UF2175 -
Diseño de Bases de Datos Relacionales

Unidad 2: Modelos conceptuales de bases de datos

By: Sergi Faura Alsina

SOC Servei d'Ocupació de Catalunya · Generalitat de Catalunya · Ministerio de Educación, Formación Profesional y Deportes · Ministerio de Trabajo y Economía Social · SEPE

Índice

2. Modelos conceptuales de bases de datos

Los **modelos conceptuales de bases de datos** son fundamentales en el diseño de sistemas de información, ya que permiten representar de manera abstracta la estructura y las relaciones de los datos en una base de datos. Estos modelos proporcionan una visión clara y simplificada de cómo se organizan los datos y cómo se relacionan entre sí, facilitando la comprensión tanto para los diseñadores de bases de datos como para los usuarios. El **modelo entidad-relación (E-R)** es uno de los modelos conceptuales más utilizados en el diseño de bases de datos relacionales. Este modelo permite representar las entidades (objetos o conceptos del mundo real), sus atributos (características o propiedades) y las relaciones entre ellas.

Además del modelo E-R básico, existe el **modelo entidad-relación extendido**, que incorpora características adicionales para manejar estructuras más complejas, como las subclases y jerarquías de especialización. Ambos modelos son esenciales para el proceso de diseño de bases de datos, ya que proporcionan un marco teórico sólido para estructurar la información.

Finalmente, un aspecto crucial en el diseño de bases de datos es la definición de **restricciones de integridad**, que aseguran la coherencia y validez de los datos. Estas restricciones pueden ser inherentes al propio modelo, o pueden ser explícitamente definidas por los diseñadores para garantizar que los datos cumplan con las reglas de negocio y las restricciones lógicas necesarias para el correcto funcionamiento del sistema.

Este capítulo explora en detalle los conceptos fundamentales del modelo entidad-relación, sus extensiones y las restricciones de integridad que deben aplicarse en una base de datos bien diseñada.

2.1. El modelo entidad-relación:

El **modelo entidad-relación (E-R)** es un enfoque estándar en el diseño conceptual de bases de datos que permite representar las estructuras de datos y sus relaciones de manera gráfica y lógica. Desarrollado por Peter Chen en 1976, el modelo E-R utiliza conceptos como entidades, atributos y relaciones para modelar datos del mundo real en un sistema de bases de datos.

En este modelo, las **entidades** representan objetos o conceptos del mundo real, como personas, lugares o cosas, que tienen una existencia independiente dentro del sistema. Cada entidad se describe mediante **atributos**, que son las características que la definen, como el nombre, la fecha o el identificador. Además, el modelo E-R describe las **relaciones** entre las entidades, representando cómo interactúan o se asocian entre sí.

El modelo E-R es particularmente útil durante la fase de diseño conceptual, ya que ayuda a los diseñadores a capturar los requisitos del sistema de una manera clara y organizada antes de la implementación física de la base de datos. Además, el uso de diagramas E-R facilita la visualización de la estructura de la base de datos y mejora la comunicación entre los diseñadores y los usuarios finales.

2.1.1. Entidades.

Las **entidades** son un concepto fundamental en el modelo entidad-relación (E-R). Representan los objetos o conceptos del mundo real sobre los cuales se desea almacenar información en una base de datos. Una entidad puede ser cualquier cosa que tenga una existencia independiente dentro del sistema de información, ya sea un objeto físico (como una persona o un producto) o un objeto abstracto (como un proyecto o una transacción).

Cada entidad en el modelo E-R se describe a través de un conjunto de atributos que contienen información relevante sobre ella. Por ejemplo, en una base de datos de una empresa, la entidad **Empleado** podría tener atributos como **Nombre**, **Fecha de Nacimiento**, **Cargo** y **Salario**. En otro contexto, la entidad **Producto** podría tener atributos como **Código de Producto**, **Nombre del Producto**, **Precio** y **Cantidad en Stock**.

Características de las Entidades

1. **Existencia Independiente**: Las entidades existen de manera independiente dentro del sistema de información, lo que significa que pueden ser identificadas de manera única y almacenadas como registros en la base de datos. Por ejemplo, cada empleado en una empresa es una entidad única con su propio conjunto de atributos.
2. **Tipos de Entidades**:
    - **Entidades Fuertes**: Son entidades que tienen una existencia propia y pueden ser identificadas de manera única por sus propios atributos. Por ejemplo, la entidad **Empleado** es una entidad fuerte, ya que cada empleado tiene un identificador único (como un número de identificación).
    - **Entidades Débiles**: Son entidades que dependen de la existencia de otra entidad fuerte para ser identificadas. Estas entidades no tienen una clave primaria propia y se identifican en combinación con la entidad fuerte a la que están asociadas. Un ejemplo de entidad débil podría ser la entidad **Dependiente**, que se identifica a través de la entidad **Empleado**.
3. **Conjunto de Entidades**: Un **conjunto de entidades** es una colección de entidades que comparten los mismos atributos. Por ejemplo, en una base de datos que registra información sobre empleados, todos los empleados forman un conjunto de entidades de tipo **Empleado**.

4. **Clave Primaria**: Las entidades suelen tener un atributo o un conjunto de atributos que sirven como **clave primaria**, la cual permite identificar de manera única cada instancia de la entidad en la base de datos. Por ejemplo, el **ID_Empleado** podría ser la clave primaria para identificar de manera única a cada empleado.

Ejemplo de Entidades

Consideremos un sistema de gestión de una universidad:

- **Entidad Estudiante**: Los estudiantes son entidades con atributos como **ID Estudiante**, **Nombre**, **Apellido**, **Fecha de Nacimiento** y **Carrera**.
- **Entidad Curso**: Los cursos son otra entidad, con atributos como **Código de Curso**, **Nombre del Curso** y **Créditos**.
- **Entidad Profesor**: Los profesores también son entidades, con atributos como **ID Profesor**, **Nombre**, **Departamento** y **Especialización**.

Cada una de estas entidades es representativa de un objeto del mundo real que tiene importancia dentro del sistema de información de la universidad. A través de las entidades, se puede almacenar y gestionar información relevante sobre cada estudiante, curso y profesor.

Relación con otras entidades

Las entidades no existen de manera aislada; a menudo, están relacionadas entre sí en el sistema de bases de datos. Por ejemplo, un **Estudiante** puede estar inscrito en varios **Cursos**, y un **Profesor** puede impartir varios **Cursos**. Estas relaciones entre las entidades se gestionan a través de relaciones en el modelo entidad-relación, que se explorarán en secciones posteriores.

En resumen, las entidades son un componente clave del diseño de bases de datos, ya que representan los objetos del mundo real sobre los que se desea almacenar información. Definir correctamente las entidades y sus atributos es fundamental para el éxito del diseño de la base de datos y su posterior implementación en un sistema de gestión de bases de datos (DBMS).

2.1.2. Interrelaciones: Cardinalidad, Rol y Grado.

En el **modelo entidad-relación (E-R)**, las **interrelaciones** describen cómo las entidades interactúan entre sí dentro del sistema de bases de datos. Las interrelaciones permiten modelar las conexiones y asociaciones entre las entidades, lo que es esencial para capturar

el comportamiento de los datos en situaciones del mundo real. Estas relaciones se caracterizan por aspectos clave como la **cardinalidad**, el **rol** y el **grado**.

Cardinalidad

La **cardinalidad** describe el número de instancias de una entidad que pueden estar asociadas con instancias de otra entidad en una relación. Es decir, la cardinalidad especifica cuántas veces una entidad puede participar en una relación en función de su vínculo con otra entidad. Existen varios tipos de cardinalidad:

1. **Uno a Uno (1:1)**: En una relación de uno a uno, una instancia de la primera entidad se asocia con solo una instancia de la segunda entidad, y viceversa. Este tipo de relación es menos común en bases de datos. Un ejemplo de relación 1:1 podría ser la relación entre **Persona** y **Pasaporte**, donde cada persona tiene un único pasaporte, y cada pasaporte está asociado con una única persona.
2. **Uno a Muchos (1)**: En una relación de uno a muchos, una instancia de la primera entidad puede estar asociada con muchas instancias de la segunda entidad, pero una instancia de la segunda entidad solo puede estar asociada con una instancia de la primera entidad. Este es un tipo de relación muy común en bases de datos. Por ejemplo, un **Departamento** puede tener muchos **Empleados**, pero cada empleado pertenece a un solo departamento.
3. **Muchos a Muchos (M)**: En una relación de muchos a muchos, una instancia de la primera entidad puede estar asociada con muchas instancias de la segunda entidad, y viceversa. Este tipo de relación también es común, pero a menudo requiere una tabla intermedia o una entidad de relación para gestionarla en una base de datos relacional. Un ejemplo clásico es la relación entre **Estudiantes** y **Cursos**: un estudiante puede inscribirse en varios cursos, y un curso puede tener muchos estudiantes inscritos.

La correcta definición de la cardinalidad es esencial para asegurar que las relaciones entre las entidades reflejen con precisión las reglas de negocio y las restricciones del sistema.

Rol

El **rol** en una relación especifica la función que desempeña cada entidad dentro de la relación. Los roles ayudan a identificar cómo cada entidad está relacionada con la otra en el contexto de la relación. Esto es particularmente útil cuando se tienen relaciones recursivas, donde una entidad se relaciona consigo misma.

Por ejemplo, en una relación de supervisión dentro de una empresa, un **Empleado** puede supervisar a otro **Empleado**. Aquí, el rol de "supervisor" y el rol de "supervisado" ayudan a clarificar las funciones de cada instancia de la entidad en la relación.

Los roles también son útiles para mejorar la legibilidad y comprensión del modelo entidad-relación, asegurando que se comprendan claramente las conexiones entre las entidades, especialmente en relaciones más complejas.

Grado

El **grado** de una relación se refiere al número de entidades que participan en la relación. Dependiendo del número de entidades involucradas, las relaciones pueden clasificarse en diferentes tipos:

1. **Relación Unaria** (Grado 1): Es una relación en la que una sola entidad se relaciona consigo misma. Este tipo de relación es también conocido como relación recursiva. Un ejemplo de relación unaria es la relación de **Empleado** con **Empleado** en un contexto de supervisión, donde un empleado puede supervisar a otro empleado.
2. **Relación Binaria** (Grado 2): Es el tipo de relación más común, donde dos entidades diferentes están relacionadas entre sí. Por ejemplo, la relación entre **Empleado** y **Departamento** es una relación binaria.
3. **Relación Ternaria** (Grado 3): Es una relación en la que participan tres entidades diferentes. Este tipo de relación puede ser más complejo de modelar, ya que implica interacciones entre múltiples entidades simultáneamente. Un ejemplo de relación ternaria podría ser la relación entre **Proveedor**, **Producto** y **Almacén**, donde se define qué productos se almacenan en qué almacenes y de qué proveedor provienen.

Ejemplos Prácticos

1. **Relación de Uno a Muchos (1
)**: Un ejemplo común en un sistema de gestión escolar es la relación entre **Profesor** y **Clase**. Un profesor puede enseñar muchas clases, pero cada clase está a cargo de un solo profesor. Aquí, la cardinalidad es 1
   , el rol del profesor es "enseñar" y el rol de la clase es "ser enseñada".
2. **Relación de Muchos a Muchos (M
)**: En una universidad, los **Estudiantes** pueden estar inscritos en varios **Cursos**, y un curso puede tener muchos estudiantes inscritos. Para gestionar esta relación, generalmente se introduce una tabla intermedia llamada **Inscripción**, que almacena las relaciones entre estudiantes y cursos.
3. **Relación Recursiva (Unaria)**: En un organigrama corporativo, un **Empleado** puede supervisar a otros empleados. Aquí, la entidad **Empleado** está relacionada consigo misma, con un rol de "supervisor" y otro de "supervisado".

Conclusión

La correcta identificación y modelado de las **interrelaciones** entre entidades es crucial para un diseño de bases de datos eficiente y preciso. La **cardinalidad** establece cuántas veces las entidades pueden participar en la relación, el **rol** define las funciones de las entidades dentro de la relación, y el **grado** determina cuántas entidades participan en la relación. Estos conceptos ayudan a capturar la complejidad de las interacciones del mundo real en el modelo entidad-relación, asegurando que las bases de datos resultantes reflejen con precisión las reglas de negocio y las interacciones de datos esperadas.

2.1.3. Dominios y valores.

En el modelo entidad-relación (E-R), el concepto de **dominios** y **valores** es esencial para definir las restricciones y características de los atributos de una entidad o una relación. Los dominios permiten garantizar que los datos almacenados en la base de datos sean válidos y coherentes con respecto a las reglas del sistema.

Dominios

El **dominio** de un atributo es el conjunto de valores posibles que dicho atributo puede tomar. En otras palabras, un dominio define el tipo de datos y los valores permitidos para un atributo específico. Por ejemplo, si un atributo es **Edad**, su dominio podría estar compuesto por números enteros en un rango de 0 a 120, lo que significa que la base de datos solo aceptará valores que se encuentren dentro de ese rango.

Los dominios se pueden clasificar de varias formas:

1. **Tipos de datos básicos**: Los dominios suelen estar asociados a tipos de datos básicos que definen la naturaleza de los valores permitidos. Algunos ejemplos incluyen:
    - **Números enteros**: Los valores deben ser números enteros (e.g., `0`, `1`, `42`).
    - **Números decimales**: Los valores deben ser números con decimales (e.g., `3.14`, `99.99`).
    - **Cadenas de texto**: Los valores deben ser cadenas de caracteres (e.g., `"Juan"`, `"Dirección"`).
    - **Fechas**: Los valores deben ser fechas válidas (e.g., `2023-08-26`).
    - **Valores booleanos**: Los valores deben ser `verdadero` o `falso`.
2. **Dominios restringidos por reglas de negocio**: Los dominios también pueden restringirse por reglas específicas del negocio o del sistema. Por ejemplo:
    - El atributo **Género** podría tener un dominio limitado a los valores `Masculino`, `Femenino` o `No Especificado`.

    - El atributo **Categoría de Producto** podría tener un dominio limitado a las categorías `Electrónica`, `Ropa`, `Alimentos`, etc.
3. **Dominios de conjunto de valores predefinidos**: Algunos atributos pueden tener dominios que limitan sus valores a un conjunto cerrado de opciones. Por ejemplo:
    - El atributo **Estado Civil** podría tener un dominio compuesto por los valores `Soltero`, `Casado`, `Divorciado`, `Viudo`.
    - El atributo **Estado del Pedido** en un sistema de compras en línea podría tener los valores `Pendiente`, `Enviado`, `Entregado`, `Cancelado`.

Valores

Los **valores** son las instancias concretas que se almacenan en la base de datos para los atributos de las entidades. Cada valor que se asigna a un atributo debe pertenecer a su dominio correspondiente, lo que asegura que los datos sean válidos y conformes a las reglas establecidas.

Por ejemplo, si tenemos una entidad **Empleado** con un atributo **Edad**, cuyo dominio es el conjunto de números enteros de 18 a 65, un valor válido para ese atributo podría ser `35`. Sin embargo, un valor como `75` no sería aceptado, ya que se encuentra fuera del dominio permitido.

El establecimiento de dominios y la verificación de que los valores cumplan con estos dominios son fundamentales para mantener la **integridad de los datos** en el sistema de bases de datos. Si no se definen dominios adecuados, se corre el riesgo de que la base de datos acepte datos erróneos o inválidos, lo que podría comprometer la calidad de la información y provocar fallos en las aplicaciones que dependen de esos datos.

Ejemplos de Dominios y Valores

1. **Entidad Estudiante**:
    - **Atributo Nombre**: Su dominio sería una cadena de texto con un límite de caracteres, por ejemplo, hasta 100 caracteres.
    - **Atributo Fecha de Nacimiento**: Su dominio sería el tipo de datos de fecha, y los valores válidos serían fechas entre un rango permitido, como desde `1900-01-01` hasta `2010-12-31`.
    - **Atributo Promedio**: Su dominio podría ser números decimales en el rango de `0.0` a `10.0`, asegurando que los promedios de los estudiantes se mantengan dentro de valores razonables.
2. **Entidad Producto**:
    - **Atributo Precio**: El dominio sería un número decimal positivo. Los valores permitidos estarían en un rango de, por ejemplo, `0.01` a `10,000.00`, dependiendo del tipo de producto.

- **Atributo Categoría**: El dominio sería un conjunto de valores predefinidos, como `Electrónica`, `Ropa`, `Alimentos`, asegurando que cada producto esté clasificado en una de las categorías válidas.

3. **Entidad Pedido**:
   - **Atributo Estado del Pedido**: El dominio podría ser un conjunto de valores que representen el estado actual del pedido, como `Pendiente`, `En Proceso`, `Completado`, `Cancelado`. Solo se permitiría uno de estos valores al registrar o actualizar un pedido.

Importancia de los Dominios

El uso adecuado de dominios no solo asegura la validez de los datos, sino que también simplifica la lógica de la aplicación al reducir la necesidad de validaciones adicionales en las capas superiores del sistema. Al definir dominios adecuados desde el diseño de la base de datos, se previenen errores y se asegura que los datos sean coherentes y utilizables a lo largo de todo el ciclo de vida del sistema.

Además, los dominios también juegan un papel crucial en la **integridad referencial** dentro de la base de datos. Por ejemplo, si un atributo en una entidad debe coincidir con una clave primaria de otra entidad (como en una relación de clave foránea), el dominio de ese atributo debe estar alineado con el dominio de la clave primaria correspondiente.

Conclusión

Los **dominios y valores** son elementos clave en el modelo entidad-relación, ya que permiten establecer las restricciones sobre los datos y asegurar que los valores almacenados en la base de datos sean válidos y coherentes con las reglas del sistema. La correcta definición de los dominios es esencial para mantener la integridad de los datos y evitar problemas a lo largo del ciclo de vida de la base de datos.

2.1.4. Atributos.

En el **modelo entidad-relación (E-R)**, los **atributos** son las características o propiedades que describen las entidades y las relaciones. Los atributos proporcionan información detallada sobre las entidades en una base de datos, y cada entidad se define mediante un conjunto de atributos específicos. Los atributos pueden variar desde valores simples hasta estructuras más complejas, dependiendo de la naturaleza de los datos que representan.

Tipos de Atributos

1. **Atributos Simples**: Son aquellos que no se pueden dividir en subpartes más pequeñas. Un atributo simple representa un único valor atómico que describe una característica de la entidad. Por ejemplo, el atributo **Nombre** en una entidad **Empleado** es un valor único que no puede subdividirse.
2. **Atributos Compuestos**: Son aquellos que pueden dividirse en partes más pequeñas, llamadas subatributos, cada una de las cuales proporciona un detalle adicional sobre la entidad. Por ejemplo, el atributo **Dirección** podría descomponerse en subatributos como **Calle**, **Ciudad**, **Código Postal** y **País**.
3. **Atributos Multivalorados**: Son aquellos que pueden tener más de un valor simultáneamente para una entidad específica. Por ejemplo, un atributo **Teléfono** en la entidad **Empleado** podría tener varios valores si un empleado tiene más de un número de teléfono registrado. En estos casos, es importante gestionar adecuadamente la relación entre los valores múltiples.
4. **Atributos Derivados**: Son aquellos cuyo valor se puede calcular o derivar a partir de otros atributos. Por ejemplo, el atributo **Edad** puede derivarse del atributo **Fecha de Nacimiento**. Estos atributos no se almacenan físicamente en la base de datos, sino que se calculan cuando son requeridos.
5. **Atributos Nulos**: Un atributo puede tener un valor nulo (NULL) si no tiene un valor definido o si la información no es aplicable en el contexto de esa entidad. Por ejemplo, el atributo **Fecha de Salida** de un empleado puede ser nulo si el empleado sigue trabajando en la empresa y aún no ha sido registrado como "salido".

Atributos Clave

En el contexto del modelo entidad-relación, algunos atributos tienen una importancia especial ya que permiten identificar de manera única cada instancia de una entidad. Estos son los **atributos clave** o **claves primarias**:

1. **Clave Primaria**: Es un atributo o conjunto de atributos que identifica de manera única cada instancia de una entidad en la base de datos. Ningún valor de la clave primaria puede ser nulo, y no puede haber dos registros en la tabla con el mismo valor en la clave primaria. Por ejemplo, el atributo **ID_Empleado** en la entidad **Empleado** podría ser la clave primaria.
2. **Clave Candidata**: Son aquellos atributos que podrían servir como clave primaria, pero que no han sido seleccionados como tal. Una entidad puede tener varias claves candidatas, pero solo una de ellas se designa como la clave primaria.
3. **Clave Foránea**: Es un atributo en una entidad que actúa como una referencia a la clave primaria de otra entidad. Las claves foráneas se utilizan para establecer relaciones entre diferentes tablas en una base de datos relacional. Por ejemplo, en la entidad **Pedido**, el atributo **ID_Cliente** podría ser una clave foránea que se refiere a la clave primaria **ID_Cliente** en la entidad **Cliente**.

Ejemplos de Atributos

1. **Entidad Estudiante**:
    - **Atributo Nombre**: Sería un atributo simple que representa el nombre del estudiante.
    - **Atributo Dirección**: Podría ser un atributo compuesto que incluye subatributos como **Calle**, **Ciudad** y **Código Postal**.
    - **Atributo Teléfono**: Podría ser un atributo multivalorado si el estudiante tiene más de un número de contacto registrado.
    - **Atributo Edad**: Podría ser un atributo derivado a partir del atributo **Fecha de Nacimiento**.
2. **Entidad Producto**:
    - **Atributo Código de Producto**: Sería un atributo simple que sirve como clave primaria para identificar de manera única a cada producto.
    - **Atributo Precio**: Sería un atributo simple que describe el precio del producto.
    - **Atributo Categoría**: Sería un atributo simple que clasifica el producto en una categoría específica, como `Electrónica` o `Ropa`.
3. **Entidad Empleado**:
    - **Atributo Nombre Completo**: Podría ser un atributo compuesto que se divide en subatributos como **Nombre**, **Apellido Paterno** y **Apellido Materno**.
    - **Atributo Fecha de Contratación**: Sería un atributo simple que almacena la fecha en que el empleado fue contratado.
    - **Atributo Salario**: Sería un atributo simple que representa el salario del empleado.
    - **Atributo Dependientes**: Podría ser un atributo multivalorado si el empleado tiene varios dependientes.

Función de los Atributos en la Base de Datos

Los atributos juegan un papel crucial en el diseño y funcionamiento de una base de datos. No solo proporcionan la información detallada sobre cada entidad, sino que también son fundamentales para garantizar la integridad y la coherencia de los datos. A través de la correcta definición de los atributos, se pueden establecer reglas de validación, restricciones y relaciones entre entidades.

Además, los atributos permiten realizar consultas y análisis en la base de datos, ya que proporcionan los datos necesarios para responder preguntas específicas o generar informes. Por ejemplo, un usuario puede querer consultar todos los empleados cuyo salario sea superior a un valor determinado, lo que requeriría filtrar los registros basados en el atributo **Salario**.

Conclusión

Los **atributos** son un componente esencial en el modelo entidad-relación, ya que definen las características de las entidades y relaciones en una base de datos. Comprender los diferentes tipos de atributos y cómo se utilizan es fundamental para diseñar una base de datos eficiente y bien estructurada. La correcta definición de los atributos asegura que los datos almacenados sean precisos, válidos y útiles para el análisis y la toma de decisiones dentro del sistema de información.

2.1.5. Propiedades identificatorias.

Las **propiedades identificatorias** en el modelo entidad-relación (E-R) son atributos o combinaciones de atributos que permiten identificar de manera única a cada instancia de una entidad. Estas propiedades son fundamentales para garantizar la unicidad de los registros dentro de una base de datos y son esenciales para establecer relaciones entre entidades. Las propiedades identificatorias aseguran que no haya duplicados y que cada entidad pueda ser referenciada de manera precisa.

Clave Primaria (Primary Key)

La **clave primaria** es la propiedad identificatoria más importante de una entidad. Es un atributo (o un conjunto de atributos) que identifica de manera única a cada instancia de una entidad dentro de un conjunto de entidades. Ningún valor de la clave primaria puede repetirse en dos registros diferentes de la misma tabla, y tampoco puede ser nulo.

- **Unicidad**: La clave primaria garantiza que no haya dos filas en una tabla con el mismo valor en la clave primaria. Por ejemplo, en la entidad **Empleado**, un atributo como **ID_Empleado** puede servir como clave primaria para asegurar que cada empleado esté representado de manera única en la base de datos.
- **No nulidad**: La clave primaria no puede tener un valor nulo, ya que un valor nulo indicaría la ausencia de un identificador único. Esto violaría la regla de unicidad requerida para la clave primaria.

La clave primaria es fundamental no solo para la integridad de la tabla en la que se define, sino también para establecer relaciones con otras tablas a través de **claves foráneas**.

Ejemplo:

- En una base de datos de una universidad, la entidad **Estudiante** podría tener el atributo **ID_Estudiante** como clave primaria. Esto asegura que cada estudiante tenga un identificador único que lo distinga de otros estudiantes.
- En una entidad **Producto**, el atributo **Código de Producto** podría actuar como la clave primaria, asegurando que cada producto en la base de datos sea único y fácilmente identificable.

Claves Candidatas

Una **clave candidata** es cualquier atributo o combinación de atributos que puede actuar como una clave primaria porque cumple con las propiedades de unicidad y no nulidad. Sin embargo, aunque una entidad puede tener varias claves candidatas, solo una de ellas se selecciona como la clave primaria, mientras que las demás se dejan como claves candidatas.

Ejemplo:

- En una entidad **Empleado**, tanto **ID_Empleado** como **Número de Seguridad Social** podrían ser claves candidatas, ya que ambos atributos son únicos y no nulos para cada empleado. Sin embargo, uno de ellos, como **ID_Empleado**, se seleccionaría como la clave primaria, mientras que el otro permanecería como una clave candidata.

Claves Alternativas

Las **claves alternativas** son las claves candidatas que no se seleccionaron como clave primaria, pero que aún cumplen con los criterios de unicidad y no nulidad. Las claves alternativas siguen siendo propiedades identificatorias válidas y pueden utilizarse en situaciones donde se requiera otra forma de identificar de manera única a una instancia de una entidad.

Ejemplo:

- Siguiendo el ejemplo anterior, si **ID_Empleado** se selecciona como la clave primaria, entonces **Número de Seguridad Social** sería una clave alternativa en la entidad **Empleado**.

Clave Foránea (Foreign Key)

Una **clave foránea** es un atributo en una entidad que se refiere a la clave primaria de otra entidad. Las claves foráneas son esenciales para establecer relaciones entre diferentes entidades en un sistema de bases de datos. A través de las claves foráneas, se pueden vincular registros en una tabla con registros en otra tabla, lo que permite la implementación de relaciones uno a muchos o muchos a muchos.

Las claves foráneas también aseguran la **integridad referencial**, garantizando que los valores de las claves foráneas correspondan a valores existentes en las claves primarias de las entidades relacionadas.

Ejemplo:

- En una base de datos de ventas, la entidad **Pedido** podría tener un atributo **ID_Cliente** que actúe como clave foránea, haciendo referencia al atributo **ID_Cliente** en la entidad **Cliente**. Esto establece una relación entre los pedidos y los clientes.

Identificadores Compuestos

En algunos casos, no es posible utilizar un único atributo para identificar de manera única a una instancia de una entidad. En estos casos, se utiliza una **clave compuesta** o **identificador compuesto**, que combina dos o más atributos para formar una clave primaria única.

Ejemplo:

- En una entidad **Detalle_Pedido** que representa los productos en un pedido específico, los atributos **ID_Pedido** y **ID_Producto** juntos podrían formar una clave compuesta, ya que cada combinación de un pedido y un producto sería única, incluso si cada uno de estos atributos por separado no lo fuera.

Consideraciones sobre las Propiedades Identificatorias

1. **Rendimiento**: La elección de la clave primaria puede afectar el rendimiento de las consultas en la base de datos. Es importante seleccionar una clave primaria que no solo sea única, sino que también permita un acceso eficiente a los datos.
2. **Integridad**: Las propiedades identificatorias juegan un papel crucial en la **integridad de los datos**. Las claves primarias y foráneas ayudan a mantener la coherencia entre los datos relacionados en diferentes tablas.
3. **Normalización**: La identificación correcta de las propiedades identificatorias es fundamental para la normalización de la base de datos, un proceso que busca eliminar redundancias y dependencias anómalas en los datos.

Conclusión

Las **propiedades identificatorias** son fundamentales en el diseño de bases de datos, ya que garantizan que cada entidad pueda ser identificada de manera única y precisa. La selección adecuada de claves primarias, candidatas, alternativas y foráneas es esencial para la integridad, rendimiento y eficiencia de una base de datos. Estas propiedades también facilitan la creación de relaciones entre las entidades, lo que permite la organización estructurada y coherente de los datos en un sistema de bases de datos relacional.

2.1.6. Diagramas entidad-relación. Simbología.

Los **diagramas entidad-relación (E-R)** son representaciones gráficas que utilizan simbología específica para modelar las entidades, atributos y relaciones en un sistema de bases de datos. Estos diagramas son una herramienta esencial en la fase de diseño conceptual de una base de datos, ya que permiten a los diseñadores visualizar la estructura de los datos y las interacciones entre ellos de manera clara y comprensible.

A través de un diagrama E-R, se pueden representar de forma visual todos los componentes clave de una base de datos, incluidas las entidades, las relaciones entre ellas, los atributos que describen esas entidades y relaciones, así como las claves primarias y foráneas que permiten establecer conexiones y asegurar la integridad de los datos.

Simbología Básica en los Diagramas Entidad-Relación

1. **Entidades**: Las entidades se representan mediante **rectángulos**. Cada rectángulo representa una entidad del sistema, como `Empleado`, `Cliente` o `Producto`. Las entidades son los objetos principales del modelo y se utilizan para organizar los datos en categorías manejables y significativas.
2. **Relaciones**: Las relaciones entre entidades se representan mediante **rombos** o **diamantes**. Un rombo conecta dos o más entidades y define cómo interactúan entre sí. Por ejemplo, una relación llamada `Trabaja` podría conectar las entidades `Empleado` y `Departamento`, indicando que los empleados trabajan en departamentos.
3. **Atributos**: Los atributos de las entidades o relaciones se representan mediante **óvalos** conectados al rectángulo de la entidad o al rombo de la relación correspondiente. Por ejemplo, el atributo `Nombre` de la entidad `Empleado` se representaría como un óvalo conectado al rectángulo de `Empleado`.
4. **Clave Primaria**: La clave primaria de una entidad, que es el atributo o conjunto de atributos que identifican de manera única a cada instancia de la entidad, se indica subrayando el nombre del atributo en el óvalo. Por ejemplo, en una entidad `Empleado`, el atributo `ID_Empleado` podría ser la clave primaria y estaría subrayado en el diagrama.
5. **Clave Foránea**: Las claves foráneas, que establecen relaciones entre entidades al hacer referencia a las claves primarias de otras entidades, se representan con una línea que conecta la entidad que contiene la clave foránea con la entidad referenciada. Aunque no existe una simbología estándar específica para las claves foráneas en los diagramas E-R básicos, se indican mediante las relaciones entre las entidades.

6. **Atributos Multivalorados**: Los atributos que pueden tener más de un valor para una entidad se representan mediante un **óvalo doble**. Por ejemplo, si un empleado puede tener varios números de teléfono, el atributo `Teléfono` sería multivalorado y se representaría con un óvalo doble.
7. **Atributos Derivados**: Los atributos cuyo valor se puede calcular a partir de otros atributos se representan mediante un **óvalo punteado**. Por ejemplo, el atributo `Edad` puede derivarse del atributo `Fecha de Nacimiento`.

Simbología de Relaciones y Cardinalidades

1. **Cardinalidad**: La cardinalidad se refiere al número de instancias de una entidad que pueden estar asociadas con las instancias de otra entidad. En los diagramas E-R, la cardinalidad se representa mediante números o notaciones específicas colocadas junto a las líneas que conectan las entidades. Existen diferentes tipos de cardinalidades:
    - **1:1 (Uno a Uno)**: Una instancia de una entidad está relacionada con una sola instancia de otra entidad. Esto se indica con un "1" en ambos extremos de la línea de relación.
    - **1**
      (Uno a Muchos): Una instancia de una entidad puede estar relacionada con muchas instancias de otra entidad. Esto se indica con un "1" en un extremo y una "N" en el otro.
    - **M**
      (Muchos a Muchos): Muchas instancias de una entidad pueden estar relacionadas con muchas instancias de otra entidad. Esto se indica con una "M" y una "N" en los extremos de la relación.
2. La cardinalidad ayuda a entender las restricciones de las relaciones y cómo interactúan las entidades entre sí.
3. **Rol**: En algunas relaciones, es importante especificar el rol que desempeña cada entidad en la relación. El rol se suele indicar mediante un texto cercano a la línea de la relación que describe la función de la entidad dentro de la relación. Por ejemplo, en una relación de supervisión entre empleados, un rol podría ser `Supervisor` y el otro `Supervisado`.
4. **Grado de la Relación**: El grado de la relación se refiere al número de entidades involucradas en la relación. La mayoría de las relaciones en los diagramas E-R son **binarias** (involucran a dos entidades), pero también pueden existir relaciones **unarias** (donde una entidad se relaciona consigo misma) o **ternarias** (que involucran a tres entidades).

Diagramas Entidad-Relación Extendidos

En sistemas más complejos, se pueden utilizar **diagramas entidad-relación extendidos** (EER) que amplían la simbología básica para manejar conceptos adicionales, como jerarquías de generalización y especialización, agregaciones y categorías.

1. **Generalización y Especialización**: En una jerarquía de generalización o especialización, una entidad padre se subdivide en entidades hijas que heredan sus atributos. Por ejemplo, una entidad `Empleado` podría generalizarse en las entidades `Empleado Administrativo` y `Empleado de Producción`. En los diagramas EER, esto se representa con un triángulo o una flecha que conecta la entidad general con las entidades específicas.
2. **Agregación**: La agregación es un concepto que permite tratar una relación entre entidades como una sola entidad. Esto es útil cuando se desea representar relaciones complejas en niveles más altos de abstracción. En un diagrama EER, la agregación se representa encerrando la relación en un rectángulo adicional y tratándola como una entidad.

Ejemplo de Diagrama Entidad-Relación

Un ejemplo simple de diagrama E-R podría incluir las entidades `Empleado`, `Departamento` y `Proyecto`. La entidad `Empleado` tiene atributos como `ID_Empleado` (clave primaria), `Nombre` y `Salario`. La entidad `Departamento` tiene atributos como `ID_Departamento` y `Nombre_Departamento`. La relación entre `Empleado` y `Departamento` podría ser una relación de **1**

(un departamento tiene muchos empleados, pero cada empleado pertenece a un solo departamento). Además, `Empleado` podría estar relacionado con `Proyecto` en una relación **M**
(un empleado puede estar asignado a varios proyectos, y un proyecto puede involucrar a varios empleados).

Este diagrama ayudaría a los diseñadores a visualizar claramente cómo se estructuran los datos y cómo interactúan entre sí, lo que facilita la posterior implementación en un sistema de gestión de bases de datos (DBMS).

Conclusión

Los **diagramas entidad-relación (E-R)** y su simbología proporcionan una herramienta gráfica y conceptual poderosa para diseñar bases de datos. Al representar visualmente las entidades, relaciones y atributos, estos diagramas facilitan la comprensión de la estructura de la base de datos y ayudan a asegurar que el diseño conceptual sea claro y preciso antes de la implementación. Además, la simbología utilizada en los diagramas E-R es

estandarizada, lo que facilita la comunicación entre los diseñadores, desarrolladores y usuarios del sistema.

2.2. El modelo entidad-relación extendido.

El **Modelo Entidad-Relación Extendido (EER)** es una extensión del modelo entidad-relación (E-R) que introduce nuevos conceptos y herramientas para modelar situaciones más complejas y estructurar mejor los datos en una base de datos. Mientras que el modelo E-R básico es adecuado para la mayoría de los diseños de bases de datos, algunas aplicaciones requieren una mayor capacidad para representar datos jerárquicos, relaciones más complejas y especializaciones de entidades. Es en estas situaciones donde el modelo EER resulta especialmente útil.

El modelo entidad-relación extendido incluye conceptos como **generalización/especialización**, **jerarquías de entidades**, **herencia de atributos**, **agregación** y **categorías**. Estos conceptos amplían las capacidades del modelo E-R y proporcionan mayor flexibilidad para capturar estructuras de datos que se asemejan más al mundo real.

Generalización y Especialización

Uno de los conceptos clave del modelo EER es la **generalización** y **especialización**, que permite modelar jerarquías de entidades. Este concepto es útil cuando se tienen entidades que comparten algunas características, pero también tienen atributos o comportamientos únicos.

- **Generalización**: La generalización es el proceso de abstraer atributos comunes de varias entidades en una entidad padre más general. Por ejemplo, si tenemos las entidades `Empleado Administrativo` y `Empleado de Producción`, podemos generalizarlas en una entidad padre llamada `Empleado`, que contiene los atributos comunes de ambas subentidades, como `Nombre` y `Salario`.
- **Especialización**: La especialización es el proceso inverso a la generalización. Se utiliza para crear subentidades a partir de una entidad general, diferenciándolas según características o roles específicos. En el ejemplo anterior, `Empleado Administrativo` y `Empleado de Producción` son especializaciones de la entidad `Empleado`, y pueden tener atributos adicionales que no comparten entre sí, como `Oficina` para los empleados administrativos o `Máquina Asignada` para los empleados de producción.

La especialización se puede hacer de dos formas:

- **Exclusiva**: Cada instancia de la entidad general puede pertenecer solo a una subentidad especializada. Por ejemplo, un empleado puede ser o `Empleado Administrativo` o `Empleado de Producción`, pero no ambos.
- **Superpuesta**: Una instancia de la entidad general puede pertenecer a más de una subentidad especializada. Por ejemplo, en una organización, un `Empleado` podría ser tanto un `Empleado Administrativo` como un `Empleado de Producción` al mismo tiempo.

Jerarquías de Herencia

El modelo EER también permite la creación de **jerarquías de herencia**, donde las subentidades heredan atributos y relaciones de sus entidades padre. Esto es especialmente útil cuando se necesita representar relaciones "es un" (is-a). Por ejemplo, un `Empleado` es también una `Persona`, lo que significa que puede heredar atributos de la entidad `Persona`, como `Nombre`, `Fecha de Nacimiento` y `Género`.

Esta jerarquía de herencia asegura que las subentidades puedan aprovechar los atributos y relaciones definidas en la entidad padre sin necesidad de duplicar información. Además, permite a los diseñadores de bases de datos capturar de manera más precisa las relaciones entre los objetos del mundo real.

Agregación

La **agregación** es un concepto que permite tratar una relación entre varias entidades como una entidad por derecho propio. Este enfoque se utiliza cuando se necesita representar relaciones complejas de nivel superior entre varias entidades y sus relaciones asociadas. La agregación se utiliza cuando una relación es tan importante que se considera como una entidad en sí misma en otro nivel de abstracción.

Por ejemplo, supongamos que una entidad `Proyecto` está relacionada con las entidades `Empleado` y `Cliente`. La relación entre `Proyecto`, `Empleado` y `Cliente` puede considerarse como una entidad en sí misma, llamada `Participación en Proyecto`, y luego esta nueva entidad puede participar en relaciones adicionales en el modelo.

La agregación es útil para simplificar el modelo y estructurarlo de manera jerárquica, permitiendo una mayor claridad en la representación de relaciones complejas.

Categorías (Unión de Entidades)

Las **categorías**, también conocidas como **tipos de unión**, son un concepto del modelo EER que permite que una entidad pertenezca a varias entidades superpuestas. Las categorías se utilizan para representar situaciones en las que una entidad puede ser parte de múltiples conjuntos de entidades al mismo tiempo.

Por ejemplo, supongamos que en una base de datos que gestiona vehículos, existe la entidad `Vehículo`, y esta entidad puede ser un `Auto`, una `Motocicleta` o un `Camión`. En este caso, `Vehículo` es una categoría que unifica diferentes tipos de vehículos, permitiendo que una instancia de `Vehículo` pertenezca a cualquiera de estas subentidades.

Restricciones de Especialización y Generalización

En el modelo EER, también se pueden definir restricciones en las especializaciones y generalizaciones:

1. **Restricción de Cobertura**: Especifica si las especializaciones cubren todas las posibles instancias de la entidad general. Puede ser:
    - **Completa**: Todas las instancias de la entidad general deben pertenecer a alguna subentidad. Por ejemplo, todos los empleados deben ser o `Empleado Administrativo` o `Empleado de Producción`.
    - **Parcial**: Algunas instancias de la entidad general pueden no pertenecer a ninguna subentidad. Por ejemplo, un `Empleado` podría no clasificarse como `Empleado Administrativo` ni como `Empleado de Producción`.
2. **Restricción de Disyunción**: Define si una instancia de la entidad general puede pertenecer a más de una subentidad especializada:
    - **Disyunta (Mutuamente exclusiva)**: Una instancia solo puede pertenecer a una subentidad.
    - **Superpuesta (No exclusiva)**: Una instancia puede pertenecer a más de una subentidad.

Ejemplo de Modelo Entidad-Relación Extendido

Un ejemplo común de uso del modelo EER es en una base de datos para una organización que gestiona empleados. En este caso, podríamos tener:

- Una entidad general `Empleado` con atributos generales como `ID_Empleado`, `Nombre`, `Fecha de Contratación`.
- Especializaciones de `Empleado` como `Empleado Administrativo` y `Empleado de Producción`, donde cada especialización tiene atributos específicos.
- Una relación de generalización que conecta a `Empleado` con sus especializaciones.

- Una jerarquía de herencia que permite a las subentidades heredar los atributos de la entidad padre `Empleado`.
- Agregación de relaciones, donde la relación entre `Empleado` y `Proyecto` podría tratarse como una entidad agregada llamada `Participación en Proyecto`.

Conclusión

El **Modelo Entidad-Relación Extendido (EER)** amplía las capacidades del modelo E-R básico al introducir conceptos más avanzados como la generalización/especialización, jerarquías de herencia, agregación y categorías. Estos conceptos permiten representar estructuras de datos más complejas y alineadas con el mundo real, lo que resulta especialmente útil en aplicaciones empresariales y científicas que requieren modelar datos jerárquicos o relaciones complejas. El modelo EER proporciona a los diseñadores de bases de datos una mayor flexibilidad y potencia para crear modelos de datos que sean más expresivos y fáciles de mantener.

2.3. Restricciones de integridad:

Las **restricciones de integridad** son un conjunto de reglas que se aplican a los datos de una base de datos para asegurar que estos sean precisos, consistentes y válidos en todo momento. Estas restricciones garantizan que la información almacenada cumpla con las reglas del negocio y los requisitos lógicos definidos para el sistema de información. Las restricciones de integridad son fundamentales para mantener la calidad y fiabilidad de los datos y se implementan para evitar errores como duplicaciones, inconsistencias o referencias inválidas entre tablas.

En una base de datos relacional, las restricciones de integridad permiten asegurar que las relaciones entre las entidades se mantengan coherentes y que los valores almacenados cumplan con las expectativas del diseño. La violación de estas restricciones podría comprometer la integridad de los datos, lo que afectaría directamente la utilidad y funcionalidad del sistema.

Existen diferentes tipos de restricciones de integridad, que se pueden dividir en dos grandes categorías: **restricciones inherentes** y **restricciones explícitas**.

2.3.1. Restricciones inherentes.

Las **restricciones inherentes** son reglas de integridad que están implícitas en la estructura y naturaleza de una base de datos. Estas restricciones no necesitan ser declaradas explícitamente por el diseñador, ya que el sistema de gestión de bases de datos (DBMS) las aplica automáticamente. Su propósito es garantizar la coherencia de los datos a través de reglas que están directamente relacionadas con la definición de las entidades, atributos y relaciones dentro de la base de datos.

Estas restricciones están integradas en la estructura misma de la base de datos y reflejan características fundamentales del modelo relacional. A continuación, se describen algunas de las restricciones inherentes más comunes:

1. Unicidad de la Clave Primaria

La **clave primaria** es un atributo o conjunto de atributos que identifica de manera única a cada instancia de una entidad. La restricción de unicidad garantiza que no puede haber dos filas en una tabla con el mismo valor en la clave primaria. Esta es una restricción inherente porque la propia definición de clave primaria implica que su valor debe ser único en toda la tabla.

- **Ejemplo**: En una tabla `Empleado`, el atributo `ID_Empleado` podría ser la clave primaria. La restricción inherente asegura que cada empleado tenga un identificador único, evitando la duplicación de empleados en la base de datos.

2. No Permitir Valores Nulos en la Clave Primaria

Otra restricción inherente es que la clave primaria no puede contener valores nulos (`NULL`). Esto se debe a que un valor nulo representa la ausencia de datos, lo cual contradice el propósito de una clave primaria, que es identificar de manera única a cada registro. Dado que la clave primaria debe ser un identificador único, es lógico que no pueda ser nula.

- **Ejemplo**: Siguiendo con la tabla `Empleado`, el campo `ID_Empleado` no puede ser nulo, ya que cada empleado debe tener un identificador válido y único.

3. Cardinalidad en las Relaciones

La **cardinalidad** define cuántas instancias de una entidad pueden estar asociadas con instancias de otra entidad en una relación. Las relaciones entre entidades en una base de datos pueden ser de uno a uno (1:1), uno a muchos (1

) o muchos a muchos (M

). Las restricciones inherentes relacionadas con la cardinalidad garantizan que estas reglas se cumplan automáticamente.

- **Ejemplo**: En una relación entre `Departamento` y `Empleado`, donde un departamento puede tener muchos empleados pero cada empleado pertenece solo a un departamento (relación de uno a muchos), la restricción inherente asegura que cada empleado solo esté asociado a un único departamento.

4. Integridad de los Tipos de Datos

Los **tipos de datos** de los atributos definen las reglas inherentes sobre los valores que pueden almacenarse en una columna de una tabla. Estas restricciones aseguran que los datos ingresados respeten el tipo de dato especificado, como números enteros, cadenas de texto, fechas, entre otros.

- **Ejemplo**: Si un atributo `Salario` está definido como un número decimal, la base de datos rechazará cualquier intento de ingresar un valor de tipo texto o una fecha en ese campo.

5. Integridad Referencial Automática

Aunque la **integridad referencial** también puede establecerse explícitamente, el diseño relacional de la base de datos ya supone que las relaciones entre las tablas deben mantener coherencia entre las claves primarias y las claves foráneas. Esto significa que una clave foránea siempre debe referirse a una clave primaria válida en otra tabla, asegurando que no existan referencias a registros inexistentes.

- **Ejemplo**: Si una tabla `Pedido` tiene una clave foránea `ID_Cliente` que referencia la clave primaria `ID_Cliente` en la tabla `Cliente`, la base de datos no permitirá que se registre un pedido para un cliente inexistente.

Importancia de las Restricciones Inherentes

Las restricciones inherentes son esenciales para la integridad del modelo relacional de una base de datos. Al estar integradas en el sistema, estas reglas ayudan a prevenir errores comunes y aseguran que la estructura lógica de los datos se mantenga consistente sin requerir una supervisión constante. Las restricciones inherentes, al ser automáticas, reducen la necesidad de validaciones adicionales en las capas de aplicación y facilitan la gestión de la base de datos.

En resumen, las restricciones inherentes son las reglas fundamentales que un sistema de gestión de bases de datos aplica automáticamente para garantizar que la información almacenada cumpla con los principios básicos del diseño relacional, protegiendo la integridad y consistencia de los datos.

2.3.2. Restricciones explícitas.

Las **restricciones explícitas** son reglas de integridad que se definen de manera deliberada y directa en el diseño de la base de datos. A diferencia de las restricciones inherentes, que son automáticas y están implícitas en la estructura de la base de datos, las restricciones explícitas deben ser especificadas por el diseñador para cumplir con las reglas del negocio y los requisitos específicos del sistema. Estas restricciones aseguran que los datos no solo sean coherentes y válidos, sino que también reflejen las particularidades y necesidades específicas de la organización.

Las restricciones explícitas se implementan a través de las características del sistema de gestión de bases de datos (DBMS) y se aplican a nivel de tabla, columna o relación entre tablas. A continuación se describen los tipos más comunes de restricciones explícitas en una base de datos relacional:

1. Restricción de Integridad de Entidades (No Nulo)

La **restricción de integridad de entidades** asegura que ciertos atributos, como las claves primarias y otros atributos esenciales, no puedan tener valores nulos (`NULL`). Esta restricción se utiliza cuando es necesario que un campo siempre contenga un valor, garantizando que la información esté completa y sea significativa.

- **Ejemplo**: En una tabla `Empleado`, el campo `Nombre` puede tener una restricción `NOT NULL`, lo que significa que cada empleado debe tener un nombre registrado. Esta restricción garantiza que no haya empleados sin nombre en la base de datos.

2. Restricción de Unicidad (Unique)

La **restricción de unicidad** asegura que los valores en una columna o en un conjunto de columnas sean únicos en toda la tabla, es decir, que no se repitan. Esta restricción es útil para garantizar que ciertos atributos, además de la clave primaria, no tengan valores duplicados.

- **Ejemplo**: En una tabla `Empleado`, el atributo `Número de Seguridad Social` debe ser único para cada empleado, ya que no puede haber dos empleados con el

mismo número de seguridad social. Esta restricción se puede implementar mediante la cláusula `UNIQUE`.

3. Restricción de Integridad Referencial (Foreign Key)

La **restricción de integridad referencial** asegura que una clave foránea en una tabla siempre apunte a un valor válido en la tabla relacionada. Esta restricción garantiza que las relaciones entre tablas se mantengan coherentes y que no existan referencias a datos inexistentes.

- **Ejemplo**: En una tabla `Pedido`, el atributo `ID_Cliente` podría ser una clave foránea que debe hacer referencia a un valor existente en la tabla `Cliente`. Esto significa que no se puede insertar un pedido en la tabla `Pedido` sin que haya un cliente válido asociado en la tabla `Cliente`.

Además, la integridad referencial puede incluir reglas sobre cómo manejar la eliminación o actualización de registros en la tabla referenciada. Existen tres comportamientos comunes:

- **Restricción de acción**: Impide la eliminación o actualización de un registro si existe una referencia en otra tabla.
- **Propagación de la acción (Cascada)**: Si un registro en la tabla referenciada se elimina o actualiza, la acción se propaga automáticamente a las tablas relacionadas (por ejemplo, eliminando también los registros relacionados).
- **Anulación de la clave (SET NULL)**: Si un registro referenciado se elimina, la clave foránea en la tabla relacionada se establece a `NULL`.

4. Restricción de Valores (Check)

La **restricción de valores** se utiliza para validar que los valores insertados en una columna cumplan con una condición específica. Esto asegura que los datos ingresados respeten reglas específicas definidas por el negocio o el sistema.

- **Ejemplo**: En una tabla `Empleado`, se puede definir una restricción de verificación (`CHECK`) para asegurarse de que el valor del atributo `Salario` sea mayor que cero. Esta restricción evitaría la inserción de empleados con salarios negativos o nulos.

5. Restricciones de Verificación (Check Constraint)

Las **restricciones de verificación** permiten establecer condiciones más complejas que deben cumplir los valores de uno o varios atributos. Estas restricciones pueden involucrar

comparaciones entre diferentes columnas o con valores constantes y se usan para aplicar reglas más específicas y detalladas.

- **Ejemplo**: En una tabla `Producto`, se podría implementar una restricción `CHECK` que verifique que el `Precio` de un producto sea mayor que cero solo si el atributo `Disponible` está marcado como `Verdadero`. Esto asegura que solo los productos disponibles tengan un precio mayor a cero.

Ejemplos de Restricciones Explícitas

1. **Entidad Estudiante**:
   - **Restricción de No Nulo**: El atributo `Nombre` del estudiante no puede ser nulo, lo que asegura que cada estudiante tenga un nombre registrado.
   - **Restricción de Unicidad**: El atributo `Correo Electrónico` debe ser único para cada estudiante, evitando que se registren múltiples estudiantes con el mismo correo electrónico.
2. **Entidad Pedido**:
   - **Restricción de Integridad Referencial**: El atributo `ID_Producto` en la tabla `Detalle_Pedido` debe referirse a un producto existente en la tabla `Producto`.
   - **Restricción de Verificación**: El atributo `Cantidad` en `Detalle_Pedido` debe ser mayor que cero, garantizando que no se procesen pedidos con cantidades inválidas.
3. **Entidad Empleado**:
   - **Restricción de Unicidad**: El atributo `Número de Seguridad Social` debe ser único para cada empleado.
   - **Restricción de Verificación**: El atributo `Fecha de Nacimiento` debe ser menor que la fecha actual, asegurando que no se ingresen empleados nacidos en el futuro.

Importancia de las Restricciones Explícitas

Las restricciones explícitas son cruciales para el correcto funcionamiento de una base de datos, ya que permiten definir reglas específicas que reflejan las necesidades del negocio y protegen la integridad de los datos. Al establecer estas restricciones, los diseñadores de bases de datos pueden evitar problemas como la entrada de datos incorrectos o inválidos, la duplicación de registros o la creación de relaciones inconsistentes entre las tablas.

Además, las restricciones explícitas ayudan a reducir la cantidad de validaciones que deben realizarse en las capas superiores de la aplicación, trasladando esa responsabilidad al sistema de gestión de bases de datos. Esto mejora la eficiencia del sistema y garantiza que las reglas de negocio se apliquen de manera uniforme en toda la base de datos.

Conclusión

Las **restricciones explícitas** son herramientas poderosas en el diseño de bases de datos que permiten imponer reglas específicas para mantener la integridad y consistencia de los datos. Estas restricciones, como la integridad referencial, las reglas de unicidad y las validaciones de valores, garantizan que la base de datos respete las reglas del negocio y protegen la calidad de los datos almacenados. Al definir cuidadosamente estas restricciones, se mejora la fiabilidad del sistema y se asegura que los datos sean correctos y coherentes en todo momento.