TEMA III

Sistemas de Gestión de Bases de Datos

Sistemas de Gestión de Bases de Datos (SGBD)

- **Objetivos:**
  - conocer la arquitectura ANSI/SPARC para sistemas de gestión de bases de datos (SGBD).
  - aprender el concepto de independencia de datos.
  - aprender mecanismos y estrategias para el control de la integridad (calidad) y seguridad (privacidad) en bases de datos (BD).
  - Conocer las organizaciones de ficheros que sirven de soporte a la implementación de relaciones en las bases de datos relacionales.

Sistemas de Gestión de Bases de Datos (SGBD)

Temario

3.1 Sistema de gestión de bases de datos: componentes y funciones.

3.2 Independencia de datos.

3.3 Integridad.

3.3.1 Concepto de transacción. Procesamiento de transacciones.

3.3.2 Integridad semántica.

3.3.3 Accesos concurrentes.

3.3.4. Reconstrucción de la base de datos.

3.4 Seguridad.

3.4.1 Control de usuarios.

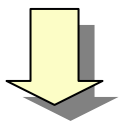
3.4.2 Control de accesos permitidos.

3.5 Implementación de Bases de Datos Relacionales.

3.1.- Sistema de Gestión de Bases de Datos

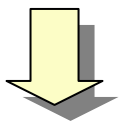
SGBD: Software que permite la creación y manipulación de bases de datos.

SGBD



Se basa

modelo de datos (modelo relacional)



Se compone

estructuras de datos y operadores asociados

3.1.1.- Componentes y funciones del SGBD

Los SGBD permiten:

- descripción unificada de los datos e independiente de las aplicaciones
- independencia de las aplicaciones respecto a la representación física de los datos
- definición de vistas parciales de los datos para distintos usuarios
- gestión de la información
- integridad y seguridad de los datos

3.1.1.- Componentes y funciones del SGBD.

Objetivos de técnicas BD

- descripción unificada e independiente de los datos
- independencia de las aplicaciones
- definición de vistas parciales

Funciones SGBD

Definición de datos a varios niveles:

- esquema lógico
- esquema interno
- esquemas externos

Componentes SGBD

Lenguajes de definición de esquemas y traductores asociados

3.1.1.- Componentes y funciones del SGBD.

Objetivos de técnicas BD	Funciones SGBD	Componentes SGBD
Gestión de la información	Manipulación de los datos: • consulta • actualización Gestión y administración de la base de datos	Lenguajes de manipulación y traductores asociados Herramientas para: • reestructuración • simulación • estadísticas • impresión

3.1.1.- Componentes y funciones del SGBD.

Objetivos de técnicas BD

Integridad y seguridad de los datos

Funciones SGBD

Control de:

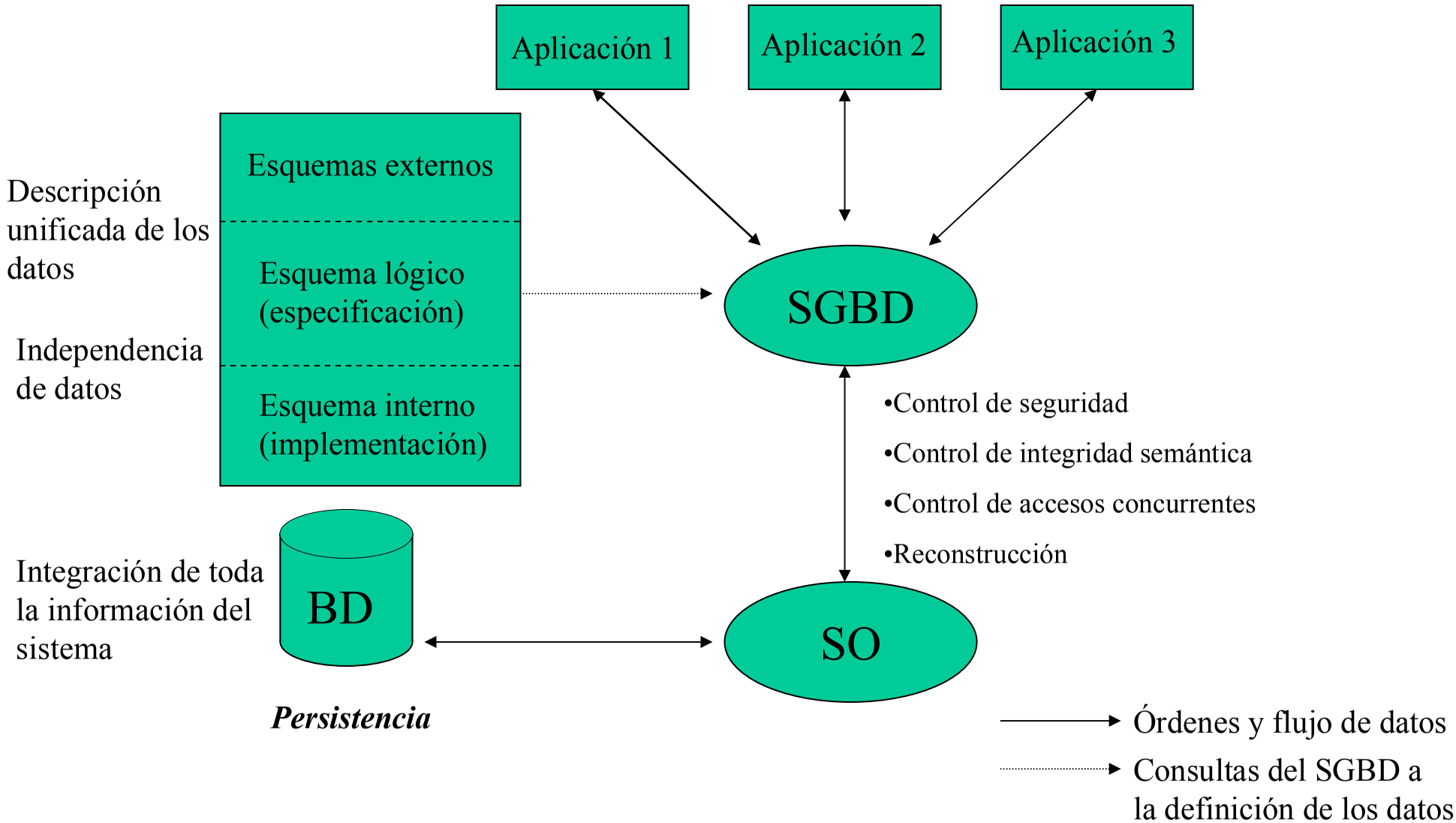
- integridad semántica
- accesos concurrentes
- reconstrucción en caso de fallo
- seguridad (privacidad)

Componentes SGBD

Herramientas para:

- control integridad
- reconstrucción
- control seguridad

3.1.2.- Esquema de acceso del SGBD a los datos



3.1.2.- Esquema de acceso del SGBD a los datos

Esquema externo aplicación 1:

```
CREATE VIEW Administrativo (dni, nombre, salario_men)
    AS SELECT dni, nombre, salario/14
        FROM Empleado
        WHERE tipo='AD'
```

Esquema lógico:

Empleado(dni, nombre, dirección, salario, tipo)
CP: {dni}

Esquema Interno:

Fichero ordenado *Empleado* con índice primario sobre el campo *dni* en el camino *h:/disco1/gerencia*

3.1.2.- Esquema de acceso del SGBD a los datos

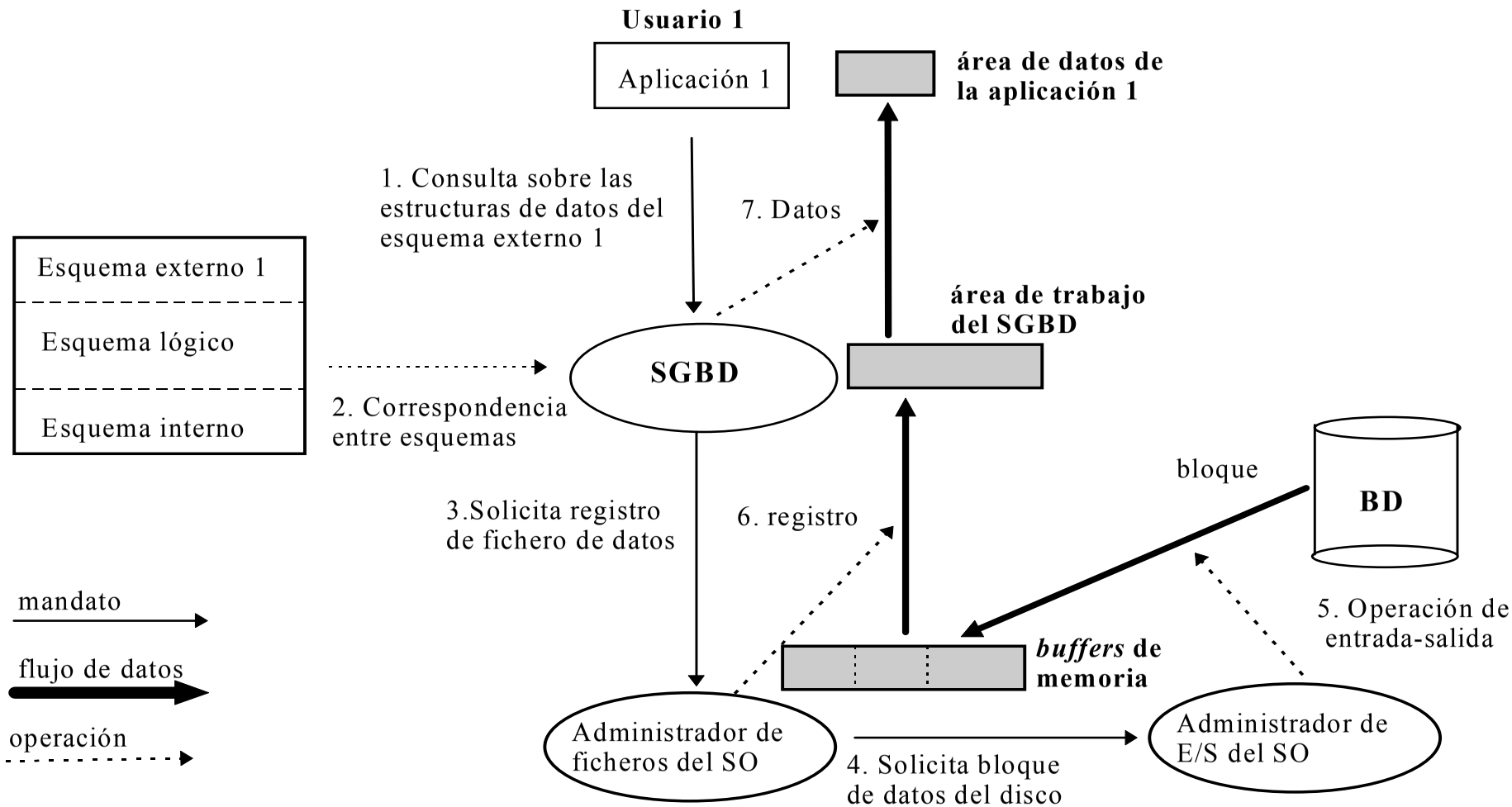
Aplicación 1: accede a la información a través del esquema externo 1

SELECT nombre, salario_men
FROM Administrativo
WHERE dni = *parámetro*

SGBD: control del acceso y resolución de la operación pedida

SO: Manipulación de los controladores de los dispositivos de memoria secundaria

3.1.2.- Esquema de acceso del SGBD a los datos



3.1.2.- EJEMPLO. Especificación

Una pequeña inmobiliaria desea mantener información sobre los edificios cuya venta gestiona. Se quiere saber:

- De cada edificio, el código, la ubicación, el distrito, el propietario, el precio solicitado por éste y el agente encargado de la venta si ya está asignado.
- De cada propietario, el código, nombre y teléfono.
- De cada agente el DNI, el nombre, la comisión por cada venta, los años de antigüedad y el teléfono.

Las restricciones que deben cumplirse son las siguientes:

- La comisión de un agente no puede exceder el 3% si su antigüedad es menor de 3 años.
- No se quiere tener información de propietarios si no se tiene al menos un edificio para la venta.

Grupos de trabajo:

- El personal de administración tiene acceso a toda la información comentada.
- El jefe de la inmobiliaria sólo desea tener información referente a los edificios con precio solicitado superior a 5 millones. De cada uno desea el código, la ubicación, y el distrito.
- El jefe es el único que puede modificar la información de los agentes.

3.1.2.- EJEMPLO. Esquema Lógico (SQL)

```
CREATE SCHEMA Inmobiliaria
CREATE TABLE Edificios
    (Código d_cod PRIMARY KEY,                    Ubicación d_ubi NOT NULL,
     Distrito d_dis NOT NULL,                  Precio d_pre NOT NULL,
     Dni_age d_dni FOREIGN KEY REFERENCES Agente
          ON UPDATE CASCADE, ON DELETE NO ACTION
     Dueño d_cod NOT NULL,  FOREIGN KEY(Dueño) REFERENCES Propietario (cod)
          ON UPDATE CASCADE ON DELETE CASCADE)
CREATE TABLE Propietarios
    (Cod d_cod PRIMARY KEY, Nombre d_nom NOT NULL, Teléfono d_tel NOT NULL)
CREATE TABLE Agentes
    (Dni_age d_dni PRIMARY KEY,  Comisión d_com, Años d_años NOT NULL,
     Tel d_tel NOT NULL, CHECK NOT (años < 3 AND comisión > 3))
CREATE ASSERTION no_propet_sin_edificios CHECK NOT EXISTS
    (SELECT * FROM Propietarios WHERE cod NOT IN (SELECT Dueño FROM Edificio))
```

3.1.2.- EJEMPLO. Esquemas Externos (SQL)

GRANT ALL ON Edificios TO PUBLIC;

GRANT ALL ON Propietarios TO PUBLIC;

GRANT SELECT ON Agentes TO PUBLIC;

ESQUEMA EXTERNO DEL JEFE:

```
CREATE VIEW más_de_5 AS
        SELECT código, ubicación, distrito
        FROM Edificios
        WHERE E.precio >= 5000000;
GRANT ALL ON más_de_5 TO Jefe;
GRANT ALL ON Agentes TO Jefe;
```

+ El resto de tablas del esquema lógico (excepto edificios)

ESQUEMA EXTERNO DEL PERSONAL ADMINISTRACIÓN :

Todas las tablas del esquema lógico

3.1.2.- EJEMPLO. Esquema Físico

Edificios :

- Fichero disperso por dni_age
- Índice B+ sobre (distrito + precio)

Propietarios

- Fichero disperso por cod
- Índice B+ sobre nombre

Agentes

- Fichero desordenado (se suponen pocos agentes).

3.1.2.- EJEMPLO. Proceso de Acceso

El jefe se pregunta:

¿código y ubicación de los edificios del distrito 05?

1. La aplicación interpreta la selección del jefe como:

SELECT código, ubicación

FROM más_de_5 WHERE distrito = '05';

2. El SGBD convierte la consulta del esquema externo al esquema lógico:

SELECT código, ubicación

FROM Edificios E

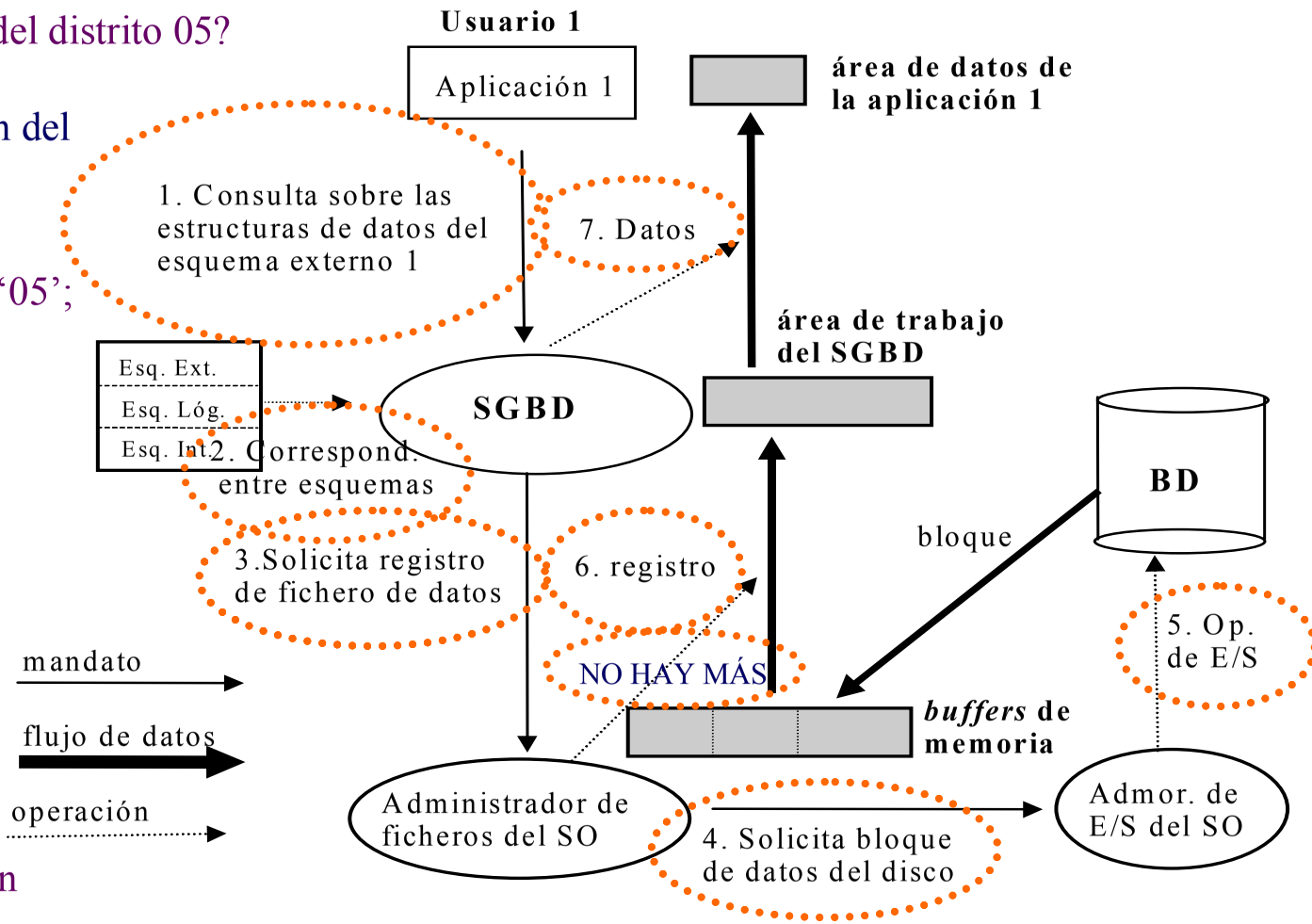
WHERE E.precio >= 5000000

AND E.distrito = '05';

3,4,5,6. REPETIR:

"Leer usando el índice B+ sobre (distrito + precio) el primer registro con distrito = '05' y precio >= 5000000:

HASTA QUE NO HAYA MÁS REGISTROS



7. Eliminar los atributos que no se han solicitado

3.2.- Independencia de datos.

Propiedad que asegura que los programas de aplicación sean independientes de los cambios realizados en **datos que no usan** o en **detalles de representación física de los datos a los que acceden.**

3.2.- Independencia de datos.

Propuesta de arquitectura del grupo de estudio ANSI/SPARC (1977) para los SGBD: plantea la definición de la base de datos a tres niveles de abstracción:

- **Nivel conceptual** $\Rightarrow$ Esquema conceptual
  - descripción de la BD con independencia del SGBD
- **Nivel interno** $\Rightarrow$ Esquema interno
  - descripción de la BD en términos de su representación física
- **Nivel externo** $\Rightarrow$ Esquema externo
  - descripción de las vistas parciales de la BD que poseen los distintos usuarios

3.2.- Independencia de datos.

Debido a que no existe un modelo conceptual generalizado y accesible a los distintos tipos de SGBD, se prefiere distinguir cuatro niveles:

- **Nivel conceptual** $\Rightarrow$ Esquema conceptual
  descripción organizativa de la BD
- **Nivel lógico** $\Rightarrow$ Esquema lógico
  descripción de la BD en términos del modelo de datos del SGBD
- **Nivel interno** $\Rightarrow$ Esquema interno
  descripción de la BD en términos de su representación física
- **Nivel externo** $\Rightarrow$ Esquema externo
  descripción de las vistas parciales de la BD que poseen los distintos usuarios

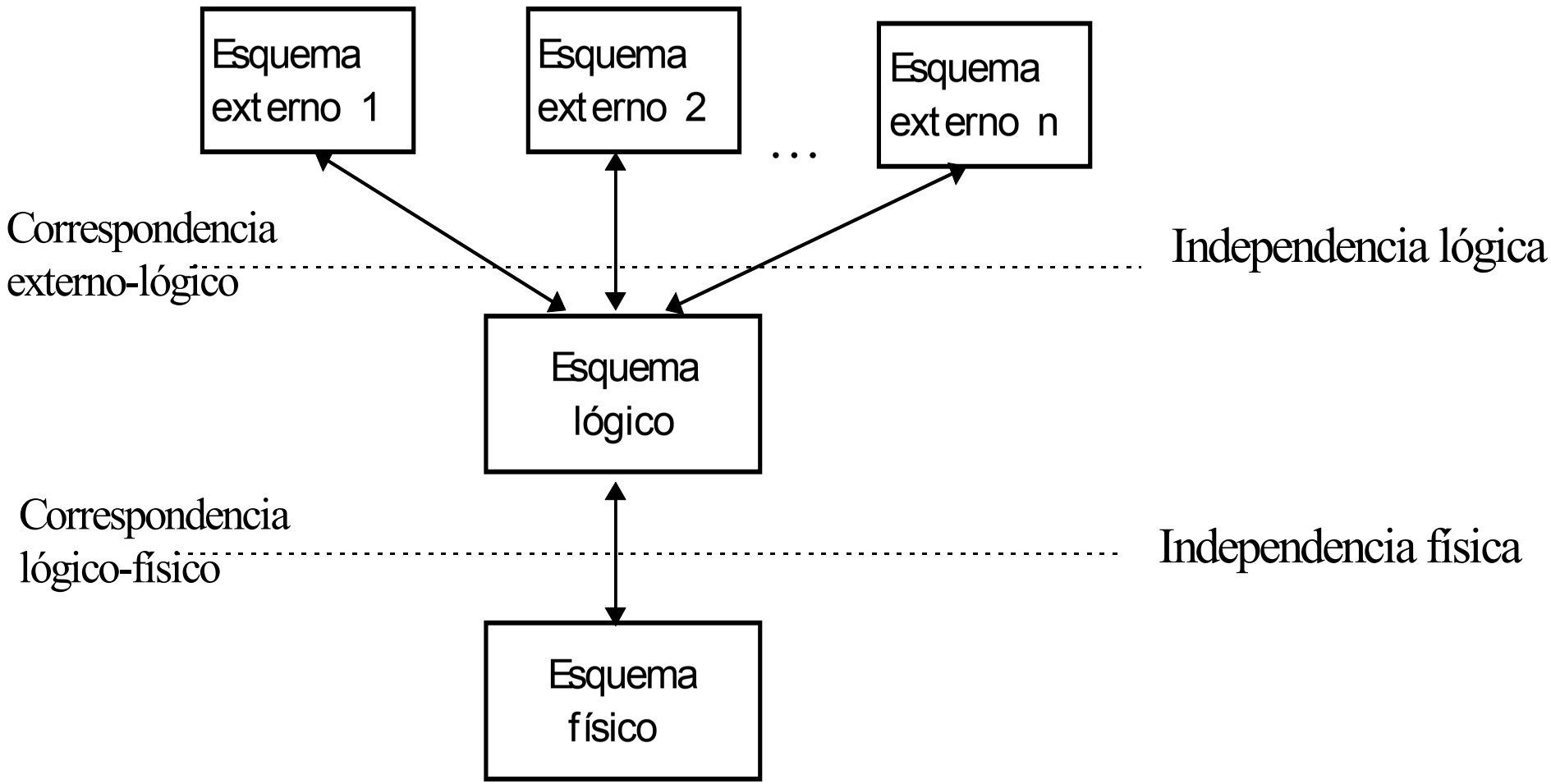
3.2.- Independencia de datos.

Un SGBD que soporte la arquitectura de niveles debe:

- permitir definir los distintos esquemas de la base de datos (a excepción del esquema conceptual)
- establecer las correspondencias entre los esquemas.
- aislar los esquemas: los cambios en un esquema no deben afectar a los esquemas de nivel superior y por tanto, tampoco a los programas de aplicación.

INDEPENDENCIA DE DATOS

3.2.- Independencia de datos.



3.2.- Independencia de datos.

- *Independencia lógica* entre el esquema lógico y los externos:
  - Los esquemas externos y los programas de aplicación no deben verse afectados por modificaciones del esquema lógico sobre datos que no usan.

    EJEMPLO: Si al edificio se le añade un campo “Estado_de_conservación”, el esquema externo del jefe no cambia y la aplicación del jefe no se tiene que modificar.

- *Independencia física* entre el esquema interno y el lógico:
  - el esquema lógico no debe verse afectado por cambios en el esquema interno referentes a la implementación de las estructuras de datos, modos de acceso, tamaños de páginas, caminos de acceso, etc.

    EJEMPLO: Si la relación edificio se cambia de localización física, el esquema lógico no se ve afectado.

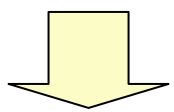
3.2.- Independencia de datos.

LIGADURA:

- Transformación del esquema externo en el esquema interno.

Tipos
- Ligadura lógica (pasos 2 y 7).
- Ligadura física (pasos 3 y 6).

- Cuando se produce la ligadura desaparece la independencia.



Es importante determinar ese momento

3.2.- Independencia de datos.

Programa de aplicación:

- Ligadura en tiempo de compilación:
  - ◊ Transformación del esquema externo que usa el programa en términos del esquema interno.
  - ◊ Cualquier cambio del esquema lógico y/o interno requiere una recompilación.
- Ligadura al ejecutar el programa:
  - ◊ No requiere ninguna acción sobre el programa.

3.2.- Independencia de datos.

Momento de la ligadura:

- en compilación o en la precompilación
- en el montaje
- al iniciarse la ejecución o en el momento de conectarse
- en cada acceso a la base de datos

Mayor independencia cuanto más tardía sea la ligadura

Menor coste cuanto más temprana sea la ligadura

3.3.- Integridad

- Objetivo de la tecnología de bases de datos
- Calidad de la información:

"*los datos deben estar estructurados reflejando adecuadamente los objetos, relaciones y las restricciones existentes en la parcela del mundo real que modela la base de datos*"

- Representación de los objetos, relaciones y restricciones en el esquema de la base de datos.
- Cambios en la realidad $\rightarrow$ Actualizaciones de los usuarios
- La información contenida en la base de datos debe preservar la definición del esquema.

3.3.- Integridad

- Calidad de la información (perspectiva de la integridad):
  - SGBD debe asegurar que los datos se almacenan correctamente
  - SGBD debe asegurar que las actualizaciones de los usuarios sobre la base de datos se ejecutan correctamente y que se hacen permanentes

3.3.- Integridad

Herramientas del SGBD orientadas a la integridad:

- Comprobar (frente a actualizaciones) las restricciones de integridad del esquema
- Controlar la ejecución correcta de las actualizaciones (entorno concurrente)
- Recuperar (reconstruir) la base de datos en caso de pérdidas o accidentes

3.3.- Integridad: accesos concurrentes

Cuentas

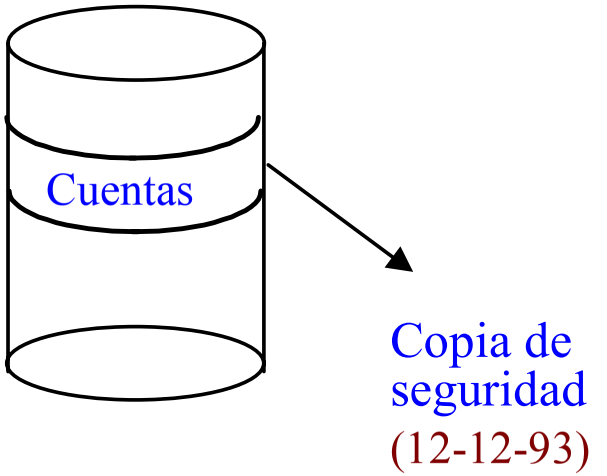
Nro.	Saldo
123	1000
555	2000

Cuentas

Nro.	Saldo
123	800
555	2000

Tiempo	P1	P2
t1	leer(123, saldo)	
t2		leer(123, saldo)
t3	saldo←saldo-100	
t4		saldo←saldo-200
t5	escribir(123, saldo)	
t6		escribir(123, saldo)

3.3.- Integridad: recuperación



13-12-93: Actualización de cuentas

transacción nro. 51: **¡fallo del sistema!**

Procedimiento de recuperación:

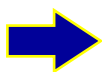
- sustituir el fichero de *Cuentas* por su copia de seguridad

Efecto negativo:

- se han perdido las actualizaciones de 50 transacciones

3.3.- Integridad: transacciones

- La integridad de la base de datos se ve en peligro generalmente por las operaciones de acceso de las aplicaciones.
- Las operaciones de acceso a una base de datos se organizan en transacciones.

TRANSACCIÓN  Secuencia de operaciones de acceso a la base de datos que constituyen una unidad lógica de ejecución

3.3.- Integridad: transacciones

Emp(dni, nombre, dir, dept)
  CP: {dni}
  CAj: {dept} $\rightarrow$ Dep

Dep(cod, nombre, ubicación)
  CP: {cod}

R1: $\forall$Dx (Dep(Dx) $\rightarrow$ $\exists$Ex (Emp(Ex) $\wedge$ Dx.cod = Ex.dept))

Inserción de un nuevo departamento:

<d2, “Personal”, “Planta 3ª”>

cuyo primer empleado es el de *dni* 20

3.3.- Integridad: transacciones

1ª Idea

1) Inserción en *Dep*: <d2, “Personal”, “Planta 3ª>

ERROR: la restricción *R1* no se cumple

2) Modificación de *Emp* en la tupla con *dni* 20

2ª Idea

1) Modificación de *Emp* en la tupla con *dni* 20

ERROR: la clave ajena sobre *dept* en *Emp* no se cumple

2) Inserción en *Dep*: <d2, “Personal”, “Planta 3ª>

3.3.- Integridad: transacciones

Operaciones de las transacciones relevantes para el SGBD:

- **leer(X)**: lectura o consulta del dato X de la base de datos sobre la variable del programa del mismo nombre
- **escribir(X)**: actualización (inserción, borrado o modificación) del dato X de la base de datos usando la variable del mismo nombre del programa

3.3.- Integridad: transacciones

Operaciones de las transacciones relevantes para el SGBD:

– leer(X):

1. buscar la dirección del bloque que contiene el dato X

2. copiar el bloque a un *buffer* de memoria principal

3. copiar el dato X del *buffer* a la variable X del programa

3.3.- Integridad: transacciones

Operaciones de las transacciones relevantes para el SGBD:

– escribir(X):

Si no se ha leído antes:

1. buscar la dirección del bloque que contiene el dato X
2. copiar el bloque a un *buffer* de memoria principal

3. copiar el dato X de la variable del programa a la posición adecuada en el *buffer*

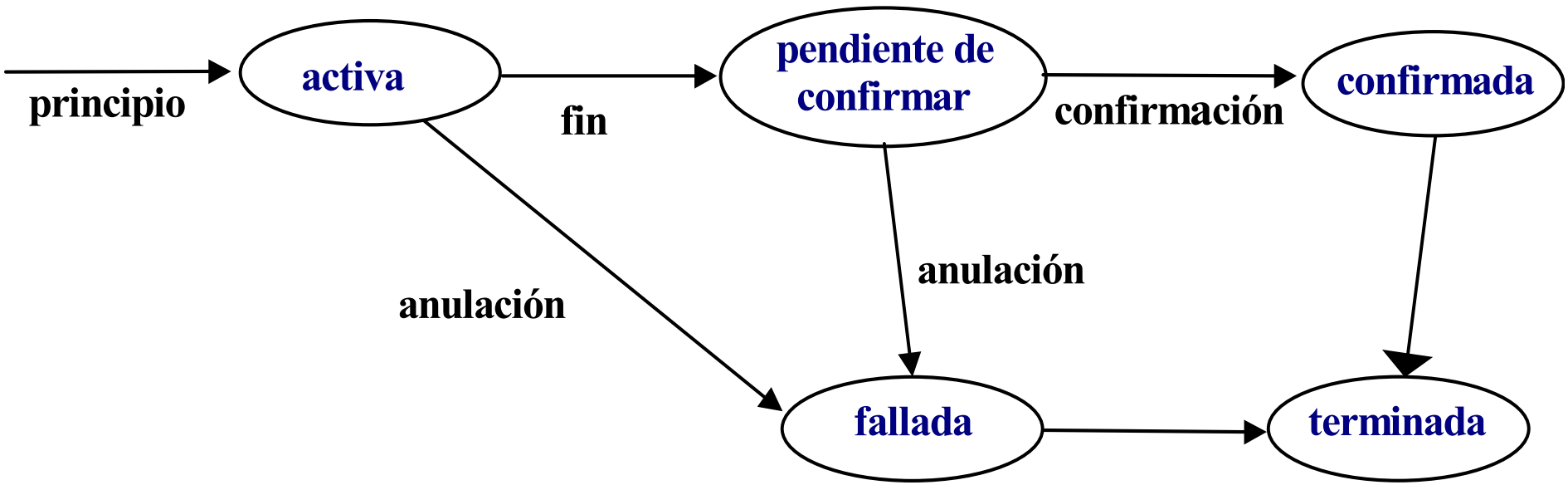
4. copiar el bloque actualizado del *buffer* al disco

3.3.- Integridad: transacciones

Operaciones de definición de las transacciones:

- **principio**: indica el comienzo de la transacción
- **fin**: indica que se han terminado todas las operaciones de la transacción.
- **confirmación**: indica el éxito de la transacción, permitiendo que el SGBD guarde los cambios efectuados en la base de datos
- **anulación**: indica el fracaso de la transacción debido a algún motivo. El SGBD deshace todos los posibles cambios efectuados por la transacción

3.3.- Integridad: transacciones



3.3.- Integridad: transacciones

Propiedades que deben cumplir las transacciones:

- **atomicidad**: una transacción es una unidad atómica de ejecución (o se ejecutan todas sus operaciones o ninguna)
- **consistencia**: la transacción debe dar lugar a un estado de la base de datos consistente (se cumplen todas las restricciones de integridad)
- **aislamiento**: las modificaciones introducidas por una transacción no confirmada no son visibles al resto de transacciones
- **persistencia**: la confirmación implica la grabación de los cambios introducidos en la base de datos, de forma que no se puedan perder por fallo del sistema o de otras transacciones

3.3.- Integridad: transacciones

Dos tipos de funcionamiento de las transacciones (según SGBD):

- **Actualización Inmediata**: las actualizaciones tienen efecto inmediato en memoria secundaria y en caso de anulación se tienen que deshacer.
- **Actualización Diferida**: las actualizaciones sólo tiene efecto inmediato en memoria principal y se transfieren a memoria secundaria cuando se confirman.

3.3.- Integridad: integridad semántica

- Restricción de integridad:

  Propiedad del mundo real que modela la base de datos

- Las restricciones se definen en el esquema lógico y el SGBD debe velar por su cumplimiento.

- La comprobación se realiza cuando la base de datos cambia (se ejecuta una operación de actualización)

- Las restricciones que no se incluyen en el esquema de la base de datos se han de mantener en los programas de aplicación

3.3.- Integridad: integridad semántica

- Tipos de restricciones de integridad:

  – estáticas: se deben cumplir en cada estado de la base de datos (representable en CRT)

  EJEMPLOS: Def. de Dominios, CP, CAj, VNN, UNIQUE, Assertions, ...

  – de transición: se deben cumplir en dos estados consecutivos

  EJEMPLO: El precio de un inmueble no puede disminuir

3.3.- Integridad: integridad semántica

- Restricciones en el SQL/92:
  - estáticas:
    - ◊ sobre dominios: de valor
    - ◊ sobre atributos: valor no nulo, de rango, etc.
    - ◊ sobre relaciones: clave primaria, unicidad y claves ajenas.
    - ◊ sobre la base de datos: condiciones de búsqueda generales* (Assertions)
    - cuando se comprueba:
      - después de cada operación (NOT DEFERRABLE)
      - al final de la transacción (DEFERRABLE)
    - acciones compensatorias:
  - de transición: se deben cumplir en dos estados consecutivos*

* (no suelen mantenerlas los sistemas comerciales)

3.3.- Integridad: integridad semántica

- Procedimientos de comprobación de la integridad (reglas de actividad, triggers, ...) :
  - programación de la comprobación por parte del diseñador
  - permiten incluir en el esquema de la base de datos las restricciones complejas
  - en los procedimientos se debe incluir:
    - ◊ operaciones que los activan (evento y condición)
    - ◊ código a ejecutar que incluye operaciones sobre la base de datos
    - ◊ acciones de rechazo o compensación en caso de violación

3.3.- Integridad: control de accesos concurrentes

- El SGBD debe controlar los accesos concurrentes de las aplicaciones.

- Problemas por interferencia de accesos concurrentes:

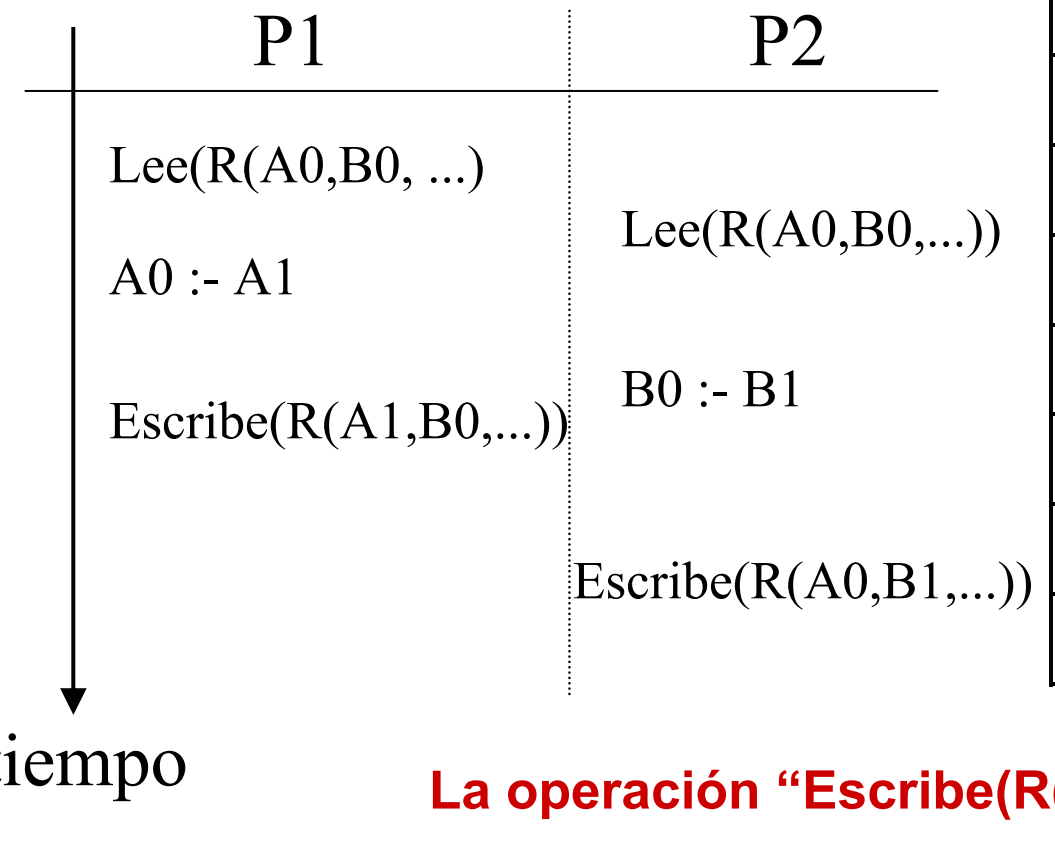
  a) pérdida de actualizaciones,

  b) obtención de información incoherente correspondiente a varios estados válidos de la base de datos, y

  c) lectura de datos actualizados (no confirmados) que han sido sometidos a cambios que todavía pueden ser anulados.

3.3.- Integridad: control de accesos concurrentes

a) Pérdida de las actualizaciones



R

...				
A0	B0	...	...	
...				

La operación "Escribe(R(A1,B0)" se pierde !!!!

3.3.- Integridad: control de accesos concurrentes

b) Obtención de información incoherente

P1: Obtención del total de saldos.

P2: Transferencia de la cuenta 100 a la 1.

P1	P2
Lee(C1, Saldo)	
Lee(C2)	...
	Lee(C100)
...	Escribe(C100, Saldo = Saldo – X)
	Lee(C1)
	Escribe(C1, Saldo = Saldo + X)
...	
Lee(C99, Saldo)	...
Lee(C100, Saldo)	

tiempo

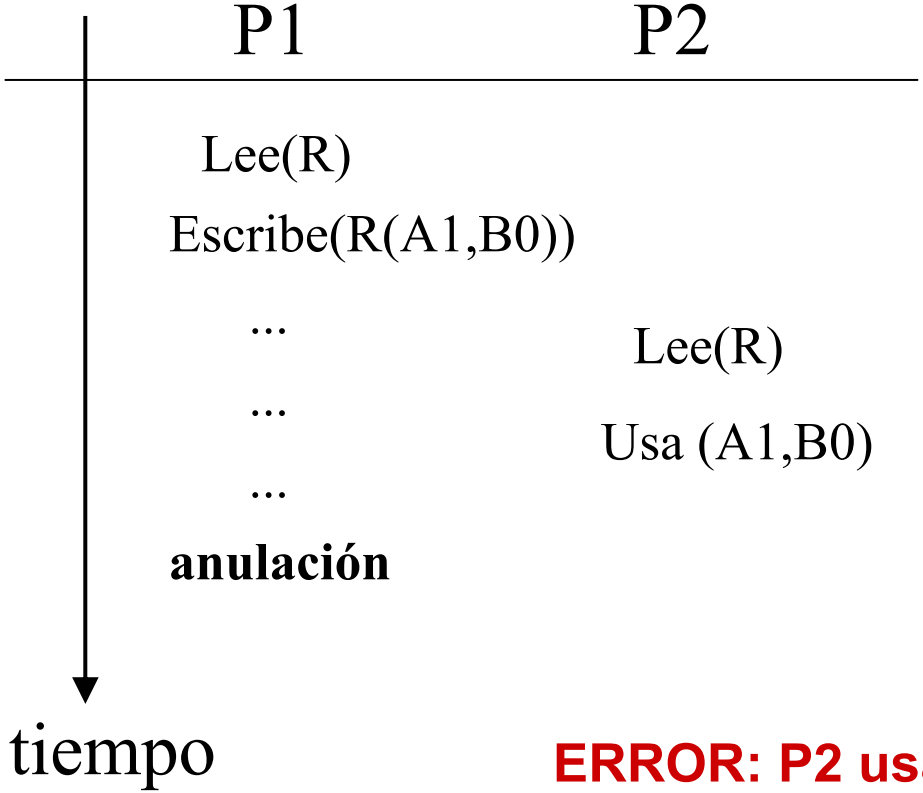
Cuentas Corrientes

C1	€200000	
C2	...	..
C100	€200000	..

ERROR!! A la suma de P1 faltan X Euros.

3.3.- Integridad: control de accesos concurrentes

c) Lectura de datos actualizados sin confirmar



R

...				
A0	B0	...	...	
...				

ERROR: P2 usa un dato inválido.

3.3.- Integridad: control de accesos concurrentes

Técnicas:

- Reserva de Ocurrencias de Datos (Locks)
  - Ejemplos a) y c) se reserva un registro.
  - Ejemplo b) se reservan todos.
  - Necesidad de controlar bloqueos (deadlocks)

- Otras soluciones (para el ejemplo c): anulación en cascada o aislamiento de transacciones.

3.3.- Integridad: reconstrucción de la base de datos

Las propiedades de atomicidad y persistencia de una transacción obligan al SGBD a asegurar que:

- si se confirma, los cambios efectuados se graban en la base de datos y no se pierdan.
- Si se anula, los cambios efectuados sobre la base de datos se deshacen.

3.3.- Integridad: reconstrucción de la base de datos

Causas del fallo de una transacción

- Locales a la transacción (funcionamiento del sistema normal)
  - errores en la transacción (acceso a la base de datos incorrecto, cálculos fallidos, etc.)
  - excepciones (violación de la integridad, de la seguridad, etc.)
  - control de la concurrencia (estado de bloqueo entre dos transacciones)
  - decisiones humanas (por programa o explícitas).

3.3.- Integridad: reconstrucción de la base de datos

Causas del fallo de una transacción

- Externas a la transacción (errores del sistema)
  - fallos del sistema con pérdida de la memoria principal.
  - fallos del sistema de almacenamiento con pérdida de la memoria secundaria.

3.3.- Integridad: reconstrucción de la base de datos

Pérdidas de memoria principal

- En el espacio de tiempo entre la confirmación de una transacción y la grabación de sus campos en memoria secundaria.
- La transacción está confirmada y sus cambios están en los bloques de los *buffers*.
- En dicho intervalo se produce un fallo con pérdida de memoria principal y los bloques de los *buffers* se pierden.

3.3.- Integridad: reconstrucción de la base de datos

Pérdidas de memoria secundaria

- Transacción confirmada cuyos cambios están grabados en la base de datos.
- Fallo en la memoria secundaria y estos cambios se pierden.

3.3.- Integridad: reconstrucción de la base de datos

Reconstrucción frente a fallos del sistema

Funciones
- Recuperar transacciones confirmadas que no han sido grabadas.
- Anular transacciones que han fallado.

- Módulo de reconstrucción.
  - Técnica más extendida: uso del ***fichero diario*** *(log o journal)*.

3.3.- Integridad: reconstrucción de la base de datos

Actividades sobre el fichero diario

- Registrar las operaciones de actualización de las transacciones.
- Se almacena en disco para evitar la desaparición por un fallo del sistema.
- Se graba periódicamente a una unidad de almacenamiento masiva.

3.3.- Integridad: reconstrucción de la base de datos

Tipo de entradas que se graban en el fichero diario

- [inicio, T]: se ha iniciado la transacción de identificador *T*.
- [escribir, T, X, valor_antes, valor_después]: la transacción *T* ha realizado una operación de actualización sobre el dato *X*.
- [leer, T, X]: la transacción *T* ha leído el dato *X*.
- [confirmar, T]: la transacción *T* ha sido confirmada.
- [anular, T]: la transacción *T* ha sido anulada.

3.3.- Integridad: reconstrucción de la base de datos

Supondremos ACTUALIZACIÓN INMEDIATA

Fallo de una transacción $T \rightarrow$ Deshacer cambios de T

- actualizar los datos modificados por T con su valor original (*valor_antes*).
- Buscar las entradas en el diario

  [escribir, T, X, valor_antes, valor_después]

Fallo del sistema $\rightarrow$ Aplicar el proceso anterior a todas las transacciones sin confirmar

3.3.- Integridad: reconstrucción de la base de datos

Fallo del sistema → • Transacciones sin confirmar

[inicio, T] en el diario sin [confirmar, T]

• Proceso anterior

→ • Transacciones confirmadas

[confirmar, T]

• Volver a ejecutarlas:

[escribir, T, X, valor_antes, valor_después]

3.3.- Integridad: reconstrucción de la base de datos

PROBLEMAS:

- Tamaño del fichero diario puede crecer muy rápidamente.
- Recuperación en caso de fallo muy costosa (hay que rehacer muchas operaciones).

SOLUCIÓN:

Puntos de verificación (checkpoints)

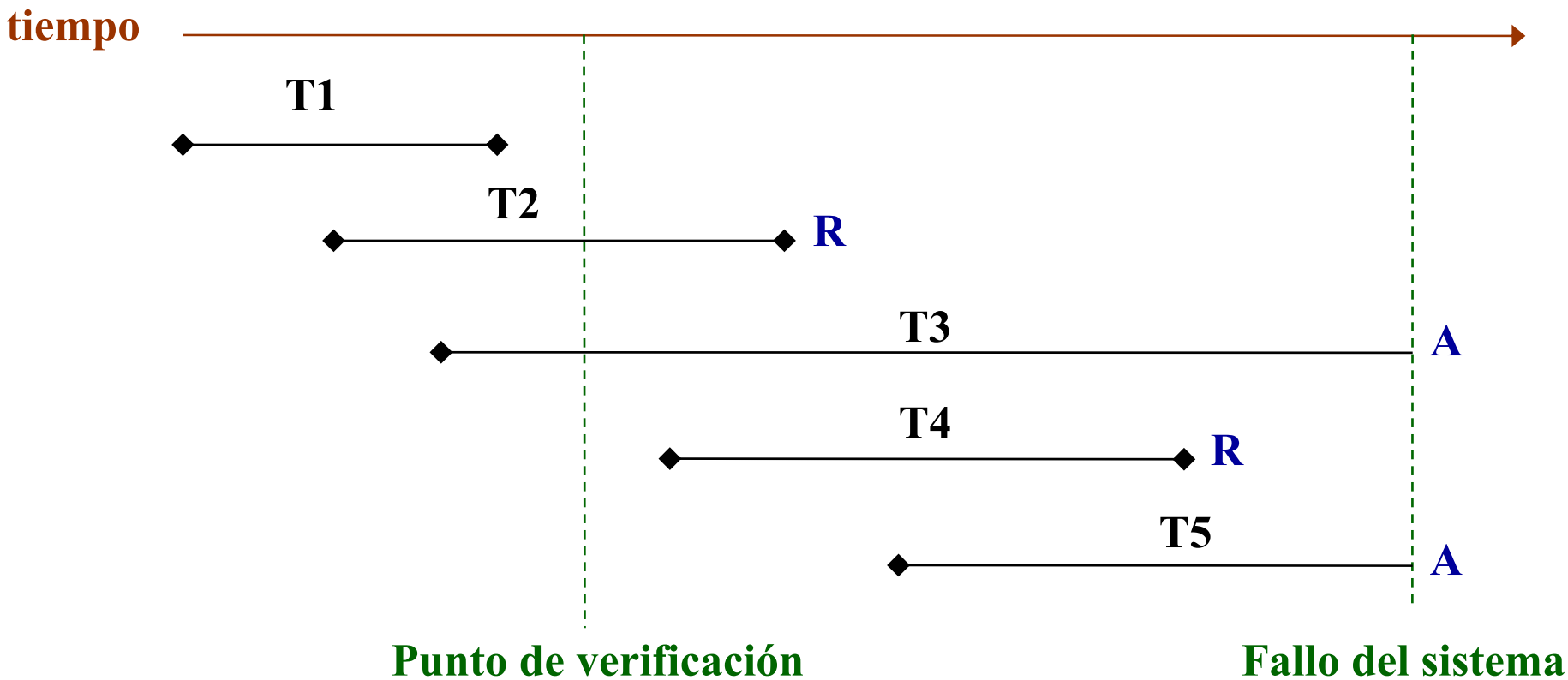
3.3.- Integridad: reconstrucción de la base de datos

Puntos de verificación → Se graban en el diario periódicamente

- Suspender temporalmente la ejecución de transacciones.
- Grabar en el diario el punto de verificación.
- Forzar la grabación de todas las actualizaciones de las transacciones confirmadas (copiar los *buffers* a disco).
- Reanudar la ejecución de las transacciones suspendidas.

3.3.- Integridad: reconstrucción de la base de datos

Puntos de verificación → Reconstrucción a partir del último



3.3.- Integridad: reconstrucción de la base de datos

Reconstrucción frente a fallos del sistema de almacenamiento

- Pérdida de memoria secundaria.
- Base de datos puede estar dañada total o parcialmente.
- Técnica: reconstruir la base de datos a partir de
  - La copia de seguridad más reciente.
  - A partir del instante de la copia utilizar el diario para rehacer las operaciones realizadas por las transacciones confirmadas.

3.3.- Integridad: reconstrucción de la base de datos

Caso ACTUALIZACIÓN DIFERIDA

El mecanismo de reconstrucción es el mismo (las confirmadas se deben repetir), exceptuando:

- Las no confirmadas no deben ser deshechas.

3.4.- Seguridad

Objetivo:

Sólo pueden acceder a la información las personas y procesos autorizados y en la forma autorizada.

3.4.- Seguridad

Técnicas:

- Identificación del usuario.
- Determinación de los accesos permitidos:

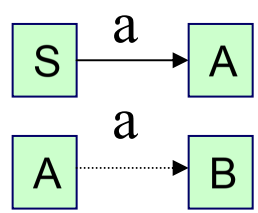
  Modos:
  - Lista de autorizaciones (objeto y operaciones permitidas) por usuario.
  - Niveles de autorización (menos flexible).

- Gestión de autorizaciones transferibles: traspaso de autorizaciones de un usuario a otro.

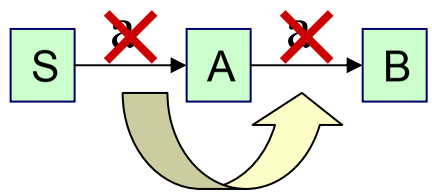
3.4.- Seguridad

Requerimientos para realizar la gestión de autorizaciones transferibles:

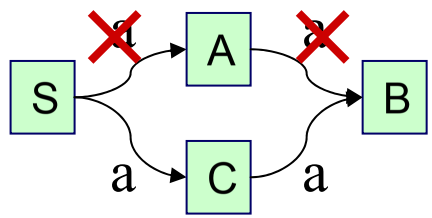
- Conocimiento de las autorizaciones de acceso de cada usuario (cuáles son transferibles a terceros y cuáles no).
- Transferencia de una autorización de un usuario a otro (en modo transferible o no).



- Revocación posterior de una autorización de acceso:
  - Si se otorgó en modo transferible, revocación de las autorizaciones que partieron de ella.



- Revocación independiente de una autorización de acceso otorgada de forma múltiple.



B debe conservar el permiso ‘a’

3.5.- Implementación de BDA relacionales.

ESQUEMA FÍSICO:

Descripción de la BD en términos de su representación física (sobre dispositivo de almacenamiento secundario).

3.5.1- Conceptos previos.

- **Fichero**: secuencia de registros para el almacenamiento en memoria secundaria. Los registros pueden ser del mismo tipo o de distinto tipo.
- **Registro**: colección de valores relacionados (en el caso de una base de datos, representa una tupla).
- **Bloqueo**: cuando el tamaño B del bloque es mayor que el tamaño R del registro, cada bloque contiene un número de registros llamado **factor de bloqueo: fb = B div R.**
- **Organización extendida:** un registro puede estar repartido en varios bloques; es frecuente que no sea extendida, porque simplifica el procesamiento de los registros.

3.5.1.- Conceptos previos

Organización de un fichero:

- Se refiere a cómo se sitúan los registros del fichero en los bloques y a las estructuras de acceso.

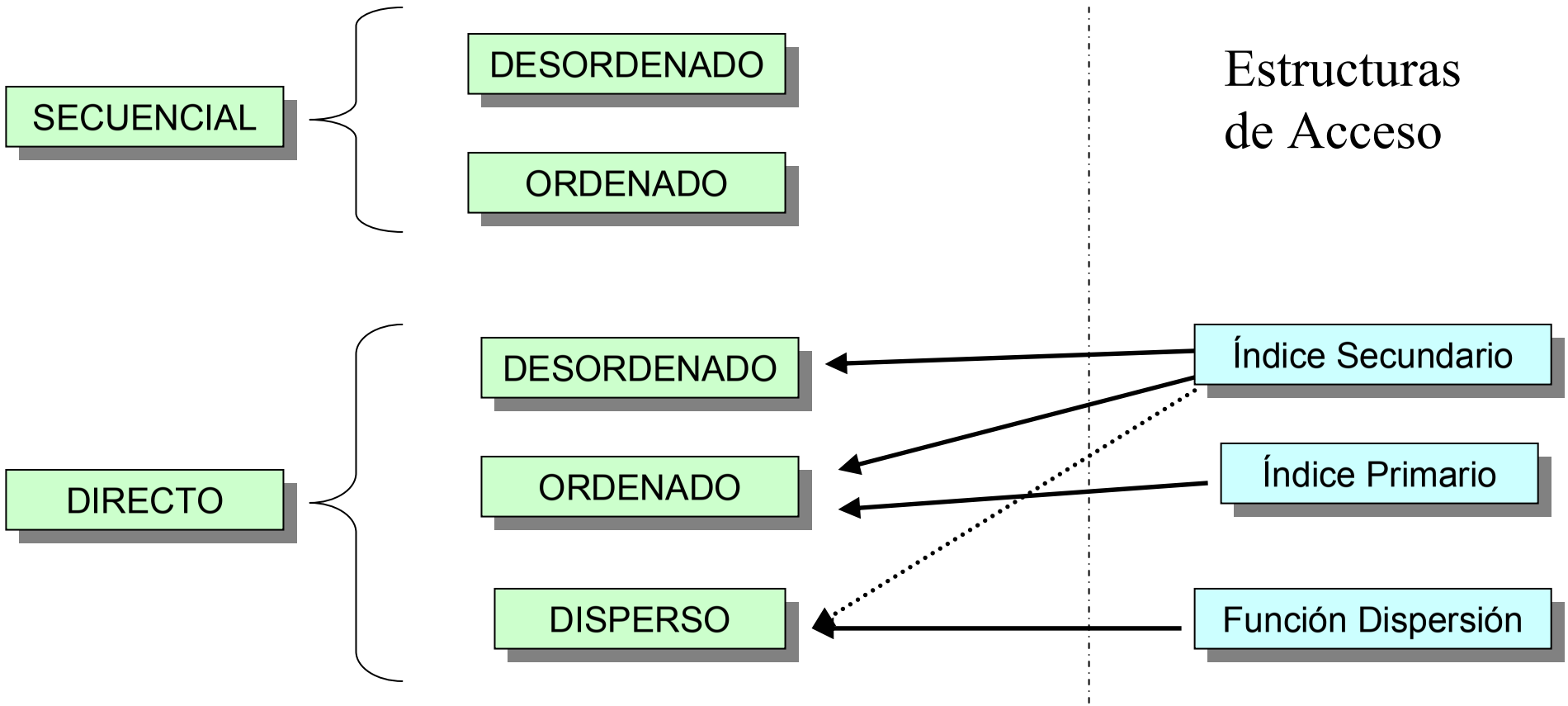
Método de acceso:

- Grupo de programas que implementan las operaciones sobre ficheros. Una organización de fichero permite diferentes métodos de acceso.

Una organización, junto con un método de acceso apropiado, debe permitir realizar de la forma más eficiente posible las operaciones que se realizan con más frecuencia.

3.5.1.- Conceptos previos

CLASIFICACIÓN DE ORGANIZACIÓN DE FICHEROS



3.5.2.- Ficheros Directos.

- Incluye aquellos modelos de organización de ficheros que permiten acceder a la localización exacta del registro buscado.

- El acceso directo se puede conseguir por distintos métodos:
  - direccionamiento relativo (ficheros ordenados)
  - dispersión
  - uso de índices.

3.5.2.1.- Ficheros ordenados.

Fichero Ordenado con Direccionamiento Relativo:

- es un fichero en el que los registros se almacenan ordenados de acuerdo al valor de uno de sus campos y que permite direccionamiento relativo.

- Ventajas:
  - lectura en orden del campo de ordenación muy eficiente.
  - encontrar el siguiente en orden del campo de ordenación no requiere accesos adicionales (excepto en el último registro de un bloque).
  - la búsqueda basada en el campo de ordenación puede ser **binaria** sobre los bloques.

3.5.2.1.- Ficheros ordenados.

- Inconvenientes:
  - El acceso basado en un campo distinto al de ordenación obliga a buscar en todo el fichero hasta que se encuentre.
  - La inserción es muy costosa, porque hay que localizar el lugar en que debe situarse el registro, y desplazar los registros posteriores para hacer sitio. Sol $\rightarrow$ huecos o desborde.
  - El borrado no es tan problemático si se marca el registro pero sin recuperar su espacio.
  - La modificación del valor del campo de ordenación, puede cambiar su posición en el fichero, lo que supone un borrado y una inserción.

3.5.2.1.- Ficheros ordenados.

- El coste de la modificación de un campo distinto al de ordenación depende sólo de la condición de búsqueda del registro a modificar.
- Para disminuir el gran coste en el caso de inserción o modificación del valor del campo de ordenación, existen dos soluciones:
  - uso de **huecos** en los bloques para sólo tener que reorganizar como mucho el tamaño del bloque. Problemas: espacio perdido y reorganizaciones periódicas cuando se llenan.
  - uso de un fichero temporal, llamado de **desborde** o de **transacción**, para ir añadiendo los registros nuevos y que se mezcla periódicamente con el **principal** en el proceso de **reorganización.**
- Los ficheros directos con direccionamiento relativo (desordenados u ordenados) sólo se suelen usar en BD con índices.

3.5.2.2.- Ficheros dispersos.

Fichero Disperso:

Caracterización:

- Esta técnica proporciona un acceso muy rápido cuando la condición de búsqueda es el valor del **campo de dispersión**, que generalmente es una clave.
- Hay una **función** llamada **de dispersión** o aleatorización que se aplica al valor del campo de dispersión de un registro y devuelve la dirección del bloque del disco donde se va a guardar el registro.
- Para la recuperación de la mayoría de registros se necesita un único acceso al disco.

3.5.2.2.- Ficheros dispersos.

Modo de funcionamiento:

- El espacio de direcciones asignado al fichero se compone de **cubos**, en cada uno de los cuales caben muchos registros.
- Supongamos que tenemos M cubos, cuyas direcciones relativas oscilan entre 0 y M–1. Tenemos que escoger una función que transforme el valor del campo de dispersión en un entero entre 0 y M–1.
- Una función de dispersión común es $d(K) = K \bmod M$, que devuelve el resto de dividir el valor K del campo de dispersión por M.

3.5.2.2.- Ficheros dispersos.

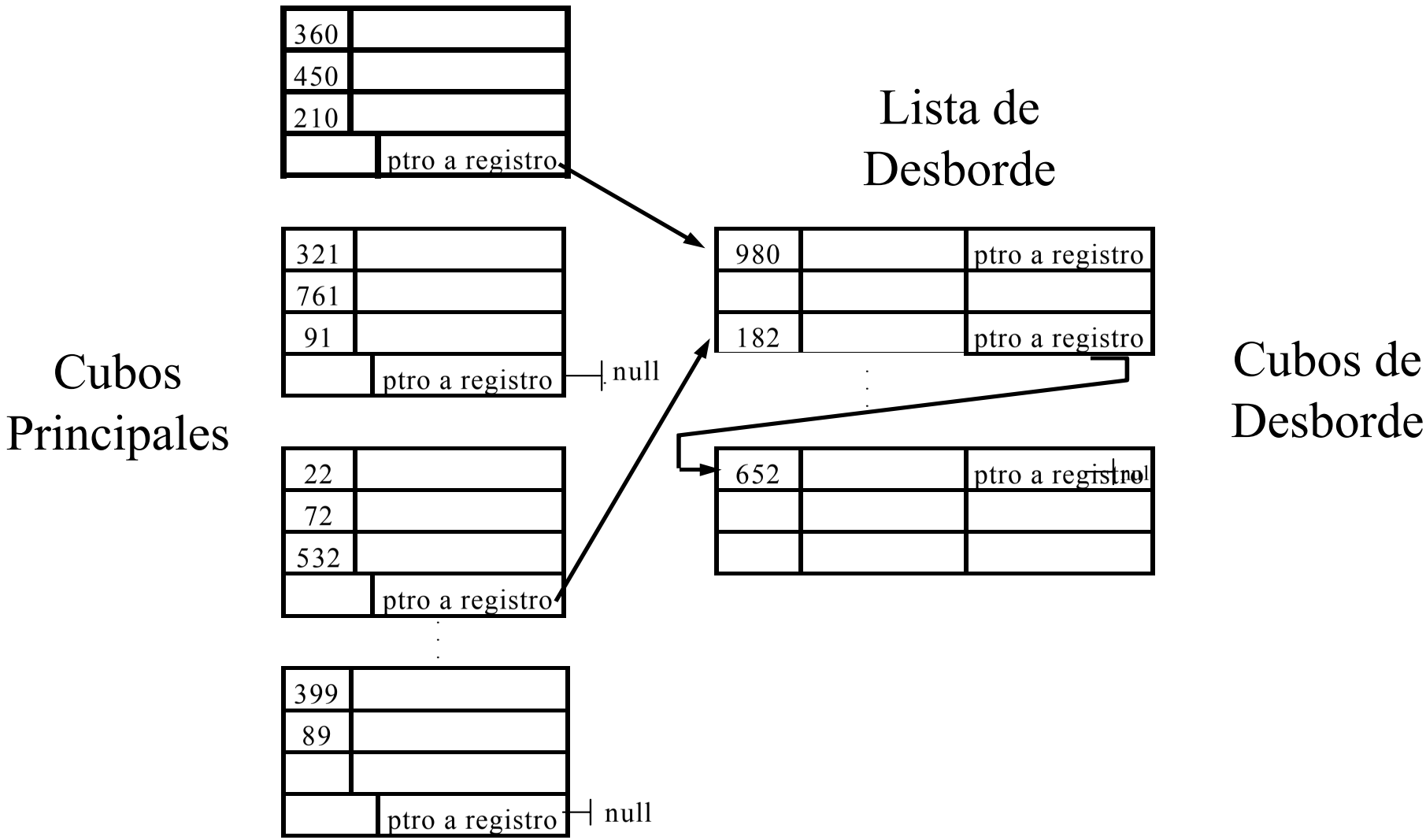
PROBLEMA:

- La mayoría de las funciones de dispersión no pueden garantizar que para distintos valores se obtendrán direcciones diferentes, porque el número de valores distintos que puede tomar el campo de dispersión es mucho mayor que el número de direcciones disponible para los registros.

SOLUCIÓN:

- Aplicar técnicas de resolución de colisiones:
  - direccionamiento abierto: a bloques siguientes
  - encadenamiento: lista de desborde
  - dispersión múltiple: se aplica una 2ª f. de dispersión.

3.5.2.2.- Ficheros dispersos.



3.5.2.2.- Ficheros dispersos.

- Ventajas:
  - Proporciona un acceso muy rápido para localizar un registro arbitrario dado su valor del campo de dispersión.

- Inconvenientes:
  - no es muy útil cuando se requieren otras aplicaciones para el mismo fichero, a no ser que se construyan caminos de acceso adicionales.
  - El espacio reservado para los ficheros es fijo (se desaprovecha mucho espacio al principio y suele estar desbordado con el tiempo).

3.5.2.3.- Ficheros indizados.

ÍNDICES:

- Un índice es una estructura de acceso definida sobre uno o varios **campos de indización**.
- Consiste en un fichero adicional cuyos registros (o entradas) están constituidos por dos campos: clave (campo de indización) y dirección.
- No afectan al fichero original.
- Un fichero puede tener varios índices para varios campos.

3.5.2.3.- Ficheros indizados.

ÍNDICES:

- Hay varios tipos de índices:

Según el Direcc. y la Estructura
- Físicos:
  - Estáticos:
    - Ordenado de un nivel.
    - Multinivel.
  - Multinivel dinámico (árbol B; árbol B^+).
- Lógicos‡

Según el Campo que Indizan
- Primarios*
- Secundarios

‡ Requiere que el fichero que indizan esté ordenado o sea disperso.

* Requieree que el fichero que indizan esté ordenado

3.5.2.3.- Ficheros indizados.

Índice ordenado de un nivel.

- Las entradas del índice están ordenadas, se puede usar la búsqueda binaria.
- El tamaño del fichero índice es mucho menor que el del fichero de datos: búsqueda más eficiente.
- Un índice se dice que es *claro* cuando posee una entrada por cada uno de los *bloques* del fichero de datos.
- Un índice se dice que es *denso* cuando posee una entrada por cada uno de los *registros* del fichero de datos.

3.5.2.3.- Ficheros indizados.

Índice ordenado de un nivel (cont.).

- **Índice primario**:
  - Hay una entrada por cada bloque del fichero principal.
  - Cada entrada posee dos campos: el del campo de ordenación del fichero principal, y un puntero a bloque.
  - Los índices primarios son *claros*.

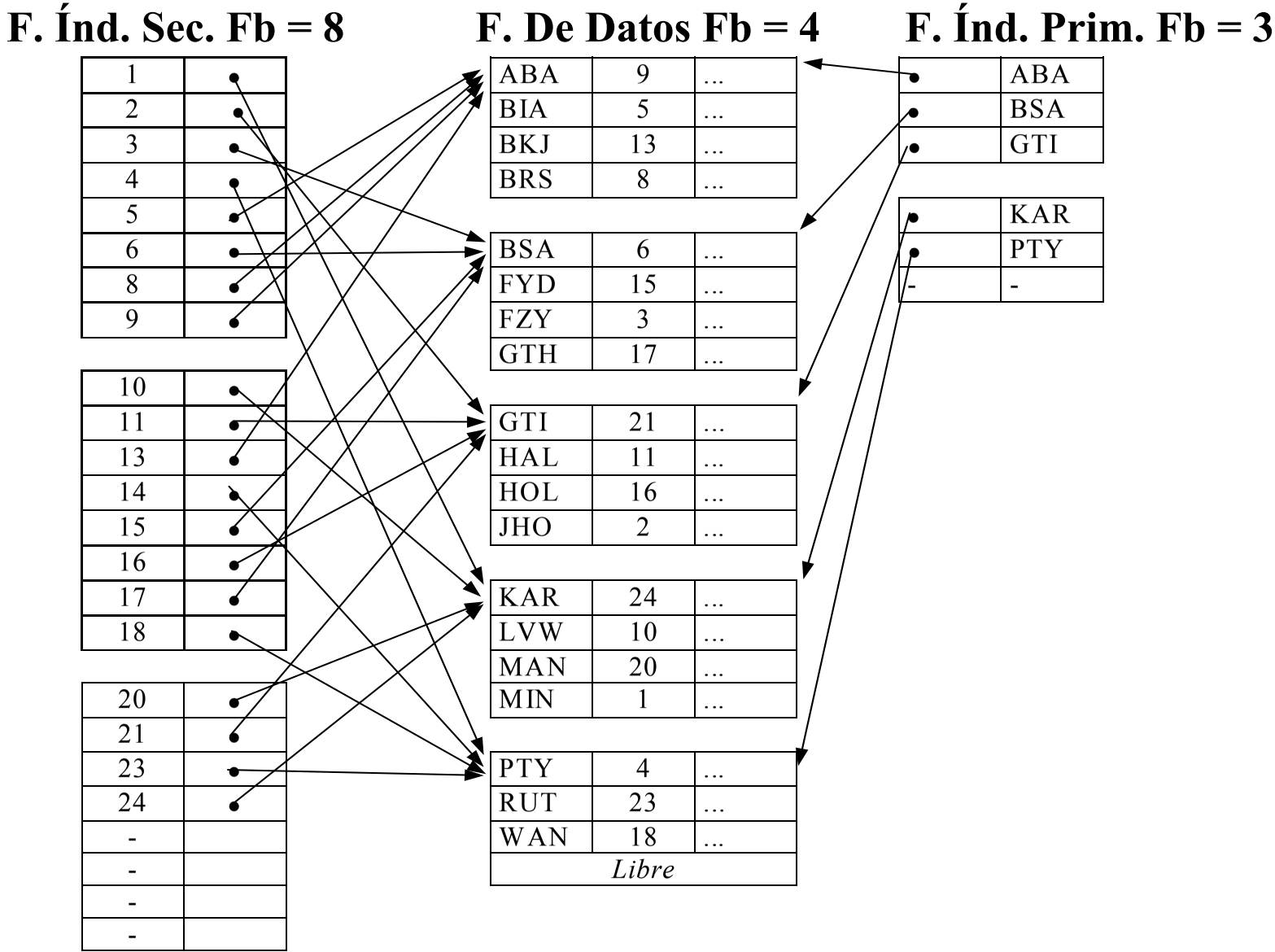
3.5.2.3.- Ficheros indizados.

Índice ordenado de un nivel (cont.).

- **Índice secundario**:
  - Hay una entrada por cada registro del fichero principal (que puede ser al principio del bloque o al registro en cuestión).
  - El fichero principal no necesita estar ordenado por el campo indizado (aunque sí el fichero de índices).
  - Si el campo de indización no es clave (no unicidad): entradas con valores repetidos, listas de punteros o bloques de punteros.
  - Los índices secundarios son densos.

3.5.2.3.- Ficheros indizados.

Ejemplo de Índices Primario y Secundario sobre dos campos clave en un mismo fichero



3.5.2.3.- Ficheros indizados.

Índice multinivel.

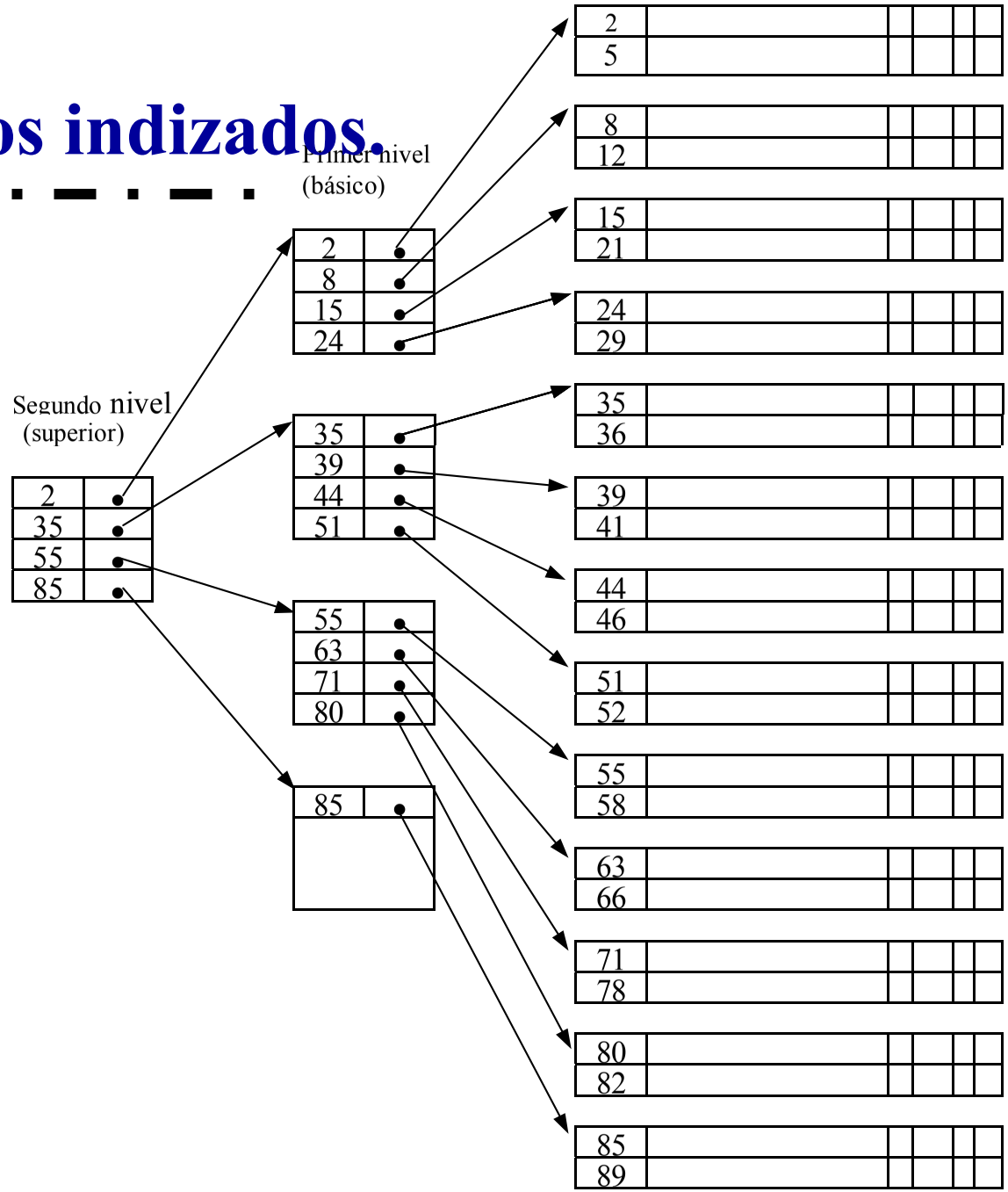
- Se intenta que en cada búsqueda se requieran aproximadamente $\mathbf{log_{fbi} b_i}$ accesos.

CONSTRUCCIÓN:

- El índice multinivel considera al fichero índice (**primer nivel** o nivel básico), como un fichero ordenado con un valor distinto para el campo de indización en cada entrada.
- Si el primer nivel ocupa más de un bloque, se crea un índice primario sobre el primer nivel. Este índice sobre el primer nivel se llama **segundo nivel** del índice multinivel.
- Se repite el proceso sucesivamente, hasta que un nivel quepa en un solo bloque que se llama el **nivel máximo**.

3.5.2.3.- Ficheros indizados.

Ejemplo de Índices Primario de dos Niveles



3.5.2.3.- Ficheros indizados.

Índice multinivel dinámico.

- Son árboles de búsqueda (B y B+) con algunas restricciones:
  - El árbol estará siempre equilibrado
  - Aseguran que el espacio inutilizado por los borrados nunca será excesivo.

- Los algoritmos de inserción y borrado son complejos, aunque la mayoría de inserciones y borrados son procesos simples.

3.5.3.- Elección de esquema físico.

- Cada SGBD ofrece una variedad de opciones para la organización de los ficheros.

- El diseñador de la base de datos debe tener en cuenta:
  - factores de tiempo de respuesta necesario,
  - utilización de espacio por los ficheros y sus estructuras de acceso,
  - frecuencia de ejecución de determinadas consultas y transacciones,
  - otros requisitos especificados para la base de datos.

3.5.3.- Elección de esquema físico.

- Los atributos que se espera que sean utilizados frecuentemente para recuperar registros deben tener definidos sobre ellos caminos de acceso primario o índices secundarios.
- A veces se hace necesario reorganizar algunos ficheros construyendo índices nuevos o cambiando los métodos de acceso primarios.
- Una opción muy popular para organizar un fichero en un sistema relacional es mantener los registros del fichero desordenados y crear tantos índices secundarios como se necesiten.

3.5.3.- Elección de esquema físico.

- Si los registros se van a recuperar frecuentemente en orden de un atributo: **ordenación** sobre ese atributo, con el índice primario correspondiente, si no va a variar mucho.
- Si el fichero va a sufrir muchas inserciones y borrados, hay que intentar **minimizar el número de índices**.
- En muchos sistemas el **índice** no es una parte integral del fichero, sino que se crea o destruye **dinámicamente**.
- Si un fichero **no** se va a usar a menudo para **recuperar registros en orden**, se puede usar **dispersión** (que debe ser dinámica si va a variar con frecuencia el tamaño del fichero).

3.5.4.- Las agrupaciones.

- Cuando dos relaciones tienen sendos atributos mediante los cuales es habitual realizar concatenaciones, se suelen utilizar **agrupaciones** o "clusters" de relaciones.
- Una agrupación consiste en guardar físicamente en el mismo bloque las tuplas de dos relaciones habitualmente concatenadas.

EJEMPLO:

- Sean **R1** y **R2** dos relaciones que se concatenan habitualmente y con esquemas de relación:

R1(a1:dom1, a2:dom2)

CP:{a1}

R2(b1:dom3, b2:dom4, b3:dom1)

CP:{b1}

Caj:{b3} $\rightarrow$ R1

3.5.4.- Las agrupaciones.

EJEMPLO (cont.):

- Extensión de R1 y R2:

R1

a1	a2
12	Doce
51	Cincuenta y uno
84	Ochenta y cuatro

R2

b1	b2	b3
9A	ASDF	84
0B	QWER	51
1L	ZXCV	12
2X	QAZ	12
3P	POIU	84
4K	MNBV	51
5T	TTTT	51
6M	MMM	12

3.5.4.- Las agrupaciones.

EJEMPLO (cont.):

- Agrupación de R1 y R2:

BLOQUE 1

a1	**a2**	
12	Doce	
	b1	**b2**
	1L	ZXCB
	2X	QAZ
	6M	MMM

BLOQUE 2

a1	**a2**	
51	Cincuenta y uno	
	b1	**b2**
	0B	QWER
	4K	MNBV
	5T	TTTT

BLOQUE 3

a1	**a2**	
84	Ochenta y cuatro	
	b1	**b2**
	9A	ASDF
	3P	POIU

3.5.4.- Las agrupaciones.

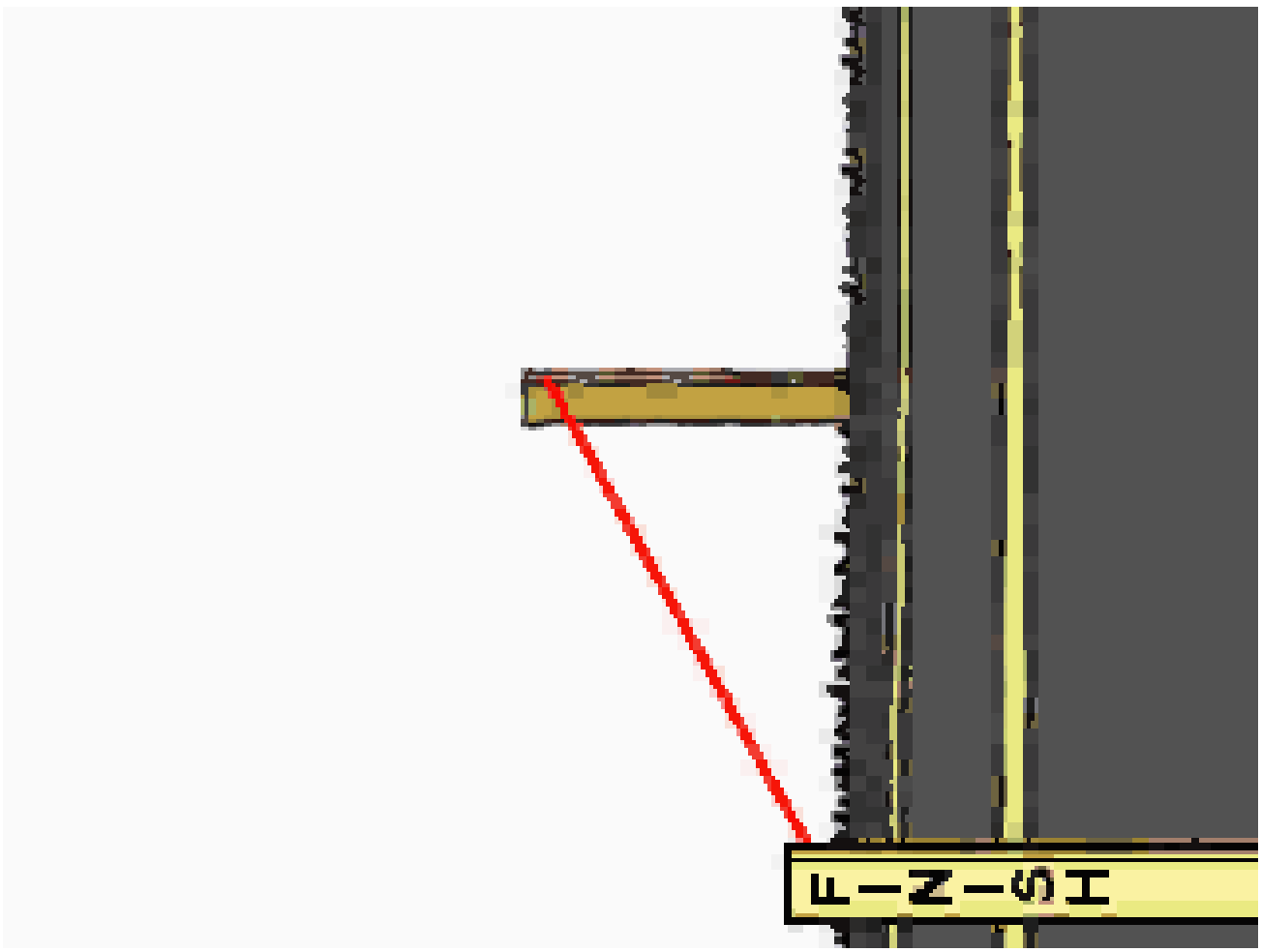
VENTAJAS:

- Las agrupaciones permiten:
    - Reducir el tiempo de acceso en concatenaciones.
    - Ahorro de espacio de almacenamiento: la clave de la agrupación sólo se almacena una vez.

INCONVENIENTES:

- Las agrupaciones reducen el rendimiento en inserciones o modificaciones.

THAT'S ALL FOLKS!


F-Z-5I