PROMETEO

Unidad 3: Diseño lógico relacional

La transformación del modelo Entidad-Relación al modelo relacional es un proceso sistemático y fundamental que convierte nuestro diseño conceptual en una estructura que los SGBD relacionales pueden implementar directamente.

Sesión 9 – Transformación E-R → modelo relacional

La transformación del modelo Entidad-Relación al modelo relacional es un proceso sistemático y fundamental que convierte nuestro diseño conceptual en una estructura que los SGBD relacionales pueden implementar directamente. Este proceso, también conocido como **diseño lógico**, sigue un conjunto de reglas algorítmicas bien definidas que garantizan que toda la información y las relaciones capturadas en el diagrama E-R se preserven en el esquema relacional resultante.

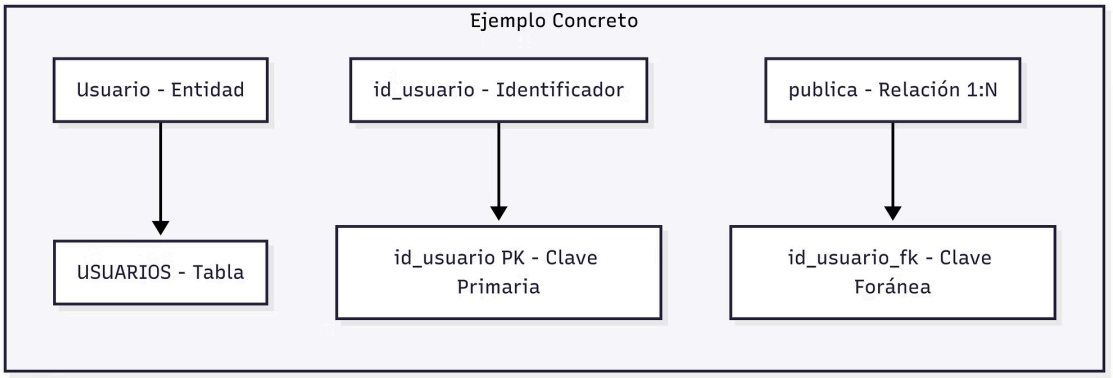
El **modelo relacional**, propuesto por Edgar F. Codd en 1970, representa todos los datos en forma de **tablas** (técnicamente llamadas **relaciones**), donde cada tabla es un conjunto de **filas** (llamadas **tuplas**) y **columnas** (llamadas **atributos**). La belleza del modelo relacional radica en su simplicidad conceptual: todo se reduce a tablas con filas y columnas, pero esta simplicidad permite representar dominios de información extremadamente complejos.

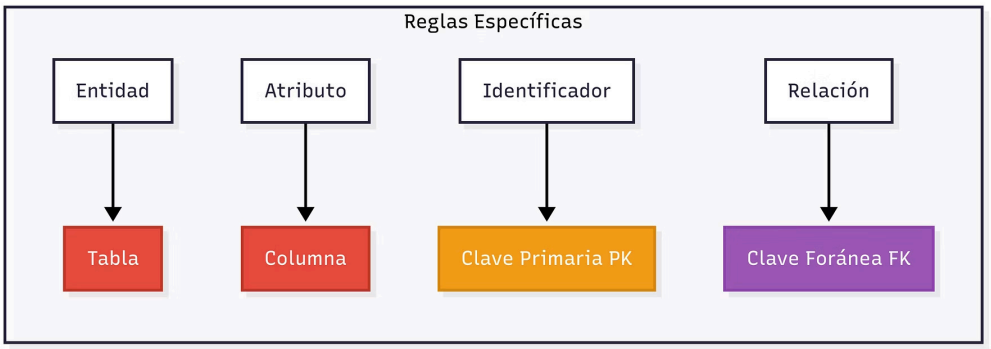
Las **reglas de transformación** son precisas y no ambiguas:

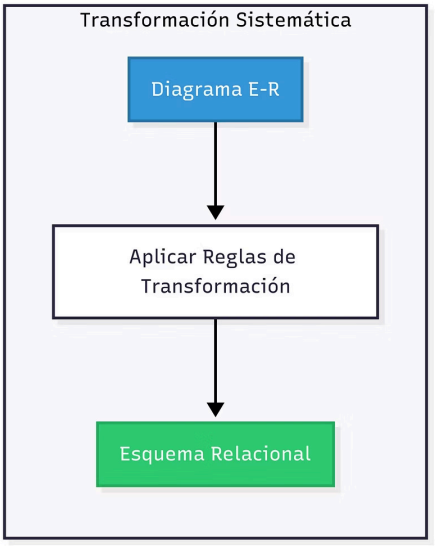
1. **Cada entidad del diagrama E-R se convierte en una tabla** en el esquema relacional. El nombre de la entidad se convierte en el nombre de la tabla.
2. **Cada atributo simple de una entidad se convierte en una columna** en su tabla correspondiente. Los atributos compuestos se descomponen en sus componentes simples.
3. **El identificador (clave candidata) de la entidad se convierte en la clave primaria (Primary Key - PK)** de la tabla. Esta clave primaria garantiza que cada fila de la tabla sea única e identificable.
4. **Las relaciones se implementan mediante claves foráneas (Foreign Keys - FK)**, que son columnas que contienen valores que apuntan a la clave primaria de otra tabla, estableciendo así el vínculo entre las tablas.

La **clave foránea** es el mecanismo fundamental que permite al modelo relacional representar las asociaciones que dibujamos como líneas en el diagrama E-R. Sin las claves foráneas, las tablas serían islas de información sin conexión entre sí.

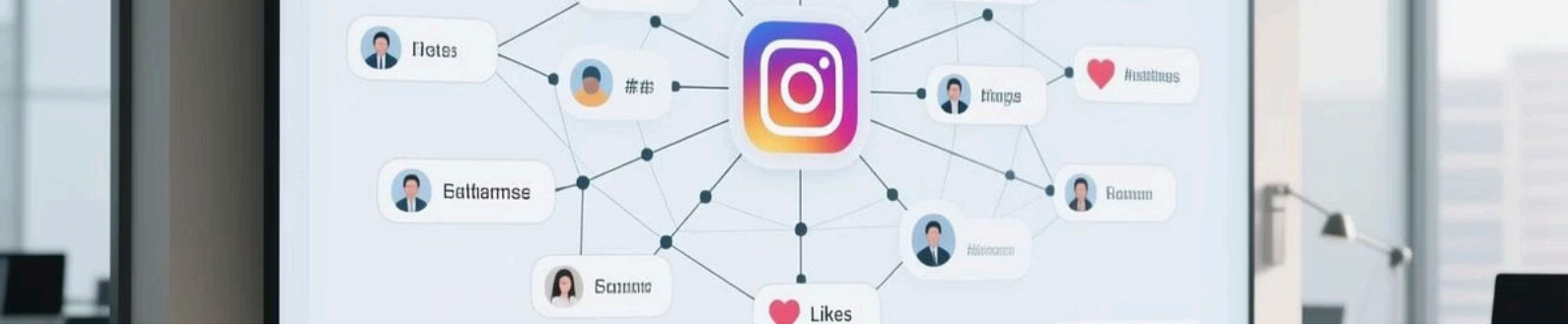
Esquema Visual







Este diagrama ilustra el **proceso sistemático** de transformación del modelo conceptual al lógico. En la parte superior, vemos el flujo general: el **Diagrama E-R** se procesa mediante **reglas de transformación** específicas para producir un **Esquema Relacional**. En el centro, se detallan las **reglas específicas**: cada elemento del modelo E-R tiene una correspondencia directa en el modelo relacional. En la parte inferior, se muestra un **ejemplo concreto**: la entidad Usuario se convierte en la tabla USUARIOS, su identificador id_usuario se convierte en la clave primaria, y la relación "publica" se implementa mediante la clave foránea id_usuario_fk en la tabla relacionada. Esta transformación preserva toda la semántica del modelo conceptual en una estructura implementable.



Caso de Estudio: La transformación del modelo de Instagram

El Contexto/Escenario: Instagram necesita implementar físicamente su modelo de datos para almacenar usuarios, fotos, comentarios y likes. El equipo de desarrollo ha completado el diseño conceptual E-R y ahora debe transformarlo en un esquema relacional que pueda implementarse en su SGBD.

La Estrategia/Ejecución: El proceso de transformación sigue las reglas sistemáticamente:

Transformación de Entidades:

- La entidad `Usuario` se convierte en la tabla `USUARIOS` con columnas: `id_usuario (PK)`, `nombre_usuario`, `email`, `fecha_registro`, `biografia`
- La entidad `Foto` se convierte en la tabla `FOTOS` con columnas: `id_foto (PK)`, `url_imagen`, `descripcion`, `fecha_publicacion`, `numero_likes`
- La entidad `Comentario` se convierte en la tabla `COMENTARIOS` con columnas: `id_comentario (PK)`, `texto`, `fecha_comentario`

Implementación de Relaciones mediante Claves Foráneas:

- La relación 1:N "un usuario publica muchas fotos" se implementa añadiendo `id_usuario_fk` a la tabla `FOTOS`
- La relación 1:N "una foto tiene muchos comentarios" se implementa añadiendo `id_foto_fk` a la tabla `COMENTARIOS`
- La relación 1:N "un usuario escribe muchos comentarios" se implementa añadiendo `id_usuario_fk` a la tabla `COMENTARIOS`

Manejo de Relaciones N:M:

- La relación N:M "usuarios dan like a fotos" requiere una tabla intermedia: `LIKES(id_usuario_fk, id_foto_fk, fecha_like)` donde la clave primaria es la combinación de ambas claves foráneas.

El Resultado/Impacto: Este esquema relacional permite a Instagram realizar todas las operaciones necesarias de forma eficiente. Para obtener todas las fotos de un usuario, simplemente consultan la tabla `FOTOS` donde `id_usuario_fk` coincida con el ID del usuario. Para mostrar los comentarios de una foto, consultan `COMENTARIOS` donde `id_foto_fk` coincida con el ID de la foto. Las claves foráneas actúan como el "pegamento" que mantiene toda la información relacionada y permite consultas complejas que combinan datos de múltiples tablas.

Herramientas y Consejos

1 Domina las reglas de transformación por cardinalidad

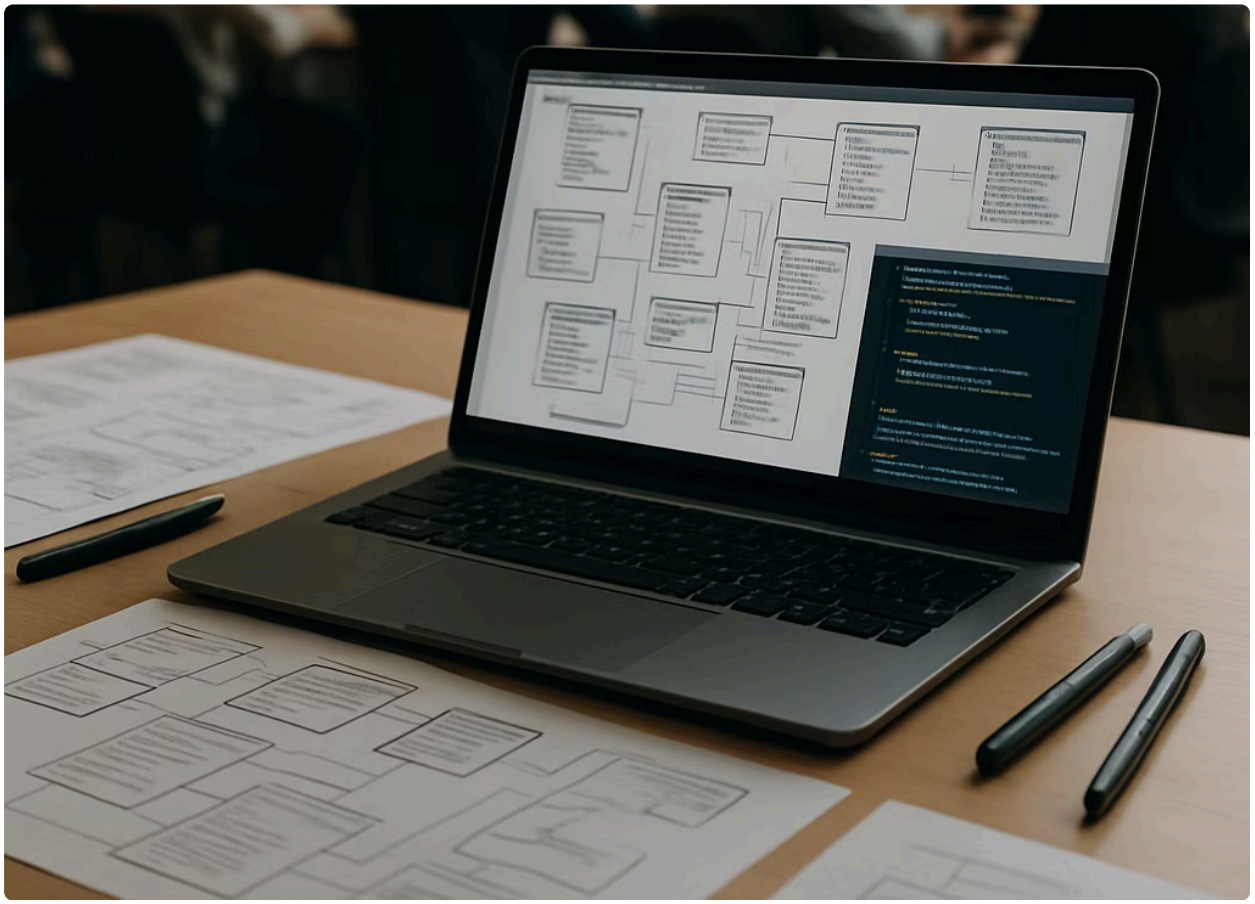
Cada tipo de relación tiene una regla específica de implementación. **Relaciones 1:1:** La clave primaria de una tabla se añade como clave foránea (y única) en la otra. **Relaciones 1:N:** La clave primaria de la tabla del lado "1" se añade como clave foránea en la tabla del lado "N". **Relaciones N:M:** Se crea una nueva tabla intermedia que contiene las claves foráneas de ambas tablas originales, y su clave primaria es típicamente la combinación de ambas claves foráneas.

2 Utiliza convenciones de nomenclatura consistentes

Adopta una convención clara para nombrar claves foráneas, como nombre_tabla_fk o id_nombre_tabla. Por ejemplo, si la tabla USUARIOS tiene la clave primaria id_usuario, la clave foránea que apunte a ella debería llamarse id_usuario_fk. Esta consistencia facilita enormemente la comprensión y el mantenimiento del esquema.

3 Aprovecha las herramientas de ingeniería directa

Herramientas como MySQL Workbench, Visual Paradigm o Enterprise Architect te permiten dibujar el diagrama E-R y luego, con un solo clic, generar automáticamente el esquema relacional y el código SQL correspondiente. Esto no solo ahorra tiempo, sino que también reduce errores y te permite visualizar inmediatamente el resultado de la transformación.



Mitos y Realidades

❌ **Mito:** "El modelo relacional es demasiado rígido y limitado para representar datos complejos del mundo real." → **FALSO.**

Aunque el modelo relacional puede parecer simple en la superficie (solo tablas con filas y columnas), es extraordinariamente potente y versátil. Mediante el uso inteligente de tablas, claves primarias, claves foráneas y las reglas de normalización, puede representar prácticamente cualquier dominio de problema de forma consistente, eficiente y robusta. La aparente simplicidad es en realidad una fortaleza que proporciona claridad y universalidad.

❌ **Mito:** "Puedo saltarme el diagrama E-R y diseñar el esquema relacional directamente, es más rápido." → **FALSO.**

Esta es una de las tentaciones más peligrosas en el diseño de bases de datos. Pensar directamente en términos de tablas y claves foráneas sin el mapa conceptual del E-R es significativamente más difícil, propenso a errores y resulta en diseños subóptimos. El diagrama E-R te permite pensar en el problema a un nivel de abstracción más alto, centrándote en las entidades y sus relaciones naturales, no en los detalles de implementación. Es la diferencia entre diseñar una casa con planos versus construir directamente con ladrillos.

Resumen Final

La transformación E-R → modelo relacional convierte el diseño conceptual en una estructura implementable mediante reglas sistemáticas: entidades se convierten en tablas, atributos en columnas, identificadores en claves primarias, y relaciones en claves foráneas. Las claves foráneas son el mecanismo fundamental para implementar las relaciones, con reglas específicas según la cardinalidad (1:1, 1:N, N:M).

Sesión 10 – Restricciones de integridad: dominio, entidad y referencial

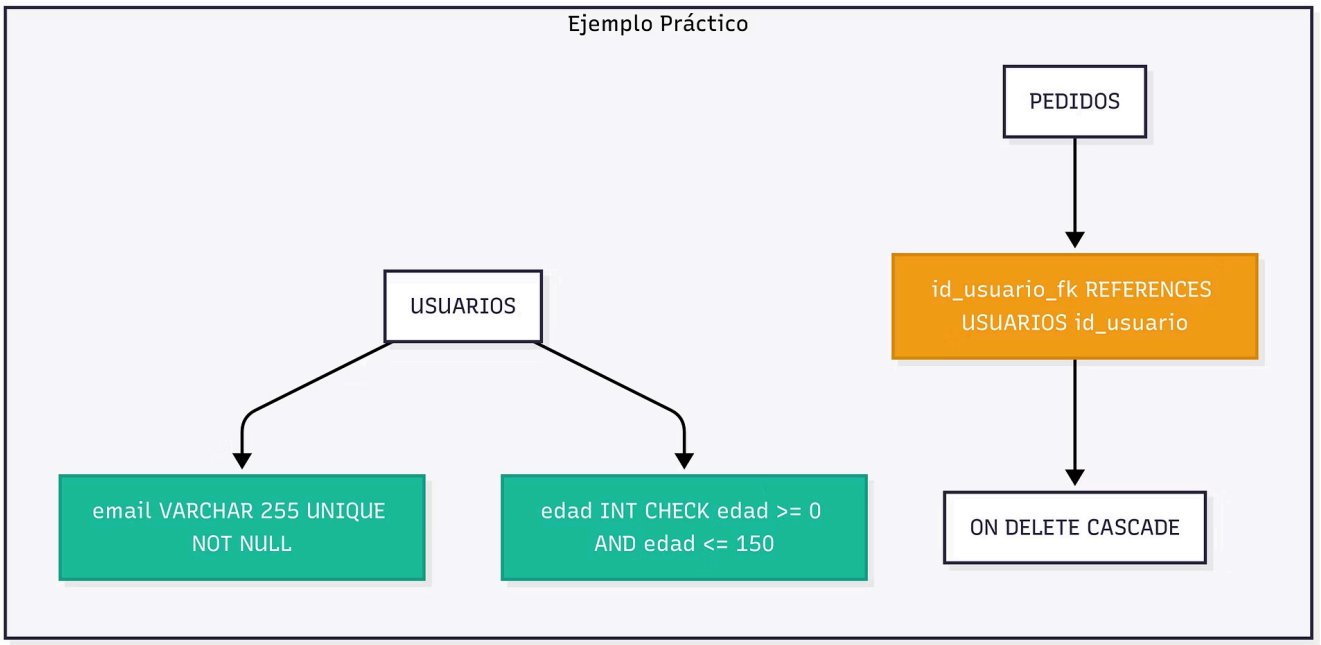
Un esquema relacional no está completo hasta que se definen las **restricciones de integridad**, que son las reglas que garantizan que los datos almacenados en la base de datos sean válidos, consistentes y confiables. Estas restricciones actúan como un sistema de control de calidad automático que previene la entrada de datos incorrectos o inconsistentes. Sin restricciones de integridad adecuadas, una base de datos puede convertirse rápidamente en un repositorio de información corrupta e inútil.

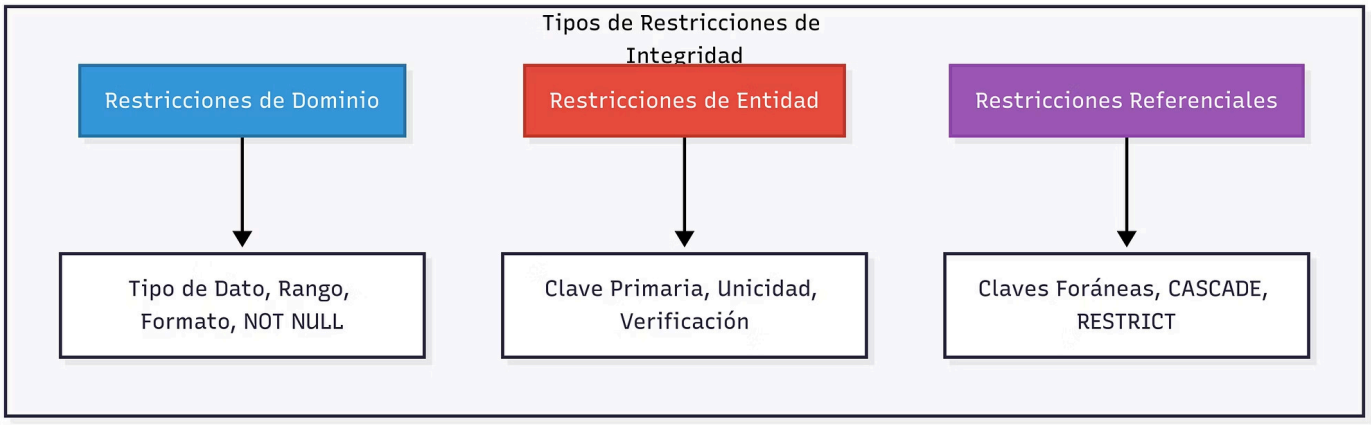
Existen tres tipos fundamentales de restricciones de integridad:

- **Restricciones de Dominio** definen qué valores son válidos para cada atributo individual. Incluyen el **tipo de dato** (entero, cadena, fecha, etc.), **restricciones de rango** (edad entre 0 y 150), **restricciones de formato** (email debe contener @), y **restricciones de nulidad** (NOT NULL para campos obligatorios). Estas restricciones aseguran que cada dato individual tenga sentido en su contexto.
- **Restricciones de Entidad** garantizan que cada fila en una tabla sea única e identificable. La restricción más importante es la **clave primaria**, que no puede ser nula y debe ser única para cada fila. También incluyen **restricciones de unicidad** para otros atributos que deben ser únicos (como el email de un usuario) y **restricciones de verificación** que validan condiciones complejas sobre múltiples atributos de la misma fila.
- **Restricciones de Integridad Referencial** mantienen la consistencia entre tablas relacionadas mediante claves foráneas. La regla fundamental es que **toda clave foránea debe apuntar a una clave primaria existente** en la tabla referenciada, o ser nula (si se permite). Estas restricciones previenen "referencias colgantes" donde una fila hace referencia a otra que no existe.

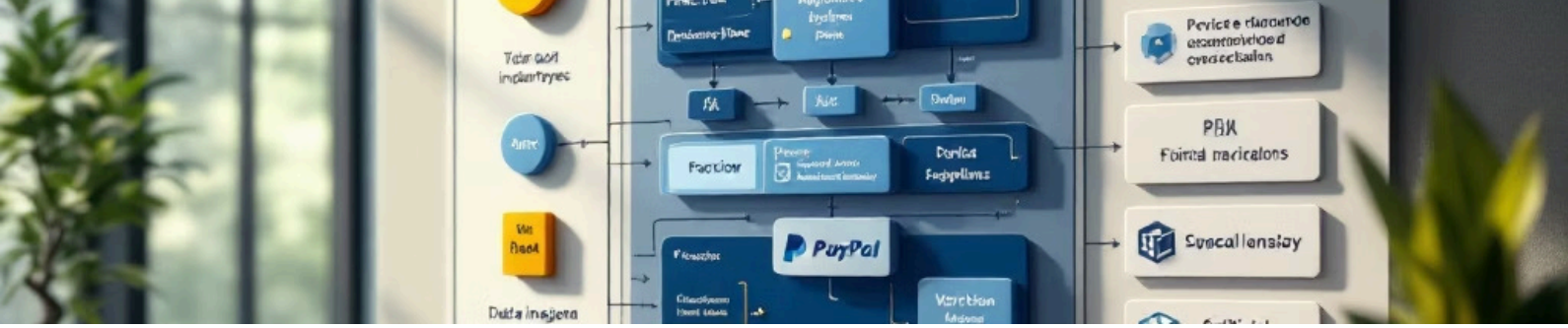
Además, las restricciones referenciales definen qué sucede cuando se intenta eliminar o modificar una fila que es referenciada por otras: **CASCADE** (propagar el cambio), **RESTRICT** (prohibir el cambio), **SET NULL** (establecer las referencias como nulas), o **SET DEFAULT** (establecer un valor por defecto).

Esquema Visual





Este diagrama ilustra los **tres tipos fundamentales** de restricciones de integridad y su aplicación práctica. En la parte superior, vemos las **categorías conceptuales**: las restricciones de dominio validan datos individuales, las de entidad aseguran la unicidad e identificabilidad de las filas, y las referenciales mantienen la consistencia entre tablas. En la parte inferior, se muestran **ejemplos concretos**: la tabla USUARIOS tiene restricciones de dominio (email válido y no nulo, edad en rango válido) y la tabla PEDIDOS tiene una restricción referencial que garantiza que cada pedido pertenezca a un usuario existente, con propagación en cascada si se elimina el usuario.



Caso de Estudio – El sistema de integridad de PayPal

El Contexto/Escenario: PayPal maneja transacciones financieras donde la integridad de los datos no es solo importante, es absolutamente crítica. Un error en los datos puede resultar en pérdidas financieras, problemas legales y pérdida de confianza de los usuarios. Su sistema de restricciones de integridad debe ser impecable.

La Estrategia/Ejecución: PayPal implementa un sistema multicapa de restricciones de integridad:

1 Restricciones de Dominio Estrictas

- Los montos monetarios utilizan tipos de datos DECIMAL con precisión fija para evitar errores de redondeo
- Los emails tienen validación de formato estricta con expresiones regulares
- Los números de tarjeta de crédito se validan usando el algoritmo de Luhn
- Las fechas de transacción tienen restricciones que impiden fechas futuras o anteriores a la creación de PayPal

2 Restricciones de Entidad Robustas

- Cada transacción tiene un ID único global que nunca se reutiliza
- Los emails de usuario deben ser únicos en todo el sistema
- Las cuentas bancarias se validan contra múltiples criterios (formato IBAN, existencia del banco, etc.)

3 Restricciones Referenciales Críticas

- Toda transacción debe referenciar cuentas de usuario válidas y existentes
- Las transferencias entre cuentas utilizan restricciones que garantizan que ambas cuentas existan
- Se implementan restricciones especiales para prevenir auto-transferencias (una cuenta no puede transferirse dinero a sí misma)
- Las restricciones de eliminación están configuradas como RESTRICT para cuentas con historial de transacciones, previniendo la pérdida accidental de datos financieros

El Resultado/Impacto: Este sistema de restricciones multicapa ha permitido a PayPal procesar billones de dólares en transacciones con una tasa de error extraordinariamente baja. Las restricciones actúan como múltiples redes de seguridad que capturan errores antes de que puedan causar problemas financieros. Cuando un usuario intenta realizar una transacción inválida, las restricciones la rechazan inmediatamente, proporcionando mensajes de error claros que permiten corregir el problema.

Herramientas y Consejos

Implementa restricciones en la base de datos, no solo en la aplicación

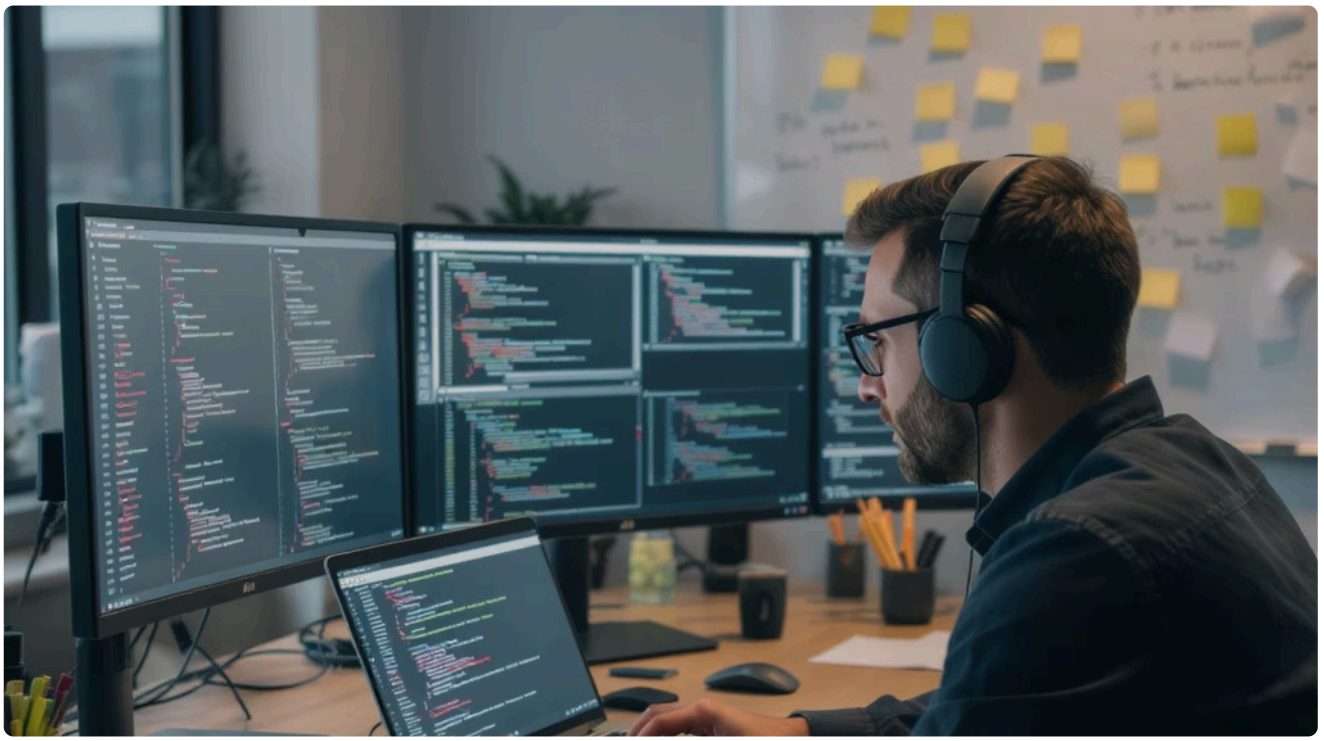
Aunque tu aplicación debe validar los datos antes de enviarlos a la base de datos, las restricciones en la base de datos son la última línea de defensa. Si los datos se modifican por otros medios (scripts de administración, otras aplicaciones, importaciones masivas), solo las restricciones de la base de datos pueden garantizar la integridad. Es un principio de defensa en profundidad.

Utiliza tipos de datos específicos y apropiados

No uses VARCHAR(255) para todo. Si un campo almacena un código postal de 5 dígitos, usa CHAR(5). Si almacena un booleano, usa BOOLEAN. Si almacena dinero, usa DECIMAL con precisión fija, no FLOAT. Los tipos de datos específicos no solo ahorran espacio, sino que también actúan como restricciones de dominio implícitas.

Diseña cuidadosamente las acciones referenciales

Las opciones CASCADE, RESTRICT, SET NULL y SET DEFAULT tienen implicaciones importantes. CASCADE puede ser peligroso si no se piensa cuidadosamente (eliminar un cliente podría eliminar todos sus pedidos). RESTRICT es más seguro pero puede impedir operaciones legítimas. Considera el contexto de negocio para cada relación.



Mitos y Realidades

❌ **Mito:** "Las restricciones de integridad ralentizan la base de datos, es mejor validar solo en la aplicación." → **FALSO.** Aunque las restricciones tienen un costo computacional mínimo, este es insignificante comparado con el costo de corregir datos corruptos. Además, las restricciones de integridad son extremadamente eficientes en los SGBD modernos. El costo de mantener la integridad es una fracción del costo de lidiar con datos inconsistentes, consultas erróneas y la pérdida de confianza en el sistema.

❌ **Mito:** "Si gestiono la integridad de los datos solo desde mi aplicación, tengo más control y flexibilidad." → **FALSO.** Esta estrategia es extremadamente arriesgada porque asume que tu aplicación es la única que accederá a los datos. En la realidad, los datos suelen ser accedidos por múltiples aplicaciones, scripts de administración, herramientas de reporting, procesos de migración, etc. Si las reglas de integridad solo existen en una aplicación, cualquier otro acceso puede corromper los datos. Las restricciones en la base de datos proporcionan una protección universal e independiente de la aplicación.

Resumen Final

Las restricciones de integridad garantizan la validez y consistencia de los datos mediante tres tipos: restricciones de dominio (validan datos individuales), restricciones de entidad (aseguran unicidad e identificabilidad), y restricciones referenciales (mantienen consistencia entre tablas). Deben implementarse en la base de datos como última línea de defensa, complementando las validaciones de la aplicación.

Sesión 11: Práctica: paso completo de modelo conceptual a lógico

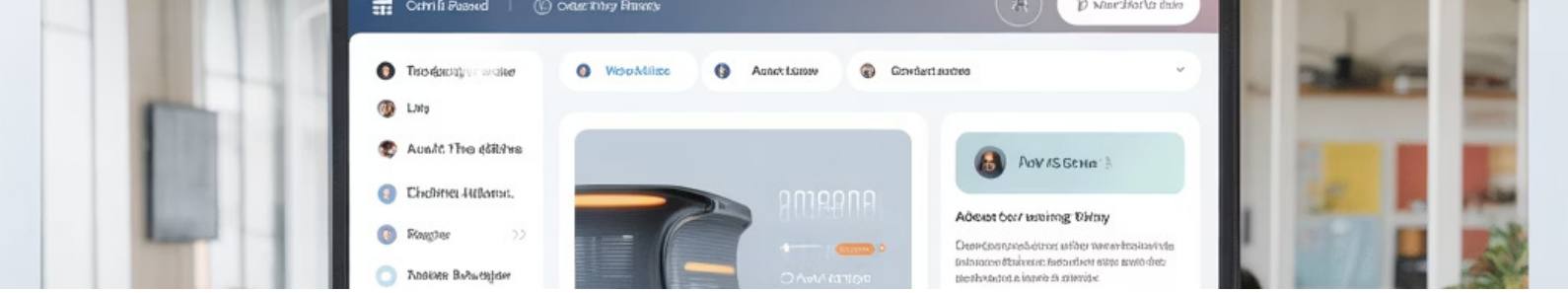
Esta sesión representa la culminación práctica de todo el aprendizaje sobre diseño lógico de bases de datos. Realizaremos el **proceso completo** desde la identificación de requisitos hasta la implementación de un esquema relacional con todas sus restricciones de integridad. Este ejercicio integral simula el flujo de trabajo real de un diseñador de bases de datos profesional y consolida la conexión crítica entre el pensamiento conceptual (cómo entendemos el problema) y la implementación técnica (cómo lo preparamos para el SGBD).

El **proceso sistemático** que seguiremos incluye múltiples fases interconectadas:

- **Fase 1 - Análisis de Requisitos:** Comprensión profunda del dominio del problema, identificación de actores, procesos de negocio y reglas que gobiernan el sistema. Esta fase es fundamental porque cualquier malentendido aquí se propagará a todo el diseño.
- **Fase 2 - Diseño Conceptual (E-R):** Traducción de los requisitos a un modelo conceptual que capture entidades, atributos, relaciones y cardinalidades. Este modelo debe ser comprensible para los stakeholders del negocio.
- **Fase 3 - Transformación a Modelo Lógico:** Aplicación sistemática de las reglas de transformación para convertir el modelo E-R en un esquema relacional con tablas, columnas, claves primarias y foráneas.
- **Fase 4 - Definición de Restricciones:** Especificación detallada de tipos de datos, restricciones de dominio, entidad y referenciales que garantizarán la integridad de los datos.
- **Fase 5 - Validación y Refinamiento:** Verificación de que el esquema resultante soporta todos los requisitos funcionales, es eficiente y mantenible.

Este proceso iterativo requiere tanto rigor técnico como pensamiento crítico, ya que las decisiones tomadas en cada fase afectarán la funcionalidad, el rendimiento y la mantenibilidad del sistema final.

Este diagrama ilustra el **flujo completo** del proceso de diseño lógico y los **artefactos** que se producen en cada etapa. En la parte superior, vemos el proceso iterativo que va desde los requisitos iniciales hasta el diseño final, con un bucle de retroalimentación crucial que permite refinar el diseño si no cumple con los requisitos. En la parte inferior, se muestran los **artefactos tangibles** que se producen: el diagrama E-R (representación visual del modelo conceptual), el esquema de tablas (estructura relacional), la definición de restricciones (reglas de integridad), y finalmente el código SQL DDL (Data Definition Language) que puede ejecutarse directamente en el SGBD para crear la base de datos física.



Caso de Estudio – Diseño completo para una plataforma de blogs como Medium

El Contexto/Escenario: Necesitamos diseñar la base de datos para una plataforma de publicación de contenido donde los autores escriben artículos, los lectores pueden comentar y seguir a autores, y el contenido se organiza mediante etiquetas. El sistema debe soportar múltiples tipos de contenido y permitir interacciones sociales básicas.

La Estrategia/Ejecución: Aplicamos el proceso completo paso a paso:

01

Análisis de Requisitos Detallado

- Los Autores se registran y publican Artículos
- Los Artículos tienen título, contenido, fecha de publicación y pueden estar en borrador o publicados
- Los Lectores pueden comentar artículos y seguir autores
- Los artículos se categorizan mediante Etiquetas (relación N:M)
- Se necesita rastrear las estadísticas de lectura y engagement

02

Diseño del Modelo E-R

Entidades: Usuario, Artículo, Comentario, Etiqueta, Seguimiento

Relaciones: Usuario "escribe" Artículo (1:N), Usuario "comenta" Artículo (N:M a través de Comentario), Artículo "tiene" Etiqueta (N:M), Usuario "sigue" Usuario (N:M)

03

Transformación a Esquema Relacional

USUARIOS(id_usuario PK, nombre_usuario, email, fecha_registro, biografia)
ARTICULOS(id_articulo PK, titulo, contenido, fecha_publicacion, estado, id_autor_fk)
COMENTARIOS(id_comentario PK, texto, fecha_comentario, id_usuario_fk, id_articulo_fk)
ETIQUETAS(id_etiqueta PK, nombre_etiqueta, descripcion)
ARTICULO_ETIQUETA(id_articulo_fk, id_etiqueta_fk) -- PK compuesta
SEGUIMIENTOS(id_seguidor_fk, id_seguido_fk, fecha_seguimiento) -- PK compuesta

04

Definición de Restricciones de Integridad

- email en USUARIOS: UNIQUE NOT NULL con validación de formato
- estado en ARTICULOS: CHECK (estado IN ('borrador', 'publicado'))
- Todas las claves foráneas con ON DELETE CASCADE apropiado
- fecha_publicacion solo puede ser NOT NULL si estado = 'publicado'

El Resultado/Impacto: Este esquema relacional completo proporciona la base técnica para implementar una plataforma de blogging robusta. Soporta todas las funcionalidades requeridas (publicación, comentarios, etiquetado, seguimiento), mantiene la integridad de los datos mediante restricciones apropiadas, y es escalable para manejar grandes volúmenes de contenido y usuarios. La estructura es lo suficientemente flexible para evolucionar con nuevos requisitos futuros.

Herramientas y Consejos

Documenta exhaustivamente tu proceso de diseño

Mantén un registro detallado de las decisiones tomadas y las razones detrás de ellas. Esto incluye por qué elegiste ciertas cardinalidades, por qué separaste o combinaste entidades, y por qué implementaste restricciones específicas. Esta documentación será invaluable para futuras modificaciones y para comunicar el diseño a otros desarrolladores.

Valida el diseño con casos de uso reales

Para cada requisito funcional del sistema, traza exactamente cómo se implementaría usando tu esquema. Por ejemplo: "¿Cómo publico un nuevo artículo?", "¿Cómo obtengo todos los comentarios de un artículo?", "¿Cómo encuentro todos los artículos de un autor que sigo?". Si no puedes responder estas preguntas de forma eficiente con tu esquema, necesitas refinarlo.

Utiliza herramientas de prototipado rápido

Herramientas como dbdiagram.io, SQLiteStudio, o incluso MySQL Workbench te permiten crear rápidamente el esquema y poblarlo con datos de prueba para validar que funciona como esperas. No subestimes el valor de crear un prototipo funcional con datos reales para probar tu diseño.

Mitos y Realidades

❌ **Mito:** "Un diseñador experimentado puede crear el esquema perfecto en el primer intento sin iteraciones." → **FALSO.**

Incluso los arquitectos de datos más experimentados pasan por múltiples iteraciones y refinamientos. El diseño de bases de datos es un proceso de descubrimiento donde la comprensión del dominio se profundiza gradualmente. Los mejores diseños emergen de la iteración consciente, la validación continua con stakeholders, y la disposición a refinar el modelo basándose en nueva información o requisitos que emergen durante el proceso.

❌ **Mito:** "El modelo lógico es solo un paso intermedio; puedo ir directamente del E-R al código SQL." → **FALSO.**

El modelo lógico es un artefacto crucial que sirve como documentación, herramienta de comunicación entre equipos técnicos y de negocio, y fuente de verdad para el desarrollo. Saltarse esta fase de documentación formal resulta en sistemas mal documentados, difíciles de mantener y propensos a malentendidos entre desarrolladores. El modelo lógico es la especificación técnica que guía toda la implementación posterior.

Resumen Final

El proceso completo de diseño lógico incluye análisis de requisitos, modelado E-R conceptual, transformación sistemática a esquema relacional, definición de restricciones de integridad, y validación iterativa. Produce artefactos documentados (diagrama E-R, esquema de tablas, restricciones, código DDL) que sirven como especificación técnica para la implementación. Es un proceso iterativo que requiere validación continua con casos de uso reales.