

CAPÍTULO 1: MODELOS DE DATOS

Material de Clase para Fundamentos de Base de Datos

1.1 Introducción a los Sistemas de Base de Datos

¿Qué es un Sistema de Base de Datos?

Un **Sistema de Base de Datos** (SBD) es un conjunto integrado de componentes que permite almacenar, organizar, gestionar y recuperar datos de manera eficiente y controlada.

Componentes principales:

1. **Base de Datos (BD):** Colección organizada de datos relacionados
2. **Sistema Gestor de Base de Datos (SGBD/DBMS):** Software que gestiona la BD
3. **Usuarios:** Personas que interactúan con el sistema
4. **Aplicaciones:** Programas que acceden a los datos

Ejemplo Real: Sistema de una Universidad

Imaginemos el sistema de información de una universidad:

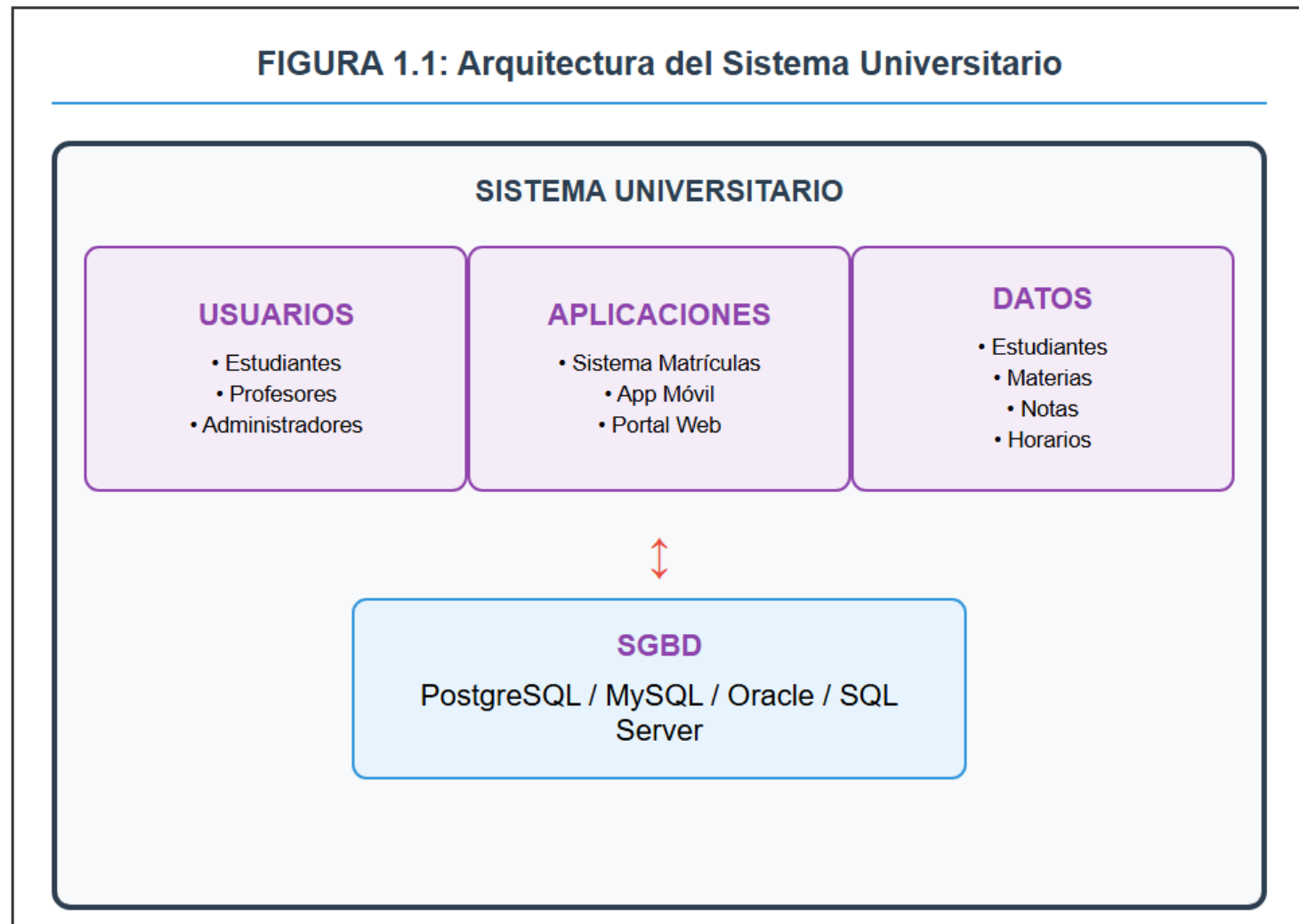
Arquitectura del Sistema Universitario:

El sistema se compone de cuatro capas principales:

1. **Capa de Usuarios:** Estudiantes, Profesores, Administradores
2. **Capa de Aplicaciones:** Sistema de Matrículas, App Móvil, Portal Web
3. **Capa de Datos:** Estudiantes, Materias, Notas, Horarios

4. Capa de SGBD: PostgreSQL, MySQL, Oracle, Sqlserver

[FIGURA 1.1: Diagrama de arquitectura del sistema universitario mostrando las cuatro capas interconectadas]



- **Datos almacenados:** Estudiantes, profesores, materias, calificaciones, horarios
- **SGBD:** PostgreSQL, MySQL, Oracle
- **Usuarios:** Estudiantes (consultan calificaciones), profesores (registran notas), administradores (gestionan matrículas)
- **Aplicaciones:** Sistema web de matrículas, app móvil de consulta de notas

Ventajas de los Sistemas de BD vs. Archivos Tradicionales:

Comparación Sistema de Archivos vs Base de Datos:

SISTEMA DE ARCHIVOS TRADICIONAL:

- Archivos separados: Archivo_Matrículas, Archivo_Biblioteca, Archivo_Cafetería
- Cada archivo contiene datos de estudiantes duplicados
- Problemas: Redundancia, inconsistencias, mantenimiento difícil, sin control de acceso

SISTEMA DE BASE DE DATOS:

- Una sola tabla de Estudiantes centralizada
- Múltiples aplicaciones acceden a los mismos datos
- Ventajas: Sin redundancia, consistencia garantizada, control de acceso, backup centralizado

[FIGURA 1.2: Diagrama comparativo mostrando archivos duplicados vs base de datos centralizada]

FIGURA 1.2: Sistema de Archivos vs Base de Datos

ARCHIVOS TRADICIONALES



- ✗ REDUNDANCIA
- ✗ INCONSISTENCIAS
- ✗ DIFÍCIL MANTENIMIENTO
- ✗ NO HAY CONTROL DE ACCESO
- ✗ BACKUP INDIVIDUAL

BASE DE DATOS



- ✓ SIN REDUNDANCIA
- ✓ CONSISTENCIA GARANTIZADA
- ✓ CONTROL DE ACCESO
- ✓ BACKUP CENTRALIZADO
- ✓ INTEGRIDAD DE DATOS

1. **Reducción de redundancia:** Un estudiante se registra una vez, no en cada materia
2. **Integridad de datos:** Las calificaciones deben estar entre 0 y 100
3. **Seguridad:** Solo los profesores pueden modificar calificaciones

4. **Concurrencia:** Múltiples usuarios pueden consultar datos simultáneamente
5. **Respaldo y recuperación:** Protección contra pérdida de datosAplicaciones: Sistema web de matrículas, app móvil de consulta de notas

Ventajas de los Sistemas de BD vs. Archivos Tradicionales:

1. **Reducción de redundancia:** Un estudiante se registra una vez, no en cada materia
 2. **Integridad de datos:** Las calificaciones deben estar entre 0 y 100
 3. **Seguridad:** Solo los profesores pueden modificar calificaciones
 4. **Concurrencia:** Múltiples usuarios pueden consultar datos simultáneamente
 5. **Respaldo y recuperación:** Protección contra pérdida de datos
-

1.2 Modelo Conceptual de Datos: Modelo Entidad-Relación

¿Qué es el Modelo Entidad-Relación (E-R)?

Es un modelo de datos conceptual que representa la estructura lógica de una base de datos mediante entidades, atributos y relaciones, independiente de cualquier SGBD específico.

Componentes del Modelo E-R:

1. ENTIDADES

Objetos o conceptos del mundo real que tienen existencia independiente.

Ejemplo: Sistema de Biblioteca

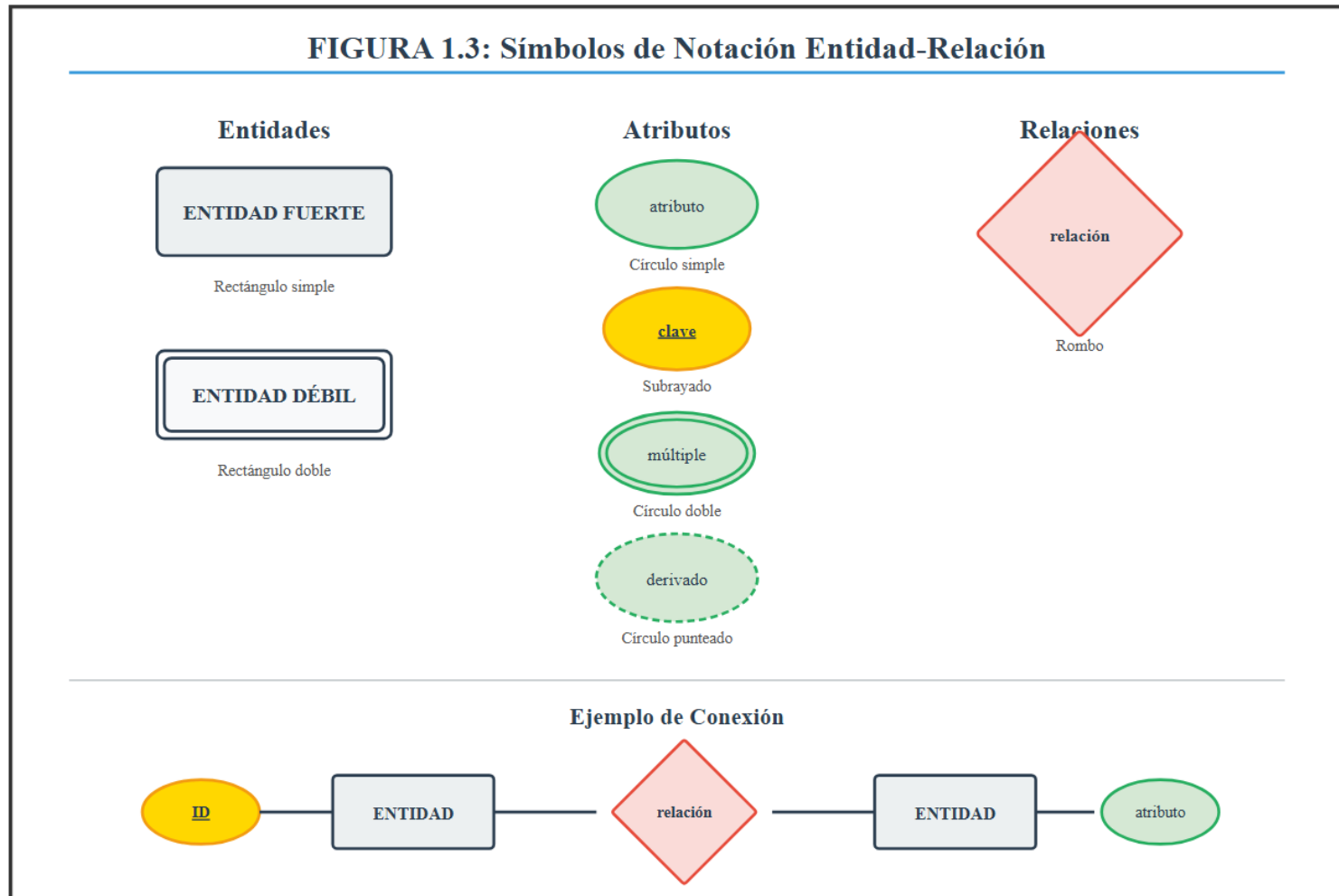
- Entidades: LIBRO, AUTOR, USUARIO, PRÉSTAMO

Notación gráfica:

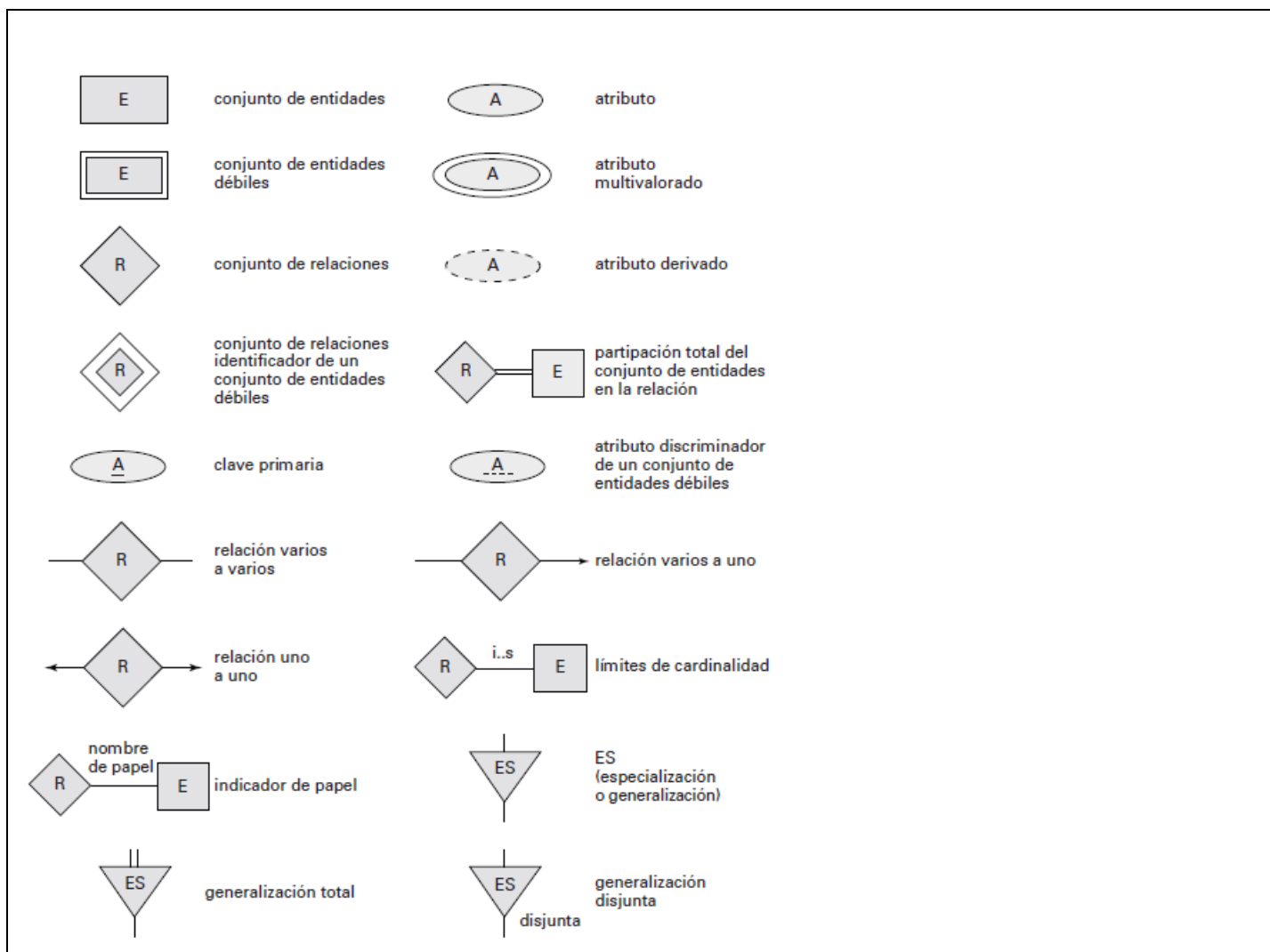
- **Entidad fuerte:** Rectángulo simple con el nombre de la entidad

- **Entidad débil:** Rectángulo doble (un rectángulo dentro de otro)

[FIGURA 1.3: Símbolos de notación E-R mostrando rectángulo simple para entidad fuerte y rectángulo doble para entidad débil]



Según korth



2. ATRIBUTOS

Propiedades que describen las entidades.

Ejemplo para la entidad LIBRO:

- ISBN (clave primaria) •

Título

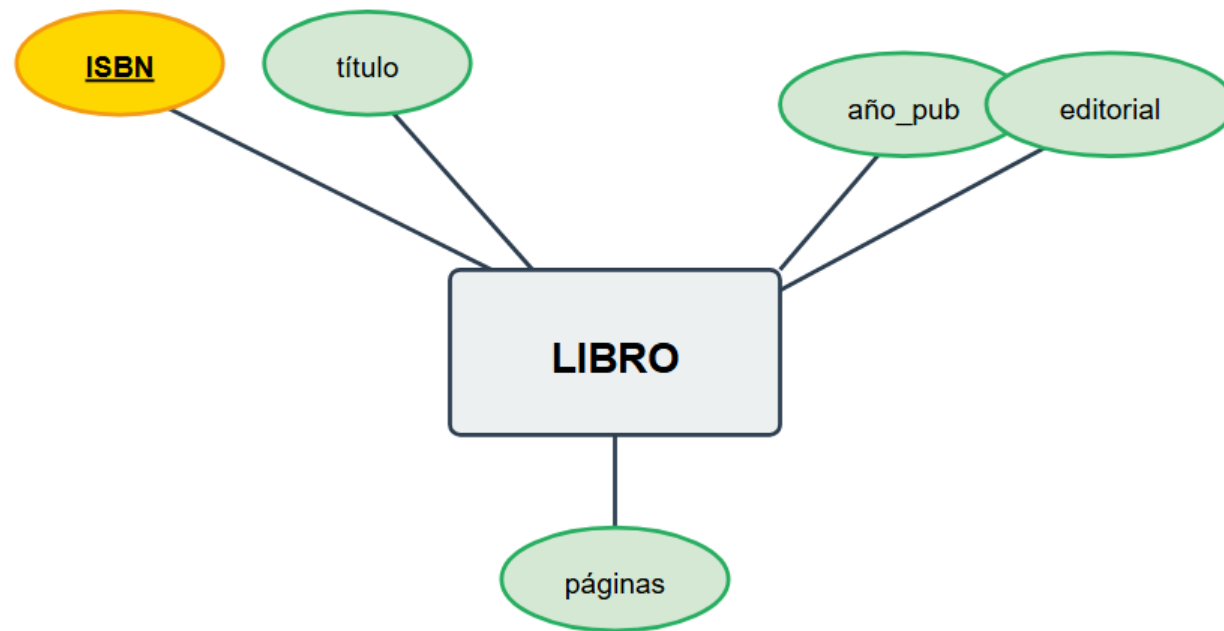
- Año_publicación
- Número_páginas
- Editorial

Notación gráfica:

- **Atributo simple:** Círculo conectado a la entidad
- **Atributo clave:** Círculo con el nombre subrayado
- **Atributo compuesto:** Círculo principal conectado a círculos secundarios
- **Atributo multivaluado:** Círculo doble
- **Atributo derivado:** Círculo con línea punteada

[FIGURA 1.4: Diagrama mostrando entidad LIBRO conectada a sus atributos usando diferentes símbolos]

FIGURA 1.4: Entidad LIBRO con Atributos



Tipos de atributos con ejemplos:

Simples: No se pueden dividir (Ejemplo: Título) **Compuestos:** Se dividen en subpartes (Ejemplo: Dirección = Calle + Ciudad + Código Postal) **Multivaluados:** Pueden tener múltiples valores (Ejemplo: Teléfonos de una persona) **Derivados:** Se calculan a partir de otros atributos (Ejemplo: Edad se calcula de fecha_nacimiento)

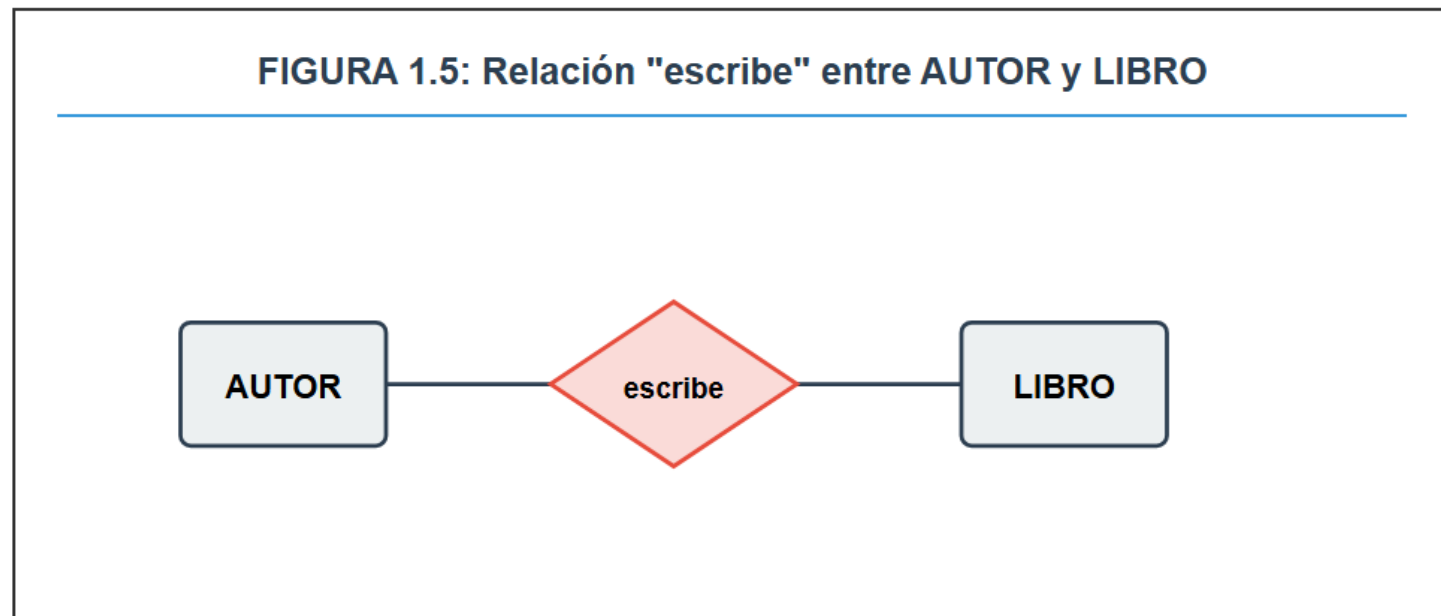
3. RELACIONES

Asociaciones entre entidades.

Notación gráfica:

- **Relación:** Rombo conectando las entidades participantes
- **Cardinalidad:** Números o letras (1, N, M) cerca de las líneas de conexión

[FIGURA 1.5: Ejemplo de relación "escribe" entre entidades AUTOR y LIBRO usando rombo]



Ejemplo:

- Un AUTOR **escribe** un LIBRO
- Un USUARIO **solicita** un PRÉSTAMO
- Un LIBRO **pertenece** a una CATEGORÍA

Cardinalidades en las Relaciones:

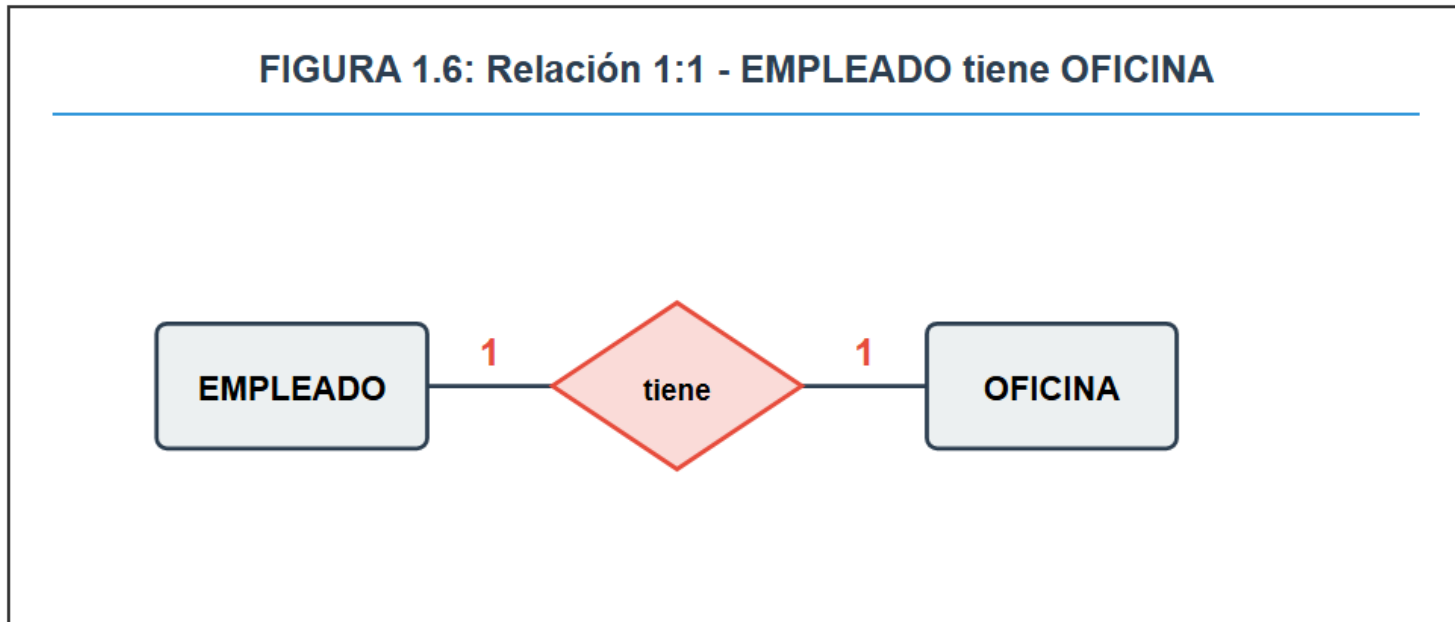
1:1 (Uno a Uno)

Una entidad se relaciona con exactamente una instancia de otra entidad.

Ejemplo: EMPLEADO tiene OFICINA

- Cada empleado tiene asignada una oficina
- Cada oficina está asignada a un solo empleado

[FIGURA 1.6: Diagrama E-R mostrando relación 1:1 entre EMPLEADO y OFICINA]



Un empleado tiene asignada una oficina. Una oficina está asignada a un solo empleado.

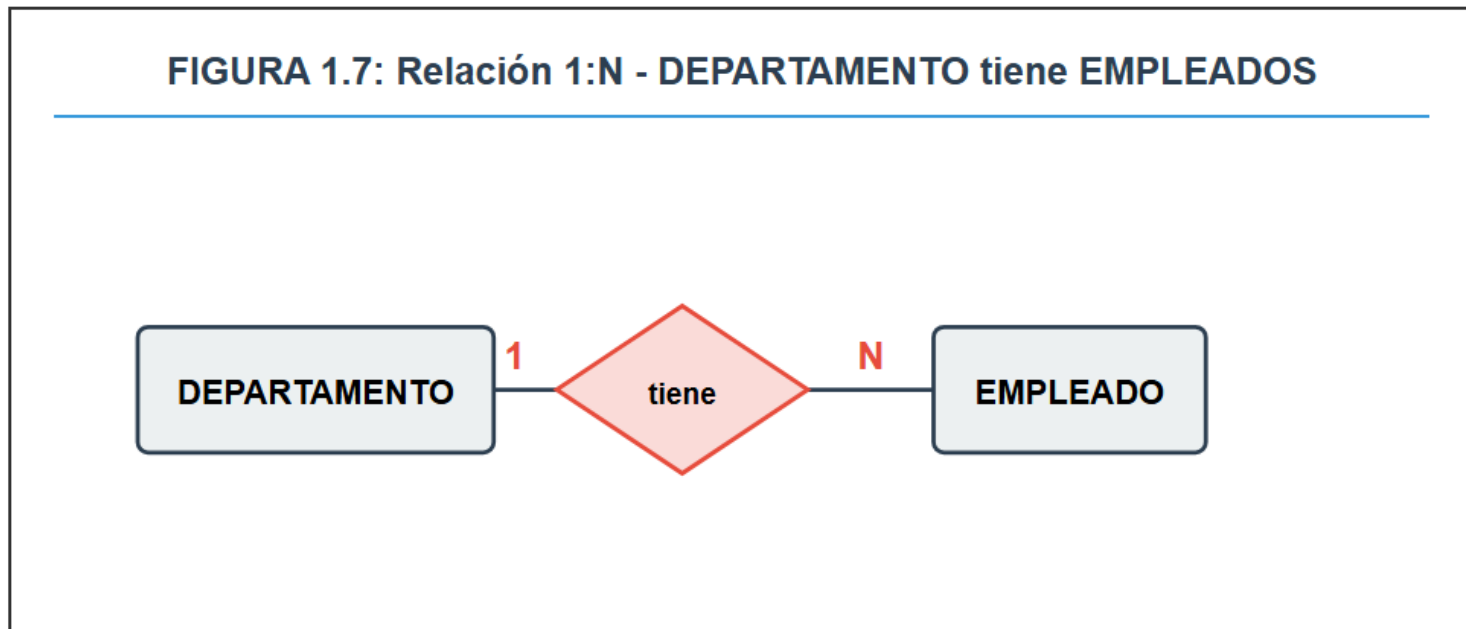
1:N (Uno a Muchos)

Una entidad se relaciona con múltiples instancias de otra entidad.

Ejemplo: DEPARTAMENTO **tiene** EMPLEADOS

- Un departamento tiene muchos empleados
- Un empleado pertenece a un solo departamento

[FIGURA 1.7: Diagrama E-R mostrando relación 1:N entre DEPARTAMENTO y EMPLEADO]



Un departamento tiene muchos empleados. Un empleado pertenece a un solo departamento.

N:M (Muchos a Muchos)

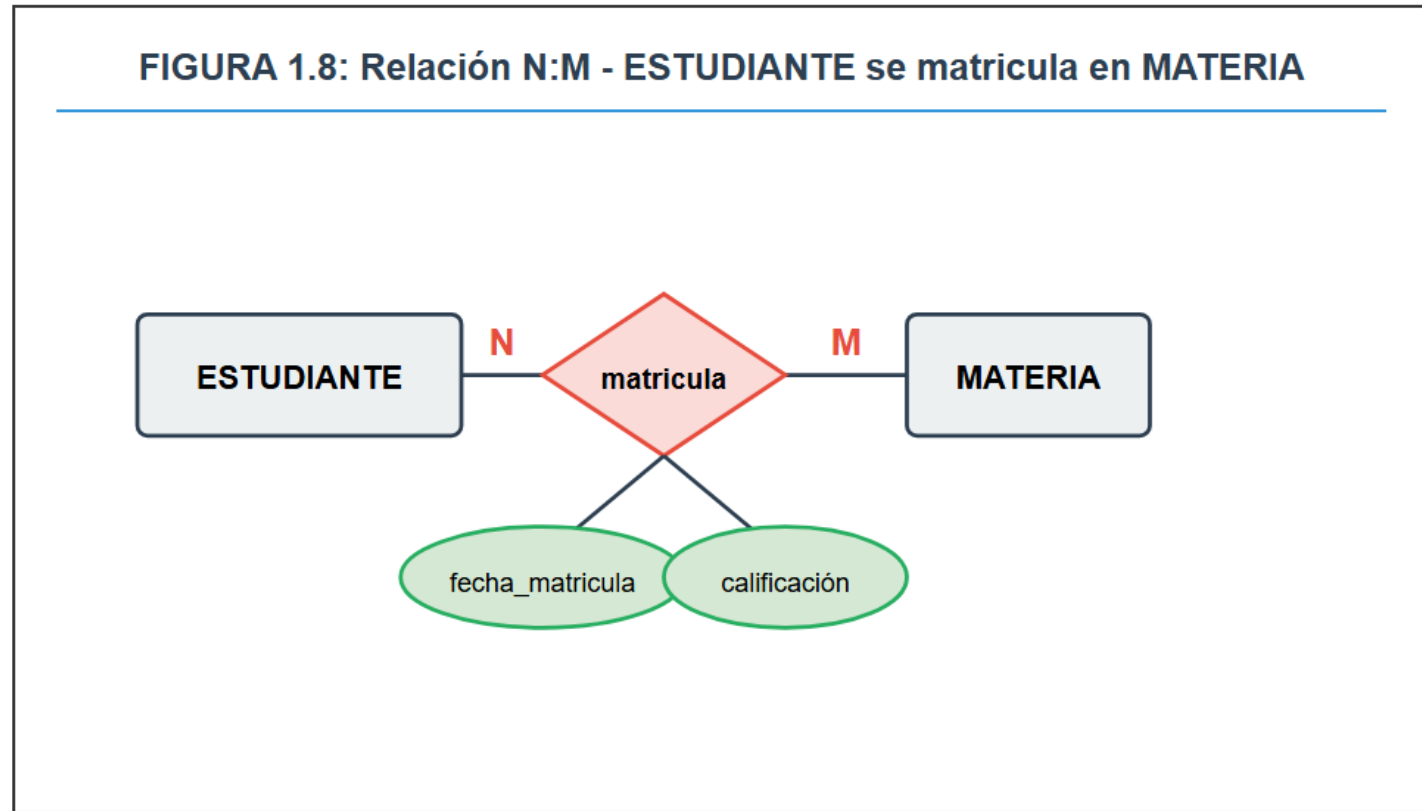
Múltiples instancias de una entidad se relacionan con múltiples instancias de otra.

Ejemplo: ESTUDIANTE **se matricula en** MATERIA

- Un estudiante puede matricularse en varias materias

- Una materia puede tener varios estudiantes

[FIGURA 1.8: Diagrama E-R mostrando relación N:M entre ESTUDIANTE y MATERIA]



Un estudiante puede matricularse en varias materias. Una materia puede tener varios estudiantes.

Ejemplo Completo: Sistema de Hospital

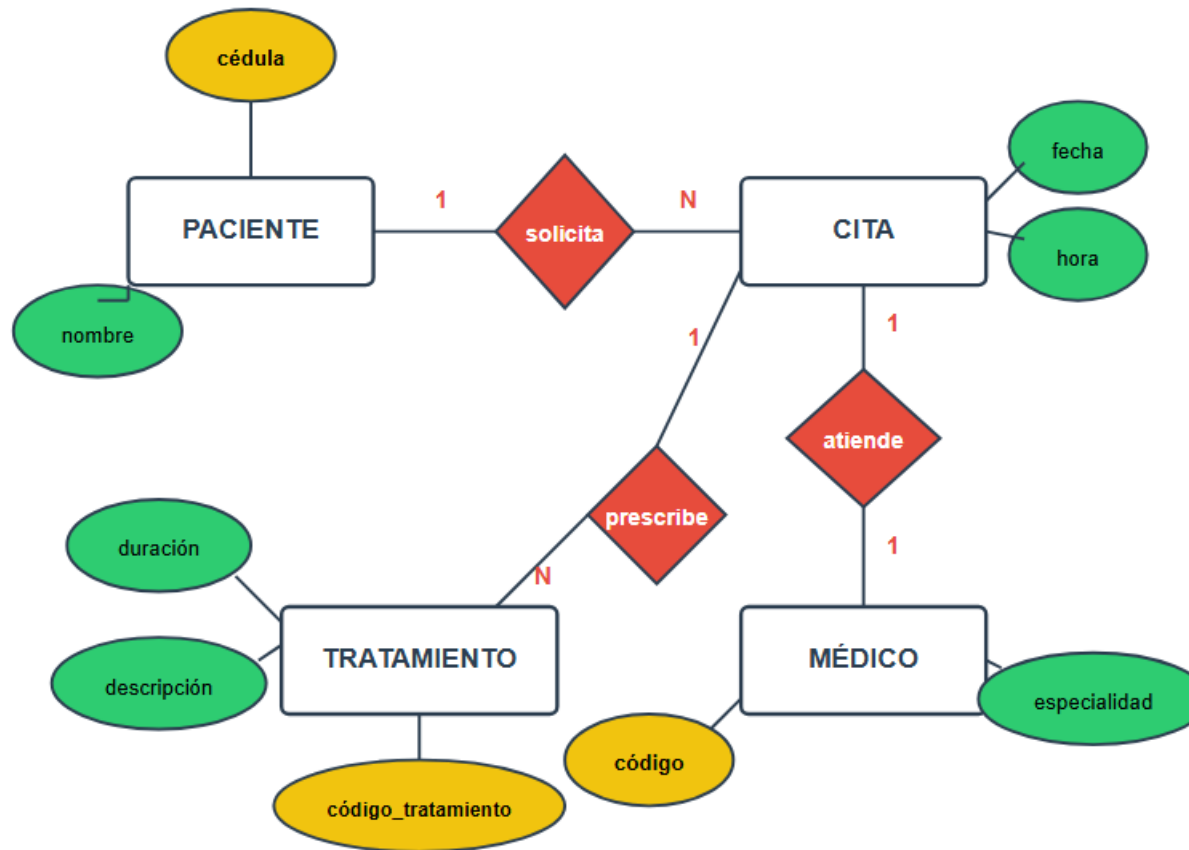
Entidades:

- PACIENTE (cédula, nombre, fecha_nacimiento, dirección, teléfono)
- MÉDICO (código_médico, nombre, especialidad, teléfono)
- CITA (número_cita, fecha, hora, motivo)

- TRATAMIENTO (código_tratamiento, descripción, duración)

Diagrama E-R Completo del Hospital:

FIGURA 1.9: Diagrama E-R Sistema Hospitalario



ENTIDADES Y ATRIBUTOS

1. PACIENTE

- **cédula** (Llave Primaria)
- nombre
- (Otros atributos implícitos: *fecha_nacimiento, dirección, teléfono*)

2. CITA

- **número_cita** (Llave Primaria - implícita)

- fecha
- hora
- *(Otros atributos implícitos: motivo)*

3. MÉDICO

- **código** (Llave Primaria)
- especialidad
- *(Otros atributos implícitos: nombre, teléfono)*

4. TRATAMIENTO

- **código_tratamiento** (Llave Primaria)
- descripción
- duración

RELACIONES Y CARDINALIDADES

1. PACIENTE — solicita — CITA

- **Cardinalidad:** 1:N
- **Lectura:**
 - Un PACIENTE puede solicitar **MUCHAS (N)** CITAS
 - Una CITA es solicitada por **UN SOLO (1)** PACIENTE

2. CITA — atiende — MÉDICO

- **Cardinalidad:** 1:1
- **Lectura:**
 - Una CITA es atendida por **UN SOLO (1)** MÉDICO
 - Un MÉDICO atiende **UNA (1)** CITA a la vez

3. CITA — prescribe — TRATAMIENTO

- **Cardinalidad:** 1:N
 - **Lectura:**
 - Una CITA puede prescribir **MUCHOS (N)** TRATAMIENTOS
 - Un TRATAMIENTO es prescrito en **UNA SOLA (1)** CITA
-

1.3 Modelo Lógico (Relacional) y Modelo Físico

Modelo Lógico de Datos (Modelo Relacional)

El **Modelo Relacional** organiza los datos en tablas (relaciones) compuestas por filas (tuplas) y columnas (atributos).

Conceptos Fundamentales:

1. **Relación (Tabla):** Conjunto de tuplas con la misma estructura
2. **Tupla (Fila):** Cada registro individual en la tabla
3. **Atributo (Columna):** Cada campo que describe una propiedad
4. **Dominio:** Conjunto de valores válidos para un atributo

Ejemplo: Tabla ESTUDIANTE

ID_Estudiante	Nombre	Apellido	Edad	Carrera	Email
001	Juan	Pérez	20	Ingeniería	juan.perez@email.com
002	María	García	19	Medicina	maria.garcia@email.com
003	Carlos	López	21	Derecho	carlos.lopez@email.com

Estructura del modelo relacional:


En el modelo relacional, los datos se organizan en tablas bidimensionales donde:

- **Filas (tuplas):** Representan registros individuales
- **Columnas (atributos):** Representan las propiedades de cada registro
- **Clave primaria:** Identifica únicamente cada fila (marcada como PK)

[FIGURA 1.10: Tabla ESTUDIANTE mostrando estructura relacional con filas y columnas]

FIGURA 1.10: Estructura de Tabla Relacional - ESTUDIANTE

ID_Estudiante (PK)	Nombre	Apellido	Edad	Carrera	Email
001	Juan	Pérez	20	Ingeniería	juan.perez@email.com
002	María	García	19	Medicina	maria.garcia@email.com
003	Carlos	López	21	Derecho	carlos.lopez@email.com

 **PK = Clave Primaria** (Identifica únicamente cada fila)

Relaciones entre tablas:

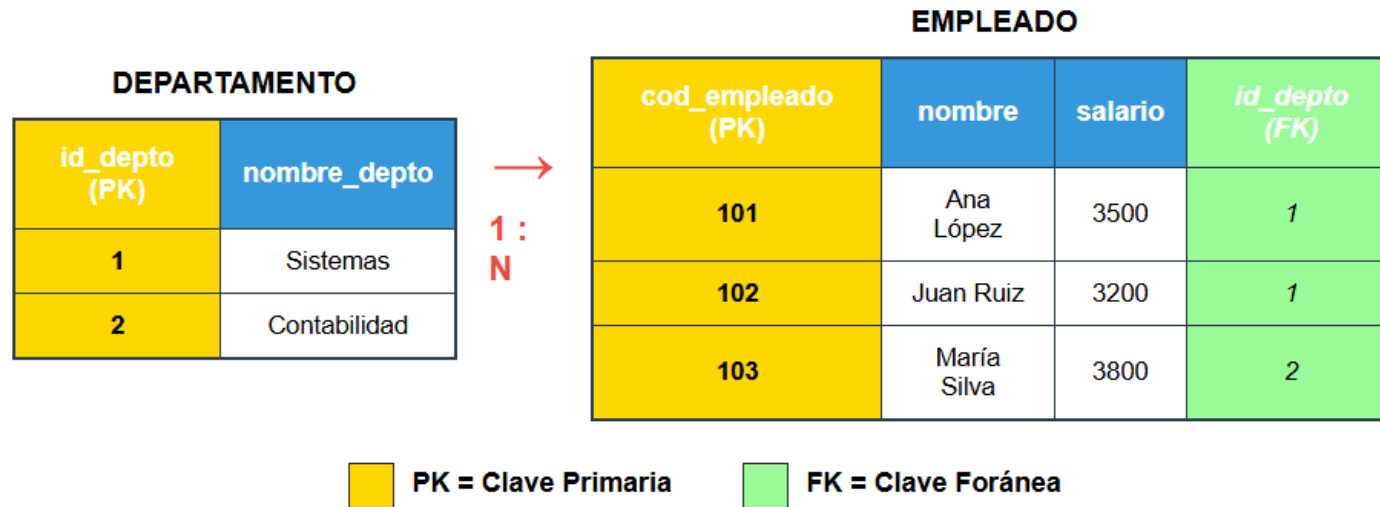
Las tablas se relacionan mediante claves foráneas que referencian claves primarias de otras tablas:

Ejemplo DEPARTAMENTO - EMPLEADO:

- Tabla DEPARTAMENTO tiene clave primaria "id_depto"
- Tabla EMPLEADO tiene clave foránea "id_depto" que referencia a DEPARTAMENTO
- Relación 1:N (un departamento, muchos empleados)

[FIGURA 1.11: Diagrama mostrando relación entre tablas DEPARTAMENTO y EMPLEADO con línea conectando las claves]

FIGURA 1.11: Relación entre Tablas DEPARTAMENTO-EMPLEADO



La clave foránea id_depto en EMPLEADO referencia la clave primaria id_depto en DEPARTAMENTO

Restricciones del Modelo Relacional:

1. Clave Primaria: Identifica únicamente cada tupla. Ejemplo: ID_Estudiante en la tabla ESTUDIANTE

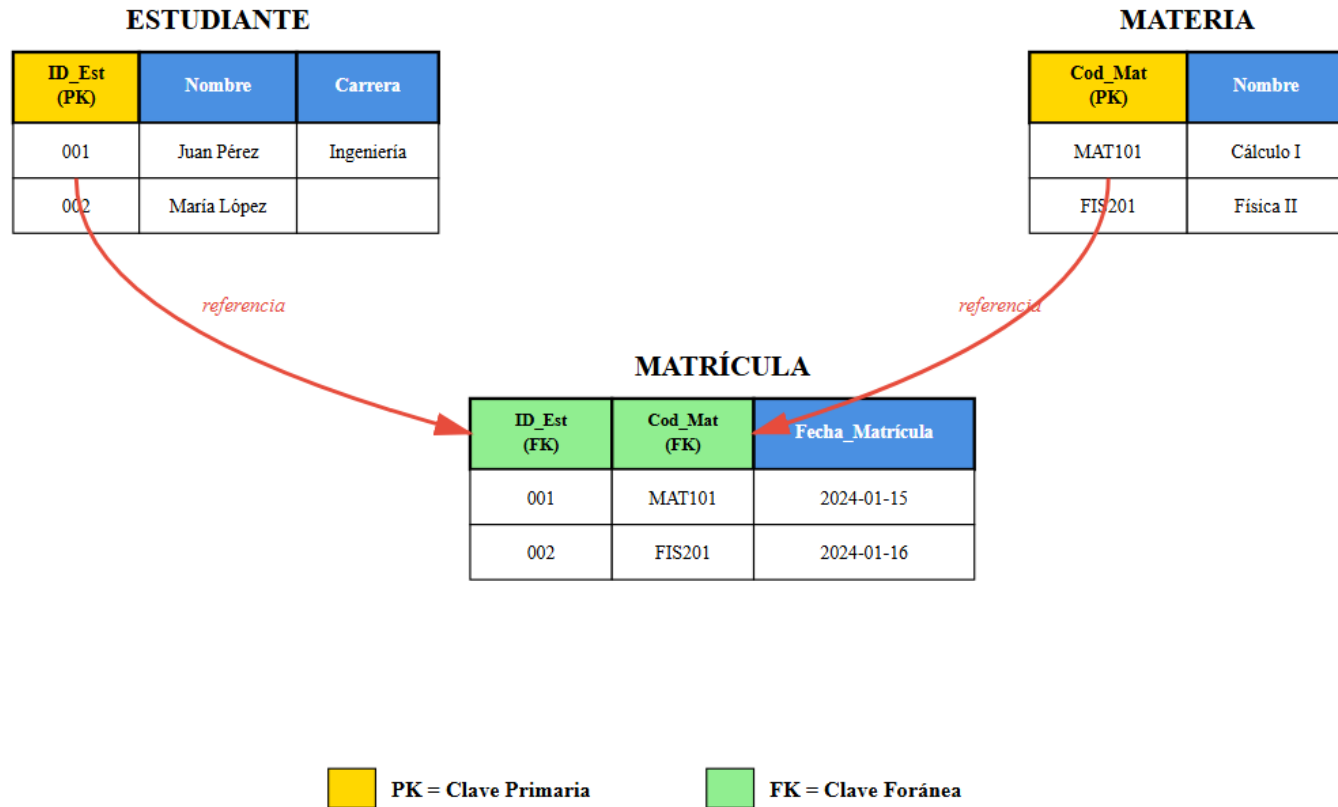
- No puede ser NULL
- Debe ser única para cada fila

2. Clave Foránea: Referencia la clave primaria de otra tabla. Ejemplo: En tabla MATRÍCULA, ID_Estudiante referencia a ESTUDIANTE

- Debe referenciar un valor existente
- Mantiene integridad referencial

[FIGURA 1.12: Diagrama de integridad referencial mostrando ESTUDIANTE, MATRÍCULA y MATERIA con claves primarias y foráneas]

FIGURA 1.12: Integridad Referencial - Sistema Académico



Las FK en MATRÍCULA referencian las PK en ESTUDIANTE y MATERIA, manteniendo la integridad referencial

Ejemplo de integridad referencial:

- Tabla ESTUDIANTE: ID_Estudiante (PK)
- Tabla MATRÍCULA: ID_Estudiante (FK), Código_Materia (FK)
- Tabla MATERIA: Código_Materia (PK)

3. Integridad de Entidad: La clave primaria no puede ser NULL

4. Integridad Referencial: Las claves foráneas deben referenciar valores existentes

Modelo Físico de Datos

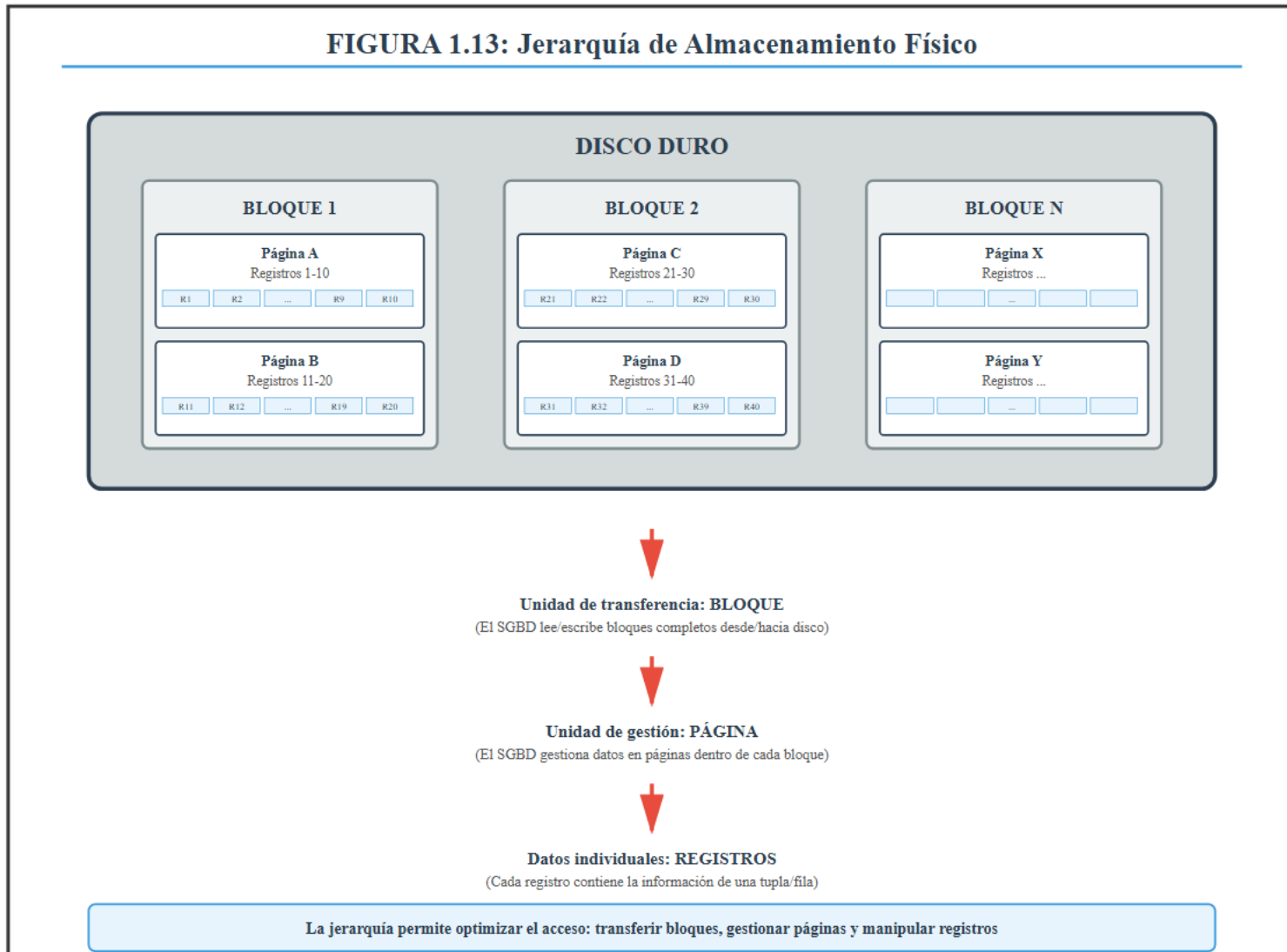
El **Modelo Físico** define cómo se almacenan físicamente los datos en el hardware, dependiendo del SGBD específico.

Niveles de almacenamiento físico:

El almacenamiento se organiza en jerarquías:

1. **Disco Duro:** Almacenamiento permanente
2. **Bloques:** Unidades de transferencia entre disco y memoria
3. **Páginas:** Unidades de gestión de memoria dentro de bloques
4. **Registros:** Tuplas individuales almacenadas en páginas

[FIGURA 1.13: Diagrama de jerarquía de almacenamiento mostrando disco → bloques → páginas → registros]



Aspectos del Modelo Físico:

1. Tipos de datos específicos del SGBD:

Tabla comparativa de tipos de datos entre SGBD:

Tipo de Dato	PostgreSQL	MySQL	Oracle	SQL Server
Entero	INTEGER	INT	NUMBER(10)	INT
Auto-incremento	SERIAL	AUTO_INCREMENT	SEQUENCE	IDENTITY
Texto variable	VARCHAR	VARCHAR	VARCHAR2	NVARCHAR
Texto grande	TEXT	TEXT	CLOB	TEXT
Fecha	DATE	DATE	DATE	DATE
Fecha/Hora	TIMESTAMP	DATETIME	TIMESTAMP	DATETIME
Decimal	DECIMAL	DECIMAL	NUMBER	DECIMAL

2. Estructuras de índices:

Índice B-Tree (estructura más común):

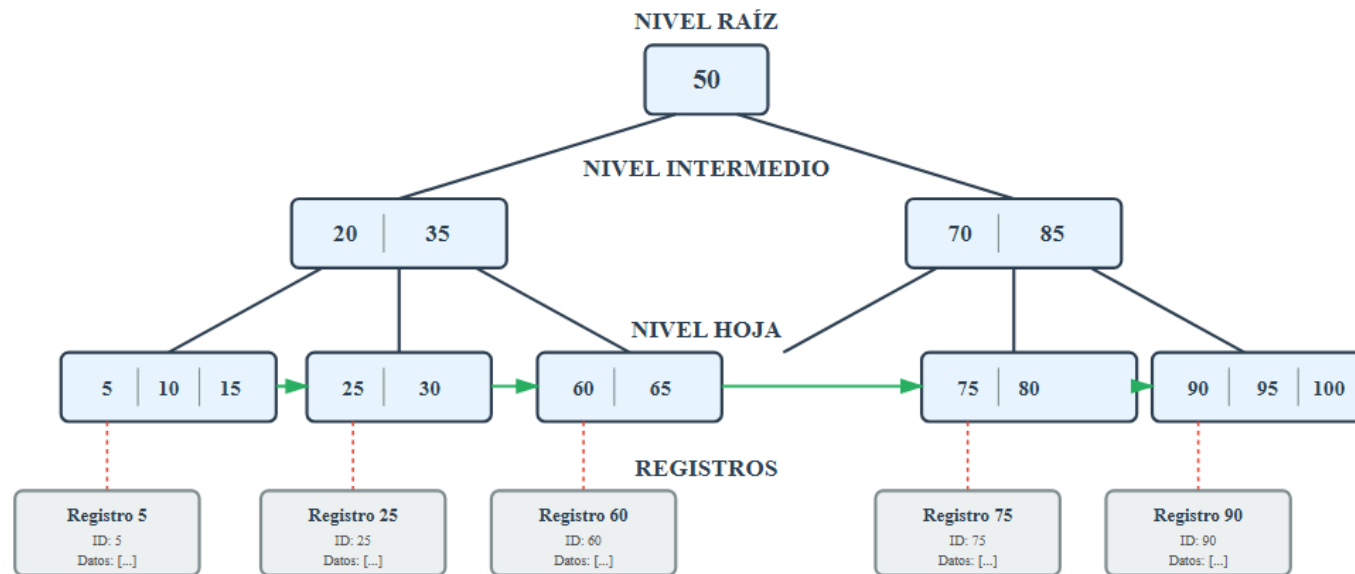
Los índices B-Tree organizan los datos en niveles jerárquicos:

- **Nivel Raíz:** Contiene valores guía para navegar

- **Niveles Intermedios:** Dirigen la búsqueda hacia hojas
- **Nivel Hoja:** Contiene punteros a los registros reales

[FIGURA 1.14: Estructura de índice B-Tree mostrando tres niveles con nodos raíz, intermedios y hojas]

FIGURA 1.14: Estructura de Índice B-Tree



Características del B-Tree

1. Estructura Balanceada:

- Todos los nodos hoja están al mismo nivel
- Garantiza tiempos de búsqueda predecibles: $O(\log n)$

2. Nodos con Múltiples Claves:

- Cada nodo puede contener múltiples valores ordenados
- Reduce la altura del árbol y mejora el rendimiento

3. Navegación Eficiente:

- Los valores en los nodos guían la búsqueda

4. Enlaces Horizontales:

- Nodos hoja enlazados (líneas verdes)
- Facilita recorridos secuenciales

5. Punteros a Registros:

- Hojas apuntan a registros reales
- Acceso directo a datos (líneas rojas)

6. Uso en SGBD:

- Índices primarios y secundarios

Ejemplo de búsqueda (valor 60):

Raíz (50) → $60 > 50$ → Derecha (70|85) → $60 < 70$ → Izquierda (60|65) → Encontrado → Registro 60

Ejemplo de navegación: Para buscar el valor 25:

1. Desde raíz (50): ir a la izquierda
2. En nivel intermedio (20, 35): ir entre 20 y 35
3. En hoja: encontrar puntero al registro con valor 25

3. Estrategias de particionamiento:

Particionamiento horizontal por rango:

Los datos se dividen por criterios específicos para mejorar el rendimiento:

Tabla VENTAS original: Contiene todos los registros **Partición 2023:** Solo registros del año 2023

Partición 2024: Solo registros del año 2024

[FIGURA 1.15: Diagrama mostrando tabla VENTAS dividida en particiones por año]

FIGURA 1.15: Particionamiento Horizontal por Rango

TABLA VENTAS (Completa)			
ID	Fecha	Ciente	Monto
1	2023-01-15	Juan	1500.00
2	2023-06-20	Maria	2300.00
3	2024-02-10	Carlos	890.00
4	2024-08-05	Ana	3200.00

↓
PARTICIONAMIENTO POR AÑO
↓

PARTICIÓN 2023		
ID	Fecha	Monto
1	01-15	1500
2	06-20	2300

Almacenamiento físico separado

PARTICIÓN 2024		
ID	Fecha	Monto
3	02-10	890
4	08-05	3200

Almacenamiento físico separado

VENTAJAS DEL PARTICIONAMIENTO	
✓ Consultas más rápidas Solo busca en la partición relevante (ej: ventas 2024)	✓ Mantenimiento más eficiente Se puede mantener/optimar una partición sin afectar otras
✓ Backup por particiones Respaldo incremental de particiones nuevas solamente	✓ Eliminación eficiente de datos antiguos Se puede eliminar una partición completa (ej: año 2020)

Ventajas del particionamiento:

- Consultas más rápidas (solo busca en partición relevante)
- Mantenimiento más eficiente

- Backup por particiones

4. Configuraciones de almacenamiento:

Clustering (Agrupación física):

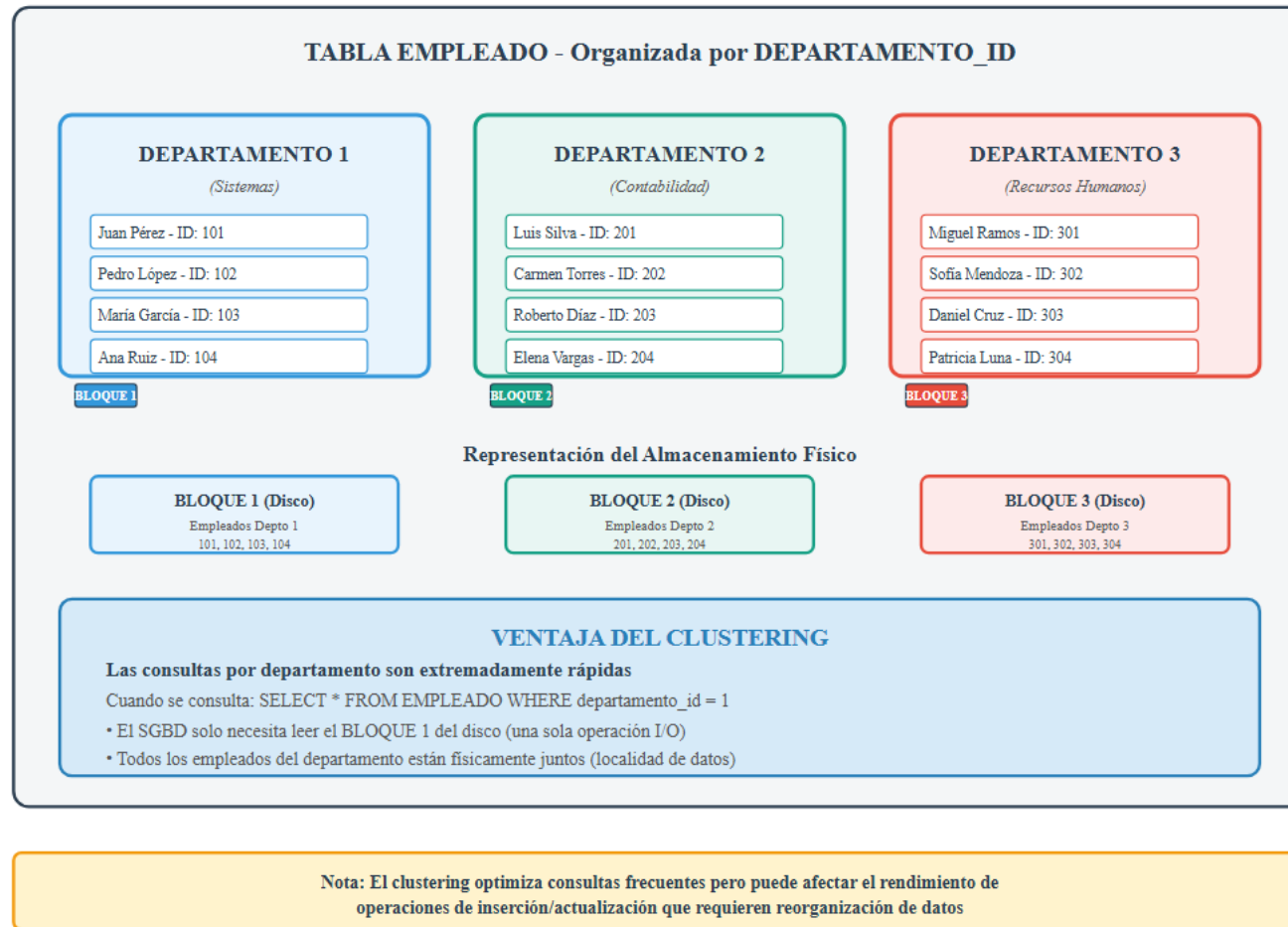
Los registros se organizan físicamente según criterios de acceso frecuente:

Tabla EMPLEADO agrupada por DEPARTAMENTO_ID:

- Todos los empleados del Departamento 1 se almacenan juntos
- Todos los empleados del Departamento 2 se almacenan juntos
- Las consultas por departamento son muy rápidas

[FIGURA 1.16: Representación de clustering mostrando registros agrupados por departamento]

FIGURA 1.16: Clustering Físico por Departamento



Compresión de datos:

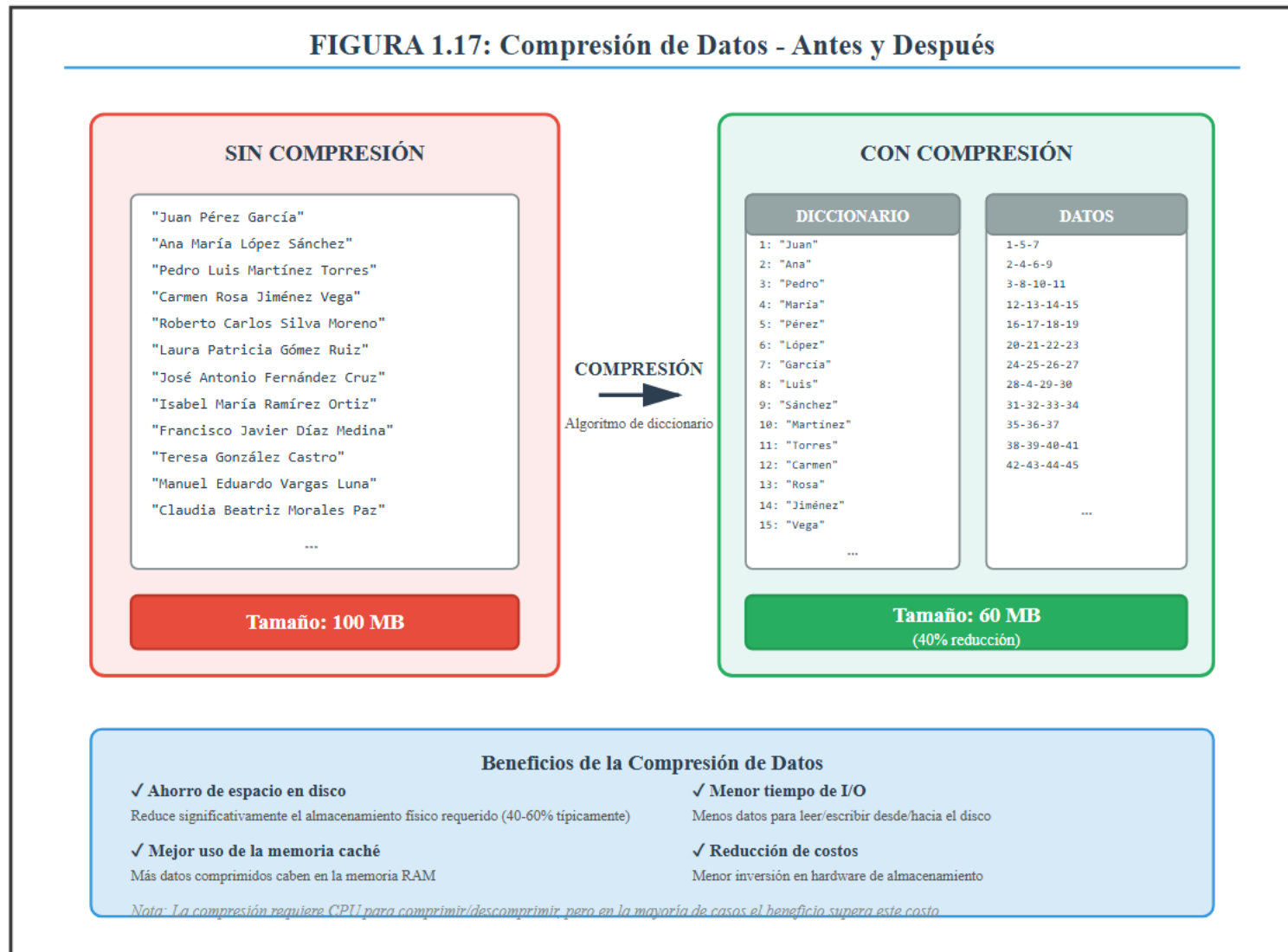
Sin compresión: Los datos se almacenan como texto completo

- Ejemplo: "Juan Pérez García", "Ana María López Sánchez"
- Espacio ocupado: 100 MB

Con compresión por diccionario:

- Diccionario: 1="Juan", 2="Ana", 3="Pedro", 4="María", 5="Pérez"
- Datos comprimidos: 1-5-8, 2-4-7-9, 3-6-10
- Espacio ocupado: 60 MB (40% de reducción)

[FIGURA 1.17: Comparación visual del espacio ocupado sin compresión vs con compresión]



Beneficios de la Compresión de Datos

✓ **Ahorro de espacio en disco**
Reduce significativamente el almacenamiento físico requerido (40-60% típicamente)

✓ **Mejor uso de la memoria caché**
Más datos comprimidos caben en la memoria RAM

✓ **Menor tiempo de I/O**
Menos datos para leer/escribir desde/hacia el disco

✓ **Reducción de costos**
Menor inversión en hardware de almacenamiento

Nota: La compresión requiere CPU para comprimir/descomprimir, pero en la mayoría de casos el beneficio supera este costo

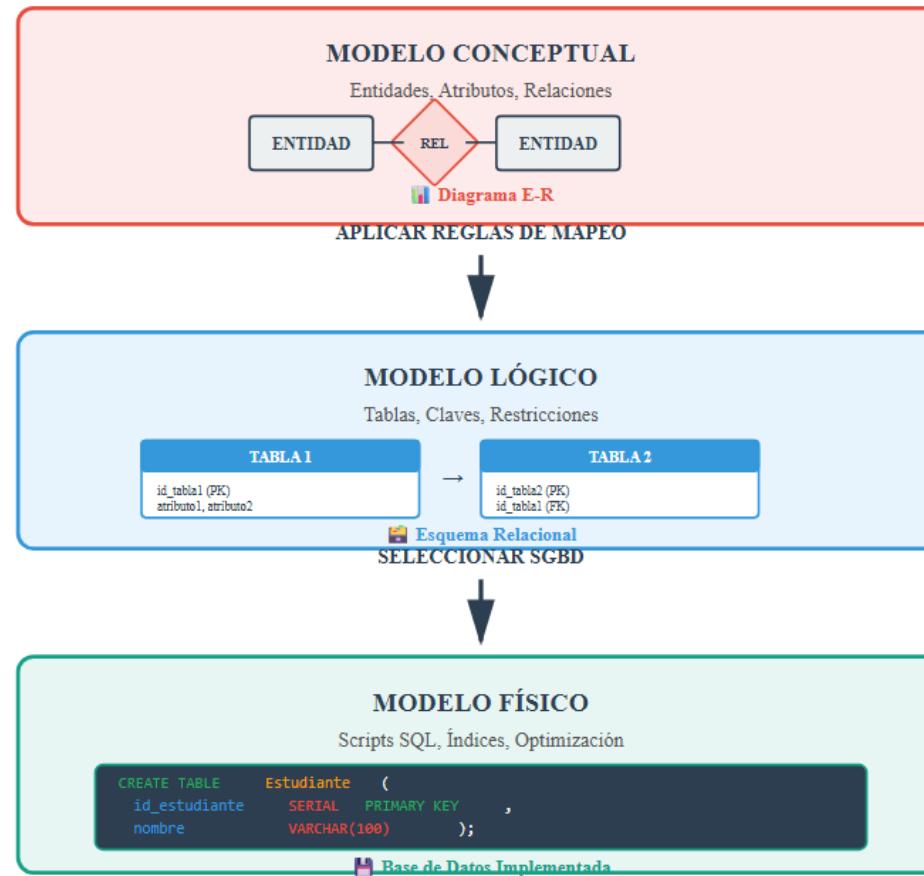
1.4 Mapeo del esquema conceptual al esquema relacional.

Proceso de Transformación E-R → Relacional

Metodología de transformación:

El proceso de conversión del modelo conceptual al lógico. Sigue reglas sistemáticas para asegurar que se preserve toda la información y las restricciones del modelo original.

FIGURA 1.18: Proceso de Transformación E-R → Relacional



Flujo completo: Diseño Conceptual (E-R) → Diseño Lógico (Tablas) → Implementación Física (SQL)

Cada nivel agrega detalles técnicos específicos mientras mantiene la integridad del diseño original

Etapas del proceso:

1. Análisis del modelo E-R: Identificar entidades, relaciones y cardinalidades
2. Aplicación de reglas de mapeo: Transformar según el tipo de componente
3. Validación del esquema: Verificar integridad y completitud
4. Optimización: Ajustar para eficiencia Regla

1: Mapeo de Entidades Regulares

Principio: Cada entidad se convierte en una tabla

Proceso de transformación:

- El nombre de la entidad se convierte en nombre de tabla
- Cada atributo se convierte en columna
- La clave primaria de la entidad se convierte en clave primaria de la tabla

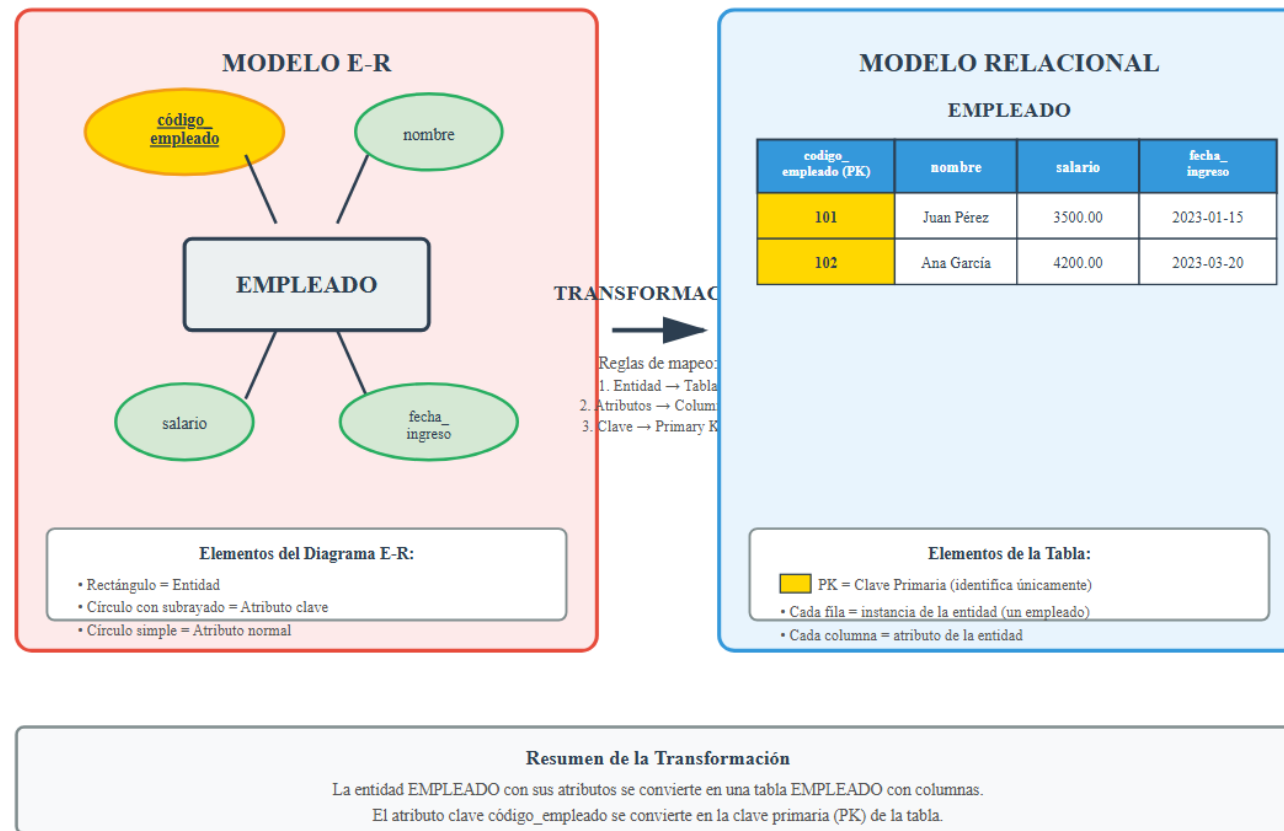
Ejemplo de transformación:

Entidad E-R: EMPLEADO con atributos (código_empleado, nombre, salario, fecha_ingreso)

Tabla relacional resultante:

- Nombre de tabla: EMPLEADO
- Columnas: codigo_empleado (PK), nombre, salario, fecha_ingreso
- Tipos de datos asignados según SGBD específico

FIGURA 1.19: Transformación de Entidad EMPLEADO a Tabla



Regla 2: Mapeo de Relaciones 1:N

Principio: La clave primaria del lado "1" se agrega como clave foránea al lado "N"

Proceso de transformación:

- Se crean dos tablas (una para cada entidad)
- La tabla del lado "N" recibe una columna adicional

- Esta columna es clave foránea que referencia la clave primaria del lado "1"

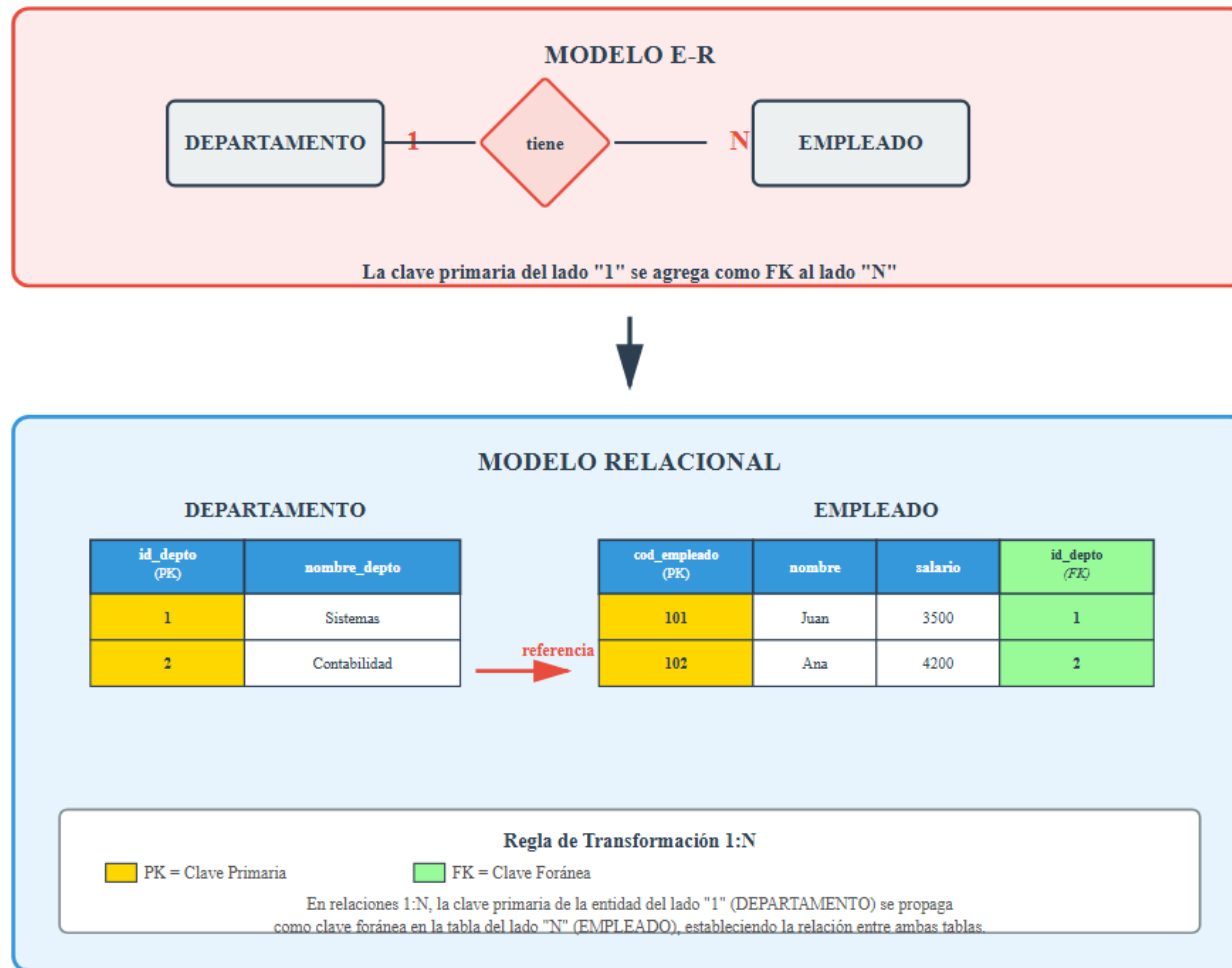
Ejemplo de transformación:

Relación E-R: DEPARTAMENTO (1) ---- tiene ----- (N) EMPLEADO

Tablas relacionales resultantes:

1. DEPARTAMENTO: id_departamento (PK), nombre_departamento
2. EMPLEADO: codigo_empleado (PK), nombre, salario, id_departamento (FK)

FIGURA 1.20: Transformación de Relación 1:N



Regla 3: Mapeo de Relaciones N:M

Principio: Se crea una nueva tabla con las claves primarias de ambas entidades

Proceso de transformación:

- Se crean tablas para ambas entidades originales
- Se crea una tabla adicional para la relación

- La nueva tabla contiene las claves primarias de ambas entidades como claves foráneas
- La clave primaria de la nueva tabla es compuesta

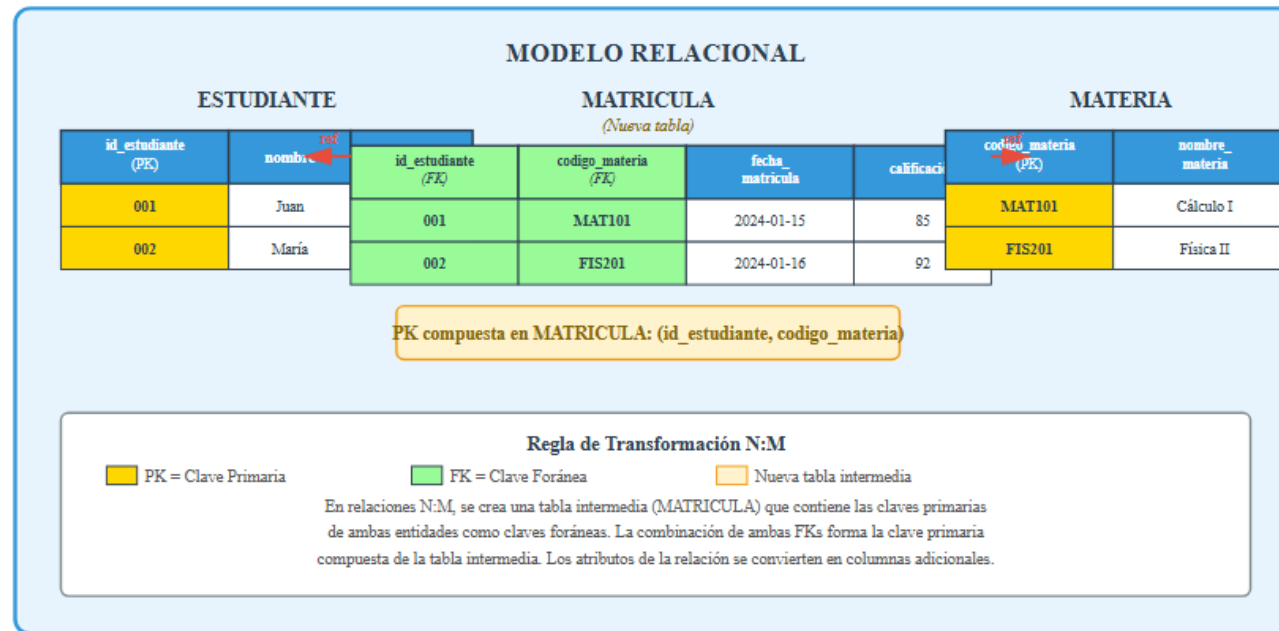
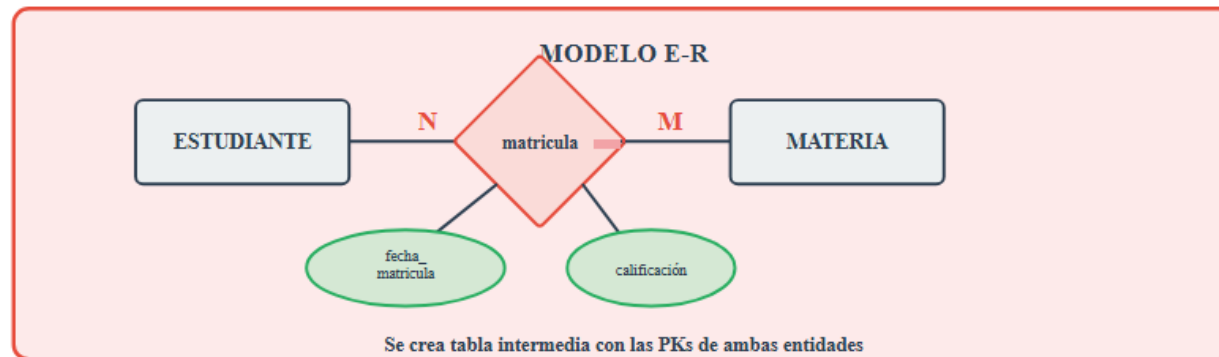
Ejemplo de transformación:

Relación E-R: ESTUDIANTE (N) ---- matricula ----- (M) MATERIA

Tablas relacionales resultantes:

1. ESTUDIANTE: id_estudiante (PK), nombre, carrera
2. MATERIA: codigo_materia (PK), nombre_materia, creditos
3. MATRICULA: id_estudiante (FK), codigo_materia (FK), fecha_matricula, calificacion
 - Clave primaria compuesta: (id_estudiante, codigo_materia)

FIGURA 1.21: Transformación de Relación N:M



Regla 4: Mapeo de Atributos Multivaluados

Principio: Se crea una tabla separada para el atributo multivaluado

Proceso de transformación:

- Se mantiene la tabla original sin el atributo multivaluado
- Se crea una nueva tabla para almacenar los múltiples valores
- La nueva tabla incluye la clave primaria de la entidad original
- La clave primaria de la nueva tabla es compuesta

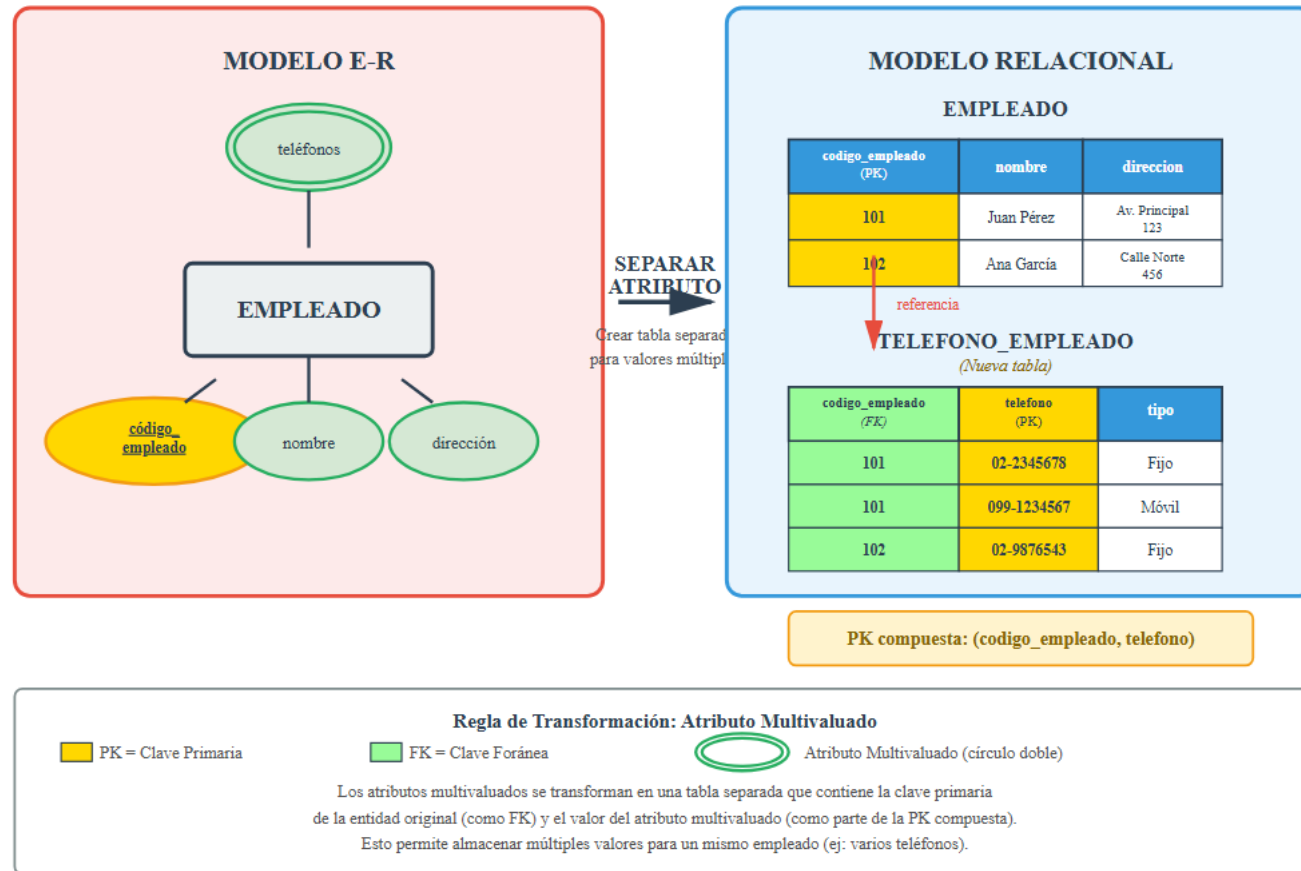
Ejemplo de transformación:

Entidad E-R: EMPLEADO con atributo multivaluado "teléfonos"

Tablas relacionales resultantes:

1. EMPLEADO: codigo_empleado (PK), nombre, direccion
2. TELEFONO_EMPLEADO: codigo_empleado (FK), telefono
 - Clave primaria compuesta: (codigo_empleado, telefono)

FIGURA 1.22: Transformación de Atributo Multivaluado



Ejemplo Completo de Mapeo

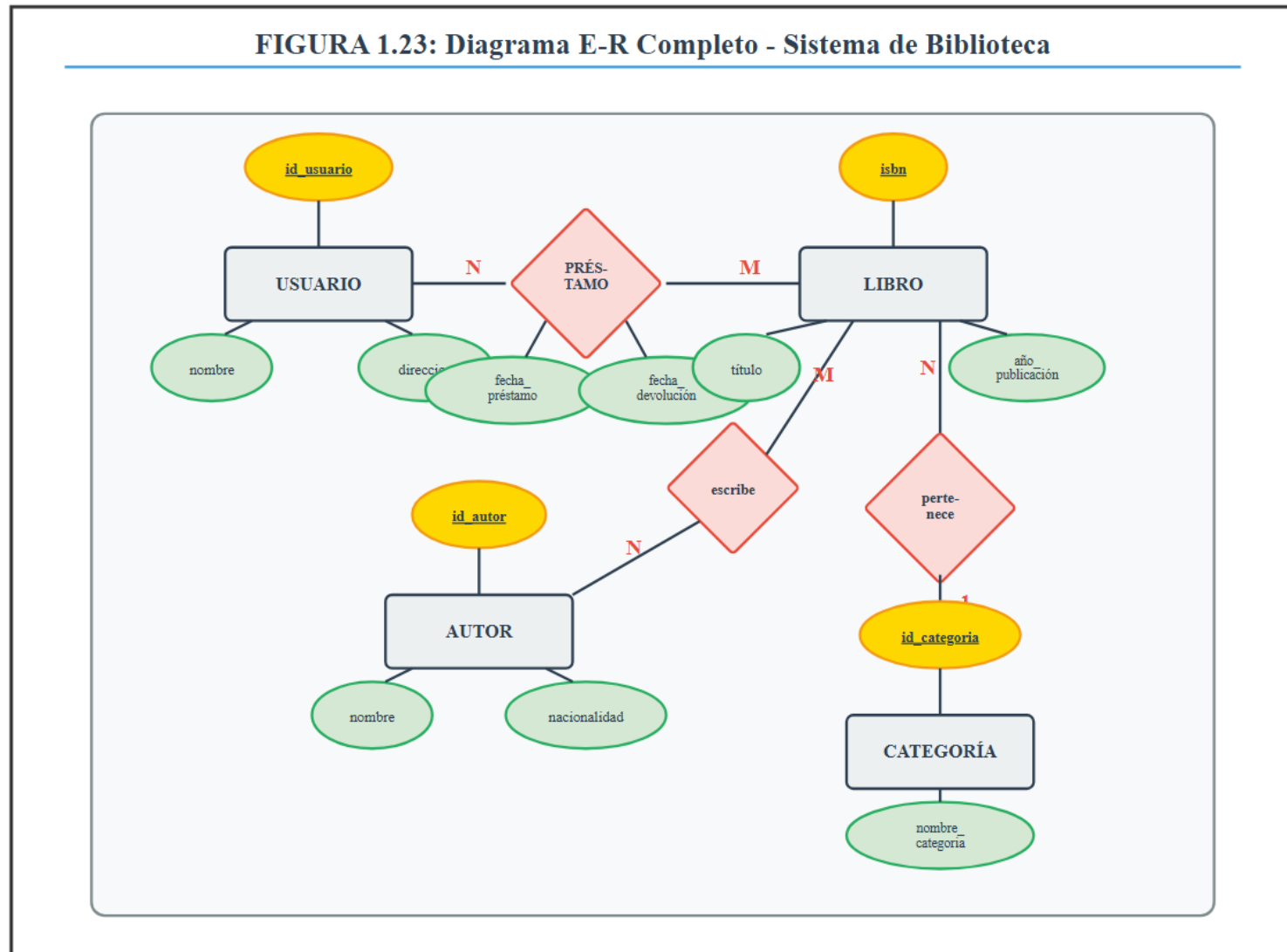
Modelo E-R: Sistema de Biblioteca

Componentes del modelo original:

- Entidades: USUARIO, LIBRO, AUTOR, CATEGORÍA

- Relaciones:
 - USUARIO solicita PRÉSTAMO de LIBRO (N:M con atributos)
 - AUTOR escribe LIBRO (N:M)
 - LIBRO pertenece CATEGORÍA (N:1)

[FIGURA 1.23: Diagrama E-R completo del sistema de biblioteca]



Esquema Relacional resultante:

Tablas para entidades:

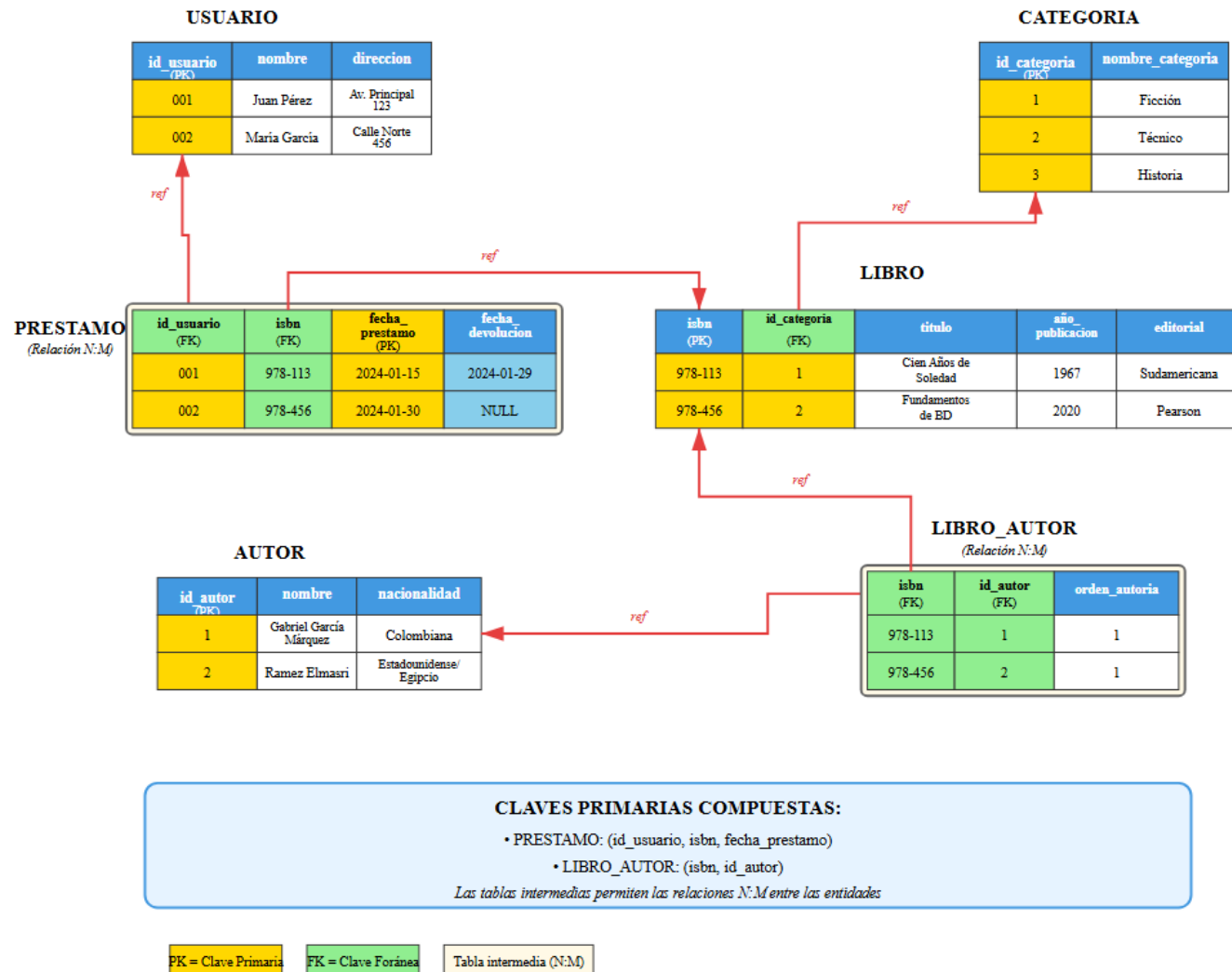
1. USUARIO: id_usuario (PK), nombre, direccion, telefono
2. LIBRO: isbn (PK), titulo, año_publicacion, editorial, id_categoria (FK)

3. AUTOR: id_autor (PK), nombre, nacionalidad
4. CATEGORIA: id_categoria (PK), nombre_categoria

Tablas para relaciones N:M:

5. PRESTAMO: id_usuario (FK), isbn (FK), fecha_prestamo, fecha_devolucion
 - Clave primaria: (id_usuario, isbn, fecha_prestamo)
6. LIBRO_AUTOR: isbn (FK), id_autor (FK)
 - Clave primaria: (isbn, id_autor)

FIGURA 1.24: Esquema Relacional Completo - Sistema de Biblioteca



Ventajas del esquema resultante:

- Elimina redundancia de datos
- Mantiene integridad referencial
- Permite consultas eficientes
- Facilita mantenimiento de datos

NIVELES DE ABSTRACCIÓN EN EL DISEÑO DE BASES DE DATOS

El diseño de bases de datos se realiza en tres niveles principales, cada uno con objetivos y características específicas:

Pirámide de los Tres Niveles de Abstracción

NIVEL CONCEPTUAL (E-R)

Características:

- Independiente de tecnología
- Enfocado en QUÉ datos almacenar
 - Orientado al usuario
- Comunicación con usuarios finales

Ejemplo: Entidad ESTUDIANTE ← relación "matricula" → Entidad MATERIA



Transformación mediante reglas de mapeo

NIVEL LÓGICO (RELACIONAL)

Características:

- Independiente del SGBD específico
- Definición de estructuras de datos
 - Especifica claves PK y FK
 - Base para implementación

Ejemplo: Tablas ESTUDIANTE, MATRICULA, MATERIA con claves definidas



Implementación en SGBD específico

NIVEL FÍSICO (IMPLEMENTACIÓN)

Características:

- Dependiente del SGBD
- Define almacenamiento físico
- Índices y optimizaciones
- Scripts SQL ejecutables

Ejemplo: CREATE TABLE con tipos específicos (VARCHAR, INT, SERIAL)

NIVEL CONCEPTUAL (Modelo E-R)

Características:

- Independiente de la tecnología
- Enfocado en qué datos almacenar

- Usa notación estándar E-R
- Fácil comunicación con usuarios finales

Ejemplo representativo:

Entidad ESTUDIANTE conectada con relación "matricula" (N:M) a entidad MATERIA, mostrando atributos como nombre, fecha_matricula, etc.

NIVEL LÓGICO (Modelo Relacional)

Características:

- Independiente del SGBD específico
- Define estructura de tablas y relaciones
- Especifica claves primarias y foráneas
- Base para la implementación

Ejemplo representativo:

Tres tablas (ESTUDIANTE, MATRICULA, MATERIA) con claves primarias y foráneas claramente definidas.

NIVEL FÍSICO (Implementación)

Características:

- Dependiente del SGBD específico
- Define tipos de datos concretos
- Incluye índices y optimizaciones
- Scripts SQL ejecutables

Ejemplo representativo:

Código SQL CREATE TABLE con tipos de datos específicos de PostgreSQL/MySQL/Oracle.

FLUJO COMPLETO DE DISEÑO DE BASE DE DATOS

Metodología paso a paso:

Flujo Completo de Diseño de Base de Datos

1. ANÁLISIS DE REQUERIMIENTOS

Entrada:
Descripción del sistema
Actividades:
Entrevistas, análisis
Salida:
Lista de requerimientos
Un estudiante se matricula en materias



2. MODELO CONCEPTUAL (E-R)

Entrada:
Lista de requerimientos
Actividades:
Identificar entidades, relaciones
Salida:
Diagrama E-R dibujado
Entidades: ESTUDIANTE, MATERIA, PROFESOR
Relaciones: MATRICULA, IMPARTE



3. MODELO LÓGICO (RELACIONAL)

Entrada:
Diagrama E-R
Actividades:
Aplicar reglas de transformación
Salida:
Esquema relacional normalizado
Tablas con PK y FK
Restricciones de integridad
Normalización aplicada



4. MODELO FÍSICO (IMPLEMENTACIÓN)

Entrada:
Esquema relacional
Actividades:
Seleccionar SGBD, definir tipos
Salida:
Scripts SQL específicos
Scripts CREATE TABLE
Definición de índices
Configuración de rendimiento



5. BASE DE DATOS FUNCIONANDO

Entrada:
Scripts SQL
Actividades:
Instalación, configurar, pruebas
Salida:
BD instalada y configurada
Aplicaciones conectadas
Usuarios realizando operaciones

Etapa 1: Análisis de Requerimientos

- Entrada: Descripción del sistema en lenguaje natural
- Actividades: Entrevistas, análisis de documentos, identificación de procesos
- Salida: Lista de requerimientos funcionales y de datos

Ejemplo: "Los estudiantes se matriculan en materias. Cada materia tiene prerrequisitos y es impartida por un profesor..."

Etapa 2: Modelo Conceptual (E-R)

- Entrada: Requerimientos de datos
- Actividades: Identificar entidades, atributos, relaciones y cardinalidades
- Salida: Diagrama Entidad-Relación validado

Componentes resultantes:

- Entidades: ESTUDIANTE, MATERIA, PROFESOR
- Relaciones: MATRICULA, IMPARTE, PRERREQUISITO
- Cardinalidades: N:M, 1:N, N:M

Etapa 3: Modelo Lógico (Relacional)

- Entrada: Diagrama E-R
- Actividades: Aplicar reglas de transformación
- Salida: Esquema relacional normalizado

Componentes resultantes:

- Tablas con claves primarias y foráneas
- Restricciones de integridad
- Normalización aplicada

Etapa 4: Modelo Físico (Implementación)

- Entrada: Esquema relacional
- Actividades: Seleccionar SGBD, definir tipos de datos, crear índices
- Salida: Scripts SQL específicos del SGBD

Componentes resultantes:

- Scripts CREATE TABLE
- Definición de índices
- Configuraciones de rendimiento

Etapa 5: Base de Datos Funcionando

- Entrada: Scripts SQL
- Actividades: Instalación, pruebas, conexión de aplicaciones
- Salida: Sistema operativo con usuarios

Componentes resultantes:

- Base de datos instalada y configurada
- Aplicaciones conectadas
- Usuarios realizando operaciones

HERRAMIENTAS RECOMENDADAS POR NIVEL

Para Modelo Conceptual:

- PowerDesigner: Herramienta profesional para modelado E-R
- Draw.io: Alternativa gratuita basada en web
- Lucidchart: Herramienta colaborativa en línea

Para Modelo Lógico:

- PowerDesigner: Transformación automática E-R → Relacional
- MySQL Workbench: Específico para MySQL con reverse engineering
- Oracle SQL Developer Data Modeler: Para entornos Oracle

Para Modelo Físico:

- Scripts SQL nativos del SGBD seleccionado
- Herramientas de administración: pgAdmin, MySQL Workbench, SQL Server Management Studio
- Herramientas de migración: Para cambios de SGBD ##

CRITERIOS DE CALIDAD EN CADA NIVEL

Calidad del Modelo Conceptual:

- Completitud: ¿Captura todos los requerimientos?
- Consistencia: ¿Las cardinalidades son coherentes?
- Simplicidad: ¿Es comprensible para los usuarios?
- Flexibilidad: ¿Permite cambios futuros?

Calidad del Modelo Lógico:

- Normalización: ¿Está en 3FN como mínimo?
- Integridad: ¿Las restricciones están bien definidas?
- Eficiencia: ¿Las consultas serán eficientes?
- Mantenibilidad: ¿Es fácil de modificar?

Calidad del Modelo Físico:

- Rendimiento: ¿Los índices están optimizados?
- Escalabilidad: ¿Soportará el crecimiento?
- Seguridad: ¿Los accesos están controlados?
- Disponibilidad: ¿Tiene mecanismos de backup?

ERRORES COMUNES Y CÓMO EVITARLOS

En el Nivel Conceptual:

- Error: Confundir entidades con atributos
- Solución: Preguntarse si el concepto puede tener atributos propios
- Error: Cardinalidades incorrectas
- Solución: Validar con ejemplos reales del dominio

En el Nivel Lógico:

- Error: No normalizar apropiadamente
- Solución: Aplicar sistemáticamente las formas normales
- Error: Claves foráneas mal definidas
- Solución: Verificar que referencien claves primarias existentes

En el Nivel Físico:

- Error: Tipos de datos inadecuados
- Solución: Considerar el rango de valores y rendimiento
- Error: Falta de índices en consultas frecuentes
- Solución: Analizar patrones de consulta antes de implementar

CASOS DE ESTUDIO

CASO DE ESTUDIO 1: SISTEMA DE GESTIÓN ACADÉMICA UNIVERSITARIA

Descripción del Sistema:

Una universidad necesita un sistema para gestionar la información académica. Los estudiantes se matriculan en carreras y toman materias. Cada materia es impartida por un profesor en un aula específica y tiene prerequisites. Los estudiantes reciben calificaciones en las materias que cursan.

CASO DE ESTUDIO 2: SISTEMA DE GESTIÓN HOSPITALARIA**Descripción del Sistema:**

Un hospital necesita gestionar pacientes, médicos, consultas, tratamientos y habitaciones. Los pacientes pueden tener múltiples consultas con diferentes médicos. Cada consulta puede resultar en uno o más tratamientos. Los pacientes pueden ser hospitalizados en habitaciones. Los médicos tienen especialidades y pueden trabajar en diferentes departamentos.

CASO DE ESTUDIO 3: SISTEMA DE GESTIÓN DE BIBLIOTECADIGITAL**Descripción del Sistema:**

Una biblioteca digital gestiona libros electrónicos, autores, usuarios, préstamos y reservas. Los libros pueden tener múltiples autores y pertenecer a varias categorías. Los usuarios pueden realizar préstamos y reservas. Existen diferentes tipos de usuarios con diferentes privilegios. La biblioteca también gestiona reseñas y calificaciones de los libros.

Comparación de SGBD para el Proyecto

Característica	Oracle	SQL Server	PostgreSQL	MySQL	Firebird
Tipo de Licencia	Comercial	Comercial	Open Source	Open Source	Open Source
Auto-incremento	SEQUENCE	IDENTITY	SERIAL	AUTO_INCREMENT	GENERATOR
Tipos de Texto	VARCHAR2, CLOB	NVARCHAR, TEXT	VARCHAR, TEXT	VARCHAR, TEXT	VARCHAR, BLOB
Fechas	DATE, TIMESTAMP	DATETIME, DATE	DATE, TIMESTAMP	DATE, DATETIME	DATE, TIMESTAMP
Restricciones	CHECK, UNIQUE	CHECK, UNIQUE	CHECK, UNIQUE	CHECK, UNIQUE	CHECK, UNIQUE
Mejor para	Empresas grandes	Entornos Microsoft	Desarrollo ágil	Aplicaciones web	Aplicaciones embebidas

Recursos Adicionales

- **Herramientas de modelado:** Draw.io, Lucidchart, MySQL Workbench, Oracle SQL Developer Data Modeler
- **SGBD considerados:** Oracle, SQL Server, PostgreSQL, MySQL, Firebird
- **Lecturas complementarias:** "Fundamentos de Sistemas de Bases de Datos" - Elmasri & Navathe