

Integridad y modelos de datos

Objetivos

- A que nos referimos con integridad referencial
- ¿Qué es una clave primaria?
- Modelos de entidad relación
- Cardinalidad
- Normalización

Integridad referencial

La integridad referencial es un concepto fundamental en el diseño de bases de datos relacionales que garantiza la coherencia y validez de los datos almacenados. En un entorno de base de datos, donde la información se organiza en tablas, la integridad referencial se centra en la consistencia de las relaciones entre estas tablas.

En términos simples, la integridad referencial se basa en dos tipos clave: la clave primaria y la clave foránea. La clave primaria es un conjunto de uno o más campos que identifican de manera única cada registro en una tabla, mientras que la clave extranjera es un campo en otra tabla que establece una conexión referencial con la clave primaria de la primera tabla.

Esta relación padre-hijo entre tablas es esencial para mantener la coherencia de los datos. La regla de unicidad dicta que los valores en la clave primaria deben ser únicos, mientras que la regla de referencialidad establece que los valores en la clave extranjera deben coincidir con los valores correspondientes en la clave primaria o ser nulos.

La integridad referencial asegura que, al modificar o eliminar datos en una tabla, se mantengan las relaciones adecuadas con otras tablas, evitando así datos huérfanos o inconsistentes. La base de datos, al aplicar la integridad referencial, impide acciones que podrían comprometer la consistencia de los datos, como eliminar registros principales que están siendo referenciados por registros secundarios.


Clave Primaria

La clave primaria (Primary Key en inglés) es un concepto fundamental en bases de datos relacionales. Se refiere a uno o más campos en una tabla de base de datos que tienen dos propiedades clave: deben contener valores únicos y no pueden contener valores nulos. La clave primaria se utiliza para identificar de manera única cada registro en la tabla.

Para añadir una clave primaria a una tabla lo podemos hacer antes de crear la tabla o después de crearla aunque se aconseja hacerlo antes para no provocar errores inesperados

Para crear la clave primaria antes de crear la tabla podemos usar el siguiente ejemplo:

```
CREATE TABLE users(  
  id INT SERIAL PRIMARY KEY,  
  nombre VARCHAR,  
  edad INT  
);
```



Para crear la clave primaria en una tabla existente podemos usar el siguiente ejemplo:

```
ALTER TABLE users ADD PRIMARY KEY (id)
```

Clave foránea

La clave foránea (Foreign Key en inglés) es un concepto crucial en bases de datos relacionales. Se trata de un campo o conjunto de campos en una tabla que establece una relación referencial con la clave primaria de otra tabla. La clave foránea actúa como un vínculo entre las dos tablas y se utiliza para mantener la integridad referencial en la base de datos.


algunas características clave de la clave foránea:

- **Relación entre Tablas:** La clave foránea crea una relación entre la tabla que la contiene y la tabla que tiene la clave primaria a la que se hace referencia. Esta relación se conoce como una relación padre-hijo, donde la tabla con la clave primaria se considera la "tabla padre" y la tabla con la clave foránea se considera la "tabla hija".
- **Coincidencia de Valores:** Los valores en la clave foránea deben coincidir con los valores correspondientes en la clave primaria de la tabla referenciada. Esta coincidencia garantiza que la información en la tabla hija esté relacionada de manera coherente con la información en la tabla padre.

- **Mantenimiento de la Integridad Referencial:** La presencia de claves foráneas ayuda a mantener la integridad referencial en la base de datos. Esto significa que no se pueden crear relaciones "huérfanas" en las que existan referencias a registros que no existen en la tabla padre.
- **Acciones en Cascada:** Al definir una clave foránea, se pueden especificar acciones que deben realizarse si se intenta modificar o eliminar registros en la tabla padre. Por ejemplo, se puede configurar para realizar una acción en cascada, que automáticamente actualiza o elimina registros en la tabla hija cuando se realiza una modificación en la tabla padre.

Siguiendo el ejemplo anterior de la tabla users vamos a crear la tabla post y vamos a referenciar mediante una clave foránea a la tabla users para que cada post se asocie a un usuario:

```
CREATE TABLE posts(  
  id INT SERIAL PRIMARY KEY,  
  titulo VARCHAR,  
  contenido VARCHAR,  
  user_id INT REFERENCES users(id)  
);
```



Modelos de entidad relación

En el diseño de bases de datos, es crucial entender cómo las entidades del mundo real están relacionadas entre sí. Los modelos entidad-relación proporcionan una representación visual de esta información, permitiendo a los diseñadores de bases de datos conceptualizar y comunicar la estructura de la base de datos de manera clara.

existen 3 tipos de modelos famosos:

- **Modelo conceptual:** El modelo conceptual de base de datos es una representación abstracta de los datos de una organización. Se basa en los conceptos del mundo real, como personas, lugares y cosas, y cómo se relacionan entre sí.

El objetivo del modelo conceptual es proporcionar una visión general de los datos que se almacenarán en la base de datos. Es un primer paso importante en el proceso de diseño de una base de datos, ya que ayuda a definir los requisitos de datos de la organización.

Los componentes principales del modelo conceptual son:

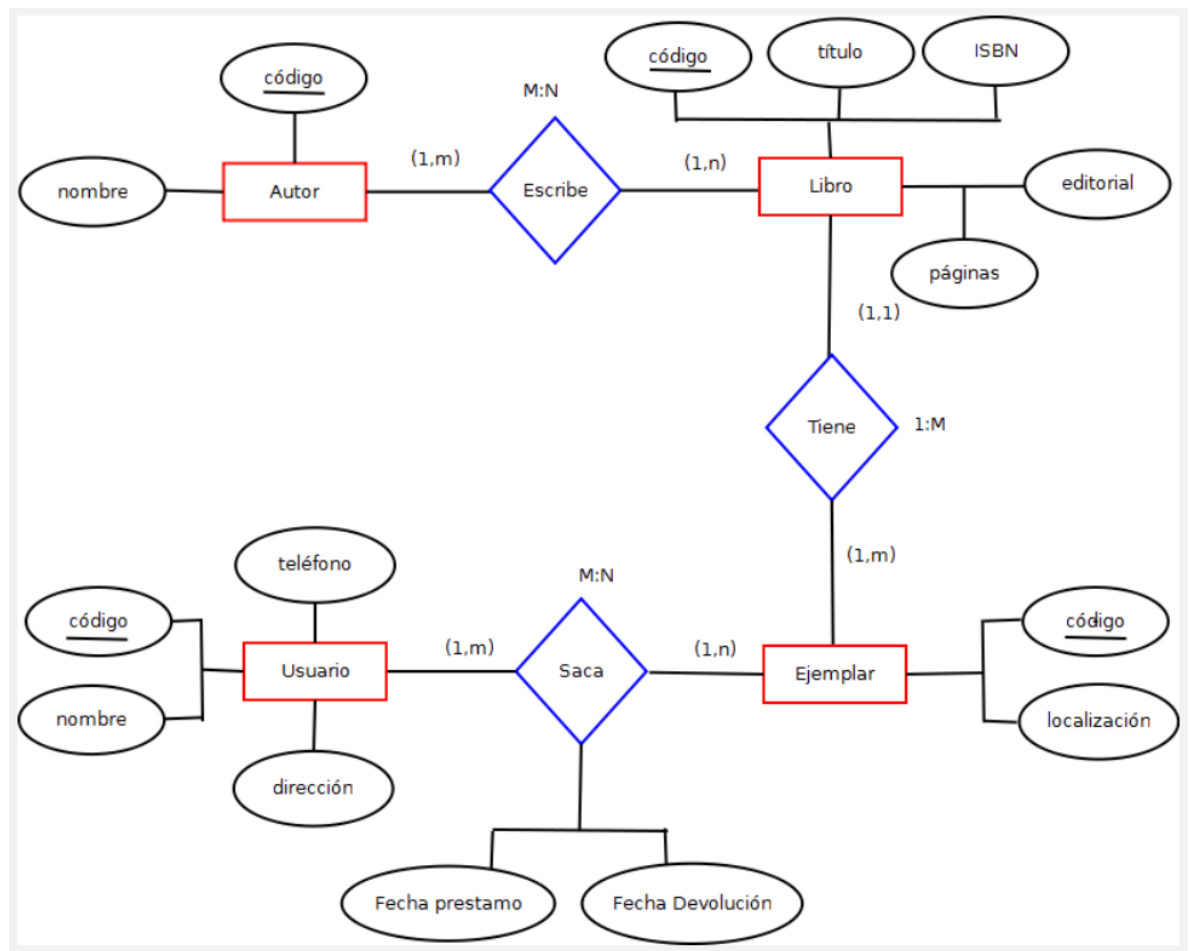
Entidades: Representan conceptos del mundo real, como personas, lugares y cosas.

Atributos: Representan las propiedades de las entidades.

Relaciones: Representan las conexiones entre las entidades.

Por ejemplo, una entidad de "Cliente" podría tener los atributos "Nombre", "Dirección" y "Teléfono". Una relación entre las entidades "Cliente" y "Pedido" podría indicar que un cliente puede realizar varios pedidos.

El modelo conceptual se puede representar mediante un diagrama de relación de entidades (ERD). Los ERD son una forma gráfica de representar los componentes del modelo conceptual.

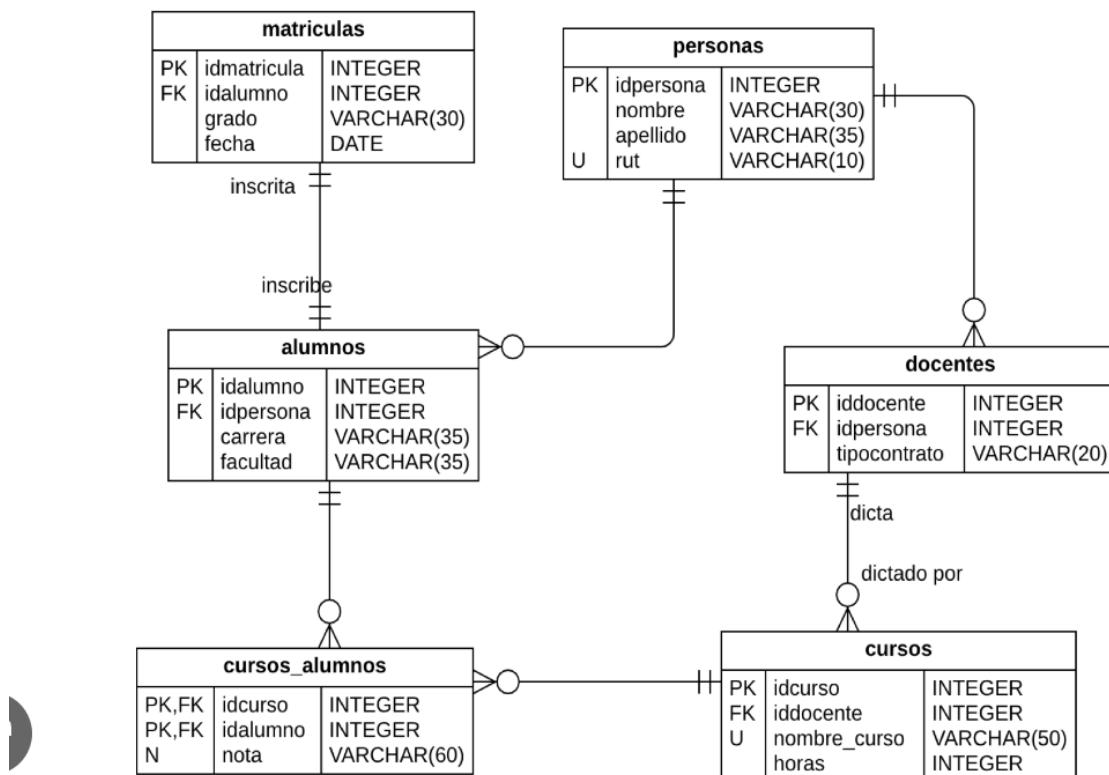


- **Modelo lógico:** El modelo lógico es la siguiente fase en el diseño de bases de datos después del modelo conceptual. Mientras que el modelo conceptual se centra en la representación abstracta de entidades y relaciones, el modelo lógico se ocupa de traducir ese diseño conceptual a una estructura que puede ser implementada en un sistema de gestión de bases de datos (SGBD). Aquí hay algunos aspectos clave del modelo lógico:

Tablas y Esquemas: En el modelo lógico, las entidades del modelo conceptual se transforman en tablas en el SGBD. Cada tabla representa una entidad y tiene una estructura bien definida con columnas que representan los atributos. Además, se definen los esquemas que especifican la organización lógica de las tablas en la base de datos.

Claves: Se identifican las claves primarias para cada tabla, que son los campos (o conjunto de campos) que identifican de manera única cada fila en la tabla. También se pueden definir claves únicas y claves foráneas para mantener la integridad referencial entre las tablas.

Tipos de Datos y Restricciones: Se especifican los tipos de datos para cada columna, como enteros, cadenas de texto, fechas, etc. También se definen restricciones de integridad, como restricciones de clave primaria, restricciones de clave foránea y restricciones de unicidad.



- **Modelo físico:** El modelo físico es la última fase en el diseño de bases de datos y se ocupa de la implementación concreta de la estructura de la base de datos en un sistema de gestión de bases de datos (SGBD) específico. Aquí hay algunos aspectos clave del modelo físico:

Definición de Tablas y Esquemas: En esta fase, se definen las tablas y esquemas exactos que se utilizarán en la base de datos física. Se especifica la estructura de cada tabla, incluyendo los nombres de las columnas, tipos de datos, restricciones y cualquier otra propiedad necesaria.

Índices y Claves: Se crean índices para mejorar el rendimiento de las consultas. Además, se implementan las claves primarias, claves foráneas y cualquier otra restricción de integridad definida en el modelo lógico.

Espacios de Almacenamiento: Se determina cómo se almacenarán físicamente los datos en el sistema de archivos del servidor de la base de datos. Esto incluye decisiones sobre tablespaces, particiones y otros aspectos relacionados con el almacenamiento físico.

Optimización de Rendimiento: Durante la fase física, se realizan ajustes para mejorar el rendimiento de las consultas. Esto puede implicar la denormalización selectiva para evitar operaciones JOIN costosas y la consideración de estrategias de almacenamiento en caché.

Seguridad: Se implementan medidas de seguridad, como permisos de acceso, roles y autenticación, para garantizar que solo usuarios autorizados tengan acceso a la base de datos y a los datos específicos.

Transacciones: Se definen y configuran parámetros relacionados con el control de transacciones, asegurando la consistencia y la integridad de los datos, incluso en escenarios de fallas.

Cardinalidad: es tipo de relación que hay entre dos tablas.

Cardinalidad

En el contexto de los modelos entidad-relación en el diseño de bases de datos, la cardinalidad se refiere a la relación numérica entre instancias de una entidad que participan en una relación con instancias de otra entidad. Describe la cantidad de ocurrencias o instancias que pueden estar asociadas a través de una relación específica.

Existen tres tipos comunes de cardinalidad en una relación:

- **Uno a Uno (1:1):** En una relación uno a uno, una instancia de una entidad está relacionada con exactamente una instancia de otra entidad, y viceversa. Por ejemplo, en un sistema de gestión de empleados, podría haber una relación uno a uno entre un empleado y su número de identificación único.
- **Uno a Muchos (1:N):** En una relación uno a muchos, una instancia de una entidad está relacionada con cero o más instancias de otra entidad, pero una instancia de la segunda entidad está relacionada con exactamente una instancia de la primera entidad. Por ejemplo, en un sistema de gestión de bibliotecas, un autor puede estar relacionado con varios libros (uno a muchos).
- **Muchos a Muchos (N:M):** En una relación muchos a muchos, una instancia de una entidad puede estar relacionada con cero o más instancias de otra entidad, y viceversa. En este caso, se utiliza una tabla intermedia (tabla de relación o tabla de unión) para representar la relación. Por ejemplo, en un sistema de gestión de cursos y estudiantes, un estudiante puede estar inscrito en varios cursos, y un curso puede tener varios estudiantes inscritos.

Cardinalidad	Chen	
Uno a uno (1:1)	1 —◆— 1	—+—+—
Uno a muchos (1:N)	1 —◆— N	—+—<—
Muchos a uno (N:1)	N —◆— 1	>—+—
Muchos a muchos (M:N)	M —◆— N	>—<—

Normalización

La normalización en bases de datos es un proceso sistemático de diseño que tiene como objetivo reducir la redundancia de datos y mejorar la integridad de la base de datos. El objetivo principal es organizar la información de manera eficiente y eliminar dependencias innecesarias entre los datos. La normalización se realiza aplicando reglas específicas a la estructura de las tablas en una base de datos relacional.

Las reglas de normalización más comunes son definidas por las formas normales. Las formas normales representan niveles crecientes de restricciones que se aplican a la estructura de las tablas para garantizar una organización óptima de los datos. Las tres formas normales más conocidas son 1NF (Primera Forma Normal), 2NF (Segunda Forma Normal) y 3NF (Tercera Forma Normal).

Primera Forma Normal (1NF): Imagina que tienes una tabla con información sobre libros en una biblioteca. En 1NF, cada celda de la tabla debe contener un solo valor, y no conjuntos de valores o listas. Por ejemplo, si tienes una columna llamada "Autores" y tiene múltiples nombres de autores separados por comas, estarías violando la 1NF. En lugar de eso, deberías tener una fila para cada autor y otra tabla que relacione los autores con los libros.

Segunda Forma Normal (2NF): Ahora, piensa en una tabla de pedidos en una tienda en línea. En 2NF, además de cumplir con la 1NF, cada columna no clave (que no es parte de la clave primaria) debe depender completamente de la clave primaria. Si tienes una tabla de pedidos con información sobre productos y la cantidad, pero también incluyes información sobre el proveedor del producto en la misma tabla, podrías estar violando la 2NF. Deberías dividir esa información en diferentes tablas para que cada dato esté relacionado solo con la clave primaria correspondiente.

Tercera Forma Normal (3NF): Vamos a seguir con la tabla de pedidos. En 3NF, además de cumplir con la 2NF, no debe haber dependencias transitivas. Esto significa que si tienes una columna que depende de otra columna que no es la clave primaria, deberías separar esas dependencias. Por ejemplo, si tienes una tabla de clientes con información sobre el código postal y también sobre la ciudad y el estado, podrías querer dividir esa información en una tabla separada de ubicaciones para evitar dependencias transitivas.

En resumen, la normalización es como organizar tus datos para que estén ordenados y sin información innecesaria o redundante. La 1NF asegura que cada celda tenga un solo valor, la 2NF asegura que cada columna no clave dependa completamente de la clave primaria, y la 3NF evita dependencias transitivas. Estas reglas ayudan a mantener la integridad y eficiencia de tu base de datos.