

## **Bases de datos**

### **¿Qué es una base de datos? (BD o DB)**

Cualquier colección de información que se encuentra relacionada. Los siguientes ejemplos son útiles:

Base de datos para una editorial

- Lista de libros por autor
- Lista de ventas por libro
- Lista de almacenes en librerías
- Lista de librerías de la editorial
- Listas adicionales

Base de datos para una tienda

- Lista de clientes
- Lista de compras al día
- Lista de ventas por cliente
- Lista de almacén por producto por stock
- Listas adicionales

Base de datos para un colegio

¿Qué listas pondrían?

Las bases de datos pueden ser guardadas en diferentes formas:

- En papel
- En google Drive
- En tu pc => Local
- En la nube => Internet.
  - Docker y Kubbernets
  - Microservicios
- En Kaggle
- Git y Github

### **Manejadores de bases de Datos**

Son softwares que permiten el manejo de bases de datos. Se pueden manejar varias al mismo tiempo.

PostgreSQL	<a href="http://www.postgresql.org">www.postgresql.org</a>	PostgreSQL
Microsoft	<a href="http://www.microsoft.com">www.microsoft.com</a>	Access, MS-SQL Server
MySQL	<a href="http://www.mysql.com">www.mysql.com</a>	MySQL
Informix	<a href="http://www.informix.com">www.informix.com</a>	Illustra, Universal Server, Dynamic Server

## Modelos de Bases de Datos importantes

Hay dos tipos de modelos de Bases de datos: Los modelos relacionales y los modelos con datos semiestructurados, que incluyen XML y estándares relacionados. En este curso, al no ser una especialización, solamente estudiaremos brevemente los modelos relacionales.

### Modelo Relacional

El modelo relacional es el que está basado en tablas. Cada columna de la tabla representa un atributo de datos y en las filas, una tupla de datos. Las operaciones en el modelo relacional están orientadas al álgebra relacional.

Año	Título	Género	Duración
2019	Los vengadores	Acción	3h:1min
2020	Había una vez en Hollywood	Comedia	2h:41min
2021	Free guy	Sci-Fi	1h:55min

Tabla 1: Tabla de modelo relacional

Por ejemplo, en esta tabla. ¿Qué es lo que puede relacionar entre sí?. ¿A qué tipo de información pertenecen los siguientes datos?

### Modelo No Relacional

Solo para mencionar, los modelos no relacionales, son los tipos de modelos que no se pueden estructurar ni se relacionan mediante tablas. Por ejemplo se puede representar datos en formato XML o en formato json, entre otros. Este curso no tendrá en cuenta dichos formatos.

```

<Movies>
  <Movie title="Los vengadores">
    <year>2019</year>
    <duration>182</duration>
    <genero>drama</genero>
  </Movie>
  <Movie title="Había una vez en Hollywod">
    <year>2020</year>
    <duration>161</duration>
    <genero>comedia</genero>
  </Movie>
  <Movie title="Free guy">
    <year>2021</year>
    <duration>115</duration>
    <genero>sci-fi</genero>
  </Movie>
  *
</Movies>

```

Imagen 1. Archivo en XML

```

{"movies":{"Los vengadores":{"año": 2019, "duracion": 181, "genero": "accion"}, "Erase una vez en Hollywod":{"año": 2020, "duracion": 161, "genero": "comedia"}, "Free guy":{"año": 2021, "duracion": 115, "genero": "sci-fi"}}}

```

Imagen 2 . Archivo en formato json

## Fundamentos del modelo relacional

Algunos conceptos básicos del modelo relacional son importantes. Por ejemplo, los **atributos** son los nombres de cada columna, la relación entre los atributos y el conjunto de datos se llama **esquema**. Por ejemplo para la tabla anterior **el esquema sería : Películas (título, año, duración y género)**. Cada fila es una **tupla de datos**.

**Nota :** Las filas no pueden contener dentro ni conjuntos, ni matrices ni listas ni ningún tipo de dato que no sea pequeño como una cadena de texto o un número.

El problema principal es cómo evaluar las tuplas, es decir, cada tupla es única, por lo que debe existir un ID para identificar cada fila de manera especial. Este identificador se denota como Primary Key. Esta llave primera, puede ser natural como el DNI o inventada.

## Claves en una tabla

- Clave Foránea: Esta llave es la que relaciona una tupla de una tabla con una tupla de otra.

Por ejemplo tenemos 2 tablas:

DNI	Nombre	Edad	email
77299381	Ernesto	19	ernesto.juarez@gmail.com
45762718	Juan	18	juan.rulien@gmail.com
57128264	Gabriela	18	gabriela.fernandez@gmail.com

ID_ciudad	Ciudad	Departamento
1	Sullana	Piura
2	Piura	Piura
3	Chiclayo	La libertad

Tenemos la tabla expandida con el valor de ciudad como clave foránea.

DNI	Nombre	Edad	email	ID_ciudad
77299381	Ernesto	19	ernesto.juarez@gmail.com	1
45762718	Juan	18	juan.rulien@gmail.com	2
57128264	Gabriela	18	gabriela.fernandez@gmail.com	2

- Clave compuesta: Se necesitan dos claves para identificar cada tupla en una tabla
- Clave de trabajo con: Son claves que relacionan dos tuplas de tablas diferentes, para formar una tabla que tiene una relación entre dos tablas diferentes

## SQL (Lenguaje de consulta estructurado) (Structure Query language)

Sql (pronunciado “sequel”) es el principal lenguaje usado para describir y manejar bases de datos relacionales. Se usa en la interacción de los RDBMS. Hay 3 aspectos importantes en SQL:

- El sublenguaje para la definición de bases de datos y esquemas (DDL)
- El sublenguaje para la manipulación de bases de datos y para modificar las bases de datos. (DML, DQL)
- El sublenguaje para el control y los permisos para los diferentes usuarios que tendrán acceso a la base de datos. (DCL)

El lenguaje SQL permite:

- Crear, actualizar, eliminar y leer datos
- Crear y manejar bases de datos
- Diseñar bases de datos
- Desarrollar tareas administrativas : Seguridad, manejos de usuarios, etc.

## Relaciones en SQL

SQL hace distinciones en 3 tipos de relaciones :

- Relaciones guardadas en tablas. Se pueden cambiar las tuplas de tablas, cambiando la tabla
- Vistas, que son relaciones definidas por computadora. Estas relaciones no son guardadas, sino que se construyen cuando se necesitan.
- Tablas temporales, que son las que se construyen cuando se realizan consultas.

**Nota :** Normalmente las 4 operaciones centrales en las bases de datos son Crear, Actualizar, Leer y eliminar información dentro de la base de datos (En las tablas) y es lo que nos centraremos en el curso.

## Consultas

Una consulta es una instrucción que se hace a un RDBMS, escrito en SQL que le dice al RDBMS la información que deseas recibir.

## Creación de tablas

La creación de tablas es una parte importante de las bases de datos. Para esto es importante, cómo definir un esquema. Para definir un esquema es importante conocer tipos de datos.

1. **Char(n) o VarChar(n) :** Se usan para definir cadenas de texto. la diferencia es que char tiene un número de caracteres fijo. Por ejemplo si definimos un char(4) y escribimos "mar" se guardará como "mar " con un espacio extra.
2. **Booleanos :** Denota valores lógicos. Verdadero o falso
3. **INT :** Números enteros
4. **FLOAT:** Números decimales.
5. **Dates :** Existen dos tipos de fechas, Date y Timestamp.
6. **DECIMAL :** Se especifica un M con toda la cantidad de números y un N con los números luego de la coma decimal.

7. **BLOB:** Binary Large object, sirve para guardar grandes datos, como imágenes.

Por ejemplo, para declarar la tabla de películas, escribimos:

```
1 • CREATE TABLE Movies(  
2     titulo char(100),  
3     año int,  
4     duracion int,  
5     genero char(15),  
6     estudioNombre char(30)  
7 );  
8  
9
```

O podemos declarar otra tabla que se llame MovieStar

```
• CREATE TABLE MovieStar (  
    name char(30),  
    address varchar(255),  
    gender char(1),  
    birthdate date  
);
```

Describir tablas

La función DESCRIBE cumple un papel importante.

```
DESCRIBE Usuarios;
```

### Modificar tablas

- **DROP TABLE R;**
- **ADD + Nombre del atributo + su tipo de variable**
- **DROP + Nombre del atributo**

Para poder utilizar las últimas declaraciones SQL se necesita agregar ALTER TABLE R + la declaración: Por ejemplo

```
alter TABLE Movies add directorNombre char(16);
```

Ejemplo práctico : Crear las tablas para un sistema de login para usuarios y que permite almacenar sus películas favoritas.

Solución:

Una idea intuitiva sería empezar por registrar a los usuarios, su nombre y contraseña.

IdEmail	Nombre	Usuario	Contraseña

Y también las películas.

pelicula_Id	Nombre de la película	Año de publicación

Entonces en MySql podemos escribir

```

1 • CREATE TABLE Usuarios(
2     email char(100) primary key,
3     nombre varchar(25),
4     usuario varchar(10),
5     contraseña char(15)
6 );
7
8 • CREATE TABLE Peliculas(
9     pelicula_id int primary key,
10    nombre_pelicula varchar(15),
11    año_publicacion int
12 );
13 • DESCRIBE Usuarios;
14 • DESCRIBE Peliculas
15

```

### Insertar valores en tablas

Para insertar valores en tablas, tenemos que utilizar la sintaxis:

**INSERT INTO + R + VALUES();**

**Forma de ingresar datos :** Los enteros se registran sin comillas, las cadenas de texto en comillas simples.

Ejemplo práctico : Ingrese un usuario en su base de datos

Solución: Para esto, debemos tener visualizado a nuestros usuarios.

IdEmail	Nombre	Usuario	Contraseña
marco.28@gmail.com	Marco	marco123	\$138&#login
ale.2918@gmail.com	Alejandra		\$182¿login

En ese caso, la forma de ingresar queda dada por:

```
INSERT INTO Usuarios VALUE(  
    'marco.28@gmail.com',  
    'Marco',  
    'marco123',  
    '$138&#login'  
);
```

Podemos ingresar también datos que no tengan todos los items, por ejemplo el caso 2 no tiene nombre de usuario, se puede colocar de la siguiente forma:

```
• INSERT INTO Usuarios(email,nombre,contraseña) VALUE(  
    'ale.2918@gmail.com',  
    'Alejandra',  
    '$182¿login'  
);
```

Para solicitar los datos que se están registrando, tenemos que utilizar la siguiente solicitud:

**SELECT \* FROM + R**

Al realizar esta solicitud obtenemos, la siguiente tabla creada.

23 • SELECT \* FROM Usuarios;

email	nombre	usuario	contraseña
ale.2918@gmail.com	Alejandra	NULL	\$182¿login
marco.28@gmail.com	Marco	marco123	\$138&#login
NULL	NULL	NULL	NULL

Como nos damos cuenta, en el espacio de usuario aparece el valor de NULL.



Para que en un atributo, no sea posible el valor de nulo, le tenemos que dar una condición particular.

```
• CREATE TABLE Usuarios(  
    email char(100) primary key,  
    nombre varchar(25),  
    usuario varchar(10) NOT NULL,  
    contraseña char(15) UNIQUE  
);
```

**NOT NULL** : Llave que no permite nulos

**UNIQUE** : No permite repeticiones

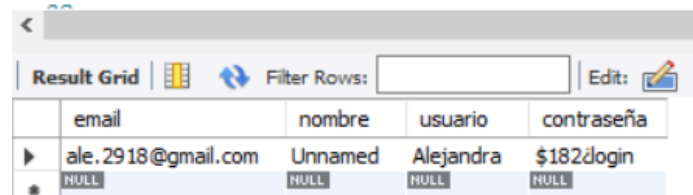
**primary key** : LLave que cumple ambas condiciones.

Algunas **declaraciones** (Constraints) importantes en SQL:

- **DEFAULT** : Agrega un valor por defecto

Ejemplo:

```
4 • CREATE TABLE Usuarios(  
5     email char(100) primary key,  
6     nombre varchar(25) DEFAULT 'Unnamed',  
7     usuario varchar(10) NOT NULL,  
8     contraseña char(15) UNIQUE  
9 );  
  
36 • INSERT INTO Usuarios(email,usuario,contraseña) VALUE(  
37     'ale.2918@gmail.com',  
38     'Alejandra',  
39     '$182¿login'  
40 );  
  
26  
27 • SELECT * FROM Usuarios;
```



	email	nombre	usuario	contraseña
▶	ale.2918@gmail.com	Unnamed	Alejandra	\$182¿login
*	NULL	NULL	NULL	NULL

En este caso, el valor por default para nombre es 'Unnamed', por lo tanto aunque no se coloca usuario en el caso de Alejandra, aparece en la tabla como 'Unnamed'.

- AUTO\_INCREMENT: Aumenta el valor del atributo por uno, aunque no se especifique.

Ejemplo:

```
4 • CREATE TABLE Usuarios(
5     numero_de_orden int AUTO_INCREMENT PRIMARY KEY,
6     email char(100) ,
7     nombre varchar(25) DEFAULT 'Unnamed',
8     usuario varchar(10) NOT NULL,
9     contraseña char(15) UNIQUE
10 );
```

La llave primaria se coloca en auto\_increment. Una aclaración: Solamente las keys pueden tener AUTO\_INCREMENT.

```
21 • INSERT INTO Usuarios (email,nombre, usuario, contraseña) VALUE(
22     'marco.28@gmail.com',
23     'Marco',
24     'marco123',
25     '$138&#login'
26 );
```

Al momento de ingresar el valor de número de orden, este no se registra, y se completará automáticamente.

```
37 • INSERT INTO Usuarios(email,usuario,contraseña) VALUE(
38     'ale.2918@gmail.com',
39     'Alejandra',
40     '$182login'
41 );
42
```

numero_de_orden	email	nombre	usuario	contraseña
1	ale.2918@gmail.com	Unnamed	Alejandra	\$182login
2	marco.28@gmail.com	Marco	marco123	\$138&#login
* NULL	NULL	NULL	NULL	NULL

## Update & Delete

Update y Delete son dos sentencias importantes en el ámbito de actualizar tablas, veamos el siguiente ejemplo

Para el usuario 2 alejandra, hay que modificar su contraseña. En ese caso las peticiones irían de la siguiente manera:

```

43 • UPDATE Usuarios
44   SET contraseña = '$210!login'
45   WHERE usuario = 'Alejandra';

```

La petición que se hace es UPDATE seguido de la tabla a modificar, luego se realiza lo que se desea modificar, en este caso vamos a colocar la contraseña cambiada, para el usuario que se llama Alejandra. También se pueden incluir casos de combinación lógica

Otro ejemplo sería que queremos eliminar el usuario Marco de nuestra base de datos. En este caso sería.

```

47 • DELETE FROM Usuarios where nombre = 'Marco';
48 • SELECT * FROM Usuarios;

```

	numero_de_orden	email	nombre	usuario	contraseña
▶	1	ale.2918@gmail.com	Unnamed	Alejandra	\$210!login
•	NULL	NULL	NULL	NULL	NULL

Solicitudes básicas (Queries)

**Select** : Sirve para seleccionar los datos de acuerdo a una selección de atributos en una tabla. Cuando queremos seleccionar todos los atributos escribimos \* luego del select:

**SELECT \* FROM + R**

Podemos reemplazar el \* por los atributos que queremos analizar. Por ejemplo si queremos los datos de los atributos nombre y usuario, escribimos:

**SELECT + atributos + FROM + R**

**ORDER BY** : Para ordenar de acuerdo un atributo, va luego del select. Tiene los métodos ASC y DESC.

**SELECT \* FROM + R + ORDER BY + ATTRIBUTE**

**LIMIT** : Para obtener un límite de filas de la tabla.

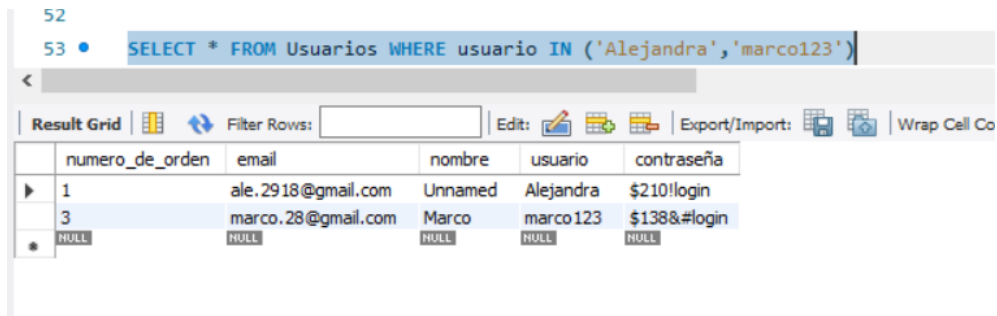
**SELECT \* FROM + R + LIMIT + VALOR**

**WHERE** : Devuelve filas que cumplen con una condición lógica

-- <, >, <=, >=, =, <>, AND, OR

Estas son las proposiciones que pueden cumplir atributos

**IN** : Para colocar varios valores de un atributo. Por ejemplo



The screenshot shows a SQL query editor with a query bar containing the following SQL statement:

```
52  
53 • SELECT * FROM Usuarios WHERE usuario IN ('Alejandra', 'marco123')
```

Below the query bar is a toolbar with options: Result Grid, Filter Rows, Edit, Export/Import, and Wrap Cell Co. Below the toolbar is a table with the following data:

	numero_de_orden	email	nombre	usuario	contraseña
▶	1	ale.2918@gmail.com	Unnamed	Alejandra	\$210!login
	3	marco.28@gmail.com	Marco	marco123	\$138&#login
*	NULL	NULL	NULL	NULL	NULL

**AS** : Permite cambiar el nombre de una columna

**SELECT email as correo From Usuarios**

**DISTINCT** : Permite elegir columnas sin repeticiones

**SELECT DISTINCT Country FROM Customers;**

**Funciones en SQL**

**COUNT()** : Contar los elementos de una columna

```
SELECT COUNT(column_name)  
FROM table_name  
WHERE condition;
```

**AVG()** : Retorna el promedio de una columna

```
SELECT AVG(column_name)  
FROM table_name  
WHERE condition;
```

**SUM()** : Retorna la suma de los valores de una columna

```
SELECT SUM(column_name)  
FROM table_name  
WHERE condition;
```

**GROUP BY** : Retorna el valor de la columna por grupos

```
SELECT column_name(s)  
FROM table_name  
WHERE condition  
GROUP BY column_name(s)
```

Ejemplo de sintaxis:

```
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;
```