

# Bases de datos

## Introducción



# ¿Qué es una BD?

- Una **base de datos (BD)** contiene información.
  - P.ej. de una empresa o institución pública.
  - Generalmente se trata de datos interrelacionados.
  - P.ej. auto, persona, dueño de
- Las BD se usan ampliamente
  - Empresas, instituciones públicas, etc.
- **Aplicaciones de BD:**
  - Permiten navegar/consultar las informaciones
  - Permiten alterar las informaciones

# Esquemas e Instancias

- Conceptos similares a tipos y variables en lenguajes de programación.
- **Esquema**: se usa para describir la **estructura** de la BD.
  - **Ejemplo**: la BD de un banco consiste de *clientes*, *cuentas* y la relación *tiene\_cuentas* entre ellos. Los clientes tienen *nombre* y *DNI*; las cuentas tienen *número* y *saldo*.

# Esquemas e Instancias

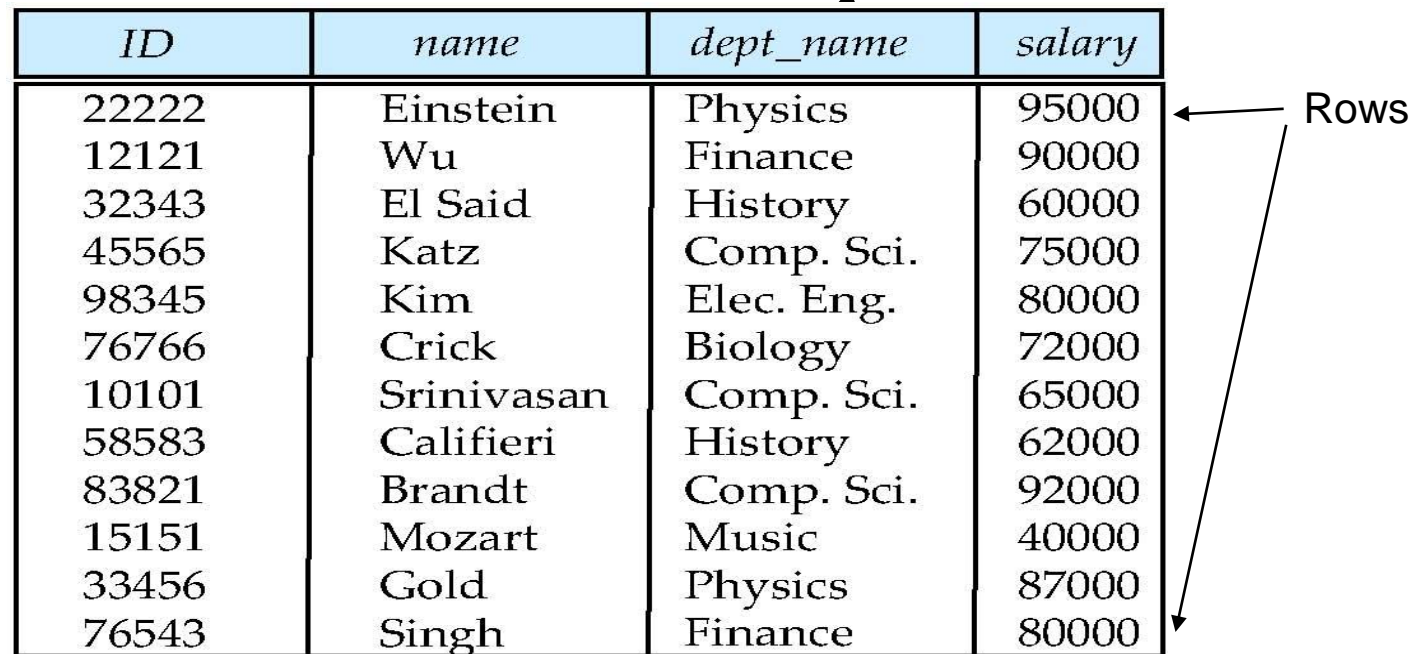
- **Instancia:** contenido actual de la BD en un momento del tiempo.
  - **Ejemplo:** *clientes* tiene los clientes Juan Pérez con DNI 333 y Diego González con DNI 444; *cuentas* tiene las cuentas 1111 de 1000\$ y 2222 de 500\$; y *tiene\_cuentas* dice que Juan Pérez tiene la cuenta 1111 y Diego González tiene la cuenta 2222.

# Actividad

- Usted es un **almacenero** que debe llevar registro de todo lo que sucede en el almacén:
  - el precio de las cosas, el stock actual, la lista de proveedores, . . .
- Decide llevar todo el registro en **cuadernos**. Determine con cuidado que información precisa registrar en cada cuaderno (esquema).
- La organización en cuadernos debe permitir contestar las siguientes **preguntas**:
  - ¿Como sabe que productos vende un determinado proveedor?
  - ¿como obtiene el stock de un producto?
  - ¿Como lleva el registro de las compras que le realizaron?
- Además considerar:
  - ¿Que tan eficiente es su proceso?
  - ¿Tiene mucha redundancia de datos? (i.e. informaciones que se repiten aquí y allá innecesariamente)

# 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).

# Lenguajes de Consultas

- Una **consulta** en una base de datos es una expresión que describe una colección de datos deseada.
- **Ejemplo:** (para la tabla instructor) hallar una expresión para: Encontrar salario y nombre de instructores que ganan más de \$ 50000.
- Para expresar consultas se usan **lenguajes de consulta**.
- Para consultar los datos en modelo relacional es muy usado en la industria el **lenguaje de consultas SQL**.
  - Una consulta aquí se refiere a ciertas tablas del esquema de datos relacional.
- **Ejemplo de consulta en SQL** (para el ejemplo anterior)

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

# Lenguajes de Consultas

- Para el modelo relacional existen lenguajes de consulta puros como **álgebra relacional, cálculo de tuplas**, etc.
  - Son más sencillos y se concentran en menos aspectos.
  - En la materia veremos una variación más expresiva del álgebra relacional llamada **álgebra de tuplas**.
- Veremos cómo el sistema gestor de BD procesa consultas para el modelo relacional.
  - Esta tarea se facilita si traducimos una consulta SQL al álgebra relacional – o álgebra de tuplas - y luego la procesamos.
- Hay **modelos de datos no relacionales** que tienen también lenguajes de consulta.
  - P.ej. Los lenguajes Mongo-DB, XQuery, etc.
  - En el taller van a estudiar Mongo DB.



# 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.
  - Esquemas relacionales se describen con:  
*Nombre de esquema = lista de nombres de atributo.*
- **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:** dos estudiantes con el mismo instructor

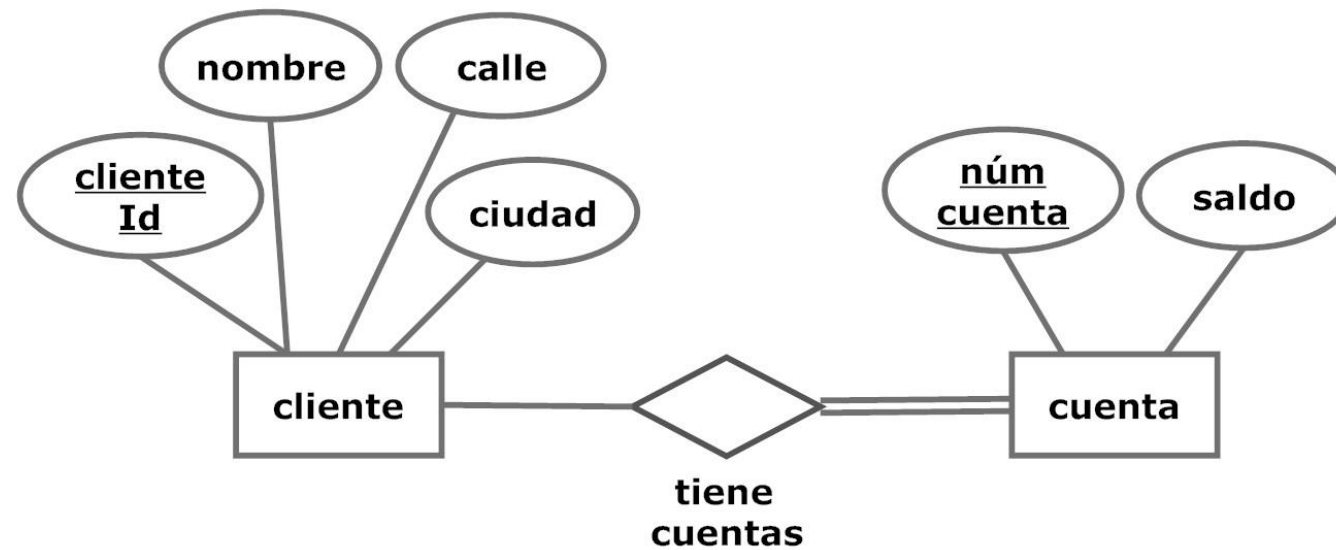
# Diseño de BD relacional

- **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 en esquemas mas 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.

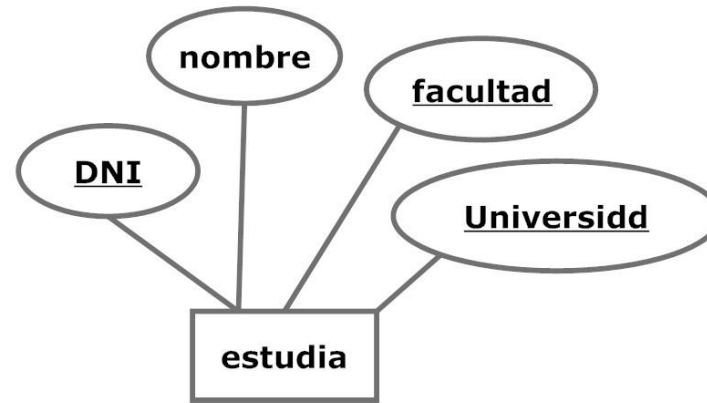
# Diseño de Entidad-Relación

- 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. Descrito por medio de atributos.
  - **Relación**: asociación entre entidades.
  - Representado diagramáticamente usado un **diagrama de entidad-relación**
- **Por ejemplo**: entidades *cliente*, entidades *cuenta*, y relaciones *tiene\_cuentas*.
  - Toda *cuenta* es de algún *cliente*

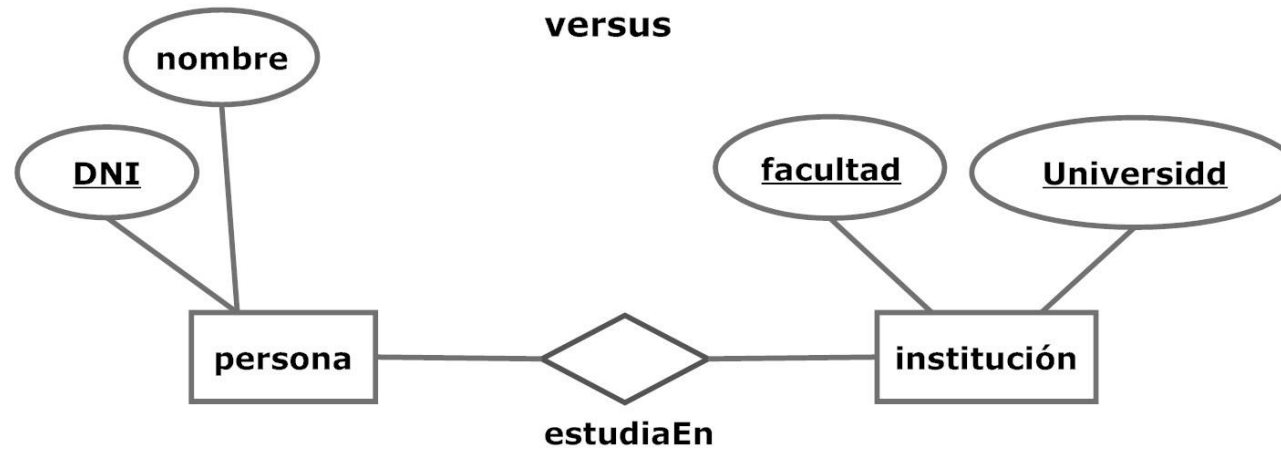
# Ejemplo de diagrama de entidad-relación



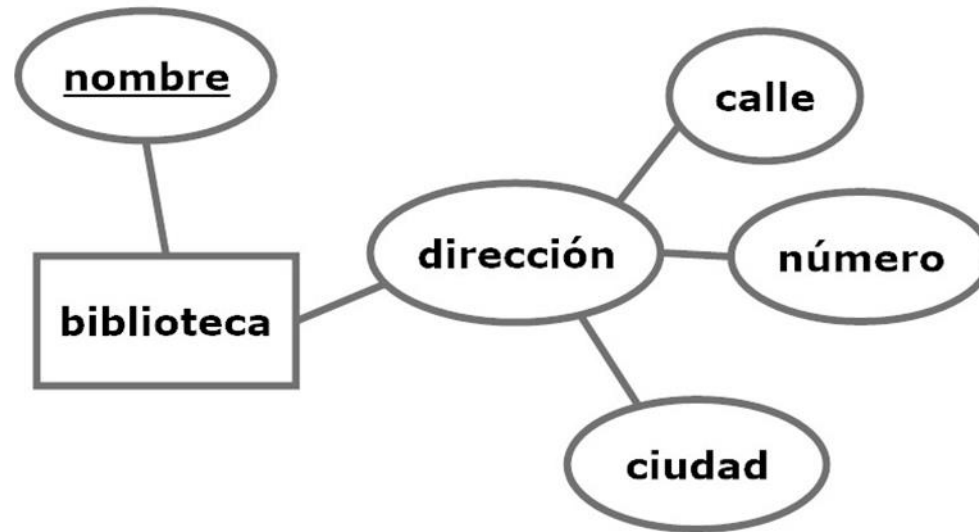
# Decisiones de Diseño



versus



# Reducción a Tablas



Se mapea a:

*biblioteca(nombre, calle, número, ciudad)*

# Actividad

- Decide informatizar el almacén. Tiene a su disposición un lenguaje de programación (Ej: C, Python, Java, etc.). Piense desde el punto de vista de los datos cómo haría para implementar un sistema que le permita operar sobre ellos de la misma forma que pensó con los cuadernos en el ejercicio anterior.
- Va a modelar los módulos del sistema, uno para cada tarea específica sobre los datos, desde la capa más abstracta hasta la más baja (sin llegar a acceso a disco de bajo nivel, ni a tipos de datos específicos como árboles o listas)
- Piense en que el sistema debe manipular los datos de forma de no estar atado a los datos específicos del almacén, que pueden cambiar. Es decir, no debería aparecer en sus módulos ninguna referencia a los datos del almacén. En particular, se pide lo siguiente:
  1. Cree un diagrama con los módulos, con sus referencias. Los módulos deben reflejar de algún modo los tipos de operaciones que realizó en el ejercicio anterior: creación de cuadernos, búsqueda y actualización de información, etc.
  2. El almacén creció, y ahora tiene dos o más cajas. Decide mantener una computadora como servidor y las cajas como clientes. Para ello puede asumir una librería/módulo que se ocupe de los aspectos de red. ¿Qué nuevos módulos necesita? ¿Qué pasa cuando se venden dos veces el mismo producto desde las distintas cajas?

# Sistema gestor de BD

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

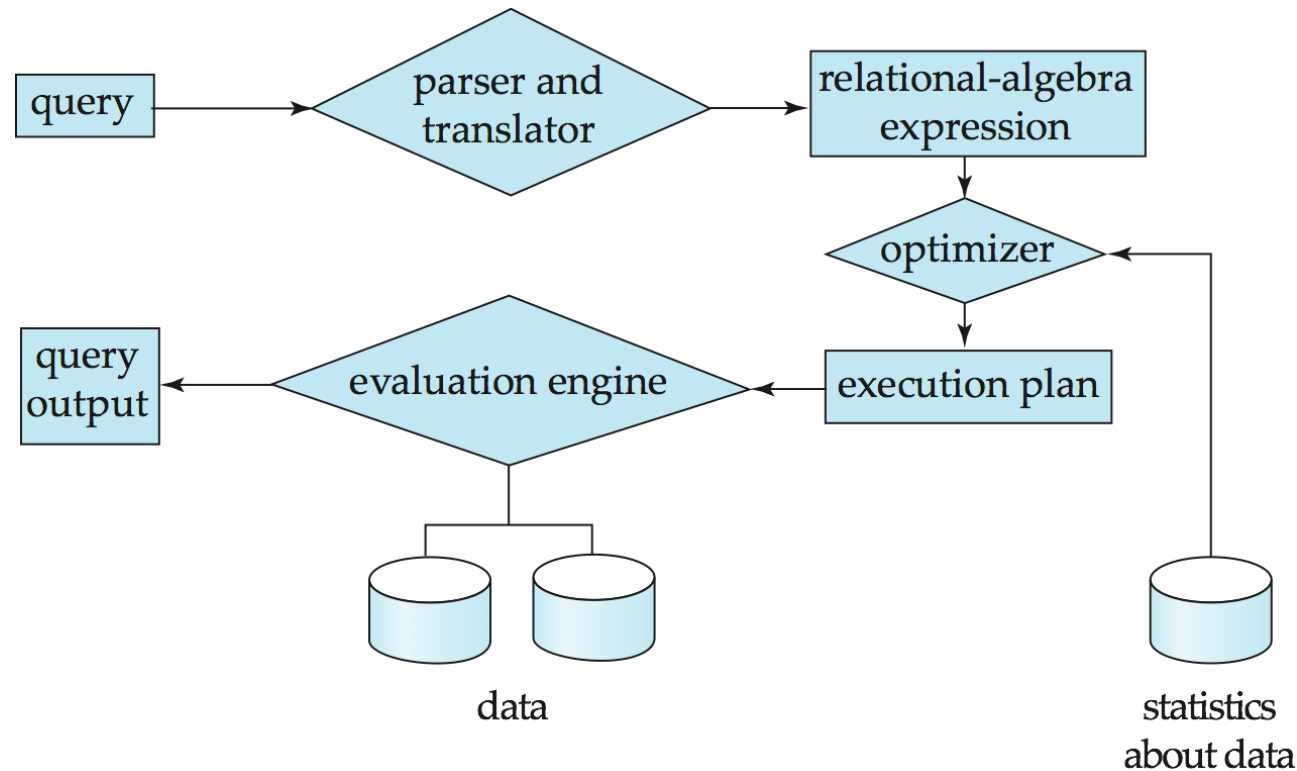


# Gestión del almacenamiento

- Los datos deben ser organizados en archivos con **estructuras apropiadas** de modo que su acceso, modificación y retorno sea eficiente.
- A **nivel físico** se explica cómo los registros de datos son almacenados en archivos.
- Para acceso, modificación y almacenamiento eficiente a los datos se pueden usar **índices** (unas estructuras de acceso especiales).
- A **nivel lógico** se describen los datos almacenados en la BD y las relaciones entre ellos.
- El **gestor de almacenamiento** provee una interfaz para los datos a nivel físico para ser usada por los programas de aplicación y consultas enviadas al sistema. Se ocupa de: acceso al almacenamiento, organización en archivos de los datos, indexado.

# Procesamiento de consultas

1. **Parsing** de la consulta y su **traducción** (p.ej. a algebra relacional)
2. **Optimización**:
  - Encontrar la manera “más eficiente” (o plan) para obtener la información descrita por la consulta.
3. **Evaluación** (siguiendo el plan optimizado)



# Procesamiento de consultas

- Una consulta se puede evaluar de **maneras alternativas**
  - Expresiones equivalentes
  - Diferentes algoritmos para cada operación
- La **diferencia de costo** entre una buena y una mala manera de evaluar una consulta puede ser enorme.
- Es necesario **estimar el costo** de las operaciones.
  - Depende de **información estadística** acerca de las relaciones que la BD debe mantener.
  - Hace falta estimar estadísticas para **resultados intermedios** para computer el costo de expresiones complejas.

# Transacciones

- Preguntas importantes:
  - ¿Qué pasa si falla el SGBD?
  - ¿Qué pasa si más de un usuario actualiza concurrentemente los mismos datos?
- Una **transacción** es una colección de operaciones que realiza una función lógica simple en una aplicación de BD
  - Una transacción es una unidad de ejecución que accede y posiblemente actualiza varios ítems de datos.
- **Ejemplo:** Transacción para transferir \$50 de la cuenta *A* a la cuenta *B*:
  1. **read**(*A*)
  2.  $A := A - 50$
  3. **write**(*A*)
  4. **read**(*B*)
  5.  $B := B + 50$
  6. **write**(*B*)

# 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 SO) y fallas de transacciones.
- **Ejemplo:** falla de transacción de transferencia bancaria: Sacar dinero de una cuenta sin ponerlo en otra es un error.
- **Problema:** ¿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.

# Transacciones

- Asegurar la atomicidad es responsabilidad del SGBD, específicamente del **gestor de recuperaciones**.
- En la ausencia de fallas todas las transacciones completan exitosamente y la atomicidad se logra fácilmente.
- Debido a los varios tipos de falla una transacción puede no completarse exitosamente.
- 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 que estaba antes que la transacción comenzara su ejecución.

# 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 ( $A$ )<br>$A := A - 50$<br>write ( $A$ )           | read ( $A$ )<br>$temp := A * 0.1$<br>$A := A - temp$<br>write ( $A$ ) |
| read ( $B$ )<br>$B := B + 50$<br>write ( $B$ )<br>commit | read ( $B$ )<br>$B := B + temp$<br>write ( $B$ )<br>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 para asegurar la consistencia de la BD.
- El **gestor de transacciones** consiste del gestor de recuperaciones y del gestor de concurrencia de transacciones.



# Arquitectura de un SGBD

