Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

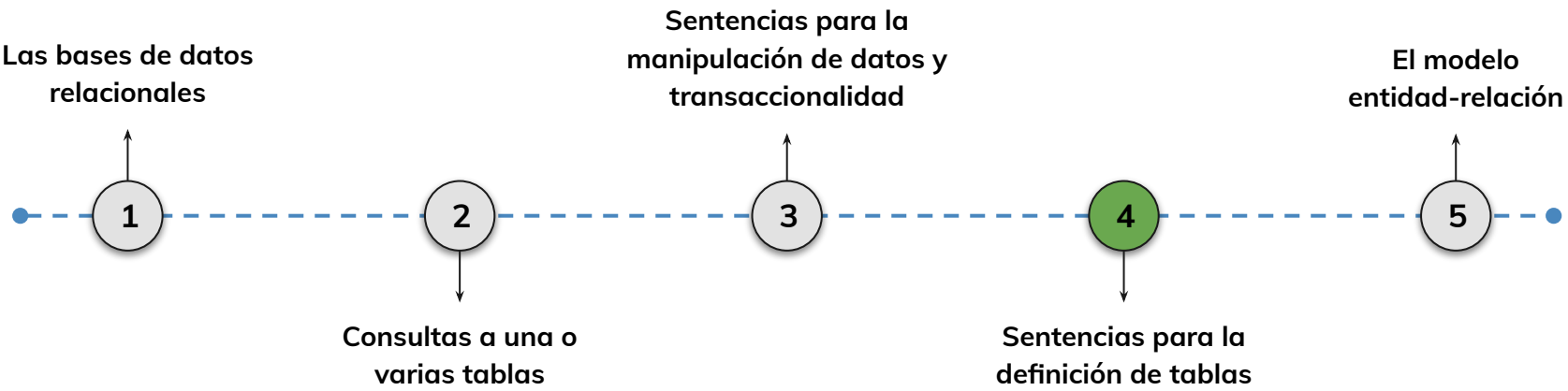
Sentencias para la definición de tablas - Parte 1

AE4: Implementar estructuras de datos relacionales utilizando lenguaje de definición de datos DDL a partir de un modelo de datos para la creación y mantención de las definiciones de los objetos de una base de datos.

alkemy

Roadmap de lecciones

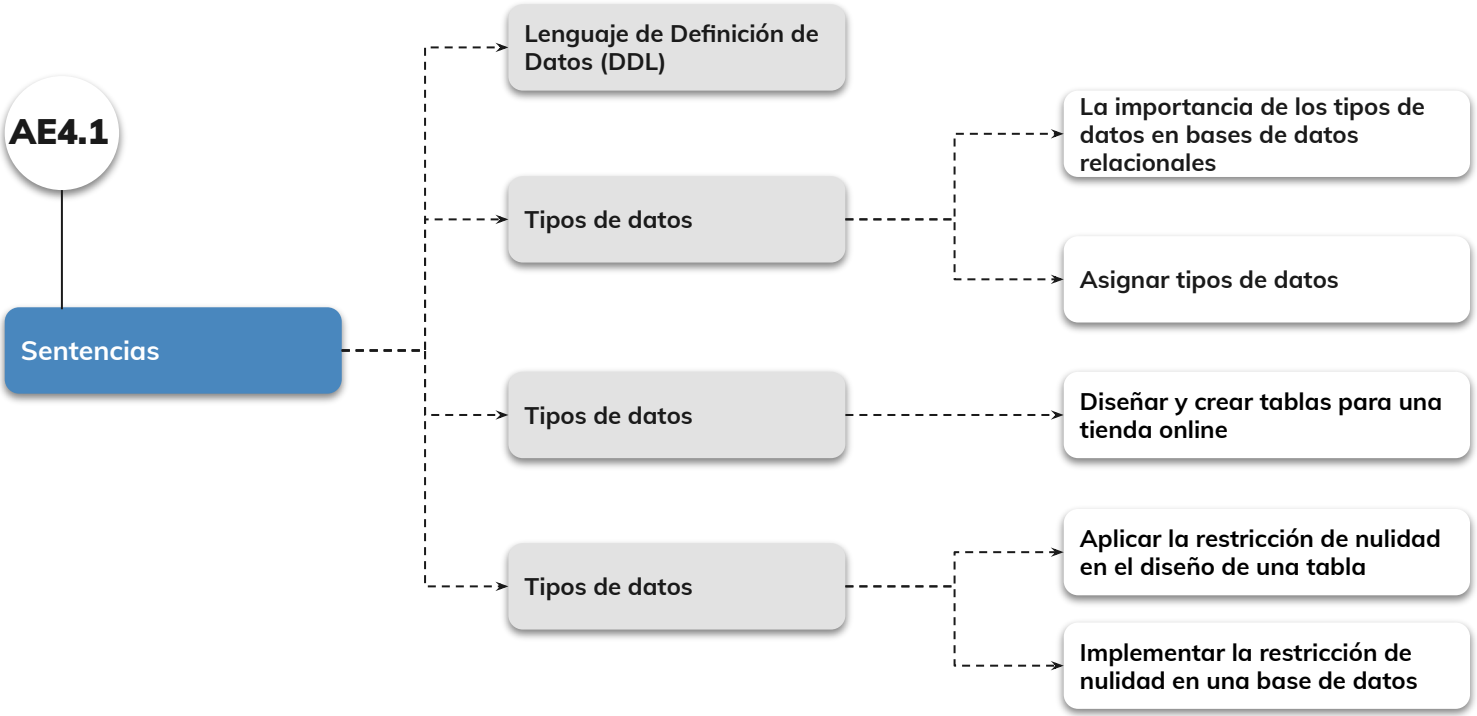
*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?



alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Definir el Lenguaje de Definición de Datos (DDL)
- Diferenciar los distintos tipos de datos que existen
- Definir los tipos de datos en una tabla
- Crear y definir campos en una tabla en una base de datos utilizando DDL
- Conocer el concepto restricción de nulidad
- Implementar la restricción de nulidad

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → Conocer el concepto de integridad referencial
- → Eliminar datos con integridad referencial
- → Actualizar datos con integridad referencial
- → Conocer los principios ACID
- → Conocer qué es la transaccionalidad en las operaciones SQL
- → Realizar transaccionalidades para confirmar y revertir cambios en la base de datos

Lenguaje de Definición de Datos (DDL)

Lenguaje de Definición de Datos (DDL)

¿Qué es DDL y para qué sirve?

El Lenguaje de Definición de Datos (DDL, por sus siglas en inglés: Data Definition Language) es un conjunto de comandos utilizados en sistemas de gestión de bases de datos (DBMS) para definir, modificar y gestionar la estructura y las características de las bases de datos.

El DDL se encarga de describir cómo se deben crear, alterar o eliminar los elementos que componen una base de datos, como **tablas, índices, vistas, restricciones** , etc.

Lenguaje de Definición de Datos (DDL)

Diferencias entre DDL y DML:

DDL y DML son dos componentes fundamentales de SQL y se utilizan para diferentes propósitos en la gestión y manipulación de bases de datos.

Aquí hay algunas diferencias clave entre DDL y DML:

DDL

Permiten la creación, modificación y eliminación de objetos como **tablas, índices, vistas y restricciones.**

Create
Drop
Alter

≠

DML

Permiten insertar, actualizar, eliminar y consultar **registros.**

Select
Insert
Update
Delete

Lenguaje de Definición de Datos (DDL)

El DDL se utiliza principalmente para:

Definir tablas y estructura: Crear nuevas tablas especificando sus columnas, tipos de datos, restricciones (como claves primarias y foráneas) y otras propiedades.

Modificar estructura: Modificación de tablas ya existentes, como añadir, eliminar o modificar columnas, cambiar tipos de datos o agregar restricciones.

Crear y gestionar índices: *Permite crear índices para mejorar el rendimiento de las consultas en la base de datos.*

Establecer restricciones: *Ayuda a definir reglas que los datos deben cumplir, como claves primarias, claves foráneas, restricciones de unicidad, entre otras.*

Lenguaje de Definición de Datos (DDL)

El DDL se utiliza principalmente para:

Crear y gestionar vistas: Posibilita la creación de vistas, que son representaciones lógicas de los datos en una base de datos, y que pueden simplificar la forma en que los usuarios interactúan con los datos.

Definir autorizaciones y permisos: Permite establecer quién puede acceder a qué partes de la base de datos y qué operaciones pueden realizar.

Lenguaje de Definición de Datos (DDL)

Algunas sentencias más comunes son:

- **CREATE TABLE:** Crea una nueva tabla.
- **ALTER TABLE:** Modifica la estructura de una tabla existente.
- **DROP TABLE:** Elimina una tabla.
- **CREATE INDEX:** Crea un nuevo índice.
- **CREATE VIEW:** Crea una nueva vista.

Algunas ejemplos más comunes son:

- **CREATE TABLE** `Empleados (`
  ```
  id_empleado INT PRIMARY KEY,
  nombre      VARCHAR(100),
  salario     DECIMAL(10,2) );
  ```

- **ALTER TABLE** `Empleados`
  ```
  ADD fecha_ingreso DATE;
  ```

- **DROP TABLE** `IF EXISTS`
  ```
  Empleados_backup;
  ```

- **CREATE INDEX** `idx_salario`
  ```
  ON Empleados (salario);
  ```

- **CREATE VIEW**
  ```
  VW_EmpleadosAltosSalarios AS
  SELECT nombre, salario
  FROM   Empleados
  WHERE  salario > 50000;
  ```

Tipos de datos

Tipos de datos

¿Qué son los tipos de datos?

Los tipos de datos definen el formato y el rango de valores que pueden almacenarse en las columnas de una tabla.
Los tipos de datos en bases de datos pueden clasificarse en varias categorías:

- Numéricos
- Caracteres
- Fecha y hora
- Booleanos
- Valores únicos
- Otros tipos especializados

Tipos de datos

TIpos numéricos: Estos tipos de datos se utilizan para almacenar valores numéricos, ya sean enteros o decimales.

- **INTEGER** : Para números enteros
- **FLOAT:** Para números de punto flotante con diferentes niveles de precisión.

Tipos de caracteres y cadenas de texto:

- **CHAR(n):** Cadena de longitud fija con un número específico de caracteres.
- **VARCHAR(n):** Cadena de longitud variable con un número máximo de caracteres.
- **TEXT:** Cadena de longitud variable para texto largo.

Lenguaje de Definición de Datos (DDL)

Tipos de fecha y hora:

- **DATE**: Almacena fechas (año, mes, día).
- **TIME**: Almacena horas del día.
- **TIMESTAMP** : Almacena fecha y hora.
- **DATETIME** : Una variante de TIMESTAMP en algunos sistemas.

Tipos booleanos: *Almacenan valores de verdadero (TRUE) o falso (FALSE).*

- ***BOOLEAN o BOOL:*** *Tipos para valores booleanos.*

Tipos de datos

Tipos binarios:

- **BINARY:** Datos binarios de longitud fija.
- **VARBINARY:** Datos binarios de longitud variable.

Tipos de valores únicos:

- **SERIAL, AUTO_INCREMENT:** Genera automáticamente valores únicos, típicamente utilizados para claves primarias.

Otros tipos especializados:

- **ARRAY:** Almacena una colección de valores del mismo tipo.
- **JSON, XML:** Almacena datos en formato JSON o XML.

LIVE CODING

Ejemplo en vivo

Diversidad de datos para tu wallet virtual.

¿Cómo asignar tipos de datos?

En la tabla llamada "Moneda" con varias columnas que utilicen diferentes tipos de datos para almacenar información sobre las diferentes monedas.

1. Crear una tabla llamada Moneda que contenga las siguientes columnas

```
currency_id, currency_name,

currency_symbol
```

2. Asignar los tipos de datos que correspondan para cada columna

Tiempo: 15 minutos

Ejercicio N° 1

Creación de tablas de Datos para una Librería

Creación de tablas de Datos para una Librería

Contexto: 🙌

Acaban de ver, paso a paso, cómo crear una tabla con la sentencia `CREATE TABLE`. Ahora practicarán dos sentencias DDL muy comunes que complementan la creación inicial: `ALTER TABLE` y la definición de claves foráneas (FK).

Ahora es el momento de poner estos conceptos y términos en práctica.

⏳ **Tiempo:** 20 minutos

Creación de tablas de Datos para una Librería

Consigna: ✍️

1. Crea una tabla llamada `Autores` con las siguientes columnas (tú decides los tipos de datos apropiados):
   - `autor_id` – clave primaria y autoincremental.
   - `nombre` – nombre de pila del autor.
   - `apellido` – apellido del autor.
   - `fecha_nacimiento – fecha de nacimiento.`
   - `nacionalidad` – país de origen.

   Pista: piensa en enteros, cadenas de longitud fija/variable o tipos de fecha según convenga.

2. **Inserta dos registros** de ejemplo en Autores con datos ficticios.
3. **Consulta** el contenido de la tabla con SELECT * FROM Autores; para verificar que todo quedó almacenado correctamente.

Creación y gestión de una base de datos para una Librería

Queries mentor

```
/* 1️⃣  Crear la tabla Autores */
CREATE TABLE Autores (
    autor_id         INT AUTO_INCREMENT PRIMARY KEY,
    nombre           VARCHAR(40)  NOT NULL,
    apellido         VARCHAR(40)  NOT NULL,
    fecha_nacimiento DATE,
    nacionalidad     VARCHAR(40)
);

/* 2️⃣  Insertar datos de ejemplo */
INSERT INTO Autores (nombre, apellido, fecha_nacimiento, nacionalidad)
VALUES
    ('Isabel',  'Allende',          '1942-08-02', 'Chile'),
    ('Gabriel', 'García Márquez',   '1927-03-06', 'Colombia');

/* 3️⃣  Comprobar resultado */
SELECT * FROM Autores;
```

alkemy

Momento:

Time-out!

⏰5 -10 min.

Crear una tabla en una base de datos utilizando DDL

Crear una tabla en una base de datos utilizando DDL

¿Cómo crear una tabla?

Crear una tabla en una base de datos utilizando DDL (Lenguaje de Definición de Datos) en SQL implica especificar la estructura de la tabla, incluyendo los nombres de las columnas, los tipos de datos y las restricciones.

Aquí tienes un ejemplo básico de cómo crear una tabla utilizando DDL:
Supongamos que estamos creando una base de datos para almacenar información sobre libros en una biblioteca.

```
CREATE TABLE Libros (
    id INT PRIMARY KEY,
    titulo VARCHAR(100),
    autor VARCHAR(50),
    ano_publicacion INT,
    disponible BOOLEAN
);
```

Crear una tabla en una base de datos utilizando DDL

Analicemos el ejemplo anterior:

Hemos creado una tabla llamada "Libros" con las siguientes columnas:

- **id:** Un número entero que servirá como clave primaria para identificar cada libro de manera única.
- **titulo:** Una cadena de caracteres de hasta 100 caracteres para almacenar el título del libro.
- **autor:** Una cadena de caracteres de hasta 50 caracteres para almacenar el nombre del autor del libro.
- **ano_publicacion:** Un número entero para almacenar el año de publicación del libro.
- disponible: Una columna booleana que indica si el libro está disponible en la biblioteca.

Crear una tabla en una base de datos utilizando DDL

Declaracion CREATE:

La declaración CREATE TABLE se utiliza para crear una nueva tabla en la base de datos. Puedes agregar restricciones adicionales, como claves foráneas, restricciones de integridad, índices, entre otros, dependiendo de tus necesidades.

En resumen:

La sintaxis pueden variar según el sistema de gestión de bases de datos que estés utilizando.

Recuerda que al crear una tabla, estás definiendo la estructura básica en la que se almacenarán los datos. Es importante elegir los tipos de datos adecuados y diseñar la estructura de manera que sea eficiente para tus necesidades de almacenamiento y consulta.

LIVE CODING

Ejemplo en vivo

Diseñar y crear tablas para una tienda online:

Debes crear tablas para almacenar información sobre productos y categorías de productos, estableciendo relaciones entre ellas.

1. *Crear una tabla llamada "Categorias" con 2 columnas: id_categoria (PK)y nombre_categoria.*
2. *Crear una tabla llamada "Productos" con 5 columnas: id_producto (PF), nombre_producto, precio, id_categoria (FK).*
3. *Insertar algunos datos de ejemplo para probar las relaciones.*

Tiempo: 15 minutos

Restricción de nulidad

Restricción de nulidad

¿Qué es la restricción de nulidad?:

La restricción de nulidad en bases de datos es una característica que se utiliza para especificar si un valor en una columna de una tabla puede ser nulo (ausente) o si debe contener siempre un valor válido.

Esta restricción ayuda a mantener la integridad y la coherencia de los datos en la base de datos, asegurando que se cumplan ciertas reglas en relación con los valores permitidos en las columnas.

Restricción de nulidad

¿Cómo se puede aplicar?

La restricción de nulidad se puede aplicar en dos modos:

1. **NOT NULL (No nulo):** Al aplicar esta restricción a una columna, se garantiza que dicha columna no pueda contener valores nulos. En otras palabras, cada registro en la tabla debe tener un valor válido en esa columna.

Por ejemplo, si aplicamos la restricción NOT NULL a la columna "Nombre" en una tabla de empleados, estaríamos asegurando que todos los registros tengan un nombre.

Restricción de nulidad

¿Cómo se puede aplicar?

La restricción de nulidad se puede aplicar en dos modos:
NULL (Nulo): Si no se aplica la restricción NOT NULL a una columna, entonces por defecto la columna permitirá valores nulos. Esto significa que los registros en la tabla podrán tener un valor en esa columna o simplemente estar vacíos (nulos)

En la mayoría de los casos **se realiza al momento de crear la tabla** o al modificar la estructura de la misma utilizando un comando DDL

Restricción de nulidad

Ejemplo de restricción NOT NULL:

Aquí tienes ejemplos de cómo aplicar la restricción de nulidad en SQL para la creación de una tabla:

```sql
CREATE TABLE Empleados (
    id INT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    salario DECIMAL(10, 2)
);
```

Analicemos el ejemplo:

En este ejemplo, la columna "nombre" tiene la restricción NOT NULL, lo que significa que todos los registros deben tener un valor en esa columna.

Restricción de nulidad

Ejemplo SIN restricción de nulidad (permitiendo valores nulos):

En este ejemplo, las columnas "nombre" y "telefono" no tienen restricciones NOT NULL, lo que permite que los registros en la tabla puedan tener valores nulos en esas columnas.

```
CREATE TABLE Clientes (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    telefono VARCHAR(20)
);
```

LIVE CODING

Ejemplo en vivo

Diseñar y crear tablas para una tienda online:

Debes crear tablas para almacenar información sobre productos y categorías de productos, estableciendo relaciones entre ellas.

1. *Crea la tabla Productos*
2. *Inserta un registro válido (con nombre)*
3. *Prueba la restricción (inserción inválida)*

Tiempo: 15 minutos

Ejercicio N° 2

Creación y Gestión de una Base de Datos para una Librería

Creación y gestión de una base de datos para una Librería

Contexto: 🙌

Una librería llamada *"Lectura Global"* quiere organizar sus datos en una base de datos relacional. Necesitan gestionar los libros, sus autores, las categorías a las que pertenecen y las ventas realizadas.

Tu tarea es diseñar el esquema de la base de datos mediante comandos **DDL** en **SQL**, asegurando restricciones adecuadas y relaciones entre las tablas.

Creación y gestión de una base de datos para una Librería

✍️ Consigna:

Crea una base de datos llamada **Librería** con las siguientes tablas:

1. **Autores**:
   - autor_id (INT, clave primaria, autoincremental)
   - nombre (VARCHAR(100), no nulo)
   - nacionalidad (VARCHAR(50))
2. **Categorias**:
   - categoria_id (INT, clave primaria, autoincremental)
   - nombre (VARCHAR(50), no nulo)
3. **Libros**:
   - libro_id (INT, clave primaria, autoincremental)
   - titulo (VARCHAR(150), no nulo)
   - autor_id (INT, clave foránea que referencia Autores(autor_id))
   - categoria_id (INT, clave foránea que referencia Categorias(categoria_id))
   - precio (DECIMAL(8,2), no nulo)
   - stock (INT, no nulo, con valor por defecto 10)
4. **Ventas**:
   - venta_id (INT, clave primaria, autoincremental)
   - libro_id (INT, clave foránea que referencia Libros(libro_id))
   - cantidad (INT, no nulo, mayor a 0)
   - fecha_venta (DATE, no nulo, valor predeterminado la fecha actual)

Creación y gestión de una base de datos para una Librería

⚙️ **Paso a paso:**

1. **Crea la base de datos** llamada `Libreria`.
2. **Crea las tablas** con las claves primarias y foráneas mencionadas.
3. **Aplica restricciones** como:
    - No permitir valores nulos en campos clave.
    - Definir valores predeterminados en `stock` y `fecha_venta`.
4. **Inserta algunos datos** de prueba en cada tabla.
5. **Verifica la estructura** consultando las tablas.

⏳ **Tiempo:** 40 minutos

Creación y gestión de una base de datos para una Librería

Queries mentor

1. Creación de la Base de Datos

```
CREATE DATABASE Libreria;
USE Libreria;
```

2. Creación de las Tablas

```
-- Tabla de Autores
CREATE TABLE Autores (
    autor_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    nacionalidad VARCHAR(50)
);

-- Tabla de Categorías
CREATE TABLE Categorias (
    categoria_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL
);

-- Tabla de Libros
CREATE TABLE Libros (
    libro_id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(150) NOT NULL,
    autor_id INT NOT NULL,
    categoria_id INT NOT NULL,
    precio DECIMAL(8,2) NOT NULL,
    stock INT NOT NULL DEFAULT 10,
    FOREIGN KEY (autor_id) REFERENCES Autores(autor_id) ON DELETE CASCADE,
    FOREIGN KEY (categoria_id) REFERENCES Categorias(categoria_id) ON DELETE CASCADE
);

-- Tabla de Ventas
CREATE TABLE Ventas (
    venta_id INT AUTO_INCREMENT PRIMARY KEY,
    libro_id INT NOT NULL,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    fecha_venta DATE NOT NULL DEFAULT (CURRENT_DATE),
    FOREIGN KEY (libro_id) REFERENCES Libros(libro_id) ON DELETE CASCADE
);
```

alkemy

Creación y gestión de una base de datos para una Librería

Queries mentor

3. Inserción de Datos de Prueba

```
-- Insertar Autores
INSERT INTO Autores (nombre, nacionalidad) VALUES
('Gabriel García Márquez', 'Colombiano'),
('Isabel Allende', 'Chilena'),
('J.K. Rowling', 'Británica');

-- Insertar Categorías
INSERT INTO Categorias (nombre) VALUES
('Novela'),
('Ciencia Ficción'),
('Fantasía');

-- Insertar Libros
INSERT INTO Libros (titulo, autor_id, categoria_id, precio, stock) VALUES
('Cien Años de Soledad', 1, 1, 25.99, 20),
('La Casa de los Espíritus', 2, 1, 19.99, 15),
('Harry Potter y la Piedra Filosofal', 3, 3, 29.99, 30);

-- Insertar Ventas
INSERT INTO Ventas (libro_id, cantidad) VALUES
(1, 2),
(3, 1),
(2, 4);
```

4. Verificación de la Base de Datos

```
-- Ver todas las tablas
SHOW TABLES;

-- Ver datos de cada tabla
SELECT * FROM Autores;
SELECT * FROM Categorias;
SELECT * FROM Libros;
SELECT * FROM Ventas;
```

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ *Comprender el el Lenguaje de Definición de Datos (DDL)*
- ✓ *Diferenciar los distintos tipos de datos que existen*
- ✓ *Crear tablas y columnas con diferentes tipos de datos según el requerimiento*
- ✓ *Crear una tabla en una base de datos utilizando DDL*
- ✓ *Implementar el método CREATE TABLE*
- ✓ *Diferenciar los dos modos de restricción de nulidad: NOT NULL y NULL*

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compártela o tráela al próximo encuentro sincrónico.

alkemy

#Checkout

¿Qué les pareció la clase de hoy?

3. Estuvo bien, pero podría mejorar
¿Qué se puede potenciar?

4. ¡Estuvo bien!
¿Qué me llevo?

2. No me convenció mucho
¿Qué no te gustó de la clase?

5. ¡Me encantó!
¿Qué fue lo mejor de la clase?

1. Red flag
¿Qué te hubiera gustado que fuera diferente?

¡Muchas gracias!

alkemy