Compás
capacitación e investigación

2025

DISEÑO LÓGICO DE BASE DE DATOS.

Enfoques conceptuales, metodológicos y prácticos para la formación universitaria

Freddy Anibal Jumbo Castillo
Mariuxi Paola Zea Ordoñez
Oscar Efrén Cárdenas Villavicencio

DISEÑO LÓGICO DE BASE DE DATOS.

Enfoques conceptuales, metodológicos y prácticos para la formación universitaria

Freddy Anibal Jumbo Castillo
Mariuxi Paola Zea Ordoñez
Oscar Efrén Cárdenas Villavicencio

ISBN: 978-9942-53-027-1 DOI: http://doi.org/10.48190/9789942530271

Compás
capacitación e investigación

ORCID: 0000-0002-5200- 7162
fjumbo@utmachala.edu.ec.
Universidad Técnica de Machala.

Mariuxi Paola Zea Ordoñez
ORCID: 0000-0001-8860- 6282
mzea@utmachala.edu.ec
Universidad Técnica de Machala

Oscar Efrén Cárdenas Villavicencio
ORCID: 0000-0001- 6570-8040
oecardenas@utmachala.edu.ec
Universidad Técnica de Machala

Primera edición, 2025-11-09

ISBN: 978-9942-53-027-1
DOI: http://doi.org/10.48190/9789942530271
Distribución online
Acceso abierto

Cita
Jumbo. F., Zea, M., Cárdenas, O. (2025) DISEÑO LÓGICO DE BASE DE DATOS. Enfoques conceptuales, metodológicos y prácticos para la formación universitaria. Editorial Grupo Compás

Este libro es parte de la colección de la Univesidad Técnica de Machala y ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad de la publicación.

Prefacio

Introducción del libro

El diseño lógico de bases de datos constituye una competencia fundamental en la formación de profesionales en Tecnologías de la Información, Ingeniería en Computación y áreas afines. En una era donde los datos se han consolidado como un activo estratégico para las organizaciones; una percepción compartida por más del 90 % de las empresas[1]. Comprender cómo estructurar la información de forma coherente, eficiente y segura resulta indispensable para garantizar la calidad, integridad y escalabilidad de los sistemas de información.

El volumen de datos a escala global ha experimentado un crecimiento exponencial, lo cual proyecta que para el año 2025 se alcanzarán 180 zettabytes, lo que pone en evidencia la magnitud y el desafío de gestionar, almacenar y analizar volúmenes masivos de información[2]. Este fenómeno está directamente relacionado con la adopción de sistemas de gestión de bases de datos, empleados actualmente por al menos tres de cada cuatro empresas. No es casual que este mercado tenga un valor estimado superior a los 63 mil millones de dólares en 2025[3], reflejando su centralidad en la economía digital contemporánea.

Este libro responde a dicha necesidad formativa con un enfoque actualizado, riguroso y didácticamente estructurado. Su contenido guía al lector desde los fundamentos conceptuales del modelado de datos, hasta las técnicas más avanzadas de diseño lógico, integrando tanto la teoría del Modelo Entidad-Relación (MER) como los principios del modelo relacional, las reglas de normalización y las restricciones de integridad.

A lo largo de sus capítulos, se promueve una comprensión progresiva y aplicada del proceso de diseño, partiendo del análisis conceptual de requerimientos, pasando por la representación gráfica de entidades, atributos y relaciones, hasta llegar a la construcción de esquemas lógicos listos para ser implementados en Sistemas de Gestión de Bases de Datos (DBMS). Se incluyen además casos prácticos contextualizados, reflexiones críticas, ejemplos con sintaxis técnica y recursos pedagógicos orientados a consolidar las competencias del estudiante en escenarios reales de gestión de información.

Más allá de su finalidad académica, esta obra busca formar profesionales capaces de diseñar bases de datos robustas, sostenibles y alineadas con las

[1] Passivesecrets.com. (2025). 20+ interesting data management statistics & trends (2025). Passivesecrets.com.
[2] G2. (2024). *85+ big data statistics to map growth in 2025*. G2.
[3] LLCBuddy.com. (2025). Database management systems statistics 2025. LLCBuddy.com.

necesidades del entorno organizacional, aportando al desarrollo de soluciones tecnológicas de calidad en un mundo cada vez más orientado al dato.

Objetivo general del libro

Proporcionar una base sólida de conocimientos teóricos y prácticos en diseño lógico de bases de datos relacionales, orientada al fomento en los estudiantes de competencias clave para el análisis de requerimientos de información, la modelación de estructuras de datos y la construcción de esquemas eficientes que respondan a necesidades reales de gestión.

Estructura del libro

El contenido de este texto se organiza en cuatro unidades secuenciales que guían al estudiante desde la comprensión conceptual de los fundamentos del diseño de bases de datos hasta su aplicación técnica en escenarios reales. Cada unidad está planteada de manera progresiva, articulando teoría y práctica para consolidar competencias en análisis, modelado y diseño lógico de estructuras de datos relacionales.

- **Unidad 1. Fundamentos del Diseño de Bases de Datos.**

  Esta primera unidad establece las bases conceptuales necesarias para comprender el papel de los datos en los sistemas informáticos. Se abordan las diferencias entre dato e información, los distintos tipos de bases de datos, las funciones de un sistema gestor de bases de datos (DBMS) y las etapas del ciclo de vida de una base de datos.

- **Unidad 2. Modelado Conceptual de Datos.**

  A partir de los fundamentos, la segunda unidad introduce el Modelo Entidad-Relación (MER) y sus extensiones, como herramientas centrales para representar, de manera abstracta, los elementos y relaciones del mundo real que conforman un dominio de aplicación. Esta transición permite pasar de la teoría básica a la capacidad de estructurar conceptualmente la información.

- **Unidad 3. Diseño Lógico Relacional.**

  Sobre la base del modelado conceptual, la tercera unidad se centra en la transformación del modelo E-R hacia el modelo relacional. Aquí se integran las dependencias funcionales, las reglas de normalización y las restricciones de integridad que aseguran la coherencia de los datos. De esta manera, el estudiante conecta el análisis conceptual con un esquema relacional estructurado y listo para su posterior uso en entornos técnicos.

- **Unidad 4. Diseño Físico de la Base de Datos.**

  Finalmente, la cuarta unidad aborda los aspectos operativos del diseño físico, incluyendo mecanismos de almacenamiento, uso de índices, gestión de seguridad, optimización del rendimiento y estrategias de respaldo. Esta sección marca la transición del diseño lógico a la ejecución práctica y el despliegue en un Sistema de Gestión de Bases de Datos (SGBD), vinculando el conocimiento teórico con la experiencia de implementación en escenarios profesionales.

Características pedagógicas

Este libro ha sido diseñado con criterios didácticos actuales que facilitan el aprendizaje significativo:

- Objetivos de aprendizaje claros al inicio de cada unidad.
- Diagramas explicativos, esquemas y ejemplos reales que facilitan la comprensión visual.
- Actividades prácticas, ejercicios de autoevaluación y casos aplicados en cada unidad.
- Glosarios temáticos que refuerzan el dominio del vocabulario técnico.
- Tipografía destacada y señalética que resalta conceptos clave, advertencias y buenas prácticas.

El enfoque es progresivo, integrador y orientado a la práctica profesional, con el fin de consolidar habilidades analíticas y técnicas en el campo del diseño de bases de datos.

Índice

Unidad 1: Fundamentos del Diseño de Bases de Datos

Introducción de la Unidad

En la actualidad, la gestión eficiente de los datos constituye una competencia transversal y estratégica en todos los ámbitos productivos, académicos y sociales. Las organizaciones dependen cada vez más de información confiable, organizada y accesible para tomar decisiones informadas, optimizar procesos y generar valor. En este contexto, las bases de datos no son únicamente herramientas tecnológicas, sino sistemas complejos que permiten estructurar el conocimiento, facilitar el análisis y garantizar la integridad de los datos.

Esta unidad ofrece una introducción integral a los fundamentos del diseño lógico de bases de datos, brindando al estudiante los conceptos clave necesarios para comprender el papel que desempeñan en los sistemas de información. Se abordarán temas esenciales como la diferencia entre dato e información, la importancia estratégica de los datos en la sociedad digital, el concepto y características de una base de datos, sus componentes estructurales, los roles involucrados en su gestión, y el ciclo de vida que rige su desarrollo. Además, se analizarán las funciones del Sistema de Gestión de Bases de Datos (DBMS), las distintas clasificaciones de bases de datos (por modelo, ubicación, contenido y propósito), así como ejemplos reales de aplicación en sectores clave como salud, educación, banca, comercio electrónico y administración pública.

El conocimiento adquirido en esta unidad sentará las bases conceptuales necesarias para avanzar hacia el diseño conceptual, lógico y físico de bases de datos, facilitando así la construcción de sistemas de información robustos, coherentes y adaptados a los requerimientos del mundo actual.

Objetivos de aprendizaje

Al finalizar esta unidad, el estudiante será capaz de:

1. Establecer una distinción clara y fundamentada entre los conceptos de dato e información, reconociendo su relevancia para el análisis, organización y procesamiento de contenidos en los sistemas informáticos.
2. Valorar la importancia estratégica que tienen los datos en la sociedad actual, comprendiendo su rol como activos organizacionales esenciales en la toma de decisiones, la innovación y la transformación digital.

3. Conceptualizar qué es una base de datos y justificar su utilidad frente a otros mecanismos de almacenamiento, identificando sus ventajas en términos de integridad, eficiencia y gestión multiusuario.
4. Identificar y describir los componentes fundamentales que conforman un sistema de base de datos, incluyendo su estructura física y lógica, así como la interacción entre sus elementos técnicos y humanos.
5. Analizar las funciones principales que desempeña un sistema gestor de base de datos (DBMS) en la administración, consulta, integridad y seguridad de la información dentro de un entorno organizacional.
6. Clasificar los distintos tipos de bases de datos existentes, atendiendo a criterios como el modelo de datos, la finalidad, el contenido, la estructura y el entorno de aplicación.
7. Reconocer los roles profesionales involucrados en el ciclo de vida de una base de datos, comprendiendo sus responsabilidades específicas y su impacto en el desarrollo y operación del sistema.
8. Explicar con profundidad las fases que componen el ciclo de vida de una base de datos, desde el análisis de requerimientos hasta su mantenimiento, destacando la importancia de un enfoque metodológico y sistemático.

Preguntas de enfoque

- ¿Qué diferencia a un dato de una información útil?
- ¿Qué papel cumplen las bases de datos en la sociedad digital?
- ¿Cuáles son los elementos básicos de un sistema de base de datos?
- ¿Cómo se relacionan los distintos actores que participan en su desarrollo?

Desarrollo de contenidos

1.1. Dato e Información

Los conceptos de dato e información constituyen la piedra angular de cualquier sistema informático. Aunque en el lenguaje cotidiano estos términos tienden a emplearse de manera indistinta, en el ámbito técnico y científico adquieren significados diferenciados y complementarios que resultan esenciales para el diseño y gestión de sistemas de información.

¿Qué es un dato?

Un dato es una representación simbólica de un atributo o variable, susceptible de ser almacenada, transmitida y procesada. Por sí solo, carece de significado

intrínseco: se trata de un fragmento de la realidad captado en forma numérica, textual, gráfica o de otra naturaleza, pero aún no interpretado.

De acuerdo con Coronel & Morris (2023), "los datos representan hechos observados o medidas recolectadas sobre personas, objetos, eventos o procesos". Un dato puede manifestarse como una cifra ("25"), una palabra ("azul"), una fecha ("2025-04-08") o cualquier otro elemento elemental capturado. Por ejemplo: El valor "23" puede referirse a la edad de un estudiante, al número de matrícula de un curso o a la cantidad de libros prestados en una biblioteca, dependiendo del contexto en que se utilice.

¿Qué es la información?

La información es el producto del procesamiento, organización y contextualización de los datos, permitiendo otorgarles significado y valor para el usuario. Es decir, es un conjunto de datos interpretados que posibilita comprender fenómenos, reducir la incertidumbre y fundamentar la toma de decisiones. Según Laudon & Laudon (2022), "la información es un conjunto de datos procesados que adquieren significado y valor para el usuario".

Ejemplo: Si conocemos que "23" corresponde a la edad de una estudiante llamada María, quien cursa la carrera de Ingeniería, entonces estamos ante información, ya que se ha agregado contexto y significado.

Diferencias clave

Mientras el dato es bruto y objetivo, la información es el resultado de un proceso de construcción que la convierte en un recurso útil. Esta distinción es crucial en el diseño de bases de datos, pues define qué elementos deben almacenarse, cómo deben estructurarse y de qué manera serán utilizados posteriormente. En la **Tabla 1.1,** se realiza la comparación entre dato e información.

Tabla 1.1. Comparación entre dato e información.

Concepto	**Dato**	**Información**
Definición	Representación simbólica sin contexto	Dato interpretado, organizado y contextualizado
Naturaleza	Bruto, objetivo	Procesado, significativo
Utilidad	Limitada en sí misma	Alta: permite comprender y tomar decisiones
Ejemplo	"1990"	"Pedro nació en 1990 en Machala"

Nota. Contenido adaptado de (Rowley, 2007) y (Elmasri & Shamkant, 2007).

Relevancia para las bases de datos

Distinguir claramente entre datos e información resulta esencial en el diseño lógico de sistemas de bases de datos. Si bien las bases almacenan datos, su propósito final es producir información significativa. Por ello, la organización eficiente, segura y coherente de los datos es determinante para posibilitar la generación de reportes, visualizaciones y análisis pertinentes.

Nota destacada: Una base de datos deficiente puede acumular grandes cantidades de datos sin lograr convertirlos en información útil. La clave radica en estructurarlos en función de los requerimientos informativos del sistema.

1.2. Importancia de los datos en la sociedad actual

Según Castells (2005), vivimos en una época caracterizada por la omnipresencia de los datos. Esta etapa, denominada sociedad de la información, se define por la producción, procesamiento y transmisión masiva de datos a través de tecnologías digitales.

En este nuevo contexto, los datos se han transformado en activos estratégicos para múltiples sectores. Las empresas los utilizan para conocer mejor a sus clientes, optimizar procesos y anticipar comportamientos. Los gobiernos los emplean para diseñar políticas públicas y monitorear indicadores sociales. La academia, por su parte, los convierte en insumo para la generación de conocimiento científico y la validación de hipótesis.

Los datos constituyen la base de la pirámide del conocimiento, representando la materia prima que, tras procesos de estructuración y análisis, se transforma en información, luego en conocimiento y finalmente en sabiduría. Este modelo, conocido como DIKW (Data, Information, Knowledge, Wisdom), ha sido ampliamente adoptado para explicar la evolución progresiva del valor de los datos (Rowley, 2007).

La proliferación de dispositivos conectados, redes sociales, sensores inteligentes y sistemas digitales ha dado origen al fenómeno del Big Data, caracterizado por el volumen, la velocidad y la variedad de los datos (Oussous et al., 2018). Este ecosistema ha impulsado el desarrollo de técnicas de Inteligencia Artificial (IA) y aprendizaje automático (machine learning), las cuales dependen críticamente de grandes volúmenes de datos para entrenar modelos y generar predicciones útiles.

Como señala Hoz Morales (2018), "la capacidad de analizar grandes volúmenes de datos ha cambiado la forma en que los gobiernos, las empresas y los ciudadanos entienden el mundo y toman decisiones". Los datos, por tanto, no solo registran eventos pasados, sino que permiten prever comportamientos futuros.

Más allá del ámbito empresarial o gubernamental, el ciudadano común participa diariamente en la cultura del dato. Aplicaciones móviles que monitorizan la actividad física, asistentes virtuales que recomiendan contenido personalizado o plataformas de e-commerce que adaptan sus ofertas son ejemplos cotidianos de cómo los datos median nuestra vida diaria.

Ante este escenario, se vuelve crucial fomentar la alfabetización digital y estadística, así como la defensa de principios éticos como la privacidad, la transparencia algorítmica y el consentimiento informado (Floridi & Taddeo, 2016).

Nota destacada: En la economía digital contemporánea, los datos son comparables al petróleo: valiosos, pero dependientes de procesos de refinamiento, almacenamiento, distribución y uso ético para liberar su potencial de manera sostenible (Murillo, 2021). Sin datos de calidad, no es posible construir información precisa ni tomar decisiones fundamentadas. De ahí la creciente inversión global en infraestructuras para el almacenamiento, procesamiento y protección de datos.

La comprensión del valor estratégico de los datos convierte al diseñador de bases de datos en un actor central de la gobernanza informacional. No se trata únicamente de construir estructuras eficientes, sino también de garantizar que los datos recolectados sean pertinentes, veraces y transformables en información valiosa para los usuarios.

Reflexión: ¿Cómo se vería afectada una decisión si se basara en datos incompletos o mal organizados? Diseñar bases de datos es también una forma de estructurar el conocimiento.

1.3. ¿Qué es una base de datos?

Una base de datos es un sistema estructurado destinado a almacenar, organizar y gestionar grandes cantidades de datos de manera eficiente y segura. Constituye una colección coherente de información relacionada, organizada para facilitar su acceso, manipulación y análisis por parte de múltiples usuarios.

Según Date (2003), “una base de datos es una colección persistente y lógicamente coherente de datos con algún significado inherente, organizada para satisfacer las necesidades de múltiples usuarios”. Esta definición enfatiza que una base de datos no es simplemente un cúmulo de datos, sino un sistema diseñado con propósitos específicos.

Coronel & Morris (2023) refuerzan esta visión al definir una base de datos como “una estructura que permite almacenar, gestionar y recuperar datos de forma eficiente, manteniendo su integridad y facilitando el acceso controlado y seguro por parte de los usuarios autorizados”.

Esto significa que las bases de datos son herramientas indispensables para gestionar grandes volúmenes de información, especialmente en entornos donde se requiere precisión, disponibilidad, confidencialidad y control de acceso.

Características de una base de datos: Las bases de datos modernas presentan una serie de características distintivas:

- **Persistencia:** Los datos permanecen almacenados de forma permanente.
- **Coherencia:** Se mantienen reglas de integridad para asegurar la validez de los datos.
- **Compartibilidad:** Facilitan el acceso concurrente por múltiples usuarios o sistemas.
- **Seguridad:** Aplican mecanismos robustos de autenticación y autorización.
- **Reducción de redundancia:** Mediante procesos de normalización, evitan la duplicación innecesaria de datos.

Estas características se administran a través de un Sistema de Gestión de Bases de Datos (DBMS), que se explorará en detalle en la sección 1.5.

Componentes básicos de una base de datos: Toda base de datos incluye los siguientes elementos fundamentales:

- **Tablas:** Estructuras que almacenan datos relacionados de una entidad específica.
- **Filas (registros):** Representan instancias individuales de la entidad modelada.
- **Columnas (atributos):** Definen propiedades o características de las entidades.
- **Claves primarias y foráneas:** Garantizan la unicidad y las relaciones entre tablas.
- **Índices:** Mejoran la velocidad de las consultas y recuperaciones de datos.
- **Vistas:** Representaciones virtuales que simplifican el acceso a datos complejos.

¿Por qué son necesarias las bases de datos?

La creciente dependencia de la información en las organizaciones ha consolidado a las bases de datos como infraestructuras críticas para:

- Gestionar grandes volúmenes de datos estructurados.
- Asegurar accesos rápidos y selectivos a la información.
- Proteger los datos mediante medidas de seguridad avanzadas.
- Facilitar el trabajo simultáneo de múltiples usuarios sobre los mismos conjuntos de datos.
- Automatizar procesos dependientes de información actualizada y confiable.

Como sostiene Date (2003), "una base de datos bien diseñada mejora la eficiencia operativa, la calidad de la información y la capacidad de toma de decisiones de una organización". Antes de la popularización de las bases de datos, muchos sistemas utilizaban archivos planos para almacenar datos. Sin embargo, estos presentan importantes limitaciones, lo cual se puede observar en la **Tabla 1.2**:

Tabla 1.2. Comparación entre archivos planos y bases de datos.

Aspecto	**Archivos planos**	**Bases de datos**
Redundancia	Alta	Mínima (bien diseñada)
Seguridad	Baja o nula	Elevada, mediante control de accesos
Integridad de datos	No garantizada	Gestionada por el DBMS
Consultas complejas	Difíciles de implementar	Optimizadas a través de SQL
Escalabilidad	Limitada	Alta

Nota. Elaboración propia a partir de (Coronel & Morris, 2023) y (Date, 2003).

Ejemplo de base de datos: En una base de datos universitaria, la tabla "estudiantes" puede contener columnas como identificador o código, nombre, carrera y fecha de Ingreso, donde cada fila representa un alumno único.

Nota destacada: Una base de datos supera ampliamente a simples listas o hojas de cálculo; constituye una estructura compleja, concebida para responder a requerimientos técnicos y funcionales precisos.

Reflexión: Una base de datos mal estructurada puede derivar en duplicación de información, pérdida de registros, errores en la toma de decisiones y graves vulnerabilidades de seguridad.

1.4. Componentes de un sistema de base de datos

Un sistema de base de datos no se limita únicamente a los datos que almacena, sino que integra un conjunto de elementos tecnológicos y humanos cuya interacción posibilita su diseño, implementación, operación y mantenimiento.

Comprender estos componentes resulta esencial para valorar la complejidad inherente a los sistemas de bases de datos y para organizar eficazmente el proceso de desarrollo de soluciones robustas.

De acuerdo con Coronel & Morris (2023), "un sistema de base de datos está conformado por una combinación de hardware, software, personas, procedimientos y datos, que trabajan conjuntamente para capturar, almacenar, procesar y distribuir información significativa para una organización". Los componentes principales de un sistema de base de datos son:

a. **Datos:** Los datos constituyen el recurso central de todo sistema de base de datos. Representan hechos relevantes del mundo real que han sido organizados con el propósito de facilitar su acceso, manipulación y análisis. Pueden clasificarse en:

   - **Datos estructurados:** almacenados en tablas relacionales tradicionales.
   - **Datos semiestructurados:** representados en formatos flexibles como XML o JSON.
   - **Datos no estructurados:** contenidos en documentos, imágenes, videos o archivos de audio.

   El diseño adecuado de una base de datos debe garantizar la integridad, coherencia, seguridad y disponibilidad permanente de los datos.

b. **Metadatos:** Los metadatos son, en esencia, "datos sobre los datos". Constituyen la información que describe la estructura, los contenidos, las restricciones y los comportamientos de los elementos almacenados en la base de datos. Entre los aspectos que engloban se encuentran:

   - Definiciones de tablas y columnas.
   - Tipos de datos y formatos.
   - Restricciones como NOT NULL o UNIQUE**.**
   - Índices, claves primarias y claves foráneas.
   - Procedimientos almacenados y vistas.

   Estos metadatos son administrados internamente por el Sistema de Gestión de Bases de Datos (DBMS) y se almacenan en el catálogo del sistema, también conocido como diccionario de datos, que sirve como referencia fundamental para todas las operaciones de gestión y consulta.

c. **Software:** El componente de software en un sistema de base de datos está encabezado por el Sistema de Gestión de Bases de Datos (DBMS),

conjunto de programas responsables de crear, mantener y consultar la información. Ejemplos populares de DBMS incluyen MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server y MongoDB. El ecosistema de software también puede incluir:

- Interfaces gráficas de usuario o herramientas de línea de comandos.
- Lenguajes de consulta como **SQL**.
- Herramientas especializadas para diseño, respaldo y replicación de bases de datos.

Como sostiene Date (2003), "el DBMS es el componente esencial que proporciona los mecanismos de abstracción, independencia, seguridad e integridad de los datos".

d. Hardware: El hardware abarca todos los recursos físicos necesarios para almacenar y procesar los datos, tales como:

- Servidores de bases de datos, que alojan los datos de manera centralizada.
- Dispositivos de almacenamiento, incluyendo discos duros, arreglos RAID o servicios de almacenamiento en la nube.
- Redes y dispositivos de comunicación, que facilitan el acceso remoto y simultáneo de múltiples usuarios.

El rendimiento y la capacidad del hardware inciden directamente en la eficiencia del sistema, especialmente en contextos de alta demanda o de procesamiento de Big Data.

e. Personas

El factor humano resulta indispensable en el ciclo de vida de un sistema de base de datos. Los distintos perfiles involucrados incluyen:

- **Administrador de bases de datos (DBA):** responsable de la configuración, mantenimiento y seguridad del sistema.
- **Diseñador de bases de datos:** encargado de modelar estructuras lógicas y definir reglas de integridad.
- **Desarrolladores de aplicaciones:** quienes integran la base de datos con sistemas de software.
- **Usuarios finales:** quienes interactúan con el sistema para consultar o ingresar información.

La colaboración efectiva entre estos roles es crítica para asegurar la correspondencia entre el diseño técnico y los requerimientos funcionales de la organización.

f. Procedimientos

Los procedimientos comprenden las políticas, reglas y rutinas operativas que regulan el funcionamiento del sistema de base de datos. Entre ellos se incluyen:

- Estrategias de respaldo y recuperación ante fallos.
- Políticas de seguridad, autenticación y autorización.
- Procesos de auditoría, monitoreo y mantenimiento preventivo.
- Protocolos para migraciones o actualizaciones de sistemas.

La correcta implementación de estos procedimientos asegura que la operación del sistema sea predecible, segura y conforme a los estándares organizacionales y normativos vigentes.

Todos los componentes del sistema interactúan de manera interdependiente. Por ejemplo, un usuario final accede a los datos a través de una aplicación de software, la cual se comunica con el DBMS que opera sobre un hardware específico, bajo políticas de seguridad y procedimientos definidos.

Reflexión: El diseño lógico de una base de datos constituye apenas una parte del sistema. La creación de soluciones exitosas exige la armonización integral de componentes técnicos, humanos y organizativos, reconociendo su mutua dependencia en la búsqueda de eficiencia, seguridad y valor estratégico.

1.5. El Sistema de Gestión de Bases de Datos (DBMS)

El Sistema de Gestión de Bases de Datos, conocido por sus siglas en inglés como DBMS o en español como SGBD, constituye el componente esencial de cualquier sistema moderno de base de datos. Se trata de un conjunto de programas que permite definir, construir, manipular y compartir datos entre múltiples usuarios y aplicaciones de forma segura, eficiente y controlada.

De acuerdo con Elmasri & Shamkant (2007), "un DBMS es una colección de programas que permite a los usuarios crear y mantener una base de datos. El sistema facilita la definición de estructuras, la manipulación de datos y su control, preservando la integridad y la seguridad de la información".

El DBMS desempeña diversas funciones críticas para el manejo efectivo de los datos, abarcando desde su definición hasta su recuperación segura en caso de fallos.

a. **Definición de datos:** Esta función permite especificar la estructura lógica de la base de datos, es decir, las tablas, columnas, claves primarias, claves foráneas, restricciones de integridad y relaciones existentes entre los distintos elementos. La definición de estructuras se realiza mediante lenguajes de definición de datos como DDL (Data Definition Language).

b. **Manipulación de datos:** Los usuarios pueden insertar, actualizar, eliminar y consultar datos a través de lenguajes de manipulación de datos (DML), siendo SQL (Structured Query Language) el estándar más ampliamente utilizado. Esta funcionalidad posibilita la creación de interfaces dinámicas, informes personalizados y análisis complejos basados en la información almacenada.

c. **Seguridad y control de acceso:** El DBMS regula el acceso a la información mediante mecanismos de autenticación de usuarios y asignación de permisos diferenciados. Puede restringir operaciones como lectura, escritura o actualización en función de los perfiles definidos, garantizando así la protección de datos sensibles o confidenciales.

d. **Control de concurrencia:** Gestiona el acceso simultáneo de múltiples usuarios a los mismos datos, asegurando que las transacciones se ejecuten de manera aislada y coherente. El control de concurrencia evita problemas como las condiciones de carrera o las lecturas inconsistentes en ambientes multiusuario.

e. **Manejo de transacciones:** De acuerdo con Date (2003), el DBMS asegura que las operaciones involucradas en una transacción se realicen en su totalidad o no se realicen en absoluto, cumpliendo las propiedades ACID: Atomicidad, Consistencia, Aislamiento y Durabilidad. Este manejo resulta fundamental para preservar la integridad de los datos en escenarios de fallo parcial.

f. **Recuperación ante fallos:** Incluye estrategias para restaurar la base de datos a un estado consistente en caso de errores de sistema, fallos eléctricos o problemas de hardware. Estos mecanismos se apoyan en el registro de transacciones y en procedimientos de respaldo periódico.

g. **Mantenimiento de la integridad:** El DBMS aplica automáticamente las reglas de integridad definidas en el esquema de la base de datos, asegurando la validez de los datos mediante restricciones como claves únicas, atributos obligatorios (NOT NULL) y relaciones referenciales entre tablas.

Ventajas del uso de un DBMS: La implementación de un DBMS ofrece numerosos beneficios para las organizaciones, optimizando la gestión de la

información y potenciando la eficiencia operativa. En la **Tabla 1.3**, se describen las ventajas del uso de un motor de base de datos:

Tabla 1.3. Beneficios de la implementación de un DBMS en las organizaciones.

Ventajas	**Descripción**
Centralización del control	Permite administrar los datos de forma unificada, minimizando duplicidades.
Reducción de redundancia	Diseños normalizados eliminan la duplicación innecesaria de información.
Acceso concurrente	Facilita el acceso seguro y simultáneo de múltiples usuarios.
Recuperación de errores	Proporciona mecanismos de respaldo, registro de transacciones y restauración ante fallos.
Escalabilidad	Se adapta a sistemas de gran tamaño, distribuidos o con altos volúmenes de datos.
Interoperabilidad	Facilita la conexión con distintos lenguajes de programación, APIs, sistemas de Business Intelligence, entre otros.

Nota. Contenido adaptado de Database Systems: Design, Implementation, & Management (Coronel & Morris, 2023).

La potencialidad de un DBMS radica en que, si en una base de datos existe una tabla de "pedidos" vinculada a una tabla de "clientes" mediante una clave foránea (cli_id), el motor impedirá la inserción de pedidos que hagan referencia a clientes inexistentes, salvaguardando así la integridad referencial.

Clasificación de los DBMS: Los sistemas de gestión de bases de datos pueden clasificarse atendiendo a distintos criterios, como el modelo de datos que emplean o su arquitectura de despliegue:

- **Relacionales (RDBMS):** Basados en el modelo de tablas y relaciones. Ejemplos: MySQL, PostgreSQL, Oracle Database, SQL Server.
- **No relacionales (NoSQL):** Adaptados al manejo de datos no estructurados o semiestructurados. Ejemplos: MongoDB, Cassandra, Redis.
- **Orientados a objetos:** Incorporan conceptos del paradigma orientado a objetos. Ejemplo: db4o.

- **Distribuidos:** Almacenan los datos en múltiples ubicaciones físicas, pero operan como una única base lógica. Ejemplo: Google Spanner.
- **Basados en la nube (Cloud DBMS):** Ofrecen servicios gestionados en plataformas como AWS, Azure o Google Cloud.

DBMS populares: La **Tabla 1.4,** resume algunas características distintivas de los DBMS más representativos en el mercado:

Tabla 1.4. Características de DBMS populares.

DBMS	Tipo	Características destacadas
MySQL	Relacional	Ligero, rápido y de código abierto.
PostgreSQL	Relacional	Robusto, extensible, con soporte avanzado de SQL.
Oracle Database	Relacional	Comercial, altamente escalable y con altos estándares de seguridad.
MongoDB	NoSQL	Basado en documentos, flexible, ideal para manejar datos en formato JSON.
SQL Server	Relacional	Integrado en el ecosistema de Microsoft, óptimo para soluciones de BI y ERP.

Nota. Contenido adaptado de (Coronel & Morris, 2023).

Nota destacada: La elección del DBMS más adecuado dependerá de múltiples factores, entre ellos: la naturaleza y volumen de los datos, la cantidad de usuarios concurrentes, los requerimientos de seguridad y el presupuesto disponible para el proyecto.

1.6. Clasificación de las bases de datos

Las bases de datos pueden ser clasificadas de múltiples maneras, atendiendo a diversos criterios tales como el modelo de datos que emplean, su ubicación física o lógica, el tipo de contenido que gestionan, o el propósito específico para el cual han sido diseñadas. Esta clasificación resulta fundamental al momento de seleccionar la tecnología más adecuada para responder a las necesidades particulares de una organización o proyecto.

Como sostienen Elmasri & Shamkant (2007), "la clasificación de las bases de datos es esencial para entender la diversidad de necesidades que existen en los sistemas de información actuales y para seleccionar tecnologías apropiadas de almacenamiento y consulta de datos".

Clasificación según el modelo de datos: Uno de los criterios más utilizados para clasificar las bases de datos es el modelo de datos sobre el cual se estructura la información. Este modelo define la manera en que se organizan, representan y relacionan los datos, así como las reglas que rigen su manipulación.

a. **Bases de datos jerárquicas:** Organizan los datos siguiendo una estructura de tipo árbol (padre-hijo), donde cada nodo puede tener múltiples hijos, pero únicamente un padre (**ver Figura 1.1**). Este modelo resulta altamente eficiente en operaciones de lectura cuando la estructura de los datos es estable y predecible. Sin embargo, experimentan la incapacidad para representar relaciones complejas entre entidades y su rigidez estructural limitan su aplicabilidad en entornos modernos, donde la flexibilidad es un requisito fundamental. Un ejemplo de este tipo de base de datos jerárquicas es IBM Information Management System (IMS).

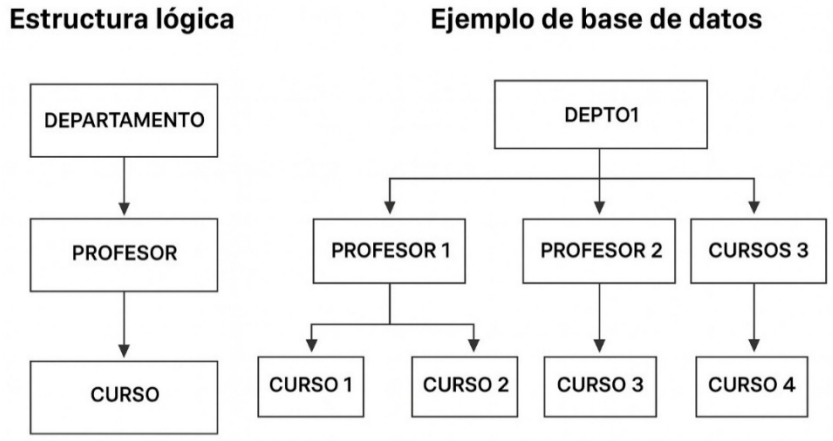


Figura 1.1. Ejemplo de estructura jerárquica de base de datos.
Fuente: Tomado de Diseño e implementación de un lexicón computacional para lexicografía y traducción automática, por (Moreno Ortiz, 2000).

b. **Bases de datos en red:** Este modelo amplía el esquema jerárquico permitiendo que un nodo hijo pueda estar vinculado a múltiples nodos padres, habilitando la representación de relaciones de muchos a muchos. Esta mayor flexibilidad, sin embargo, introduce una complejidad adicional en la navegación de los datos. Aunque ofrecen ventajas frente a las bases jerárquicas, su complejidad motivó su reemplazo progresivo por modelos de datos más sencillos y manejables. La Integrated Data Store (IDS) de Honeywell es un ejemplo dentro de este tipo de base de datos. Su representación gráfica se puede apreciar en la **Figura 1.2.**

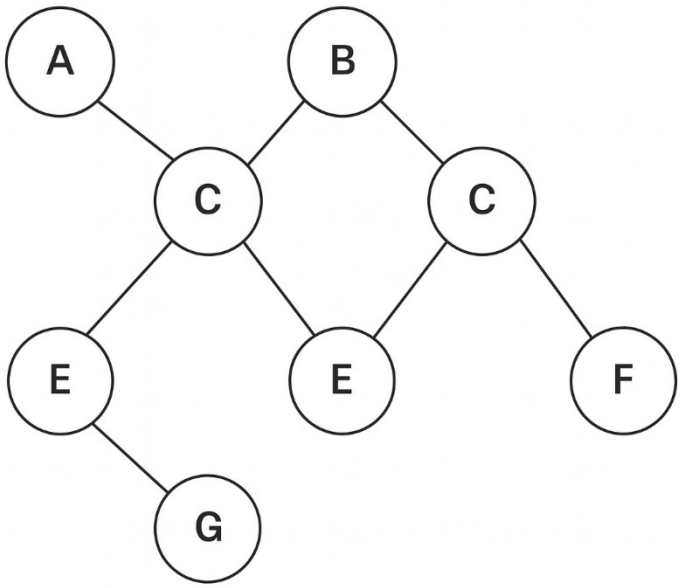


Figura 1.2. Ejemplo de estructura de base de datos en red.
Fuente: Tomado de Diseño e implementación de un lexicón computacional para lexicografía y traducción automática, por (Moreno Ortiz, 2000).

c. **Bases de datos relacionales:** Actualmente constituyen el modelo dominante en el mercado. Organizan los datos en tablas o relaciones compuestas por filas y columnas, donde cada tabla representa una entidad del mundo real, y las relaciones entre ellas se expresan a través de claves primarias y foráneas. Este modelo, propuesto por (Codd, 1970), se convirtió en el estándar gracias a su simplicidad conceptual, poder expresivo y facilidad de consulta (Date, 2003). Entre los ejemplos de este tipo de base de datos son: MySQL, PostgreSQL, Oracle Database, SQL Server. La representación gráfica de este tipo de base de datos se puede observar en la **Figura 1.3**.

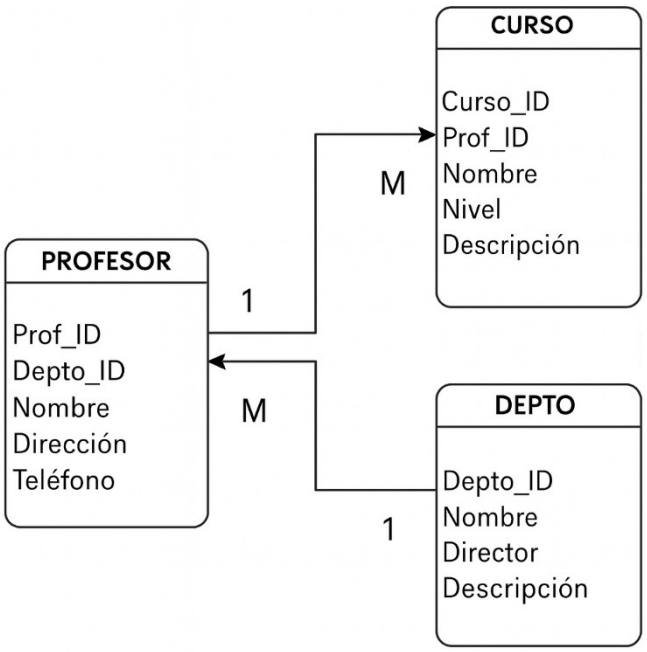


Figura 1.3. Ejemplo de estructura de base de datos relacional.
Fuente: Tomado de Diseño e implementación de un lexicón computacional para lexicografía y traducción automática, por (Moreno Ortiz, 2000).

d. **Bases de datos orientadas a objetos:** Integran principios del paradigma orientado a objetos con el almacenamiento persistente de datos. Modelan la información como objetos que encapsulan atributos y métodos, favoreciendo la representación de estructuras más complejas. Se utilizan en entornos de ingeniería, diseño asistido por computadora (CAD), investigación científica y desarrollo de sistemas de alta complejidad. Ejemplos de este tipo de base de datos son ObjectDB y db4o.

e. **Bases de datos NoSQL:** Surgen como alternativa a las bases relacionales tradicionales, diseñadas para gestionar grandes volúmenes de datos distribuidos, no estructurados o semiestructurados. Se clasifican en distintas categorías:

- **Documentales:** Almacenan documentos en formatos como JSON o XML.
  Ejemplo: MongoDB.
- **Clave-valor:** Organizan datos mediante pares sencillos de clave y valor.
  Ejemplo: Redis.

- **Columnares:** Optimizadas para el almacenamiento masivo de datos en columnas.
  Ejemplo: Cassandra.
- **De grafos:** Especializadas en modelar relaciones complejas entre entidades.
  Ejemplo: Neo4j.

De acuerdo con el criterio de Stonebraker & Kemnitz (1991), estas bases de datos han ganado amplia popularidad en aplicaciones web, redes sociales y soluciones móviles, donde la escalabilidad horizontal y la flexibilidad son prioritarias

Clasificación según la ubicación física de los datos, constituye otro criterio relevante para clasificar las bases de datos. Dependiendo de cómo estén distribuidos y administrados los datos, podemos distinguir dos grandes tipos: bases de datos centralizadas y bases de datos distribuidas.

a. **Bases de datos centralizadas:** En las bases de datos centralizadas, toda la información se almacena en una única ubicación física, bajo la administración y el control de un único punto central (ver **Figura 1.4**).

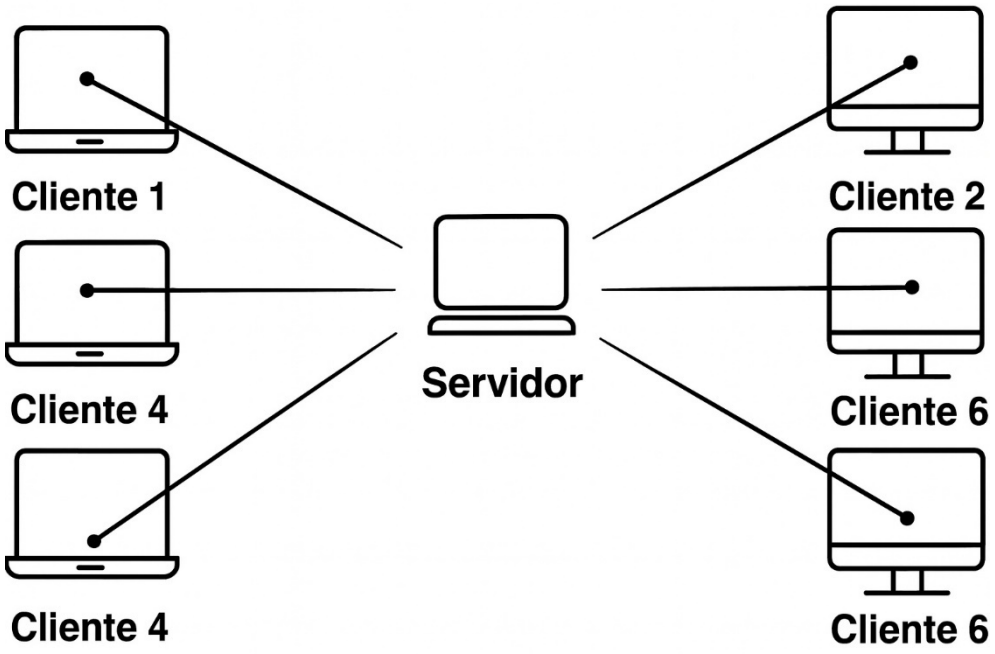


Figura 1.4. Ejemplo de estructura de base de datos centralizada.

Fuente: Elaboración propia.

Este enfoque tradicional ofrece algunas ventajas importantes en términos de simplicidad de gestión.

Ventajas:

- Facilidad de mantenimiento, ya que todas las actualizaciones, respaldos y operaciones administrativas se realizan en un solo lugar.
- Mayor control sobre la seguridad y la consistencia de los datos.

Desventajas:

- Vulnerabilidad ante fallos del sistema: si la infraestructura central sufre una interrupción, el acceso a la base de datos queda comprometido.
- Limitaciones de rendimiento y escalabilidad al atender a múltiples usuarios concurrentes, especialmente en entornos de alta demanda.

b. Bases de datos distribuidas: En contraste, las bases de datos distribuidas almacenan los datos en múltiples ubicaciones físicas que pueden ser centros de datos en distintas regiones geográficas, pero se presentan a los usuarios como si fueran una única base de datos lógica coherente, la cual se representa gráficamente en la **Figura 1.5**.

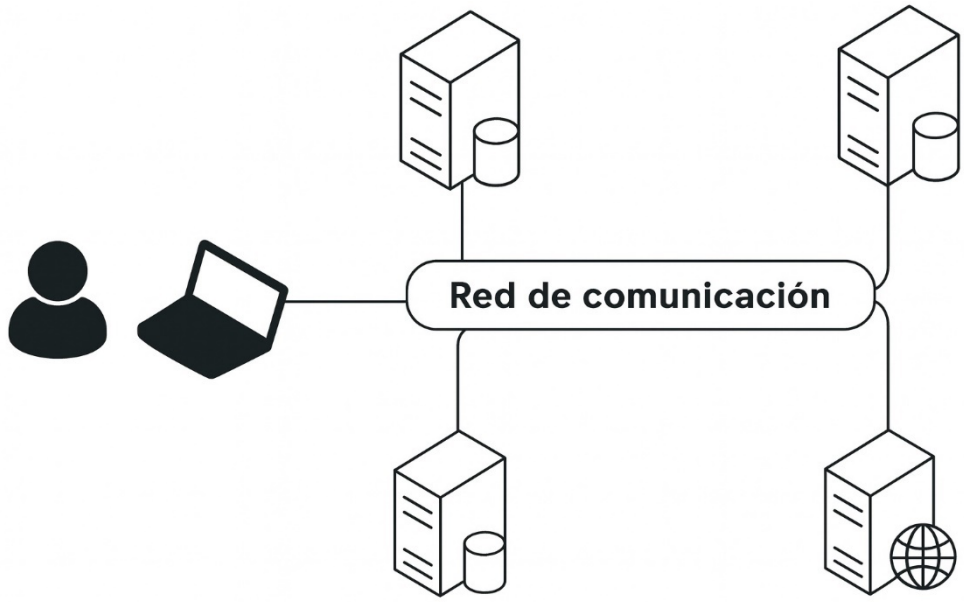


Figura 1.5. Ejemplo de estructura de base de datos distribuida.

Fuente: Elaboración propia.

A continuación, se describen las ventajas y desventajas de este tipo de base de datos:

Ventajas:

- Mayor disponibilidad, ya que la caída de un nodo no implica la inaccesibilidad total de los datos.

- Redundancia que permite la recuperación ante fallos de manera más ágil.
- Escalabilidad superior, facilitando el crecimiento horizontal conforme aumentan los volúmenes de datos o la cantidad de usuarios.

Desventajas:

- Mayor complejidad en la sincronización de datos entre nodos.
- Desafíos en la gestión de la consistencia y en el mantenimiento operativo del sistema distribuido.

Sistemas como Google Spanner y Apache Cassandra son paradigmas destacados de bases de datos distribuidas, diseñadas para soportar entornos de alta disponibilidad y escalabilidad global.

Clasificación según el tipo de contenido:

a. **Bases de datos textuales:** Estas bases de datos están especializadas en el almacenamiento y la gestión de información en formato de texto. Incluyen documentos, artículos científicos, libros digitales, reportes académicos, entre otros tipos de material textual.

   Aplicaciones típicas:

   - Bibliotecas digitales.
   - Sistemas de gestión documental.
   - Plataformas de archivo de publicaciones académicas.

b. **Bases de datos multimedia:** Gestionan contenido en formatos ricos y variados, tales como imágenes, archivos de audio, videos o animaciones. Debido a la naturaleza pesada de estos archivos, estas bases de datos requieren estructuras especializadas que permitan no solo almacenar los objetos multimedia, sino también asociarles metadatos que faciliten su búsqueda y recuperación eficiente.

   Aplicaciones típicas:

   - Plataformas de streaming de música y video.
   - Sistemas de gestión de contenidos digitales (CMS).
   - Archivos digitales de patrimonio cultural.

c. **Bases de datos espaciales y geográficas:** Diseñadas específicamente para gestionar datos que incorporan dimensiones espaciales, como coordenadas geográficas, mapas, polígonos y rutas. Estas bases de datos son esenciales para el desarrollo de Sistemas de Información

Geográfica (SIG), la planificación urbana, la navegación asistida y aplicaciones de geolocalización.

Ejemplos:

- PostGIS, extensión espacial para PostgreSQL.
- Oracle Spatial, módulo especializado de Oracle Database.

Aplicaciones típicas:

- Sistemas de cartografía digital.
- Aplicaciones de transporte y logística.
- Sistemas de monitoreo ambiental y planificación territorial.

d. Bases de datos temporales: Permiten almacenar no solo el valor actual de un dato, sino también su evolución a lo largo del tiempo. De esta manera, es posible registrar cambios históricos, consultas retrospectivas y auditorías de los datos.

Aplicaciones típicas:

- Sistemas financieros que deben mantener el historial de saldos y transacciones.
- Historias clínicas electrónicas en el sector salud.
- Registros de auditoría y cumplimiento normativo en sistemas corporativos.

e. Bases de datos operacionales (OLTP): Las bases de datos operacionales, también conocidas como OLTP (Online Transaction Processing), están orientadas a gestionar transacciones diarias en tiempo real. Están diseñadas para realizar operaciones rápidas de inserción, actualización, eliminación y consulta de datos, optimizando tanto la velocidad como la consistencia de las transacciones.

Aplicaciones típicas:

- Sistemas de ventas y puntos de venta.
- Gestión de inventarios.
- Banca en línea y servicios financieros.
- Plataformas de comercio electrónico.

Su principal fortaleza radica en su capacidad para soportar grandes volúmenes de operaciones concurrentes, garantizando la integridad y disponibilidad inmediata de la información.

f. **Bases de datos analíticas (OLAP):** Las bases de datos analíticas, conocidas como OLAP (Online Analytical Processing), están orientadas al análisis de grandes volúmenes de datos históricos. Su objetivo es facilitar la generación de informes complejos, dashboards interactivos y herramientas avanzadas de inteligencia de negocios (Business Intelligence).

   Aplicaciones típicas:

   - Almacenes de datos (Data Warehouses), que consolidan información de múltiples fuentes para análisis estratégico.
   - Cubos de datos multidimensionales, que permiten analizar métricas desde distintas perspectivas (por ejemplo, ventas por región y por trimestre).

   A diferencia de las bases OLTP, las bases OLAP priorizan la capacidad de realizar consultas analíticas complejas sobre grandes conjuntos de datos, optimizando el rendimiento en procesos de exploración y minería de datos.

1.7. Ejemplos de aplicación de las bases de datos

Las bases de datos se encuentran hoy en día integradas en prácticamente todos los ámbitos de la vida moderna. Su capacidad para organizar, almacenar y recuperar grandes volúmenes de información de manera eficiente las convierte en una herramienta indispensable para instituciones públicas y privadas, empresas, organismos gubernamentales y usuarios particulares.

Como señalan Coronel & Morris (2023), "las bases de datos son el núcleo de la mayoría de los sistemas de información contemporáneos, ya que sustentan procesos críticos de operación, análisis y toma de decisiones". Esta afirmación destaca el papel transversal que desempeñan las bases de datos en sectores estratégicos como la salud, la educación, el comercio, la administración pública, las finanzas y la tecnología.

A continuación, se describen algunos de los entornos más representativos en los que se aplican sistemas de bases de datos:

a. **Sector financiero: Banco Original y Oracle Exadata**

   Contexto y problema: Banco Original, banco digital ubicado en Brasil, enfrentaba desafíos centrados en el procesamiento de transacciones bancarias en tiempo real, escalabilidad del sistema y experiencia del cliente. Según (Oracle, 2021), era necesaria una plataforma robusta que garantizara alta disponibilidad, rendimiento transaccional y servicio

continuo sin interrupciones: características críticas para un banco 100 % digital.

Solución implementada: La entidad adoptó una solución con *Oracle Exadata X8M*, incluyendo múltiples racks (cuatro en producción y uno para pruebas), además de integrar sistemas como Oracle GoldenGate para migración de datos y Oracle Zero Data Loss Recovery Appliance (ZDLRA) para respaldos seguros. Esta arquitectura fue deseñada para garantizar procesamiento eficiente, alta disponibilidad (disponibilidad superior al 99.98 %) y recuperación instantánea de datos.

Resultados:

- El rendimiento transaccional mejoró en un promedio del 70 %**,** alcanzando hasta un 90 % en eficiencia en ciertos casos, debido a la reducción del 90 % en latencia de I/O gracias al uso de memoria persistente y tecnología RDMA sobre Ethernet convergente (RDMA-RoCE).
- La conciliación bancaria nocturna se completó 4 horas más rápido, procesándose 46.000 archivos por hora, a diferencia de los 6.000 anteriores.
- El respaldo completo se redujo de 48 horas a solo 7 horas, mientras que los incrementales diarios se realizan en 15 minutos.
- El nivel de disponibilidad durante la migración fue superior a 99.98 %**.**

b. Sector salud: Sistema de distribución de establecimientos de salud y profesionales médicos

Contexto y problema: En el estado de São Paulo, Brasil, se desarrolló una herramienta para analizar la distribución de establecimientos de salud y profesionales médicos mediante la creación de una base de datos relacional actualizable y escalable. Este sistema fue implementado para abordar la fragmentación y falta de acceso oportuno a datos sanitarios en múltiples municipios.

Solución implementada: Se construyó un pipeline ETL sobre una base de datos PostgreSQL que consolidaba datos del Registro Nacional de Establecimientos de Salud (CNES). El sistema generaba informes dinámicos con gráficos, paneles interactivos y métricas actualizadas mensualmente.

Resultados:

- Se logró automatizar la actualización mensual de datos clínicos.

- Se habilitaron dashboards dinámicos para consultar indicadores sobre la distribución de profesionales y centros de atención.
- El sistema demostró escalabilidad y apoyo a la toma de decisiones públicas para identificar zonas con déficit de médicos (M. A. Marques et al., 2025).

c. **Sector educativo: Sistema académico - Student Information Management System (SIMS)**

Contexto y problema: Una institución educativa enfrentaba dificultades con registros manuales o fragmentados de estudiantes, prolongando procesos administrativos y generando errores en datos académicos.

Solución implementada: Se diseñó e implementó un Sistema de Información Escolar utilizando Microsoft SQL Server 2008 R2 junto con Microsoft Visual Studio 2010 (.NET Framework)**.** El sistema estructuró una base de datos relacional que centralizó información crítica, tales como datos estudiantiles, matriculación, cursos y transacciones académicas; esto en un entorno digital, eficiente y consistente.

Resultados: La solución alcanzó una actualización eficaz y flexible de datos por parte del personal administrativo. La interfaz gráfica facilitó el uso, mientras que la lógica relacional aseguró integridad y consistencia de la información (Hassan, 2018).

d. Sector hotelero: Sistema de Gestión Hotelera con MySQL

Contexto y problema: En un hotel educativo gestionado por estudiantes y docentes de turismo (The Wing Ed Hotel, Politécnico Estatal de Bali, Indonesia), se requería digitalizar y optimizar los procesos administrativos usando un sistema de información. El objetivo central era pasar de registros manuales a una gestión automatizada de reservas, clientes y operaciones hoteleras.

Solución implementada: Desarrollaron un sistema de gestión hotelera (Hotel Management System) basado en una base de datos relacional usando MySQL, que permitió gestionar reservas, control de habitaciones, clientes y facturación desde un entorno digital, mejorando la eficiencia operativa.

Resultados:

- Eliminación de registros manuales redundantes, con reducción de errores y aumento en la fiabilidad de datos.
- Optimización en tiempos de atención y procesamiento de reservas.

- Facilidades en la generación de reportes administrativos y operacionales (Putra et al., 2019).

1.8. Roles en el entorno de la base de datos

El desarrollo, mantenimiento y operación de un sistema de base de datos requiere la participación coordinada de distintos actores, cada uno con funciones específicas. Estos roles permiten garantizar que el sistema funcione de manera eficiente, segura y adaptada a las necesidades de la organización.

Como señalan Coronel & Morris (2023), "el entorno de una base de datos debe ser visto como un ecosistema colaborativo en el que intervienen perfiles técnicos, administrativos y operativos, cada uno con responsabilidades que impactan en la calidad de la información y el rendimiento del sistema".

A continuación, se describen los principales roles que intervienen en el entorno de una base de datos:

a. **Administrador de bases de datos (DBA):** El Administrador de Bases de Datos (DBA) es el responsable principal de la gestión técnica y operativa del sistema. Su función es crítica para garantizar que la base de datos opere de manera segura, eficiente y confiable. De acuerdo con (Palacios Postigio, 2022), el DBA debe definir los esquemas de almacenamiento, establecer políticas de acceso y seguridad, monitorear el rendimiento, y diseñar mecanismos robustos de respaldo y recuperación ante fallos.

   Funciones clave del DBA:

   - Instalar, configurar y actualizar el software del DBMS.
   - Definir estrategias de respaldo y recuperación de datos.
   - Monitorear el rendimiento del sistema y optimizar consultas.
   - Controlar los accesos mediante la asignación de permisos y perfiles de usuario.
   - Establecer normas de seguridad, auditoría y cumplimiento normativo.
   - Supervisar el espacio de almacenamiento y los recursos disponibles.

   Ejemplo aplicado: Un DBA puede definir políticas de acceso que limiten quién puede consultar o modificar información sensible, como registros académicos o historiales médicos.

b. **Diseñador de bases de datos:** El Diseñador de Bases de Datos es el profesional encargado de estructurar los datos de manera lógica y

conceptual, utilizando modelos que representen con precisión la realidad organizacional.

Según Manzano & Avalos (2023), el diseñador traduce los requerimientos de información en esquemas normalizados que aseguran la integridad, eficiencia y expansión futura del sistema.

Tareas principales del diseñador:

- Recopilar y analizar requerimientos de información.
- Elaborar modelos conceptuales (Entidad-Relación) o UML.
- Normalizar las relaciones y definir claves primarias y foráneas.
- Planificar la escalabilidad y crecimiento del sistema.
- Validar el diseño junto a los usuarios antes de su implementación.

Ejemplo aplicado: En una universidad, el diseñador puede modelar entidades como "estudiante", "docente", "materia" y "matrícula", definiendo las relaciones que reflejen la realidad académica.

c. **Desarrollador de aplicaciones:** El Desarrollador de Aplicaciones construye las interfaces y sistemas de software que interactúan directamente con la base de datos, permitiendo a los usuarios consultar, ingresar y procesar información de forma amigable y segura.

De acuerdo con Date (2003), los desarrolladores actúan como puente entre la lógica de negocio y la gestión de datos, diseñando soluciones funcionales que traduce operaciones humanas en transacciones de bases de datos.

Responsabilidades principales:

- Programar consultas SQL, procedimientos almacenados y funciones.
- Desarrollar interfaces gráficas de usuario (formularios, paneles de control).
- Integrar la base de datos con aplicaciones web, móviles o de escritorio.
- Validar datos desde las interfaces para asegurar la integridad de la información.
- Garantizar la usabilidad y la experiencia de usuario.

Ejemplo aplicado: Un desarrollador puede diseñar un formulario web donde los docentes registren las calificaciones de los estudiantes, alimentando directamente la base de datos académica.

d. Usuarios finales: Los Usuarios Finales son los destinatarios principales del sistema, quienes utilizan las interfaces para consultar, ingresar o procesar información en el curso de sus actividades cotidianas. No requieren conocimientos técnicos profundos sobre el funcionamiento interno de la base de datos.

Clasificación de usuarios finales:

- **Usuarios ocasionales:** Consultan información esporádicamente.
- **Usuarios frecuentes:** Ingresan y actualizan datos de forma diaria.
- **Usuarios ejecutivos o de análisis:** Utilizan reportes y paneles para la toma de decisiones.

Características generales:

- Se enfocan en las tareas funcionales más que en la arquitectura del sistema.
- Dependen de la disponibilidad, precisión y facilidad de uso del sistema.
- Proporcionan retroalimentación valiosa para optimizar el diseño y la experiencia del usuario.

Ejemplo aplicado: Un estudiante universitario accede a su historial académico a través de una plataforma web, realizando consultas en la base de datos institucional mediante una interfaz amigable.

e. Otros roles complementarios: En entornos organizacionales más complejos, otros perfiles especializados pueden complementar el ecosistema de gestión de bases de datos:

- **Analistas de datos:** Especializados en extraer conocimiento mediante técnicas de estadística, visualización e inteligencia de negocios.
- **Auditores de seguridad:** Encargados de verificar el cumplimiento de políticas de protección de datos y normativas regulatorias.
- **Ingenieros de datos:** Responsables del diseño de flujos de integración de datos (ETL) y arquitecturas de procesamiento a gran escala.

En la **Tabla 1.5,** se resumen los roles en el entorno de la base de datos:

Tabla 1.5. Funciones de los principales roles en el entorno de base de datos.

Rol	**Función principal**
Administrador de bases de datos (DBA)	Gestionar técnica y operativamente el sistema de base de datos
Diseñador de bases de datos	Modelar y estructurar los datos para su implementación
Desarrollador de aplicaciones	Programar aplicaciones que interactúan con la base de datos
Usuario final	Utilizar la base de datos a través de interfaces amigables
Analista / Auditor / Ingeniero de datos	Realizar tareas especializadas de análisis, control o integración

Nota. Adaptado de (Coronel & Morris, 2023) y (Elmasri & Shamkant, 2007).

Reflexión: El éxito de un sistema de base de datos no depende exclusivamente de la tecnología implementada, sino de la colaboración efectiva entre los distintos roles humanos que lo componen. Diseñadores, desarrolladores, administradores y usuarios finales deben comprender sus responsabilidades y mantener una comunicación fluida para lograr un sistema robusto, eficiente y sostenible en el tiempo.

1.9. Ciclo de vida de una base de datos

El ciclo de vida de una base de datos comprende el conjunto de etapas que atraviesa un sistema de bases de datos desde su concepción inicial hasta su eventual desactivación o reemplazo. Comprender este proceso resulta fundamental para planificar, diseñar, implementar y mantener bases de datos de manera sistemática, minimizando errores, optimizando recursos y garantizando la calidad de la información gestionada.

Según (Piñeiro Gómez, 2024), “el desarrollo de una base de datos implica un conjunto de fases estructuradas que se inician con la recopilación y análisis de requerimientos, y concluyen con su mantenimiento continuo, contemplando además posibles procesos de evolución o migración según las necesidades organizacionales”. Este enfoque organizado permite alinear el diseño técnico con los objetivos del sistema de información y las necesidades reales de los usuarios.

A continuación, se describen las etapas más representativas del ciclo, que pueden variar ligeramente según el modelo metodológico adoptado (cascada, incremental, ágil, entre otros):

a. **Análisis de requerimientos:** Esta fase inicial consiste en identificar y documentar las necesidades de información que debe satisfacer la base de datos. Incluye tanto requisitos funcionales (qué datos se requieren, qué operaciones deben realizarse) como no funcionales (niveles de seguridad, rendimiento, escalabilidad, entre otros).

   Ejemplo aplicado: En una universidad, el análisis de requerimientos puede contemplar la necesidad de gestionar información relativa a estudiantes, docentes, asignaturas y matrículas.

   La participación activa de usuarios finales, analistas de sistemas y expertos en el dominio resulta crucial para capturar una visión precisa del problema a resolver. La calidad de esta fase condiciona directamente el éxito del sistema final.

b. **Diseño conceptual:** El diseño conceptual define la estructura lógica de los datos de forma independiente de cualquier sistema de gestión de bases de datos específico. Se utilizan modelos conceptuales como el Modelo Entidad-Relación (MER) o UML para representar entidades, atributos, relaciones y restricciones de manera abstracta y comprensible.

   Según Cherencio (2018), el objetivo del diseño conceptual es "construir un modelo semántico que represente de manera precisa y entendible la realidad del negocio, sin involucrarse aún en consideraciones técnicas de almacenamiento físico".

c. **Diseño lógico:** En esta fase, el modelo conceptual se traduce a un modelo de datos compatible con el DBMS elegido, generalmente el modelo relacional. Se especifican las tablas, los tipos de datos de cada atributo, las claves primarias y foráneas, las reglas de integridad y los índices necesarios para optimizar el acceso a los datos.

   Ejemplo aplicado: La entidad conceptual "estudiante" se convierte en una tabla con columnas como código, nombre, correo electrónico y fecha de Ingreso.

   Además, durante el diseño lógico se aplican técnicas de normalización para eliminar redundancias y asegurar la consistencia de la base de datos, considerando formas normales como:

   - Primera Forma Normal (1FN).
   - Segunda Forma Normal (2FN).
   - Tercera Forma Normal (3FN).
   - Forma Normal de Boyce-Codd (BCNF).

d. Diseño físico: El diseño físico se encarga de planificar cómo se almacenarán los datos en el hardware disponible, considerando factores de rendimiento, eficiencia y escalabilidad.

Aspectos fundamentales en esta fase incluyen:

- Organización de archivos y métodos de acceso.
- Definición de índices para acelerar búsquedas.
- Particionamiento y replicación de datos en entornos distribuidos.
- Configuración de parámetros de rendimiento en el DBMS.

Según Jiménez (2015), "el diseño físico transforma las estructuras lógicas en mecanismos concretos de almacenamiento y acceso, incidiendo directamente en el rendimiento global del sistema".

e. Implementación: La implementación consiste en la creación efectiva de la base de datos dentro del DBMS seleccionado. Se realiza mediante el uso de scripts SQL (Data Definition Language, DDL) y otras herramientas especializadas.

Tareas principales en esta fase:

- Construcción de las tablas y establecimiento de restricciones de integridad.
- Carga de datos iniciales.
- Desarrollo de vistas, procedimientos almacenados y disparadores (triggers).
- Realización de pruebas básicas de inserción, actualización, eliminación y consulta.
- Capacitación de los usuarios finales y validación del sistema frente a los requerimientos definidos.

f. Mantenimiento y evolución: Una vez desplegada, la base de datos entra en operación continua, pero requiere mantenimiento para conservar su rendimiento, integridad y adaptabilidad a los cambios organizacionales.

Las actividades de mantenimiento incluyen:

- Respaldo periódico y verificación de copias de seguridad.
- Corrección de errores o inconsistencias detectadas en la operación.
- Optimización de consultas y restructuración de índices.

- Incorporación de nuevos requerimientos funcionales o de negocio.
- Migraciones a nuevas versiones del DBMS o a infraestructuras más modernas.

De acuerdo con Ángeles (2021) destacan que "las bases de datos son sistemas vivos que evolucionan junto con la organización, por lo que su mantenimiento debe ser considerado una actividad continua y estratégica".

El ciclo de vida como proceso iterativo

Aunque la representación tradicional del ciclo de vida sugiere una progresión lineal de etapas, en la práctica el proceso es frecuentemente iterativo. Cambios en los requisitos, corrección de errores o nuevas oportunidades de mejora pueden requerir retornar a fases anteriores. Por esta razón, muchas organizaciones adoptan metodologías ágiles, basadas en ciclos cortos e incrementales, que permiten adaptar el diseño y la implementación de manera flexible y continua.

Comprender y aplicar correctamente las fases del ciclo de vida de una base de datos es fundamental para garantizar un diseño sólido, una implementación exitosa y una operación sostenible a largo plazo. Cada etapa aporta valor estratégico, contribuyendo a transformar los datos en un activo organizacional crítico que respalde la toma de decisiones, la innovación y la eficiencia operativa.

Resumen de la unidad

Esta unidad ha introducido al estudiante en los pilares conceptuales que sustentan el diseño y gestión de bases de datos modernas, partiendo de la distinción clave entre dato e información, eje central en todo sistema informático. Se analizó cómo los datos, en su forma más básica, se transforman en información valiosa cuando se organizan, interpretan y contextualizan, permitiendo así fundamentar decisiones en entornos personales, empresariales y gubernamentales.

Se abordó la relevancia estratégica de los datos en la sociedad contemporánea, destacando su papel como recurso esencial en la era digital y en el desarrollo de tecnologías emergentes como el Big Data, la inteligencia artificial y el aprendizaje automático. Desde esta perspectiva, se enfatizó la necesidad de una alfabetización crítica en el uso, protección y análisis de datos.

Posteriormente, se exploró con detalle el concepto de base de datos, entendida como un sistema estructurado para el almacenamiento y acceso eficiente a grandes volúmenes de información. Se profundizó en sus características esenciales como la persistencia, integridad, seguridad y compartibilidad y se analizaron los principales componentes de un sistema de base de datos**,** incluyendo el hardware, software, metadatos, personas y procedimientos que lo integran.

Asimismo, se estudió el rol del DBMS, pieza clave en la definición, manipulación, protección y recuperación de la información, destacando sus funciones y ventajas, así como las distintas clasificaciones existentes (relacionales, NoSQL, distribuidos, en la nube, etc.). Se examinaron también las diversas clasificaciones de las bases de datos según modelo, contenido y propósito, permitiendo al estudiante comprender su aplicabilidad en contextos reales y especializados.

La unidad culminó con una revisión de los principales entornos de aplicación de las bases de datos**,** incluyendo la banca, la salud, la educación, el comercio electrónico, las redes sociales, el gobierno y la investigación científica, lo cual evidenció su papel transversal en la sociedad actual. Finalmente, se explicó el ciclo de vida de una base de datos**,** enfatizando la necesidad de un enfoque sistemático e iterativo para garantizar la calidad, sostenibilidad y alineación del sistema con los objetivos organizacionales.

En conjunto, estos contenidos sientan las bases teóricas y prácticas necesarias para comprender la lógica estructural detrás del almacenamiento de información y su gestión eficiente, preparando al estudiante para abordar con rigor las siguientes etapas del diseño conceptual, lógico y físico de bases de datos.

Glosario de términos clave

Dato: Representación simbólica sin interpretación o contexto.

Información: Dato procesado, interpretado y contextualizado que permite tomar decisiones.

Base de datos: Colección estructurada de datos relacionados, accesibles electrónicamente.

DBMS/SGBD: Software que administra la creación, acceso, manipulación y seguridad de la BD.

Metadato: Información sobre los datos (estructura, tipo, restricciones, etc.).

Normalización: Proceso para reducir redundancias y mejorar integridad en las tablas.

Ciclo de vida: Etapas por las que atraviesa una base de datos, desde el diseño hasta el uso.

DBA: Administrador que mantiene la base de datos segura, optimizada y operativa.

Consulta SQL: Instrucción para recuperar o manipular datos en una base de datos.

Integridad referencial: Regla que asegura la coherencia de relaciones entre tablas.

Preguntas de autoaprendizaje

Instrucciones: Selecciona la opción correcta en cada caso.

1. **¿Cuál de los siguientes elementos es un ejemplo de información?**
   - **a.** "1998"
   - **b.** "Juan Pérez"
   - **c.** "Juan Pérez nació en 1998 en Loja"
   - **d.** "true"
2. **¿Cuál es una función esencial de un DBMS?**
   a. Diseñar la interfaz del sistema operativo
   b. Enviar correos electrónicos automáticamente
   c. Controlar el acceso y la integridad de los datos
   d. Formatear discos duros
3. **¿Qué tipo de base de datos se utiliza comúnmente para gestionar datos en forma de grafos?**

a. Relacional
b. Jerárquica
c. Documental
d. NoSQL de grafos

4. ¿Qué rol tiene la responsabilidad de modelar entidades y relaciones?

a. DBA
b. Diseñador de bases de datos
c. Usuario final
d. Desarrollador de hardware

5. ¿Cuál es la primera etapa del ciclo de vida de una base de datos?

a. Diseño físico
b. Implementación
c. Análisis de requerimientos
d. Mantenimiento

Respuestas: 1(c); 2(c); 3(d); 4(b); 5(c).

Ejercicios prácticos para resolver

1. Clasificación práctica

En la **Tabla 1.6,** clasifica los siguientes sistemas de información según el tipo de base de datos más adecuada:

Tabla 1.6. Funciones de los principales roles en el entorno de base de datos.

Sistema	**Tipo de base de datos (Relacional, NoSQL, Temporal, entre otros)**
Plataforma de redes sociales	
Registro académico de estudiantes universitarios	
Aplicación meteorológica con históricos diarios	
Catálogo de productos en una tienda online	

Nota. Fuente: Elaboración propia.

2. Análisis de caso

Lee la siguiente descripción y responde:

Una clínica desea implementar un sistema que permita registrar pacientes, programar citas, almacenar historiales médicos y generar reportes médicos. Los datos deben ser accesibles desde varias áreas de la clínica y contar con mecanismos de seguridad para cumplir normativas legales.

Preguntas:

- ¿Qué tipo de base de datos recomendarías? ¿Por qué?
- ¿Qué roles deberían participar en su diseño y gestión?
- ¿Qué fase del ciclo de vida es crítica para este proyecto?

3. Aplicación conceptual

En la **Tabla 1.7,** completa el siguiente modelo conceptual en formato texto. Define al menos tres entidades con atributos y relaciones simples.

Tabla 1.7. Funciones de los principales roles en el entorno de base de datos.

Entidades	**Relación esperada**
Estudiante, Curso, Matrícula	un estudiante puede matricularse en varios cursos, y cada curso puede tener varios estudiantes.

Nota. Fuente: Elaboración propia.

Referencias bibliográficas de la Unidad 1

Ángeles, M. del P. (2021). Sistema de archivos, gestores de base de datos y Hadoop: ¿evolución o retroceso? *Revista Digital Universitaria*, *22*(6). https://doi.org/10.22201/cuaieed.16076079e.2021.22.6.6

Castells, M. (2005). La era de la información: Economía, sociedad y cultura. Vol. 1: La sociedad red. In *La sociedad red* (Nueva edición). Alianza Editorial.

Cherencio, G. (2018). Implementación Asistida de Bases de Datos Relacionales en Firebird/Interbase. *Revista Tecnología y Ciencia*, *33*. https://doi.org/10.33414/rtyc.33.157-174.2018.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6), 377–387. https://doi.org/10.1145/362384.362685.

Coronel, C., & Morris, S. (2023). *Database systems : design, implementation, & management* (14th ed.). Cengage.

Date, C. J. (2003). *Introduction to Database Systems* (8th ed.). Addison-Wesley.

Elmasri, R., & Shamkant, N. (2007). *Fundamentos de Sistemas de Bases de Datos* (5th ed.). Pearson Educación.

Floridi, L., & Taddeo, M. (2016). What is data ethics? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2083), 20160360. https://doi.org/10.1098/rsta.2016.0360

Hassan, I. A. (2018). Design and Implement of a Novel Student Information Management System-Case Study International Journal of Computer Science and Mobile Computing Design and Implement a Novel Student Information Management System-Case Study. In *International Journal of Computer Science and Mobile Computing* (Vol. 7, Issue 7). https://www.researchgate.net/publication/327201996

Hoz Morales, H. A. (2018). Mayer-Schönberger, V. y Cukier, K. (2013). Big Data. La revolución de los datos masivos. *Clivajes. Revista de Ciencias Sociales*, *9*, 189. https://doi.org/10.25009/clivajes-rcs.v0i9.2536

Jiménez Capel, M. Y. (2015). *Bases de datos relacionales y modelado de datos (UF1471)*. IC Editorial. https://elibro.net/es/lc/utmachala/titulos/44139

Laudon, K., & Laudon, J. (2022). *Management information Systems Managing the Digit firmal* (17th ed.). Pearson Education. https://api.pageplace.de/preview/DT0400.9781292403571_A42098351/preview-9781292403571_A42098351.pdf

Manzano, F. A., & Avalos, D. (2023). Análisis de calidad de los datos en las estadísticas públicas y privadas, ante la implementación del Big Data. *Ciencias Administrativas*, *22*. https://doi.org/10.24215/23143738e119

Marques, M. A., Maximiano, C. F. C., Martins, T. G. dos S., & Nascimento, A. V. do. (2025). Analysis of the distribution of health services in the State of São Paulo: focus on Big Data. *Einstein (São Paulo)*, *23*. https://doi.org/10.31744/einstein_journal/2025ao1070

Moreno Ortiz, A. (2000). Diseño e implementación de un lexicón computacional para lexicografía y traducción automática. *Estudios de Lingüística Española*, *9*.

Murillo, J. (2021). *Los datos son el nuevo petróleo*. Red Forbes.

Oracle. (2021). *Banco Original speeds performance 70% with Oracle Exadata*.

Oussous, A., Benjelloun, F.-Z., Ait Lahcen, A., & Belfkih, S. (2018). Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, *30*(4), 431–448. https://doi.org/10.1016/j.jksuci.2017.06.001

Palacios Postigio. (2022). Gestión de bases de datos. *Ediciones Paraninfo, S.A., 2021*.

Piñeiro Gómez, J. (2024). *Diseño de bases de datos relacionales*. Ediciones Paraninfo.

Rowley, Jennifer. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706

Stonebraker, M., & Kemnitz, G. (1991). *THE POSTGRES NEXT-GENERATION DATABASE MANAGEMENT SYSTEM*. *34*(10).

Unidad 2: Modelo Conceptual de Datos

Introducción de la Unidad

El modelado conceptual constituye una etapa clave en el diseño lógico de bases de datos, ya que permite representar de forma abstracta, precisa y comprensible las estructuras de información que dan soporte a los procesos organizacionales. Antes de implementar cualquier sistema, resulta esencial comprender qué entidades forman parte del dominio del problema, cómo se relacionan entre sí y qué atributos las caracterizan. Esta unidad introduce los fundamentos del modelado de datos como técnica para capturar los requerimientos informativos del mundo real, facilitando su traducción posterior en estructuras lógicas y físicas eficientes.

A lo largo de esta unidad, el estudiante adquirirá competencias para construir modelos conceptuales rigurosos y expresivos, utilizando la notación de Chen como estándar gráfico para representar objetos, relaciones, atributos y restricciones de cardinalidad. Se abordarán temas como la clasificación de entidades (fuertes y débiles), el análisis de atributos y dominios, los diferentes tipos de relaciones (1:1, 1:M, M:N) y los mecanismos de extensión del Modelo Entidad-Relación (MER) tales como: Generalización, especialización y agregación. Además, se proporcionará una metodología paso a paso para la elaboración de diagramas, complementada con un caso práctico aplicado al contexto de un sistema de gestión bibliográfica.

El propósito es que el estudiante desarrolle la capacidad de representar con claridad la semántica del sistema, estableciendo una base sólida para las etapas posteriores del diseño lógico relacional. El modelado conceptual no solo mejora la calidad técnica de los sistemas de bases de datos, sino que también fortalece la comunicación entre usuarios, analistas y desarrolladores, asegurando que el sistema represente fielmente las reglas del negocio y las necesidades de la organización.

Objetivos de aprendizaje

Al finalizar esta unidad, el estudiante será capaz de:

1. Comprender el propósito y la relevancia del modelado conceptual en el diseño de bases de datos**,** valorando su papel como etapa clave para representar con precisión las necesidades informativas del mundo real antes de la implementación técnica.
2. Identificar y describir los componentes esenciales del MER, incluyendo entidades, atributos y relaciones, analizando su función estructural dentro del proceso de modelado.

3. Clasificar adecuadamente los diferentes tipos de entidades, relaciones y atributos, reconociendo sus características distintivas, niveles de abstracción y aplicaciones en diversos contextos organizacionales.
4. Aplicar correctamente las reglas de notación simbólica del MER, elaborando diagramas precisos, estandarizados y consistentes con las convenciones del diseño conceptual.
5. Detectar errores frecuentes en el modelado conceptual de bases de datos, reflexionando sobre su impacto en la posterior fase de diseño lógico y adoptando buenas prácticas que aseguren la coherencia y la escalabilidad del modelo.
6. Construir diagramas MER a partir del análisis estructurado de requerimientos de información, demostrando capacidad para abstraer, organizar y representar elementos clave del dominio de aplicación en esquemas conceptuales sólidos.

Preguntas de enfoque

- ¿Por qué es necesario modelar los datos antes de construir una base de datos?
- ¿Qué elementos conforman el MER y cómo interactúan entre sí?
- ¿Cómo se representan las cardinalidades, restricciones y demás reglas semánticas en un modelo conceptual de datos?

Desarrollo de contenidos

2.1. Introducción al modelado de datos

El modelado de datos es el proceso mediante el cual se construye una representación abstracta y organizada de la información relevante para un sistema. Esta representación conceptual actúa como una guía esencial para la posterior construcción de una base de datos, sirviendo de fundamento para asegurar la coherencia, la eficiencia y la adaptabilidad del sistema a largo plazo.

Antes de iniciar cualquier etapa de desarrollo técnico, resulta imprescindible comprender en profundidad el contexto organizacional y los requerimientos de información de los usuarios. Solo mediante una correcta interpretación de las necesidades del entorno será posible diseñar un sistema de bases de datos que cumpla eficazmente su propósito.

Como sostienen Elmasri & Shamkant (2007), “el modelado de datos ayuda a capturar los aspectos semánticos de los datos que serán almacenados, permitiendo construir una representación abstracta que luego será transformada en una estructura lógica y física”. Esto implica que el modelado conceptual no es simplemente una tarea técnica, sino también una actividad

comunicacional y analítica que requiere una estrecha colaboración entre usuarios, analistas de sistemas y diseñadores de bases de datos.

A través del proceso de modelado conceptual, los diseñadores tienen la posibilidad de:

- Identificar las entidades clave que forman parte del dominio del sistema.
- Determinar los atributos relevantes de cada objeto.
- Establecer las relaciones existentes entre las distintas colecciones.
- Definir restricciones que aseguren la integridad y validez de los datos.

El modelo conceptual actúa, de este modo, como un puente entre los requerimientos del mundo real y la estructura lógica que adoptará finalmente la base de datos dentro del sistema gestor. El uso de modelos conceptuales bien diseñados ofrece múltiples ventajas:

- Facilita la integridad de los datos, al establecer desde un inicio las reglas de consistencia.
- Optimiza la eficiencia de las consultas y operaciones posteriores.
- Asegura la escalabilidad y la adaptabilidad del sistema ante futuras modificaciones o ampliaciones.

Así, el modelado de datos no solo representa una etapa inicial del diseño, sino que constituye un pilar estratégico en la construcción de sistemas de información sólidos, confiables y orientados a las necesidades reales de las organizaciones.

2.2. MER

De acuerdo con Pons (2005), el MER, propuesto por Peter Chen en 1976, constituye una herramienta estándar ampliamente aceptada para la representación conceptual de la estructura de los datos en sistemas de información. Su principal objetivo es ofrecer una visión clara y ordenada del universo del discurso, utilizando representaciones gráficas intuitivas que incluyen entidades, atributos y relaciones.

En el MER:

- Las entidades representan objetos u objetos conceptuales del mundo real que poseen existencia propia dentro del dominio de la aplicación.
- Los atributos son propiedades o características que describen a cada objeto.
- Las relaciones establecen asociaciones lógicas entre diferentes componentes, reflejando interacciones o dependencias entre ellas.

El uso de diagramas E-R facilita la visualización y análisis estructurado de estos componentes, permitiendo a los diseñadores capturar de manera ordenada la complejidad del sistema antes de avanzar hacia su diseño lógico o físico. Según Peter P. Chen (1976), el MER proporciona un enfoque uniforme para la representación de datos, permitiendo a los diseñadores definir y comunicar claramente la estructura de los datos. Esta uniformidad y claridad explican por qué los diagramas se han convertido en una herramienta fundamental durante la fase de análisis de sistemas, siendo su correcta elaboración un factor determinante para la calidad del diseño lógico que se derive posteriormente.

El MER ha demostrado ser flexible y escalable, adaptándose eficazmente tanto a sistemas de pequeña escala como a bases de datos empresariales de gran complejidad. De acuerdo con Nevado Cabello (2010), su adopción universal en obras académicas, herramientas CASE (Computer-Aided Software Engineering) y Sistemas de Gestión de Base de Datos (DBMS) refleja su importancia estratégica en el desarrollo de sistemas de información sólidos y bien estructurados.

2.3. Importancia del análisis de entidades y relaciones

El análisis exhaustivo de objetos y relaciones es crítico para el desarrollo de modelos conceptuales sólidos, consistentes y alineados con las necesidades del sistema de información (Schmal, 2001).

Las entidades fuertes permiten capturar objetos que poseen existencia independiente, mientras que las débiles modelan dependencias lógicas o subdivisiones relevantes en el dominio de aplicación. Por otro lado, las relaciones establecen las asociaciones lógicas entre componentes, expresando las reglas de negocio que subyacen al sistema, tales como:

- **Pertenencia:** Un estudiante pertenece a una carrera académica.
- **Dependencia:** Una cuota de pago depende de un contrato de préstamo.
- **Composición:** Una factura se compone de varios ítems.
- **Especialización:** Un empleado puede ser especializado como ingeniero o administrativo.

Una relación puede involucrar dos o más entidades, y debe ser diseñada cuidadosamente para reflejar correctamente:

- Las cardinalidades (cuántos registros de un objeto se asocian con registros de otra).
- Las reglas de participación, que pueden ser obligatorias u opcionales según el contexto.

Un diseño cuidadoso de objetos y relaciones no solo optimiza la estructuración del modelo conceptual, sino que también garantiza:

- La integridad de los datos.
- La eficiencia de las consultas.
- La escalabilidad y adaptabilidad del sistema ante cambios futuros.

En consecuencia, el modelado preciso de estas estructuras se traduce en bases de datos más robustas, mantenibles y alineadas con las estrategias informacionales de las organizaciones.

2.4. Notación de Chen

La notación de Chen, propuesta por Peter Chen en 1976, constituye uno de los enfoques fundamentales para el modelado conceptual de datos. Su propósito principal es representar, de manera abstracta y estructurada, los datos y sus relaciones antes de su implementación en un DBMS. A través del MER se introduce una herramienta de abstracción que facilita la visualización de la estructura de la información y de las restricciones semánticas de los datos (Peter P. Chen, 1976), lo cual resulta útil para:

- Describir semánticamente las realidades del mundo real.
- Facilitar la comunicación entre diseñadores, usuarios y desarrolladores de sistemas.
- Servir como base conceptual para la conversión hacia esquemas relacionales u otros modelos de bases de datos (Elmasri & Shamkant, 2007).

A continuación, se presentan las principales formas gráficas utilizadas en la notación de Chen, fundamentadas teóricamente y aplicadas en la práctica del modelado de datos.

- **Entidad:** Gráficamente se representa mediante un rectángulo sencillo (ver **Figura 1.1**), con el cual se representa un conjunto de objetos del mundo real que tienen existencia independiente, tales como “empleado” o “producto”. Además, son consideradas como las unidades básicas del MER, caracterizadas por atributos distintivos que las identifican (Coronel & Morris, 2023).

Figura 2.1. Representación gráfica de Entidad.

Fuente: Elaboración propia.

- **Entidad Débil:** Gráficamente se representa mediante un rectángulo doble (ver **Figura 3.2**), utilizado para representar objetos que dependen de otra entidad fuerte para su existencia e identificación. Requieren obligatoriamente de una relación identificadora, utilizando parte de su clave primaria para poder ser identificada (Elmasri & Shamkant, 2007).

Figura 4.2. Representación gráfica de Entidad Débil.

Fuente: Elaboración propia.

- **Relación:** Gráficamente se representa mediante un rombo sencillo (ver **Figura 5.3**), empleado para modelar las asociaciones o interacciones que existen entre dos o más objetos, como en los casos de relaciones "Trabaja en" o "Vende".
  Las relaciones permiten capturar de forma semántica los vínculos entre conjuntos de componentes (Harrington, 2016).

Figura 6.3. Representación gráfica de Relación.

Fuente: Elaboración propia.

- **Relación Identificadora:** Gráficamente se representa mediante un rombo con líneas discontinuas (ver **Figura 7.4**). Este símbolo se utiliza para identificar relaciones que establecen la dependencia entre una entidad débil y su entidad fuerte correspondiente.
  El rombo punteado indica que la participación de la entidad débil en esta relación es total y necesaria para su existencia (Coronel & Morris, 2023).

Figura 8.4. Representación gráfica de Relación Identificadora.

Fuente: Elaboración propia.

- **Atributo:** Gráficamente se representa mediante una elipse sencilla (ver **Figura 9.5**).

El atributo describe una característica o propiedad de una entidad o de una relación, como pueden ser el nombre de un empleado o el monto de una factura.
Cada objeto debe poseer atributos que enriquezcan su representación semántica (Elmasri & Shamkant, 2007).

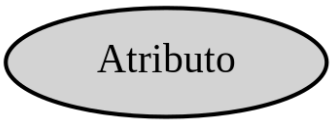


Figura 10.5. Representación gráfica de Atributo.

Fuente: Elaboración propia.

- **Atributo Clave:** Gráficamente se representa mediante una elipse sencilla subrayada (ver **Figura 11.6**). Un atributo clave permite identificar de manera única cada instancia de una entidad dentro del conjunto de datos. La unicidad garantizada por los atributos clave es fundamental para establecer las claves primarias del modelo (Date, 2003).

Figura 12.6. Representación gráfica de Atributo Clave.

Fuente: Elaboración propia.

- **Atributo Multivaluado:** Gráficamente se representa mediante una elipse doble (ver **Figura 13.7**). Este símbolo se utiliza para describir atributos que pueden contener múltiples valores para una misma colección, como por ejemplo varios números telefónicos asociados a un solo cliente. Durante el proceso de normalización, los atributos multivaluados suelen dar lugar a la creación de nuevas entidades (Coronel & Morris, 2023).

Figura 14.7. Representación gráfica de Atributo Multivaluado.

Fuente: Elaboración propia.

- **Atributo Compuesto:** Gráficamente se representa mediante una elipse conectada a subatributos (ver **Figura 15.8**). Un atributo compuesto puede descomponerse en varios subcomponentes, permitiendo un análisis granular de la información; por ejemplo, “nombre completo”

dividido en “nombre” y “apellido”. Esta representación favorece un modelado detallado de las características de las entidades (Elmasri & Shamkant, 2007).

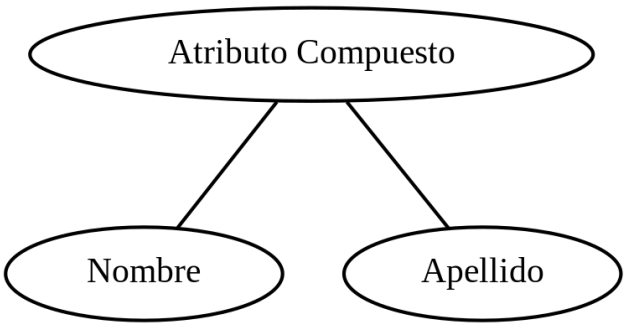


Figura 16.8. Representación gráfica de Atributo Compuesto.

Fuente: Elaboración propia.

- **Cardinalidad:** Gráficamente se representa mediante números o símbolos junto a las líneas de conexión entre las entidades y las relaciones (ver **Figura 17.9**). La cardinalidad indica el número de instancias de un objeto que pueden estar asociadas con instancias de otra colección en una relación. Una correcta definición de la cardinalidad es fundamental para reflejar adecuadamente las restricciones del dominio (Harrington, 2016).

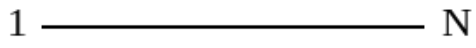


Figura 18.9. Representación gráfica de Cardinalidad.

Fuente: Elaboración propia.

Nota destacada: La notación de Chen se ha consolidado como una metodología clave en el modelado conceptual de bases de datos gracias a su claridad semántica y a su efectividad visual. Cada forma gráfica dentro de este enfoque cumple una función específica que permite traducir, de manera estructurada y comprensible, las complejidades del mundo real a representaciones formales que guían el diseño y la implementación de sistemas de bases de datos robustos y escalables. La correcta aplicación de la notación de Chen no solo facilita el diseño lógico posterior, sino que también mejora la comunicación entre los diferentes perfiles involucrados en el desarrollo de sistemas de información.

2.5. Tipos de entidades y relaciones

Ferraggine & Rivero (2005), mencionan que la correcta identificación y clasificación de las entidades constituye un paso fundamental en la construcción de modelos conceptuales de bases de datos. Desde una

perspectiva semántica, los objetos pueden clasificarse principalmente en dos tipos, en función de su grado de dependencia o independencia respecto de otras colecciones. Esta distinción incide de manera directa en la definición de claves primarias y en el establecimiento de restricciones de integridad referencial en el modelo de datos.

Entidad fuerte

Según Osorio Rivera (2008), una entidad fuerte se caracteriza por tener existencia propia dentro del universo de discurso, sin requerir de ningún otro objeto para su identificación. Su atributo distintivo es la posesión de una clave primaria que garantiza su identificación única en el conjunto de datos. Esta independencia semántica permite que las colecciones fuertes modelen objetos de interés por sí mismos, tales como estudiantes, clientes, empleados o productos. Son, por tanto, el eje central sobre el cual se estructuran muchas bases de datos.

Entidad débil

En contraste, una entidad débil carece de una clave primaria autónoma que le permita ser identificada de forma independiente. Su existencia y singularidad dependen directamente de una asociación con una entidad fuerte.

Para su identificación, una entidad débil combina su propio identificador parcial con la clave primaria de la entidad fuerte a la que está vinculada. Este tipo de colecciones resulta útil para modelar estructuras jerárquicas o dependientes, como los ítems de una factura (detallefactura) o los pagos periódicos de un crédito (pagocuota). En la **Tabla 2.1,** se ejemplifican este tipo de objetos:

Tabla 2.1. Tipos de entidades.

Tipo de Entidad	Descripción	Ejemplo
Entidad fuerte	Representa objetos o conceptos con existencia independiente dentro del dominio de aplicación. Se identifica mediante una clave primaria propia.	estudiante, cliente, producto
Entidad débil	Representa objetos cuya existencia depende funcionalmente de una entidad fuerte para su identificación. Se identifican combinando su identificador parcial con la clave de la entidad fuerte.	detallefactura, hijo, pagocuota

Nota. Adaptado de Piattini y Castaño (1999).

2.6. Atributos y su representación

Los atributos son elementos descriptivos que proporcionan información específica sobre las entidades o relaciones en un modelo conceptual de datos. La correcta identificación, clasificación y definición de los atributos es fundamental para lograr un modelo de datos eficiente, coherente y comprensible, además de sentar las bases para una adecuada normalización en fases posteriores del diseño lógico (Codd, 1970).

Los atributos pueden clasificarse en las siguientes categorías principales:

- **Simples:** Son atributos atómicos, que contienen un único valor indivisible por instancia. Ejemplo: Nombre, número de identificación.
- **Compuestos**: Son atributos que pueden subdividirse en varios componentes más pequeños, cada uno de los cuales posee significado propio. Ejemplo*:* Dirección (compuesta por calle, ciudad y país).
- **Multivaluados:** Son atributos que pueden contener más de un valor para una misma instancia de entidad o relación. Ejemplo*:* Números de teléfonos de contacto asociados a un cliente.
- **Derivados:** Son atributos cuyo valor se calcula o infiere a partir de otros atributos ya almacenados. Ejemplo*:* La edad de una persona derivada de su fecha de nacimiento.

En el diagrama del MER, los atributos se representan gráficamente de la siguiente manera:

- Mediante un óvalo sencillo para los atributos simples.
- Mediante un óvalo doble para los atributos multivaluados.
- Mediante un óvalo punteado para los atributos derivados.
- Los atributos clave primaria se identifican mediante el subrayado del nombre del atributo.

Definir atributos de manera precisa y sistemática tiene un impacto directo en:

- La calidad del modelo lógico, facilitando su normalización y eficiencia estructural.
- La calidad de los datos almacenados, evitando redundancias, ambigüedades o inconsistencias.
- La claridad semántica del modelo conceptual, mejorando la comunicación entre analistas, diseñadores y usuarios.

Definición de dominios de atributos: Un dominio establece el conjunto de valores válidos para un atributo. A continuación, ejemplos típicos:

- **emp_sueldo:** Número decimal positivo, con dos decimales, rango [0.00 … 999 999.99].
- **est_nivel:** Enumeración {“básico”, “intermedio”, “avanzado”}.
- **prod_descripción:** Texto de hasta 255 caracteres, sin saltos de línea.

Nota destacada: Un atributo mal definido; por ejemplo, aquel que combina múltiples valores en un solo campo o aquel que presenta ambigüedad en su interpretación puede comprometer seriamente la consistencia, la integridad y la funcionalidad de toda la base de datos. Por lo tanto, el análisis riguroso y la correcta categorización de los atributos constituyen una etapa crítica en el proceso de modelado conceptual de datos.

2.7. Clasificación de las Relaciones entre Entidades

En el modelo conceptual, las relaciones representan las asociaciones lógicas existentes entre diferentes entidades dentro del universo del discurso. Capel (2025), menciona que un aspecto fundamental en el análisis de relaciones es la cardinalidad, que describe cuántas instancias de un objeto pueden estar asociadas con cuántas instancias de otra.

Tipos de relación: De acuerdo con Aldana & Motta (2021), los tipos de relaciones más comunes son: Uno a uno (1:1); Uno a muchos (1:M); Muchos a muchos (M:N). A continuación, se describe cada una de ellas:

a. **Uno a uno (1:1):** En una relación uno a uno (1:1) (ver **Figura 19.10**), a cada instancia de la entidad A le corresponde, como máximo, una única instancia de la entidad B, y recíprocamente, cada instancia de B se asocia como máximo a una única instancia de A.

   Es útil cuando los elementos de ambos objetos deben identificarse y asociarse de forma única, como ocurre, por ejemplo, en sistemas gubernamentales o de control migratorio. Desde el punto de vista del diseño, esta relación permite elegir en qué colección se almacenará la clave foránea.

Figura 20.10. Representación gráfica de la relación Uno a uno (1:1).

Fuente: Elaboración propia.

Caso de estudio 1: Relación 1:1 entre persona y pasaporte: Una persona puede tener asignado un único pasaporte, y cada pasaporte corresponde exclusivamente a una sola persona (ver **Figura 21.11**).

Esta relación refleja una cardinalidad uno a uno (1:1), en la que ambas entidades mantienen una correspondencia individual y exclusiva.

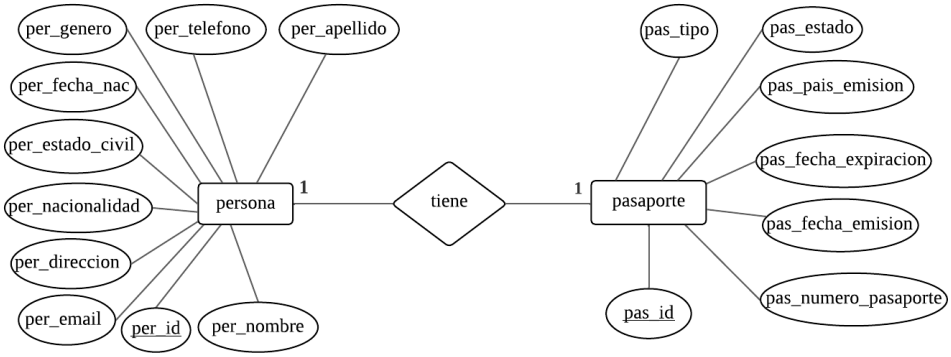


Figura 22.11. Ejemplificación de la relación Uno a uno (1:1), persona - pasaporte.

Fuente: Elaboración propia.

En este caso, lo más coherente es incluir el identificador de la persona (per_id) como clave foránea dentro del objeto pasaporte, ya que la existencia del pasaporte depende lógicamente de la persona que lo porta.

Caso de estudio 2: Relación 1:1 entre vehículo y registro vehicular: Cada vehículo posee exactamente un registro vehicular, y, de manera recíproca, cada registro vehicular está asociado de forma exclusiva a un único vehículo (ver **Figura 23.12**). Esta correspondencia individual refleja una relación de cardinalidad uno a uno (1:1), donde cada instancia de un objeto se vincula de manera única con una instancia de la otra.

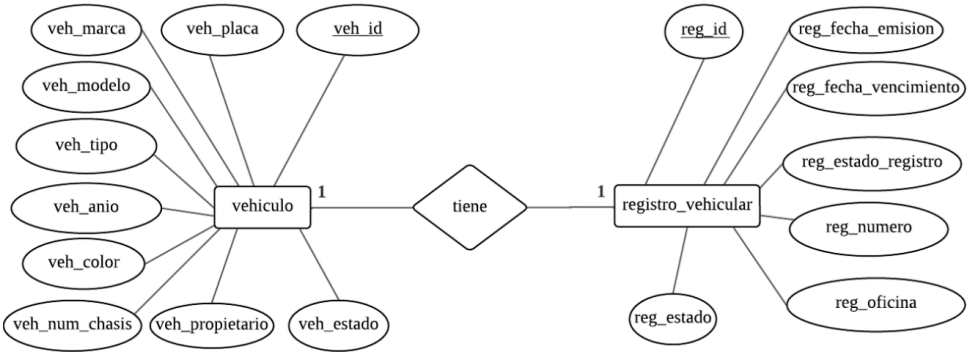


Figura 24.12. Ejemplificación de la relación Uno a uno (1:1), vehículo–registro vehicular.

Fuente: Elaboración propia.

El diseño con cardinalidad 1:1 se adopta cuando se requiere separar datos por razones lógicas, administrativas o de confidencialidad. En este caso, mantener el registro vehicular como una entidad separada

del vehículo facilita la actualización de la información legal o documental sin necesidad de alterar los atributos técnicos del vehículo. Asimismo, permite un mejor control del ciclo de vida de los registros, como renovaciones, cancelaciones o cambios de estado.

b. **Uno a muchos (1: M):** Este tipo de relación se presenta cuando una sola instancia de una entidad A puede estar asociada con una o más instancias de una entidad B, mientras que cada instancia de B solo puede estar relacionada con una única instancia de A (ver **Figura 25.13**).

Figura 26.13. Ejemplificación de la relación Uno a muchos (1:M).

Fuente: Elaboración propia.

Caso de estudio 1: Relación 1:M entre ciudad y cliente: Una ciudad puede estar asociada a múltiples clientes, mientras que cada cliente reside en una única ciudad (ver **Figura 27.14**). Esta configuración refleja una relación de cardinalidad uno a muchos (1:N), en la que una instancia de la entidad ciudad se vincula con varias instancias del objeto cliente, pero cada cliente mantiene su relación exclusivamente con una sola ciudad.

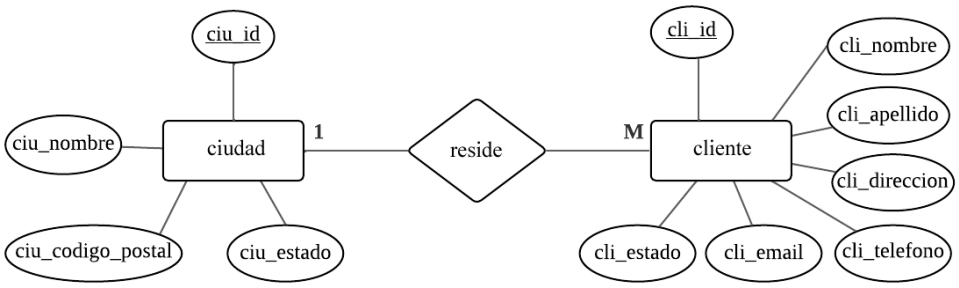


Figura 28.14. Ejemplificación de la relación Uno a muchos (1:M), ciudad – cliente.

Fuente: Elaboración propia.

Este tipo de relación es común en sistemas de información administrativos o comerciales, donde se necesita organizar los datos de los usuarios según su ubicación. Por ejemplo, una empresa de telecomunicaciones puede gestionar su base de datos de clientes agrupándolos por ciudad, lo que facilita operaciones como facturación local, análisis de mercado o asignación de recursos.

Caso de estudio 2: Relación 1:M entre categoría y producto: Una categoría puede agrupar a múltiples productos, mientras que cada producto pertenece exclusivamente a una sola categoría (ver **Figura 29.15**). Esta configuración representa una relación de cardinalidad uno a muchos (1:N), en la cual una instancia de la entidad categoría se asocia con varias instancias de la colección producto, pero cada producto está vinculado únicamente a una categoría.

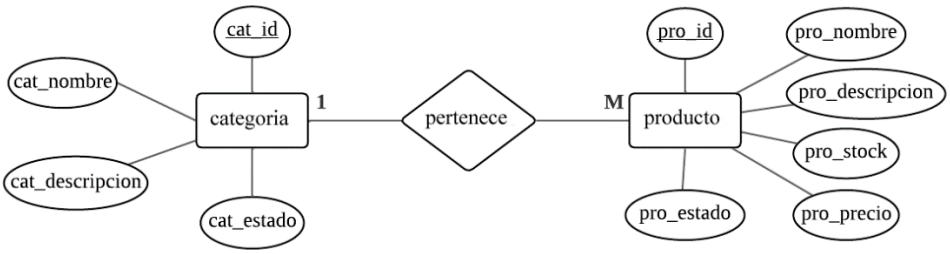


Figura 30.15. Ejemplificación de la relación Uno a muchos (1:M), categoría – producto.

Fuente: Elaboración propia.

Este tipo de estructura es ampliamente utilizada en sistemas de comercio electrónico, supermercados, farmacias, librerías y otros entornos donde los productos se organizan en categorías para facilitar la navegación, el análisis y la toma de decisiones. Por ejemplo, en una tienda en línea, una categoría como "Tecnología" puede agrupar productos como "Laptops", "Auriculares", "Cámaras", etc., mientras que cada uno de esos productos pertenece exclusivamente a esa categoría.

c. **Muchos a muchos (M:N):** Este tipo de relación se presenta cuando varias instancias de una entidad A pueden estar asociadas con múltiples instancias de una entidad B, y viceversa (ver **Figura 31.16**). Es decir, no existe una correspondencia exclusiva entre los elementos de ambos objetos, sino que cada uno puede vincularse con muchos otros. Esta configuración es común en escenarios reales donde se establecen asociaciones recíprocas y dinámicas. Por ejemplo:

- Un estudiante puede estar inscrito en varios cursos, y cada curso puede tener múltiples estudiantes inscritos.
- Un producto puede estar presente en varios pedidos, y un pedido puede incluir varios productos distintos.
- Un autor puede escribir varios libros, y un libro puede ser coescrito por varios autores.

Cuando la relación posee características o información adicional que se desea almacenar, tales como fecha, cantidad, estado, entre otras; estos datos se representan como atributos del propio rombo, es decir, la

relación adquiere atributos propios. Posteriormente, durante el diseño lógico relacional, esta relación se transformará en una tabla intermedia que contendrá las claves foráneas de ambas entidades y, en caso de existir, los atributos específicos de la relación.

Figura 32.16. Ejemplificación de la relación Muchos a Muchos (M:N).

Fuente: Elaboración propia.

Caso de estudio 1: Relación M:N entre pedido y producto: Un producto puede formar parte de múltiples pedidos, al mismo tiempo que un pedido puede contener varios productos (ver **Figura 33.17**). Esta interacción refleja una relación de cardinalidad muchos a muchos (M:N), en la que múltiples instancias de la entidad producto se asocian con múltiples instancias de la colección pedido, estableciendo una vinculación recíproca y flexible entre ambos conjuntos de datos.

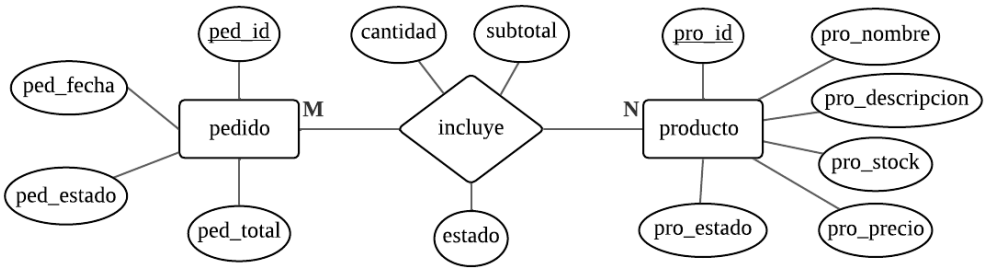


Figura 34.17. Ejemplificación de la relación Muchos a muchos (M:N), pedido – producto.
Fuente: Elaboración propia.

La relación M:N permite gestionar atributos propios en el rombo, tales como: cantidad del producto solicitado, subtotal por ítem, estado del producto en el pedido. Esto permite capturar no solo el vínculo entre entidades, sino también información transaccional clave. Este patrón es ampliamente utilizado en el diseño de sistemas de ventas o facturación. Por ejemplo: En una tienda virtual, un cliente puede generar un pedido que contenga tres productos distintos: un teclado, un monitor y un mouse. Cada uno de estos productos, a su vez, puede aparecer en otros pedidos realizados por otros clientes en fechas diferentes.

Caso de estudio 2: Relación M:N entre autor y libro: Un autor puede escribir múltiples libros, y, a su vez, un libro puede ser escrito en colaboración por varios autores (ver **Figura 35.18**). Esta situación representa una relación de cardinalidad muchos a muchos (M:N), donde diversas instancias de la entidad autor se asocian con múltiples

instancias de la colección libro, reflejando la posibilidad de contribuciones conjuntas en obras literarias, científicas o académicas.

En el MER, esta relación se representa mediante una entidad asociativa denominada escribe, que permite gestionar no solo el vínculo entre autor y libro, sino también atributos propios del proceso de colaboración, tales como: rol del autor en la obra (autor principal, coautor, editor, entre otros), orden_autor, que indica la secuencia de aparición en la publicación. Estos atributos resultan fundamentales en contextos científicos y editoriales, donde la posición y el rol del autor reflejan su nivel de contribución intelectual.

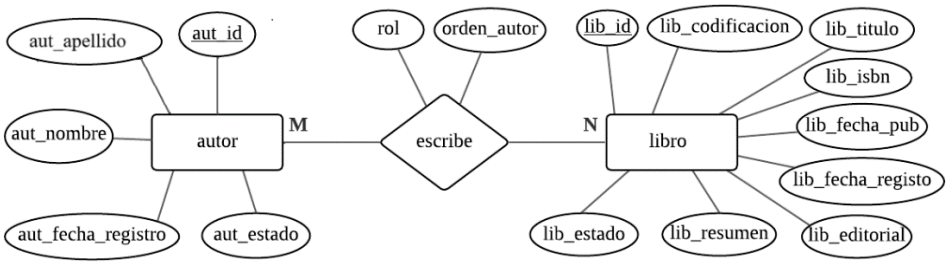


Figura 36.18. Ejemplificación de la relación Muchos a muchos (M:N), autor - libro.

Fuente: Elaboración propia.

Este modelo es ampliamente aplicable en sistemas de bibliotecas, editoriales, revistas científicas o repositorios académicos. Por ejemplo: En un libro de investigación sobre inteligencia artificial, pueden haber colaborado tres autores: la primera persona como autor principal, la segunda como coautor y la tercera como editor técnico. Cada uno tiene un rol específico y un orden de aparición reconocido en la publicación.

Participación en las relaciones

Según Merchán-Manzano (2018), además de la cardinalidad, las relaciones entre objetos también se caracterizan por el tipo de participación, que indica si la presencia de una instancia de un objeto en una relación es obligatoria o no.

- **Participación total:** Todas las instancias de la entidad deben participar en la relación. Se representa gráficamente mediante una línea doble en los diagramas E-R. Ejemplo: Cada factura debe estar asociada a un cliente.
- **Participación parcial:** Solo algunas instancias de la colección participan en la relación. Se representa mediante una línea simple. Ejemplo: No todos los clientes tienen asociadas órdenes activas en un momento dado.

Estas distinciones resultan fundamentales para implementar de manera adecuada las restricciones de integridad referencial en el diseño lógico de la base de datos.

Relevancia del análisis de cardinalidad y participación

Un análisis preciso de las cardinalidades y tipos de participación es esencial para evitar ambigüedades en la representación del dominio de aplicación. Como afirman Cornelio et al. (2004), un estudio riguroso de las cardinalidades y los tipos de participación en cada relación evita confusiones sobre cómo deben asociarse las entidades en nuestro modelo. De este modo, reflejamos con precisión las reglas de negocio y aseguramos la integridad global del diseño.

Estas decisiones impactan directamente en:

- La estructura de las tablas en el modelo relacional.
- La definición de claves primarias y foráneas.
- La aplicación de restricciones de integridad.
- La forma en que se diseñan las relaciones lógicas y físicas dentro del DBMS.

Según Silberschatz et al. (2002), comprender y aplicar correctamente los conceptos de cardinalidad y participación resulta fundamental para la creación de bases de datos robustas, coherentes y alineadas con los requerimientos reales de las organizaciones. Estos conceptos son esenciales para garantizar que la representación del modelo conceptual refleje fielmente las reglas del negocio y las restricciones inherentes al dominio de aplicación.

En el contexto de los MER, las relaciones entre entidades se clasifican principalmente según su cardinalidad, la cual determina cuántas instancias de un objeto pueden estar asociadas con cuántas instancias de otra colección (PIÑEIRO GOMEZ, 2024). Las cardinalidades más comunes incluyen relaciones uno a uno (1:1), uno a muchos (1:N) y muchos a muchos (M:N).

Además de la cardinalidad, las relaciones también se caracterizan por su tipo de participación, que indica si la intervención de una colección en una relación es obligatoria u opcional:

- **Participación total:** Todas las instancias de un objeto deben necesariamente participar en la relación. Se representa gráficamente mediante una línea doble en los diagramas del MER.
- **Participación parcial:** Solo algunas instancias de la entidad participan en la relación. Se representa mediante una línea simple.

Estas distinciones son fundamentales para implementar correctamente las restricciones de integridad referencial en el diseño lógico de la base de datos. Una definición precisa de las cardinalidades y de las participaciones asegura que el modelo evite inconsistencias, ambigüedades y posibles errores durante su implementación y operación.

Como sostienen Coronel & Morris (2023), "el análisis preciso de la cardinalidad y la participación permite evitar ambigüedades en la representación del dominio, lo cual es esencial para garantizar la integridad del modelo". Estas decisiones de diseño afectan de manera directa:

- La estructura de las tablas en el modelo relacional.
- La forma en que se aplican claves primarias y foráneas.
- La correcta implementación de restricciones de integridad.
- La eficiencia y escalabilidad de las consultas y transacciones en el sistema.

Por ello, el análisis riguroso de la cardinalidad y la participación constituye un pilar esencial para el éxito del diseño conceptual y, en última instancia, para la calidad y sostenibilidad del sistema de información construido.

2.8. Extensiones del MER

Para representar conceptos más avanzados y captar estructuras complejas del mundo real, el MER puede ser extendido mediante el uso de mecanismos conceptuales adicionales. Según Saiedian (1997), estas extensiones permiten modelar con mayor precisión situaciones donde existen jerarquías, herencias o relaciones contextuales entre entidades.

Los mecanismos principales de extensión del MER son los siguientes:

a. **Generalización:** La generalización es el proceso mediante el cual se agrupan varias entidades específicas que comparten atributos o comportamientos comunes en una colección más general.

   Por ejemplo, los objetos profesor y administrador pueden generalizarse en una entidad superior llamada empleado, la cual contiene los atributos comunes, mientras que las colecciones específicas mantienen sus características particulares (ver **Figura 37.19**).

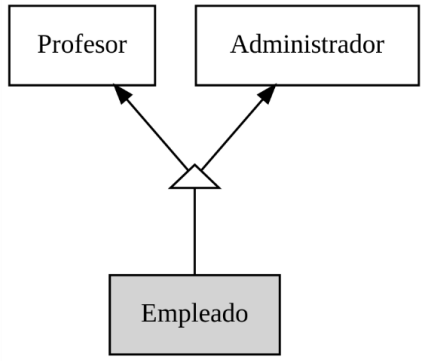


Figura 38.19. Ejemplificación de la extensión Generalización.

Fuente: Elaboración propia.

Este mecanismo facilita la abstracción y evita la redundancia en el modelo conceptual, promoviendo un diseño más limpio y coherente.

b. Especialización: La especialización es el proceso inverso a la generalización: a partir de una entidad general, se derivan subentidades que poseen características particulares adicionales.

Por ejemplo, la el objeto empleado puede especializarse en las subentidades profesor y técnico, cada una con atributos y roles específicos que no necesariamente comparten (ver **Figura 39.20**).

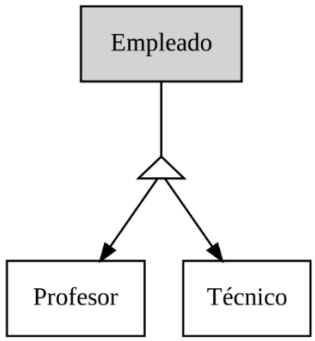


Figura 40.20. Ejemplificación de la extensión Especialización.

Fuente: Elaboración propia.

La especialización permite reflejar diferencias específicas dentro de un conjunto generalizado, asegurando la correcta representación de variantes funcionales o estructurales dentro del dominio modelado.

c. Agregación: La agregación permite que una relación entre colecciones sea tratada conceptualmente como una entidad propia en el contexto de una nueva relación más compleja.

Por ejemplo, la relación asigna entre profesor y curso puede ser agregada como una entidad para modelar posteriormente otra

relación denominada evaluar, en la cual un Coordinador Académico evalúa las asignaciones (ver **Figura 41.21**).

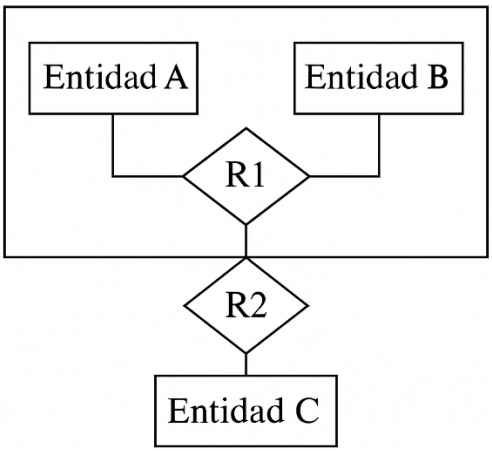


Figura 42.21. Ejemplificación de la extensión Agregación.
Fuente: Elaboración propia.

La agregación resulta especialmente útil en estructuras jerárquicas o en sistemas que requieren modelar relaciones entre relaciones (metarelaciones), como ocurre en contextos administrativos, hospitalarios o de gestión de recursos humanos.

Relevancia de las Extensiones

Estas extensiones son de particular importancia en el diseño de modelos conceptuales complejos, como los utilizados en:

- Sistemas académicos, donde existen jerarquías de roles (estudiantes, profesores, administrativos).
- Entornos hospitalarios, que requieren distinguir entre distintos tipos de personal médico o relaciones complejas entre pacientes y tratamientos.
- Sistemas de recursos humanos, donde empleados, contratistas y consultores deben ser modelados con variantes específicas pero coherentes.

El uso adecuado de mecanismos como la generalización, especialización y agregación contribuye a construir modelos conceptuales más expresivos, claros y fieles a la realidad organizacional.

2.9. Metodología para construir un MER

De acuerdo con criterio de Marques (2011), el desarrollo de un modelo conceptual de datos debe seguir una metodología estructurada que garantice la integridad, la coherencia y la pertinencia del modelo respecto al dominio de aplicación Una secuencia metodológica bien definida permite capturar de manera precisa los requerimientos del sistema, minimizar errores

conceptuales y asegurar que el modelo refleje fielmente las necesidades del negocio o la organización.

A continuación, se describe una secuencia típica de actividades para la construcción de un modelo conceptual:

a. **Recolección de requerimientos:** Esta etapa inicial consiste en identificar las necesidades de información mediante técnicas como:
    - Entrevistas con usuarios clave y expertos del dominio.
    - Análisis documental de procesos, manuales y reglamentos existentes.
    - Observación directa de las operaciones o flujos de trabajo.

    La calidad de esta fase es determinante, ya que proporciona la base sobre la cual se construirá el modelo conceptual.

b. **Identificación de entidades:** A partir del análisis de los requerimientos, se procede a identificar las colecciones relevantes. Generalmente, las entidades se derivan de los sustantivos clave presentes en la descripción del dominio, como cliente, producto, factura, entre otros. Una correcta identificación de objetos es crucial para evitar omisiones o redundancias en el modelo.

c. **Definición de atributos:** En esta fase, se definen los atributos que describen las características de cada entidad o relación. Es importante:
    - Precisar el tipo de dato (numérico, alfanumérico, fecha, etc.).
    - Establecer las unidades de medida, si corresponde.
    - Identificar si los atributos son simples, compuestos, multivaluados o derivados.

    Una definición adecuada de atributos mejora la claridad semántica y facilita la posterior normalización del modelo lógico.

d. **Establecimiento de relaciones:** Se identifican y modelan las relaciones entre las entidades, utilizando generalmente verbos que describen las acciones o asociaciones del mundo real, tales como “compra”, “asigna”, “pertenece a”, entre otros. Cada relación debe capturar adecuadamente la interacción entre los objetos involucradas.

e. **Determinación de cardinalidades:** En esta etapa se definen las cardinalidades de las relaciones, especificando:
    - Cuántas instancias de una entidad pueden asociarse con instancias de otra.

- Las reglas de negocio que rigen dichas asociaciones (por ejemplo, un cliente puede realizar muchos pedidos, pero un pedido pertenece a un solo cliente).

Una correcta definición de las cardinalidades garantiza la integridad referencial del modelo.

f. Representación gráfica: Se elabora el diagrama MER, utilizando las convenciones gráficas adecuadas (por ejemplo, la notación de Chen).

g. Es fundamental que el diagrama sea claro, completo y de fácil interpretación para los distintos actores involucrados.

h. Validación del modelo: Finalmente, se realiza una validación del modelo conceptual con los usuarios finales, analistas y otros interesados. Esta revisión busca asegurar que el modelo:

- Representa fielmente la realidad del dominio.
- Cumple con los requerimientos de información establecidos.
- Carece de inconsistencias, omisiones o ambigüedades.

Nota destacada:

La aplicación rigurosa de esta metodología no solo promueve la claridad conceptual, sino que también minimiza errores en etapas posteriores del desarrollo de la base de datos, como el diseño lógico y físico. Un modelo conceptual bien construido constituye una herramienta estratégica que facilita la comunicación entre los distintos perfiles del proyecto y asegura que el sistema de **información** esté alineado con las verdaderas necesidades del negocio

2.10. Casos prácticos de modelado conceptual

a. **Sistema de biblioteca.**

Una biblioteca ha iniciado un ambicioso proyecto de modernización de su sistema de gestión bibliográfica, con el objetivo de reemplazar los registros tradicionales en papel por un Sistema de Información de Biblioteca completamente informatizado y adaptado a las necesidades contemporáneas.

El nuevo sistema tendrá como alcance principal las siguientes funcionalidades:

- **Registro detallado de autores y obras bibliográficas:** Captura y administración de la información esencial de autores y materiales bibliográficos, incluyendo sus metadatos y relaciones.
- **Administración de préstamos y devoluciones de libros:** Gestión eficiente de las transacciones de préstamo, devolución, y control de sanciones o renovaciones.

- **Control de usuarios y personal bibliotecario:** Registro de clientes (usuarios de la biblioteca) y de los funcionarios encargados de la operación y supervisión del sistema.
- **Categorización temática de los materiales:** Clasificación sistemática de libros, artículos y otros recursos en áreas temáticas que faciliten la búsqueda y recuperación de información.
- **Preparación para futuras funcionalidades avanzadas:** El sistema será diseñado considerando su futura integración con técnicas de Inteligencia Artificial, orientadas a análisis predictivo, generación de recomendaciones de lectura personalizadas y optimización del uso de los fondos bibliográficos.

El primer paso en el desarrollo de este sistema consiste en la elaboración del modelo conceptual, utilizando la notación de Chen como estándar de representación.

En esta fase se abordarán:

- La identificación de las entidades principales involucradas en el dominio bibliotecario.
- La definición de atributos clave para cada objeto.
- El establecimiento de relaciones entre colecciones, indicando sus cardinalidades y tipos de participación.
- La representación clara y precisa de las restricciones semánticas relevantes para el funcionamiento del sistema.

La construcción de este modelo conceptual será fundamental para:

- Garantizar que el sistema de información responda a los procesos reales de la biblioteca.
- Facilitar la posterior transición hacia el diseño lógico y físico del sistema de base de datos.
- Asegurar que la arquitectura del sistema permita su evolución futura, integrando tecnologías emergentes como el análisis inteligente de datos.

El diagrama MER base se puede apreciar en la **Figura 43.22**, y tiene como propósito representar las entidades autor, obra_bibliográfica, editorial, así como la relación escribe y publica, que las vincula de manera lógica y estructurada.

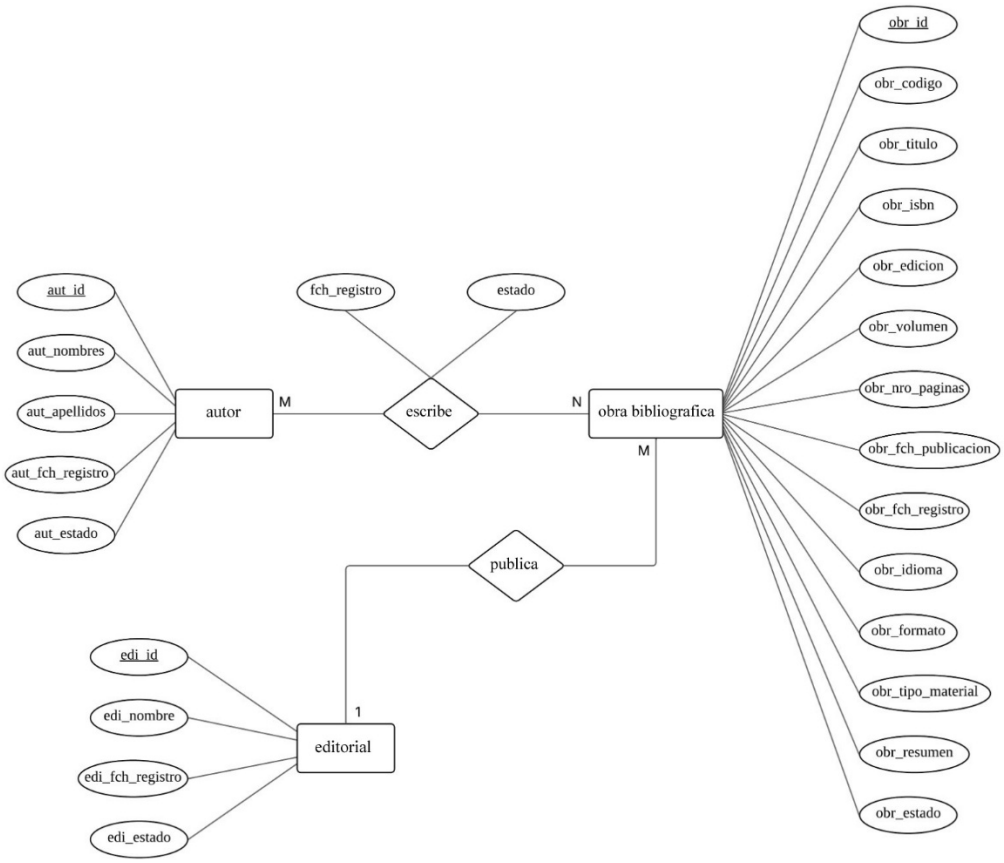


Figura 44.22. Diagrama MER del sistema de biblioteca.

Fuente: Elaboración propia.

En el diagrama anterior, los conjuntos de atributos han sido estandarizados mediante el uso de prefijos identificadores, los cuales permiten asociar de manera inmediata cada atributo con su entidad correspondiente. La estructura adoptada sigue el formato:

- <prefijo_entidad>_<nombre_campo>, donde el prefijo indica la colección a la que pertenece el atributo, y el nombre del campo representa la característica específica modelada. Todas las entidades están nombradas en minúsculas y en plural.

A continuación, se presentan de manera detallada los objetos, sus atributos y las relaciones representadas en el modelo conceptual, los cuales conforman la estructura fundamental del sistema de información bibliográfico propuesto.

Entidad: autor

La entidad autor, representa a los individuos responsables de la creación intelectual de las obras bibliográficas registradas en el sistema de la biblioteca (ver **Tabla 2.2**).

Tabla 2.2. Atributos de la entidad autor.

Atributo	**Descripción**
aut_id	Identificador único del autor.
aut_nombres	Nombres completos del autor.
aut_apellidos	Apellidos.
aut_estado	Estado actual (activo, fallecido, retirado).
aut_fecha_registro	Fecha de inclusión en el sistema.

Nota. Fuente: Elaboración propia.

Entidad: obra_bibliográfica

La entidad obra_bibliográfica agrupa la información relativa a cada material intelectual registrado en la biblioteca, incluyendo libros, tesis, informes técnicos y publicaciones académicas (ver **Tabla 2.3**).

Tabla 2.3. Atributos de la entidad obra_bibliográfica.

Atributo	**Descripción**
obr_id	Identificador único.
obr_codigo	Código patrimonial o de inventario.
obr_titulo	Título oficial de la obra.
obr_isbn	Código ISBN si aplica.
obr_edicion	Número de edición.
obr_volumen	Volumen correspondiente.
obr_nro_paginas	Total de páginas.
obr_fecha_publicacion	Fecha en la que fue publicada.
obr_fecha_registro	Fecha en la que se ingresó al sistema.
obr_idioma	Idioma principal de la obra.
obr_formato	Formato físico, digital, mixto.
obr_tipo_material	Libro, tesis, artículo, etc.
obr_resumen	Breve descripción del contenido.
obras_estado	Estado (disponible, prestado, inactivo).

Nota. Fuente: Elaboración propia.

Relación: escribe (M:N)

La relación escribe establece una correspondencia de muchos a muchos (M:N) entre las entidades autor y obra_bibliográfica. Esta relación permite representar de manera precisa los casos en que una obra puede haber sido escrita por uno o varios autores, y a su vez, un mismo autor puede haber participado en la creación de múltiples obras (ver **Tabla 2.4**).

Tabla 2.4. Atributos de la relación escribe.

Atributo	Descripción
estado	Tipo de autoría (autor principal, coautor, editor).
fecha_registro	Fecha en que se registró la colaboración.

Nota. Fuente: Elaboración propia.

Entidad: editorial

La entidad editorial representa a las casas editoriales responsables de la publicación de las obras bibliográficas registradas en la biblioteca. Su inclusión en el modelo conceptual permite capturar y gestionar información clave relacionada con el origen editorial de los materiales disponibles, lo cual es esencial para la correcta catalogación, identificación de ediciones y trazabilidad documental (ver **Tabla 2.5**).

Tabla 2.5. Atributos de la entidad editorial.

Atributo	Descripción
edi_id	Identificador único.
edi_nombre	Nombre oficial.
edi_fecha_registro	Fecha de inclusión en el sistema.
edi_estado	Activa, fusionada, cerrada.

Nota. Fuente: Elaboración propia.

La relación publica establece una correspondencia de uno a muchos (1:N) entre la entidad editorial y la entidad obra_bibliográfica. Esta relación refleja que una misma editorial puede ser responsable de la publicación de múltiples obras, mientras que cada obra bibliográfica está asociada únicamente a una editorial específica en el sistema.

Requerimientos adicionales sugeridos

Para construir un sistema bibliotecario funcional, escalable y orientado a un servicio integral, se recomienda ampliar el modelo conceptual con las siguientes entidades clave, cada una con sus respectivos atributos y relaciones asociadas. Estas incorporaciones permitirán no solo la administración básica de materiales, sino también el control de usuarios, préstamos, devoluciones y categorización temática, elementos esenciales en cualquier sistema moderno de gestión bibliográfica.

a. **Entidad: usuario (cliente)**

   Representa a los miembros de la comunidad universitaria o institucional (estudiantes, docentes, personal administrativo o

externos autorizados) que utilizan los servicios bibliotecarios, en particular el préstamo de obras bibliográficas (ver **Tabla 2.6**).

Tabla 2.6. Atributos de la entidad usuario (cliente).

Atributo	**Descripción**
usr_id	Identificador único del usuario.
usr_nombres	Nombres del usuario.
usr_apellidos	Apellidos del usuario.
usr_correo	Correo electrónico de contacto.
usr_telefono	Número telefónico personal.
usr_tipo	Tipo de usuario (estudiante, docente, administrativo).
usr_estado	Estado actual (activo/inactivo).

Nota. Fuente: Elaboración propia.

b. Entidad: bibliotecaria

Representa al personal responsable de gestionar los procesos internos de préstamo y devolución de materiales. Estas personas son quienes validan las operaciones, supervisan el sistema y dan soporte a los usuarios (ver **Tabla 2.7**).

Tabla 2.7. Atributos de la entidad bibliotecaria.

Atributo	**Descripción**
bib_id	Identificador único del personal bibliotecario.
bib_nombres	Nombres de la bibliotecaria.
bib_apellidos	Apellidos.
bib_turno	Turno asignado (mañana, tarde, noche).
bib_correo_institucional	Correo institucional/laboral.
bib_estado	Estado actual (activa/inactiva).

Nota. Fuente: Elaboración propia.

c. Entidad: **préstamo**

Registra cada evento de préstamo realizado por un usuario. Cada préstamo puede involucrar uno o varios materiales bibliográficos y es autorizado por una bibliotecaria (ver **Tabla 2.8**).

Tabla 2.8. Atributos de la entidad préstamo.

Atributo	**Descripción**
pre_id	Identificador único del préstamo (clave primaria).
usr_id (FK)	Usuario que realiza el préstamo.
bib_id (FK)	Bibliotecaria que autoriza el préstamo.

pre_fecha	Fecha en que se realiza el préstamo.
pre_fecha_devolucion_teorica	Fecha límite para devolver el préstamo (aplica al conjunto de obras).
pre_estado	Estado del préstamo (pendiente, entregado, atrasado, cancelado, etc.).

Nota. Fuente: Elaboración propia.

d. Entidad: devolución

Captura cada evento de devolución, sea total o parcial, asociado a un préstamo previamente registrado. Permite controlar el estado de cumplimiento del usuario con las políticas de préstamo (ver **Tabla 2.9**).

Tabla 2.9. Atributos de la entidad devolución.

Atributo	**Descripción**
dev_id	Identificador único de la devolución.
pre_id (FK)	Referencia al préstamo correspondiente.
dev_fecha	Fecha real de la devolución.
dev_observaciones	Notas sobre el estado del material o retrasos.

Nota. Fuente: Elaboración propia.

e. Entidad: categoría

Permite clasificar las obras bibliográficas por áreas temáticas, disciplinas académicas o tipos de contenido, favoreciendo la organización del catálogo y la recuperación de información por temas (ver **Tabla 2.10**).

Tabla 2.10. Atributos de la entidad categoría.

Atributo	**Descripción**
cat_id	Identificador único de la categoría.
cat_nombre	Nombre breve de la categoría.
cat_descripcion	Definición más amplia del área temática.

Nota. Fuente: Elaboración propia.

Relaciones sugeridas para las nuevas entidades:

Los requerimientos adicionales necesitan el establecimiento de relaciones, en la **Tabla 2.11,** se describen las relaciones sugeridas:

Tabla 2.11. Relaciones sugeridas.

Entidades relacionadas	**Tipo de Relación**	**Descripción**
usuario → préstamo	1:N	Un usuario puede tener varios préstamos.
bibliotecaria → préstamo	1:N	Una bibliotecaria puede gestionar múltiples préstamos.
préstamo ↔ obra_bibliográfica	M:N	Un préstamo puede incluir varias obras, y una obra puede estar en varios préstamos.
préstamo → devolución	1:1	Cada préstamo puede tener una devolución registrada o estar pendiente.
categoría → obra_bibliográfica	1:N	Una categoría agrupa muchas obras.

Nota. Fuente: Elaboración propia.

Recuerda que, aplicando la notación de Chen las relaciones pueden poseer atributos propios, lo cual permite capturar detalles específicos de cada instancia de la relación directamente en el rombo que vincula las colecciones participantes. Por tanto, en la relación que se establezca entre las entidades préstamo y obra_bibliográfica, se pueden incorporar atributos como cantidad, observaciones o estado_del_ejemplar dentro de la relación misma, sin necesidad de definir una colección adicional. Esta característica fortalece la expresividad del modelo conceptual, permitiendo representar de forma más directa y natural aquellas situaciones en las que una transacción involucra múltiples elementos con información asociada por cada vínculo.

b. Sistema para la gestión de ventas.

En el ámbito empresarial, contar con un sistema eficiente para gestionar las ventas es crucial para el control operativo, la atención al cliente y la

toma de decisiones estratégicas. Este caso práctico plantea el diseño de un Sistema de Información para la gestión de ventas en una empresa comercial que distribuye productos al por menor. El sistema está orientado a digitalizar y automatizar el registro de clientes, empleados, productos y transacciones de ventas, garantizando integridad de los datos y trazabilidad de cada operación.

El sistema propuesto tiene como propósito fundamental permitir:

- El registro completo de los datos personales y de contacto de los clientes.
- La administración del personal encargado de registrar las ventas.
- El manejo eficiente del inventario de productos comercializados.
- La trazabilidad detallada de cada venta, incluyendo productos vendidos, cantidades y precios unitarios aplicados en el momento de la transacción.

A partir del análisis de requerimientos, se identifican las siguientes funcionalidades clave:

i. **Gestión de Clientes:** El sistema debe permitir almacenar información relevante de cada cliente, incluyendo:

- Identificador único (cli_id).
- Nombres (cli_nombres).
- Apellidos (cli_apellidos).
- Dirección (cli_direccion).
- Teléfono (cli_telefono).
- Correo electrónico (cli_email).
- Fecha de registro en el sistema(cli_fch_registro).
- Estado (cli_estado).

ii. **Gestión de Empleados:** Cada empleado que registra ventas debe contar con una ficha que contenga:

- Identificador (emp_id).
- Nombres (emp_nombres).
- Apellidos (emp_apellidos).
- Cargo (emp_cargo).
- Fecha de incorporación (emp_fch_ingreso).
- Estado laboral (emp_estado).

iii. **Gestión de Productos:** El inventario de productos requiere mantener actualizado un conjunto de atributos esenciales como:

- Identificador único del producto (pro_id)
- Nombre (pro_nombre).
- Descripción del artículo (pro_descripcion).
- Precio unitario (pro_precio_unitario).
- Stock disponible (pro_stock).
- Fecha de registro (pro_fecha_registro).
- Estado del producto (pro_estado).

iv. **Información de la factura:** Cada transacción de venta debe registrar información clave para control administrativo y contable, incluyendo:

- Identificador único de la venta (fac_id).
- Fecha de registro en el sistema (fac_fch_registro).
- Total facturado (fac_total).
- Estado de la venta (fac_est).

Relaciones y cardinalidades del proceso de venta

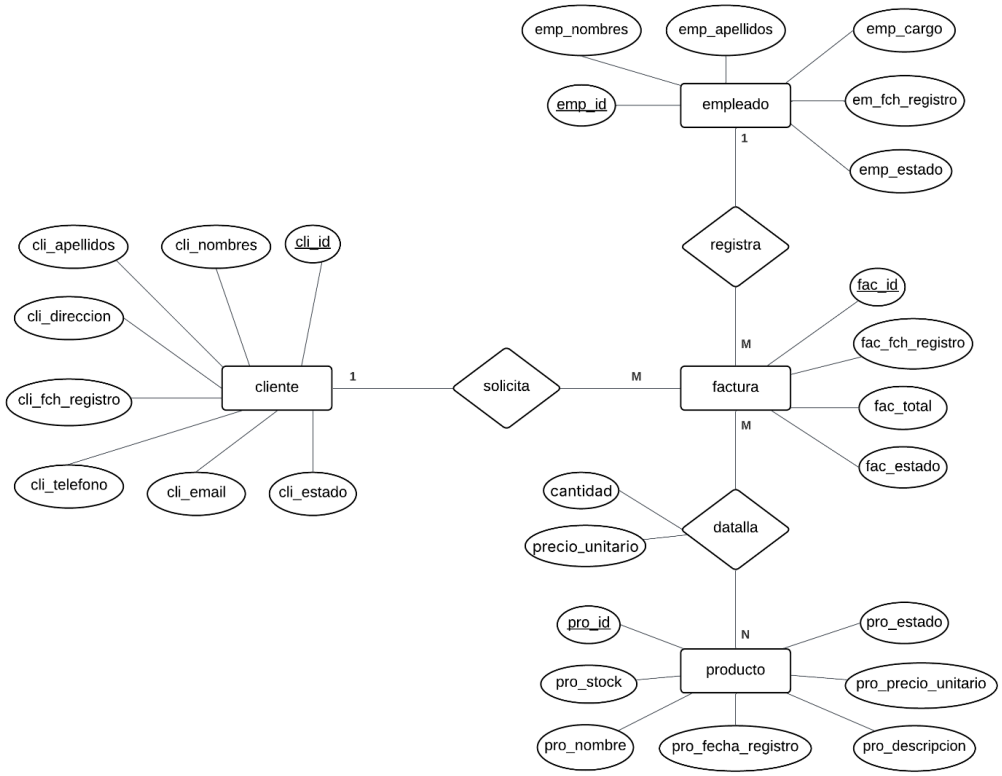
Las relaciones entre entidades se modelan según las siguientes reglas de negocio:

- Cada cliente puede solicitar múltiples facturas, pero una factura pertenece a un único cliente: Relación 1:M entre cliente y factura.
- Cada factura debe ser registrada por un solo empleado, aunque un empleado puede registrar muchas facturas: Relación 1:M entre empleado y facturas.
- Cada factura puede incluir varios productos, y un producto puede estar presente en múltiples facturas: Relación M:N entre factura y producto, implementada mediante la entidad asociativa detalla.
- La entidad detalla incluye atributos claves para registrar el detalle de cada línea de venta: cantidad y precio_unitario.

El diagrama MER de la **Figura 2.23** tiene como propósito representar de forma lógica y estructurada las principales entidades involucradas en el proceso de ventas de una organización comercial. En particular, se

modelan las entidades cliente, empleado, producto y factura, junto con sus respectivos atributos clave que permiten registrar información relevante para la gestión operativa y administrativa del negocio.

Este modelo también incorpora relaciones fundamentales que reflejan el comportamiento real del sistema: la relación "solicita" vincula a los clientes con las facturas; la relación "registra" identifica al empleado responsable de cada transacción; y la relación "detalla" permite representar la naturaleza de la relación de muchos a muchos (M:N) entre



factura y productos mediante una entidad asociativa, incluyendo atributos como cantidad y precio unitario. En síntesis, este esquema constituye un punto de partida esencial para el diseño conceptual de bases de datos transaccionales orientadas al control de ventas, la gestión de inventarios y la atención al cliente, ofreciendo una visión integral de las operaciones clave en entornos comerciales.

Figura 45.23. Diagrama MER del sistema de ventas

Fuente: Elaboración propia.

c. Sistema para la gestión de citas médicas.

En el ámbito de los servicios de salud, disponer de un sistema informatizado para la gestión de citas médicas resulta esencial para optimizar la atención al paciente, mejorar la coordinación del personal

sanitario y garantizar la trazabilidad de cada consulta. Este caso práctico plantea el diseño de un Sistema de Información orientado a digitalizar y automatizar los procesos de registro de pacientes, especialidades, médicos y citas médicas, permitiendo una administración integral de la información clínica y administrativa.

El sistema propuesto tiene como propósito fundamental permitir:

- El registro completo de los datos personales y de contacto de los pacientes.
- La administración de la información relacionada con los médicos y sus especialidades.
- El control de la agenda de citas, incluyendo fecha, hora, motivo y observaciones.
- La trazabilidad de la atención brindada, registrando el rol del médico en cada cita y el estado de la atención.

A partir del análisis de requerimientos, se identifican las siguientes funcionalidades clave:

i. **Gestión de Pacientes:** El sistema debe almacenar información relevante de cada paciente, incluyendo:
   - Identificador único (pac_id).
   - Nombres y apellidos.
   - Fecha de nacimiento y sexo.
   - Dirección y datos de contacto.
   - Estado (activo/inactivo).

ii. **Gestión de Médicos y Especialidades:** Cada médico debe estar asociado a una especialidad, registrando información como:
   - Identificador del médico (med_id).
   - Nombres, apellidos y número de colegiatura.
   - Fecha de registro y estado laboral.
   - Identificador de la especialidad (esp_id), junto con nombre, descripción y estado de la misma.

iii. **Gestión de Citas:** Cada cita debe contener información clave para la administración del proceso asistencial:
   - Identificador único de la cita (cit_id).
   - Fecha y hora de la consulta.

- Motivo de la cita y observaciones.
- Estado de la cita (agendada, atendida, cancelada).
- Fecha de registro en el sistema.

iv. **Información de la Atención:** El sistema debe registrar el rol desempeñado por el médico en la cita (por ejemplo, titular o asistente), así como el estado de la atención brindada.

Relaciones y cardinalidades del proceso de gestión de citas
Las relaciones entre entidades se modelan siguiendo estas reglas de negocio:

- Cada paciente puede agendar múltiples citas, pero cada cita pertenece a un único paciente: Relación 1:M entre paciente y cita.
- Cada cita es atendida por al menos un médico, y un médico puede atender múltiples citas: Relación M:N entre médico y cita, con atributos adicionales como rol y estado en la relación asociativa.
- Cada médico pertenece a una única especialidad, mientras que una especialidad puede tener múltiples médicos asociados: Relación 1:M entre especialidad y médico.

Con base en los requerimientos descritos anteriormente, se presenta el diagrama MER de la **Figura 2.24**, el cual representa de manera lógica y estructurada las principales entidades involucradas en el proceso de atención médica de una institución de salud. En particular, se modelan entidades tales como paciente, médico, especialidad y cita, junto con sus atributos esenciales que aseguran un registro íntegro y confiable del proceso de gestión de citas. Asimismo, se incluyen las relaciones *agenda*, que vincula a los pacientes con las citas; *atiende*, que documenta la participación de los médicos en cada consulta; y *pertenece*, que refleja la asociación entre médicos y sus especialidades. En síntesis, este esquema constituye un punto de partida fundamental para el diseño conceptual de sistemas de información hospitalarios, orientados a garantizar la eficiencia en la gestión de citas, la adecuada asignación de recursos médicos y la mejora continua en la atención al paciente.

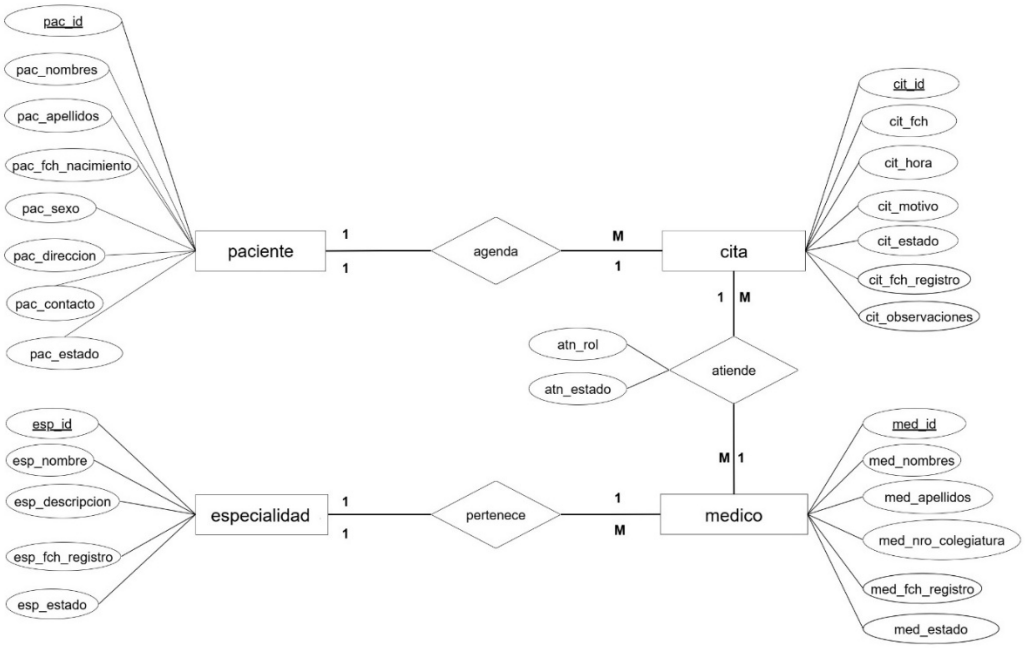


Figura 46.24. Diagrama MER del sistema de citas médicas.

Fuente: Elaboración propia.

Resumen de la unidad

La Unidad 2 ha permitido al estudiante comprender y aplicar los principios fundamentales del modelado conceptual de datos, abordando el proceso mediante el cual se abstrae y organiza la información relevante de un sistema antes de su implementación en un DBMS. Se estudió el MER como herramienta estándar para representar gráficamente los objetos, sus atributos, relaciones y restricciones semánticas mediante la notación de Chen.

A lo largo de la unidad, se profundizó en la clasificación de entidades (fuertes y débiles), el análisis de atributos (simples, compuestos, multivaluados y derivados), y la identificación de relaciones según su cardinalidad (1:1, 1:M, M:N) y participación (total o parcial). También se introdujeron las extensiones del MER, como generalización, especialización y agregación, que permiten modelar estructuras complejas con mayor expresividad.

Se expuso una metodología estructurada para construir modelos conceptuales eficientes, desde la recolección de requerimientos hasta la validación del modelo final, destacando la importancia de una representación clara y validada del universo del discurso. Finalmente, se aplicaron los conocimientos adquiridos en un caso práctico de modelado conceptual orientado al diseño de un sistema bibliotecario, reforzando la aplicación real y contextualizada de las técnicas aprendidas.

La unidad sentó así las bases para el diseño lógico relacional que será abordado en la siguiente etapa, fortaleciendo la capacidad del estudiante para construir modelos robustos, coherentes y alineados con los objetivos informacionales de cualquier organización.

Glosario de términos clave

Atributo: Propiedad que describe características específicas de una objeto o relación.

Relación: Asociación lógica entre dos o más colecciones que comparten un vínculo semántico.

Cardinalidad: Indica el número mínimo y máximo de instancias de una colección que pueden asociarse con otra.

Participación: Define si la existencia de una entidad depende obligatoriamente de su aparición en una relación (total o parcial).

Entidad, objeto o colección: Elemento del modelo de datos que agrupa atributos y representa de forma estructurada una realidad del mundo que se desea almacenar en la base de datos.

Entidad débil: Objeto cuya existencia depende de una entidad fuerte. No posee clave primaria propia.

Generalización: Abstracción de atributos comunes de varias colecciones específicas en una entidad más general.

Especialización: Proceso inverso a la generalización; permite derivar subentidades con atributos o roles particulares.

Agregación: Tratamiento de una relación como objeto, cuando se vincula a otra relación en un nivel superior.

MER: Representación conceptual de la estructura de datos mediante entidades, relaciones y atributos, útil para el análisis previo al diseño lógico.

Preguntas de autoaprendizaje

Instrucciones: Marca la opción correcta en cada caso.

1. ¿Cuál de las siguientes afirmaciones describe mejor una entidad débil?

- **a.** Tiene clave primaria propia y existe de manera independiente.
- **b.** Es una subentidad que hereda atributos.
- **c.** Depende de otra entidad y no tiene clave primaria propia.
- **d.** Representa atributos derivados de una entidad.

2. ¿Qué notación gráfica se utiliza para representar los atributos en un MER tradicional?

- **a.** Rectángulo
- **b.** Óvalo
- **c.** Rombos

d. Llaves

3. ¿Cuál es una característica de una relación muchos a muchos (M:N)?

a. Solo una instancia de una entidad puede relacionarse con otra.

b. Varias instancias de una entidad pueden relacionarse con varias de otra.

c. Requiere clave compuesta y no admite participación parcial.

d. No se utiliza en modelos conceptuales.

4. ¿Qué tipo de atributo se obtiene a partir de otro existente?

a. Compuesto

b. Multivaluado

c. Derivado

d. Clave ajena

5. ¿Qué representa la participación total en una relación?

a. Todas las entidades tienen claves externas.

b. Todas las instancias de la entidad deben participar en la relación.

c. Las entidades no participan en la relación.

d. No se pueden establecer relaciones entre entidades.

Respuestas: 1(c); 2(b); 3(b); 4(c); 5(b).

Ejercicios prácticos para resolver

Ejercicio 1: Clasificación de elementos.

Clasifica los siguientes elementos del modelo E-R en su tipo correspondiente (ver **Tabla 2.12**):

Tabla 2.12. Actividad de clasificación de elementos.

Elemento	**Tipo (Entidad, Atributo, Relación, Cardinalidad)**
Cliente	
Nombre	
Compra	
1:N	

Nota. Fuente: Elaboración propia.

Ejercicio 2: Análisis de requerimientos.

Lee la siguiente descripción y construye un modelo E-R textual:

Una institución educativa ha identificado la necesidad de implementar un sistema de información académico que permita gestionar de manera eficiente el proceso de matrícula de los estudiantes en los distintos cursos ofertados durante cada período académico.

El sistema debe permitir que cada estudiante se inscriba en uno o varios cursos por período, y a su vez, cada curso pueda ser impartido por uno o más docentes. Asimismo, un curso puede contar con múltiples estudiantes inscritos, lo cual implica una relación de muchos a muchos (M:N) entre estudiantes y cursos, con una dependencia adicional respecto a los docentes asignados.

Cada proceso de matrícula debe registrar de forma detallada la siguiente información:

- El estudiante que se matricula.
- El curso en el que se inscribe.
- El docente asignado al curso para ese estudiante.
- La fecha de matrícula.
- El estado de la matrícula, que puede tomar valores como: activo, retirado, aprobado o reprobado.

Además, el sistema debe:

- Almacenar información relevante de los docentes, incluyendo nombres completos, especialidad disciplinar y estado laboral (activo/inactivo).
- Mantener un historial completo de matrículas para cada estudiante, organizado por período académico, lo que facilitará el seguimiento del desempeño académico y la planificación curricular.

Este escenario servirá como base para construir el modelo conceptual utilizando la notación de Chen, identificando claramente las entidades principales, sus atributos, y las relaciones necesarias para representar las interacciones académicas de manera estructurada y coherente.

Ejercicio 3: Evaluación de modelos.

Observa el siguiente fragmento de un MER (ver **Figura 2.25**):

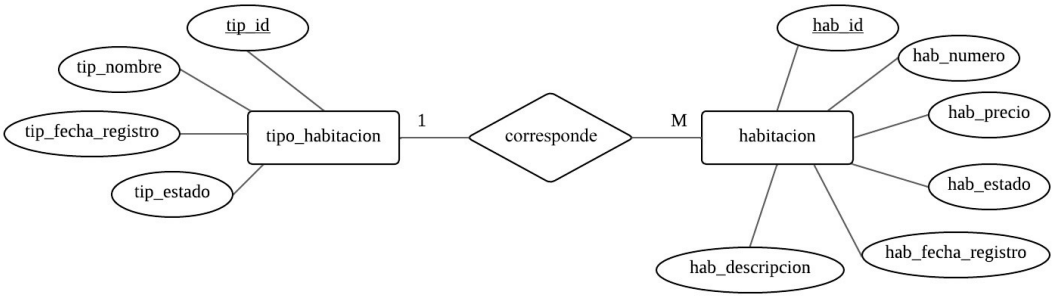


Figura 47.25. Fragmento de modelo E-R.
Fuente: Elaboración propia.

Responde lo siguiente:

a. Identifica las dos entidades fuertes (tipo_habitacion y habitación) y explica por qué no hay entidades débiles ni jerarquías de subtipos en este fragmento.

b. Justifica que tip_id sea la clave de tipo_habitacion y hab_id la de habitación.

c. ¿Podría considerarse hab_numero clave candidata si se combina con otro atributo? Indica cuál y por qué.

d. La relación corresponde es 1:M de tipo_habitacion a habitación. ¿Cómo sabes que la participación es obligatoria en el lado "1" y opcional en el lado "N"?

e. ¿Qué implicaría conceptualmente cambiar la participación mínima de habitación de 0 a 1?

f. El atributo hab_descripción aparece conectado a habitación.

g. Para cada entidad, clasifica sus atributos en:

- Identificador **(**tip_id / hab_id).
- Atributos simples **(**tip_nombre, tip_fecha_registro, tip_estado, hab_numero, hab_precio, hab_fecha_registro, hab_estado).
- Atributos derivados (si se te ocurre alguno que pudiera derivarse, por ejemplo, duración promedio de estancia)

h. Define el dominio de los atributos de cada entidad, considera como ejemplo lo siguiente:

- tip_nombre (Texto corto)
- tip_estado / hab_estado (Lista cerrada de estados, por ejemplo "activo"/"inactivo")
- hab_precio (Número positivo)

i. Agrega una entidad equipamiento y una relación M:N con habitación.

Ejercicio 4: Redacción de escenarios y representación gráfica mediante el MER.

Elabora tres ejemplos de relaciones **1: M (uno a muchos)** y tres ejemplos de relaciones **M: N** (muchos a muchos) en un contexto realista. Para cada ejemplo:

a. Identifica las entidades involucradas.
b. Describe la cardinalidad (por qué es 1:M o M:N).
c. Justifica brevemente con un escenario cotidiano (por qué este modelado refleja fielmente el caso).
d. Dibuja el MER utilizando la notación de Chen.

Por ejemplo:

> Relación 1:M: "Una ciudad puede tener varias sucursales bancarias, pero cada sucursal pertenece a una única ciudad."

Ejercicio 5: Dibuja el MER para los siguientes enunciados, estableciendo la cardinalidad de 1:M o M:N, según corresponda.

a. Un departamento académico puede tener varios profesores, pero cada profesor pertenece a un único departamento.

Una instancia de "departamento" (por ejemplo, Ingeniería de Sistemas) agrupa múltiples instancias de "profesor", mientras que cada profesor está adscrito exclusivamente a un solo departamento.

b. Un profesor puede impartir varias asignaturas (cursos), y a la vez una asignatura puede ser dictada por varios profesores (por ejemplo, clases teóricas y prácticas).

Para implementar esta relación en el modelo relacional creamos una tabla intermedia dicta(prof_id, asi_id, rol), donde rol podría indicar "titular" o "asistente".

c. En un sistema de gestión de contenidos, una categoría, por ejemplo Tecnología, engloba muchos posts, pero cada post está etiquetado inicialmente con una sola categoría principal. La categoría 'Tecnología' contiene 80 artículos, y cada artículo figura únicamente bajo una categoría principal.
d. Un estudiante puede inscribirse en varios cursos, y cada curso agrupa a múltiples estudiantes. Por ejemplo: María cursa Matemáticas, Programación y Base de Datos; a su vez, cada uno de esos cursos reúne a docenas de estudiantes diferentes.

En una base de datos de cine, una película puede contar con varios actores, y un actor puede aparecer en numerosas películas a lo largo de su carrera. Por ejemplo: El Gran Viaje es protagonizada por tres actores, mientras que Ana López aparece en más de veinte títulos distintos

Referencias bibliográficas de la Unidad 2

Aldana, C. J. S., & Motta, F. I. M. (2021). Modelo Entidad Relación. In *Modelamiento de base de datos*. https://doi.org/10.2307/j.ctv25dh3fp.6.

Capel, M. Y. J. (2025). *Bases de datos relacionales y modelado de datos. IFCT0310*. IC Editorial.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6), 377–387. https://doi.org/10.1145/362384.362685.

Cornelio, E., Rivas, C., & Hernández, J. C. (2004). *Bases de datos relacionales: diseño físico*.

Coronel, C., & Morris, S. (2023). *Database systems : design, implementation, & management* (14th ed.). Cengage.

Date, C. J. (2003). *Introduction to Database Systems* (8th ed.). Addison-Wesley.

Elmasri, R., & Shamkant, N. (2007). *Fundamentos de Sistemas de Bases de Datos* (5th ed.). Pearson Educación.

Ferraggine, V. E., & Rivero, L. C. (2005). *Patrones de Reglas del Negocio para el Enriquecimiento de un MER*. 101.

Harrington, J. L. (2016). *Relational Database Design and Implementation* (4th ed.). Morgan Kaufmann.

Marques, M. (2011). Modelo relacional Introducción y objetivos. *Bases de Datos*, *2009*.

Merchán-Manzano, O. (2018). Diseño de base de datos. In *Universidad del Azuay Casa Editora*. Casa Editora Universidad del Azuay. https://doi.org/10.33324/ceuazuay.5

Nevado Cabello, M. V. (2010). *Introducción a las Bases de Datos Relacioneles* (1st ed.). Visión Libros.

Osorio Rivera, F. L. (2008). *BASE DE DATOS RELACIONALES. TEORÍA Y PRÁCTICA* (Issue 1). Fondo Editorial ITM.

Peter P. Chen. (1976). The Entity-Relationship Model—toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, *1*(1), 9–36. https://doi.org/10.1145/320434.320440;WGROUP:STRING:ACM

Piattini, M., & Castaño, A. de M. (1999). *Fundamentos y modelos de bases de datos* (2nd ed.). Ra-Ma S.A.

PIÑEIRO GOMEZ, J. M. (2024). Diseño de base de datos relacionales. *Ediciones Paraninfo*.

Pons, O. (2005). *Introduccion a Las Bases De Datos: El Modelo Relacional* (1st ed.). Ediciones Paraninfo, S.A.

Saiedian, H. (1997). Una evaluación del del modelo entidad relación extendido. In *Information and Software Technology* (Vol. 39).

Schmal, R. (2001). *Modelamiento de datos y el modelo entidad-relación*. Universidad de Talca.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *FUNDAMENTOS DE BASES DE DATOS* (4th ed.). McGraw-Hill.

Unidad 3: Diseño Lógico Relacional

Introducción de la Unidad

Una vez establecido el modelo conceptual del sistema mediante la representación Entidad-Relación, el siguiente paso en el desarrollo de una base de datos robusta es su transformación al modelo lógico relacional. Esta unidad profundiza en dicha transición, abordando de manera sistemática el proceso mediante el cual las entidades, relaciones y atributos del modelo conceptual se convierten en estructuras formales compatibles con un Sistema de Gestión de Bases de Datos Relacional (RDBMS).

Se inicia con la presentación de los fundamentos del modelo relacional propuesto por Codd, analizando sus principales componentes: relaciones, atributos, dominios, claves primarias y foráneas. A continuación, se introduce una metodología detallada para convertir el Modelo Entidad-Relación (MER) en esquemas relacionales, preservando la semántica, la integridad de los datos y las restricciones del diseño original. Este proceso se refuerza con el estudio de diversos casos prácticos que reflejan escenarios reales, incluyendo relaciones 1:1, 1:M y M:N, así como la representación de entidades débiles y atributos multivaluados.

La unidad también incorpora el estudio de dependencias funcionales como fundamento teórico para la normalización, técnica indispensable para optimizar la estructura lógica del sistema, eliminar redundancias y prevenir anomalías de inserción, actualización o eliminación. Se revisan las formas normales (1FN, 2FN, 3FN y BCNF) con ejemplos aplicados y análisis de sus implicaciones estructurales.

Finalmente, se analizan las restricciones de integridad que permiten mantener la calidad y consistencia de los datos dentro del esquema lógico, y se culmina con la aplicación de todos los conocimientos adquiridos en un caso práctico de transformación de un modelo conceptual bibliotecario, sentando así las bases para una implementación eficiente y escalable en entornos reales.

Objetivos de aprendizaje

Al finalizar esta unidad, el estudiante será capaz de:

1. Comprender los principios fundamentales del modelo relacional, reconociendo su relevancia como pilar estructural en los sistemas de gestión de bases de datos y como estándar en el diseño lógico contemporáneo.
2. Transformar de manera rigurosa los esquemas conceptuales generados mediante el MER en esquemas lógicos relacionales, garantizando su correcta preparación para la implementación en un RDBMS.

3. Aplicar de forma sistemática las reglas de normalización**,** avanzando progresivamente hasta la Tercera Forma Normal (3FN) y, en casos pertinentes, a la Forma Normal de Boyce-Codd (BCNF), con el fin de optimizar la integridad y la eficiencia del diseño.
4. Detectar e interpretar anomalías de diseño derivadas de la redundancia, la actualización, la inserción o la eliminación, proponiendo soluciones adecuadas mediante la restructuración del modelo relacional.
5. Implementar restricciones de integridad referencial y de dominio, asegurando la coherencia semántica de los datos, así como la validez y consistencia operativa del sistema.
6. Evaluar críticamente la calidad estructural de un diseño lógico relacional, considerando criterios como eficiencia, simplicidad, integridad y adaptabilidad futura en distintos contextos de aplicación.

Preguntas de enfoque

- ¿De qué manera se representa estructuralmente la información en el modelo relacional, y cuáles son sus componentes esenciales?
- ¿Qué se entiende por normalización en el contexto del diseño de bases de datos, y por qué resulta fundamental para evitar inconsistencias y redundancias?
- ¿Cuál es el procedimiento adecuado para transformar un MER, en un conjunto de tablas relacionales preservando la semántica y las restricciones del modelo conceptual?

Desarrollo de contenidos

3.1. Fundamentos del Modelo Relacional

De acuerdo con Codd (1990), el Modelo Relacional se constituye en una de las estructuras más robustas y ampliamente adoptadas en el ámbito de las bases de datos. Este modelo organiza la información en relaciones, comúnmente representadas como tablas, compuestas por filas o tuplas y columnas o atributos (Estrada Rios, 2020). Cada relación posee un nombre único y representa una entidad o una asociación significativa del mundo real.

Características clave del modelo relacional:

- Cada relación se representa como una tabla bidimensional que contiene filas y columnas.
- Cada tupla (fila) corresponde a una instancia única o registro individual.

- Cada atributo (columna) se asocia a un dominio de valores, es decir, a un tipo de dato específico.
- Toda relación debe contar con una clave primaria, que identifique de forma unívoca a cada tupla.
- Las claves foráneas se utilizan para establecer relaciones entre diferentes tablas, garantizando la integridad referencial del modelo.

Según Date (2003), "el Modelo Relacional se caracteriza por su simplicidad, su base matemática sólida y su capacidad para representar cualquier tipo de información estructurada". Esta combinación de rigor formal y flexibilidad práctica lo convierte en la base conceptual de los Sistemas de Gestión de Bases de Datos Relacionales (RDBMS), utilizados en múltiples contextos organizacionales y tecnológicos.

3.2. Componentes estructurales del Modelo Relacional

De acuerdo con Nevado Cabello (2010), el modelo relacional no solo representa los datos en forma de tablas; también se apoya en una estructura interna precisa y bien definida, fundamentada en principios matemáticos, que permite garantizar la consistencia, coherencia y trazabilidad de la información. Comprender estos componentes es esencial para diseñar bases de datos eficientes y alineadas con los requerimientos funcionales de cualquier sistema de información.

a. Relación (Tabla)

Una relación es el componente central del modelo. Se representa gráficamente como una tabla que contiene filas y columnas (ver **Figura 3.1**). Cada relación posee un nombre único dentro del esquema y representa una entidad o una relación significativa del mundo real (como estudiante, curso, matrícula, entre otras).

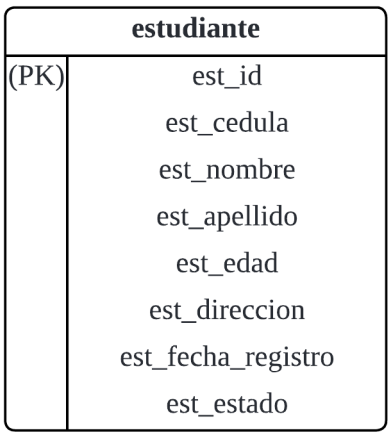


Figura 3.1. Tabla o entidad.

Fuente: Elaboración propia.

b. Tupla (Fila)

Cada tupla es una fila individual dentro de una tabla, que almacena los valores correspondientes a una instancia específica de la entidad. Por ejemplo, una tupla en la tabla estudiantes podría representar a una persona particular con su nombre, correo y número de matrícula (ver **Tabla 3.1**).

Tabla 3.1. Ejemplificación de Tupla o Fila.

est_id	est_cedula	est_nombre	est_apellido	est_edad	est_direccion	est_fecha_registro	est_estado
1	0102030405	Laura	González	21	Av. Central 123	2025-05-05	Activo

Nota. Fuente: Elaboración propia.

c. **Atributo (Columna)**

Los atributos son las columnas de una relación, y cada uno representa una característica o propiedad del objeto modelado. Por ejemplo, est_cédula, est_nombre, est_apellido, est_edad, est_fecha_registro y est_estado serían atributos de la entidad estudiante (ver **Figura 3.2**).

est_nombre
Ana

Figura 3.2. Atributo o columna de una entidad.

Fuente: Elaboración propia.

El esquema relacional define la estructura general de la base de datos, es decir, el conjunto de todas las relaciones, junto con los atributos de cada una, las claves primarias y foráneas, y las restricciones asociadas. Es el plano lógico sobre el cual se construye el sistema.

d. Dominio

Cada atributo en una relación está asociado a un dominio, que especifica el conjunto de valores válidos que puede contener. Este dominio define el tipo de dato (como entero, cadena de texto, fecha) y en algunos casos, valores restringidos (como enumeraciones o rangos).

e. Grado (Aridad)

El grado de una relación se refiere al número de atributos que contiene. Por ejemplo, si una tabla tiene cinco columnas, su grado es cinco. Este

concepto es útil para analizar la complejidad de una relación. Por ejemplo, para la tabla estudiante (ver **Figura 3.1)**, su grado es 8.

f. Cardinalidad

La cardinalidad representa el número de tuplas que hay en una relación en un momento dado. Una tabla con 200 registros tiene una cardinalidad de 200. A nivel del MER, la cardinalidad también se refiere a la cantidad de instancias que participan en una relación entre entidades.

g. Clave Primaria

La clave primaria es un conjunto mínimo de atributos que identifica de forma única cada tupla dentro de una relación. Es esencial para evitar duplicados y asegurar la unicidad de los datos. Por ejemplo, en la **Figura 3.3,** est_id es clave primaria (PK) en la entidad estudiante.

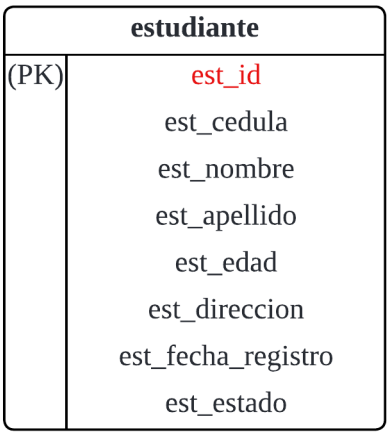


Figura 3.3. Clave primaria en entidad estudiante.

Fuente: Elaboración propia.

h. Clave Foránea

Una clave foránea (foreign key) es un atributo o conjunto de atributos en una tabla que establece una relación con la clave primaria de otra tabla. Su función es garantizar la integridad referencial, asegurando que los datos vinculados existan y se mantengan consistentes. Por ejemplo, en la **Figura 3.4,** ciu_id es FK en la entidad estudiante.

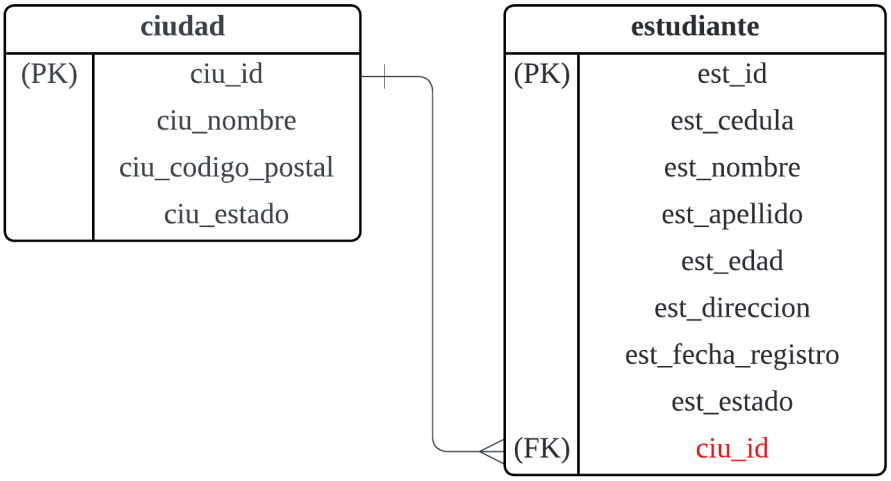


Figura 3.4. Clave foránea en entidad estudiante.

Fuente: Elaboración propia.

Estos componentes no solo permiten representar información de forma estructurada, sino que también constituyen la base para aplicar reglas de normalización, restricciones de integridad y consultas SQL eficaces. Entender cómo interactúan entre sí es clave para desarrollar esquemas lógicos robustos, sostenibles y alineados con las necesidades reales de las organizaciones.

3.3. Conversión del MER al Modelo Relacional

De acuerdo con el criterio de Lorente Puchades et al. (2019), la conversión de un MER al modelo relacional constituye una etapa crítica en el proceso de diseño lógico. Esta transformación permite traducir la representación conceptual del sistema compuesta por entidades, relaciones y atributos, en un esquema relacional estructurado y listo para ser implementado en un RDBMS. Este proceso no solo garantiza la fidelidad semántica del modelo original, sino que también optimiza la organización de los datos, reduciendo la ambigüedad y preservando las reglas de integridad.

Reglas básicas para la conversión

1. **Entidades fuertes → Tablas individuales**

   Cada entidad fuerte se convierte en una tabla, donde sus atributos se traducen directamente en columnas. La clave primaria se mantiene para identificar de forma única a cada tupla.

   Ejemplo:

   La entidad vehículo se convierte en una tabla con atributos como veh_placa, veh_marca, veh_modelo, etc. Su clave primaria veh_id garantiza la unicidad de cada fila, lo cual se puede apreciar en la **Figura 3.5**.

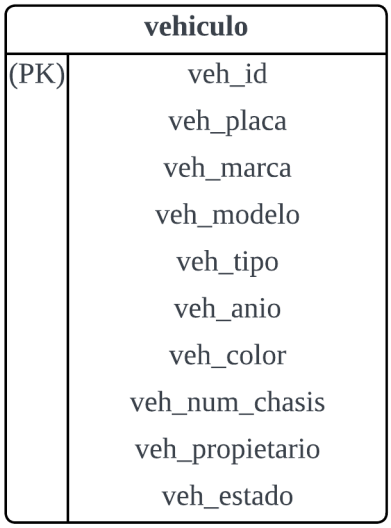


Figura 3.5. Entidad fuerte - vehiculo.

Fuente: Elaboración propia.

2. **Atributos compuestos → Atributos simples**

   Los atributos compuestos se descomponen en sus componentes más simples. Por ejemplo, nombre_completo podría dividirse en nombre y apellido.

3. **Atributos multivaluados → Tablas adicionales**

   Cuando un atributo puede tener múltiples valores para una misma entidad (como varios teléfonos), se crea una **nueva tabla** que los almacene, incluyendo una **clave foránea** que haga referencia a la entidad original.

4. **Entidades débiles → Tablas con clave compuesto**

   En el MER, las entidades débiles carecen de una clave propia que las identifique de forma única; su existencia depende de una entidad fuerte (Miguel Castaño et al., 2000).Al llevar este diseño al modelo relacional, cada entidad débil se transforma en una tabla cuya clave primaria consta de:

   - La clave primaria de la entidad fuerte (como clave foránea).
   - Uno o más de sus propios atributos, conformando así una clave primaria compuesta.

   De este modo, se preserva la integridad referencial y se asegura que cada registro de la entidad dependiente pueda distinguirse inequívocamente, al combinar su identificador propio con el de la tabla de la que depende.

Caso de estudio 1: empleado – carga_familiar

La **Figura 3.6** ilustra un MER en el que la entidad empleado actúa como elemento fuerte y se vincula en una relación 1:M con la entidad carga_familiar, de carácter débil. En este esquema, un mismo empleado puede tener asignadas múltiples cargas familiares, pero cada registro de carga familiar carece de identidad independiente y se identifica únicamente combinando su propio atributo clave con la clave primaria del empleado. Esta construcción, al emplear una clave primaria compuesta en la tabla de cargas familiares, garantiza tanto la unicidad de cada dependencia como la integridad referencial entre las dos entidades.

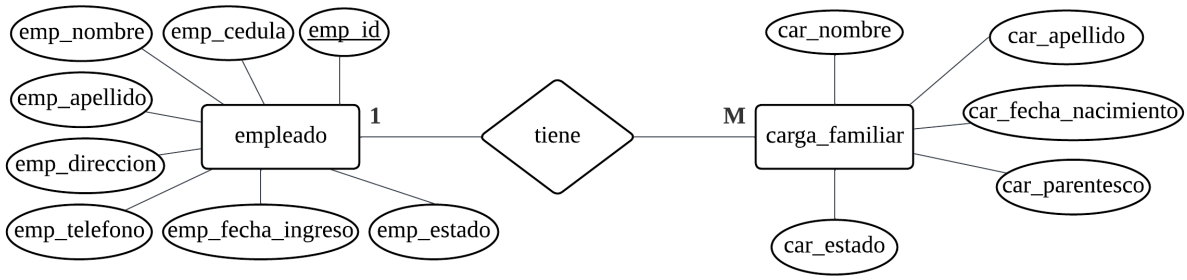


Figura 3.6. MER entidad débil carga_familiar.

Fuente: Elaboración propia.

La transformación al modelo relacional de la entidad débil carga_familiar, se realiza como una tabla cuyo emp_id funciona como clave foránea y, a su vez, se combina con su propio atributo car_nombre para conformar una clave primaria compuesta. De este modo, cada registro de carga familiar queda ligado inequívocamente a su empleado correspondiente.

La **Figura 3.7** ilustra este esquema resultante: la tabla empleado conserva una clave primaria sencilla emp_id, mientras que carga_familiar usa la dupla emp_id y car_nombre para asegurar la unicidad de cada dependencia y mantener la integridad referencial entre ambas tablas.

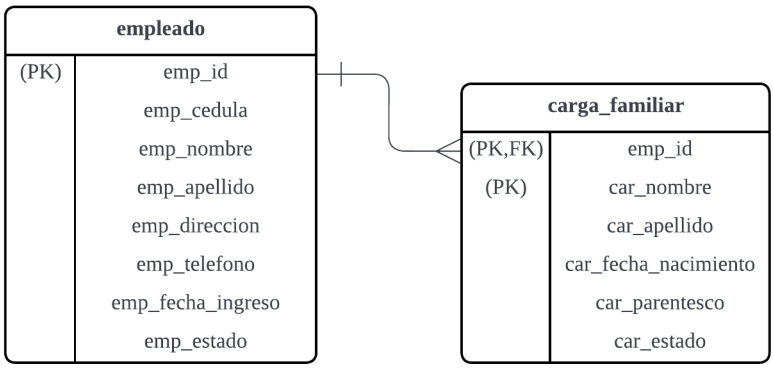


Figura 3.7. Entidad débil carga_familiar con clave compuesta.

Fuente: Elaboración propia.

5. **Relaciones 1:1 → Clave foránea en una de las tablas**

   En una relación uno a uno, se puede agregar una clave foránea en cualquiera de las dos tablas, preferentemente en aquella que tenga participación total o menor volumen de registros.

   Caso de estudio 1: persona – pasaporte

   El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 48.11,** Del mismo modo, una persona puede tener asignado un único pasaporte, y cada pasaporte está vinculado a una sola persona. En este caso, la tabla pasaporte incluye per_id como clave foránea, estableciendo así la relación exclusiva entre ambos (ver **Figura 3.8)**.

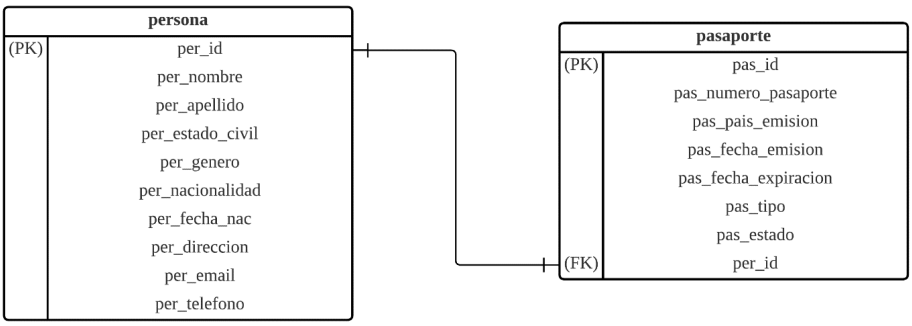


Figura 3.8. Relación 1:1 persona - pasaporte.

Fuente: Elaboración propia.

Caso de estudio 2: vehículo – registro_vehicular

El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 49.12,** Cada vehículo tiene un único registro vehicular y viceversa. La clave primaria veh_id se

convierte en clave foránea dentro de la tabla registro_vehicular (ver **Figura 3.9)**.

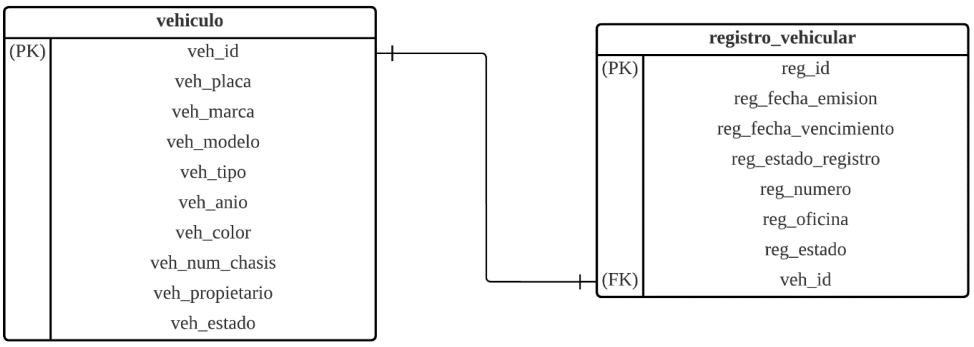


Figura 3.9. Relación 1:1 vehículo – registro_vehicular.

Fuente: Elaboración propia.

Esta estrategia de diseño permite mantener la integridad referencial y facilita la consulta eficiente de datos relacionados. En términos de implementación, la ubicación de la clave foránea se decide considerando criterios prácticos como la obligatoriedad de existencia, el acceso más frecuente, o la frecuencia de actualización de los registros.

6. **Relaciones 1:M → Clave foránea en el lado "muchos"**

   Las relaciones uno a muchos (1:M) son sumamente comunes en el diseño de bases de datos relacionales. En este tipo de vínculo, una entidad del "lado uno" puede estar relacionada con múltiples instancias del "lado muchos", mientras que cada instancia del lado muchos se asocian exclusivamente con una del lado uno.

   En el modelo relacional, la clave primaria de la tabla del lado "uno" se convierte en una clave foránea dentro de la tabla del lado "muchos". Esta clave foránea permite establecer el vínculo directo entre ambas tablas y mantener la integridad referencial entre los datos relacionados.

   Caso de estudio 1: ciudad y cliente

   El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 50.14,** en este ejemplo una ciudad puede tener múltiples clientes asociados, mientras que cada cliente reside en una única ciudad (ver **Figura 3.10**). La relación de uno a muchos se representa incorporando el atributo ciu_id (clave primaria de la tabla ciudad) como clave foránea en la tabla cliente. Esto permite identificar fácilmente en qué ciudad vive cada cliente, manteniendo la integridad de la relación entre ambas entidades.

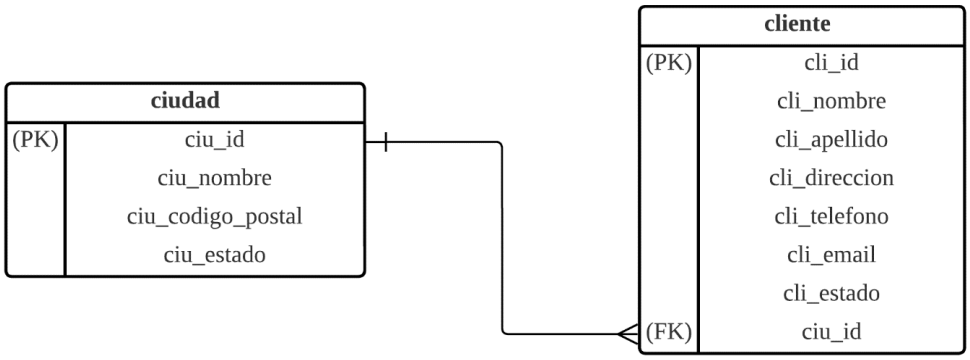


Figura 3.10. Relación 1:M ciudad – cliente.

Fuente: Elaboración propia.

Caso de estudio 2: categoría y producto

El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 51.15,** en este escenario una categoría puede agrupar múltiples productos, pero cada producto solo puede pertenecer a una categoría (ver **Figura 3.11**). Por tanto, el identificador cat_id (clave primaria de categoría) se convierte en clave foránea en la tabla producto, reflejando esta relación uno a muchos. Esta estructura permite clasificar adecuadamente los productos y establecer filtros o agrupaciones con base en su categoría.

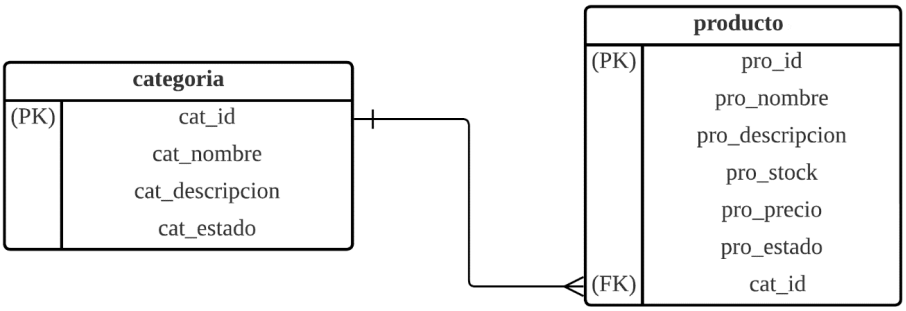


Figura 3.11. Relación 1:M categoría – producto.

Fuente: Elaboración propia.

7. **Relaciones M:N → Tabla intermedia**

   En el modelo relacional, una relación de tipo muchos a muchos (M:N) no puede representarse de manera directa entre dos tablas. Para modelarla correctamente, se requiere la creación de una tabla intermedia que contenga como claves foráneas las claves primarias de las dos entidades relacionadas. Estas claves foráneas, combinadas, conforman normalmente una clave primaria compuesta, garantizando la unicidad de cada combinación de registros relacionados.

Además, esta tabla intermedia puede incluir atributos propios de la relación, aquellos que no pertenecen exclusivamente a una entidad, sino que surgen de la interacción entre ambas.

Caso de estudio 2: Relación M:N entre pedido y producto

El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 52.17,** en el cual un pedido puede incluir varios productos, y un producto puede ser parte de múltiples pedidos (ver **Figura 3.12)**. Para reflejar esta dinámica, se emplea la tabla intermedia pedido_producto, que registra no solo las claves foráneas (ped_id, pro_id), sino también información crucial de la transacción como la cantidad solicitada, el subtotal correspondiente y el estado del producto en ese pedido.

Este enfoque permite llevar un control detallado de los productos contenidos en cada pedido, su valor y su estado operativo.

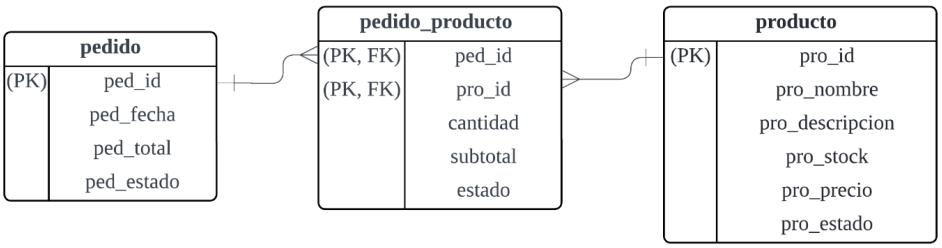


Figura 3.12. Relación M:N pedido–pedido_producto-producto.

Fuente: Elaboración propia.

Caso de estudio 1: Relación M:N entre autor y libro

El desarrollo del modelo relacional tiene como base el MER desarrollado en la **Unidad 2- Figura 53.18,** en el cual un autor puede participar en la redacción de múltiples libros, y a su vez, un libro puede ser coescrito por varios autores. Para modelar esta relación en el esquema relacional, se crea la tabla autor_libro, que actúa como intermediaria entre las tablas autor y libro (ver **Figura 3.13)**.

En esta tabla intermedia no solo se almacenan los identificadores (aut_id, lib_id), sino también atributos adicionales como el rol desempeñado por el autor y su orden de aparición, lo cual enriquece el significado de la relación.

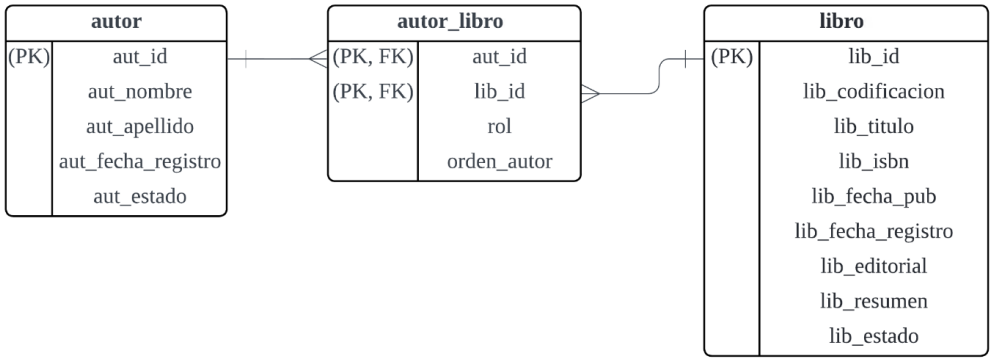


Figura 3.13. Relación M:N autor–autor_libro-libro.

Fuente: Elaboración propia.

3.4. Dependencias funcionales

Las dependencias funcionales constituyen un concepto esencial en el análisis lógico de bases de datos, ya que permiten expresar relaciones lógicas entre conjuntos de atributos dentro de una tabla (Gardarin, 1990). Estas se producen cuando el valor de un atributo (o conjunto de atributos) determina de forma única el valor de otro atributo. Formalmente, se dice que B depende funcionalmente de A si, para cada valor de A, existe un único valor asociado de B.

Comprenderlas resulta clave para la correcta aplicación de los procesos de normalización y para garantizar la integridad estructural de los datos.

Notación formal:

A → B

Se interpreta como: "A determina funcionalmente a B", o "B depende funcionalmente de A".

Por ejemplo; si en una relación estudiantes, el atributo est_id determina el est_nombre, se expresa como: est_id → est_nombre. Esto significa que, para cada identificador de estudiante (est_id), debe haber un único nombre asociado. Si este criterio no se cumple, estaríamos ante un diseño inconsistente o mal estructurado.

Una dependencia funcional expresa una relación lógica entre dos conjuntos de atributos. Se dice que un atributo B depende funcionalmente de A si cada valor de A está asociado con un único valor de B.

3.5. Proceso de Normalización

Según Alma Muñoz (2019), el proceso de normalización representa una de las técnicas más importantes en el diseño lógico de bases de datos. Su propósito fundamental es mejorar la estructura de las tablas, eliminando redundancias

innecesarias y previniendo anomalías en operaciones de inserción, eliminación o actualización de datos.

Desde una perspectiva académica y profesional, normalizar una base de datos involucra aplicar un conjunto de reglas formales denominadas formas normales, las cuales permiten refinar progresivamente los esquemas relacionales hasta alcanzar una estructura más coherente, funcional y alineada con los requerimientos del sistema (Giménez, 2019).

¿Por qué es necesaria la normalización?

- Para reducir la duplicación de datos que conlleva a inconsistencias.
- Para evitar anomalías de modificación, como tener que actualizar el mismo dato en múltiples lugares.
- Para garantizar integridad semántica, asegurando que las relaciones entre datos se mantengan válidas y claras.

¿Qué implica aplicar formas normales?

Cada forma normal aborda un tipo particular de problema estructural:

- La **Primera Forma Normal (1FN)** se enfoca en la atomicidad de los datos.
- La **Segunda Forma Normal (2FN)** elimina dependencias parciales.
- La **Tercera Forma Normal (3FN)** elimina dependencias transitivas.
- La **Forma Normal de Boyce-Codd (BCNF)** proporciona un refinamiento adicional para casos avanzados.

A medida que avancemos por cada una de estas formas normales, exploraremos sus fundamentos, criterios técnicos y ejemplos prácticos aplicados a esquemas reales. Comprender este proceso no solo mejora la calidad de diseño, sino que también contribuye a una mayor eficiencia en las consultas, un menor costo de mantenimiento y una evolución más segura del sistema de información (Santiso & Nichita, 2018).

Primera Forma Normal (1FN)

Según Albarak & Bahsoon (2018), la Primera Forma Normal (1FN) es el punto de partida obligatorio en el proceso de normalización. Su objetivo es garantizar que todos los atributos de una tabla contengan valores atómicos, es decir, indivisibles. Esto significa que cada celda de la tabla debe contener una sola unidad de información, sin repeticiones internas ni listas de valores múltiples.

El principio fundamental de 1FN, enfatiza en que una relación está en 1FN si y solo si cada uno de sus atributos contiene únicamente valores simples, y no

existen grupos repetitivos o conjuntos de valores dentro de una misma columna. Los problemas comunes cuando no se cumple 1FN, son los siguientes:

- Dificultad para realizar búsquedas precisas.
- Ambigüedad en los valores almacenados.
- Anomalías de inserción y eliminación de datos parciales.

A continuación, se ejemplifica de forma práctica la 1FN: Supongamos la tabla estudiante (ver **Tabla 3.2**), no normalizada:

Tabla 3.2. Tabla estudiante_telefonos.

est_id	est_nombre	est_telefonos
101	Ana Ruiz	099112233, 072345678
102	Pedro León	098998877

Nota. Fuente: Elaboración propia.

Aquí, el atributo est_telefonos contiene múltiples valores separados por comas, lo que viola el principio de atomicidad. La solución al aplicar 1FN consiste en: Dividir el atributo multivaluado en una tabla adicional (ver **Tabla 3.3;** ver **Tabla 3.4)**:

Tabla 3.3. Tabla estudiante.

est_id	est_nombre
101	Ana Ruiz
102	Pedro León

Nota. Fuente: Elaboración propia.

Tabla 3.4. Tabla estudiante_telefonos.

est_id	teléfono
101	099112233
101	072345678
102	098998877

Nota. Fuente: Elaboración propia.

De esta manera, se garantiza que cada valor esté aislado en una celda individual, y que el diseño cumpla con la 1FN. Los beneficios de cumplir 1FN son los siguientes:

- Facilita las consultas SQL.
- Mejora la integridad de los datos.
- Permite aplicar formas normales más avanzadas en etapas posteriores.

La 1FN es más que una formalidad técnica: es una base estructural imprescindible para construir modelos relacionales sólidos, precisos y sostenibles.

Segunda Forma Normal (2FN)

De acuerdo con el criterio de Efendy (2018), la Segunda Forma Normal (2FN) representa un paso esencial tras cumplir con la 1FN. Su objetivo es eliminar las dependencias parciales que existen cuando un atributo no clave depende solo de una parte de una clave primaria compuesta, y no de toda ella.

El principio fundamental de la 2FN establece que una relación está en 2FN si y solo si está en 1FN y todos los atributos no clave dependen completamente de la clave primaria, no solo de una parte de ella.

Los problemas comunes cuando no se cumple 2FN incluyen:

- Redundancia de datos.
- Anomalías de actualización.
- Anomalías de inserción o eliminación de registros parciales.

A continuación, se ejemplifica de forma práctica la 2FN: Supongamos tiene una tabla **matricula_curso** en 1FN (ver **Tabla 3.5)**, pero no en 2FN:

Tabla 3.5. matricula_curso

est_id	curso_id	est_nombre	curso_nombre
201	INF101	Carla Paz	Informática
202	INF101	Diego Ríos	Informática

Nota. Fuente: Elaboración propia.

Aquí, la clave primaria es compuesta (est_id, curso_id). Sin embargo, los atributos est_nombre y curso_nombre no dependen de toda la clave compuesta, sino solo de una parte (est_id y curso_id, respectivamente). Esto viola la 2FN.

La solución al aplicar 2FN consiste en: Separar la información en tres tablas relacionadas (ver **Tabla 3.6;** ver **Tabla 3.7;** ver **Tabla 3.8)**:

Tabla 3.6. Tabla estudiante

est_id	est_nombre
201	Carla Paz
202	Diego Ríos

Nota. Fuente: Elaboración propia.

Tabla 3.7. Tabla curso

curso_id	curso_nombre
INF101	Informática

Nota. Fuente: Elaboración propia.

Tabla 3.8. Tabla matricula

est_id	curso_id
201	INF101
202	INF101

Nota. Fuente: Elaboración propia.

Con esta restructuración, cada atributo está asociado únicamente a su clave primaria correspondiente, y la relación de matrícula solo almacena las asociaciones entre estudiantes y cursos. Los beneficios de cumplir 2FN son los siguientes:

- Disminuye la redundancia.
- Facilita la integridad de los datos.
- Mejora la eficiencia en las actualizaciones y consultas.

Aplicar correctamente la 2FN es fundamental para construir modelos lógicos más consistentes y resistentes a errores de diseño estructural.

Tercera Forma Normal (3FN)

La Tercera Forma Normal (3FN) representa un refinamiento adicional en el proceso de normalización. Su propósito es eliminar las dependencias transitivas, es decir, cuando un atributo no clave depende de otro atributo no clave, en lugar de depender directamente de la clave primaria (Demba, 2013).

El principio fundamental de la 3FN establece que una tabla está en Tercera Forma Normal si:

- Cumple con la 2FN.
- No existen atributos no clave que dependan transitivamente de la clave primaria.

Problemas comunes cuando no se cumple 3FN:

- Inconsistencias de datos.
- Redundancia innecesaria.
- Anomalías en las actualizaciones.

A continuación, se ejemplifica de forma práctica la 3FN: Supongamos tiene una tabla empleado_departamento (ver **Tabla 3.9**):

Tabla 3.9. Tabla empleado_departamento**.**

emp_id	emp_nombre	dept_id	dept_nombre
1	Laura M.	D01	Finanzas
2	José P.	D02	Marketing

Nota. Fuente: Elaboración propia.

Aquí, emp_id es la clave primaria. El atributo dept_nombre depende de dept_id, que a su vez depende de emp_id, lo que constituye una dependencia transitiva. La solución al aplicar 3FN consiste en: Separar las dependencias transitivas creando nuevas tablas (ver **Tabla 3.10;** ver **Tabla 3.11**):

Tabla 3.10. Tabla empleados.

emp_id	emp_nombre	dept_id
1	Laura M.	D01
2	José P.	D02

Nota. Fuente: Elaboración propia.

Tabla 3.11. Tabla departamentos.

dept_id	dept_nombre
D01	Finanzas
D02	Marketing

Nota. Fuente: Elaboración propia.

Los beneficios de cumplir 3FN son los siguientes:

- Se reduce aún más la redundancia.
- Mejora la consistencia y claridad del modelo.
- Facilita el mantenimiento y la evolución del sistema.

La 3FN contribuye significativamente a mejorar la estructura lógica de la base de datos, eliminando relaciones indirectas entre los datos que pueden comprometer la integridad del sistema.

Forma Normal de Boyce-Codd (BCNF)

La Forma Normal de Boyce-Codd (BCNF) es una versión más estricta de la 3FN. Fue propuesta por Raymond Boyce y Edgar F. Codd para resolver ciertos casos en los que, a pesar de que una relación cumple con la 3FN, aún persisten algunas anomalías debido a dependencias funcionales no deseadas en presencia de claves candidatas compuestas.

Una relación está en BCNF si, para cada una de sus dependencias funcionales no triviales ($A \rightarrow B$), el lado izquierdo A es una superclave (Köhler & Link, 2018). Es decir, todo determinante debe ser una clave candidata. Mientras que en

3FN se permite que un atributo no clave determine otro si este último es parte de una clave candidata, en BCNF ni siquiera esto se permite. BCNF exige que no exista ninguna dependencia funcional en la que un atributo no clave determine otro atributo, aunque sea parte de una clave.

A continuación, se ejemplifica de forma práctica la BCNF: Supongamos una tabla con información sobre asignaciones de profesores a cursos y aulas (ver **Tabla 3.12**):

Tabla 3.12. Tabla asignacion_curso_aula_profesor.

curso_id	aula_id	profesor
MAT101	A1	Luis Pérez
BIO102	A2	Alberto Torres
MAT101	A1	Luis Pérez

Nota. Fuente: Elaboración propia.

Dependencias funcionales:

- La dependencia funcional curso_id, aula_id → profesor indica que la combinación del curso y el aula determina de forma única al profesor asignado. Esto significa que si conocemos el identificador del curso (curso_id) y el identificador del aula (aula_id), podemos saber exactamente qué profesor está a cargo de esa asignación. Esta dependencia sí cumple con la Forma Normal de Boyce-Codd (BCNF) porque el lado izquierdo de la dependencia (curso_id, aula_id) es una clave primaria compuesta, es decir, una superclave que identifica de manera única cada fila de la tabla.
- La dependencia funcional profesor → aula_id sugiere que cada profesor está asociado a una única aula, lo cual implica que, si conocemos al profesor, podemos determinar el aula correspondiente. No obstante, esta dependencia viola la Forma Normal de Boyce-Codd (BCNF) si la clave primaria de la tabla es compuesta, como curso_id, aula_id. En este caso, profesor no es una superclave, pero determina otro atributo (aula_id), lo que genera una dependencia funcional parcial que no parte de una clave candidata completa. Este tipo de diseño puede provocar anomalías de actualización, como redundancia innecesaria y pérdida de información ante eliminaciones. Para cumplir con BCNF, la tabla debería reestructurarse separando la relación profesor → aula_id en una nueva tabla, asegurando que todas las dependencias funcionales partan exclusivamente de superclaves.

La solución aplicando BCNF, consiste en dividir la tabla en dos relaciones que eliminan la dependencia indebida (ver **Tabla 3.13;** ver **Tabla 3.14**):

Tabla 3.13. profesor_aula.

profesor	aula_id
Prof. Pérez	A1
Prof. Torres	A2

Nota. Fuente: Elaboración propia.

Tabla 3.14. Tabla curso_profesor.

curso_id	profesor
MAT101	Luis Pérez
BIO102	Alberto Torres

Nota. Fuente: Elaboración propia.

En este nuevo diseño:

- Todas las dependencias funcionales tienen como determinante una clave.
- Se evita la redundancia y se mejora la consistencia.

Los beneficios de cumplir 3FN son los siguientes:

- Elimina anomalías residuales que no resuelve 3FN.
- Aumenta la robustez lógica del diseño.
- Mejora la consistencia en bases de datos con múltiples claves candidatas.

BCNF es especialmente relevante en sistemas complejos donde existen múltiples claves candidatas o relaciones de dependencia inusuales. Aunque su aplicación puede llevar a una mayor fragmentación de las tablas, ofrece una base sólida para garantizar la integridad lógica del modelo relacional.

3.6. Anomalías de diseño y cómo evitarlas

Según Elmasri & Shamkant (2007), las anomalías de diseño son inconsistencias que surgen cuando una base de datos no ha sido estructurada correctamente. Estas irregularidades afectan la calidad de los datos y la eficiencia de las operaciones comunes, como insertar, modificar o eliminar registros.

Existen tres tipos principales de anomalías:

- **Anomalía de inserción**: Se presenta cuando no es posible registrar cierta información porque depende de datos adicionales que aún no existen.
- **Anomalía de actualización**: Ocurre cuando un cambio en un dato obliga a realizar múltiples modificaciones en distintos lugares, lo que

puede generar incoherencias si alguna instancia no se actualiza correctamente.

- **Anomalía de eliminación**: Se manifiesta cuando al borrar un dato también se pierde información relevante que debería conservarse.

Estas situaciones suelen indicar un diseño deficiente y pueden evitarse mediante la normalización, que permite descomponer tablas complejas en estructuras más simples y coherentes (Díaz Rodríguez, 2015). Al asegurar que cada tabla represente una sola entidad o relación, y que las dependencias funcionales estén bien definidas, se mejora la integridad del sistema y se reducen los riesgos de errores operativos.

A continuación, se desarrolla un ejemplo ilustrativo de anomalías de diseño:

Supongamos una tabla no normalizada llamada curso_profesor, que almacena información sobre los cursos universitarios y los docentes asignados (ver **Tabla 3.15)**:

Tabla 3.15. Tabla curso_profesor.

curso_id	**curso_nombre**	**docente_nombre**	**docente_correo**
MAT101	Matemáticas I	María García	mgarcia@universidad.edu.ec
BIO102	Biología General	Andrés Torres	atorres@universidad.edu.ec
MAT101	Matemáticas I	María García	mgarcia@universidad.edu.ec

Nota. Fuente: Elaboración propia.

Con base en la tabla anterior, se plantean escenarios con los tipos principales de anomalías:

a. **Anomalía de inserción:** La institución contrata a un nuevo docente, pero aún no tiene asignado ningún curso. Debido al diseño actual de la tabla, no es posible registrar su información sin asociarla previamente a un curso, impidiendo almacenar anticipadamente datos relevantes como su nombre y correo institucional.
b. **Anomalía de actualización:** Si María García cambia su dirección de correo electrónico, dicho dato deberá actualizarse en todas las filas donde aparezca. Si una de ellas se olvida, el sistema conservará datos contradictorios sobre el mismo docente, lo que genera inconsistencias.
c. **Anomalía de eliminación:** Si por algún motivo se elimina el curso MAT101, también se perderá la información asociada a María García,

incluso si continúa vinculada a la institución. Esto ocurre porque el esquema actual une datos de cursos con datos personales que deberían mantenerse de forma independiente.

Estas situaciones reflejan la necesidad de aplicar procesos de normalización en el diseño de bases de datos. Separar los datos en tablas coherentes y relacionadas adecuadamente ayuda a reducir redundancias, evitar errores lógicos y facilitar la gestión de la información a largo plazo.

3.7. Restricciones de integridad en el modelo relacional

Las restricciones de integridad son fundamentales para preservar la calidad, coherencia y validez de los datos dentro de una base de datos relacional (Lorente Puchades et al., 2019). Estas reglas definen los límites lógicos que deben respetarse durante el almacenamiento, actualización o eliminación de información, y forman parte esencial del diseño estructural del sistema.

Entre los tipos principales de restricciones se encuentran:

a. **Integridad de entidad:** Esta restricción asegura que cada fila en una tabla pueda identificarse de manera única. Para ello, cada tabla debe contar con una clave primaria (PRIMARY KEY) que no puede contener valores nulos. Este requisito garantiza la unicidad e individualización de cada registro (Gutiérrez, 2018).

b. **Integridad referencial:** Se encarga de mantener la coherencia entre relaciones mediante claves foráneas (FOREIGN KEY). Una clave foránea en una tabla debe coincidir con un valor existente en la clave primaria de otra tabla. Esto evita referencias huérfanas y asegura que las relaciones entre registros estén correctamente establecidas (Piñeiro Gómez, 2013).

c. **Restricciones de dominio:** Establecen qué tipo de valores son válidos para un atributo determinado. Por ejemplo, una fecha de nacimiento no puede ser posterior a la fecha actual. Estas restricciones pueden implementarse con cláusulas como CHECK, NOT NULL, o a través del tipo de dato asignado al atributo (Mora, 2014).

Por ejemplo, en el contexto de una tabla conceptual llamada estudiante, se pueden establecer las siguientes restricciones de integridad:

- **Clave primaria (est_id):** Garantiza que cada estudiante pueda ser identificado de forma única dentro del sistema.
- **Restricción de obligatoriedad (est_nombre):** Asegura que el campo nombre no quede vacío, ya que es esencial para identificar al estudiante.

- **Restricción de unicidad (est_correo):** Impide que se repitan direcciones de correo electrónico, evitando ambigüedades entre estudiantes.
- **Restricción de dominio (est_edad ≥ 16):** Establece una condición mínima válida para la edad, de acuerdo con una política institucional.

Según Osorio Rivera (2008), estas restricciones se definen como parte del diseño lógico del esquema relacional, y su implementación exacta se realizará posteriormente durante el diseño físico, utilizando las herramientas que ofrece el DBMS elegido

3.8. Caso práctico: aplicación del diseño lógico relacional

A partir del modelo E-R del sistema de biblioteca (Unidad 2: Ver **Figura 2.22**), se desea construir el diseño lógico relacional, partiendo del modelo relacional que se muestra en la **Figura 3.14**, en el que aparecen las siguientes tablas: obra_bibliografica, autor, autor_obra (relación M:N) y editorial.

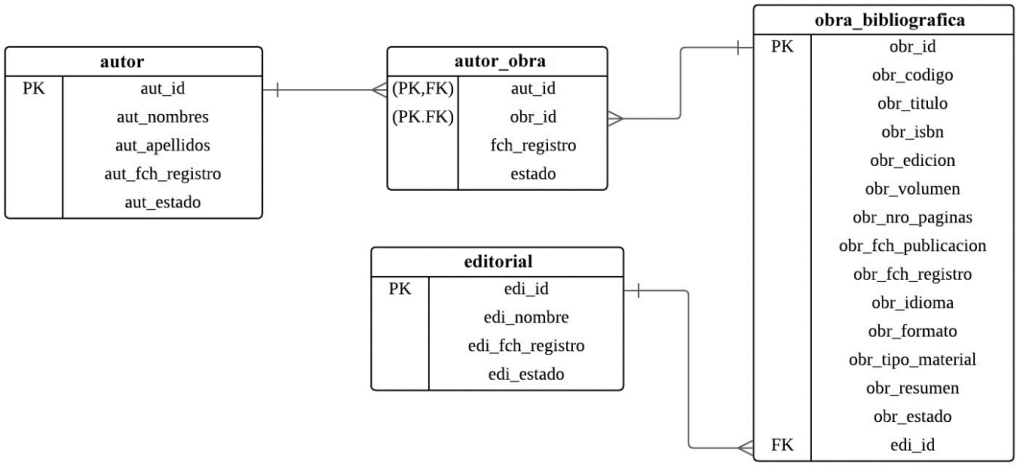


Figura 3.14. Modelo Relacional biblioteca.

Fuente: Elaboración propia.

Con base en el diagrama MER completo realizado por usted en la Unidad 1, y tomando como referencia el modelo relacional propuesto, realice la transformación completa al modelo lógico siguiendo estos pasos:

1. **Reconocimiento de tablas**
    - Identifiquen las tablas que provienen de las entidades del MER (autor, obra_bibliografica, editorial, entre otras)
    - Identifiquen las tablas de unión que modela la relación M:N.

2. **Listado de atributos con dominios**

   Para cada tabla, elabore una lista de sus columnas y asígnenles un dominio conceptual (Identificador, Texto corto, Texto largo, Fecha, Número, entre otros).

   - Ejemplo:
     - aut_id → Identificador
     - aut_nombres → Texto corto
     - obr_resumen → Texto largo

3. **Definición de claves**
   - Marquen la clave primaria (PK) de cada tabla.
   - En las tablas generadas por la relación M:N, indique la clave compuesta.

4. **Claves foráneas y cardinalidades**
   - Señalen las claves foráneas (FK) que unen las entidades o tablas.
   - Precisen la cardinalidad de cada relación (1;1; 1:M, M:N).

5. **Restricciones de integridad**
   - Identifiquen qué atributos deben ser NOT NULL (obligatorios).
   - Señalen qué atributos requieren unicidad (por ejemplo, obr_codigo, obr_isbn).
   - Definan posibles listas de valores válidos (Enum) para campos de estado (por ejemplo: aut_estado, edi_estado, obr_estado, entre otros).

6. **Entregables**
   - Un diccionario de datos en formato tabla con columnas: Tabla, Atributo, Dominio, PK, FK, Restricciones.
   - Un diagrama relacional (crows-foot) que refleje tablas, PK, FK y cardinalidades.

Resumen de la unidad

La Unidad 3 ha proporcionado las herramientas conceptuales y técnicas necesarias para llevar a cabo el diseño lógico relacional de una base de datos a partir de un MER. Se exploraron en profundidad los componentes esenciales del modelo relacional, tales como relaciones (tablas), atributos (columnas), tuplas (filas), dominios, claves primarias y claves foráneas, comprendiendo su función estructural y su aplicación en sistemas de bases de datos reales.

Se desarrolló una metodología clara para la conversión de MER a modelos relacionales, abordando casos particulares como entidades fuertes y débiles, atributos compuestos y multivaluados, así como relaciones 1:1, 1:M y M:N, con énfasis en la preservación de la semántica y las restricciones del modelo conceptual. Asimismo, se estudiaron las dependencias funcionales como base lógica para la aplicación del proceso de normalización, alcanzando la Tercera Forma Normal (3FN) y la Forma Normal de Boyce-Codd (BCNF). A través de ejemplos prácticos, se identificaron y corrigieron anomalías de diseño, optimizando así la estructura lógica y mejorando la integridad del sistema.

Se abordaron también las restricciones de integridad: integridad de entidad, referencial y de dominio, estableciendo reglas que garantizan la validez de los datos y evitan inconsistencias. La unidad culminó con un caso práctico aplicado a un sistema bibliotecario, integrando todos los conceptos mediante el diseño lógico de un modelo relacional completo.

En síntesis, esta unidad consolida el aprendizaje lógico del estudiante, preparándolo para la implementación física del sistema en un RDBMS, con esquemas normalizados, eficientes y alineados con los objetivos informacionales de la organización

Glosario de términos clave

Relación: Representación tabular de una entidad o asociación.

Tupla: Fila en una tabla; representa una instancia de la entidad.

Atributo: Columna en una tabla; representa una propiedad.

Clave primaria: Atributo que identifica de manera única una tupla.

Clave foránea: Atributo que referencia una clave primaria de otra tabla.

Dependencia funcional: Relación entre atributos donde uno determina al otro.

Normalización: Proceso para eliminar redundancia y mejorar la estructura de datos.

Forma normal: Regla que define condiciones estructurales para evitar anomalías.

Anomalía: Inconsistencia producida por un mal diseño relacional.

Integridad referencial: Regla que asegura la validez entre claves primarias y foráneas.

Preguntas de autoaprendizaje

Instrucciones: Marca la opción correcta en cada caso.

1. ¿Qué representa una relación en el modelo relacional?

a. Una estructura jerárquica.

b. Una tabla de datos con filas y columnas.

c. Una clave compuesta.

d. Un procedimiento almacenado.

2. ¿Cuál de las siguientes condiciones define la 2FN?

a. Todos los valores son atómicos.

b. Toda dependencia funcional tiene una superclave.

c. No existen dependencias parciales de la clave compuesta.

d. No existen claves alternativas.

3. ¿Cuál de estas afirmaciones corresponde a una anomalía de eliminación?

a. No se puede insertar un dato sin otro.

b. Se pierde información al borrar un dato relacionado.

c. Se actualiza un valor de forma automática.

d. Se genera una clave primaria inválida.

4. ¿Cuál es una diferencia entre 3FN y BCNF?

a. 3FN permite relaciones multivaluadas.

b. BCNF requiere que toda dependencia tenga como determinante una superclave.

c. 3FN elimina claves compuestas.

d. BCNF admite dependencias transitivas.

5. ¿Qué condición garantiza la integridad referencial?

a. Que todos los datos sean únicos.

b. Que toda clave foránea coincida con una clave primaria existente.

c. Que los datos no se repitan.

d. Que las claves estén compuestas.

Respuestas: 1(b); 2(c); 3(b); 4(b); 5(b).

Ejercicios prácticos para resolver

Ejercicio 1: Identificación de anomalías

Una institución registra la participación de los docentes en los distintos cursos impartidos en cada semestre. Por simplicidad, toda la información se almacena actualmente en una única tabla docente_curso **(**ver **Tabla 3.16),** la cual constan los siguientes atributos:

- **doc_id:** Identificador único del docente.
- **doc_nombre:** Nombre completo del docente.
- **cur_id:** Identificador único del curso.
- **cur_nombre:** Denominación del curso.
- **cur_horas:** Número de horas lectivas o carga horaria asignada al curso.
- **semestre:** Período académico en que el docente imparte un curso.

Tabla 3.16. Tabla docente_curso.

doc_id	doc_nombre	cur_id	cur_nombre	cur_horas	semestre
D001	María Fernández	C101	Matemáticas I	60	2025-I
D001	María Fernández	C102	Física II	45	2025-I
D002	Pedro Álvarez	C101	Matemáticas I	60	2025-I
D003	Lucía Gómez	C103	Programación Básica	75	2025-I

Nota. Fuente: Elaboración propia.

Realiza las siguientes actividades:

a) Análisis Preliminar

1. Clave primaria actual

- ¿Qué combinación de atributos identifica de manera única cada registro?

2. Dependencias funcionales iniciales

- Anota las dependencias funcionales que se desprenden del escenario, por ejemplo:
  - doc_id → doc_nombre
  - cur_id → cur_nombre, cur_horas
  - (doc_id, cur_id, semestre) →?

3. Redundancias y anomalías

- Señala los valores repetidos (por ejemplo, María Fernández aparece en dos filas) y comenta las consecuencias en inserciones, actualizaciones y eliminaciones futuras.

b) Verificación de la 1FN

1. Atomicidad de los atributos

- ¿Contiene algún campo valores compuestos o listados?
- Grupos repetitivos
  - ¿Existen atributos que natural o lógicamente puedan tener varias entradas en la misma fila?

2. Atomicidad de los atributos

- ¿Contiene algún campo valores compuestos o listados?

2. Grupos repetitivos

- ¿Existen atributos que natural o lógicamente puedan tener varias entradas en la misma fila?

c) Aplicación de la 2FN

1. Clave primaria compuesta

- ¿La clave que definiste en la actividad a, es compuesta? ¿Qué partes la forman?

2. Dependencias parciales

- Identifica atributos que dependen solo de una parte de esa clave compuesta.

3. Nueva partición de tablas

- Propón las tablas necesarias para eliminar las dependencias parciales.

4. Esquema relacional resultante

- Escribe el nuevo esquema con claves primarias y foráneas.

d) Aplicación de la 3FN

1. Dependencias transitivas

- ¿Algún atributo no clave depende de otro atributo no clave en la tabla de 2FN?

2. Resolución

- Si las hay, define qué nuevas tablas o ajustes eliminan esas dependencias.

3. Modelo final en 3FN

- Redacta el conjunto completo de tablas finales, indicando PK y FK.

e) Validación y Documentación

1. Anomalías resueltas

- Explica qué anomalías de inserción, actualización o eliminación quedan corregidas con tu diseño.

2. Beneficios del modelo normalizado

- Describe brevemente las mejoras en mantenimiento, consistencia y extensibilidad.

Ejercicio 2: Conversión del MER a Modelo Relacional

Con base en el siguiente modelo conceptual (ver **Figura 3.15**), aplica correctamente las reglas de transformación de MER al modelo relacional, identificando entidades, relaciones, claves primarias y claves foráneas.

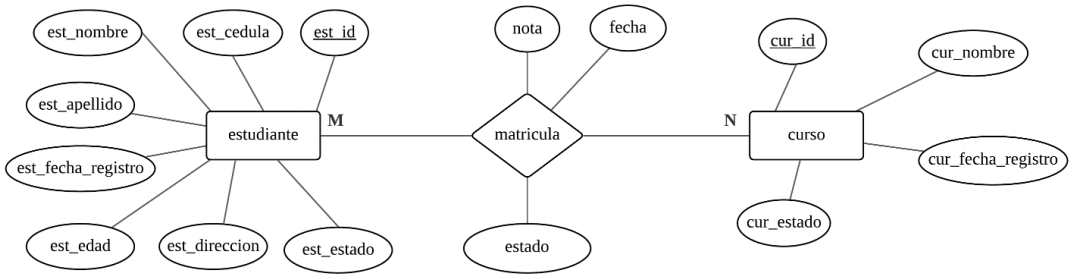


Figura 3.15. MER del sistema de matrícula.

Fuente: Elaboración propia.

Realiza las siguientes actividades:

1. Identifica las entidades presentes en el MER.

2. Enumera los atributos de cada entidad y subraya cuál es la clave primaria en cada una.

3. Analiza la relación matrícula: ¿qué tipo de relación es y qué atributos posee?

4. Transforma cada entidad (estudiante, curso) en una tabla relacional con sus respectivos atributos.

5. Crea la tabla correspondiente a la relación matrícula, utilizando una clave primaria compuesta.

6. Define las claves foráneas necesarias en la tabla matrícula para mantener la integridad referencial.

7. Redacta el esquema relacional completo con claves primarias y foráneas.

8. Representa gráficamente el diagrama del modelo relacional.

Ejercicio 4: Revisa cuidadosamente las dependencias funcionales y los posibles problemas de redundancia en este esquema (ver **Figura 3.16)**. En

particular, observa cómo lib_editorial se repite para cada libro y se presta a convertirse en su propia entidad.

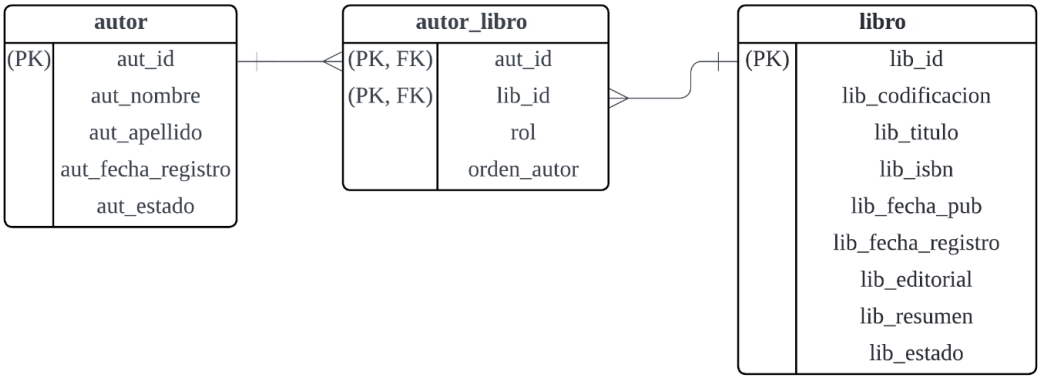


Figura 3.16. Modelo relacional autor-autor_libro-libro.

Fuente: Elaboración propia.

Realiza las siguientes actividades

1. Identifica las DF

- Anota todas las dependencias funcionales que existan en cada tabla. *Ejemplo:* lib_id → lib_titulo, lib_id → lib_editorial.
- Detecta si existen dependencias transitivas, como: *Ejemplo:* lib_id → lib_editorial y lib_editorial → edi_nombre.

2. Determina las claves candidatas

- Verifica que aut_id sea la clave única en la tabla de autores, lib_id en la de libros, y que la combinación (aut_id, lib_id) sea la clave candidata en autor_libro.
- Comprueba que no existan otras combinaciones posibles como claves candidatas en cada relación.

3. Aplica las formas normales (1FN, 2FN y 3FN)

- Asegúrate de que cada relación esté en 1FN: sin valores multivaluados ni repetitivos.
- Elimina dependencias parciales para alcanzar la 2FN.
- Elimina dependencias transitivas para alcanzar la 3FN.

4. Extrae la entidad editorial

- Crea una nueva entidad editorial con un identificador único (edi_id) y atributos descriptivos relevantes, como: edi_nombre, edi_pais, edi_estado.
- Sustituye el atributo lib_editorial en la tabla libro por una clave foránea edi_id.

5. Redefine el modelo relacional y el MER actualizado

- Escribe el nuevo conjunto de tablas y sus atributos, indicando claramente claves primarias y foráneas.
- Verifica que todas las relaciones estén al menos en 3FN y sin dependencias indebidas.
- Elabora el diagrama MER actualizado, incorporando explícitamente la entidad editorial y su relación con libro.
- Elabora el diagrama del modelo relacional final, incluyendo claves primarias, foráneas y relaciones entre las tablas.

6. Argumenta tu diseño

- Explica brevemente por qué este nuevo diseño es más eficiente: cómo elimina redundancias y previene anomalías de inserción, actualización y eliminación.
- Justifica la separación de la entidad editorial como entidad propia, y su utilidad para escalabilidad, mantenimiento y control de integridad referencial.

Referencias bibliográficas de la Unidad 3

Albarak, M., & Bahsoon, R. (2018). *Prioritizing Technical Debt in Database Normalization Using Portfolio Theory and Data Quality Metrics*. https://arxiv.org/abs/1801.06989.

Alma Muñoz. (2019). *¿Por qué es tan importante la normalización de base de datos?* GESTIÓN DE CATAGOLO.

Codd, E. F. (1990). *The Relational Model for Database Management: Version 2*. Addison-Wesley. https://doi.org/10.5555/77708

Date, C. J. (2003). *Introduction to Database Systems* (8th ed.). Addison-Wesley.

Demba, M. (2013). Algorithm for Relational Database Normalization Up to 3NF. *International Journal of Database Management Systems*, *5*(3). https://doi.org/10.5121/ijdms.2013.5303

Díaz Rodríguez, O. (2015). Metodología para Diseñar Bases de Datos Relacionales con Base en el Análisis de Escenarios, sus Políticas y las Reglas del Negocio. *3C TIC. Cuadernos de Desarrollo Aplicados a Las TIC*, *4*(3). https://doi.org/10.17993/3ctic.2015.43.197-209

Efendy, Z. (2018). NORMALIZATION IN DATABASE DESIGN. *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer Dan Teknologi Informasi*, *4*(1). https://doi.org/10.24014/coreit.v4i1.4382

Elmasri, R., & Shamkant, N. (2007). *Fundamentos de Sistemas de Bases de Datos* (5th ed.). Pearson Educación.

Estrada Rios, E. (2020). *Sistema de Detección y Reconocimiento Patrones en Base de Datos de Adicciones Utilizando la Inteligencia Artificial*.

Gardarin, Georges. (1990). *Bases de datos : gestión de ficheros, el modelo relacional, algoritmos y lenguajes, seguridad de los datos* (2nd ed.). Paraninfo.

Giménez, J. A. (2019). Buenas prácticas en el diseño de bases de datos Base de Datos. *Revista Científica Internacional ARANDU UTIC*, *VI*.

Gutiérrez, P. (2018). Fundamentos de las bases de datos. *Genbeta*.

Köhler, H., & Link, S. (2018). SQL schema design: foundations, normal forms, and normalization. *Information Systems*, *76*. https://doi.org/10.1016/j.is.2018.04.001

Lorente Puchades, I., Díaz Llobet, M., Sistac i Planas, J., & Rodríguez González, E. (2019). *Diseño lógico de bases de datos*.

Transformación a relacional a partir del modelo ER (1st ed.). Universitat Oberta de Catalunya.

Miguel Castaño, A., Piattini Velthuis, M. G., & Marcos Martínez, E. (2000). *Diseño de bases de datos relacionales*. Alfaomega, Rama.

Mora, A. (2014). Bases de datos. Diseño y gestión. In *Bases de datos. Diseño y gestión*.

Nevado Cabello, M. V. (2010). *Introducción a las Bases de Datos Relacioneles* (1st ed.). Visión Libros.

Osorio Rivera, F. L. (2008). *BASE DE DATOS RELACIONALES. TEORÍA Y PRÁCTICA* (Issue 1). Fondo Editorial ITM.

Piñeiro Gómez, J. (2013). *Bases de datos relacionales y modelado de datos*. Ediciones Paraninfo, S.A.

Santiso, J. M. J., & Nichita, P. (2018). Diseño de bases de datos relacionales con DBDStudio. *Actas de Las Jornadas Sobre La Enseñanza Universitaria de La Informática (JENUI)*, *3*, 56.

Unidad 4: Diseño Físico de la Base de Datos
Introducción de la Unidad

El diseño físico de bases de datos representa la fase culminante en el proceso de desarrollo de un sistema de información relacional. En esta etapa, el modelo lógico previamente construido con sus tablas, claves, relaciones y restricciones se transforma en instrucciones SQL concretas que permiten su implementación sobre un Sistema de Gestión de Bases de Datos (DBMS). Este proceso no solo implica definir la estructura final de las tablas, sino también especificar aspectos críticos como los tipos de datos, la selección y colocación de índices, las estrategias de almacenamiento, la gestión de particionamiento, y la planificación de mecanismos para garantizar la alta disponibilidad.

A diferencia del modelado conceptual y lógico que priorizan la fidelidad semántica del dominio del problema, el diseño físico se orienta a maximizar el rendimiento del sistema bajo condiciones reales de operación. Esto requiere tomar decisiones técnicas que afectan directamente el tiempo de respuesta, la eficiencia en operaciones CRUD, el uso de memoria y disco, y la capacidad de escalado del sistema.

En esta unidad, el estudiante desarrollará competencias clave para llevar a cabo un diseño físico sólido y optimizado. A lo largo del capítulo, se estudiará la creación de tablas y definición de tipos de datos, la implementación de claves primarias y foráneas, así como las restricciones de integridad y valores por defecto. Posteriormente, se abordarán los índices y técnicas de optimización de consultas, el uso de vistas, funciones y triggers, y las estrategias de carga de datos mediante scripts y herramientas de migración. Finalmente, se profundizará en los mecanismos de replicación y alta disponibilidad propios de DBMS como PostgreSQL, y se presentará un caso práctico integrador que permitirá aplicar todos los conocimientos adquiridos en un escenario de implementación real.

Con ello, esta unidad ofrece no solo el dominio técnico para implementar soluciones eficientes, sino también el juicio crítico para tomar decisiones informadas sobre cómo garantizar la escalabilidad, integridad y sostenibilidad de la base de datos a largo plazo.

Objetivos de aprendizaje

Al culminar esta unidad, el estudiante será capaz de:

1. Comprender el rol estratégico del diseño físico dentro del ciclo de vida de una base de datos, diferenciándolo claramente de las etapas conceptuales y lógicas, y valorando su impacto en el rendimiento y sostenibilidad del sistema.

2. Identificar y describir las principales estructuras físicas de almacenamiento gestionadas por un DBMS, analizando cómo influyen en la eficiencia operativa y en el uso óptimo de los recursos.
3. Aplicar técnicas de indexación orientadas a optimizar el acceso a los datos, mejorando los tiempos de respuesta en consultas y operaciones de inserción, actualización y eliminación.
4. Diseñar esquemas físicos robustos y eficientes, teniendo en cuenta variables críticas como el volumen de datos, la frecuencia de transacciones, los patrones de uso y las limitaciones de hardware y software.
5. Implementar mecanismos esenciales de seguridad, respaldo y recuperación, garantizando la disponibilidad de la información y la protección frente a fallos, accesos no autorizados o contingencias operativas.

Preguntas de Enfoque

- ¿En qué se distingue el diseño físico del diseño lógico en términos de objetivos, abstracción y ejecución dentro de un proyecto de base de datos?
- ¿Qué tipos de índices existen en los DBMS modernos y cómo afectan directamente el rendimiento del sistema ante cargas de trabajo diversas?
- ¿Cuáles son las mejores prácticas y estrategias que permiten asegurar un acceso eficiente, escalable y seguro a los datos en entornos reales?

Desarrollo de contenidos

4.1. Definición y propósito del Diseño Físico

De acuerdo con el criterio de Cornelio et al. (2004), el diseño físico representa la etapa final del ciclo de desarrollo de una base de datos, donde el modelo lógico previamente construido se materializa sobre una plataforma tecnológica específica. Esta fase requiere tener en cuenta aspectos del entorno operativo real, como el hardware disponible, el sistema operativo y, sobre todo, las características internas del DBMS que se utilizará.

El propósito del diseño físico es definir con precisión cómo serán almacenados, organizados, indexados y accedidos los datos en el nivel físico, es decir, sobre los dispositivos de almacenamiento (disco). Este enfoque busca optimizar el rendimiento del sistema, garantizando que las operaciones de lectura y escritura se ejecuten con eficiencia, minimizando el uso de recursos y reduciendo la latencia en entornos de alta demanda (Garcia-Molina et al., 2009).

Durante esta fase, el diseñador debe tomar decisiones técnicas clave que influirán directamente en la performance del sistema. Estas incluyen la selección de estructuras de almacenamiento adecuadas (por ejemplo, tipo heap, hash o clustered), la definición y colocación de índices, la ubicación física de las tablas en distintos tablespaces, así como la configuración fina de parámetros del motor del DBMS para una gestión eficaz de la memoria, cachés, buffers y concurrencia. En la **Tabla 4.1**, se describen las diferencias entre el Diseño Lógico y el Diseño Físico

Tabla 4.1. Diferencias entre el Diseño Lógico y el Diseño Físico.

Aspecto	**Diseño Lógico**	**Diseño Físico**
Enfoque	Conceptual y abstracto	Técnico y específico
Propósito	Garantizar la coherencia, integridad y estructura de los datos	Optimizar el rendimiento operativo y el uso de recursos
Independencia	No depende del sistema físico de implementación	Depende del hardware, SO y características del SGBD seleccionado
Nivel de detalle	Describe entidades, atributos, relaciones, claves primarias y foráneas	Define estructuras de almacenamiento, índices, disposición de datos en disco y parámetros del motor

Nota. Fuente: Elaboración propia.

Según Elmasri & Navathe (2016), el diseño físico determina cómo se almacenan realmente los datos en disco y cómo se accede a ellos de manera eficiente**,** con énfasis en aspectos operativos como el rendimiento, la escalabilidad y la capacidad de respuesta del sistema bajo carga.

4.2. Creación de tablas y tipos de datos

Según Jiménez & Armstrong (2018), una vez definido el diseño lógico de una base de datos el siguiente paso consiste en implementar físicamente dicho diseño utilizando instrucciones del Lenguaje Estructurado de Consultas (SQL). La instrucción CREATE TABLE permite crear estructuras tabulares que materializan las entidades y relaciones previamente modeladas, especificando sus atributos, tipos de datos y restricciones asociadas. A continuación, se describe la sintaxis general de CREATE TABLE para la creación de una tabla:

CREATE TABLE nombre_tabla (nombre_atributo tipo_dato restricciones,...

);

De acuerdo con el criterio de Schönig (2015), cada atributo debe declararse junto a un tipo de dato compatible, el cual dependerá del tipo de información que almacenará y del motor de base de datos utilizado. Entre los tipos de datos más comunes en PostgreSQL y otros SGBD se encuentran:

- VARCHAR(n): para cadenas de texto de longitud variable (hasta *n* caracteres).
- CHAR(n): para cadenas de longitud fija.
- DATE: para fechas del calendario.
- DECIMAL (p, s): para valores numéricos con precisión (*p*) y escala (*s*), común en importes y medidas exactas.
- INTEGER, BIGINT, SERIAL: para valores numéricos enteros.

Definición de dominios con restricciones

Más allá de los tipos de datos, es posible imponer restricciones adicionales sobre los valores permitidos en una columna mediante el uso de:

- CHECK: permite validar condiciones lógicas al momento de insertar o actualizar registros.
- ENUM: en PostgreSQL, define un conjunto cerrado de valores válidos, útil para atributos categóricos como estados, niveles o clasificaciones.

Ejemplo de definición con restricción CHECK:

```
CREATE TABLE empleado (
  emp_id SERIAL PRIMARY KEY,
  emp_nombre VARCHAR(100) NOT NULL,
  emp_salario DECIMAL(10,2) CHECK (emp_salario > 0)
);
```

Ejemplo con tipo ENUM en PostgreSQL:

```
CREATE TYPE estado_empleado AS ENUM ('activo', 'inactivo');
CREATE TABLE empleado (
  emp_id SERIAL PRIMARY KEY,
  emp_estado estado_empleado NOT NULL
);
```

Ejemplo práctico: empleado y carga_familiar

Consideremos un caso donde se necesita registrar a los empleados de una empresa y a sus respectivas cargas familiares. La entidad empleado tiene atributos como identificador, nombre, y fecha de ingreso. Por su parte, carga_familiar es una entidad débil, ya que no puede identificarse sin hacer referencia al empleado del cual depende. A continuación, se describe las sentencias requeridas:

1. **Tabla empleado**

```
CREATE TABLE empleado (
    emp_id SERIAL PRIMARY KEY,
    emp_nombre VARCHAR(100) NOT NULL,
    emp_fecha_ingreso DATE NOT NULL,
    emp_estado estado_empleado DEFAULT 'activo'
);
```

2. **Tabla carga_familiar**

```
CREATE TABLE carga_familiar (
    emp_id INT,
    car_nombre VARCHAR(100) NOT NULL,
    car_fecha_nacimiento DATE,
    PRIMARY KEY (emp_id, car_nombre),
    FOREIGN KEY (emp_id) REFERENCES empleado(emp_id)
);
```

En este ejemplo, la tabla carga_familiar utiliza una clave primaria compuesta (emp_id, car_nombre) para garantizar que cada carga familiar se identifique en el contexto del empleado correspondiente. La relación entre ambas tablas queda definida por la clave foránea emp_id, que garantiza la integridad referencial del diseño.

Este tipo de implementación refleja con precisión las decisiones tomadas durante el modelado lógico, y establece las bases para consultas seguras, eficientes y coherentes dentro del sistema de información.

4.3. Claves primarias y foráneas

Según Silva (2020), en el diseño físico de bases de datos, la correcta implementación de claves primarias y claves foráneas es crucial para asegurar la integridad de los datos y mantener la coherencia entre las distintas tablas.

Estas claves no solo permiten identificar registros de forma única, sino también establecer relaciones consistentes entre entidades, reproduciendo la lógica del modelo conceptual en una estructura física funcional y eficiente. A continuación, se describe la funcionalidad de clave primaria y clave foránea:

3. **Clave primaria**

   La clave primaria (PRIMARY KEY) es una restricción que garantiza que cada fila de una tabla pueda identificarse de forma única. Esta clave:

   - No puede contener valores nulos.
   - No permite duplicados.
   - Puede estar compuesta por uno o más atributos (clave compuesta).

   Ejemplo simple:

```
CREATE TABLE autor (
  aut_id SERIAL PRIMARY KEY,
  aut_nombre VARCHAR(100) NOT NULL
);
```

4. **Clave foránea:**

   La clave foránea (FOREIGN KEY) se utiliza para establecer una relación con otra tabla, refiriéndose a su clave primaria. Esta restricción permite mantener la integridad referencial, asegurando que los valores ingresados en la columna foránea existan previamente en la tabla referenciada.

   Ejemplo con clave foránea:

```
CREATE TABLE autor_libro (
  aut_id INT,
  lib_id INT,
  PRIMARY KEY (aut_id, lib_id),
  FOREIGN KEY (aut_id) REFERENCES autor(aut_id),
  FOREIGN KEY (lib_id) REFERENCES libro(lib_id)
);
```

   En este ejemplo, la tabla autor_libro actúa como tabla intermedia en una relación de muchos a muchos entre autor y libro, siendo sus claves foráneas aut_id y lib_id.

Comportamiento de integridad referencial

Al declarar una clave foránea, es posible definir el comportamiento del sistema ante operaciones de eliminación (DELETE) o actualización (UPDATE) de registros referenciados. Para ello, se emplean las cláusulas ON DELETE y ON UPDATE:

- CASCADE: Propaga el cambio (elimina o actualiza también el registro dependiente).
- SET NULL: Asigna un valor nulo en el registro dependiente.
- RESTRICT o NO ACTION: Impide la operación si hay registros dependientes.

Ejemplo avanzado:

```
CREATE TABLE autor_libro (
  aut_id INT,
  lib_id INT,
  PRIMARY KEY (aut_id, lib_id),
  FOREIGN KEY (aut_id) REFERENCES autor(aut_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (lib_id) REFERENCES libro(lib_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

En este caso:

- Si se elimina un autor, también se eliminarán automáticamente sus registros en autor_libro.
- Si se elimina un libro, el sistema no lo permitirá si hay registros en autor_libro que lo referencien.
- Cualquier cambio en los identificadores primarios de autor o libro se propaga automáticamente a la tabla intermedia.

Ejemplo práctico: Relación autor ↔ autor_libro

Consideremos una base de datos para una biblioteca o editorial donde un autor puede escribir uno o varios libros, y a su vez un libro puede ser escrito por más de un autor. Esta relación de muchos a muchos (M:N) se implementa mediante una tabla intermedia autor_libro.

A continuación, se estructuran las SQL requeridas para la creación de las tablas:

5. **Tabla autor**

```
CREATE TABLE autor (
  aut_id SERIAL PRIMARY KEY,
  aut_nombre VARCHAR(100) NOT NULL
);
```

6. **Tabla libro**

```
CREATE TABLE libro (
  lib_id SERIAL PRIMARY KEY,
  lib_titulo VARCHAR(150) NOT NULL
);
```

7. **Tabla intermedia autor_libro**

```
CREATE TABLE autor_libro (
  aut_id INT,
  lib_id INT,
  PRIMARY KEY (aut_id, lib_id),
  FOREIGN KEY (aut_id) REFERENCES autor(aut_id) ON DELETE CASCADE,
  FOREIGN KEY (lib_id) REFERENCES libro(lib_id) ON DELETE CASCADE
);
```

Esta estructura garantiza:

- La unicidad de la combinación autor–libro.
- La integridad referencial entre entidades.
- La eliminación automática de relaciones cuando un autor o libro es eliminado del sistema.

4.4. Restricciones de integridad y valores por defecto

Las restricciones de integridad son elementos fundamentales del diseño físico de una base de datos. Su propósito es garantizar la validez, coherencia y consistencia de los datos almacenados, evitando errores o inconsistencias desde el momento en que se insertan o modifican los registros. Estas restricciones se implementan directamente en el esquema de la base de datos a través de instrucciones SQL, y permiten establecer reglas que deben cumplirse de forma automática en cada operación sobre la tabla (Chávez, 2019).

A continuación, se describen y ejemplifican las siguientes restricciones:

1. **NOT NULL: Obligatoriedad de un valor**

   La restricción NOT NULL impide que un campo quede vacío. Es útil para aquellos atributos que son esenciales para la identidad o descripción de una entidad.

   Ejemplo:

   emp_nombre VARCHAR(100) NOT NULL

   En este caso, todo empleado debe tener obligatoriamente un nombre.

2. **UNIQUE: Valores no repetidos**

   UNIQUE asegura que los valores de un atributo o combinación de atributos no se repitan entre filas. A diferencia de la clave primaria (PRIMARY KEY), permite valores nulos (excepto en algunos DBMS).

   Ejemplo:

   emp_correo VARCHAR(150) UNIQUE

   Esto evita que dos empleados tengan el mismo correo electrónico, lo cual es fundamental para autenticación o comunicación.

3. **DEFAULT: Valor por omisión**

   La cláusula DEFAULT define un valor predeterminado para un campo, que se utilizará automáticamente si el usuario no lo especifica al insertar un registro.

   Ejemplo:

   emp_estado VARCHAR(20) DEFAULT 'activo'

   Si no se especifica el estado del empleado al insertar el dato, se asumirá automáticamente que está “activo”.

4. **CHECK: Validaciones de dominio**

   La restricción CHECK permite definir reglas de validación sobre los valores que pueden contener ciertos campos, actuando como una forma explícita de definir el dominio de un atributo.

   Ejemplo:

   emp_estado VARCHAR(20) CHECK (emp_estado IN ('activo', 'inactivo'))

   Esta validación asegura que los únicos valores posibles para el estado del empleado sean "activo" o "inactivo".

Ejemplo práctico: Definición de tabla con restricciones

```
CREATE TABLE empleado (
   emp_id SERIAL PRIMARY KEY,
   emp_nombre VARCHAR(100) NOT NULL,
   emp_correo VARCHAR(150) UNIQUE,
   emp_estado VARCHAR(20) DEFAULT 'activo' CHECK (emp_estado IN ('activo', 'inactivo')),
   emp_fecha_ingreso DATE NOT NULL
);
```

En este diseño, se combinan distintas restricciones para garantizar que:

- No haya empleados sin nombre ni fecha de ingreso (NOT NULL).
- No se repitan correos electrónicos (UNIQUE).
- Se asigne un estado por defecto si no se proporciona (DEFAULT).
- El estado sea uno de los valores válidos predefinidos (CHECK).

Estas restricciones no solo refuerzan la calidad del diseño físico, sino que también constituyen una primera línea de defensa contra errores y aseguran que los datos ingresados cumplan con las reglas del negocio desde su origen.

4.5. Índices y optimización de consultas

En el contexto del diseño físico de bases de datos, los índices representan una estructura fundamental para mejorar el rendimiento de las consultas. Su función principal es acelerar la localización y recuperación de datos, evitando la exploración secuencial completa de una tabla (conocida como full table scan). Los índices actúan como estructuras auxiliares que organizan los valores de ciertos atributos de manera eficiente, permitiendo búsquedas rápidas, ordenamientos optimizados y acceso selectivo (López Herrera, 2016).

Creación de índices simples y compuestos

Un índice simple se crea sobre un único atributo de una tabla, mientras que un índice compuesto involucra dos o más columnas que son comúnmente utilizadas juntas en consultas (Salazar Cárdenas, 2014).

A continuación, se describe la sintaxis general:

```
CREATE INDEX nombre_indice
ON nombre_tabla (columna1 [, columna2, ...]);
```

Ejemplo de índice simple:

```
CREATE INDEX idx_empleado_nombre
ON empleado (emp_nombre);
```

Ejemplo de índice compuesto:

```
CREATE INDEX idx_factura_cliente_fecha
ON factura (cli_id, fac_fecha);
```

En este caso, el índice está optimizado para búsquedas que combinan el cliente y la fecha de la factura.

Importante: Un índice compuesto solo se aprovecha si las consultas usan los campos del índice en el mismo orden en que fueron definidos.

Índices parciales en PostgreSQL

PostgreSQL permite crear índices parciales, es decir, estructuras que solo se aplican sobre un subconjunto de las filas que cumplen una condición determinada. Esta estrategia resulta útil cuando las consultas frecuentes se realizan sobre valores filtrados.

Ejemplo:

```
CREATE INDEX idx_emp_activos
ON empleado (emp_nombre)
WHERE emp_estado = 'activo';
```

Este índice solo incluirá empleados activos, reduciendo el espacio en disco y mejorando el rendimiento de las consultas específicas sobre este subconjunto.

Índices basados en funciones

De acuerdo con Silva (2020), otra característica poderosa de PostgreSQL es la posibilidad de crear índices sobre expresiones o funciones aplicadas a columnas. Esto permite optimizar consultas que no usan directamente los valores originales, sino una transformación de ellos.

Ejemplo:

```
CREATE INDEX idx_empleado_upper_nombre
ON empleado (UPPER(emp_nombre));
```

Esto acelera las búsquedas que usan expresiones como WHERE UPPER(emp_nombre) = 'JUAN'.

Consideraciones prácticas

Aunque los índices mejoran significativamente el rendimiento de las consultas, también implican costos adicionales:

- **Espacio en disco:** Cada índice requiere almacenamiento adicional.
- **Tiempo de mantenimiento:** Cada vez que se inserta, actualiza o elimina un registro, los índices asociados también deben actualizarse.

Por ello, es fundamental evaluar cuidadosamente qué columnas indexar, considerando la frecuencia de las consultas, los criterios de filtrado más comunes y la necesidad de ordenamientos.

Nota destacada: Los índices constituyen un recurso esencial en el diseño físico para garantizar consultas eficientes. Su correcta implementación puede marcar una diferencia sustancial en el rendimiento global del sistema, especialmente en bases de datos con grandes volúmenes de información o con alta concurrencia de acceso.

4.6. Vistas y procedimientos almacenados

Según Ibáñez (2014), en el diseño físico de bases de datos, uno de los objetivos clave es facilitar la reutilización de consultas, simplificar el acceso a los datos y centralizar la lógica de negocio. Para ello, el uso de vistas, funciones y disparadores (triggers) constituye una estrategia avanzada y eficaz, alineada con los principios de modularidad, seguridad y mantenimiento del sistema. A continuación. se describen y ejemplifican cada uno de ellos:

a. Vistas: abstracción y simplificación de consultas

Una vista es una tabla virtual definida a partir de una o más tablas reales mediante una consulta SELECT. Las vistas no almacenan físicamente los datos, sino que presentan una representación lógica de ellos (Piñeiro Gómez, 2013; Yannakoudakis, 1988).

Su utilidad radica en:

- Simplificar consultas complejas reutilizadas frecuentemente.
- Proporcionar niveles de abstracción para diferentes perfiles de usuarios.

- Restringir el acceso a ciertas columnas o filas, como mecanismo de seguridad.

Sintaxis general:

```
CREATE VIEW vista_nombre AS
SELECT columnas
FROM tabla
WHERE condición;
```

Ejemplo práctico:

```
CREATE VIEW vista_clientes_activos AS
SELECT cli_id, cli_nombre, cli_correo
FROM cliente
WHERE cli_estado = 'activo';
```

Esta vista permite acceder únicamente a los clientes activos, ocultando la lógica de filtrado y facilitando el mantenimiento.

b. Funciones: encapsulación de lógica repetitiva

Una función almacenada es un bloque de código que puede recibir parámetros de entrada, ejecutar operaciones internas y devolver un resultado (Ramakrishnan & Gehrke, 2002). Las funciones se utilizan comúnmente para:

- Validar datos.
- Realizar cálculos personalizados.
- Automatizar procesos que deben repetirse.

Ejemplo (PostgreSQL):

```
CREATE FUNCTION calcular_iva(precio NUMERIC)
RETURNS NUMERIC AS $$
BEGIN
  RETURN precio * 0.12;
END;
$$ LANGUAGE plpgsql;
```

Esta función puede invocarse desde consultas o procedimientos para obtener automáticamente el valor del IVA.

c. Triggers: automatización basada en eventos

De acuerdo con el criterio de ITAI et al. (2013), un trigger o disparador es un tipo especial de procedimiento almacenado que se ejecuta automáticamente cuando ocurre un evento específico en una tabla. Por ejemplo: una inserción, actualización o eliminación.

Los disparadores se emplean para:

- Aplicar reglas de negocio automáticamente.
- Registrar auditorías o logs de cambios.
- Validar condiciones complejas antes o después de una operación.

Ejemplo:

```
CREATE TRIGGER log_actualizacion_empleado
AFTER UPDATE ON empleado
FOR EACH ROW
EXECUTE FUNCTION registrar_cambio();
```

Este trigger invoca la función registrar_cambio() cada vez que se actualiza un registro en la tabla empleado, lo cual es útil para tareas de seguimiento y trazabilidad.

Nota destacada: Las vistas, funciones y disparadores permiten trasladar parte de la lógica de negocio directamente al nivel del DBMS. Esta práctica mejora la cohesión del diseño, refuerza la seguridad y garantiza que las reglas se apliquen de forma uniforme, independientemente del cliente o aplicación que interactúe con la base de datos.

4.7. Carga de datos y scripts de población

Según Lelarge & Rouhaud (2023), una vez diseñadas y creadas las estructuras físicas de la base de datos, es indispensable alimentarlas con datos iniciales. Este proceso se conoce como población de datos o carga inicial, y constituye una fase esencial en entornos tanto de desarrollo como de producción. A través de comandos como INSERT, COPY y herramientas especializadas de migración, es posible automatizar la inserción masiva de registros, garantizando eficiencia, integridad y trazabilidad del proceso.

Comandos INSERT y COPY: estrategias para cargar datos

El comando INSERT permite agregar registros individualmente o por lotes pequeños, siendo ideal para scripts personalizados o para el ingreso manual de información (Tezuysal & Ahmed, 2024).

Ejemplo simple:

```
INSERT INTO producto (pro_id, pro_nombre, pro_precio)
VALUES ('P001', 'Teclado', 30.00);
```

No obstante, cuando se trata de insertar grandes volúmenes de datos (cientos o miles de registros), se recomienda utilizar el comando COPY, que permite importar directamente desde archivos planos (CSV, TXT, etc.) de manera eficiente y rápida.

Ejemplo con PostgreSQL:

```
COPY cliente (cli_id, cli_nombre, cli_correo)
FROM '/ruta/datos/clientes.csv'
DELIMITER ',' CSV HEADER;
```

Este comando importa datos desde un archivo CSV respetando las columnas definidas en la tabla, separadas por comas y con cabecera incluida.

Scripts de población

Un script de población es un archivo de comandos SQL que contiene una serie de instrucciones INSERT o COPY predefinidas, con el propósito de:

- Poner en funcionamiento un entorno de pruebas.
- Restaurar datos tras una migración.
- Automatizar la carga de catálogos o valores de referencia.

Estos scripts suelen formar parte de los entregables en proyectos profesionales, especialmente cuando se realiza la documentación técnica de despliegue.

Herramientas de migración de datos

Dombrovskaya et al. (2024), mencionan que además de los comandos SQL, existen herramientas especializadas que facilitan la migración de datos desde hojas de cálculo, otros sistemas de bases de datos o aplicaciones externas. Algunas de las más utilizadas son:

- **pgAdmin** (PostgreSQL): permite importar datos desde archivos CSV o Excel.
- **DBeaver**: herramienta multiplataforma con asistentes gráficos para carga masiva.
- **ETL Tools** (Extract, Transform, Load): como Talend, Pentaho o Apache NiFi, orientadas a procesos de integración complejos entre múltiples fuentes de datos.

Estas herramientas permiten aplicar transformaciones intermedias, validar datos, mapear campos y automatizar tareas recurrentes en entornos empresariales.

Nota destacada: Dominar las técnicas de carga de datos desde el uso básico de INSERT hasta procesos avanzados con COPY y herramientas ETL permite no solo agilizar el desarrollo y puesta en marcha de sistemas, sino también garantizar la calidad y consistencia del contenido de la base de datos desde sus primeras etapas de operación.

4.8. Replicación y estrategias de alta disponibilidad

En los entornos actuales, donde las bases de datos deben estar disponibles en todo momento y tolerar posibles fallos del sistema, resulta imprescindible implementar mecanismos que aseguren la continuidad operativa y la resiliencia ante fallos. Dos pilares fundamentales para lograr este objetivo son la replicación y las estrategias de alta disponibilidad.

Réplicas en streaming en PostgreSQL

Según Chávez (2019), PostgreSQL ofrece una solución robusta de replicación basada en el concepto de streaming replication, una técnica que permite copiar en tiempo real los cambios realizados en la base de datos principal (master o primary) hacia una o más instancias secundarias (standby). Esta arquitectura permite:

- Reducir el tiempo de inactividad en caso de fallos del servidor principal.
- Distribuir la carga de lectura, utilizando los nodos secundarios para consultas de solo lectura.
- Probar copias en tiempo real sin afectar la base de datos principal.

La replicación en streaming se basa en la transferencia continua de los archivos WAL (Write-Ahead Log) desde el nodo primario hacia los secundarios, permitiendo una sincronización casi instantánea. Este enfoque garantiza consistencia y baja latencia en la replicación, sin afectar significativamente el rendimiento del sistema.

Conmutación por error y planificación de respaldos

De acuerdo con el criterio de Schönig (2015), en el diseño de sistemas confiables, no basta con replicar datos; también se debe estar preparado para conmutaciones por error (failover). Esto implica la capacidad del sistema de redirigir automáticamente las operaciones al nodo secundario en caso de que el nodo primario deje de funcionar. Herramientas como Patroni, pg_auto_failover o soluciones integradas con Kubernetes permiten automatizar este proceso de recuperación.

Además, la planificación de respaldos (backups) constituye una práctica crítica para la preservación de los datos a largo plazo. PostgreSQL proporciona utilidades eficientes como:

- **pg_dump**: ideal para respaldos lógicos, permite exportar estructuras y datos en formatos personalizables.
- **pg_basebackup**: recomendable para respaldos físicos completos del clúster, compatible con entornos de replicación.

Un buen plan de alta disponibilidad debe incluir políticas periódicas de respaldo, pruebas de restauración, monitoreo del estado de los nodos y procedimientos definidos para recuperación ante desastres.

4.9. Caso práctico integrador: Sistema de Gestión Bibliográfica

En esta sección se propone al estudiante la implementación completa del Modelo Relacional desarrollado en la Unidad 3 y que tuvo como base el diagrama que contempla la gestión de obras bibliográficas, autores y editoriales (ver **Figura 3.14**). Este ejercicio busca consolidar los conocimientos adquiridos sobre diseño físico, incluyendo la creación de estructuras de datos, definición de restricciones de integridad, optimización mediante índices y ejecución de consultas SQL complejas.

Las actividades establecidas son las siguientes:

a. Creación de tablas: implementación del esquema físico

El diseño lógico previamente modelado se traduce al siguiente conjunto de instrucciones SQL, donde se definen las tablas, sus atributos, tipos de datos, claves primarias y claves foráneas:

```
CREATE TABLE IF NOT EXISTS autor (
   aut_id SERIAL,
   aut_nombres CHARACTER VARYING (250),
   aut_apellidos CHARACTER VARYING (250),
   aut_fch_regidtro DATE,
   aut_estado CHARACTER VARYING (100),
   CONSTRAINT pk_autor PRIMARY KEY (aut_id)
);
CREATE TABLE IF NOT EXISTS editorial (
   edi_id SERIAL,
   edi_nombre CHARACTER VARYING (1000),
```

```
    edi_fch_registro DATE,
    edi_estado CHARACTER VARYING (100),
    CONSTRAINT pk_editorial PRIMARY KEY (edi_id)
);
CREATE TABLE IF NOT EXISTS obra_bibliografica (
    obr_id SERIAL,
    edi_id INTEGER,
    obr_codigo INTEGER,
    obr_titulo CHARACTER VARYING (1000),
    obr_isbn CHARACTER VARYING (25),
    obr_edicion CHARACTER VARYING (1000),
    obr_volumen CHARACTER VARYING (1000),
    obr_nro_paginas INTEGER,
    obr_fch_publicacion DATE,
    obr_fch_registro DATE,
    obr_idioma CHARACTER VARYING (500),
    obr_formato CHARACTER VARYING (500),
    obr_tipo_material CHARACTER VARYING (500),
    obr_resumen TEXT,
    obr_estado CHARACTER VARYING (100),
    CONSTRAINT pk_obra_bibliografica PRIMARY KEY (obr_id),
    CONSTRAINT fk_obr_bib_edit FOREIGN KEY (edi_id) REFERENCES
editorial(edi_id)
);
CREATE TABLE IF NOT EXISTS autor_obra (
    aut_id INTEGER,
    obr_id INTEGER,
    fch_registro DATE,
    estado CHARACTER VARYING (100),
    CONSTRAINT pk_autor_obra PRIMARY KEY (aut_id, obr_id),
```

```
    CONSTRAINT fk_autor FOREIGN KEY (aut_id) REFERENCES autor(aut_id),
    CONSTRAINT fk_obra FOREIGN KEY (obr_id) REFERENCES obra_bibliografica(obr_id)
);
```

Estas sentencias establecen las relaciones clave entre las entidades: un autor puede participar en múltiples obras, y cada obra pertenece a una editorial específica. Se garantiza la integridad referencial mediante claves foráneas.

b. Inserción de datos de ejemplo

Una vez creadas las tablas, se puede proceder con la carga inicial de datos mediante instrucciones INSERT INTO, por ejemplo:

```
INSERT INTO editoriales (edi_nombre, edi_fch_registro, edi_estado)
VALUES ('Editorial Alfa', '2024-01-10', 'activo');
INSERT INTO autores (aut_nombres, aut_apellidos, aut_fch_regidtro, aut_estado)
VALUES ('Laura', 'Martínez', '2024-03-01', 'activo');
INSERT INTO obras_bibliograficas (edi_id, obr_codigo, obr_titulo, obr_isbn, obr_fch_publicacion, obr_estado)
VALUES (1, 101, 'Fundamentos de Bases de Datos', '978-1234567890', '2024-02-15', 'activo');
INSERT INTO autores_obras (aut_id, obr_id, fch_registro, estado)
VALUES (1, 1, '2024-03-05', 'activo');
```

c. Consultas complejas y análisis de rendimiento

Se pueden plantear consultas que integren múltiples tablas utilizando JOIN anidados. Por ejemplo:

```
SELECT a.aut_nombres, a.aut_apellidos, o.obr_titulo, e.edi_nombre
FROM autores a
JOIN autores_obras ao ON a.aut_id = ao.aut_id
JOIN obras_bibliograficas o ON ao.obr_id = o.obr_id
JOIN editoriales e ON o.edi_id = e.edi_id;
```

Esta consulta devuelve una vista consolidada de autores, títulos de obras y sus respectivas editoriales. Para analizar su rendimiento, se recomienda utilizar:

```
EXPLAIN ANALYZE
```

```
SELECT a.aut_nombres, a.aut_apellidos, o.obr_titulo, e.edi_nombre
FROM autores a
JOIN autores_obras ao ON a.aut_id = ao.aut_id
JOIN obras_bibliograficas o ON ao.obr_id = o.obr_id
JOIN editoriales e ON o.edi_id = e.edi_id;
```

La salida de EXPLAIN ANALYZE permite al estudiante examinar el plan de ejecución de la consulta, identificar posibles cuellos de botella y evaluar si sería conveniente la creación de índices adicionales sobre columnas clave como obr_id, aut_id o edi_id.

4.10. Tendencias actuales en bases de datos relacionales

El universo de las bases de datos relacionales continúa evolucionando, fortaleciendo su relevancia técnica y práctica en escenarios contemporáneos. A continuación, se destacan las siguientes tendencias emergentes:

1. **Optimización de rendimiento y escalabilidad en RDBMS de código abierto:** Los sistemas PostgreSQL, MySQL y MariaDB, pilares del ecosistema relacional libre, están experimentando mejoras sustanciales en ejecución de consultas, indexación avanzada y capacidad de carga. Asimismo, se observa una creciente integración con tecnologías emergentes como la nube y el aprendizaje automático, lo cual amplía sus aplicaciones en entornos empresariales exigentes (Akinola, 2025).
2. **Integración de Inteligencia Artificial en bases SQL:** La inteligencia artificial y el aprendizaje automático están transformando el paisaje SQL: se emplean algoritmos avanzados para optimizar consultas de forma adaptativa, permitir análisis predictivos en tiempo real y fortalecer la seguridad. Esto ha dado lugar a sistemas híbridos que combinan lo relacional con capacidades propias de las bases de datos NoSQL, ampliando sus usos en contextos de big data y analítica avanzada (Islam, 2024).
3. **Arquitecturas en la nube y bases relacionales serverless:** La adopción de arquitecturas sin servidor (serverless) en entornos relacionales y la transición hacia bases de datos nativas de la nube están redefiniendo la accesibilidad, elasticidad y gestión administrativa de las RDBMS modernas. Esto permite una reducción significativa en los costos de administración y una escalabilidad dinámica frente a cargas variables de trabajo Li (2023).
4. **Modelos relacionales avanzados y enfoque entidad-relación:** Recientes investigaciones abogan por elevar el nivel de abstracción de los sistemas relacionales hacia modelos basados en entidades y relaciones, lo que puede mejorar la independencia lógica y promover

una innovación más profunda en el diseño de bases de datos relacionales. Un ejemplo de esta tendencia es el desarrollo de ErbiumDB, que explora la integración de la abstracción E-R como núcleo del motor de base de datos (Deshpande, 2025).

5. **Modelos de fondo para RDB integrando aprendizaje profundo:** Finalmente, se observa un avance hacia el uso de modelos de base entrenados específicamente para bases de datos relacionales. Un caso reciente es Griffin, un modelo fundacional que combina tareas diversas mediante codificadores avanzados y arquitecturas de atención cruzada, logrando resultados claramente transferibles a distintos ámbitos de aplicación de la gestión de datos (Wang et al., 2025).

Resumen de la unidad

La Unidad 4 aborda de forma integral el diseño físico de bases de datos, última fase del proceso de desarrollo, en la que se traduce el modelo lógico en estructuras reales implementadas sobre un sistema gestor de bases de datos (DBMS), como PostgreSQL. Esta etapa es determinante para el rendimiento y la escalabilidad del sistema, ya que define cómo serán almacenados, organizados, protegidos y accedidos los datos en el nivel físico, considerando características del hardware, sistema operativo y del propio motor de base de datos.

El diseño físico inicia con la creación de tablas mediante la instrucción CREATE TABLE, donde se especifican atributos, tipos de datos (*VARCHAR, DATE, DECIMAL*, entre otros) y restricciones asociadas. Se incluyen también mecanismos para definir dominios personalizados, como CHECK y tipos ENUM, útiles para validar estados o clasificaciones predefinidas. Posteriormente, se profundiza en la definición de claves primarias y foráneas, esenciales para garantizar la unicidad de los registros y mantener la integridad referencial entre tablas relacionadas. Se incorporan cláusulas como ON DELETE **y** ON UPDATE para gestionar el comportamiento de las relaciones en operaciones de actualización y eliminación.

La unidad introduce también las restricciones de integridad (NOT NULL, UNIQUE, DEFAULT) y validaciones de dominio, que fortalecen el control de calidad de los datos desde el momento en que son ingresados. Se presentan ejemplos detallados aplicados a modelos reales como *empleado, libro* y *autor_libro*. Otro pilar esencial del diseño físico es la optimización de consultas mediante índices, incluyendo índices simples, compuestos, parciales y basados en funciones (CREATE INDEX). Se analizan sus beneficios y costos, así como su impacto en el rendimiento de operaciones SELECT en grandes volúmenes de datos.

La unidad también incorpora el uso de vistas (CREATE VIEW) y funciones almacenadas (FUNCTIONS), además de triggers, como mecanismos que promueven la abstracción, la reutilización de lógica y la aplicación automática de reglas de negocio. Se dedica un apartado a la carga de datos y estrategias de población inicial, diferenciando entre INSERT, COPY y herramientas gráficas o ETL para migraciones complejas. Asimismo, se abordan buenas prácticas para la creación de scripts de población en proyectos reales. En el contexto de la alta disponibilidad, se introduce la replicación en streaming de PostgreSQL, junto con estrategias de conmutación por error (failover) y respaldo (pg_dump, pg_basebackup) como parte de un enfoque integral de continuidad operativa.

El contenido incluye un caso práctico integrador en el que los estudiantes implementan un modelo relacional completo mediante instrucciones SQL reales; cargan datos, crean índices y ejecutan consultas complejas. El análisis de rendimiento mediante EXPLAIN ANALYZE permite reflexionar sobre las decisiones de diseño físico tomadas y evaluar mejoras.

Además, esta unidad conecta los aprendizajes con las tendencias actuales en bases de datos relacionales, tales como la integración de inteligencia artificial para la optimización automática de consultas (Islam, 2024), la adopción de arquitecturas en la nube y modelos serverless que permiten escalabilidad dinámica (Modernization of Databases in the Cloud Era, 2024), y la evolución hacia modelos relacionales híbridos que combinan el enfoque entidad-relación con capacidades propias de bases de datos avanzadas (Deshpande, 2025). Estas tendencias refuerzan la pertinencia del diseño físico, ya que permiten comprender cómo los fundamentos tradicionales se integran con innovaciones que impulsan la gestión moderna de la información.

En síntesis, esta unidad proporciona al estudiante un enfoque profesional del diseño físico, conectando la teoría con la implementación concreta en un SGBD moderno y, al mismo tiempo, con las innovaciones que marcarán el futuro de la disciplina. Le otorga, por tanto, las herramientas necesarias para construir sistemas robustos, eficientes, escalables y alineados con las exigencias actuales del entorno digital.

Glosario de términos clave

Diseño físico: Etapa del desarrollo de bases de datos donde el modelo lógico se implementa físicamente sobre un sistema gestor, optimizando almacenamiento y acceso.

CREATE TABLE: Instrucción SQL utilizada para crear una nueva tabla en una base de datos, especificando atributos, tipos de datos y restricciones.

Tipo de dato: Clasificación que define la naturaleza de los valores que puede almacenar un atributo (Por ejemplo: VARCHAR, DATE, INTEGER).

CHECK: Restricción SQL que impone una condición lógica que deben cumplir los valores insertados en una columna.

ENUM: Tipo de dato definido por el usuario en PostgreSQL, que restringe los valores posibles a un conjunto cerrado predefinido.

PRIMARY KEY: Restricción que identifica de forma única cada fila de una tabla; no permite valores nulos ni duplicados.

FOREIGN KEY: Restricción que establece una relación referencial entre tablas, asegurando coherencia entre claves primarias y foráneas.

ON DELETE / ON UPDATE: Cláusulas que definen el comportamiento del sistema ante eliminación o actualización de registros referenciados.

NOT NULL: Restricción que impide que una columna quede sin valor, exigiendo un dato obligatorio.

UNIQUE: Restricción que asegura que todos los valores en una columna sean distintos entre sí.

DEFAULT: Valor que se asigna automáticamente a una columna si no se proporciona uno explícito durante la inserción.

Índice: Estructura que permite acelerar la búsqueda y recuperación de datos dentro de una tabla.

Índice compuesto: Índice que abarca dos o más columnas, optimizando consultas que las utilizan en conjunto.

Índice parcial: Índice construido sobre un subconjunto de filas que cumplen una condición específica.

VISTA: Consulta almacenada que actúa como una tabla virtual, utilizada para simplificar el acceso y proporcionar abstracción.

FUNCIÓN almacenada: Bloque de código SQL reutilizable que encapsula lógica de negocio, recibe parámetros y devuelve resultados.

TRIGGER: Procedimiento automático que se ejecuta ante eventos específicos como inserciones o actualizaciones.

COPY: Comando SQL utilizado para cargar datos masivamente desde archivos externos a una tabla.

pg_dump: Herramienta de PostgreSQL para realizar respaldos lógicos de estructuras y datos de una base de datos.

pg_basebackup: Utilidad de PostgreSQL para crear respaldos físicos completos del clúster de base de datos.

Streaming replication: Técnica en PostgreSQL para replicar datos en tiempo real desde un servidor principal hacia servidores secundarios.

EXPLAIN ANALYZE: Comando que analiza y muestra el plan de ejecución de una consulta SQL, útil para evaluar su rendimiento.

Preguntas de autoaprendizaje

Instrucciones: Marca la opción correcta en cada caso.

1. ¿Cuál es el propósito principal del diseño físico en una base de datos relacional?

- **a.** Representar visualmente entidades y relaciones en un diagrama.
- **b.** Optimizar el almacenamiento y acceso eficiente a los datos en disco.
- **c.** Identificar claves primarias y foráneas.
- **d.** Asignar nombres lógicos a las entidades del modelo conceptual.

2. ¿Qué instrucción SQL se utiliza para crear una nueva tabla en el esquema de una base de datos?

- **a.** INSERT TABLE
- **b.** NEW TABLE
- **c.** CREATE TABLE
- **d.** ADD STRUCTURE

3. ¿Cuál de las siguientes restricciones garantiza que una columna no pueda contener valores nulos?

- **a.** NOT NULL
- **b.** UNIQUE
- **c.** CHECK
- **d.** DEFAULT

4. En PostgreSQL, ¿qué tipo de índice permite acelerar búsquedas sobre subconjuntos de datos filtrados por una condición?

a. Índice compuesto

b. Índice parcial

c. Índice hash

d. Índice primario

5. ¿Qué instrucción permite definir una relación entre dos tablas a través de una clave foránea?

a. CREATE VIEW

b. CONSTRAINT DEFAULT

c. TRIGGER ASSIGN

d. FOREIGN KEY REFERENCES

Respuestas 1(b); 2(c); 3(a); 4(b); 5(d).

Ejercicios prácticos para resolver

Ejercicio 1. Creación del esquema físico

De acuerdo con el Modelo Relacional de la Unidad 03 (ver Figura 3.11), crea las tablas categoría y producto utilizando SQL, definiendo adecuadamente tipos de datos, claves primarias y foráneas. Para tal propósito realiza la siguiente actividad:

Redacta las instrucciones SQL para crear ambas tablas con las siguientes características:

- categoría:
    - cat_id: entero, clave primaria, autoincremental.
    - cat_nombre: cadena de texto obligatoria y única.
    - cat_descripcion: cadena de texto opcional.
    - cat_estado: valor por defecto 'activo' y debe ser 'activo' o 'inactivo'.
- producto:
    - pro_id: entero, clave primaria.
    - pro_nombre: texto obligatorio.
    - pro_descripcion: texto opcional.
    - pro_stock: entero mayor o igual a cero.
    - pro_precio: decimal positivo.

- pro_estado: restringido a 'activo', 'inactivo'.
- cat_id: clave foránea que referencia a categoria(cat_id).

Indicaciones técnicas: Utiliza restricciones NOT NULL, CHECK, DEFAULT y FOREIGN KEY.

Ejercicio 2. Inserción y validación de datos

El propósito del ejercicio es cargar registros y verificar integridad referencial y restricciones de dominio, para lo cual realiza las siguientes actividades:

1. Inserta al menos **tres categorías** con distintos valores en cat_estado.
2. Inserta **cinco productos** asociados a estas categorías.
3. Intenta insertar un producto sin cat_id válido o con pro_precio negativo. Explica el resultado.

Ejercicio 3. Creación de índices

Optimiza el rendimiento de las búsquedas frecuentes, ejecutando las siguientes tareas:

1. Crea un índice simple sobre pro_nombre para acelerar búsquedas por nombre.
2. Crea un índice parcial que solo incluya productos cuyo estado sea 'activo'.
3. Verifica su efecto utilizando la instrucción EXPLAIN ANALYZE con una consulta SELECT filtrada.

Ejercicio 4. Definición de vistas

Simplifica el acceso a información útil y recurrente, realizando lo siguiente:

1. Crea una vista vista_productos_activos que muestre:
   - pro_nombre, pro_precio, cat_nombre
     …de los productos que estén activos, usando JOIN con categoria.

Ejercicio 5. Automatización y control

Implementa la lógica automatizada con funciones y triggers, para lo cual realiza las siguientes actividades:

1. Define una función que registre en una tabla log_precio cualquier cambio en el precio de un producto.
2. Crea un trigger AFTER UPDATE sobre producto que invoque la función si pro_precio es modificado.

Ejercicio 6. Migración y carga masiva

Simula la carga inicial de datos desarrollando las siguientes tareas:

1. Crea un archivo .csv con al menos 10 registros de productos.
2. Usa el comando COPY para cargar ese archivo en la tabla producto.
3. Verifica que todos los registros se insertaron correctamente, y realiza una consulta resumen por categoría.

Referencias bibliográficas de la Unidad 4

Akinola, S. (2024). Trends in Open Source RDBMS: Performance, Scalability and Security Insights. Journal of Research in Science and Engineering, 6(7), 22–28. https://doi.org/10.53469/jrse.2024.06(07).05.

Chávez, J. (2019). FUNDAMENTOS DE POSTGRESQL. IEASS, Editores.

Cornelio, E., Rivas, C., & Hernández, J. C. (2004). Bases de datos relacionales: diseño físico.

Deshpande, A. (2025). Beyond Relations: A Case for Elevating to the Entity-Relationship Abstraction. arXiv. https://arxiv.org/abs/2505.03536.

Dombrovskaya, H., Novikov, B., & Bailliekova, A. (2024). PostgreSQL Query Optimization. In PostgreSQL Query Optimization (2nd ed.). Apress L. P. https://doi.org/10.1007/979-8-8688-0069-6.

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson.

Garcia-Molina, Hector., Ullman, J. D. ., & Widom, Jennifer. (2009). Database systems : the complete book (2nd ed.). Pearson Prentice Hall.

Ibáñez, L. H. (2014). Administración de Sistemas Gestores de Bases de Datos. In RA-MA Editorial (Vol. 39, Issue 5).

Islam, S. (2024). Future Trends in SQL Databases and Big Data Analytics: Impact of Machine Learning and Artificial Intelligence. SSRN. https://doi.org/10.2139/ssrn.5064781.

ITAI, Y., OLUDELE PhD, A., & GOGA PhD, N. (2013). Trigger and Database Security. INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY, 4(1). https://doi.org/10.24297/ijct.v4i1b.3060.

Jiménez, C., & Armstrong, T. (2018). El rol del lenguaje SQL en los SGBDR y en la implementación del Modelo Relacional. El Rol Del Lenguaje SQL En Los SGBDR y En La Implementación Del Modelo Relacional.

Lelarge, G., & Rouhaud, J. (2023). PostgreSQL - Architecture et notions avancées (5th ed.). D-Booker éditions.

Li, F. (2023). Modernization of databases in the cloud era: Building databases that run like Legos. *Proceedings of the VLDB Endowment, 16*(12), 3478–3487. https://doi.org/10.14778/3611540.3611639

López Herrera, P. (2016). Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL. Universidad Tecnológica Del Sur Del Estado de México.

Modernization of Databases in the Cloud Era. (2024). Proceedings of the VLDB Endowment, 17(12), 3560-3572. https://doi.org/10.14778/3611540.3611639

Piñeiro Gómez, J. (2013). Bases de datos relacionales y modelado de datos. Ediciones Paraninfo, S.A.

Ramakrishnan, R., & Gehrke, J. (2002). Database Management Systems (3rd ed.). McGraw-Hill.

Salazar Cárdenas, J. E. (2014). Análisis comparativo de dos bases de datos SQL y dos bases de datos no SQL.

Schönig, H.-J. (2015). PostgreSQL Replication - Second Edition (2nd ed.). Packt Publishing.

Silva, R. (2020). Essential Postgres: Database Development using PostgreSQL.

Tezuysal, A., & Ahmed, I. (2024). Database design and modeling with PostgreSQL and MySQL. Packt Publishing.

Yannakoudakis, E. J. (1988). The Architectural Logic of Database Systems. In The Architectural Logic of Database Systems. Springer London. https://doi.org/10.1007/978-1-4471-1616-5.

Wang, Y., Wang, X., Gan, Q., et al. (2025). *Griffin: Towards a Graph-Centric Relational Database Foundation Model*. arXiv. https://arxiv.org/abs/2505.05568.

Freddy Jumbo Castillo, ORCID: 0000-0002-5200- 7162. Correo: fjumbo@utmachala.edu.ec. Filial: Universidad Técnica de Machala. Es Ingeniero en Sistemas, con Maestría Universitaria en Inteligencia Artificial, Maestría en Ciencia de Datos Aplicada, Master of Science en Geographical Information Science & Systems y Maestría en Educación Superior. Actualmente se desempeña como docente en la carrera de Tecnologías de la Información de la Universidad Técnica de Machala. Ha participado activamente en proyectos de investigación y vinculación, destacándose por su contribución al diseño lógico de bases de datos en iniciativas como el Inventario Participativo de Recursos Hídricos (ex SENAGUA) y PROMOWEAPP del GAD Santa Rosa. Su formación interdisciplinaria y experiencia universitaria fortalecen su aporte al desarrollo de soluciones tecnológicas y al impulso de una educación superior ética, crítica y orientada al entorno.

Mariuxi Zea Ordoñez, ORCID: 0000-0001-8860- 6282. Correo: mzea@utmachala.edu.ec. Filial: Universidad Técnica de Machala. Ingeniera en Computación, con Maestría en Sistemas de Información Gerencial y Máster en Docencia Universitaria, se desempeña como docente en la carrera de Tecnologías de la Información de la Universidad Técnica de Machala. Su trayectoria académica se ha caracterizado por una participación activa en procesos de formación, tutoría estudiantil y actividades de vinculación con la sociedad, fortaleciendo de manera efectiva las competencias profesionales en el ámbito de las TIC. Ha contribuido al diseño conceptual de bases de datos en proyectos institucionales como SIMMO, aportando con enfoques pedagógicos y técnicos orientados a la sostenibilidad de soluciones tecnológicas. Actualmente, dirige proyectos de investigación dentro del Grupo GIIS y mantiene un firme compromiso con la mejora continua de la educación superior en el entorno digital.

Oscar Cárdenas Villavicencio, ORCID: 0000-0001- 6570-8040. Correo: oecardenas@utmachala.edu.ec. Filial: Universidad Técnica de Machala. Ingeniero de Sistemas, con Maestría en Telecomunicaciones y Maestría en Ciencia de Datos Aplicada, ejerce funciones como docente en la carrera de Tecnologías de la Información de la Universidad Técnica de Machala. Su compromiso académico se manifiesta en la dirección del proyecto de vinculación SIMMO, en el cual lideró el diseño lógico y físico de la base de datos, asegurando su eficiencia y escalabilidad. Además, integra el Grupo de Investigación GIIS y coordina el Colectivo de Prácticas Laborales, promoviendo el vínculo entre la formación académica y el entorno profesional. Su labor se orienta al fortalecimiento del perfil técnico de los estudiantes y al desarrollo de soluciones tecnológicas contextualizadas a las necesidades institucionales y sociales.

ISBN: 978-9942-53-027-1

9 789942 530271

Compás
capacitación e investigación