Introducción a las bases de datos

RELACIONES

En SQL, se pueden relacionar tablas mediante la creación de una clave foránea que haga referencia a una clave primaria en otra tabla. Esto se conoce como "integridad referencial" y es una parte importante del diseño de bases de datos relacionales Para relacionar tablas en SQL, se deben seguir estos pasos:

1. Crear la tabla primaria: Primero, se debe crear la tabla principal que contendrá la clave primaria que se utilizará para la relación. La clave primaria es una columna o conjunto de columnas que identifican de forma única cada fila en la tabla.
2. Crear la tabla secundaria: A continuación, se debe crear la tabla secundaria que contendrá la clave foránea que se utilizará para establecer la relación con la tabla primaria.
3. Definir la clave primaria: En la tabla primaria, se debe definir la clave primaria utilizando la cláusula "PRIMARY KEY". Esta clave primaria se utilizará como referencia para la clave foránea en la tabla secundaria.
4. Definir la clave foránea: En la tabla secundaria, se debe definir la clave foránea utilizando la cláusula "FOREIGN KEY". Esta clave foránea debe hacer referencia a la clave primaria en la tabla primaria.
5. Establecer las reglas de integridad referencial: Se pueden establecer reglas adicionales para la integridad referencial utilizando las cláusulas "ON DELETE" y "ON UPDATE". Estas cláusulas especifican qué debe ocurrir si se intenta eliminar o actualizar una fila en la tabla primaria que tiene filas relacionadas en la tabla secundaria.

EJEMPLO:

```
CREATE TABLE usuarios (
  id INT PRIMARY KEY,
  nombre VARCHAR(50),
  direccion VARCHAR(100),
  email VARCHAR(100)
);

CREATE TABLE pedidos (
  num_pedido INT PRIMARY KEY,
  fecha_pedido DATE,
  id_usuario INT,
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id) ON DELETE CASCADE ON UPDATE
CASCADE)
```

En este ejemplo, la tabla "usuarios" tiene una clave primaria "id" y la tabla "pedidos" tiene una clave foránea "id_usuario" que hace referencia a la clave primaria en la tabla "usuarios". Además, se ha especificado que si se elimina o actualiza una fila en la tabla "usuarios" que tiene filas relacionadas en la tabla "pedidos", estas filas relacionadas también se eliminarán o actualizarán en consecuencia.

Constraints (restricciones)

Los constraints (restricciones) son reglas que se pueden aplicar a las tablas de una base de datos con el fin de asegurar la integridad de los datos y mantener la coherencia de la información almacenada. Estas restricciones pueden ser aplicadas a los campos individuales de una tabla o a la tabla en su conjunto, y pueden ser utilizadas para controlar qué datos se pueden insertar, actualizar o eliminar de una tabla.

Existen diferentes tipos de constraints que se pueden utilizar en una base de datos, como:

- **Primary key constraint**: Se utiliza para identificar de manera única cada fila en una tabla. Cada tabla debe tener una primary key constraint.
- **Foreign key constraint**: Se utiliza para asegurar que los datos en una tabla estén relacionados con los datos en otra tabla. La foreign key constraint hace referencia a la primary key de la tabla relacionada.
- **Unique constraint**: Se utiliza para asegurar que los valores de un campo o conjunto de campos sean únicos dentro de una tabla.
- **Check constraint**: Se utiliza para asegurar que los valores de un campo cumplan con una condición especificada.
- **Not null constraint**: Se utiliza para asegurar que un campo no tenga valores nulos.

Al utilizar constraints en una base de datos, se pueden evitar errores y problemas de integridad de datos que podrían surgir si se permitiera la inserción de datos no válidos o contradictorios en las tablas. Además, al garantizar que los datos se almacenen de manera coherente y precisa, se puede mejorar la calidad de la información almacenada y la eficacia de las operaciones que se realizan sobre ella.

- Las restricciones de **nivel 1** son aquellas que pueden ser verificadas al momento de la inserción de un registro en una tabla. Estas restricciones se aplican en el mismo nivel de la fila en el que se está insertando el registro. Las restricciones de nivel 1 incluyen las restricciones NOT NULL, PRIMARY KEY, UNIQUE y FOREIGN KEY. Son importantes porque garantizan que los datos que se inserten en una tabla sean válidos y estén en línea con las reglas de integridad de la base de datos
- Las restricciones de **nivel 2** son aquellas que se verifican después de que se ha insertado un registro en una tabla. Estas restricciones se aplican a nivel de la tabla completa y no a nivel de fila. Las restricciones de nivel 2 incluyen las restricciones CHECK y los TRIGGERS. Son importantes porque permiten verificar reglas más complejas que no se pueden verificar a nivel de fila y que pueden requerir el acceso a múltiples registros en una tabla

En general, se recomienda utilizar restricciones de nivel 1 siempre que sea posible, ya que pueden detectar y evitar errores de manera temprana en el proceso de inserción de datos. Las restricciones de nivel 2 deben utilizarse con cuidado y solo cuando sea necesario para garantizar la integridad de los datos almacenados.

En las bases de datos, las restricciones se pueden aplicar a nivel de columna y de fila para garantizar la precisión y consistencia de los datos almacenados. A continuación, se explican los tipos de restricciones a nivel de columna y de fila en las bases de datos, junto con algunos ejemplos:

Restricciones a nivel de columna

Las restricciones a nivel de columna se aplican a una sola columna y garantizan que los valores almacenados en esa columna sean precisos y consistentes. Los siguientes son algunos ejemplos de restricciones a nivel de columna:

- Restricción NOT NULL: Una restricción NOT NULL se utiliza para garantizar que una columna no tenga valores nulos (vacíos). Esto significa que la columna debe contener un valor en cada registro. Ejemplo: En una tabla de empleados, la columna de número de identificación del empleado puede tener una restricción NOT NULL para garantizar que cada empleado tenga un número de identificación único.

```
CREATE TABLE employees (
   id INT NOT NULL,
   name VARCHAR(50),
   age INT
);
```

- Restricción UNIQUE: Una restricción UNIQUE se utiliza para garantizar que los valores en una columna sean únicos. Esto significa que no puede haber dos registros en una tabla con el mismo valor en la columna. Ejemplo: En una tabla de productos, la columna de nombre del producto puede tener una restricción UNIQUE para garantizar que no haya dos productos con el mismo nombre.

```
CREATE TABLE products (
   id INT PRIMARY KEY,
   name VARCHAR(50) UNIQUE,
   price DECIMAL(10, 2)
);
```

Restricciones a nivel de fila

Las restricciones a nivel de fila se aplican a cada registro y garantizan que los valores almacenados en cada registro sean precisos y consistentes. Los siguientes son algunos ejemplos de restricciones a nivel de fila:

- Restricción **CHECK**: Una restricción CHECK se utiliza para garantizar que los valores en una fila cumplan con una condición específica. Esto significa que una fila solo se puede insertar si cumple con la condición especificada. Ejemplo: En una tabla de clientes, la columna de edad del cliente puede tener una restricción CHECK que solo permita la inserción de filas si la edad del cliente es mayor o igual a 18 años.

```
CREATE TABLE customers (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   age INT CHECK (age >= 18)
);
```

- Restricción **DEFAULT**: Una restricción DEFAULT se utiliza para establecer un valor predeterminado para una columna en caso de que no se proporcione un valor para esa columna durante la inserción. Esto significa que si un valor no se especifica durante la inserción de una fila, se utilizará el valor predeterminado especificado. Ejemplo: En una tabla de pedidos, la columna de fecha de pedido puede tener una restricción DEFAULT que establezca la fecha de pedido en la fecha actual si no se especifica una fecha durante la inserción.

```
CREATE TABLE orders (
   id INT PRIMARY KEY,
   order_date DATE DEFAULT CURRENT_DATE,
   customer_id INT,
   total_amount DECIMAL(10, 2)
);
```

En resumen, las restricciones a nivel de columna se aplican a una sola columna y garantizan que los valores almacenados en esa columna sean precisos y consistentes. Las restricciones a nivel de fila se aplican a cada registro y garantizan que los valores almacenados en cada registro sean precisos y consistentes. Ambos tipos de restricciones se utilizan para garantizar la integridad y consistencia de los datos en una base de datos.

INTEGRIDAD REFERENCIAL

La integridad referencial es un conjunto de reglas y restricciones que garantizan que las relaciones entre las tablas en una base de datos sean precisas y consistentes. En términos simples, la integridad referencial garantiza que los datos relacionados entre tablas sean precisos y coherentes, y que las relaciones entre las tablas se mantengan en todo momento.

En una base de datos relacional, la integridad referencial se logra mediante el uso de claves foráneas (foreign keys), que son columnas en una tabla que hacen referencia a las claves primarias (primary keys) de otra tabla. Al crear una clave foránea en una tabla, se establece una relación entre esa tabla y la tabla referenciada. La clave foránea se utiliza para garantizar que los datos insertados en la tabla sean coherentes con los datos de la tabla referenciada.

La integridad referencial se puede implementar de varias maneras, pero en general, implica las siguientes reglas:

1. Una clave foránea solo puede hacer referencia a una clave primaria en otra tabla.
2. Los valores en la columna de clave foránea deben existir en la columna de clave primaria en la tabla referenciada.
3. Si se borra o actualiza un registro en la tabla referenciada, los registros correspondientes en la tabla con la clave foránea deben ser actualizados o eliminados en consecuencia.

La integridad referencial es importante para garantizar que los datos en una base de datos sean precisos y consistentes en todo momento. Si no se implementa correctamente la integridad referencial, pueden surgir problemas como datos duplicados, datos faltantes, inconsistencias en los datos y errores en las consultas.

En resumen, la integridad referencial es un conjunto de reglas y restricciones que garantizan la precisión y coherencia de los datos en una base de datos relacional. Se implementa mediante el uso de claves foráneas y garantiza que las relaciones entre las tablas sean precisas y consistentes en todo momento.

EJEMPLO INTEGRIDAD REFERENCIAL

Considera dos tablas "orders" y "customers" con las siguientes definiciones:

```
CREATE TABLE customers (
  customer_id INT PRIMARY KEY,
  customer_name VARCHAR(50)
);

CREATE TABLE orders (
  order_id INT PRIMARY KEY,
  customer_id INT,
  order_date DATE,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

La tabla "orders" tiene una clave ajena que hace referencia a la columna "customer_id" de la tabla "customers". Esto asegura que cada pedido en la tabla "orders" esté relacionado con un cliente existente en la tabla "customers".

Para insertar un nuevo pedido, primero necesitamos asegurarnos de que el cliente exista en la tabla "customers": `INSERT INTO customers (customer_id, customer_name) VALUES (1, 'John Doe');`

Luego, podemos insertar el pedido para ese cliente: `INSERT INTO orders (order_id, customer_id, order_date) VALUES (1001, 1, '2023-03-14');`

Si intentamos insertar un pedido para un cliente que no existe en la tabla "customers", recibiremos un error de integridad referencial: `INSERT INTO orders (order_id, customer_id, order_date) VALUES (1002, 2, '2023-03-14');`

Este es el mensaje de error que recibiríamos: `ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`database_name`.`orders`, CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`))`

FALTAN COSAS REVISA LOS EJERCICIOS

MODIFICAR ESTRUCTURA DE UNA TABLA

Con ALTER, puedes agregar, modificar o eliminar columnas, cambiar el nombre de una tabla o cambiar la definición de una columna, entre otras cosas

1. Agregar una columna a una tabla existente `ALTER TABLE tabla_nombre ADD columna_nombre tipo_dato;`
2. Eliminar una columna de una tabla existente `ALTER TABLE tabla_nombre DROP COLUMN columna_nombre;`
3. Modificar el tipo de datos de una columna en una tabla existente `ALTER TABLE tabla_nombre ALTER COLUMN columna_nombre nuevo_tipo_dato;`
4. Cambiar el nombre de una tabla `ALTER TABLE tabla_nombre RENAME TO nuevo_nombre;`