Teo: Introducción

☑ Archive ☐

Introducción

Las **bases de datos** son bases que contienen información interrelacionada de una organización. Entre sus **aplicaciones** tenemos:

- Consultar la información
- Alterar la información

La **organización lógica** de las bases de datos es estructurada usando **esquemas**. El uso de esquemas es similar al uso de tipos y variables en lenguajes de programación.

Ejemplo: en un banco, el esquema de una base de datos consiste de *clientes, cuentas* y la relación *tiene_cuenta* entre ellos. Los clientes tienen **atributos** *nombre y DNI*, las cuentas tienen atributos *número y saldo*.

Luego, diremos que las **instancias** son el contenido actual de la base de datos en un momento del tiempo.

Ejemplo: *clientes* tiene los clientes Juan Pérez con DNI 333 y Mariela González con DNI 444; *cuentas* tiene las cuentas 1111 de 1000$ y 2222 de 500$; y *tiene_cuentas* dice que Juan Pérez tiene la cuenta 1111 y Mariela González tiene la cuenta 2222.

⏩ Entonces, tenemos que una base de datos es estructurada a través de esquemas o tablas que contienen datos o instancias que pueden ser consultadas o manipuladas.

Modelo Relacional

El modelo relacional de base de datos es el que se basa en la construcción de tablas y en las relaciones. Como vimos en el laboratorio, **las columnas son los atributos** y **las filas son tuplas**.

Como vimos más atrás, los datos en las bases de datos pueden consultarse o manipularse. Para ambas acciones, se utiliza un lenguaje de consulta llamado **SQL**

En este modelo tenemos el lenguaje **SQL** que nos permite entre muchas cosas, **consultar** los datos. En SQL, las consultas son **expresiones que describen una colección de datos deseada**.

Ejemplo:

Ejemplo: (para la tabla instructor) hallar una expresión para: Encontrar salario y nombre de instructores que ganan más de $ 50000.

Ejemplo de consulta en SQL (para el ejemplo anterior)

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

Aparte de SQL, existen otros lenguajes de consulta **puros** como álgebra relacional, cálculo de tuplas, etc. En la materia veremos una **variación más expresiva del álgebra relacional** llamada álgebra de tuplas.

Veremos como el sistema gestor de base de datos procesa consultas para el modelo relacional, tarea que se facilita si traducimos una consulta SQL al álgebra relacional (o álgebra de tuplas) y luego la procesamos.

Buen Diseño de BD Relacional

Un buen diseño de una base de datos relacional equivale a un buen esquema. Algunas pautas para hacer buenos esquemas son:

- No almacenar toda la información en una sola tabla

  (ejemplo: esquema Universidad = (instructorID, nombre, nombreDepto, salario, estudianteID)

- Descomprimir esquemas grandes en unos más chicos.

(ejemplo: Univ = (instructorID, estudianteID), Instructor = (instructorID, nombre, nombreDepto, salario)

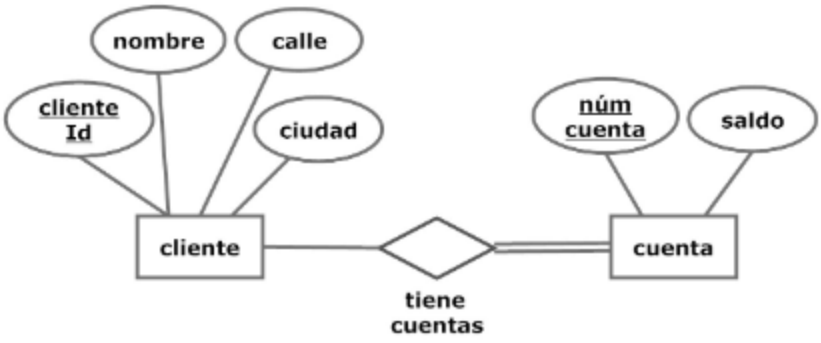
- Seguir la **teoría de normalización**.

Estudiando más la estructura lógica de las bases de datos, tenemos que las mismas se **modelan** como una **colección de entidades y relaciones**

- Entidad: objeto en organización distinguible de otros objetos, descrito por medio de atributos.
- Relación: asociación entre entidades.

La representación gráfica es usando **diagramas de entidad-relación**.

(ejemplo: entidades cliente, cuenta y relaciones tiene_cuentas)



Sistema Gestor de BD

Los sistemas gestores son conjuntos de software que se usan para administrar, almacenar, organizar y gestionar de forma eficiente grandes cantidades de datos de forma estructurada. Permiten a usuarios y aplicaciones acceder y manipular los datos de una manera controlada y segura.

- Gestor/Motor de almacenamiento

  Se encarga de administrar la forma en que los datos se almacenan físicamente en el sistema de almacenamiento subyacente, como discos duros. Controla la lectura, escritura y manipulación de los datos en el nivel físico. Se usan **archivos** e **indices**

para la interacción con los datos. Provee una interfaz para los datos a nivel físico para ser usada por los programas de app y consultas enviadas al sistema.

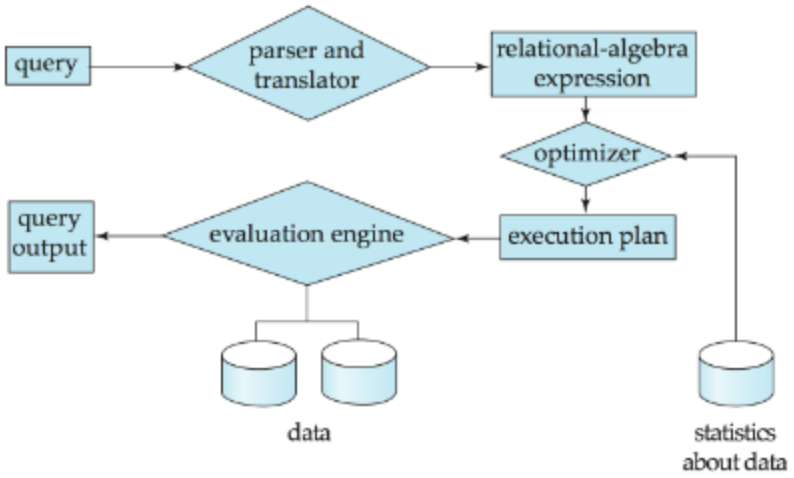
Es el **intermediario** entre la capa física y la capa lógica

- Procesamiento de consultas

  Se encarga de interpretar y ejecutar las consultas enviadas al sistema por usuarios o aplicaciones. Traduce las consultas en operaciones físicas que el gestor de almacenamiento puede comprender y ejecutar.

  - Realiza el parsing de la consulta y su **traducción** (ejemplo: álgebra relaciona)
  - Optimización: encuentra la manera más eficiente para obtener la información descrita por la consulta.
  - Evaluación: busca la data que queremos usando el plan de optimización.

  Por qué tenemos que se puede optimizar una consulta? Ésto es porque la difierencia de costo entre una buena y una mala manera de evaluar una consulta puede ser muy grande. Es por eso que se puede **estimar el costo** de las operaciones usando **información estadística**.



- Gestor de transacciones

  Este componente gestiona las transacciones, que son secuencias de operaciones que se consideran una unidad indivisible. Garantiza la integridad y consistencia de los datos incluso en situaciones de fallas o interrupciones.

> 💡 Una **transacción** es una colección de operaciones que realiza una función lógica simple en una app de BD. ES una unidad de ejecución que accede y posiblemente actualiza varios items de datos.

Compuesto de:

- Gestor de recuperaciones

  Asegura atomicidad ante fallas en el sistema. Para asegurar atomicidad, **la falla de una transacción no debe tener efecto en el estado de la base de datos** (o sea, la BD debe restaurarse al estado en el que estaba antes de que la transacción comenzara su ejecución).

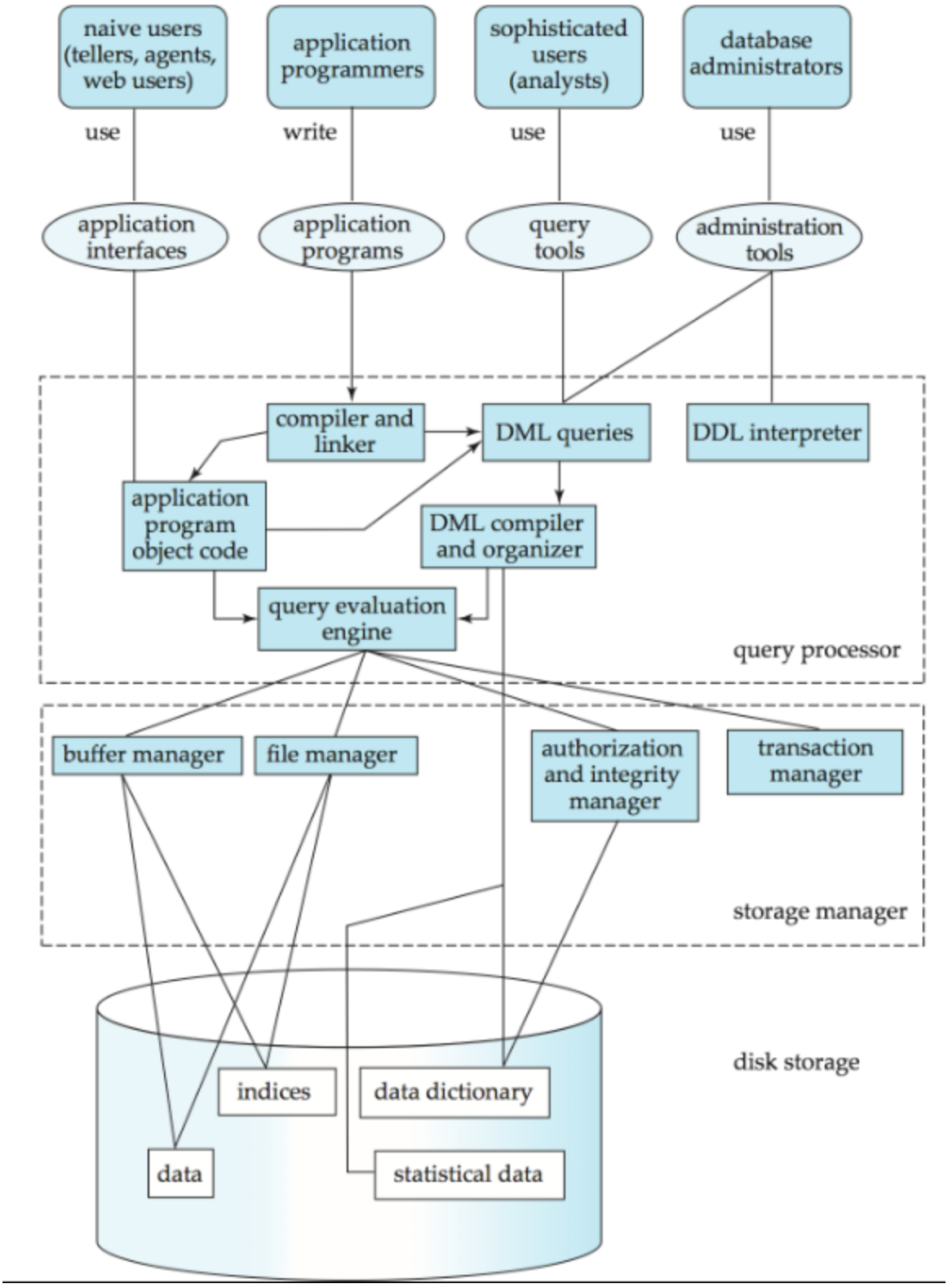
- Gestor de concurrencia de transacciones

  controla la interacción entre transacciones concurrentes para asegurar la consistencia de la BD.

Se usa la **planificación** para indicar el orden cronológico en el cual las instrucciones de transacciones concurrentes son ejecutadas.

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

Arquitectura de un SGBD


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data