

Bases de datos

Introducción



¿Qué es una BD?

- Una **base de datos (BD)** contiene conjunto organizado de información.
 - P.ej. de una empresa o institución pública.
 - Generalmente se trata de datos estructurados e interrelacionados.
 - P.ej. Datos sobre personas, vehículos y sus relaciones. Se tienen *autos*, *personas*, y relación *dueño de*.
 - Se gestionan mediante **sistemas de gestión de bases de datos** (SGBD)
- Las BD se usan ampliamente:
 - Empresas, instituciones públicas, organizaciones educativas, etc.

Aplicaciones de Bases de Datos

- **Las aplicaciones de BD permiten:**
 - **Navegar y consultar** las informaciones (búsquedas, filtros, reportes)
 - **Modificar** la información (agregar, eliminar, editar registros)
 - **Controlar el acceso** y garantizar la seguridad de la información (gestión de usuarios, permisos).
- **Aplicación web de BD:**
 - El **usuario accede** a la aplicación desde un navegador web (browser) que contiene la **interfaz del usuario** y envía solicitudes.
 - El **servidor web procesa solicitudes** (pasando consultas al SGBD) y **genera respuestas** (a partir de resultados enviados por el SGBD).
 - El **servidor web interactúa con el SGBD** para **almacenar, recuperar, o modificar** datos según las acciones del usuario.
 - **Ejemplo:** tienda online (ver zapatillas deportivas)

Esquemas e Instancias

- **Conceptos similares a** tipos (para esquemas) y variables (para instancias) en lenguajes de programación.
- **Esquema**: Describe la **estructura** de la BD.
 - **Idea**: describir **entidades**, **relaciones** entre ellas y **atributos** de las entidades.
 - **Ejemplo**: la BD de un *banco* consiste de **entidades** *clientes* y *cuentas* y la **relación** *tiene cuentas* entre ellos.
 - Los *clientes* tienen **atributos** *nombre* y *DNI*;
 - Las *cuentas* tienen **atributos** *número* y *saldo*.
 - *Tiene cuentas* es **relación** entre *clientes* y *cuentas*.

Esquemas e Instancias

- **Instancia:** **contenido actual** de la BD en un momento dado.
 - Corresponde a los **valores concretos almacenados**.
 - **Ejemplo:** para la idea de entidades, relaciones y atributos, de sistema bancario:
 - *Clientes:* Juan Pérez con DNI: 333 y Diego González con DNI: 444.
 - *Cuentas:* tiene las cuentas (número: 1111, saldo: \$ 1000) y (número: 2222, saldo: \$500).
 - *Tiene cuentas:* Juan Pérez tiene la cuenta 1111 y Diego González tiene la cuenta 2222.

Diseño de Bases de Datos

- **Diseñar una base de datos** es construir un **esquema** (que defina su estructura) y **especificar el comportamiento esperado**.
- **Comportamiento esperado** = **propiedades** que las instancias deben satisfacer.
 - A estas propiedades se las llama **restricciones de integridad**.
- **Ejemplos de restricciones de integridad**: para el esquema del sistema bancario:
 - El DNI no se repite en diferentes clientes.
 - Un cliente puede tener una o más cuentas.
- **En resumen:**

Diseño de BD = estructura + restricciones de integridad
- **¿por qué es necesario definir las restricciones de integridad?**
- **Para la validación de las instancias:**
 - Cada vez que se **modifica una instancia** (agregar, eliminar, actualizar datos),
 - se debe verificar que las restricciones de integridad se mantengan.

Diseño de Bases de Datos

- **Actividad de diseño: el almacenero y sus cuadernos**

- Usted es un **almacenero** que debe llevar registro de todo lo que sucede en el almacén:
 - Precio de los productos
 - Stock actual
 - Lista de proveedores
 - Compras realizadas
 - Decide llevar todo el registro en **cuadernos**.
- Tarea
 - Diseñe la organización de los cuadernos.
 - Determine qué información precisa registrar en cada uno (esquema).

Diseño de Bases de Datos

- **Actividad de diseño: el almacenero y sus cuadernos (continuación)**
 - Su diseño debe permitir responder:
 - ¿Qué productos vende un determinado proveedor?
 - ¿Cómo obtiene el stock de un producto?
 - ¿Cómo lleva el registro de las compras que le realizaron?
 - Además, reflexione sobre:
 - ¿Qué tan eficiente es su proceso?
 - ¿Tiene redundancia de datos innecesaria?

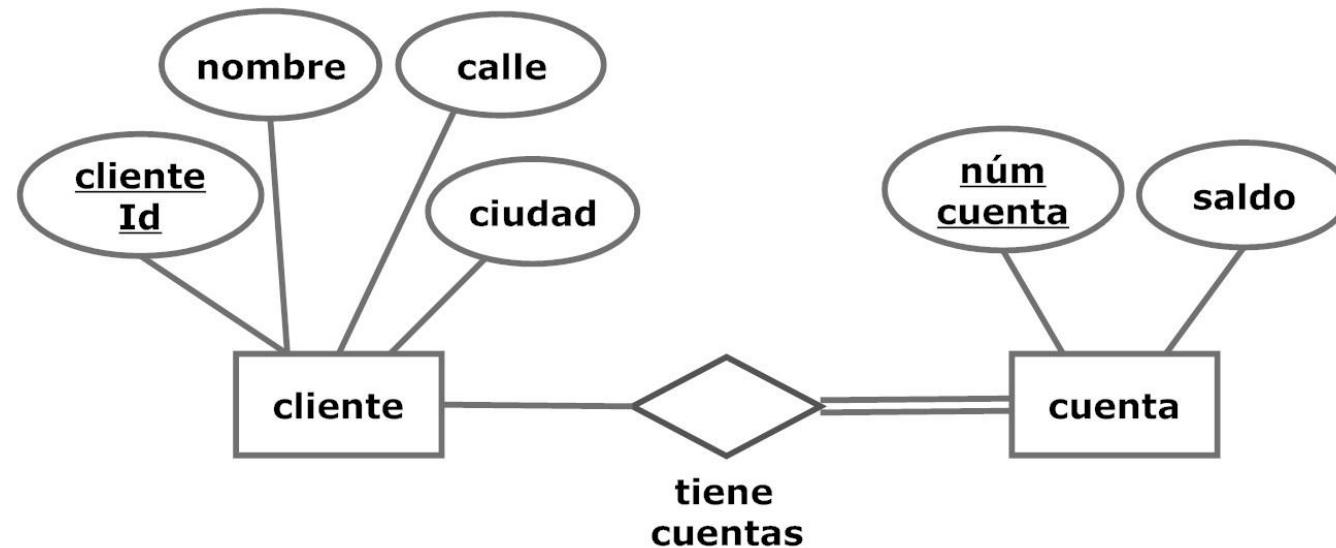
Diseño de Bases de Datos

- Estudiaremos como diseñar dos tipos distintos de esquemas de bases datos:
 - Diseño de modelos de entidad-relación
 - Diseño de modelos relacionales

Diseño de Entidad-Relación

- Se modela una la base de datos de una organización como una **colección de entidades y relaciones**.
 - **Entidad**: objeto en organización distinguible de otros objetos.
 - Una entidad **se describe** por medio de **atributos**.
 - Un **atributo** es una **propiedad** de una entidad;
 - esa propiedad tiene un **valor** en un **dominio**.
 - Un **dominio** es un conjunto de valores.
 - **Relación**: asociación entre entidades.
- **Ejemplo**: en sistema bancario tenemos entidades *cliente*, entidades *cuenta*, y relación *tiene cuentas*.
 - Toda *cuenta* es de algún *cliente*
- Un diseño de entidad-relación se **representa diagramáticamente**,
 - usado un **diagrama de entidad-relación**.

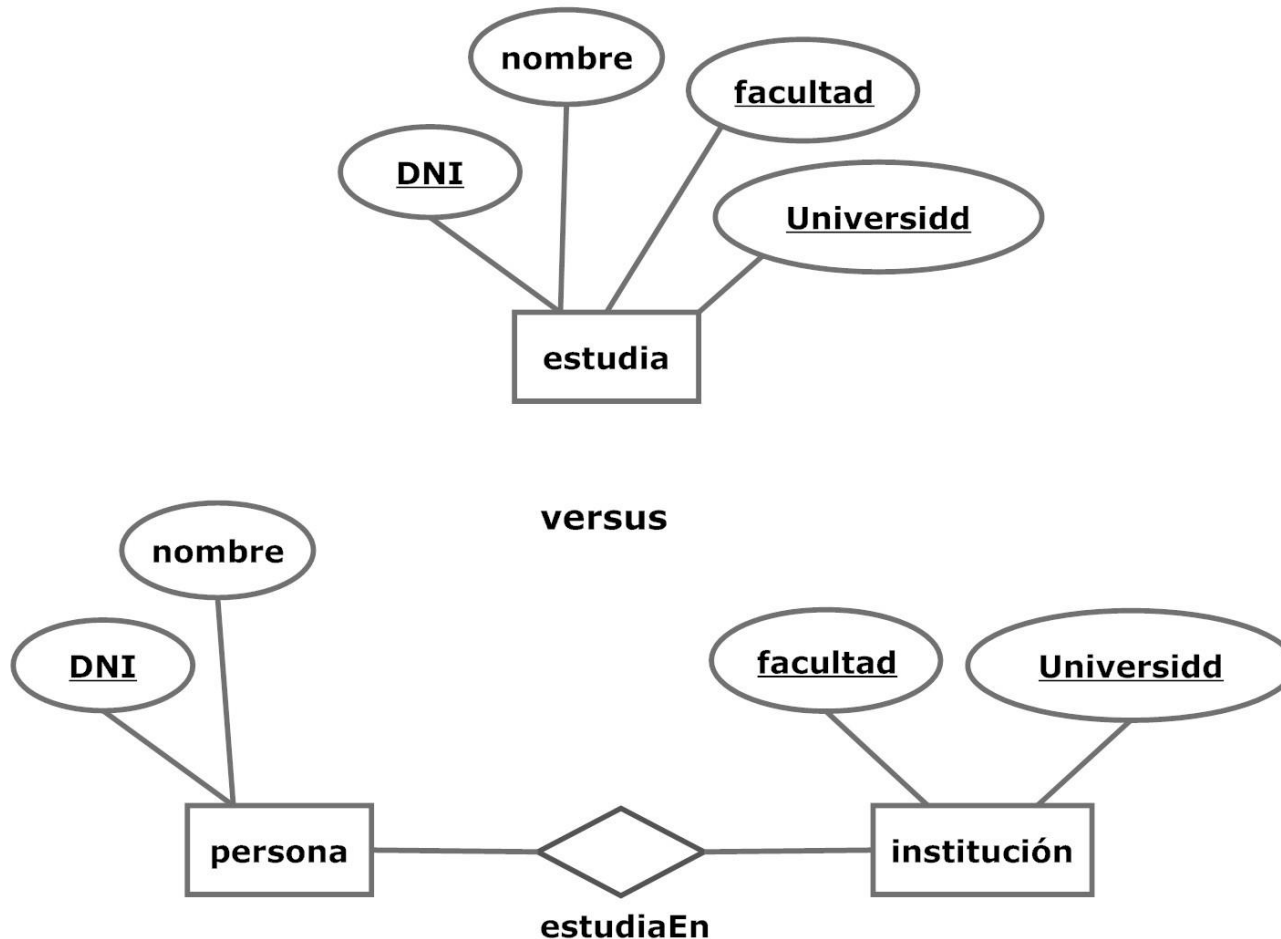
Ejemplo de diagrama de entidad-relación



- Los atributos se reflejan con **óvalos**,
- los conjuntos de entidades con **rectángulos**,
- y los conjuntos de relaciones con **rombos**.
- **Restricciones de integridad** en el diagrama de arriba.

Decisiones de Diseño

- El diseño ER **no tiene una única solución**.
 - Al modelar una BD, suelen surgir **múltiples alternativas de diseño**.
- **Ejemplo:**



Tipo de decisión de diseño:
Conjunto de entidades versus
Conjunto de relaciones

Diseño de Entidad-Relación

- **Naturaleza de las alternativas de diseño:**
 - No todos los diseños alternativos son equivalentes.
 - Algunos diseños pueden ser **inválidos**:
 - si presentan **propiedades indeseables**;
 - como redundancia, ambigüedad, o violaciones de integridad.
 - **Ejemplo**: en la figura anterior el diseño de arriba es inválido (¿por qué?)
 - Unos diseños son **preferibles**:
 - por su claridad, eficiencia, o capacidad de representar correctamente los requisitos.
 - **Ejemplo**: en la figura anterior el diseño de abajo es preferible (¿por qué?)

Diseño de Entidad-Relación

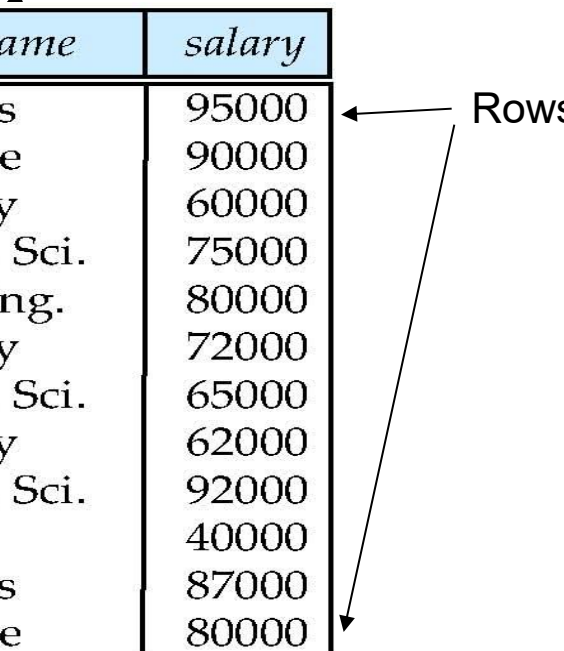
- Cada vez que se construye un modelo de entidad-relación, es necesario tomar **decisiones de diseño** entre alternativas.
 - Para esto **ayuda** considerar algunos **criterios**
 - para diseños inválidos y diseños preferibles.
 - y tener una clasificación de las decisiones de diseño típicas.
 - **Ejemplo**: ¿qué tipo de decisión hay en el ejemplo anterior?
- **Estudiaremos**:
 - Los **tipos de decisiones de diseño** que pueden surgir.
 - **Criterios** para decidir entre alternativas.
 - **Buenas prácticas** para guiar el proceso de diseño.
 - **Cosas a evitar** cuando se diseña (tipos de propiedades indeseables).

Diseño relacional

- Un **esquema relacional** es una **lista de nombres de atributos**
 - Esquemas relacionales tienen dos partes:
Nombre de esquema = lista de nombres de atributo.
- **Por ejemplo:**
 - Instructor = (ID, nombre, nombre de departamento, salario)
- Normalmente **en la práctica** se define un conjunto de esquemas relacionales,
 - para describir la **estructura** de la base de datos.
 - A esto se le llama **diseño de base de datos relacional**.
- Una **instancia** de un esquema relacional es una **tabla**
 - A esta tabla se le llama **relación**.
 - **Ejemplo:** tabla para instructor (página siguiente)
 - No confundir una tabla con una relación matemática
 - hay un orden entre las filas, a veces se permiten filas duplicadas.

Modelo relacional

- Los datos (instancias) son almacenados en **tablas**.
- Ejemplo de tabla en el modelo relacional



<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Las columnas representan **atributos** (o propiedades) para los elementos de la tabla (tuplas).

Esquemas e instancias relacionales

- Una **instancia** de un diseño de base de datos relacional
 - son las **tablas para los esquemas** de la misma.
- Ahora que vimos los conceptos de esquema relacional y de relación surge la pregunta:
 - ¿Cómo puedo juntar esos conceptos y qué se gana con hacerlo?
- **Notación:** $r(R)$ significa r es una relación (tabla) con esquema de relación R .
 - O sea, las columnas de r tienen como nombres los atributos de R .
- **Ejemplo:**
 - $persona(Persona)$
 - $Persona = (nombre, edad, DNI)$
- También lo puedo decir así:
 - $persona(nombre, edad, DNI)$

Atributos

- Una **consulta** sería como una pregunta
 - expresada en algún lenguaje,
 - expresada en términos de las tablas de la base de datos,
 - y da como resultado una tabla.
- **Ejemplo:**
 - persona(nombre, apellido, DNI) y la consulta:
 - listar el apellido de las personas que tienen DNI = 1111.
- El conjunto de valores permitidos para cada atributo se llama **dominio del atributo**.
 - Un dominio puede tener el **valor nulo**.
 - **Significado** del valor nulo.

Atributos

- **Exigencia:**

- Los valores de los atributos deben ser **atómicos**; esto es, indivisibles.
- Esto se refiere **al uso que se hace** de los atributos.
 - O sea que en las consultas o restricciones de integridad no vamos a dividir el valor de un atributo en partes.
 - Esto va a simplificar la descripción de consultas o restricciones de integridad.

- **Ejemplo:**

- En *persona(nombre, DNI)*
- No puedo hacer consultas como “listar los apellidos de personas que tienen DNI”.
 - DNI tiene un valor distinto de nulo.
 - El nombre es atómico y representa el nombre entero;
 - no podemos usar las partes del nombre en una consulta.
- Para que esta consulta pueda usarse necesitamos atributo *apellido*.

Superclaves

- **Propósito:** Ahora vamos a ver algunos tipos de **restricciones de integridad** muy comunes cuando se trabaja con esquemas relacionales.
- Si tenemos una relación de un esquema de relación
 - queremos poder **identificar las tuplas** de una relación,
 - por medio de los valores de **ciertos atributos**.
- **Ejemplo:**
 - *instructor(ID, name, dept name, salary)*
 - Se pueden identificar los instructores por medio de ID.
- A los atributos usados para identificar las tuplas se los llama superclaves.
- **Superclaves:** Sea $K \subseteq R$, R esquema de relación; K es una **superclave** de R si
 - los valores para K son suficientes para identificar una tupla única en cada posible relación $r(R)$.
- **Observar** que se habla de relaciones para el problema o situación del mundo real que está siendo considerando.
- **Ejemplo:**
 - *instructor(ID, name, dept name, salary)*
 - $\{ID\}$ e $\{ID, name\}$ son superclaves de *instructor*.

Claves candidatas y claves primarias

- En la práctica no nos interesan todas las superclaves.
 - En el ejemplo anterior no nos interesa la segunda.
 - Nos interesan las **superclaves minimales**.
 - **Minimal significa que:** para todo atributo de K si se lo quito a K dejo de tener una superclave.
 - **Ejemplo:** {ID} minimal de *instructor*.
 - **¡Atención!** No confundir superclave minimal con superclave de cardinalidad mínima.
 - **Ejemplo:** biblioteca(nombre, calle, número)
- A las superclaves minimales se les llama **claves candidatas**.
 - Porque son candidatas de ser elegidas.
 - Una de las claves candidatas es la elegida y a esa se le llama la **clave primaria**.
- **Notación:**
 - Se indican los atributos de una clave primaria para un esquema de relación R **subrayando** los atributos de R que forman la clave primaria.

Claves foráneas

- **Restricción de clave foránea (o de integridad referencial):**
dice que el valor en una relación debe aparecer en otra.
 - **Ejemplo:**
 - ❑ *instructor*(ID, name, dept name, salary)
 - ❑ *department*(dept name, building, budget)
 - El valor de *dept name* en *instructor* debe aparecer en *department*
 - Relación **referenciante** y relación **referenciada**.
 - ❑ *Instructor* es la referenciante y *department* es la referenciada.
 - ❑ **For instructor foreign key dept name references department**

Claves foráneas

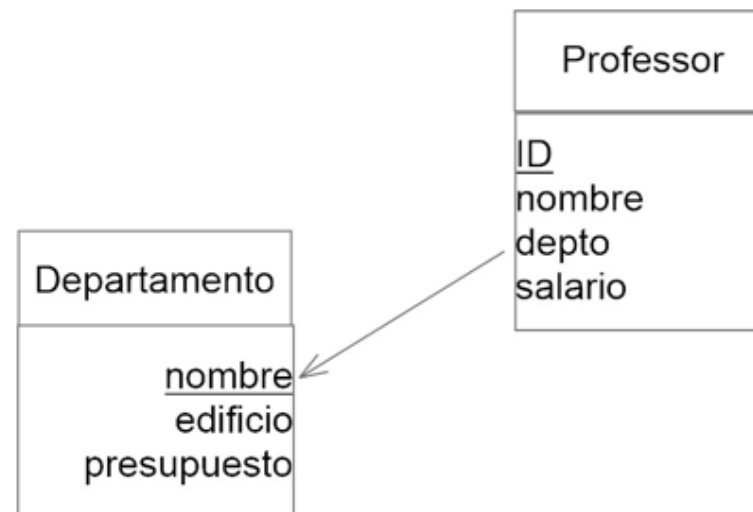
- **Generalizando y siendo un poco más precisos:**
 - **Restricción de clave foránea:** Los valores de uno o más atributos en una tupla de la relación referenciante aparecen en uno o más atributos de una tupla en la relación referenciada.
 - Los atributos referenciados en la relación referenciada suelen formar una **clave primaria** del esquema de la relación referenciada.
 - **Generalizando aun más:** los atributos referenciados de la relación referenciada pueden formar una **clave candidata** del esquema de la relación referenciada.

Claves foráneas

- **Representación gráfica:**

- ❑ Los esquemas se representan con rectángulos conteniendo nombre de esquema y nombres de atributos.
- ❑ Los atributos de clave primaria se subrayan.
- ❑ Las restricciones de clave foránea se representan mediante flechas que van de atributos referenciantes (de esquema referenciante) a atributos referenciados (de esquema referenciado).

- **foreign key** *depto of Profesor* **references** *nombre of Departamento*



Diseño de BD relacional

- **Problema:** ¿Cómo hacer un buen diseño de BD relacional?
 - Esto es lo mismo que encontrar un “buen” esquema de una BD relacional.
- **Mal diseño:**
 - *Universidad = (instructorID, nombre, nombreDepto, salario, estudianteID)*
- Almacenar toda la información en una sola tabla resulta en:
 - **Repetición de la información** (llamado **redundancia de datos**)
 - **Ejemplo:** varios estudiantes con el mismo instructor (se repite la información del instructor)

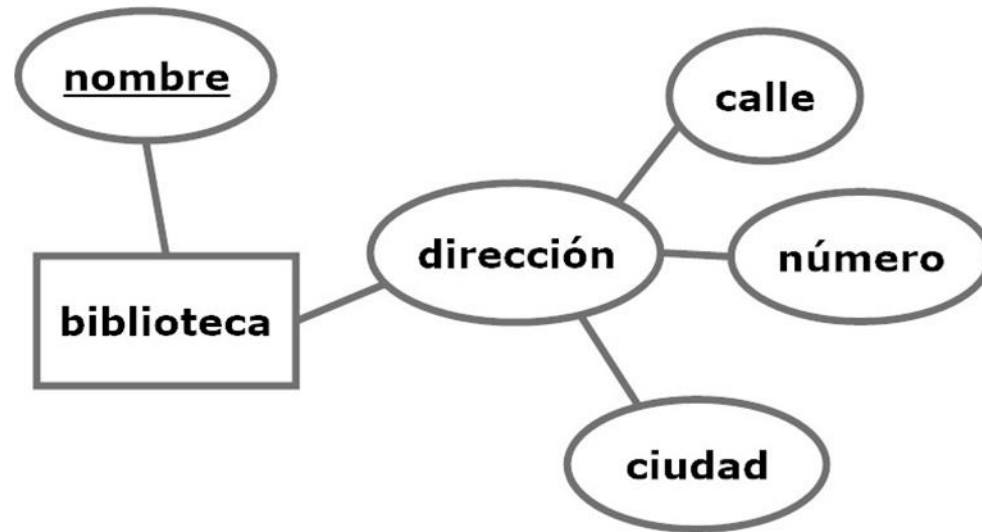
Diseño de BD relacional

- **Meta:** diseñar un esquema de la BD que no contenga redundancia de datos.
 - veremos técnicas para hacer esto.
- **Idea:** Obtener un buen diseño **descomponiendo el esquema** que contiene todos los atributos en esquemas más chicos.
 - **Ejemplo:** *Univ = (instructorID, estudianteID)*
Instructor = (instructorID, nombre, nombreDepto, salario)
 - Notar que desapareció el problema del ejemplo previo.
 - La **teoría de normalización** trabaja con esta idea y trata con cómo diseñar buenos esquemas de BD relacionales.

Procesos de diseño

- Vamos a estudiar los siguientes 2 **procesos de diseño**:
 - **Proceso 1**: Diseño de entidad-relación y pasaje a tablas
 - se hace un diseño de entidad-relación primero y
 - luego **traduce** ese diseño de entidad-relación a un conjunto de esquemas de relación.
 - **Proceso 2**: Descomponer esquema relacional con todos los atributos usando restricciones de integridad.
 - que comienza con esquema relacional con todos los atributos atómicos del problema y un conjunto de restricciones de integridad y
 - **calcula** un esquema de la base de datos que cumple ciertas propiedades deseables.
 - A esto se le llama **normalización**.

Traducción de DER a esquema relacional



Se mapea a:

biblioteca(nombre, calle, número, ciudad)

Lenguajes de Consultas

- Una **consulta** es una expresión que describe una colección de datos deseada.
- **Ejemplo**: Encontrar el nombre y salario de instructores que ganan más de \$ 50000.
- Para expresar consultas se usan **lenguajes de consulta**, que varían según el modelo de datos.
- **Modelo Relacional**:
 - **SQL**: Lenguaje estándar en la industria.
 - **Ejemplo en SQL** (para el ejemplo anterior)

```
select name, salary
from instructor
where salary > 50000
```

Lenguajes de Consultas

- **Modelo relacional – continuación:**
 - **Álgebra relacional y cálculo de tuplas:** son lenguajes formales más simples y enfocados en aspectos fundamentales.
- **En la materia veremos:**
 - Una variación más expresiva del álgebra relacional: **álgebra de tablas**.
 - Cómo el sistema gestor de BD procesa consultas.
 - Cómo traducir una consulta SQL a álgebra de tablas para facilitar el procesamiento.
- **Más allá del modelo relacional:**
 - Existen **modelos de datos no relacionales** con sus propios lenguajes de consulta.
 - P.ej. Los lenguajes MongoDB (documentos), XQuery (XML).
 - En el laboratorio de la materia van a estudiar MongoDB.

Sistema gestor de BD relacionales

- Un **sistema gestor de BD** (SGBD) relacionales se compone de:
 - Gestor de almacenamiento
 - Procesamiento de consultas
 - Gestor de transacciones

Gestión del almacenamiento

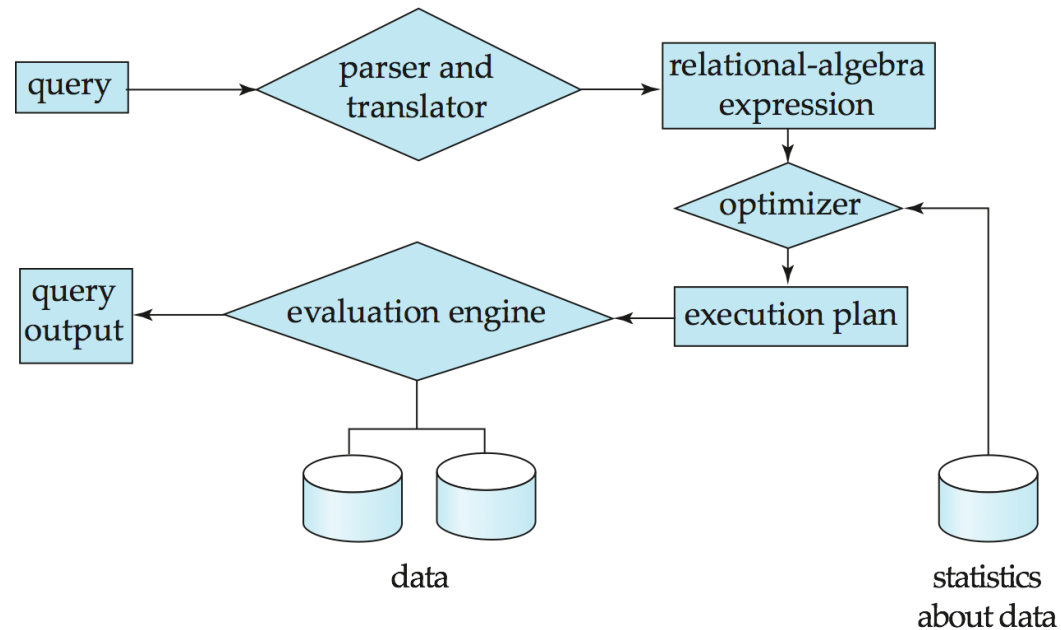
- La **gestión de almacenamiento** se encarga de organizar los datos para que su acceso, modificación y recuperación sean eficientes.
- **Nivel físico:**
 - Describe **como se almacenan los registros** en archivos (con estructuras apropiadas).
 - Usa **estructuras de acceso** como **índices** para mejorar la eficiencia.
 - **Se ocupa de:**
 - Organización de datos en archivos
 - Acceso al almacenamiento
 - Indexación.
- **Nivel lógico:**
 - Describe qué datos están almacenados y cómo se relacionan.
 - Es el nivel visible para los usuarios y aplicaciones.
- **Rol del gestor de almacenamiento:**
 - El **gestor de almacenamiento** provee una interfaz entre el nivel físico y los programas de aplicación y (consultas, actualizaciones, etc.)
 - Se ocupa de: acceso al almacenamiento, organización en archivos de los datos, indexado.

Procesamiento de consultas

- Transformar una consulta en lenguaje declarativo como SQL en una ejecución eficiente sobre los datos implica tres etapas fundamentales:

1. Parseo y traducción:

- Se analiza la **sintaxis** y **semántica** de la consulta.
- Se **traduce** la consulta a una **representación interna**, como el álgebra de relacional.



Procesamiento de consultas

2. Optimización:

- Se generan **planes de ejecución alternativos**.
- Se selecciona el **más eficiente** según métricas como:
 - Accesos a disco
 - Uso de índices
 - Tamaño de las relaciones
 - Estadísticas del sistema
- La **diferencia de costo** entre una buena y una mala estrategia puede ser enorme.

3. Evaluación (siguiendo el plan optimizado)

- Se ejecuta la consulta siguiendo el plan optimizado
- Se aplican operadores y se accede a los datos para obtener el resultado.

Procesamiento de consultas

- ¿Por qué optimizar?

- Una misma consulta puede evaluarse de **formas alternativas**.
- La **eficiencia** del plan elegido impacta directamente en el **tiempo de respuesta** y el **uso de recursos**.
- Por eso, la optimización de consultas es una etapa crítica en los sistemas de bases de datos.

Transacciones

- Una **transacción** es una unidad lógica de trabajo que agrupa operaciones sobre la BD, como lecturas y escrituras, para cumplir una función específica.
- **Ejemplo:** Transferencia bancaria de \$50:
 1. **read**(A)
 2. $A := A - 50$
 3. **write**(A)
 4. **read**(B)
 5. $B := B + 50$
 6. **write**(B)
- Si ocurre una falla entre los pasos, por ejemplo después **write**(A) pero antes de **write**(B) , la BD queda en un estado **inconsistente**.

Transacciones

- La **componente de manejo de transacciones** asegura que BD permanezca en un estado consistente (correcto) a pesar de fallas del sistema (e.g. fallas de energía, caídas de sistema operativo) y fallas de transacciones.
- ¿Cómo hacer para que una transacción se ejecute indivisiblemente?
- **Solución:** Aplicar **atomicidad**.
 - **Atomicidad** significa o todas las operaciones de la transacción son reflejadas en la BD o ninguna lo es.
 - Es esencial para evitar errores como "sacar dinero de una cuenta sin depositarlo en otra".

Transacciones

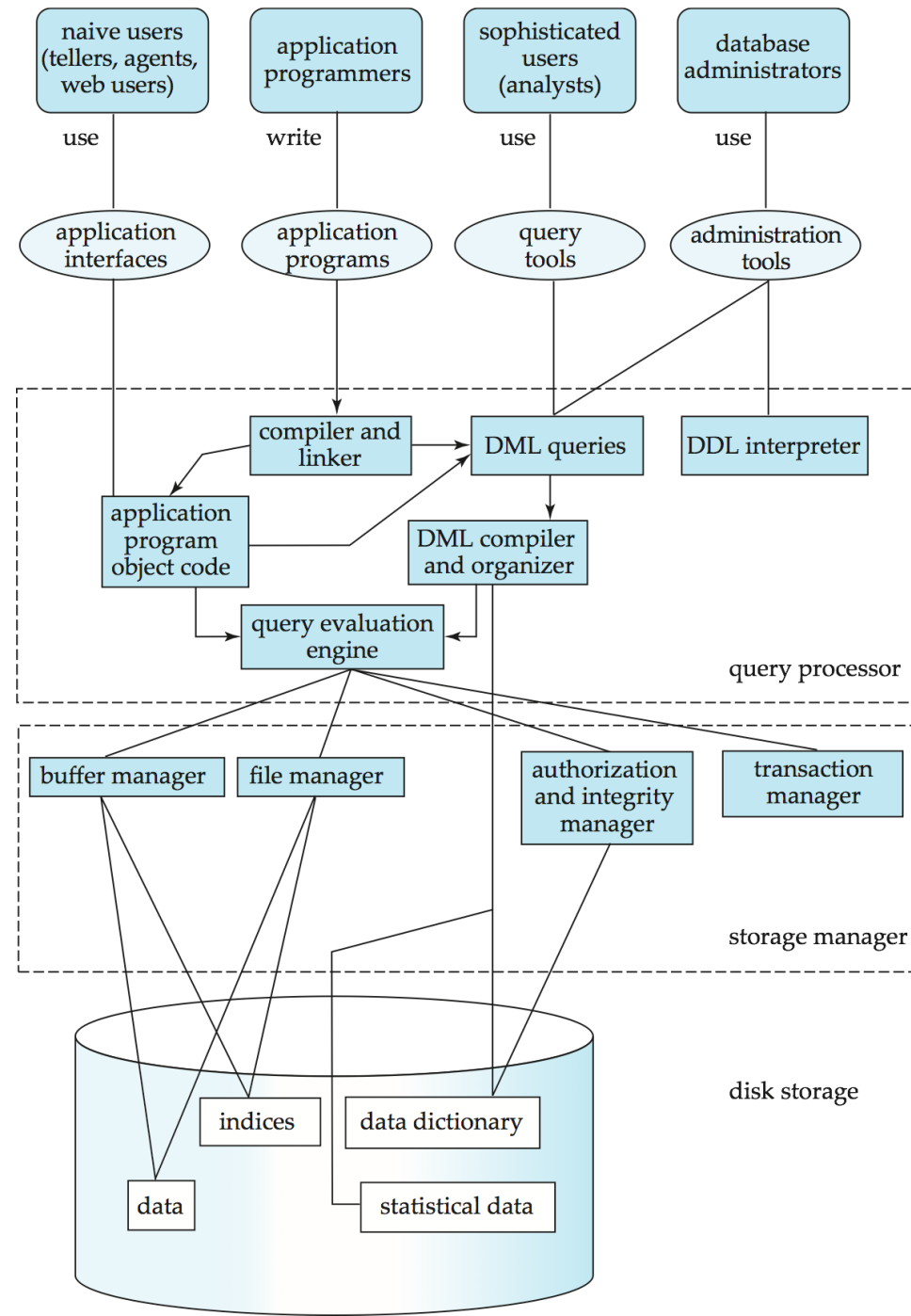
- **Planificaciones:** secuencias que indican el orden cronológico en el cual las instrucciones de transacciones concurrentes son ejecutadas.
- **Ejemplo de planificación:**

T_1	T_2
read (A) $A := A - 50$ write (A)	read (A) $temp := A * 0.1$ $A := A - temp$ write (A)
read (B) $B := B + 50$ write (B) commit	read (B) $B := B + temp$ write (B) commit

Transacciones

- Cuando varias transacciones actualizan la BD concurrentemente, la consistencia de los datos puede dejar de ser preservada, aun cuando cada transacción individual es correcta.
 - Se debe ejecutar una planificación adecuada que no genere problemas.
- **Gestor de concurrencia de transacciones:**
 - controla la **interacción entre transacciones concurrentes**
 - Asegura que la BD permanezca **consistente** mediante:
 - Bloqueos
 - Control de versiones
 - Protocolos de serialización

Arquitectura de un SGBD



Retorno de la información

- El **retorno de la información** es el proceso de recuperar documentos relevantes desde una colección de documentos, en respuesta a una consulta del usuario.
- **Características de los documentos:**
 - Los documentos suelen estar en lenguaje natural no estructurado.
 - No se organizan en tablas ni siguen esquemas rígidos como en bases de datos relacionales.
- **Consultas en sistemas de RI:**
 - Un tipo muy común de consultas usan palabras clave y conectivos proposicionales.
 - **Ejemplo:** inteligencia artificial AND ética
- **Resultados de la búsqueda:**
 - El sistema retorna:
 - Una **lista de identificadores de documentos**
 - Fragmentos de texto relevantes.
- Los documentos suelen ordenarse por **puntaje de relevancia**.

Retorno de la información

- **¿Cómo organizaremos el estudio de un SRI?**
 - **Arquitectura general de un sistema de retorno de la información**
 - **Cómo está organizado** un sistema de RI.
 - Detalle de la componentes que lo integran.
 - **Procesamiento de documentos**
 - Que información se extrae y almacena
 - Como se representan los documentos para facilitar la búsqueda.
 - **Procesamiento de consultas**
 - Cómo se interpreta una consulta
 - Como se accede a la información procesada de los documentos
 - Cómo se devuelven los resultados relevantes.

Retorno de la información


- **Dos casos de estudio:**
 - **Sistemas locales:** con **colecciones de archivos en una PC.**
 - **Motores de búsqueda web** (p.ej. Google o Bing)
- **¿Qué problemas resuelve cada tipo de Sistema?**
 - **Sistemas locales de retorno de información:**
 - Ranking de documentos según relevancia
 - Eficiencia en la búsqueda (uso de índices invertidos)
 - Representación de documentos y consultas (modelo vectorial, booleano, etc.)
 - Evaluación de resultados (precisión, exhaustividad)

Retorno de la información

○ Motores de búsqueda web:

- Además de lo anterior, enfrentan problemas adicionales:
 - Escalabilidad extrema: millones o billones de documentos
 - Distribución geográfica de servidores e índices
 - Actualización constante del contenido (web dinámica)
 - Detección de spam y manipulación de resultados
 - Personalización según usuario, ubicación, historial
 - Interpretación de consultas ambiguas o mal formuladas
- Mientras que los sistemas locales se centran en la **eficiencia y relevancia** dentro de una colección acotada, los motores web deben lidiar con la **magnitud, diversidad y dinamismo de la información global**.

Retorno de la información

Aspecto	 Sistemas Locales	 Motores de Búsqueda Web
Tamaño de la colección	Limitado (miles a cientos de miles)	Masivo (millones a billones de documentos)
Indexación	Local, estática	Distribuida, dinámica y continua
Ranking de resultados	Basado en similitud textual	Incluye popularidad, enlaces, comportamiento
Actualización de contenido	Poco frecuente	Constante (web cambiante)
Consulta del usuario	Directa, sin personalización	Personalizada (historial, ubicación, etc.)
Spam y ruido	Poco frecuente	Muy frecuente, requiere detección activa
Ambigüedad en consultas	Menor, contexto más controlado	Alta, requiere interpretación semántica
Evaluación de resultados	Métricas clásicas (precisión, recall)	Métricas más complejas (CTR, satisfacción)
Escalabilidad	No crítica	Fundamental para rendimiento global

Bot de chat conversacional inteligente

- ¿Qué es un bot de chat conversacional inteligente?
- Un **bot de chat conversacional inteligente** es una aplicación de software que utiliza:
 - Inteligencia Artificial (IA)
 - Procesamiento de Lenguaje Natural (PLN)

... para **comprender lo que una persona necesita y ayudarla a lograrlo** mediante el diálogo.

- ¿Qué hacen estos bots?
 - Comprenden el lenguaje humano (interpretan intenciones, contexto, emociones).
 - Generan lenguaje (responden de forma coherente, relevante y contextual).
- **Analogía: El Bot como Asistente Experto**
 - Imagina que tienes acceso a un asistente experto que ha leído millones de libros, artículos y conversaciones.
 - Le haces una pregunta → te responde con precisión y contexto.
 - Le das instrucciones → las ejecuta con claridad.
 - Cuanto más claro seas, mejor será su ayuda.

Bot de chat conversacional inteligente

- **Tipos de bots de chat:**
 - **De texto:** interacción solo por palabras escritas
 - **Multimedia:** incluyen texto + imágenes, videos, voz.
- En esta materia estudiaremos los bots de texto.
- **Ejemplos de bots de texto:**
 - chatGPT
 - Google Gemini
 - Perplexity AI
- **¿Cómo interactuar con ellos?**
 - Se usan **consultas claras y concisas**, llamadas **prompts**.
 - Un buen prompt mejora la calidad de la respuesta.
- Estudiaremos como diseñar prompts efectivos siguiendo una **estructura recomendada**.

Bot de chat conversacional inteligente

- **Procesamiento de lenguaje natural (PLN)**

- El PLN permite que los bots comprendan y generen lenguaje humano.
- **Componentes clave:**
 - **Comprensión del lenguaje natural:** interpretar lo que el usuario dice.
 - **Generación del lenguaje natural:** responder de forma coherente y contextual.
 - **Contexto y sus tipos:** contexto inmediato (última frase), contexto extendido (tema de conversación), contexto externo (conocimiento general)

- **Modelos grandes de lenguaje**

- Los LLM son redes entrenadas con grandes cantidades de texto.
- Aprenden **patrones lingüísticos y relaciones semánticas**.
- Pueden responder preguntas, traducir, resumir, crear contenido, etc.
- Ejemplos: GPT 4, PaLM, Claude
- Son como cerebros artificiales entrenados en millones de textos.

Bot de chat conversacional inteligente

- **Arquitectura de Transformers**

- La arquitectura de transformers es la base técnica de los LLM modernos.
- Los transformers se componen de **dos partes principales**:
 - **Codificador**
 - Recibe el texto de entrada
 - Analiza su estructura y significado
 - Extrae representaciones internas útiles para comprender el contenido
 - **Decodificador**
 - Toma la representación interna generada por el codificador
 - Genera texto de salida palabra por palabra, manteniendo coherencia y contexto
- Podemos pensar al codificador como quien “lee y entiende”, y al decodificador como quien “responde con sentido”.

Bot de chat conversacional inteligente

- **¿Cómo se conectan los LLM con los bots conversacionales?**
 - Los **Modelos de Lenguaje de gran escala** (LLM) son el **motor principal** detrás de los bots conversacionales inteligentes.
 - Un bot como **ChatGPT** funciona como una **interfaz de diálogo** que:
 - **Recibe:**
 - El texto actual que escribe el usuario (**prompt**)
 - El **historial de la conversación** (prompts y respuestas)
 - **Procesa** todo este contexto usando un **LLM** que analiza el lenguaje y genera una respuesta coherente.
 - **Retorna** texto como respuesta al nuevo prompt, manteniendo el **hilo conversacional**.

Bot de chat conversacional inteligente

- En resumen:
 - **Bot conversacional inteligente** = LLM + interfaz de diálogo + funciones específicas + contexto conversacional
- Enfoque del curso: ¿Qué estudiaremos sobre los bots?
 - En lugar de abordar los bots desde el punto de vista técnico de la inteligencia artificial que requiere cursos avanzados, nos enfocaremos en el **qué hacen** y **no en el cómo lo hacen**.

Bot de chat conversacional inteligente

- **¿Cómo lo abordaremos?**

- Veremos las distintas etapas del Procesamiento de Lenguaje Natural (comprensión y generación del lenguaje) y de los Transformers (codificador y decodificador) como **procesos divididos en funciones**.
- **Cada función:**
 - Recibe texto
 - Retorna texto o texto estructurado
- Analizaremos y especificaremos qué tarea realiza cada función, con ejemplos concretos.

- **Ventajas de ese enfoque:**

- Permite **entender el funcionamiento general** sin requerir conocimientos avanzados de IA.
- Facilita el **uso práctico** de los bots en aplicaciones reales.
- Prepara el terreno para **diseñar buenos prompts** y comprender cómo interactuar con estos sistemas.