vamos a comenzar a grabar la clase

SQL

Structured Query Language

SQL

SQL es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

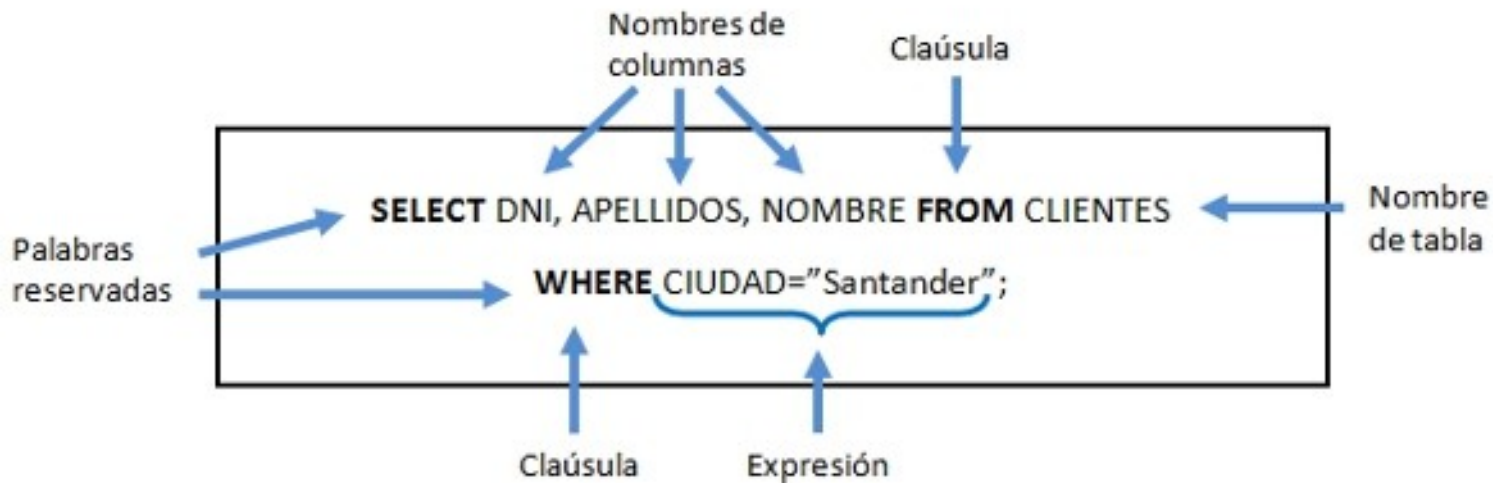
Sintaxis

INSTRUCCION {FORMA1 | FORMA2} [OPCIONAL] un_nombre

- En mayúsculas: palabras propias de SQL
- Entre llaves: pueden cambiar según entorno
- Entre corchetes: instrucciones opcionales
- En minúsculas: nombres de datos variables puestos o no por el programador

```
CREATE TABLE alumnos (
  nombre VARCHAR(50) NOT NULL,
  apellido VARCHAR(50) NOT NULL,
  edad INT(3) NOT NULL,
  sexo CHAR(1) NOT NULL,
  PRIMARY KEY (nombre, apellido)
);
```

Estructura de una instrucción SQL



SQL - DDL

Lenguaje de definición de datos
Creando la base de datos

Crear BBDD

```
CREATE {DATABASE | SCHEMA}  
[IF NOT EXISTS] db_name  
[DEFAULT CHARACTER SET utf8]  
[COLLATE utf8_spanish_ci];
```

Character set: cómo SQL almacena internamente el dato

Collate: cómo SQL compara y ordena campos de texto



SQL - DDL(2)

Creando y modificando la estructura de las tablas

Crear Tabla

```
CREATE TABLE nombre_tabla (  
  nombre_columna1 [definición_columna][PRIMARY KEY],  
  nombre_columna2 [definición_columna]...  
  nombre_columnaN [definición_columna  
  FOREIGN KEY(nombre_de_otra_tabla)  
  nombre_de_otra_tabla(nombre_columna_pk_otra_tabla)  
);
```

Definición de columna: indicamos qué tipo de dato y si se admite que quede vacío.

Crear Tabla: Ejemplo

```
CREATE TABLE artistas (  
  id_artista INTEGER PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL);
```

```
CREATE TABLE temas (  
  id_tema INTEGER PRIMARY KEY,  
  nombre_tema VARCHAR(50) NOT NULL,  
  id_artista INTEGER,  
  FOREIGN KEY(id_artista) REFERENCES  
  artistas(id_artista));
```

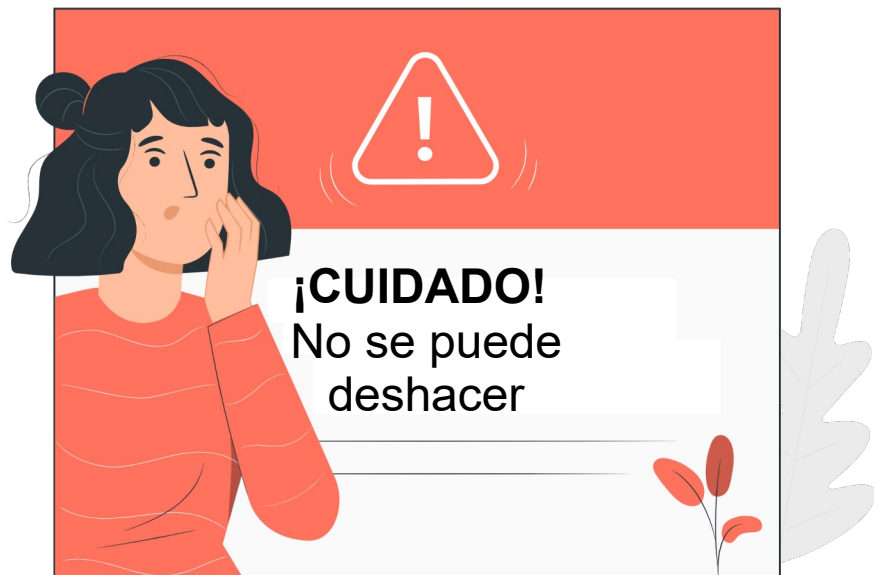
Eliminar Tabla

```
DROP TABLE  
[IF EXISTS] nombre_tabla
```

Ejemplo:

```
DROP TABLE IF EXISTS temas
```

No podemos borrar una tabla que está relacionada por FK con otra tabla sin eliminar previamente la relación o la otra tabla.



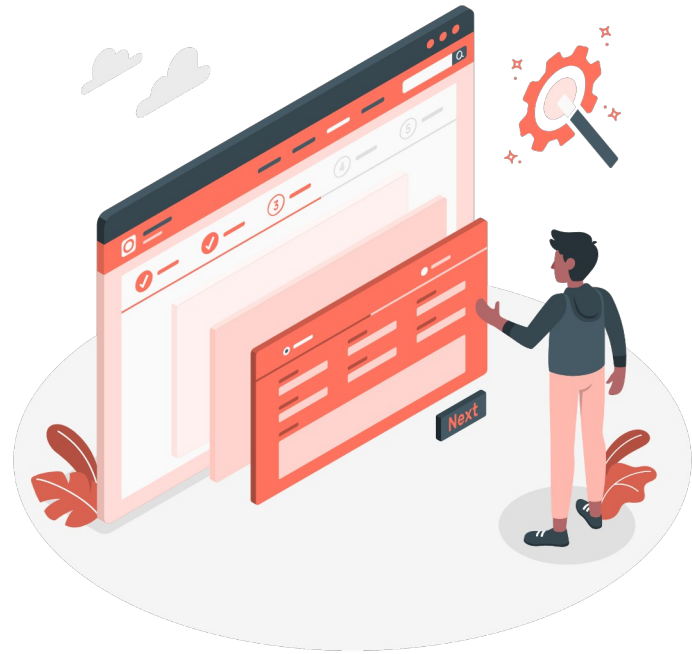
Modificar Tabla

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;
```



Modificar Tabla: Ejemplo

```
ALTER TABLE temas ADD genero VARCHAR(20);
```

```
ALTER TABLE temas ADD otra_columna INTEGER;
```

```
ALTER TABLE temas DROP COLUMN otra_columna;;
```

SQL - DML

Data Management Language
Insertando datos en la base de
datos

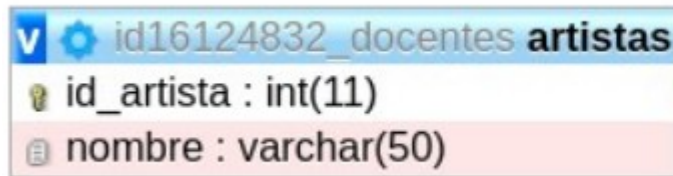
Insertar Registro

Sintaxis:

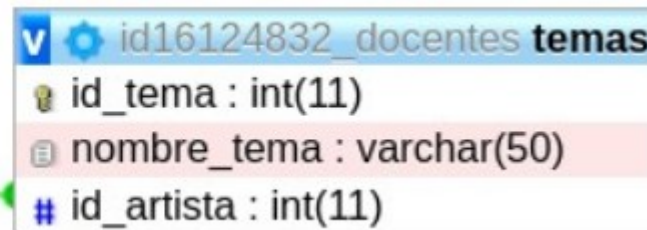
```
INSERT INTO nombre_tabla (campo1, campo2...)  
VALUES (dato1, dato2...);
```

Ejemplo:

```
INSERT INTO artistas (id_artista, nombre)  
VALUES (1, 'Beyoncé');
```



id16124832_docentes	artistas
💡	id_artista : int(11)
📄	nombre : varchar(50)



id16124832_docentes	temas
💡	id_tema : int(11)
📄	nombre_tema : varchar(50)
#	id_artista : int(11)

SQL - DML(2)

Modificando y borrando registros

Actualizar Registro

Sintaxis:

```
UPDATE nombre_tabla SET cambio  
WHERE condición;
```

Ejemplo:

```
UPDATE artistas SET nombre = 'Prince'  
WHERE id_artista = 1;
```

v	id16124832_docentes	artistas
💡		id_artista : int(11)
📋		nombre : varchar(50)

v	id16124832_docentes	temas
💡		id_tema : int(11)
📋		nombre_tema : varchar(50)
#		id_artista : int(11)

Borrar Registro



Sintaxis :

```
DELETE FROM nombre_tabla  
WHERE condición;
```

Ejemplo:

```
DELETE FROM artistas  
WHERE nombre = 'Prince';
```

SQL - DML(3)

Consultando la base de datos

Consultas Básicas

Muestra todos los campos de todos los registros

```
SELECT * FROM nombre_tabla;
```

Muestra los campos **nom_columna** de todos los registros

```
SELECT nom_columna1 [,nom_columna2] FROM  
nombre_tabla;
```

Consultas Básicas: Ejemplos

```
SELECT * FROM empleados;
```

```
SELECT apellido, nombre FROM empleados;
```

```
SELECT apellido, nombre, sector, salario  
FROM empleados;
```

```
SELECT nombre FROM artistas;
```

Consultas - Cláusulas

Cláusula	Descripción
FROM	Especifica la tabla de la cual se van a seleccionar los registros
WHERE	Especifica las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Separa los registros seleccionados en grupos específicos
HAVING	Expresa la condición que debe satisfacer cada grupo
ORDER BY	Ordena los registros seleccionados de acuerdo a un orden específico

Consultas

Sintaxis :

```
SELECT * | [DISTINCT] nombre-columna  
FROM nombre_tabla  
WHERE condición  
GROUP BY columnas_de_agrupación  
HAVING condición  
ORDER BY columnas_de_ordenación [DESC];
```


Consulta condicional

Muestra todos los campos de todos los registros que cumplan con el requisito especificado en la cláusula WHERE

```
SELECT campo1[,campo2...] FROM tabla WHERE  
condición_a_cumplir;
```

Consulta condicional: Ejemplo

```
SELECT nombre FROM artistas WHERE genero =  
"Folklore";
```

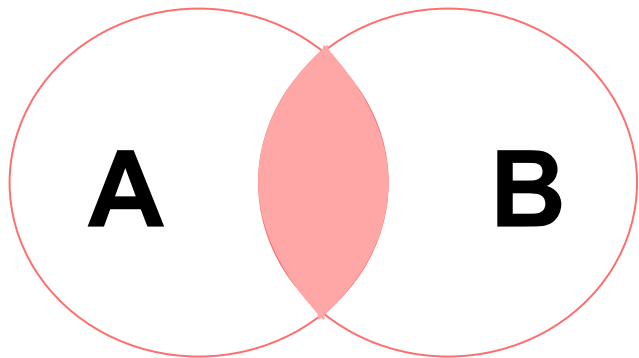
```
SELECT nombre FROM empleados WHERE sueldo  
> 100000;
```

```
SELECT nombre FROM empleados WHERE NOT  
apellido = "Hernández"
```

JOINS

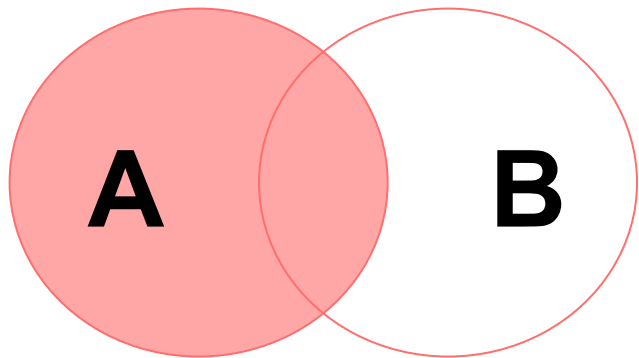
Consultando tablas relacionadas

Joins: INNER JOIN



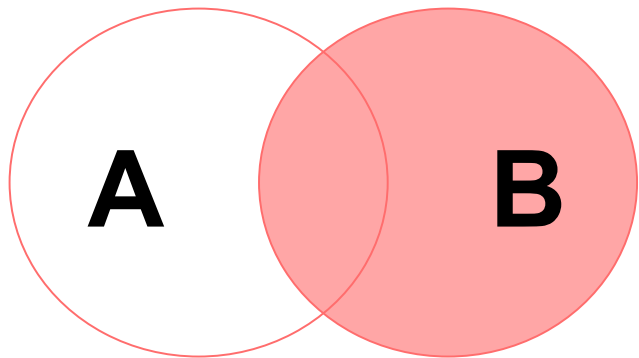
```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

Joins: LEFT JOIN



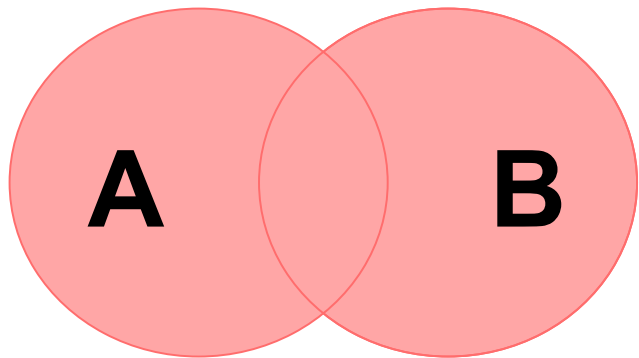
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
[WHERE B.key IS NULL]
```

Joins: RIGHT JOIN



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key  
[WHERE B.key IS NULL]
```

Joins: FULL JOIN



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key  
[WHERE A.key IS NULL  
OR B.key IS NULL]
```

Desafío

Práctica integradora

Recurso sugerido: <http://sqlfiddle.com/>

Crear una tabla de ciudades llamada “city”, según los siguientes requisitos:

id	INT(11)	NOT NULL
nombre	VARCHAR(35)	NOT NULL
codigo	VARCHAR(3)	NOT NULL
distrito	VARCHAR(20)	NOT NULL
nro_habitantes	INT(11)	NOT NULL

Recordatorio: Crear tabla

Crear Tabla

```
CREATE TABLE nombre_tabla (  
  nombre_columna1 [definición_columna][PRIMARY KEY],  
  nombre_columna2 [definición_columna]...  
  nombre_columnaN [definición_columna]  
  FOREIGN KEY(nombre_de_otra_tabla)  
  nombre_de_otra_tabla(nombre_columna_pk_otra_tabla)  
);
```

Definición de columna: indicamos qué tipo de dato y si se admite que quede vacío.

Práctica integradora

2. Cargar los siguientes datos en la tabla:

id	nombre	codigo	distrito	nro_habitantes
69	Buenos Aires	ARG	Ciudad Autónoma de Buenos Aires	2982146
70	La Matanza	ARG	Buenos Aires	1266461

3. Obtener el resto de los datos y cargarlos (excepto los ya introducidos en el punto 2)

Fuente: <https://gist.github.com/programming-Regina/f6ddb3c3bc055f7f8ff982c1724f6536>

Práctica integradora

4. Realizar búsquedas para obtener:

- a. Todos los campos de todos los registros.
- b. Todos los datos de Florencio Varela.
- c. Sólo los nombres de las ciudades.
- d. Los nombres de ciudades y cantidad de habitantes.
- e. Idem anterior para más de 400.000 habitantes.
- f. Idem d. para ciudades entre 90.000 y 200.000 habitantes.
- g. Idem d. Ordenados alfabéticamente.

Enlaces

Ejercicios:

- Guía práctica de SQL, Cimino + Archivos (worl.sql y inupde.sql): en el aula virtual

Documentación SQL

Software:

- [DB Browser for SQL Lite](#)
- [MySQL Workbench](#)
- [Visual Studio Code](#)
- [SQL Fiddle](#)
- [Extensiones VSC](#)
 - MySQL de Jun Han
 - MySQL Syntax de Jake Bathman
- [XAMPP](#)

Créditos: Template de [Slidesgo](#), íconos de [Flaticon](#), infografías
e imágenes de [Freepik](#) e ilustraciones de [Stories](#)

Recordá:

Revisar tu casilla de Spam.

**Chequear la cartelera de Novedades y
dejar tus consultas en el Foro del Aula
Virtual.**

No te olvides de dar el presente

¿Consultas?