

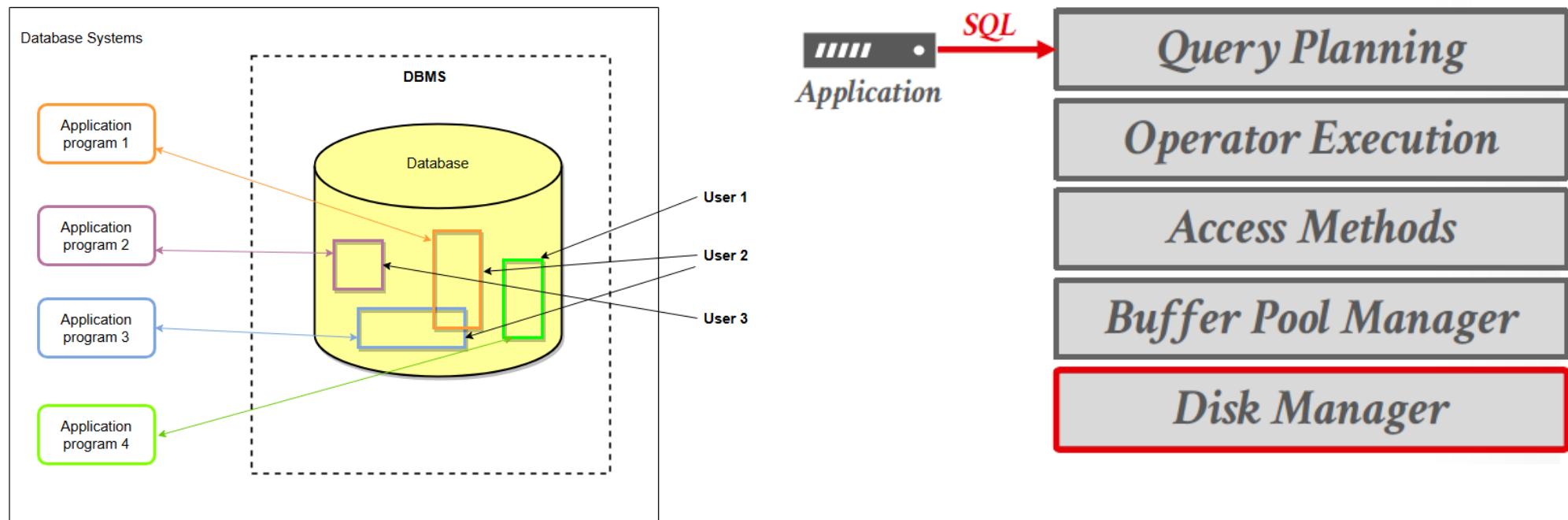
**Curso** —————  
**Estructuras de datos**

**UNIVERSIDAD  
DE ANTIOQUIA**

Clase 1 – Introducción y conceptos claves

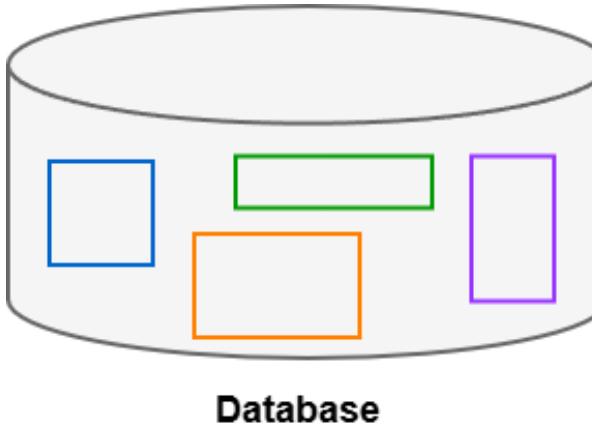
# Sobre el curso

- En este curso se abordaran algunos conceptos relacionados con el diseño y la implementación de sistemas de gestión de bases de datos orientados a disco.
- No es un curso sobre cómo usar una base de datos para construir aplicaciones ni sobre cómo administrar una base de datos.



# ■ Base de datos

- Colección organizada de datos interrelacionados que modela algún aspecto del mundo real.



- Las bases de datos son el componente central de la mayoría de las aplicaciones informáticas.



Uber



# ■ Aplicaciones de las bases de datos

- **Información empresarial**
  - Ventas: clientes, productos, compras
  - Contabilidad: pagos, recibos, activos
  - Recursos Humanos: información sobre empleados, salarios, nómina e impuestos
- **Manufactura**
  - Gestión de la producción, inventarios, pedidos y cadena de suministro
- **Banca y finanzas**
  - Información de clientes, cuentas, préstamos y transacciones bancarias
  - Transacciones con tarjetas de crédito
  - Finanzas: compras y ventas de instrumentos financieros (por ejemplo, acciones y bonos), almacenamiento de datos de mercado en tiempo real
- **Universidades**
  - Matrícula (registro) y calificaciones



# ■ Aplicaciones de las bases de datos

- **Aerolíneas:** reservas, horarios
- **Telecomunicaciones:** registros de llamadas, mensajes de texto y uso de datos; generación de facturas mensuales y mantenimiento de saldos en tarjetas prepago
- **Servicios basados en la web**
  - Comercios en línea: seguimiento de pedidos, recomendaciones personalizadas
  - Publicidad en línea
- **Bases de datos de documentos**
- **Sistemas de navegación:** mantenimiento de la ubicación de diversos puntos de interés junto con las rutas exactas de carreteras, sistemas de trenes, autobuses, etc.



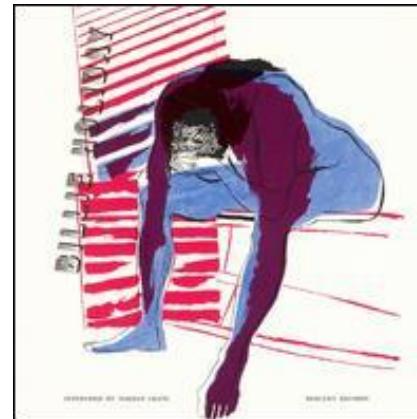
# Ejemplo de base de datos

**Crear una base de datos que modele una aplicación de música digital para llevar el registro de artistas y álbumes.**

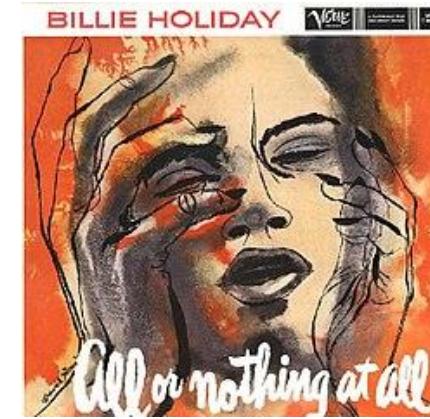
- Información que necesitamos llevar en nuestra tienda:
  - Información sobre los artistas
  - Los álbumes que esos artistas han publicado



Billie Holiday [\[link\]](#)



Billie Holiday Sings  
(1952) [\[link\]](#)



All or Nothing at All  
(1958) [\[link\]](#)

# ■ Almacenamiento de la información

- Una forma simplificada de almacenar nuestra base de datos es mediante archivos de valores separados por comas (CSV) que administramos directamente desde el código de la aplicación.
  - Usar un archivo separado por cada entidad.
  - La aplicación debe interpretar (parsear) los archivos cada vez que quiera leer o actualizar registros.

**Artist(name, year, country)**

```
"Billie Holiday", 1933, "USA"  
"Ella Fitzgerald", 1917, "USA"  
"Louis Armstrong", 1922, "USA"
```

**Album(name, artist, year)**

```
"Lady in Satin", "Billie Holiday", 1958  
"Ella and Louis", "Ella Fitzgerald", 1957  
"Ella and Louis", "Louis Armstrong", 1957  
"What a Wonderful World", "Louis Armstrong", 1967
```



# ■ Almacenamiento de la información

- **Ejemplo:** Obtener el año en que Louis inició su carrera como solista.

**Artist(name, year, country)**

```
"Billie Holiday", 1933, "USA"  
"Ella Fitzgerald", 1917, "USA"  
"Louis Armstrong", 1922, "USA"
```



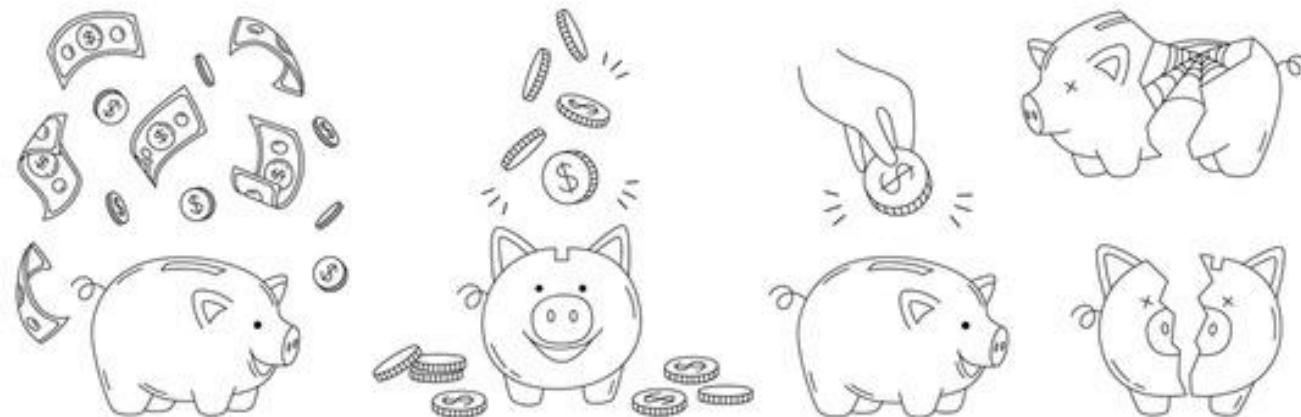
```
for line in file.readlines():  
    record = parse(line)  
    if record[0] == "Louis Armstrong":  
        print(int(record[1]))
```

# ■ Información: Integridad de los datos

- ¿Cómo garantizar que el nombre del artista sea consistente en todas las entradas de sus álbumes?
  - Ejemplo: "Louis Armstrong" vs. "Luis armstrong"
- ¿Qué pasa si alguien sobrescribe el año del álbum con una cadena de texto inválida?
- ¿Qué ocurre si hay múltiples artistas en un mismo álbum?
- ¿Qué sucede si eliminamos un artista que tiene álbumes asociados?

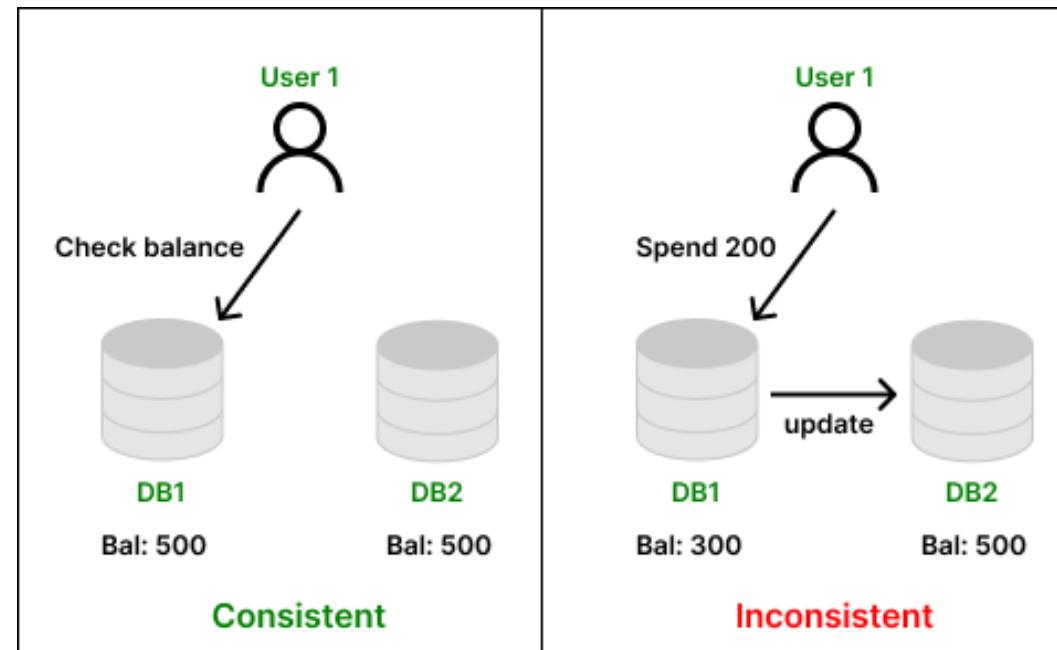
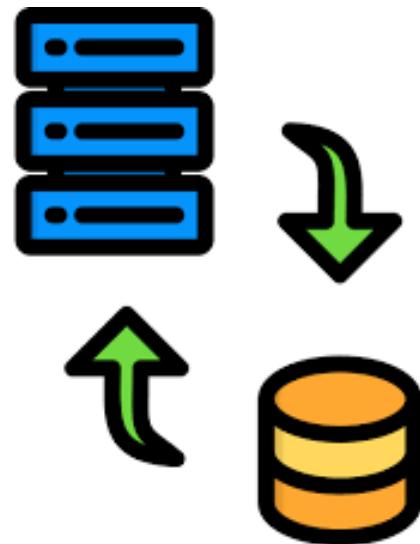
# ■ Información: Implementación

- ¿Cómo se encuentra un registro específico?
- ¿Qué ocurre si ahora queremos crear una nueva aplicación que use la misma base de datos?
- ¿Y si esa aplicación se está ejecutando en una máquina diferente?
- ¿Qué pasa si dos hilos (threads) intentan escribir en el mismo archivo al mismo tiempo?



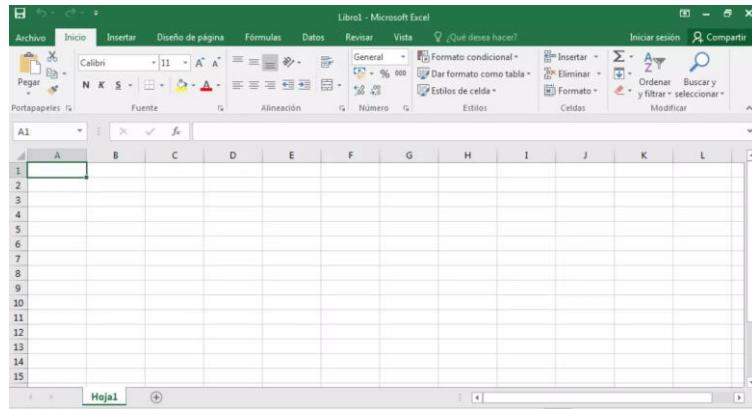
# ■ Información: Durabilidad

- ¿Qué pasa si la computadora se bloquea mientras nuestro programa está actualizando un registro?
- ¿Qué ocurre si queremos replicar la base de datos en múltiples máquinas para lograr alta disponibilidad?



# Sistema basado en archivos

- En este enfoque, el programa de aplicación es quien gestiona la información almacenando sus datos en diferentes archivos dentro del sistema operativo.
- Características principales:
  - **Estructura Rudimentaria:** Los datos se guardan como cadenas de texto (ej. CSV, TXT) y su interpretación depende totalmente de la lógica del programa que los lee.
  - **Dependencia del Programador:** Las búsquedas, validaciones y relaciones entre datos deben ser programadas explícitamente para cada aplicación.
  - **Acceso mediante el SO:** Se basan exclusivamente en las jerarquías de directorios del sistema operativo para localizar registros.



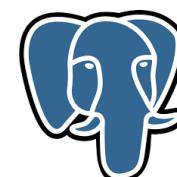
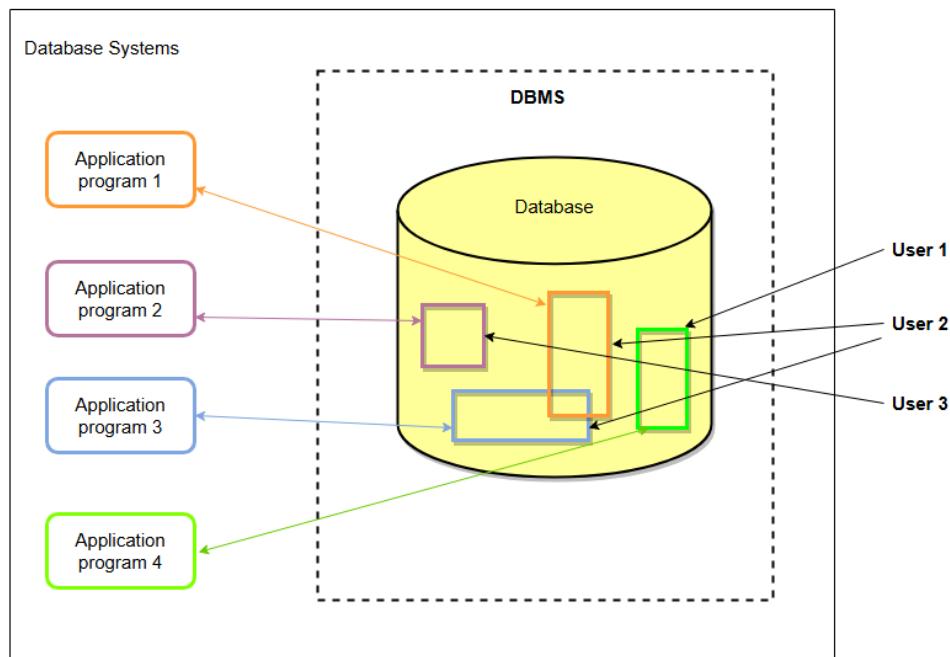
## Problemas principales de los sistemas basados en archivos:

- **Redundancia de datos:** La misma información se guarda en múltiples archivos.
- **Inconsistencia:** Diferentes copias del mismo dato no coinciden.
- **Difícil acceso a la información:** Se debe conocer la ubicación exacta del archivo.
- **Problemas de seguridad:** No hay control centralizado de accesos.
- **Falta de concurrencia:** Cuando un archivo está abierto, otros usuarios no pueden acceder.



# ■ Database Management System (DBMS)

- Un **sistema de gestión de bases de datos (DBMS)** es un software que permite a las aplicaciones almacenar y analizar información en una base de datos.
- Un DBMS de propósito general soporta la definición, creación, consulta, actualización y administración de bases de datos de acuerdo con algún modelo de datos.
- Los DBMS se utilizan en casi todas las aplicaciones, sitios web y sistemas de software que pueda imaginar.



# ■ Arquitectura de tres capas

- Un sistema de bases de datos está diseñado para separar lo que el usuario ve, de cómo están organizados los datos, y de cómo realmente se almacenan físicamente.
- Los niveles de abstracción en un DBMS son capas diseñadas para ocultar la complejidad técnica del almacenamiento de datos y facilitar la interacción de diferentes tipos de usuarios con el sistema.

## Nivel de Vista (View Level)

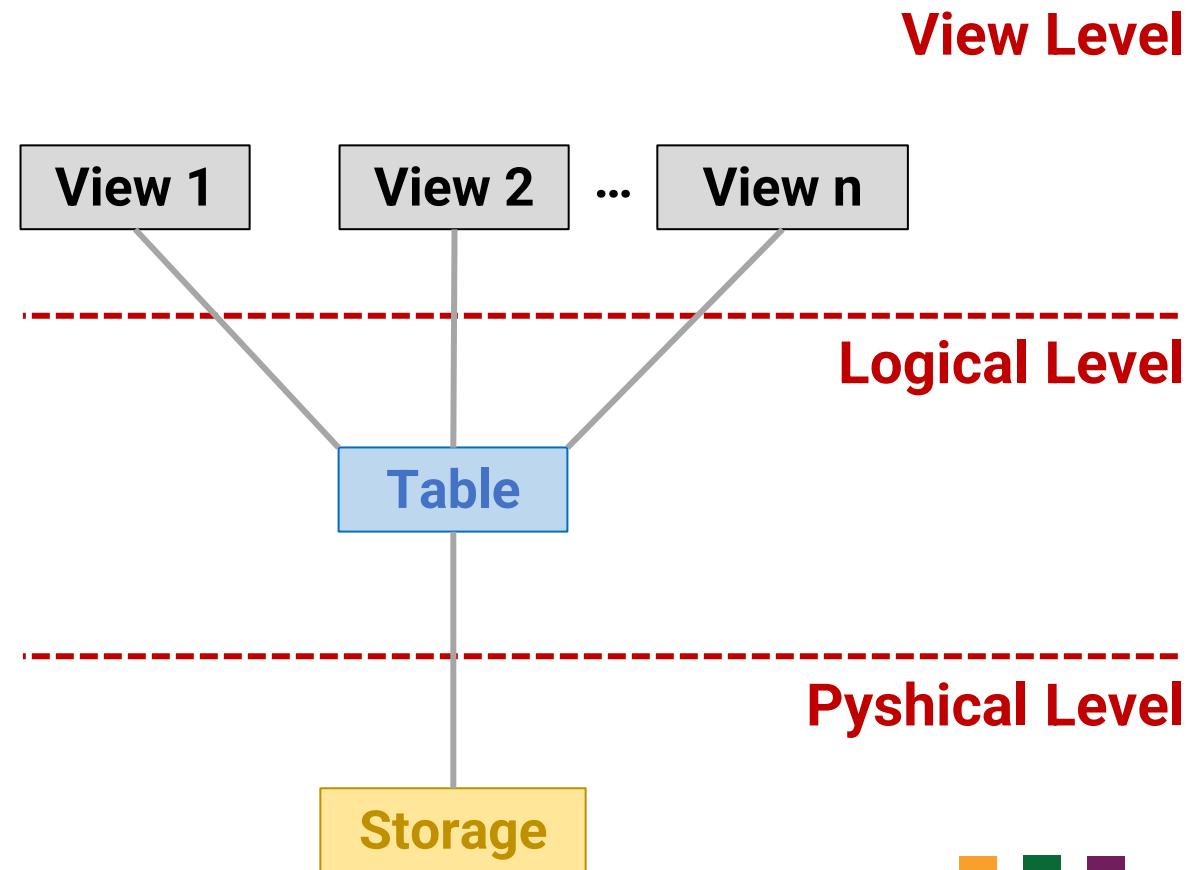
Qué información se expone a los usuarios, qué se les permite ver.

## Nivel Lógico (Logical Level)

Qué datos están allí, qué atributos tienen, qué restricciones deberían aplicar el BDMS...

## Nivel Físico (Physical Level)

Cómo se almacenan los datos, dónde están ubicados, cuántos bytes, qué tipo de índices...



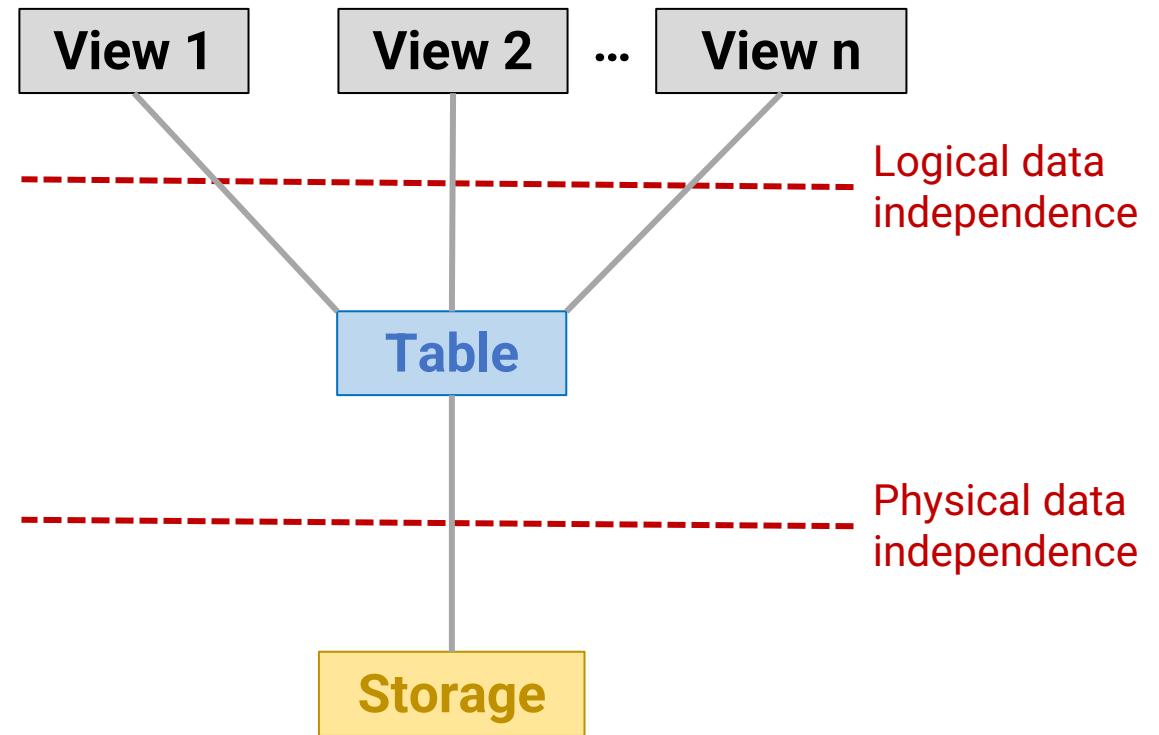
# ■ Arquitectura de tres capas

## Independencia Lógica de Datos

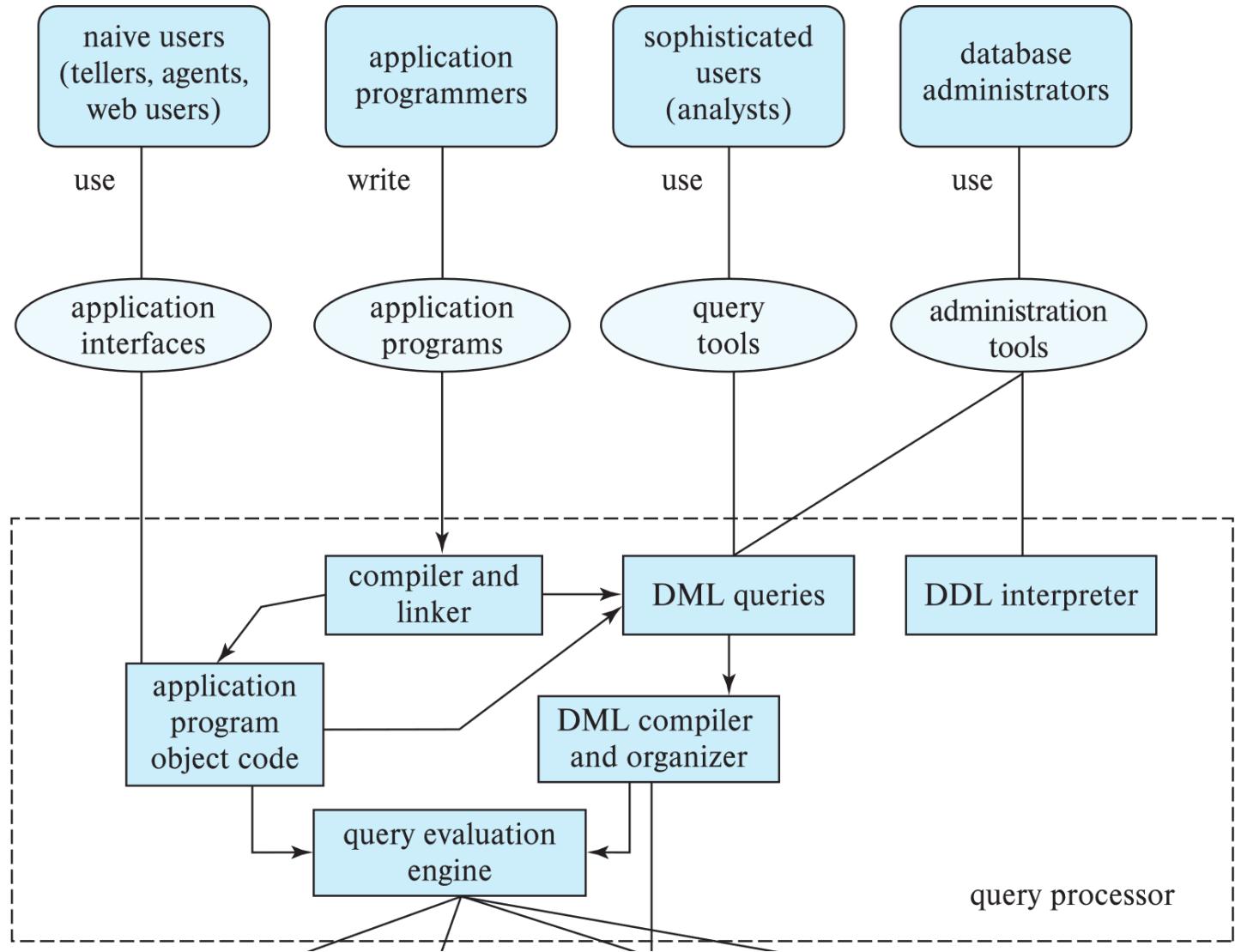
- Podemos modificar las definiciones de nuestras tablas sin tener que cambiar las vistas de la aplicación.
- **Ejemplo:**
  - Agregar / eliminar / renombrar atributos de una tabla.
  - Renombrar una tabla.

## Independencia Física de Datos

- Podemos cambiar cómo/dónde los objetos de la base de datos están representados en el almacenamiento físico.
- **Ejemplos:**
  - Usar 32 bits en lugar de 64 bits para enteros.
  - Convertir un índice de un Árbol B+ a una SkipList.
  - Comprimir una tabla cuando está almacenada en disco.
  - Mover una tabla a otro disco/máquina.



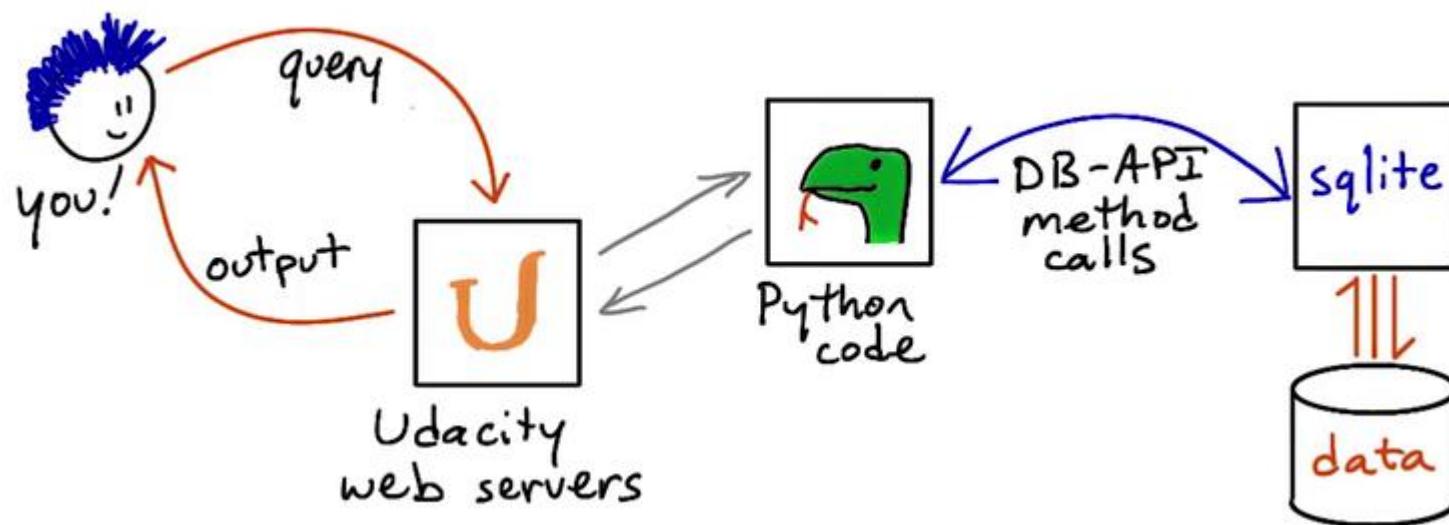
# ■ Tipos de usuarios



# ■ Tipos de usuarios

Las personas que trabajan con bases de datos se dividen en diferentes categorías según su nivel de interacción técnica y sus responsabilidades.

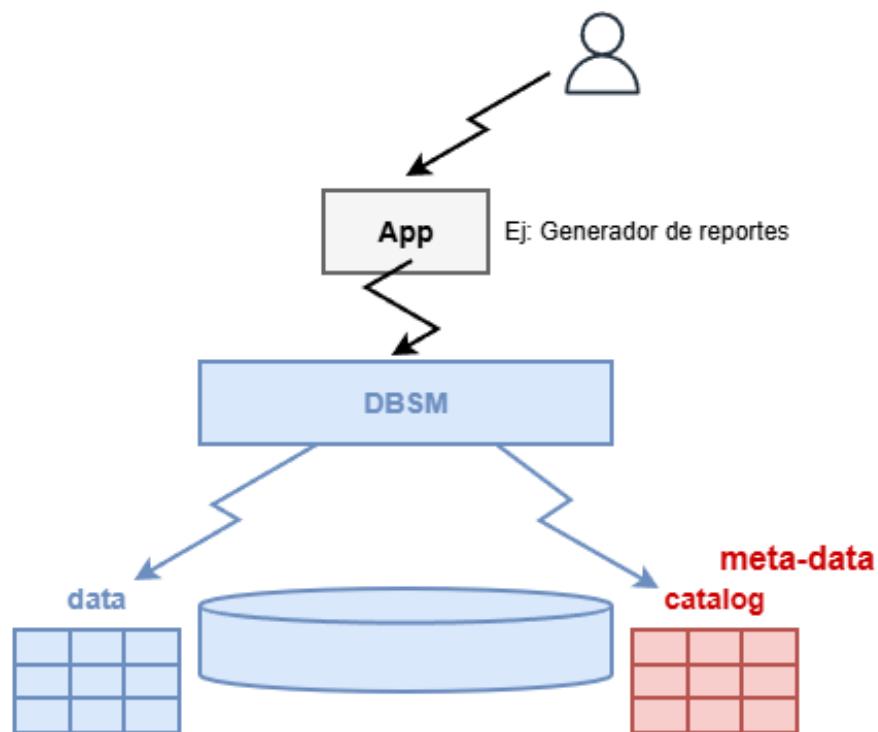
- Naive users (usuarios ingenuos / finales)
- Casual users (usuarios ocasionales)
- Application programmers (programadores de aplicaciones)
- DBA – Database Administrator



# ■ Tipos de usuarios

## Naive users (usuarios ingenuos / finales)

- Son el grupo más numeroso y, por lo general, no son profesionales de la computación.
- No tienen que preocuparse sobre el funcionamiento interno del DBMS.
- Acceden a la base de datos a través de aplicaciones o interfaces simplificadas (como formularios web o terminales de agentes de viajes) escritas por programadores.



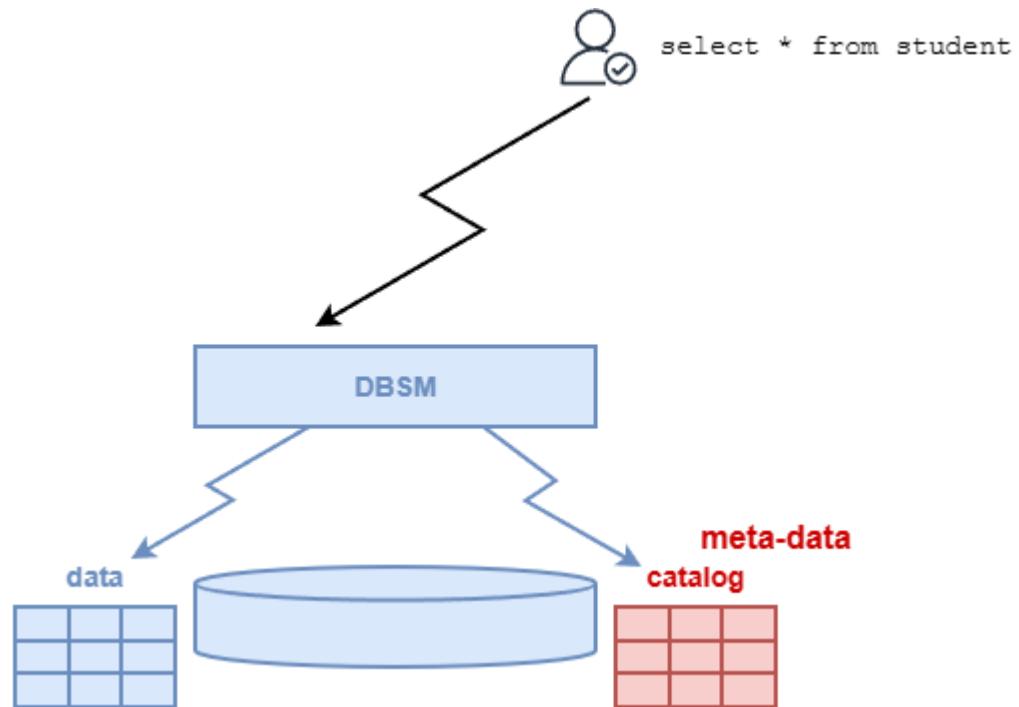
El **usuario final** cree que usa una aplicación, pero en realidad está usando una base de datos



# ■ Tipos de usuarios

## Casual users (usuarios ocasionales o sofisticados)

- Realizan un uso más extenso del sistema y suelen escribir sus propias consultas para obtener la información que necesitan.
- Requieren un entendimiento más profundo de las características del DBMS y de la estructura de los datos.



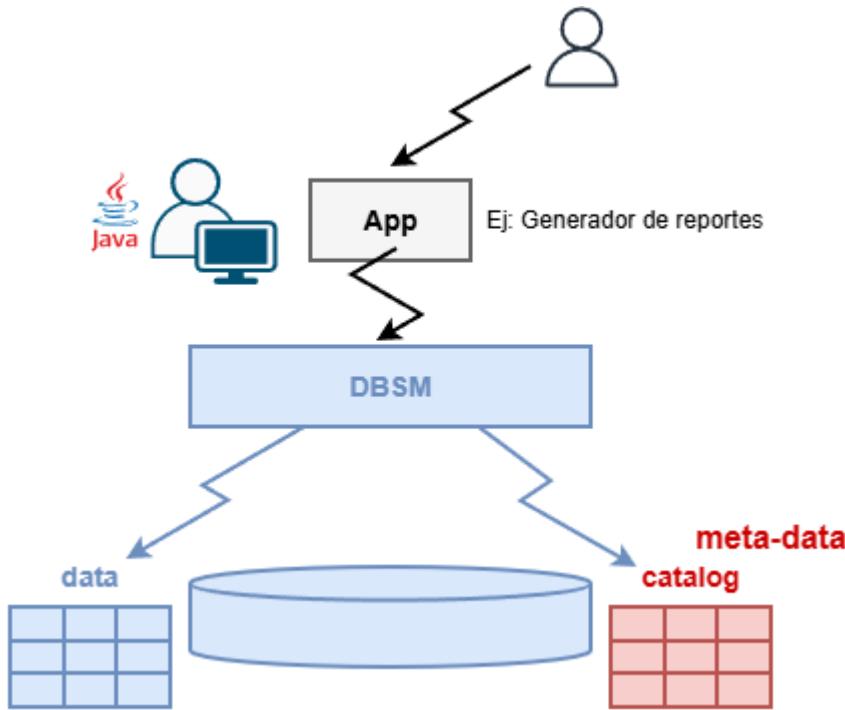
Los **usuarios ocasionales** saben qué datos quieren y para qué, pero no construyen el sistema.



# ■ Tipos de usuarios

## Application programmers (programadores de aplicaciones)

- Son los profesionales encargados de construir las herramientas para los usuarios finales.
- Desarrollan las aplicaciones que facilitan el acceso a los datos para los usuarios no profesionales.
- Utilizan lenguajes de programación (como Java o C++) combinados con lenguajes de datos proporcionados por el DBMS



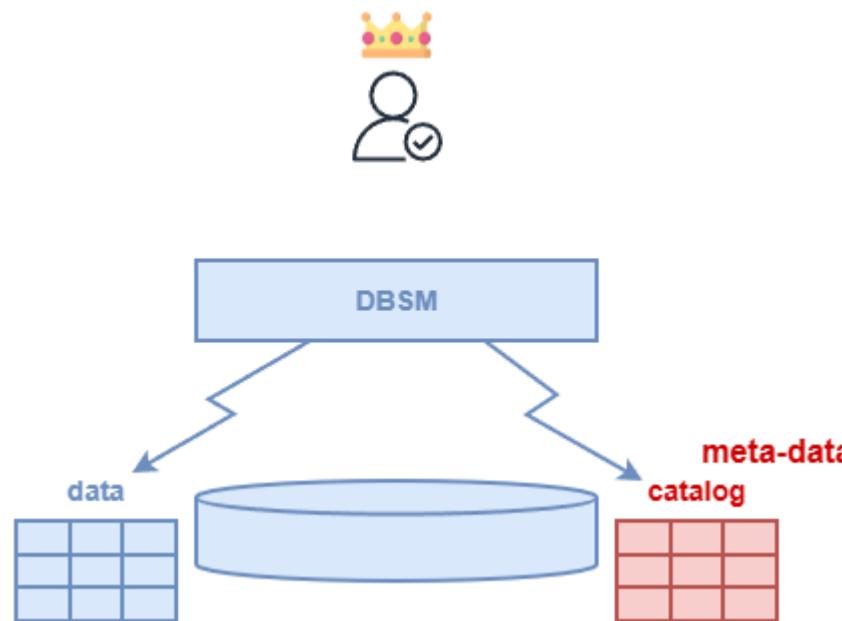
El **programador** no manipula directamente los datos, sino que construyen los caminos para acceder a ellos.



# ■ Arquitectura de tres capas

## DBA – Database Administrator

- Es el profesional responsable de la gestión integral y el mantenimiento del sistema
- Sus tareas son críticas para el funcionamiento de cualquier empresa y se dividen en cuatro áreas principales:
  - Diseño de esquemas
  - Seguridad y autorización.
  - Disponibilidad y recuperación.
  - Ajuste (Database Tuning).



El DBA no usa la base de datos: la cuida.



# Diseño de una base de datos

El proceso de diseñar la estructura general de la base de datos implica:

- **Diseño Lógico** - Decidir el esquema de la base de datos. El diseño de la base de datos requiere que encontremos una "buena" colección de esquemas de relación.
  - **Decisión del negocio:** Consiste en entender el dominio del problema - ¿Qué atributos deberíamos registrar en la base de datos?
  - **Decisión de Ciencias de la Computación:** Implica decidir cómo distribuir los datos de forma eficiente - ¿Qué esquemas de relación deberíamos tener y cómo deberían distribuirse los atributos entre los diversos esquemas de relación?
- **Diseño Físico:** Etapa donde el esquema lógico (tablas, relaciones y atributos) se traduce en una implementación real dentro de un sistema de almacenamiento – Decidir la disposición física de la base de datos (estructuras de almacenamiento, métodos de acceso, particionamiento, etc).

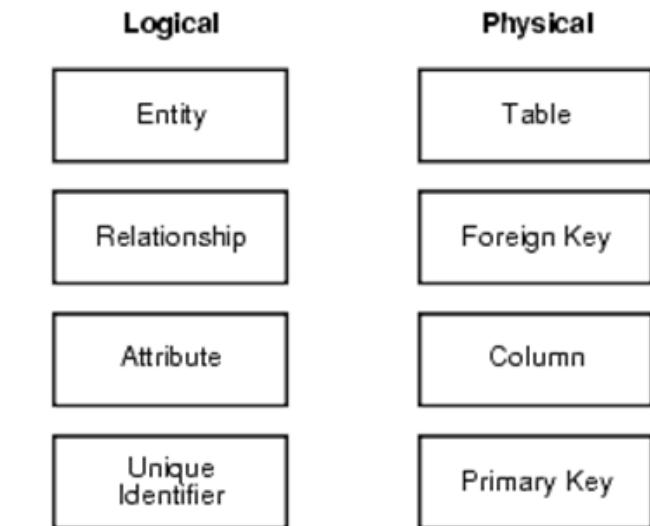
En resumen, un buen **diseño lógico** garantiza coherencia mientras que un buen **diseño físico** garantiza eficiencia.



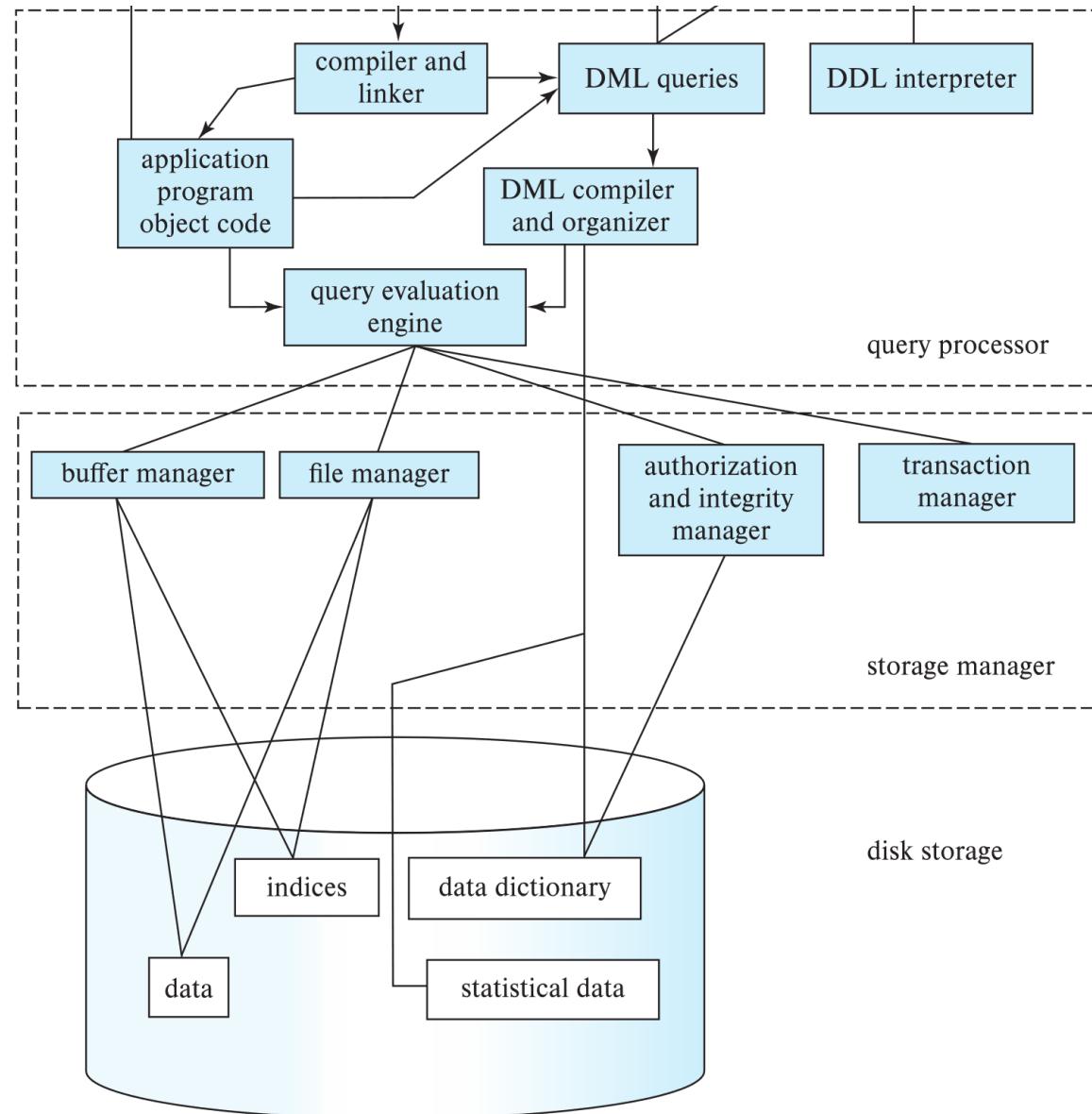
# Diseño de una base de datos

## Comparación

Característica	Diseño Lógico	Diseño Físico
Objetivo	Representar los requisitos de información.	Optimizar el rendimiento y el almacenamiento.
Enfoque	"Qué" datos se necesitan.	"Cómo" se guardan y acceden.
Independencia	Independiente del software (DBMS) y hardware.	Totalmente dependiente del software y hardware.
Entidades clave	Tablas, Atributos, Relaciones, Claves Primarias.	Índices, Particiones, Espacios de tabla (Tablespaces).
Responsable	Analistas de datos y expertos de negocio.	Administradores de Bases de Datos (DBA) y Arquitectos.
Meta principal	Evitar redundancia y asegurar integridad (Normalización).	Maximizar la velocidad de respuesta (Eficiencia).



# ■ Motor de base de datos (DBMS)



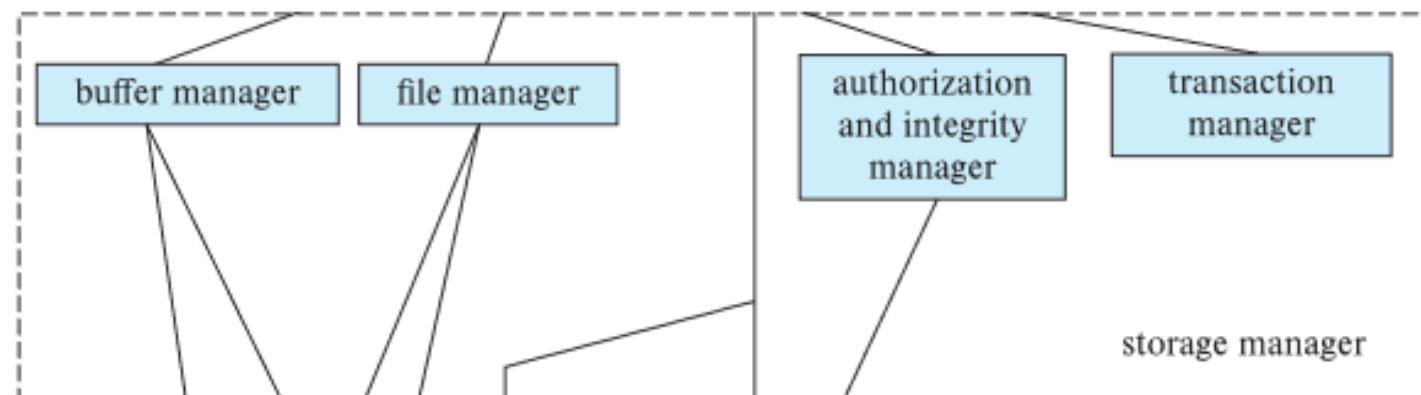
# ■ Motor de base de datos (DBMS)

- Un sistema de bases de datos está dividido en módulos que se encargan de cada una de las responsabilidades del sistema en su conjunto.
- Los componentes funcionales de un sistema de bases de datos pueden dividirse en:
  - El administrador de almacenamiento (storage manager)
  - El procesador de consultas (query processor)
  - El componente de gestión de transacciones (transaction management component).

# ■ Motor de base de datos (DBMS)

## Storage Manager (Gestor de almacenamiento) – Parte 1

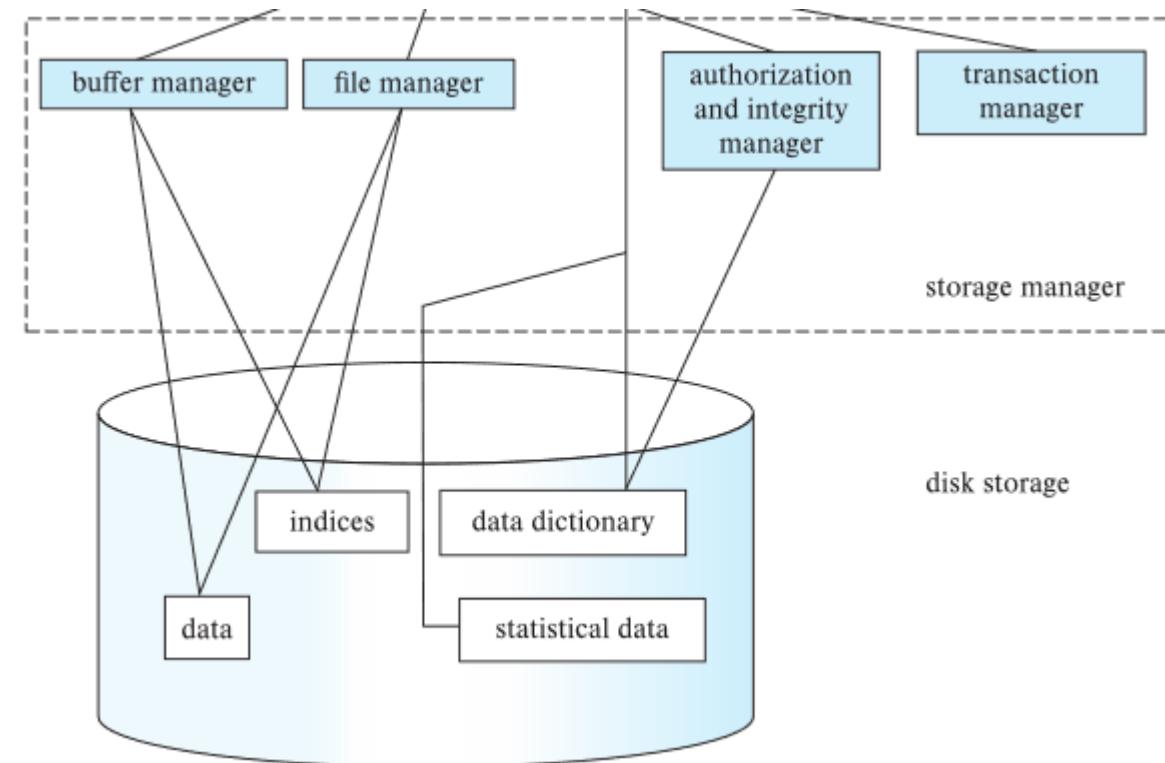
- Este es el modulo de programa que proporciona la interfaz entre el almacenamiento de datos de bajo nivel en la base de datos y los programas de aplicación y consultas enviadas al sistema.
- El administrador de almacenamiento es responsable de las siguientes tareas:
  - Interacción con el administrador de archivos del sistema operativo.
  - Almacenamiento, recuperación y actualización eficiente de los datos.
- Los componentes del administrador de almacenamiento incluyen:
  - Administrador de autorización e integridad.
  - Administrador de transacciones.
  - Administrador de archivos.
  - Administrador de búfer.



# ■ Motor de base de datos (DBMS)

## Storage Manager (Gestor de almacenamiento) – Parte 2

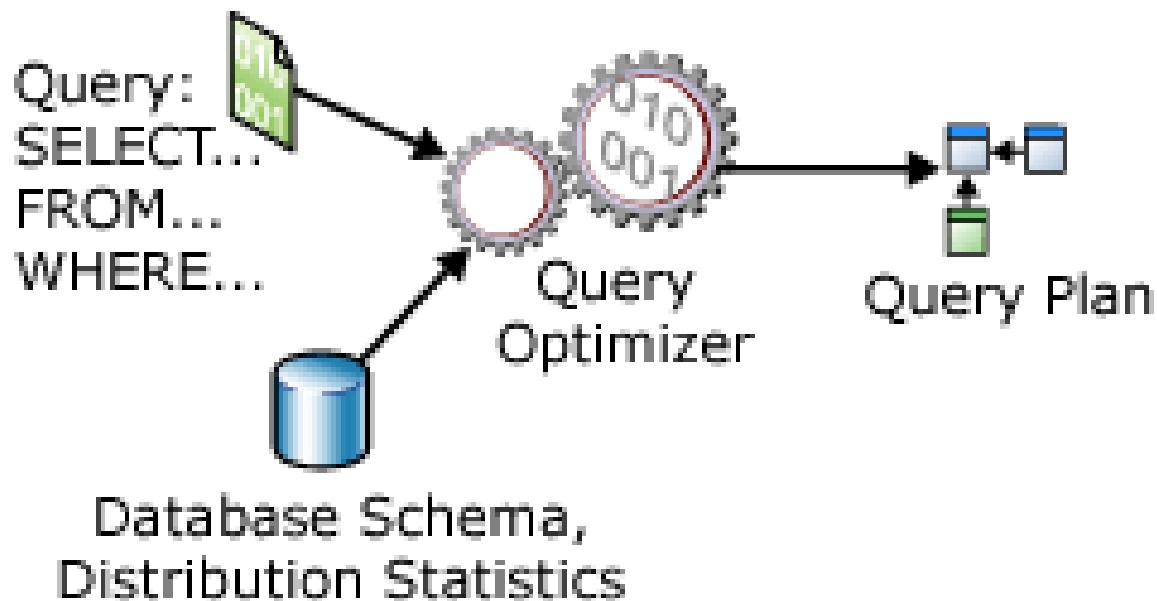
- El administrador de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema:
  - **Archivos de datos:** almacenan la base de datos propiamente dicha.
  - **Diccionario de datos:** almacena metadatos sobre la estructura de la base de datos, en particular el esquema de la base de datos.
  - **Índices:** pueden proporcionar acceso rápido a los elementos de datos. Un índice de base de datos contiene punteros a aquellos elementos de datos que tienen un valor particular.



# ■ Motor de base de datos (DBMS)

## Procesador de Consultas (Query Processor)

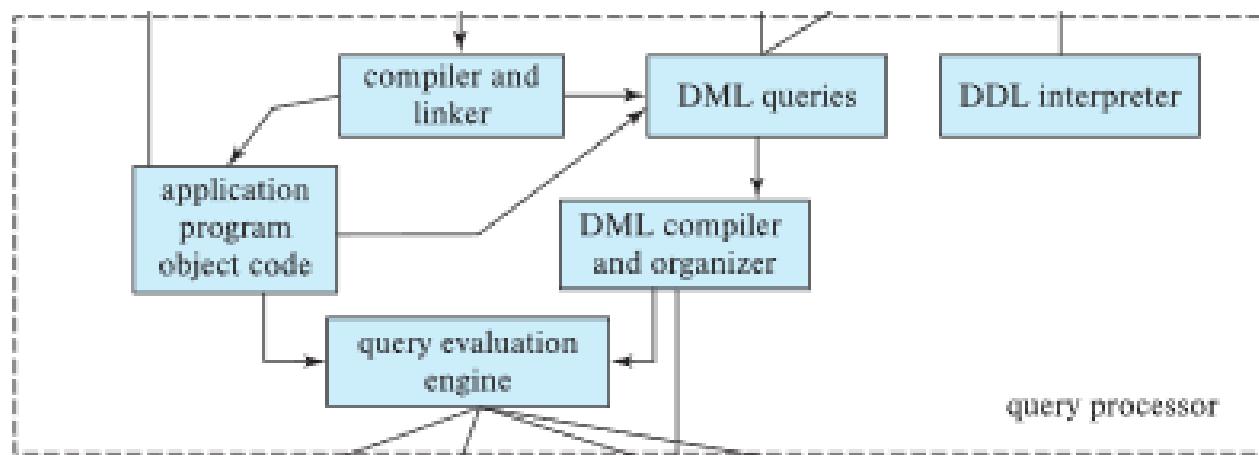
- El procesador de consultas es el componente del DBMS encargado de determinar la forma más eficiente de ejecutar las peticiones de los usuarios.
- Su función principal es transformar una consulta (generalmente escrita en un lenguaje como SQL) en una serie de operaciones que el motor de almacenamiento pueda ejecutar.



# ■ Motor de base de datos (DBMS)

## Procesador de Consultas (Query Processor)

- Los componentes del procesador de consultas incluyen:
  - **Intérprete DDL:** interpreta las sentencias DDL y registra las definiciones en el diccionario de datos.
  - **Compilador DML:** traduce las sentencias DML en un lenguaje de consulta a un plan de evaluación que consiste en instrucciones de bajo nivel que el motor de evaluación de consultas puede entender.
    - El compilador DML realiza la optimización de consultas; es decir, selecciona el plan de evaluación de menor costo entre las diferentes alternativas.
  - **Motor de evaluación de consultas (Query evaluation engine):** ejecuta las instrucciones de bajo nivel generadas por el compilador DML



# ■ Motor de base de datos (DBMS)

## Procesador de Consultas (Query Processor)

### 1. Traducción (Parsing):

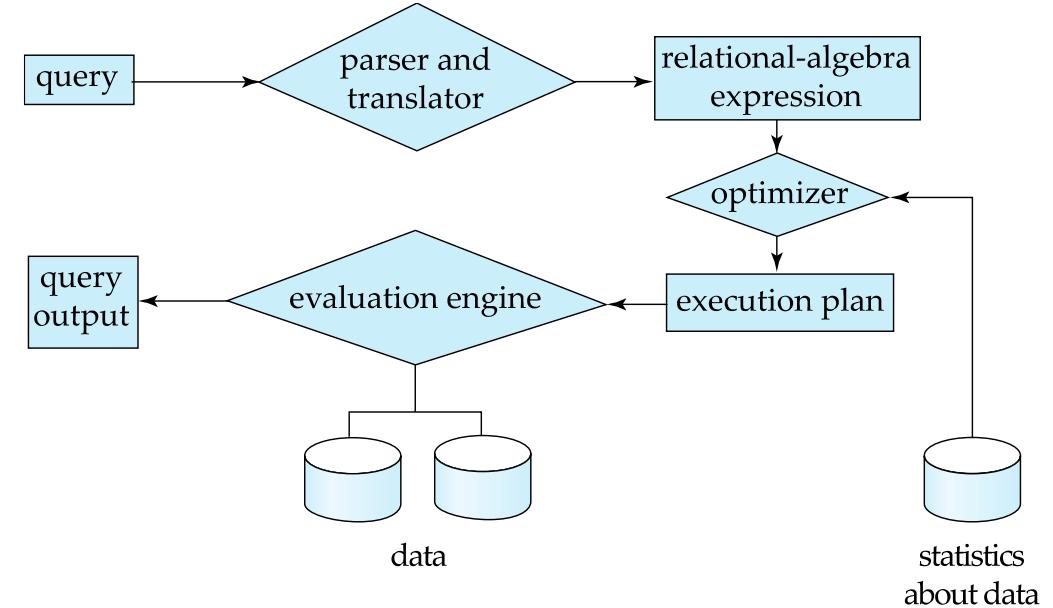
- Convierte el SQL (Abstracción) en Álgebra Relacional (Modelo Lógico).
- Verifica sintaxis y reglas del sistema

### 2. Optimización (El Cerebro):

- Analiza múltiples rutas de ejecución para una misma consulta.
- Usa estadísticas para elegir el plan de menor costo (más rápido).
- Transforma el "Qué" (Lógico) en un "Cómo" (Físico).

### 3. Evaluación (Ejecución):

- El motor ejecuta el plan optimizado sobre los datos reales.
- Coordina con el Gestor de Almacenamiento para acceder al disco. usuarios.



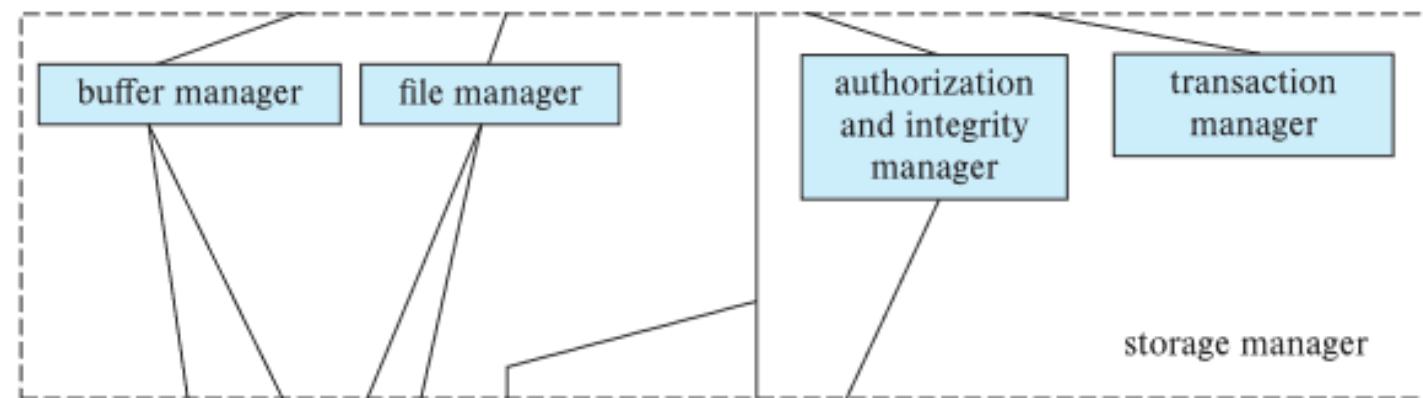
**En resumen:**

El SQL es la **petición**, el optimizador es la **estrategia** y el motor de evaluación es la **acción**.

# ■ Motor de base de datos (DBMS)

## Transaction manager (gestor de transacciones)

- Una **transacción** es un conjunto de operaciones que realiza una única función lógica en una aplicación de bases de datos.
- El **gestor de transacciones** garantiza que la base de datos permanezca en un estado consistente (correcto) a pesar de fallos del sistema (por ejemplo, cortes de energía y fallos del sistema operativo) y fallos de las transacciones.
- El administrador de control de concurrencia controla la interacción entre transacciones concurrentes, para asegurar la consistencia de la base de datos.



# ■ Fin clase 1

UNIVERSIDAD  
DE ANTIOQUIA



# ■ Referencias

- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). **Database System Concepts (7th Edition)**. McGraw-Hill – Capítulo 1
- <https://15445.courses.cs.cmu.edu/spring2026/>
- <https://www.mongodb.com/resources/basics/databases/nosql-explained>
- <https://blog.bytebytego.com/p/sql-vs-nosql-choosing-the-right-database>
- <https://cs186berkeley.net/notes/note17/>
- <https://www.ibm.com/es-es/think/topics/database>
- [https://docs.oracle.com/cd/A83908\\_02/NT816EE/DOC/server.816/a76994/toc.htm](https://docs.oracle.com/cd/A83908_02/NT816EE/DOC/server.816/a76994/toc.htm)
- <https://learn.microsoft.com/en-us/sql relational-databases/query-processing-architecture-guide?view=sql-server-ver17>

# UNIVERSIDAD DE ANTIOQUIA

Curso de estructuras de datos  
Clase 1 – Introducción y conceptos claves