**TECNICATURA EN PROGRAMACIÓN**

**Base de datos 2**

🔹 **PRÁCTICA GUIADA: Integridad y Consistencia de Datos en SQL** 📌 **Objetivo:**

Aprender a aplicar restricciones (Constraints) para garantizar la integridad y consistencia de los datos en una base de datos.

📌 **Temas cubiertos:**

✅ Claves Primarias y Foráneas

✅ Restricciones UNIQUE, NOT NULL, CHECK, DEFAULT

✅ Uso de TRIGGERS para mantener la integridad

🔹 **ESCENARIO: Sistema de Gestión de Ventas**

En este ejercicio, vamos a diseñar un sistema de base de datos para una tienda que maneja **Clientes**, **Pedidos** y **Productos**.

🔹 **RELEVAMIENTO INICIAL:**

A continuación, se presenta una narrativa que describe el sistema. Con base en lo aprendido sobre integridad y consistencia de datos, los alumnos deberán identificar los campos necesarios para cada tabla y definir las características de cada campo (tipo de dato, restricciones, etc.).

**Narrativa:**

1. **Clientes:** 
   1. Cada cliente debe tener una forma única de ser identificado en el sistema.

○ El nombre del cliente es un dato obligatorio.

○ El correo electrónico es un dato importante que debe cumplir ciertas condiciones para evitar duplicados.

1. **Productos:** 
   1. Cada producto debe tener una forma única de ser identificado en el sistema.

○ El nombre del producto es un dato obligatorio.

○ El precio del producto debe cumplir ciertas condiciones para evitar valores inválidos.

○ El stock del producto debe tener un valor predeterminado si no se especifica.

1. **Pedidos:** 
   1. Cada pedido debe tener una forma única de ser identificado en el sistema.

○ Un pedido está asociado a un cliente y a un producto.

○ La cantidad de productos pedidos debe cumplir ciertas condiciones para evitar valores inválidos.

○ La fecha del pedido es un dato obligatorio.

○ Si un cliente se elimina, los pedidos asociados deben manejarse de alguna forma específica.

○ Si un producto se elimina, los pedidos asociados deben manejarse de alguna forma específica.

○ No se debe permitir hacer un pedido si no hay suficiente stock del producto.

🔹 **PASO 1: Crear la tabla Clientes**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
|  |
| CREATE TABLE Clientes (  id\_cliente INT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(50) NOT NULL,  correo\_electronico VARCHAR(100) NOT NULL UNIQUE,  CONSTRAINT chk\_correo CHECK (correo\_electronico REGEXP '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')  ); |
|  |
|  |

🔹 **PASO 2: Crear la tabla Productos**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
|  |
| CREATE TABLE Productos (  id\_producto INT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(50) NOT NULL,  precio DECIMAL(10, 2) NOT NULL,  stock INT DEFAULT 0,  CONSTRAINT chk\_precio\_positivo CHECK (precio > 0),  CONSTRAINT chk\_stock\_no\_negativo CHECK (stock >= 0)  ); |
|  |
|  |

🔹 **PASO 3: Crear la tabla Pedidos con Claves Foráneas**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
|  |
| CREATE TABLE Pedidos (  id\_pedido INT AUTO\_INCREMENT PRIMARY KEY,  id\_cliente INT NOT NULL,  id\_producto INT NOT NULL,  cantidad INT NOT NULL CHECK (cantidad > 0),  fecha DATE NOT NULL,  CONSTRAINT fk\_pedido\_cliente FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente) ON DELETE CASCADE,  CONSTRAINT fk\_pedido\_producto FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto) ON DELETE CASCADE  ); |
|  |

🔹 **PASO 4: Agregar un TRIGGER para Mantener la Integridad**

Con base en la narrativa, los alumnos deben definir un trigger que evite hacer pedidos si no hay suficiente stock.

|  |
| --- |
| DELIMITER //  CREATE TRIGGER valida\_stock\_pedido  BEFORE INSERT ON Pedidos  FOR EACH ROW  BEGIN  DECLARE stock\_disponible INT;    SELECT stock INTO stock\_disponible  FROM Productos  WHERE id\_producto = NEW.id\_producto;    IF NEW.cantidad > stock\_disponible THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'No hay suficiente stock para realizar el pedido';  END IF;  END;  //  DELIMITER ; |
|  |
|  |
|  |
|  |
|  |

🔹 **PRUEBAS: Insertar Datos y Verificar Integridad**

Los alumnos deben insertar datos de prueba y verificar que las restricciones y el trigger funcionen correctamente. ✅ **Insertar Clientes**

-- insert into Clientes(nombre, correo\_electronico)

values ('pepe', 'pepe@hotmail.com');

✅ **Insertar Productos**

-- insert into Productos(nombre, precio, stock)

values ('azucar', '1000', 10);

❌ **Prueba de Restricciones**

insert into Productos(nombre, precio, stock)

values ('azucar', '1000', -10);

❌ **Prueba de TRIGGER**

**--insert into Pedidos(id\_cliente, id\_producto, cantidad, fecha)**

**values (1, 1 , 11, now());**

✅ **Insertar Pedido Correcto**

-- insert into Pedidos(id\_cliente, id\_producto, cantidad, fecha)

values (1, 1 , 9, now());

🔹 **PRÁCTICA: Gestión de Biblioteca con Restricciones y Triggers** 📌 **Objetivo:**

Crear una base de datos para gestionar una biblioteca utilizando restricciones (constraints) y triggers para garantizar la integridad y consistencia de los datos.

📌 **Requisitos de la práctica:**

* Crear tablas: **Libros**, **Clientes**, **Préstamos**.
* Aplicar restricciones: claves primarias, foráneas, CHECK, NOT NULL, UNIQUE, DEFAULT.
* Crear un TRIGGER: Para evitar que se preste un libro si no hay stock suficiente.

🔹 **DESCRIPCIÓN DEL SISTEMA:**

Se va a gestionar la información sobre:

* **Clientes:** Personas que se registran en la biblioteca.
* **Libros:** Títulos disponibles para préstamo.
* **Préstamos:** Información sobre los libros prestados a los clientes.

🔹 **RELEVAMIENTO INICIAL:**

A continuación, se presenta una narrativa que describe el sistema. Con base en lo aprendido sobre integridad y consistencia de datos, los alumnos deberán identificar los campos necesarios para cada tabla y definir las características de cada campo (tipo de dato, restricciones, etc.).

**Narrativa:**

1. **Clientes:** 
   1. Cada cliente debe tener una forma única de ser identificado en el sistema.

○ El nombre del cliente es un dato obligatorio.

○ El teléfono y el correo electrónico son datos importantes que deben cumplir ciertas condiciones para evitar duplicados o valores inválidos.

1. **Libros:** 
   1. Cada libro debe tener una forma única de ser identificado en el sistema.

○ El título y el autor son datos obligatorios.

○ El stock del libro debe cumplir ciertas condiciones para evitar valores inválidos.

○ El precio del libro debe cumplir ciertas condiciones para evitar valores inválidos.

1. **Préstamos:** 
   1. Cada préstamo debe tener una forma única de ser identificado en el sistema.

○ Un préstamo está asociado a un cliente y a un libro.

○ La fecha del préstamo es un dato obligatorio.

○ Si un cliente se elimina, los préstamos asociados deben manejarse de alguna forma específica.

○ Si un libro se elimina, los préstamos asociados deben manejarse de alguna forma específica.

○ No se debe permitir hacer un préstamo si no hay suficiente stock del libro.

🔹 **PASO 1: Crear la tabla Clientes**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
|  |
| CREATE TABLE Clientes (  id\_cliente INT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(50) NOT NULL,  telefono VARCHAR(20) NOT NULL,  correo\_electronico VARCHAR(100) NOT NULL UNIQUE,  CONSTRAINT chk\_correo CHECK (correo\_electronico REGEXP '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),  CONSTRAINT chk\_telefono CHECK (telefono REGEXP '^[0-9]{6,15}$')  ); |
|  |

🔹 **PASO 2: Crear la tabla Libros**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

CREATE TABLE Libros (

id\_libro INT AUTO\_INCREMENT PRIMARY KEY,

titulo VARCHAR(100) NOT NULL,

autor VARCHAR(100) NOT NULL,

stock INT NOT NULL,

precio DECIMAL(10,2) NOT NULL,

CONSTRAINT chk\_stock CHECK (stock >= 0),

CONSTRAINT chk\_precio CHECK (precio >= 0)

);

🔹 **PASO 3: Crear la tabla Préstamos con Claves Foráneas**

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
| CREATE TABLE Prestamos (  -- Definir los campos y restricciones aquí |
| ); |

🔹 **PASO 4: Agregar un TRIGGER para Validar el Stock**

Con base en la narrativa, los alumnos deben definir un trigger que evite hacer préstamos si no hay suficiente stock.

|  |
| --- |
| CREATE TRIGGER validar\_stock |
| BEFORE INSERT ON Prestamos  FOR EACH ROW |
| BEGIN  -- Definir la lógica del trigger aquí |
| END; |

🔹 **PRUEBAS: Insertar Datos y Verificar Integridad**

Los alumnos deben insertar datos de prueba y verificar que las restricciones y el trigger funcionen correctamente.

✅ **Insertar Clientes**

-- Ejemplo de inserción de clientes

✅ **Insertar Libros**

-- Ejemplo de inserción de libros

❌ **Prueba de Restricciones**

-- Intentar insertar datos que violen las restricciones

❌ **Prueba de TRIGGER**

-- Intentar hacer un préstamo sin suficiente stock

✅ **Insertar Préstamo Correcto**

-- Ejemplo de inserción de un préstamo válido

📌 **PUNTOS A VERIFICAR:**

* **Restricciones:** Asegúrate de que las restricciones como NOT NULL, UNIQUE, CHECK estén funcionando correctamente.
* **Trigger:** Verifica que el trigger impida la inserción de préstamos cuando no hay stock disponible.
* **Integridad referencial:** Comprueba que las claves foráneas entre las tablas Clientes y Libros funcionen correctamente.

De esta manera, la narrativa guía a los alumnos sin darles todas las respuestas, lo que les permite pensar y aplicar los conceptos de integridad y consistencia de datos que han aprendido.

# 🔹 PRÁCTICA: Sistema de Reservas de Hotel con Restricciones y Triggers

📌 **Objetivo:**

Crear una base de datos para gestionar las reservas de un hotel, utilizando restricciones (constraints) y triggers para garantizar la integridad y consistencia de los datos.

📌 **Requisitos de la práctica:**

* Crear tablas: **Habitaciones**, **Clientes**, **Reservas**.
* Aplicar restricciones: claves primarias, foráneas, CHECK, NOT NULL, UNIQUE, DEFAULT.
* Crear un TRIGGER: Para evitar que se reserve una habitación en fechas no disponibles.

🔹 **DESCRIPCIÓN DEL SISTEMA:**

Se va a gestionar la información sobre:

* **Habitaciones:** Las habitaciones disponibles en el hotel, con sus características.
* **Clientes:** Las personas que realizan reservas en el hotel.
* **Reservas:** Las reservas realizadas por los clientes, incluyendo fechas de check-in y check-out.

🔹 **RELEVAMIENTO INICIAL:**

A continuación, se presenta una narrativa que describe el sistema. Con base en lo aprendido sobre integridad y consistencia de datos, los alumnos deberán identificar los campos necesarios para cada tabla y definir las características de cada campo (tipo de dato, restricciones, etc.).

**Narrativa:**

1. **Habitaciones:** 
   1. Cada habitación tiene un número único que la identifica.

○ El tipo de habitación (individual, doble, suite) es un dato obligatorio.

○ El precio por noche debe ser mayor a 0.

○ La habitación puede estar disponible o no, dependiendo de las reservas.

1. **Clientes:** 
   1. Cada cliente tiene una forma única de ser identificado en el sistema.

○ El nombre y el correo electrónico son datos obligatorios. ○ El correo electrónico debe ser único para cada cliente.

1. **Reservas:** 
   1. Cada reserva tiene un identificador único.

○ Una reserva está asociada a un cliente y a una habitación.

○ Las fechas de check-in y check-out son obligatorias.

○ No se puede reservar una habitación si ya está ocupada en las fechas solicitadas.

○ Si un cliente se elimina, sus reservas también deben eliminarse.

○ Si una habitación se elimina, las reservas asociadas deben manejarse de alguna forma específica.

# 🔹 PASO 1: Crear la tabla Habitaciones

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
| CREATE TABLE Habitaciones ( |
| -- Definir los campos y restricciones aquí |
| ); |
|  |

# 🔹 PASO 2: Crear la tabla Clientes

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
| CREATE TABLE Clientes (  -- Definir los campos y restricciones aquí |
| ); |

# 🔹 PASO 3: Crear la tabla Reservas con Claves Foráneas

Con base en la narrativa, los alumnos deben definir los campos y sus características, aplicando las restricciones necesarias para garantizar la integridad de los datos.

|  |
| --- |
| CREATE TABLE Reservas ( |
| -- Definir los campos y restricciones aquí |
| ); |

# 🔹 PASO 4: Agregar un TRIGGER para Validar la Disponibilidad de la Habitación

Con base en la narrativa, los alumnos deben definir un trigger que evite reservar una habitación si ya está ocupada en las fechas solicitadas.

|  |
| --- |
| CREATE TRIGGER validar\_disponibilidad |
| BEFORE INSERT ON Reservas |
| FOR EACH ROW |
| BEGIN  -- Definir la lógica del trigger aquí |
| END; |

# 🔹 PRUEBAS: Insertar Datos y Verificar Integridad

Los alumnos deben insertar datos de prueba y verificar que las restricciones y el trigger funcionen correctamente. ✅ **Insertar Habitaciones**

-- Ejemplo de inserción de habitaciones

✅ **Insertar Clientes**

-- Ejemplo de inserción de clientes

❌ **Prueba de Restricciones**

-- Intentar insertar datos que violen las restricciones

❌ **Prueba de TRIGGER**

-- Intentar hacer una reserva en fechas no disponibles

✅ **Insertar Reserva Correcta**

-- Ejemplo de inserción de una reserva válida

📌 **PUNTOS A VERIFICAR:**

* **Restricciones:** Asegúrate de que las restricciones como NOT NULL, UNIQUE, CHECK estén funcionando correctamente.
* **Trigger:** Verifica que el trigger impida la inserción de reservas cuando la habitación no está disponible en las fechas solicitadas.
* **Integridad referencial:** Comprueba que las claves foráneas entre las tablas Clientes y Habitaciones funcionen correctamente.

📌 **CONCLUSIÓN:**

✅ Aplicar restricciones (Constraints) para garantizar que los datos sean correctos.

✅ Usar claves foráneas para mantener relaciones entre tablas.

✅ Implementar un trigger para evitar inconsistencias en la disponibilidad de habitaciones.

**Resumen: ¿Cuál usar?**

|  |  |  |
| --- | --- | --- |
| **Característica** | **CHECK** | **TRIGGER** |
| **Nivel de aplicación** | Fila (columna específica) | Fila o conjunto de filas |
| **Validación simple** | ✅ Sí | ❌ No siempre |
| **Validación compleja** | ❌ No | ✅ Sí |
| **Acciones adicionales** | ❌ No | ✅ Sí (puede modificar otras tablas o lanzar errores personalizados) |
| **Impacto en el rendimiento** | Bajo | Puede ser alto (depende de la lógica) |

**Regla general:**

* **Si la validación es simple y solo afecta una fila → Usa CHECK.**
* **Si la validación es más compleja o involucra otras tablas o múltiples filas → Usa TRIGGER.**

💡 **TIP:** Usa CHECK siempre que sea posible, ya que es más eficiente que un TRIGGER.

Solo usa TRIGGER cuando CHECK no sea suficiente.