

Tecnología de Gestión de Datos

Tema 2: Tecnología Relacional


MU Ingeniería y Tecnología de Sistemas Software
Profesor: Juan Carlos Casamayor

Tecnología Relacional

Objetivos:

- ✓ revisar las características de la tecnología de bases de datos relacionales.
- ✓ revisar las técnicas para el control de la independencia, y la integridad de los datos en los sistemas relacionales.
- ✓ revisar la evolución de la tecnología relacional
- ✓ analizar una implementación de la tecnología relacional: SGBD Oracle

Tecnología Relacional

1. Modelo relacional de datos. 
2. Independencia de datos: arquitectura ANSI/SPARC.
3. Integridad de los datos.
4. Evolución de la tecnología relacional.
5. Implementaciones

1. Modelo relacional de datos

SGBD



herramienta (software) para la gestión (creación y manipulación) de bases de datos.

SGBD



modelo de datos



- ✓ tipos de estructuras de datos
- ✓ lenguajes de manipulación

1. Modelo relacional de datos

Familias de SGBD:



SGBD	modelo	estructuras
jerárquicos	jerárquico	registro (segmento), <u>árbol</u>
en red	red	registro, <u>lista</u> (set)
relacionales	relacional	registro (tupla), <u>tabla</u> (relación)
objeto- relacionales	relacional + OO	registro (tupla), tabla (relación) + constructores de tipos
OO	OO	constructores de tipos

1. Modelo relacional de datos

Estructuras de datos: tupla, relación

Tupla: tupla \longleftrightarrow registro

Un esquema de tupla es un conjunto de pares de la forma:

$$\{(A_1, D_1), (A_2, D_2), \dots, (A_n, D_n)\}$$

Una tupla de esquema $\{(A_1, D_1), (A_2, D_2), \dots, (A_n, D_n)\}$ es un conjunto de pares (nombre_atributo, valor) de la forma:

$$\{(A_1, v_1), (A_2, v_2), \dots, (A_n, v_n)\} \text{ tal que } \forall i \ v_i \in D_i$$

1. Modelo relacional de datos

Estructuras de datos: tupla, relación

Una relación es un conjunto de tuplas del mismo esquema al que se denomina esquema de la relación

Definición de una **relación** R de esquema:
 $\{(A_1, D_1), (A_2, D_2), \dots, (A_n, D_n)\}$

esquema de R 

definición de R 

$R (A_1:D_1, A_2:D_2, \dots, A_n:D_n)$

valor de R 

$R = \{t: t = \{(A_1, v_1), (A_2, v_2), \dots, (A_n, v_n)\} \mid \forall i \ v_i \in D_i\}$

Los dominios del esquema de una relación deben ser escalares (SQL92)

1. Modelo relacional de datos

Álgebra Relacional

Operadores de la estructura relación:

- ✓ **insertar** una tupla en una relación
- ✓ **borrar** una tupla de una relación
- ✓ **seleccionar** las tuplas de una relación que cumplen una condición
- ✓ **concatenar** (unir) las tuplas de dos relaciones por el valor de uno (o varios) atributos.
- ✓ **unión** de relaciones
- ✓ **diferencia** de relaciones
- ✓ **intersección** de relaciones
- ✓

1. Modelo relacional de datos

Álgebra Relacional

operadores
constructores



inserción
borrado

operadores
consultores



selección
proyección
concatenación
división

operadores relacionales

unión
diferencia
intersección
producto cartesiano

operadores conjuntistas

1. Modelo relacional de datos

El lenguaje estándar SQL

- lenguaje de tipo lógico (declarativo)
- (sub)lenguaje de definición de datos (DDL)
- (sub)lenguaje de manipulación de datos (DML)

1. Modelo relacional de datos

El lenguaje estándar SQL

Definición de un esquema relacional en SQL.

- definiciones de dominios

`CREATE DOMAIN ...`

- definiciones de relaciones: esquema y restricciones.

`CREATE TABLE ...`

- definiciones de restricciones de integridad generales

`CREATE ASSERTION ...`

1. Modelo relacional de datos

El lenguaje estándar SQL

Manipulación de datos (consulta y actualización):


SELECT (consulta)

INSERT (inserción de tuplas)

DELETE (borrado de tuplas)

UPDATE (modificación de tuplas)

Tecnología Relacional

1. Modelo relacional de datos.
2. Independencia de datos: arquitectura ANSI/SPARC. 
3. Integridad de los datos.
4. Evolución de la tecnología relacional.
5. Implementaciones

2. Independencia de datos: arquitectura ANSI/SPARC

Independencia de datos: independencia de las aplicaciones (programas) respecto a la representación física (implementación) de los datos.

independencia de datos



definición de la base de datos a distintos niveles de abstracción



esquemas de base de datos

2. Independencia de datos: arquitectura ANSI/SPARC

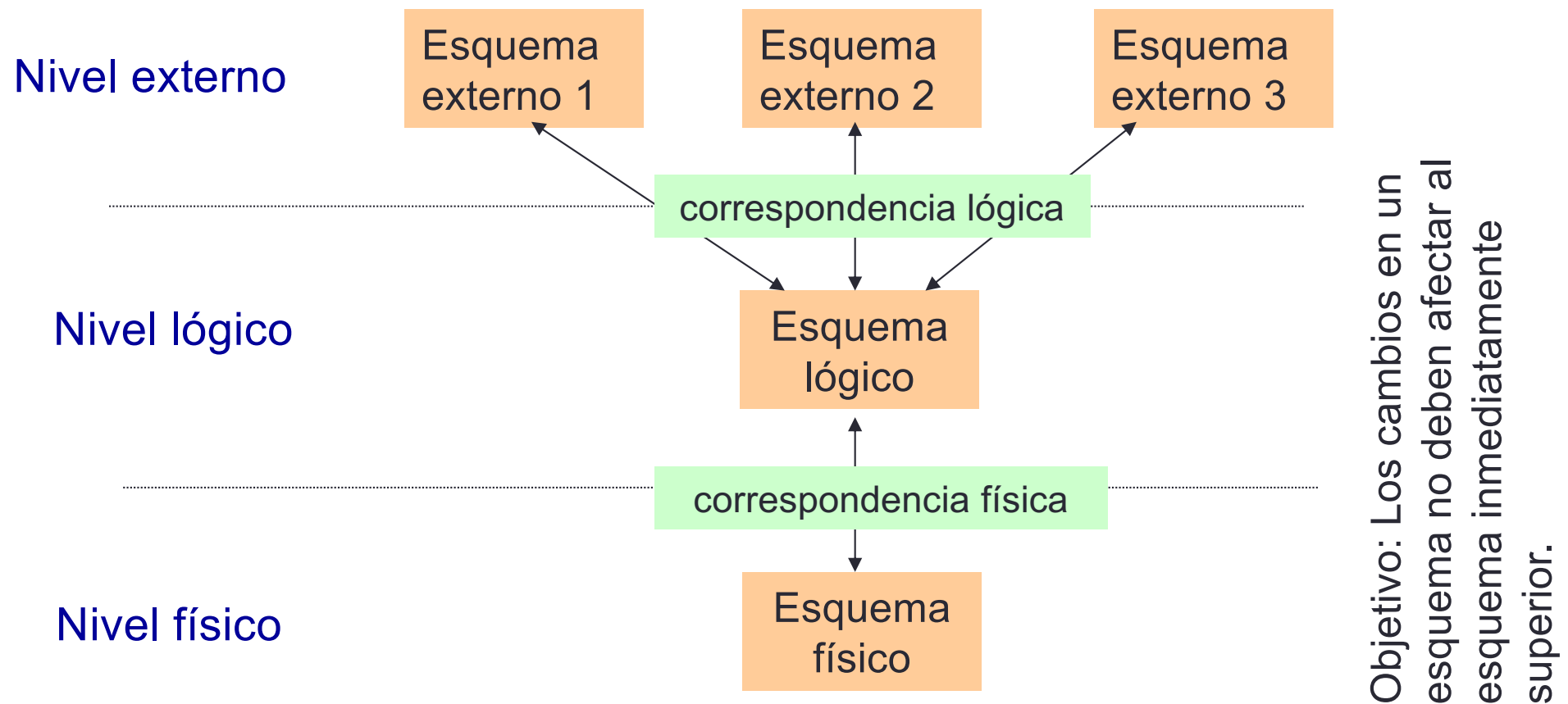
Arquitectura de niveles del SGBD
(Arquitectura ANSI/SPARC)

esquema externo: “subconjunto” del esquema lógico (vistas).

esquema lógico: definición de las estructuras de datos de la base de datos.

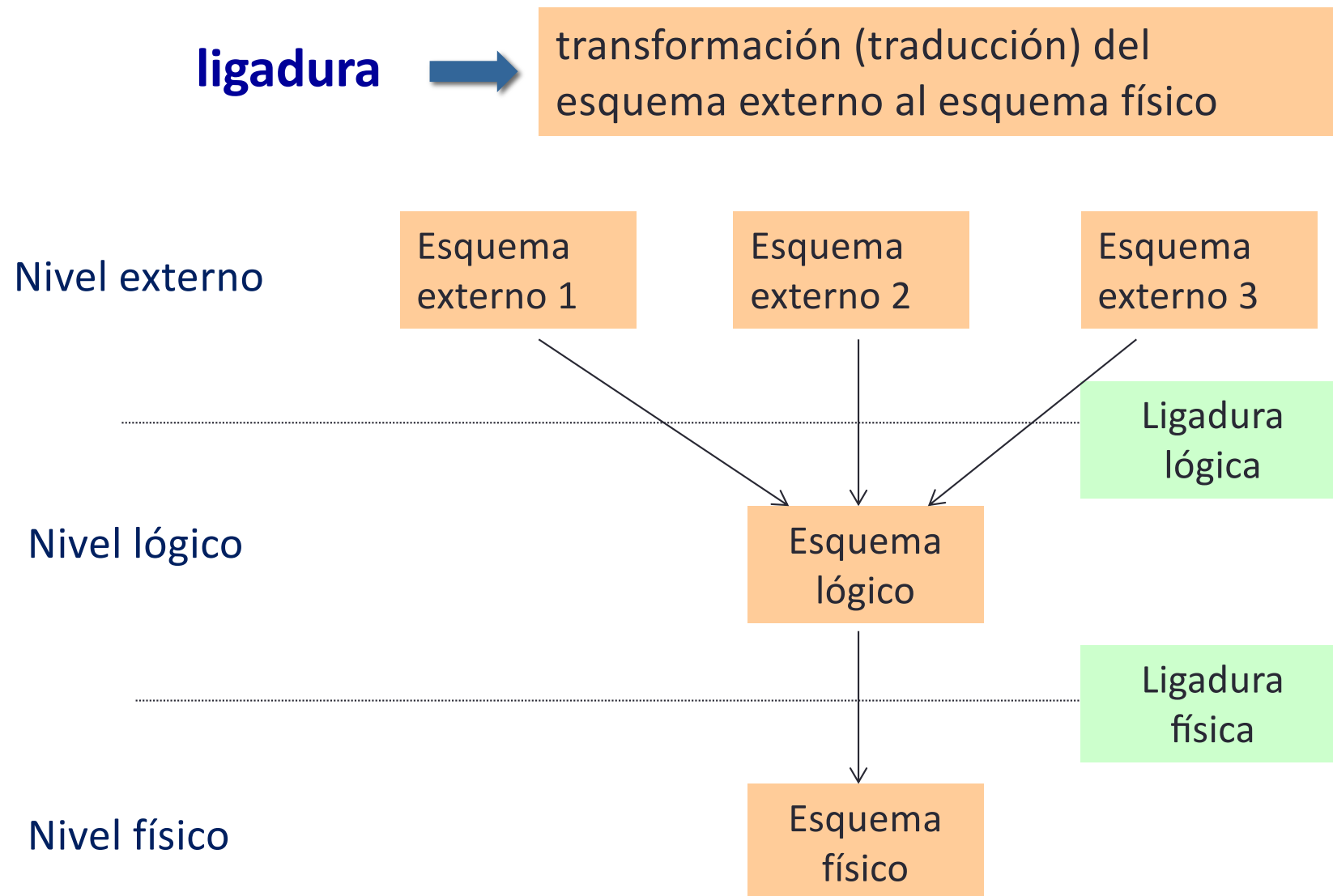
esquema físico: implementación de las estructuras de datos definidas en el esquema lógico.

2. Independencia de datos: arquitectura ANSI/SPARC

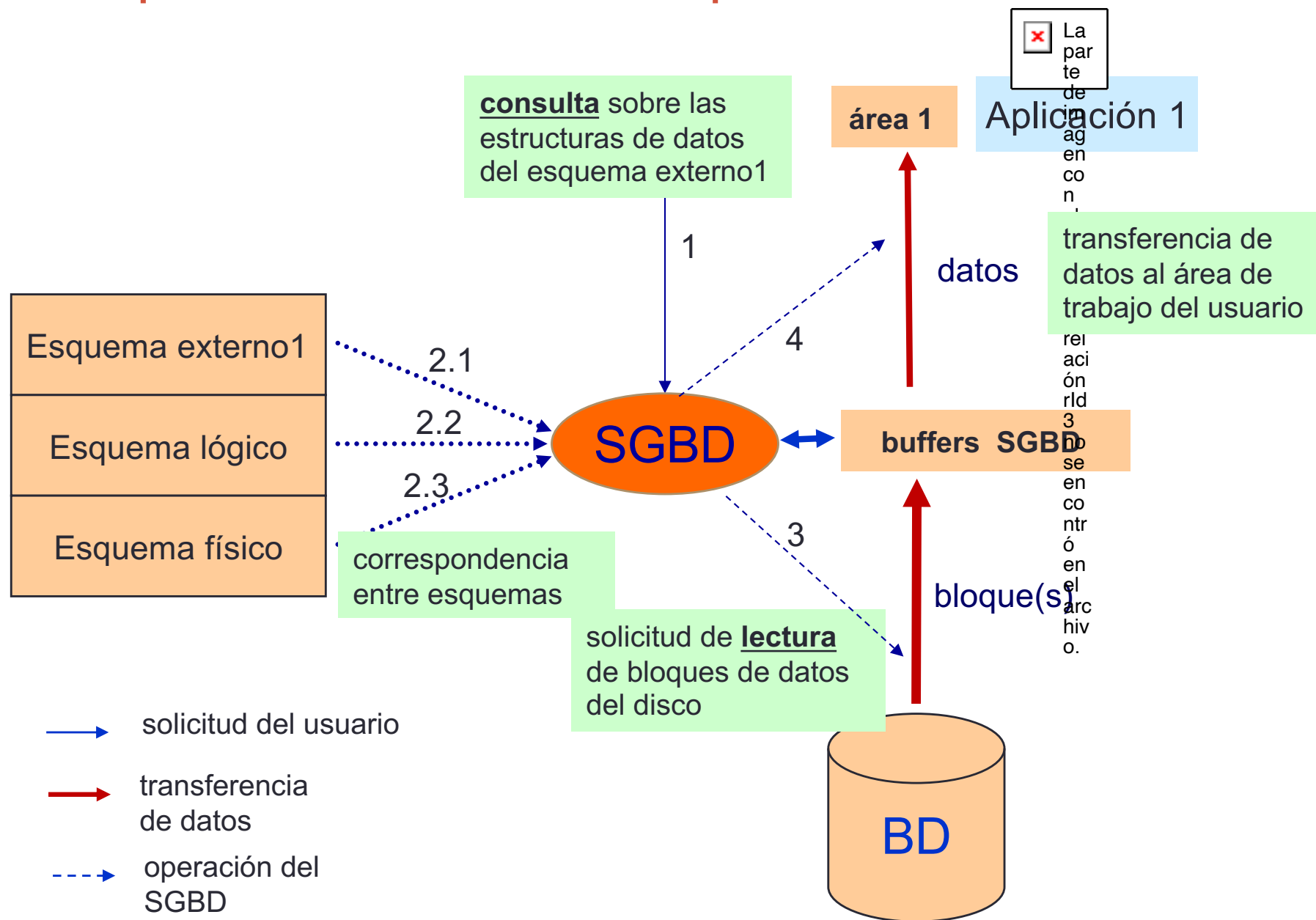


Correspondencia: definición de cada *elemento* de un esquema en términos de *elementos* del esquema inmediatamente inferior

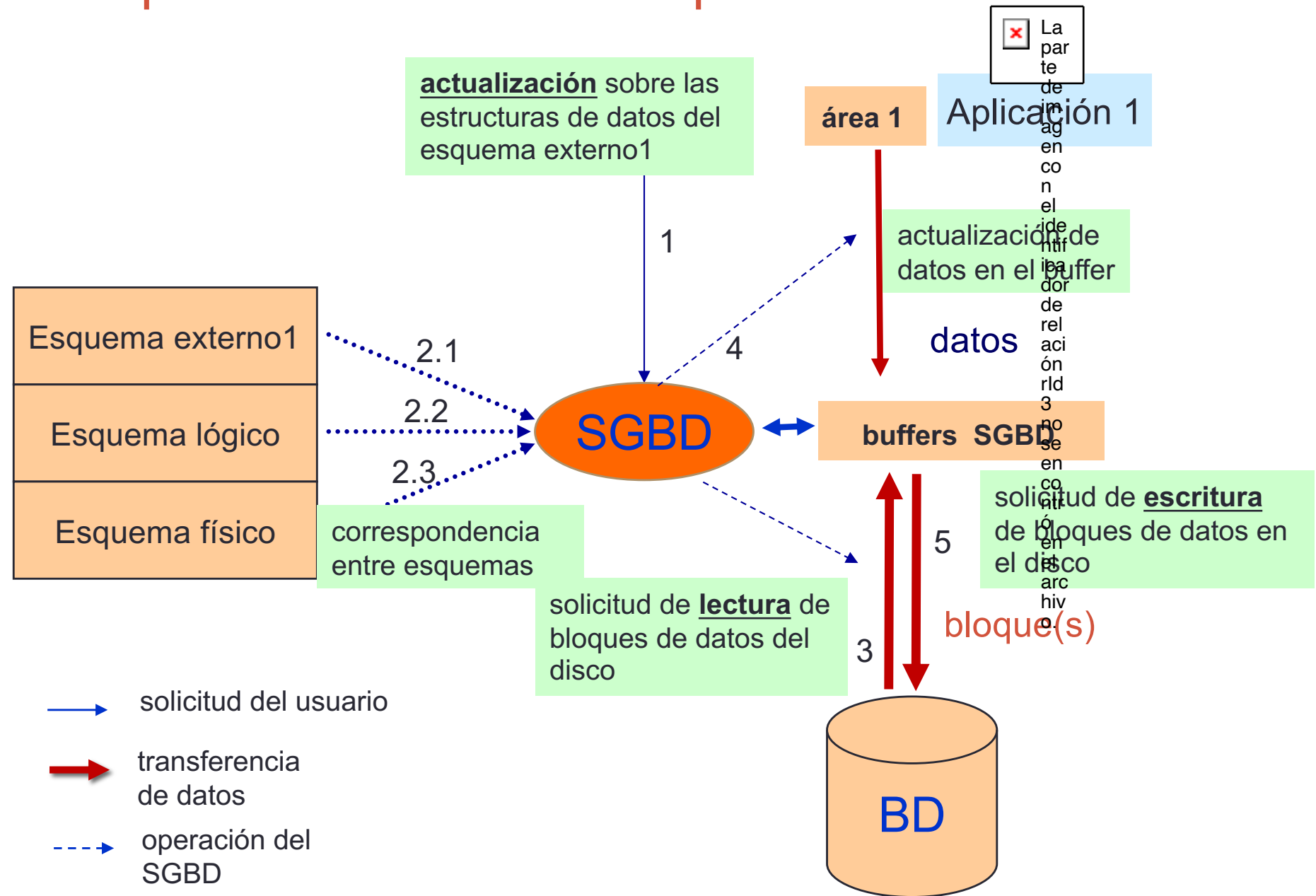
2. Independencia de datos: arquitectura ANSI/SPARC



2. Independencia de datos: arquitectura ANSI/SPARC



2. Independencia de datos: arquitectura ANSI/SPARC



2. Independencia de datos: arquitectura ANSI/SPARC

Definición del **esquema lógico** en SQL.

esquema lógico: definición de las estructuras de datos de la base de datos.

- ✓ definiciones de dominios

CREATE DOMAIN ...

- ✓ definiciones de relaciones: esquema y restricciones.

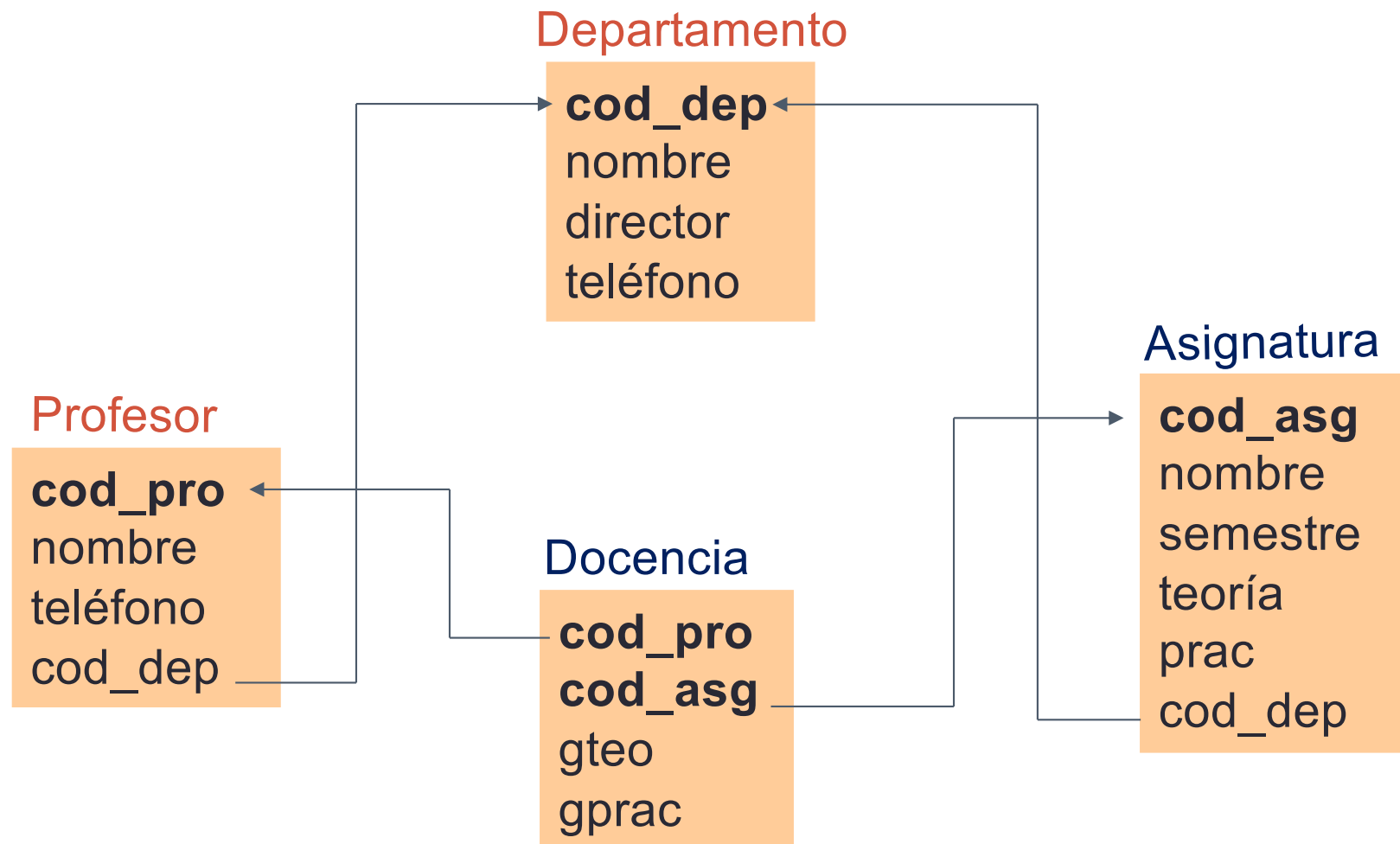
CREATE TABLE ...

- ✓ definiciones de restricciones de integridad generales

CREATE ASSERTION ...

2. Independencia de datos: arquitectura ANSI/SPARC

Ejemplo: base de datos “Docencia”



2. Independencia de datos: arquitectura ANSI/SPARC

CREATE TABLE Departamento

```
(cod_dep CHAR (5),
nombre VARCHAR(40) NOT NULL,
director CHAR (25),
teléfono CHAR (15),
PRIMARY KEY (cod_dep) )
```

CREATE TABLE Profesor

```
(cod_pro CHAR (5) ,
nombre VARCHAR (40) NOT NULL,
teléfono CHAR (15),
cod_dep CHAR (5) NOT NULL,
PRIMARY KEY (cod_pro),
FOREING KEY (cod_dep) REFERENCES Departamento (cod_dep) )
```

CREATE TABLE Asignatura

```
(cod_asg CHAR (5) ,
nombre VARCHAR (40) NOT NULL,
semestre CHAR (2),
teoría NUMBER,
prac NUMBER,
cod_dep CHAR (5) NOT NULL,
PRIMARY KEY (cod_asg),
FOREING KEY (cod_dep) REFERENCES Departamento (cod_dep) )
```

CREATE TABLE Docencia

```
(cod_pro CHAR (5) ,
cod_asg CHAR (5) ,
gteo NUMBER,
gprac NUMBER,
PRIMARY KEY (cod_pro, cod_asg),
FOREING KEY (cod_pro) REFERENCES Profesor (cod_pro),
FOREING KEY (cod_asg) REFERENCES Asignatura (cod_asg) )
```

Esquema lógico

2. Independencia de datos: arquitectura ANSI/SPARC

Definición del **esquema externo** en SQL.

esquema externo: “subconjunto” del esquema lógico (vistas).

Definición de vistas

definición_vista::=

CREATE VIEW nombre_vista

[(nom-atributo1, nom-atributo2, ...)] AS *sentencia_SELECT*

[WITH CHECK OPTION]

2. Independencia de datos: arquitectura ANSI/SPARC

```
CREATE VIEW Profesor-DSIC
```

```
AS SELECT cod_pro, nombre, teléfono  
FROM Profesor  
WHERE cod_dep = 'DSIC'
```

Esquema externo
del DSIC

```
CREATE VIEW Asignatura-DSIC
```

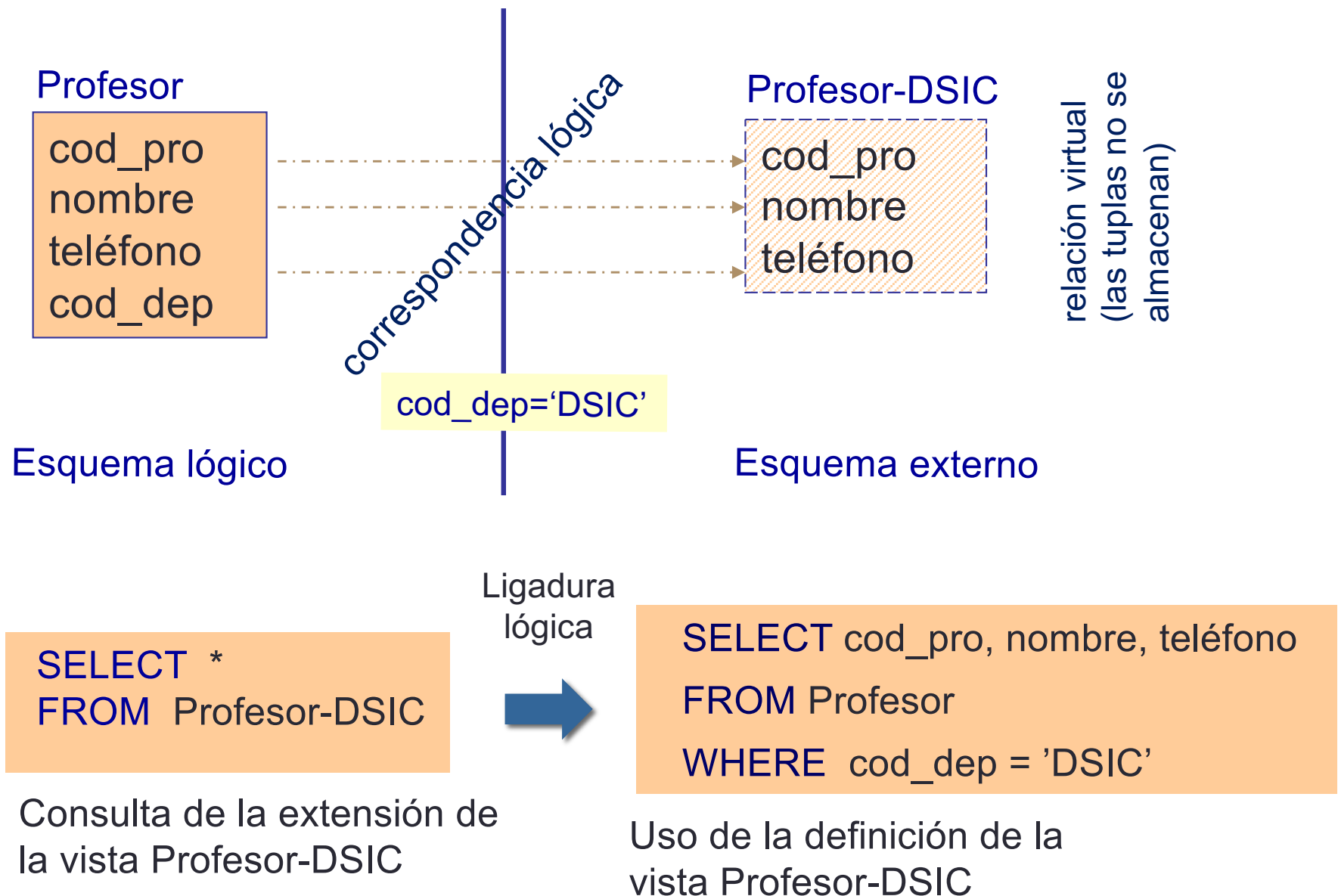
```
AS SELECT cod_asg, nombre, semestre, teoría, prac  
FROM Asignatura  
WHERE cod_dep = 'DSIC'
```

```
CREATE VIEW Docencia-DSIC
```

```
AS SELECT cod_pro, cod_asg, gteo, gprac  
FROM Docencia  
WHERE cod_asg IN (SELECT cod_asg  
FROM Asignatura  
WHERE cod_dep = 'DSIC')
```

Esquema externo del departamento DSIC

2. Independencia de datos: arquitectura ANSI/SPARC



2. Independencia de datos: arquitectura ANSI/SPARC

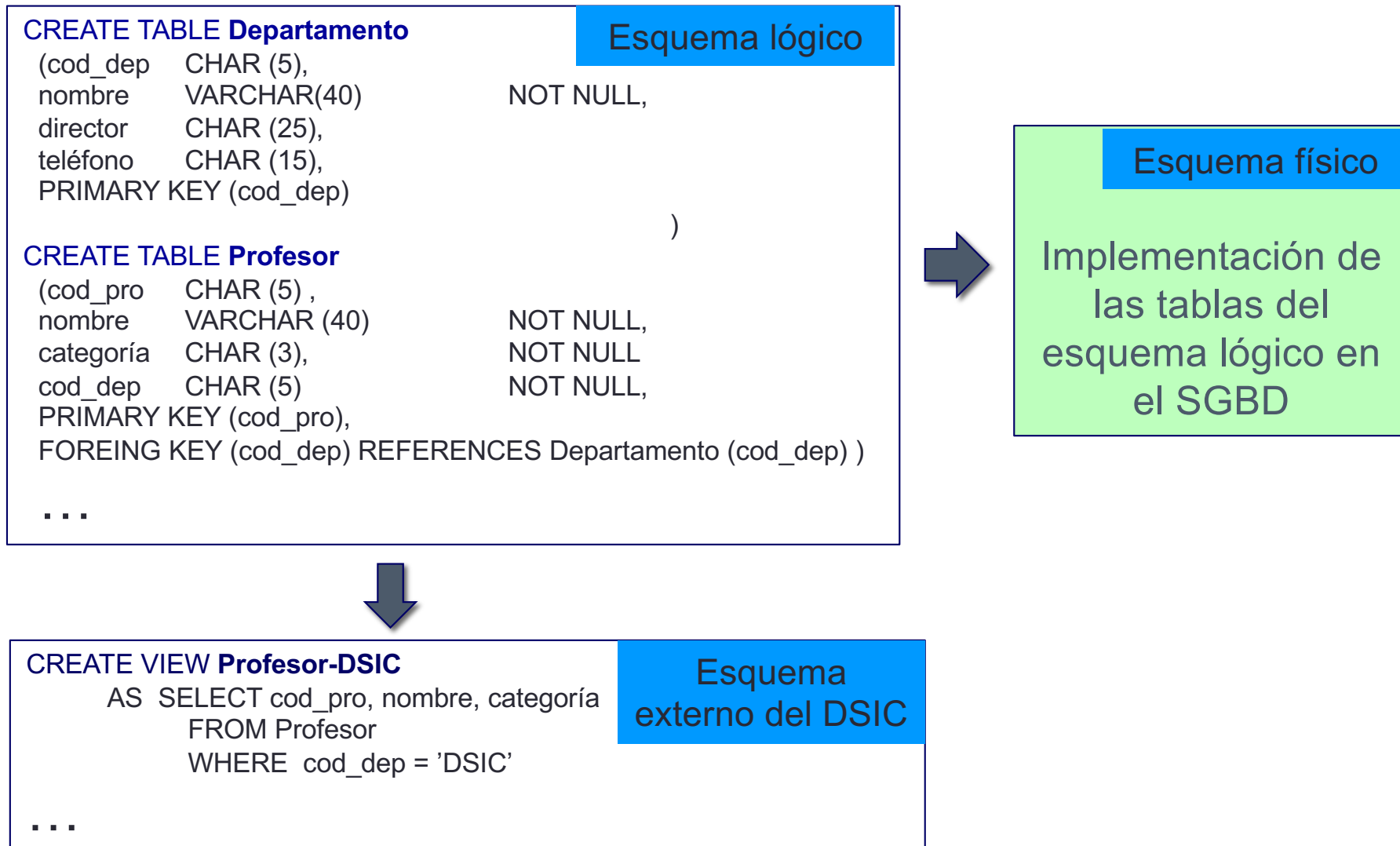
Un esquema externo en SQL es un conjunto de definiciones de vistas.

Una vista es una relación virtual cuyas tuplas no se almacenan explícitamente: la extensión de la vista se calcula cuando ésta es consultada.

En la definición de la vista se especifica el esquema de la relación (vista) y su extensión implícita dada en forma de sentencia SELECT.

```
CREATE VIEW Profesor-DSIC  
AS SELECT cod_pro, nombre, teléfono  
FROM Profesor  
WHERE cod_dep = 'DSIC'
```

2. Independencia de datos: arquitectura ANSI/SPARC



2. Independencia de datos: arquitectura ANSI/SPARC

ligadura



transformación (traducción) del
esquema externo al esquema físico

```
SELECT nombre  
FROM Profesor-DSIC  
WHERE categoría='CU'
```

Consulta sobre la vista
Profesor-DSIC

Ligadura
lógica



```
SELECT nombre  
FROM Profesor  
WHERE cod_dep = 'DSIC'  
AND categoría = 'CU'
```

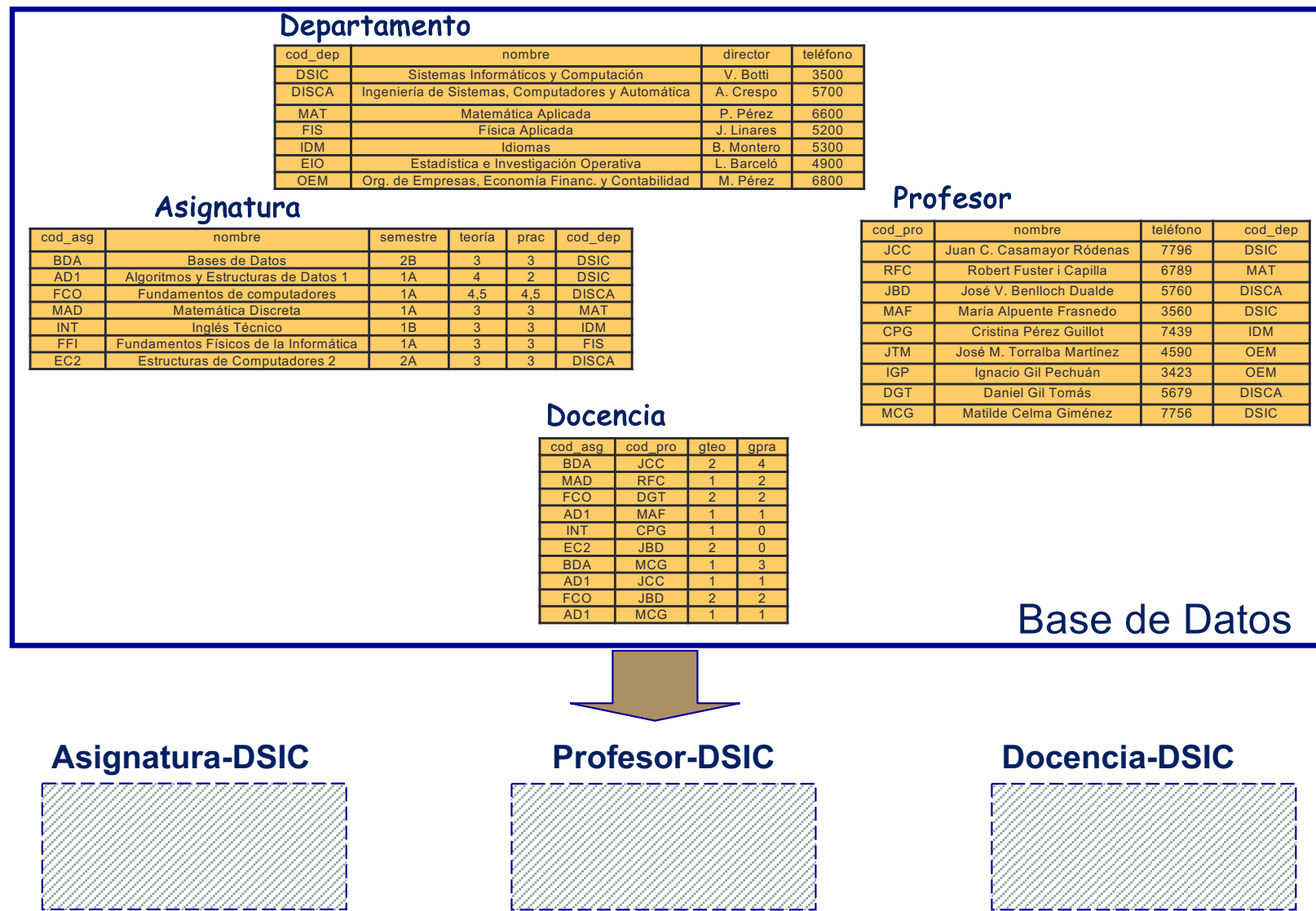
Uso de la definición de la
vista Profesor-DSIC

Ligadura
física



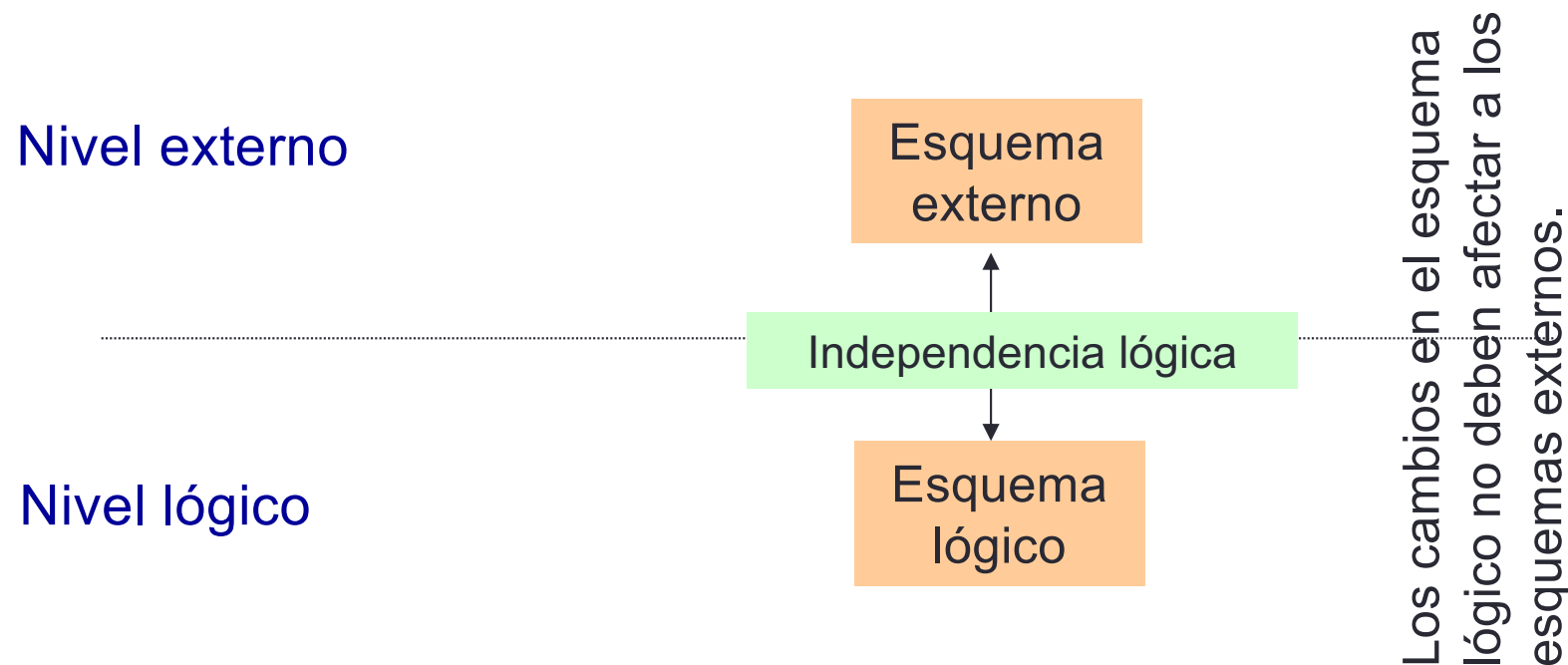
Operaciones de
lectura de bloques
de disco

2. Independencia de datos: arquitectura ANSI/SPARC



Esquema externo: Vistas (relaciones virtuales)

2. Independencia de datos: arquitectura ANSI/SPARC



Modificaciones del esquema lógico que no obligarían a modificar el esquema externo del DSIC en un SGBD con independencia lógica:

- ✓ eliminar la relación Departamento
- ✓ ampliar la relación Profesor con más atributos

2. Independencia de datos: arquitectura ANSI/SPARC

Modificaciones del esquema lógico que no obligarían a modificar el esquema externo del DSIC en un SGBD con independencia lógica:

“Si se reestructura el esquema lógico reagrupando los datos en estructuras distintas a las originales, las aplicaciones desarrolladas sobre el esquema externo del DSIC no se verán afectadas siempre que sea posible definir el mismo esquema externo (vistas) a partir del nuevo esquema lógico. Sólo será preciso cambiar la definición de las vistas (correspondencia lógica)”


2. Independencia de datos: arquitectura ANSI/SPARC

“Docencia del DSIC por semestres”:

Consulta en el esquema externo del DSIC

```
SELECT A.semestre, A.cod_asg, SUM(D.gteo), SUM(D.gprac)
FROM Asignatura-DSIC A, Docencia-DSIC D
WHERE A.cod_asg = D.cod_asg
GROUP BY A.semestre, A.cod_asg
```

SGBD
ligadura lógica



se combina la consulta con la
definición de las vistas

```
SELECT A.semestre, A.cod_asg, SUM(D.gteo), SUM(D.gprac)
FROM Asignatura A, Docencia D
WHERE A.cod_asg = D.cod_asg
      AND
      A.cod_dep = 'DSIC'
      AND
      D.cod_asg IN (SELECT cod_asg FROM Asignatura
                   WHERE cod_dep='DSIC')
GROUP BY A.semestre, A.cod_asg
```


2. Independencia de datos: arquitectura ANSI/SPARC

Actualización de vistas:

- ✓ Una operación de actualización sobre una vista debe traducirse en las operaciones de actualización sobre las relaciones básicas (subyacentes a la vista) que sean necesarias para satisfacer el requerimiento de actualización del usuario.
- ✓ Debido a que una vista puede definirse a través de cualquier sentencia SELECT, la traducción de una operación de actualización sobre la vista en actualizaciones sobre las relaciones básicas no es siempre posible ya que pueden presentarse ambigüedades, por este motivo la actualización de vistas está sometida a ciertas restricciones que evitan esta posible ambigüedad.
- ✓ Cuando una vista se define con la opción [WITH CHECK OPTION] el sistema rechaza cualquier actualización (INSERT o UPDATE) de tuplas sobre la vista que violen su definición (que no podrían ser seleccionadas por la vista).

2. Independencia de datos: arquitectura ANSI/SPARC

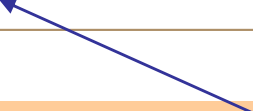
Definición de vistas

definición_vista::=

CREATE VIEW nombre_vista

[(nom-atributo1, nom-atributo2, ...)] AS *sentencia_SELECT*

[WITH CHECK OPTION]



prohíbe las actualizaciones a través de la vista que violen la definición de la vista.

2. Independencia de datos: arquitectura ANSI/SPARC

```
CREATE VIEW Primer-Ciclo
  SELECT cod_asg, nombre, semestre, teoría, prac, cod_dep
  FROM Asignatura
  WHERE semestre IN ('1A', '2A', '3A', '1B', '2B', '3B')
  WITH CHECK OPTION
```

```
INSERT INTO Primer-Ciclo
  VALUES ('PR2', 'Programación2', '4A', 3, 3, 'DSIC')
```

2. Independencia de datos: arquitectura ANSI/SPARC

```
CREATE VIEW Profesor-DSIC
```

```
AS SELECT cod_pro, nombre, teléfono  
FROM Profesor  
WHERE cod_dep= 'DSIC'
```

```
CREATE VIEW Asignatura-DSIC
```

```
AS SELECT cod_asg, nombre, semestre, teoría, prac  
FROM Asignatura  
WHERE cod_dep= 'DSIC'
```

```
CREATE VIEW Docencia-DSIC
```

```
AS SELECT D.cod_pro, D.cod_asg, D.gteo, D.gprac  
FROM Asignatura A, Docencia D  
WHERE cod_asg IN (SELECT cod_asg  
FROM Asignatura  
WHERE cod_dep='DSIC'))
```

Actualización en el
esquema externo del
DSIC

```
DELETE FROM Profesor-DSIC WHERE cod_pro= 'MCG'
```

2. Independencia de datos: Arquitectura ANSI/SPARC

```
DELETE FROM Profesor-DSIC
WHERE cod_pro= 'MCG'
```

Ligadura
lógica



```
CREATE VIEW Profesor-DSIC AS
SELECT cod_pro, nombre, teléfono
FROM Profesor
WHERE cod_dep= 'DSIC'
```

```
DELETE FROM Profesor
WHERE cod_pro='MCG'
AND
cod_dep='DSIC'
```

Existe una correspondencia
entre las tuplas de la vista
Profesor-DSIC y las tuplas de
la relación *Profesor*.

Profesor-DSIC

cod_pro	nombre	teléfono
JCC	Juan C. Casamayor Ródenas	7796
MAF	María Alpuente Frasnado	3560
MCG	Matilde Celma Giménez	7756

Profesor

cod_pro	nombre	teléfono	cod_dep
JCC	Juan C. Casamayor Ródenas	7796	DSIC
RFC	Robert Fuster i Capilla	6789	MAT
JBD	José V. Benlloch Dualde	5760	DISCA
MAF	María Alpuente Frasnado	3560	DSIC
CPG	Cristina Pérez Guillot	7439	IDM
JTM	José M. Torralba Martínez	4590	OEM
IGP	Ignacio Gil Pechuán	3423	OEM
DGT	Daniel Gil Tomás	5679	DISCA
MCG	Matilde Celma Giménez	7756	DSIC

2. Independencia de datos: arquitectura ANSI/SPARC

Docencia (créditos) asignada
a los departamentos

```
CREATE VIEW Docencia-Dptos (codigo, nombre, credits)
AS SELECT D.cod_dep, D.nombre,
        SUM (Doc.gteo * A.teoría + Doc.gprac * A.prac)
FROM Asignatura A, Departamento D, Docencia Doc
WHERE A.cod_asg = Doc.cod_asg
AND
        A.cod_dep = D.cod_dep
GROUP BY D.cod_dep, D.nombre
```

~~UPDATE Docencia-Dptos
SET credits=credits + 10
WHERE codigo= 'DSIC'~~

No existe una correspondencia entre las tuplas de la vista *Docencia-Dptos* y las tuplas de la relación *Docencia*.

2. Independencia de datos: arquitectura ANSI/SPARC

```
CREATE VIEW Docencia-DSIC
AS SELECT cod_pro, cod_asg, gteo, gprac
FROM Docencia
WHERE cod_asg IN (SELECT cod_asg
                  FROM Asignatura
                  WHERE cod_dep='DSIC')
```

```
UPDATE Docencia-DSIC
SET gteo=gteo+1
WHERE cod_pro= 'MCG'
      AND
      cod_asg= 'BDA'
```

2. Independencia de datos: arquitectura ANSI/SPARC

```
UPDATE Docencia-DSIC
SET gteo=gteo+1
WHERE cod_pro='MCG'
      AND
      cod_asg='BDA'
```



```
UPDATE Docencia
SET gteo=gteo+1
WHERE cod_pro='MCG'
      AND
      cod_asg='BDA'
      AND
      cod_asg IN (SELECT cod_asg
                  FROM Asignatura
                  WHERE cod_dep='DSIC')
```

Docencia-DSIC

cod_asg	cod_pro	gteo	gpra
BDA	JCC	2	4
AD1	MAF	1	1
BDA	MCG	1	3
AD1	JCC	1	1
AD1	MCG	1	1

Docencia

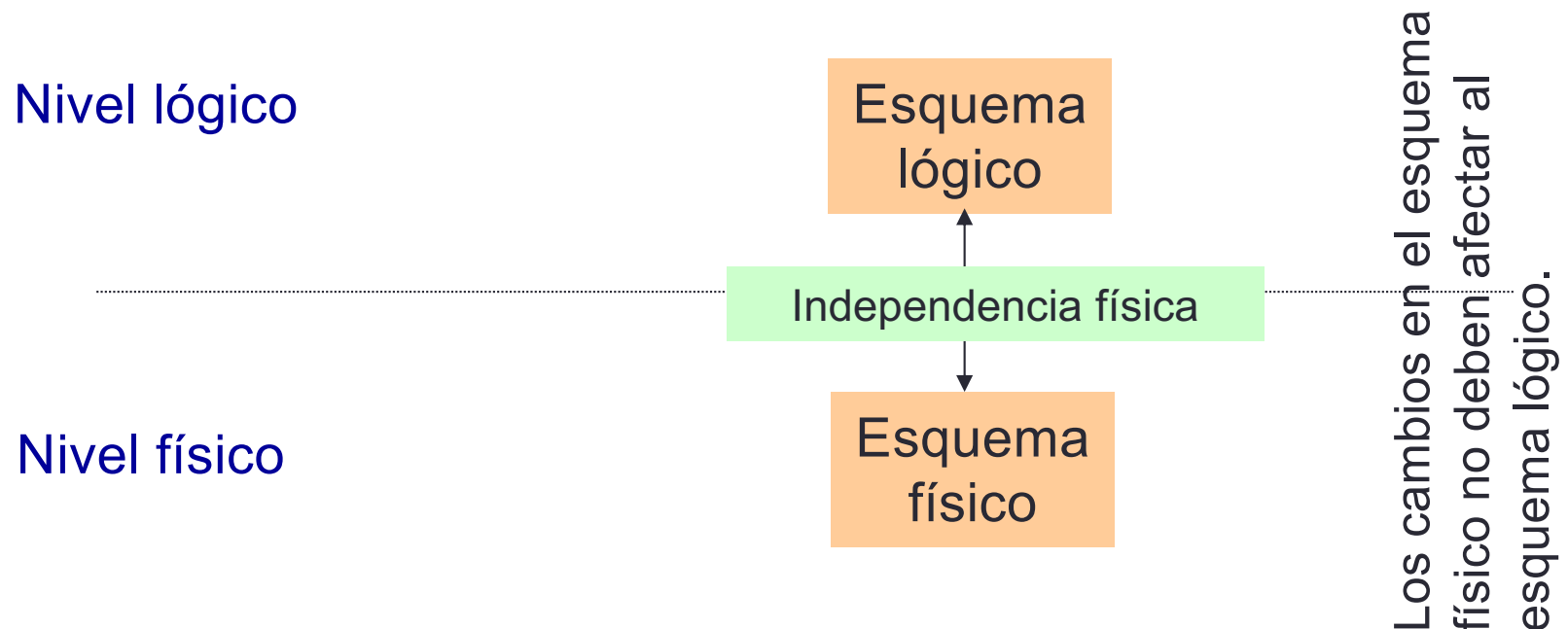
cod_asg	cod_pro	gteo	gpra
BDA	JCC	2	4
MAD	RFC	1	2
FCO	DGT	2	2
AD1	MAF	1	1
INT	CPG	1	0
EC2	JBD	2	0
BDA	MCG	1	3
AD1	JCC	1	1
FCO	JBD	2	2
AD1	MCG	1	1

Existe una correspondencia entre las tuplas de la vista Docencia-DSIC y las tuplas de la relación Docencia.

2. Independencia de datos: arquitectura ANSI/SPARC

Definición del **esquema físico** de la base de datos.

esquema físico: implementación de las estructuras de datos definidas en el esquema lógico.



2. Independencia de datos: arquitectura ANSI/SPARC

Definición del **esquema físico** de la base de datos.

- Las estructuras de datos de la BD se implementan como ficheros almacenados en memoria secundaria.

Fichero: estructura de datos en memoria secundaria

- secuencia de registros

- El espacio en disco se divide en bloques

Bloque:

- unidad de direccionamiento en disco
- unidad de transferencia de datos
- fichero: secuencia de bloques

- Los bloques son transferidos a buffers de memoria principal para consultar y actualizar los datos de la BD

2. Independencia de datos: arquitectura ANSI/SPARC

SGBD



objetivo

Procesar eficientemente las operaciones
(consulta y actualización) sobre la BD



Elegir la implementación física más
adecuada de la base de datos

2. Independencia de datos: arquitectura ANSI/SPARC

Las estructuras de datos de la BD (tablas) se implementan como ficheros almacenados en memoria secundaria:

tabla --> organización de fichero

tupla --> registro

Formato del fichero (SGBD)

- Formato de registro: distribución de los campos en el registro
- Formato de bloque: distribución de los registros en el bloque
- Directorio de bloques: distribución de los bloques en el disco.

Organización del fichero: organización de los registros en secuencia (decisiones de diseño).

- Fichero desordenado
- Fichero ordenado
- Fichero disperso (direccionamiento calculado)

2. Independencia de datos: arquitectura ANSI/SPARC

- Fichero **desordenado**: los registros están ordenados en el orden de inserción (los nuevos registros se insertan al final del fichero)
- Fichero **ordenado**: los registros están ordenados según el valor de alguno(s) de sus campos (**campo de ordenación**).
- Fichero **disperso** (Hash): los registros se almacenan en bloques determinados (**función de dispersión**) por el valor de un campo del registro (**campo de dispersión**). (ver Anexo I)

2. Independencia de datos: arquitectura ANSI/SPARC

Fichero ordenado

Operación	Coste
BuscarRegistro (campo de ordenación)	Eficiente (binaria)
BuscarRegistro (otro campo)	Ineficiente (lineal)
BúsquedaOrdenada (campo de ordenación)	Muy eficiente
Inserción	Muy ineficiente
Borrado	Eficiente
Modificación	Eficiente

Fichero desordenado

Operación	Coste
BuscarRegistro	Ineficiente (lineal)
BúsquedaOrdenada	Muy ineficiente
Inserción	Muy eficiente
Borrado	Ineficiente
Modificación	Ineficiente

Fichero disperso (Hash)

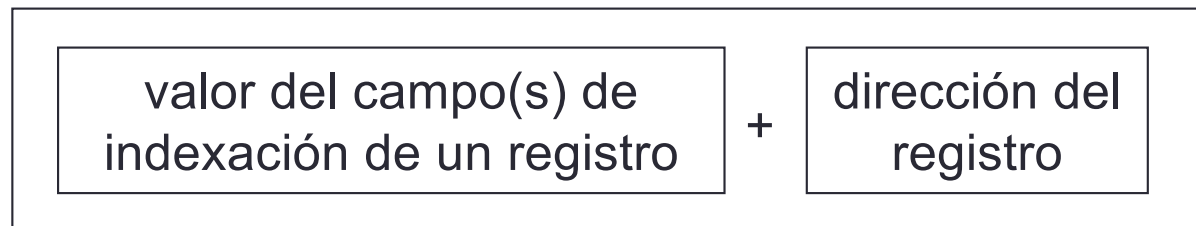
Operación	Coste
BuscarRegistro (campo de direccionamiento)	Muy eficiente
BuscarRegistro (otro campo)	Ineficiente (lineal)
BúsquedaOrdenada	Ineficiente
Inserción	Muy eficiente
Borrado	Muy eficiente
Modificación	Muy eficiente

2. Independencia de datos: arquitectura ANSI/SPARC

Índice: (ver Anexo I)

- Estructura de datos en memoria secundaria que permite el acceso a los registros de un fichero por el valor de un campo(s): campo(s) de indexación.

Elementos de un índice (entradas del índice):



- Los índices permiten el acceso directo y el acceso ordenado a los registros del fichero por el campo(s) de indexación.
- Estructuras de datos utilizadas para implementar índices:

Árboles de búsqueda → **Índices en árbol B**

Los árboles B y B⁺ son árboles n-arios de búsqueda equilibrados que garantizan una ocupación eficiente del espacio en los nodos.

2. Independencia de datos: arquitectura ANSI/SPARC

Implementación (estándar) de una base de datos relacional:

□ Tabla --> organización de fichero

tupla --> registro

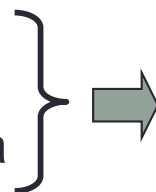
□ Estructuras de acceso --> índices

□ Tabla  Fichero desordenado

○ Excepciones:

▪ Tabla pequeña

▪ Tabla muy estática



Fichero ordenado
Fichero disperso

□ Atributos (unicidad)  índices B⁺

□ Atributos utilizados en:


○ Condiciones de igualdad

○ Condiciones de rango

○ Usados en concatenaciones (claves ajenas)



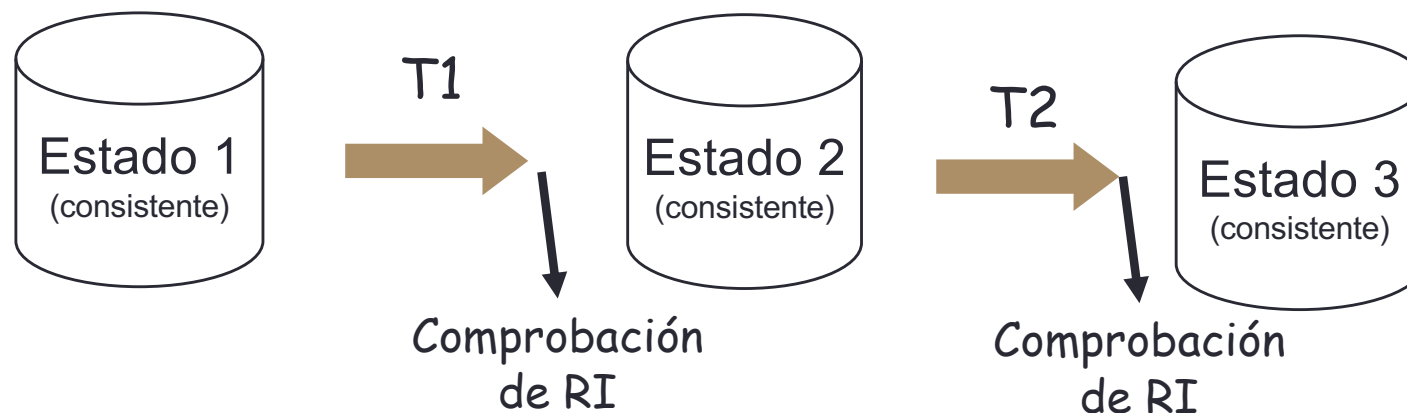
Tecnología Relacional

1. Modelo relacional de datos.
2. Independencia de datos: arquitectura ANSI/SPARC.
3. Integridad de los datos. 
4. Evolución de la tecnología relacional.
5. Implementaciones

3. Integridad de los datos

Integridad de los datos: calidad de los datos almacenados.

- una BD evoluciona por la ejecución de operaciones de actualización de los usuarios
- necesidad del concepto de transacción: unidad lógica de ejecución.
- el SGBD debe ejecutar correctamente las transacciones (en un entorno concurrente) y asegurar que en cada estado de la BD se satisfacen las restricciones de integridad del esquema.



3. Integridad de los datos

TRANSACCIÓN

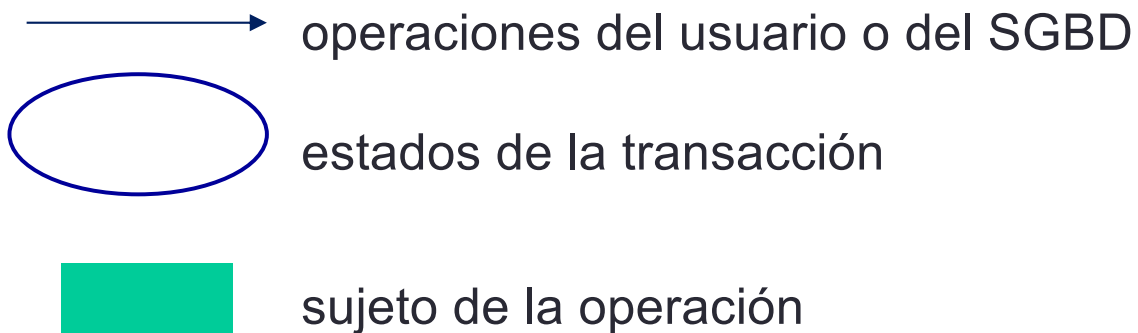
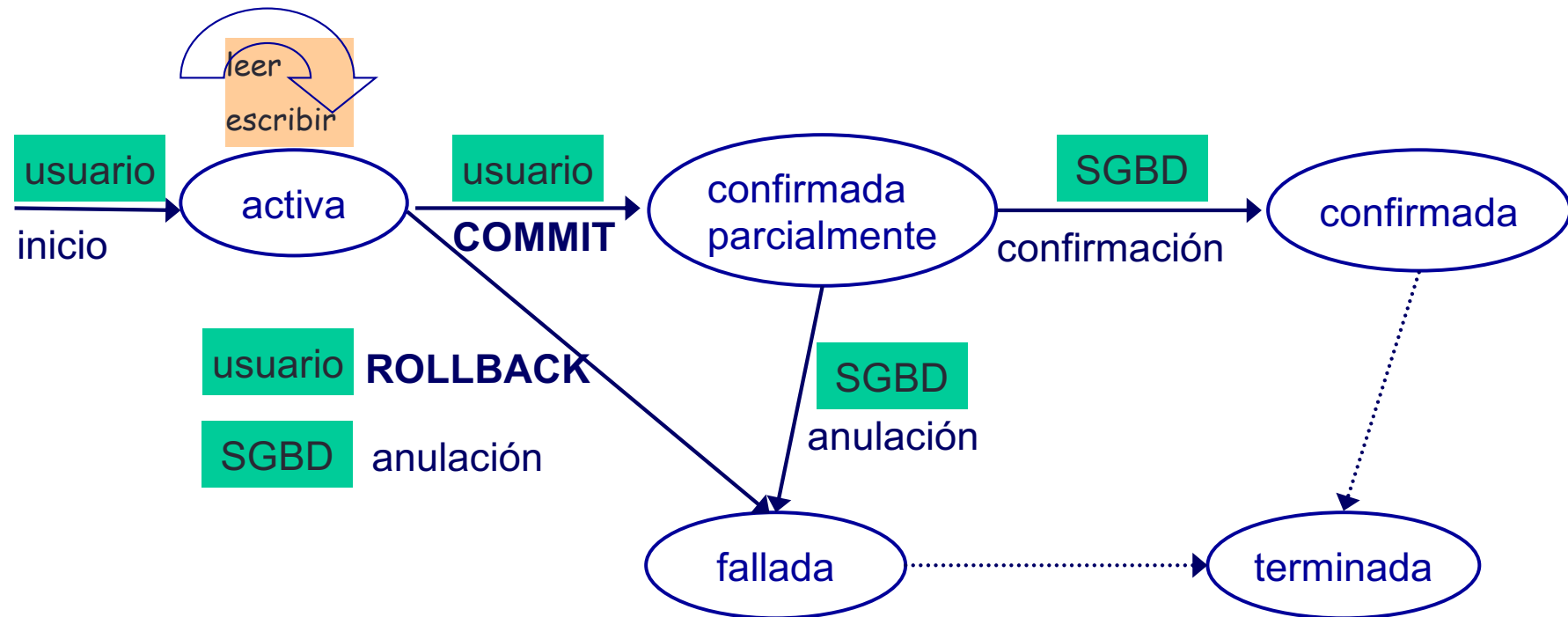


Secuencia de operaciones de acceso a la base de datos (consulta o actualización) que constituyen una unidad lógica de ejecución

- (a) todas las operaciones de la transacción se ejecutan con éxito y su efecto queda registrado permanentemente en la BD, o
- (b) la transacción no tiene ningún efecto en la BD ni en otras transacciones (**entorno concurrente**).

Principio **ACID** en el procesamiento de transacciones:
Atomicidad, **C**onsistencia, **A**islamiento, **P**ersistencia.

3. Integridad de los datos



3. Integridad de los datos

Procesamiento de transacciones:

✓ transacciones confirmadas (por el usuario): **COMMIT**

Las actualizaciones de una transacción confirmada (definitivamente por el SGBD) deben quedar registradas permanentemente en la base de datos.

✓ transacciones anuladas (por el usuario o por el SGBD):

- usuario: **ROLLBACK**

- SGBD: comprobación de RI, control de la concurrencia, errores en la ejecución, ...

Las actualizaciones de una transacción anulada no deben quedar registradas en la base de datos.

✓ transacciones interrumpidas (por un fallo):

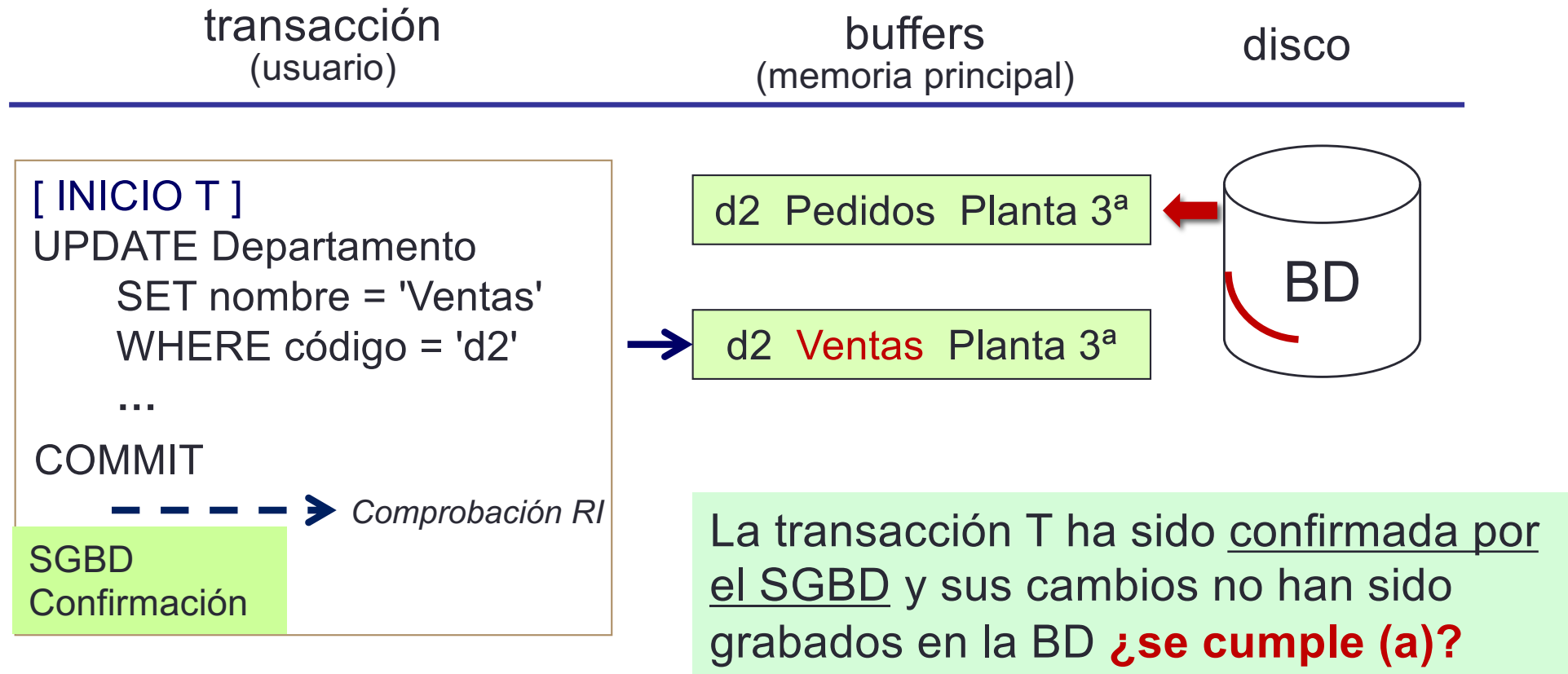
Las actualizaciones de una transacción interrumpida no deben quedar registradas en la base de datos.

3. Integridad de los datos

Aspectos relacionados con el mantenimiento de la integridad de los datos:

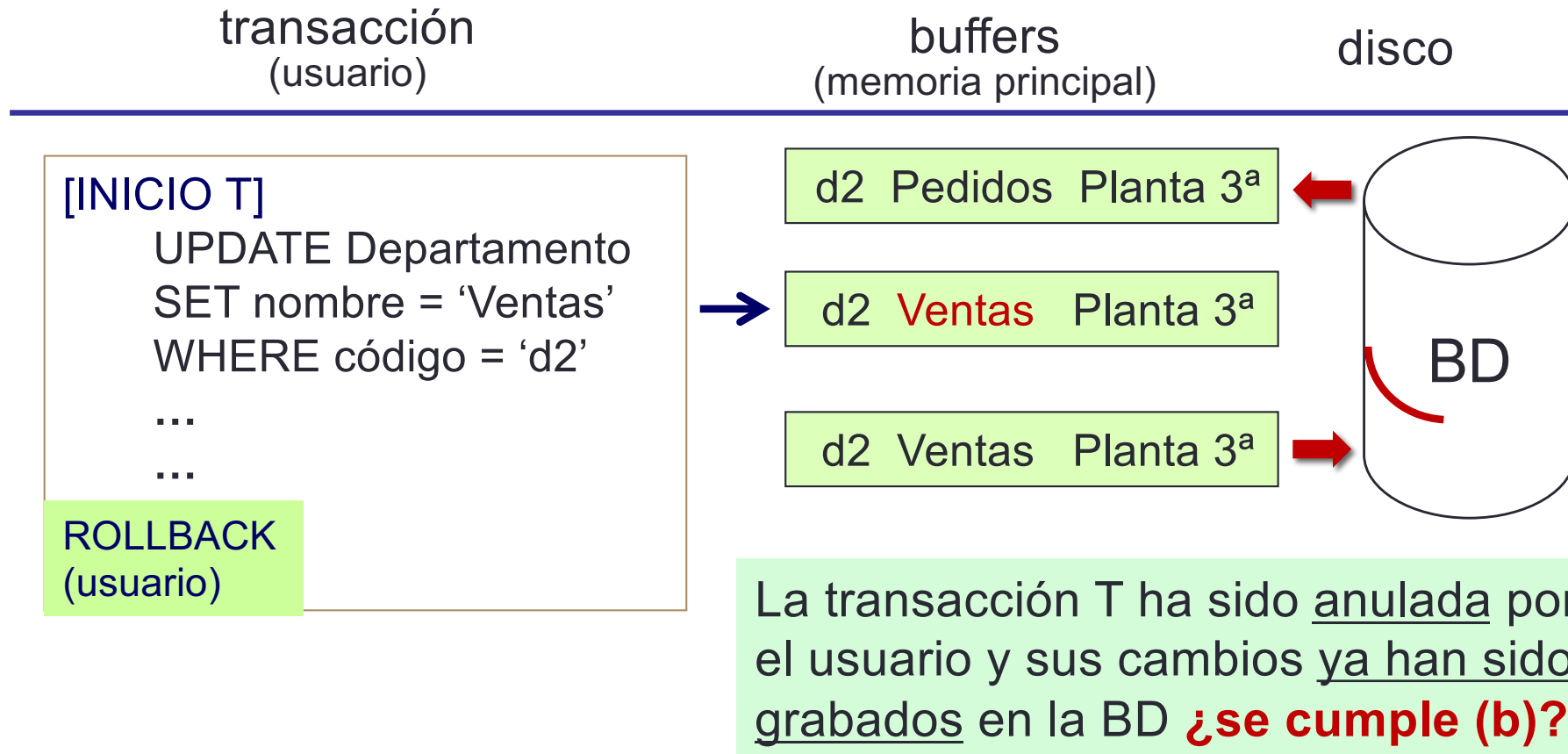
- Recuperación de transacciones.
- Control de la ejecución concurrente de transacciones
- Comprobación de restricciones de integridad

3. Integridad de los datos



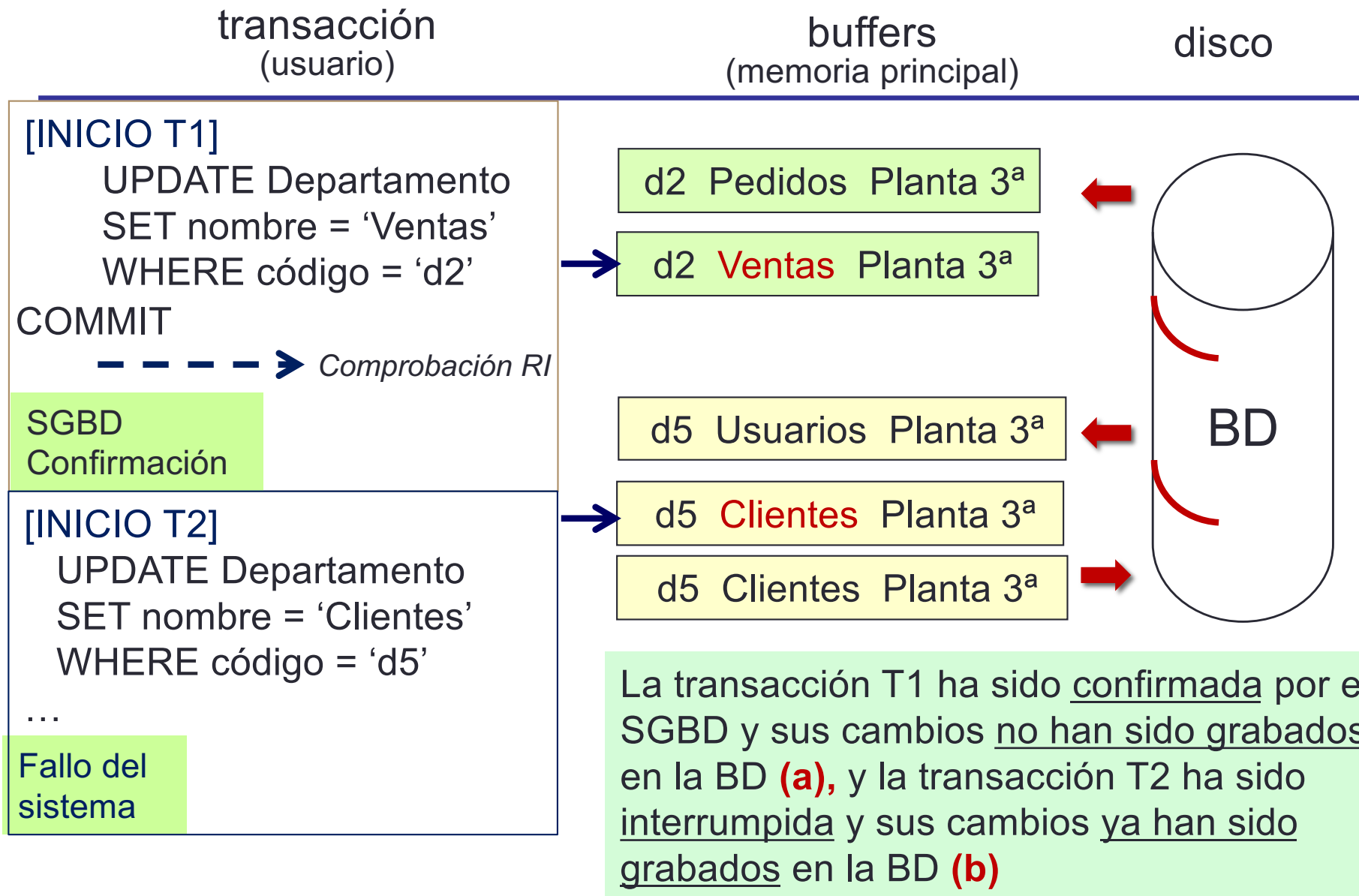
Debido a la política de transferencia de bloques seguida por el SGBD, los bloques actualizados por una transacción confirmada pueden no haber sido grabados a disco.

3. Integridad de los datos



Debido a la política de transferencia de bloques seguida por el SGBD, los bloques actualizados por una transacción pueden haber sido grabados a disco antes de que la transacción finalice.

3. Integridad de los datos



3. Integridad de los datos

- Una posible solución a esta situación que garantizaría los objetivos (a) y (b) del procesamiento de transacciones, podría consistir en: “transferir los bloques actualizados por una transacción a disco **sólo** cuando la transacción es confirmada por el SGBD”.
- Este comportamiento significa una política de transferencia de bloques a disco muy rígida: el SGBD debe economizar las operaciones de acceso a disco que son las más costosas y debe tener libertad para liberar espacio (bloques) de memoria principal cuando lo necesite.
- En los SGBD se contemplan distintas políticas de transferencia de bloques de memoria principal a disco: **estrategias de actualización de la BD.**

3. Integridad de los datos

Debido a la política de transferencia de bloques seguida por el SGBD: estrategia de actualización de la BD

- cuando una transacción es confirmada por el SGBD, algunas de sus actualizaciones pueden no haber sido grabadas en la BD.
- cuando una transacción es anulada (usuario, o SGBD) algunas de sus actualizaciones pueden haber sido ya grabadas en la BD.
- cuando una transacción es interrumpida por un fallo del sistema algunas de sus actualizaciones pueden haber sido ya grabadas en la BD.

Para procesar correctamente transacciones, el SGBD debe incorporar **técnicas de recuperación** que aseguren los objetivos **(a)** y **(b)**.

3. Integridad de los datos

Técnicas de recuperación: aseguran el correcto procesamiento de las transacciones (objetivos (a) o (b))

Herramientas para la recuperación:

- ✓ Fichero de diario
- ✓ Copias de seguridad de la BD

3. Integridad de los datos

Fichero de diario: fichero en el que se registran todas las operaciones ejecutadas por las transacciones. (ver Anexo III)

- ✓ El fichero de diario se almacena en disco.
- ✓ Para la actualización del diario se sigue la estrategia de transferencia de bloques entre los buffers de memoria principal y la memoria secundaria: existen buffers específicos destinados a contener bloques del fichero de diario.
- ✓ Para prevenir pérdidas del diario por fallos en el disco, periódicamente se hacen copias de seguridad del diario en cinta magnética (o en otros dispositivos).

3. Integridad de los datos

Fichero de diario: entradas (registros) del diario.

[inicio, T]

[escribir, T, X, valor_antes, valor_después]

[leer, T, X]

[confirmar, T]

[anular, T]

3. Integridad de los datos

Estrategia* básica de recuperación de transacciones confirmadas frente a fallos del sistema con pérdida de memoria principal.

Las actualizaciones de una transacción confirmada pueden no haber sido grabadas en la base de datos



Rehacer transacciones a partir del diario

Nota: esta estrategia básica asume algunas hipótesis de funcionamiento del SGBD: los bloques actualizados pueden ser transferidos a disco en cualquier momento y la confirmación de la transacción no implica transferir los bloques actualizados a disco.

3. Integridad de los datos

Estrategia básica de recuperación de transacciones confirmadas frente a fallos del sistema con pérdida de memoria principal.

Herramienta de recuperación: **fichero de diario**

Técnica de recuperación:

Transacciones T confirmadas en el fichero de diario:

([inicio, T] [confirmar, T])



deben rehacerse las
actualizaciones de T
(valor_después)

[escribir, T, X, valor_antes, valor_después]

3. Integridad de los datos

Estrategia básica de recuperación de transacciones falladas.
(transacciones anuladas o interrumpidas).

Las actualizaciones de una transacción anulada o interrumpida pueden haber sido ya grabadas en la base de datos.



Deshacer transacciones a
partir del diario

[escribir, T, X, valor_antes, valor_después]

3. Integridad de los datos

Estrategia básica de recuperación de transacciones falladas.
(transacciones anuladas o interrumpidas).

Herramienta de recuperación: **fichero de diario**

Técnica de recuperación:

Transacciones T **anuladas** en el
fichero de diario:

([inicio,T] [anular, T])



deben deshacerse las
actualizaciones de T
(valor_antes)

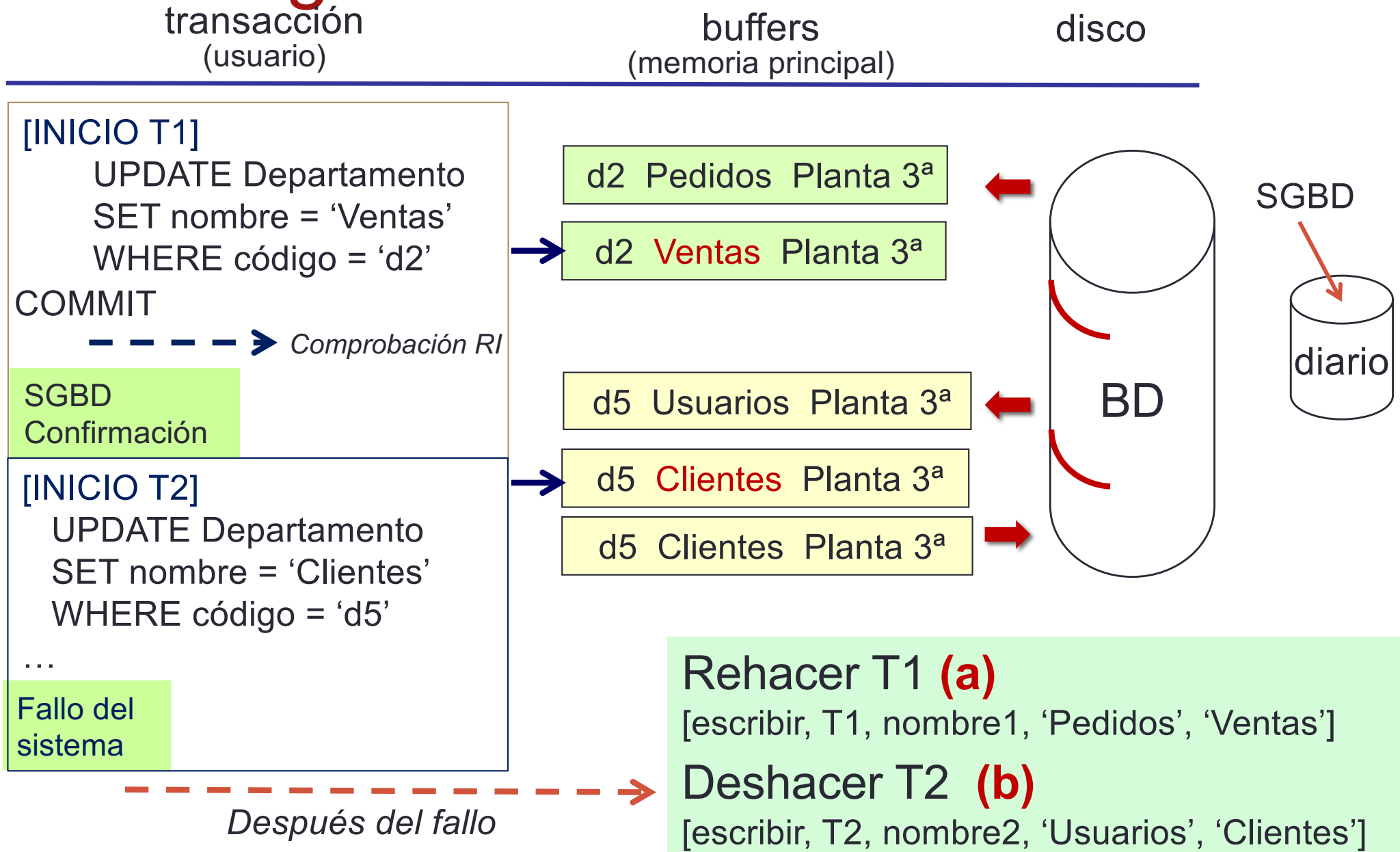
Transacciones T **interrumpidas** en el
fichero de diario:

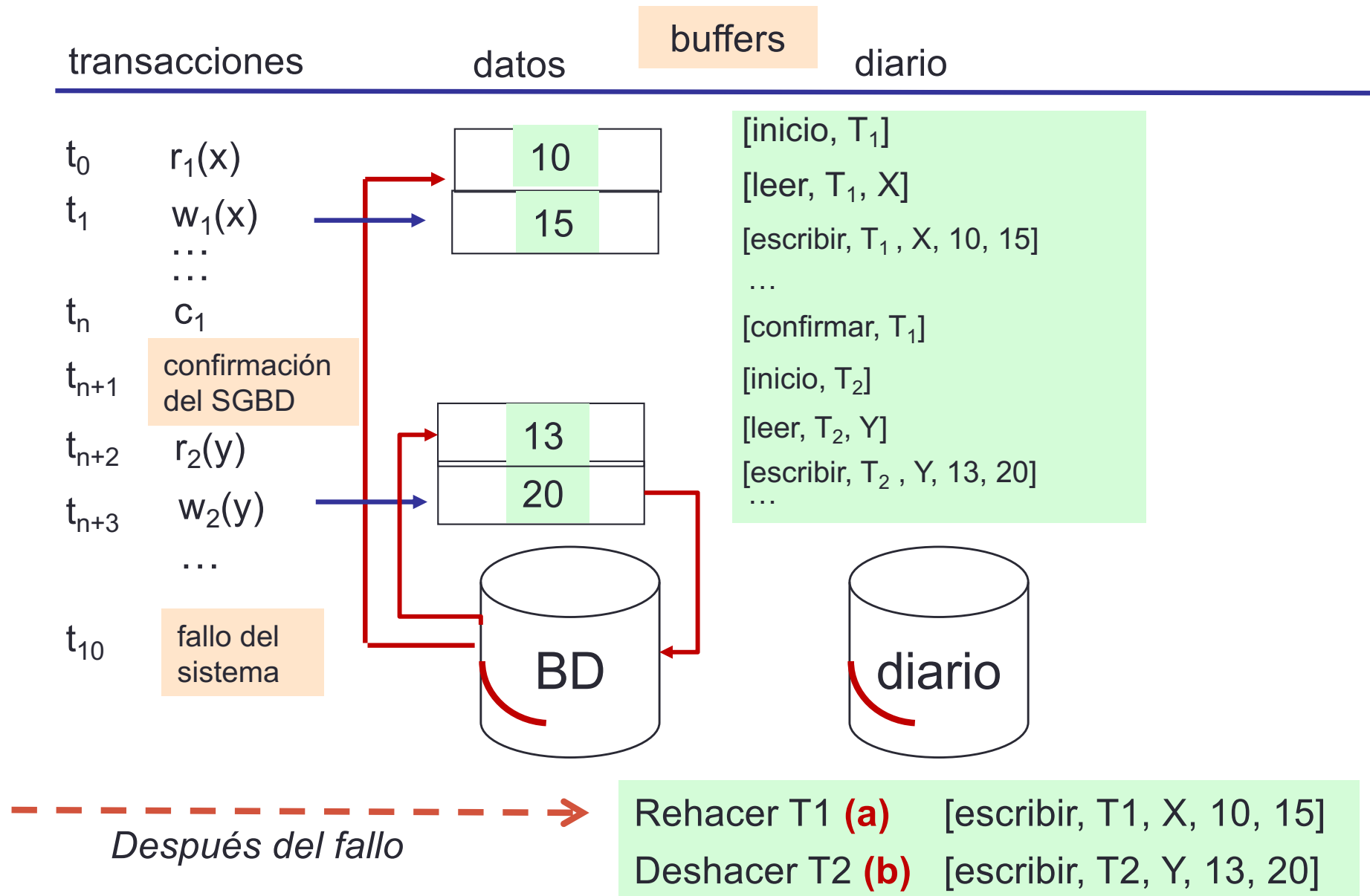
([inicio,T])



deben deshacerse las
actualizaciones de T
(valor_antes)

3. Integridad de los datos





¿Será posible la recuperación?: NO (se ha perdido el buffer de diario)
(Ver anexo III: gestión de un fichero de diario)

3. Integridad de los datos

Estrategia básica de recuperación de transacciones confirmadas frente a fallos del sistema de almacenamiento secundario.

Herramienta de recuperación:

- ✓ copias de seguridad de la base de datos
- ✓ fichero de diario

Técnica de recuperación:

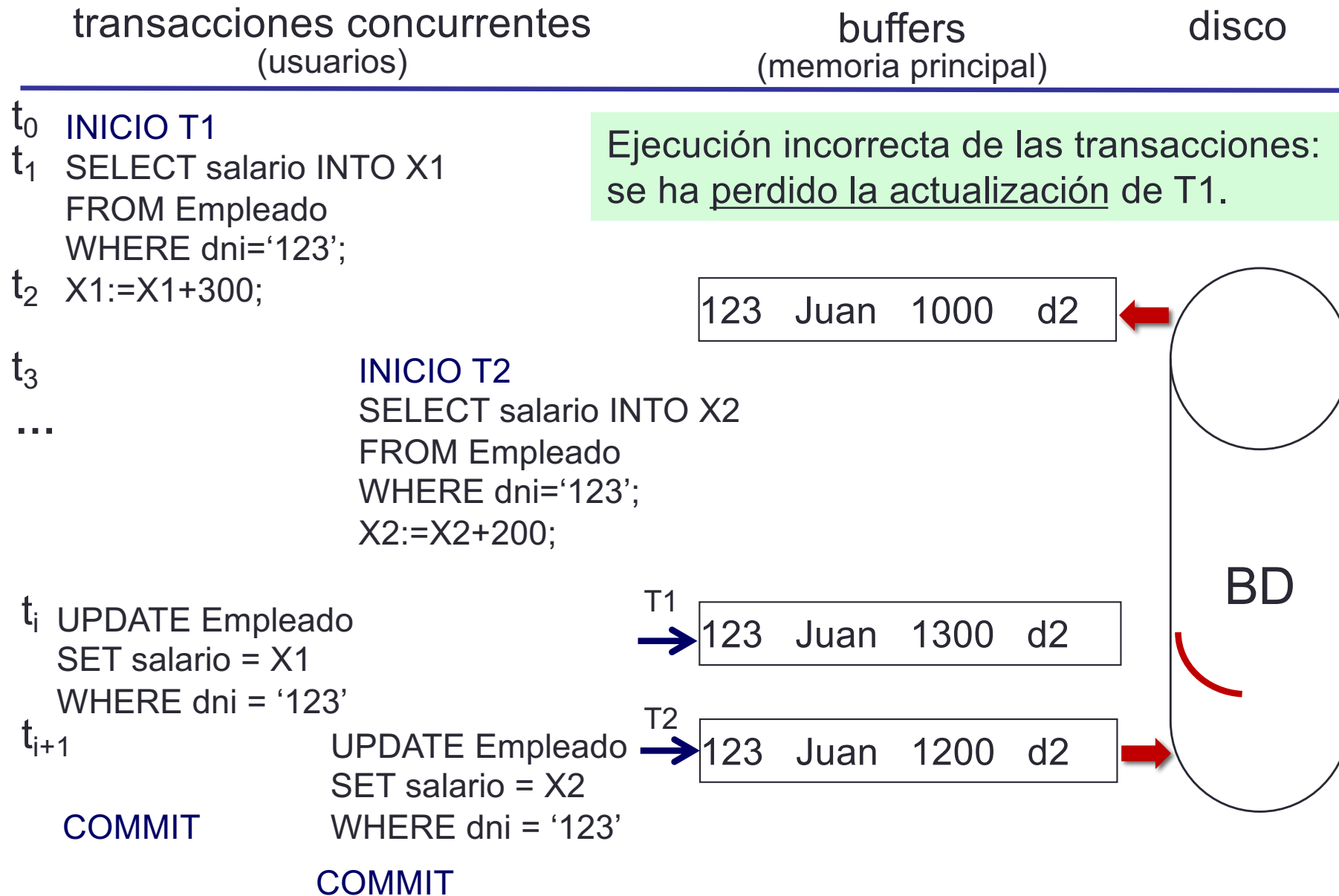
Cargar la base de datos a partir de la última copia de seguridad y a continuación rehacer todas las transacciones que aparecen confirmadas en el diario desde la fecha de la copia.

3. Integridad de los datos

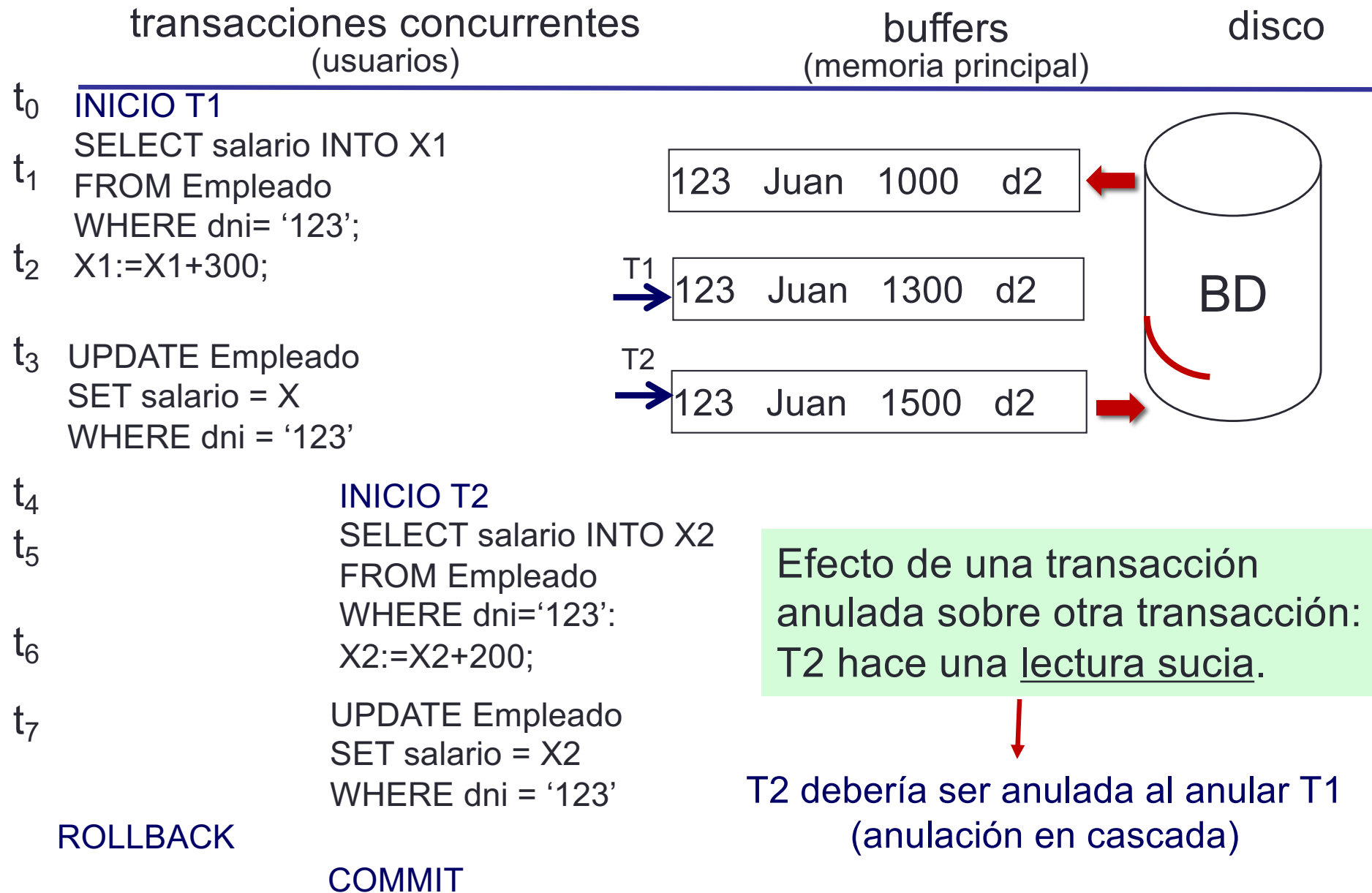
Aspectos relacionados con el mantenimiento de la integridad de los datos:

- Recuperación de transacciones.
- Control de la ejecución concurrente de transacciones
- Comprobación de restricciones de integridad

3. Integridad de los datos



3. Integridad de los datos



3. Integridad de los datos

Debido a:

- ✓ la política de actualización de la BD (en disco) seguida por el SGBD: transferencia de bloques
- ✓ la ejecución concurrente de transacciones

Para procesar transacciones correctamente el SGBD debe incorporar mecanismos de **control de la concurrencia** y de **recuperación** de transacciones que aseguren los objetivos **(a)** y **(b)** así como la ejecución concurrente (esperada).

3. Integridad de los datos

Ejecución concurrente de transacciones: anomalías

- Pérdida de actualizaciones:

$r_1(x), r_2(x), w_1(x), w_2(x), c_1, c_2$

- Lectura sucia:

$r_1(x), w_1(x), r_2(x), w_2(x), c_2, a_1$

- Lectura no repetible:

$r_1(x), r_2(x), w_2(x), c_2, r_1(x), c_1$

Causa de las anomalías: operaciones en conflicto

- 1) pertenecen a distintas transacciones,
- 2) acceden al mismo elemento de datos, y
- 3) al menos una de ellas es una operación de escritura

3. Integridad de los datos

Técnicas de control de
la concurrencia
(ver Anexo II)



PROTOCOLOS



Objetivo: Asegurar que el plan es serializable (por conflictos):
equivalente (en las operaciones
en conflicto) a un plan en serie.

Plan en serie: las operaciones de cada transacción se ejecutan consecutivamente en el plan. (no se intercalan las operaciones de distintas transacciones). Un plan en serie siempre es correcto.

3. Integridad de los datos

BD:

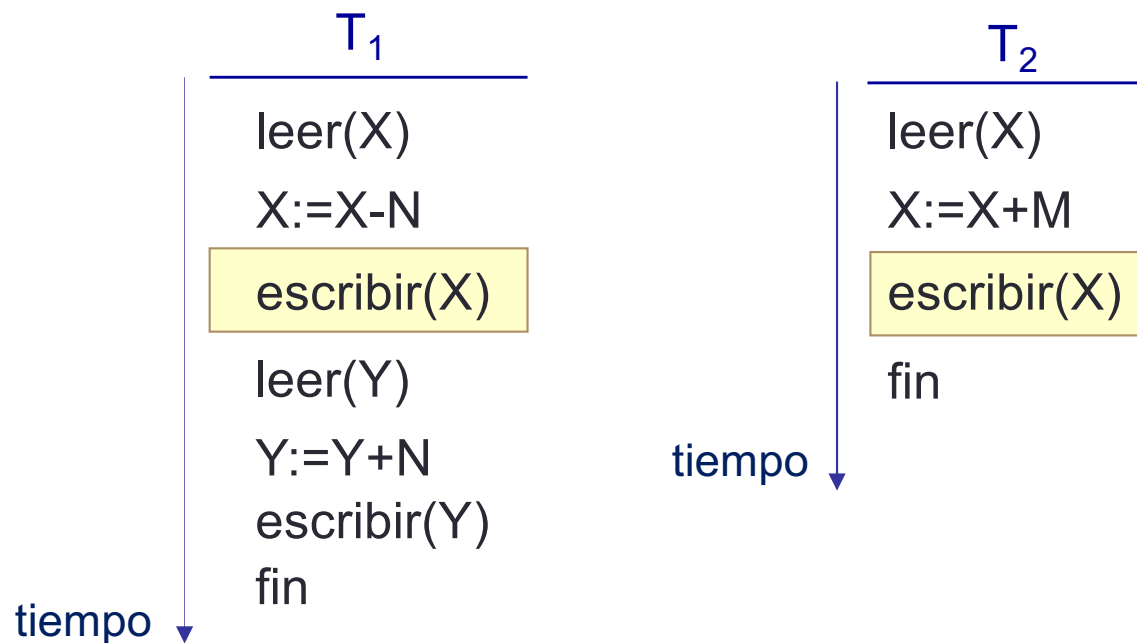
X=90

Y=90

Constantes:

M=2

N=3



3. Integridad de los datos

BD:

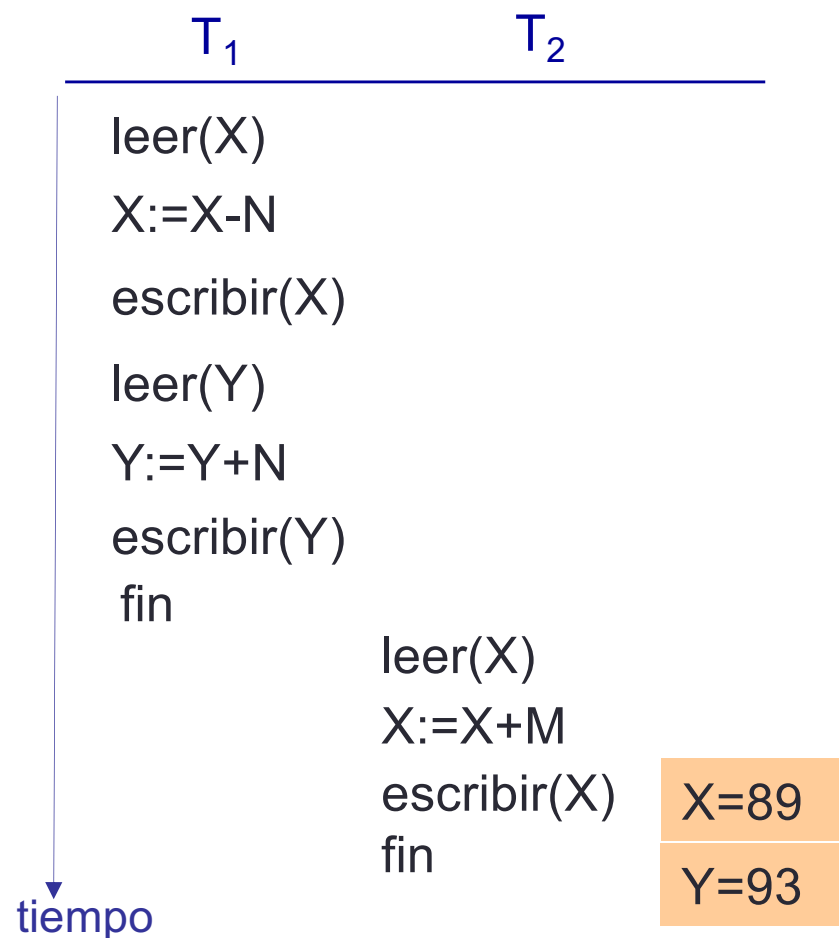
X=90

Y=90

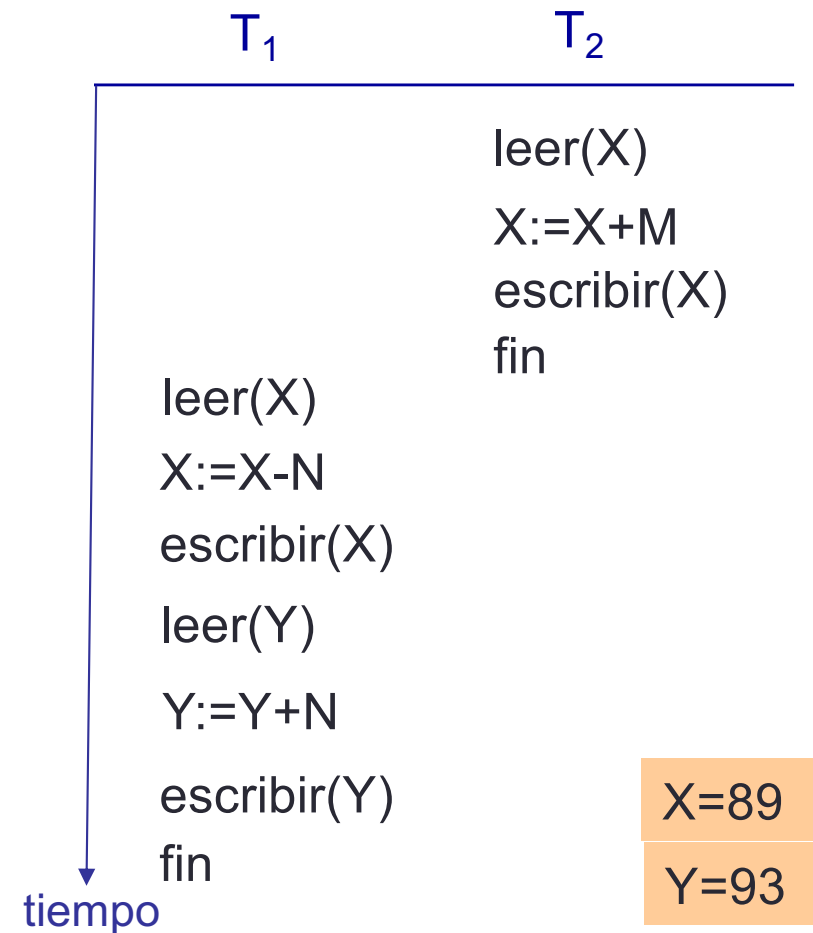
Constantes:

M=2

N=3



A: Plan en serie para T₁ y T₂



B: Plan en serie para T₁ y T₂

3. Integridad de los datos

BD:

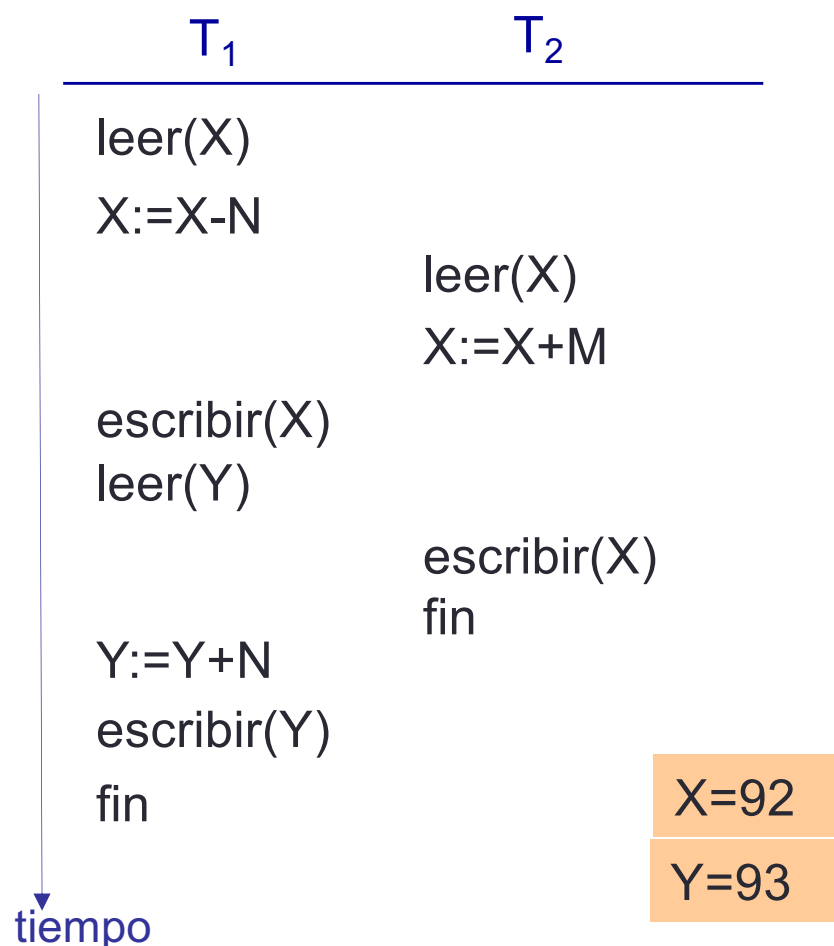
X=90

Y=90

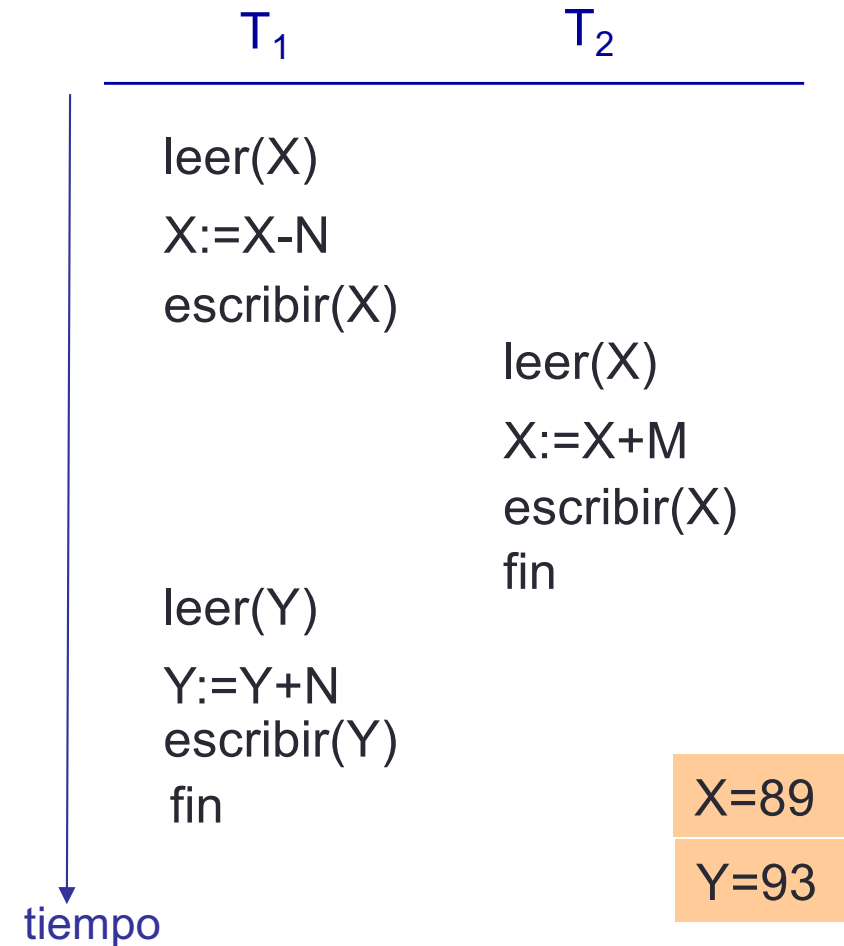
Constantes:

M=2

N=3

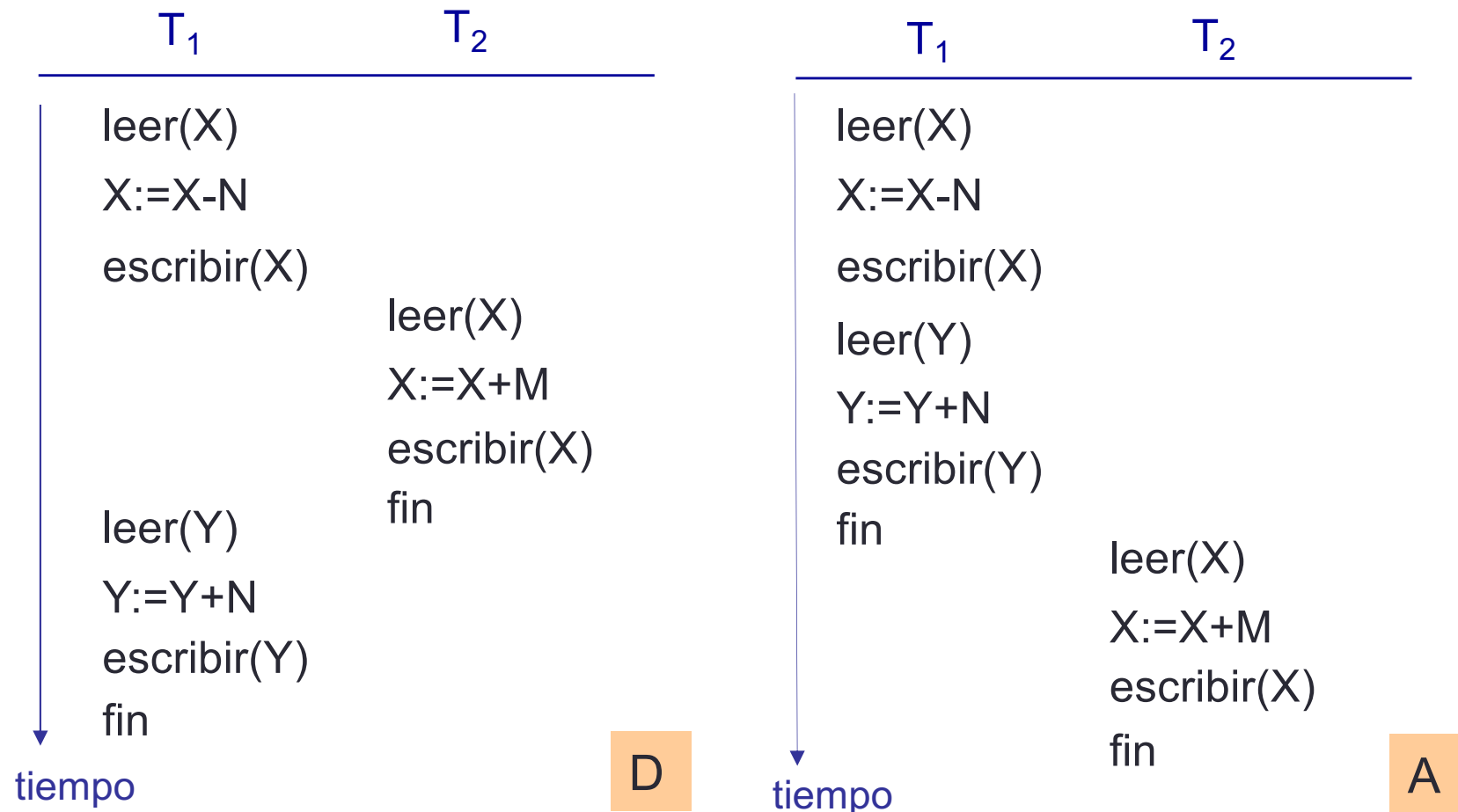


C: Plan concurrente para T_1 y T_2



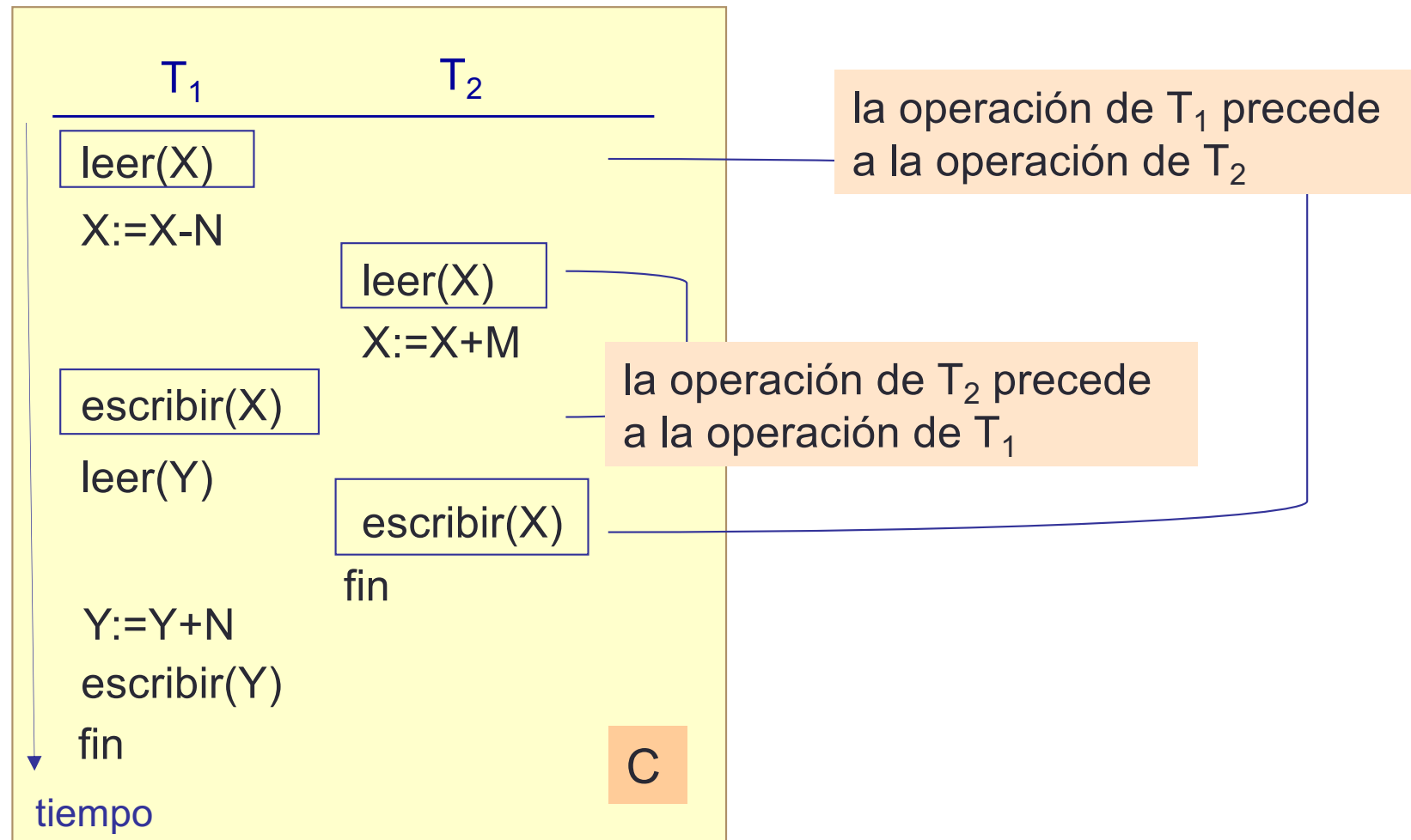
D: Plan concurrente para T_1 y T_2

3. Integridad de los datos



El plan D es equivalente por conflictos al plan en serie A por lo tanto es serializable por conflictos: es correcto

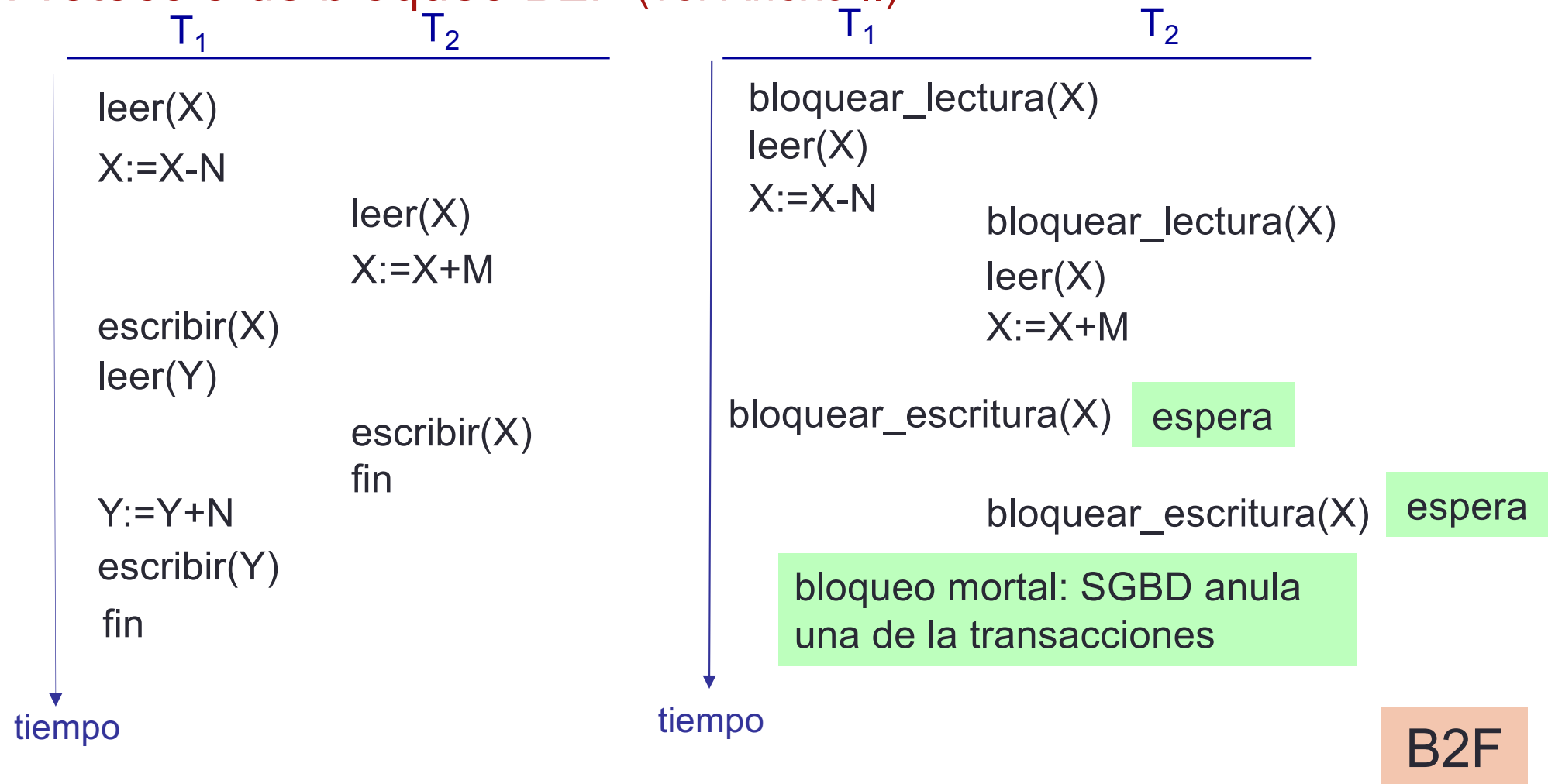
3. Integridad de los datos



El plan C no es equivalente por conflictos a ningún plan en serie: no es correcto

3. Integridad de los datos

Protocolo de bloqueo B2F (ver Anexo II)

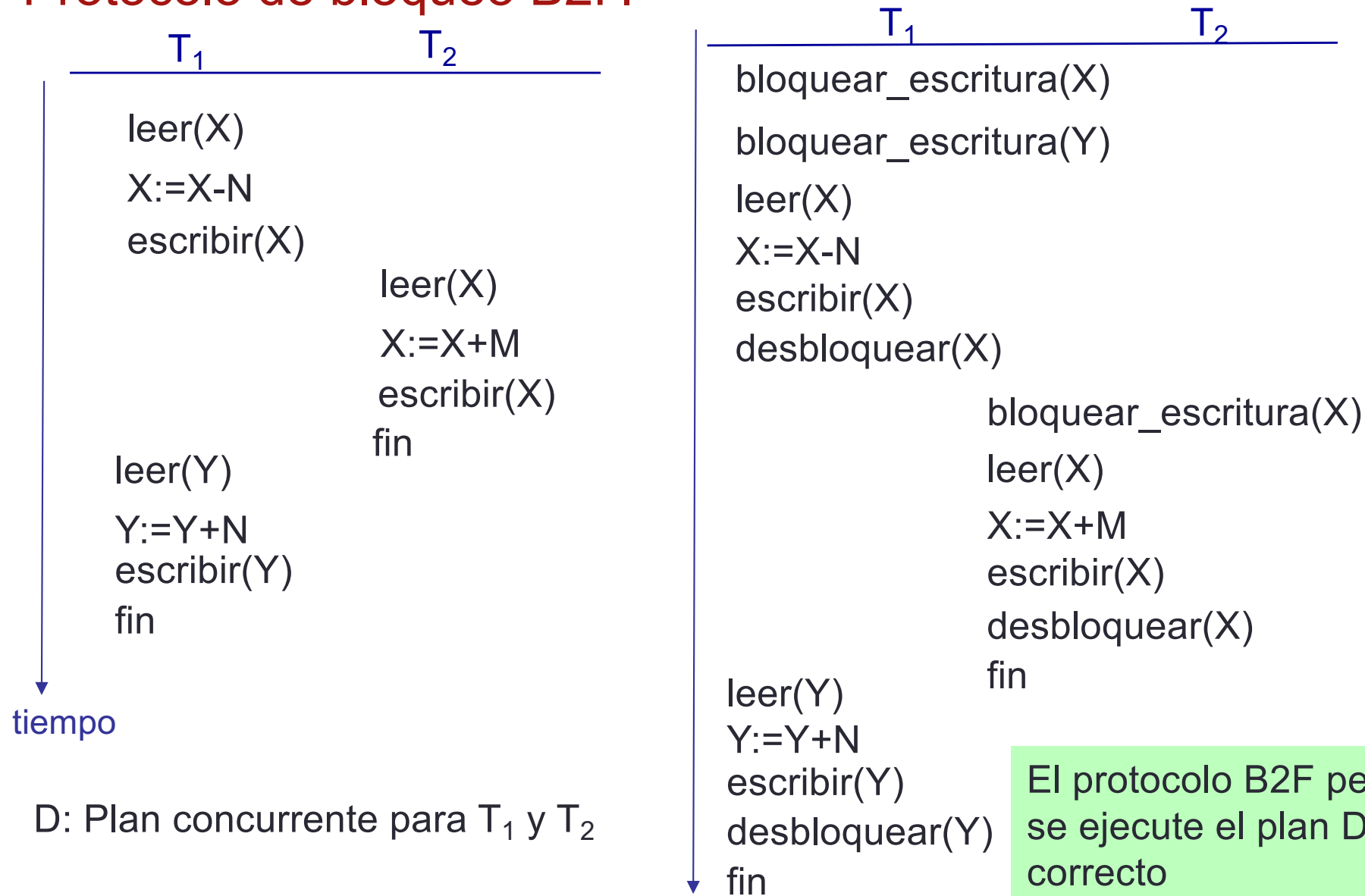


C: Plan concurrente para T₁ y T₂

El protocolo B2F no permite que se ejecute el plan C: evita la anomalía

3. Integridad de los datos

Protocolo de bloqueo B2F.



B2F

El protocolo B2F permite que se ejecute el plan D: plan correcto

3. Integridad de los datos

Protocolo de bloqueo B2F.

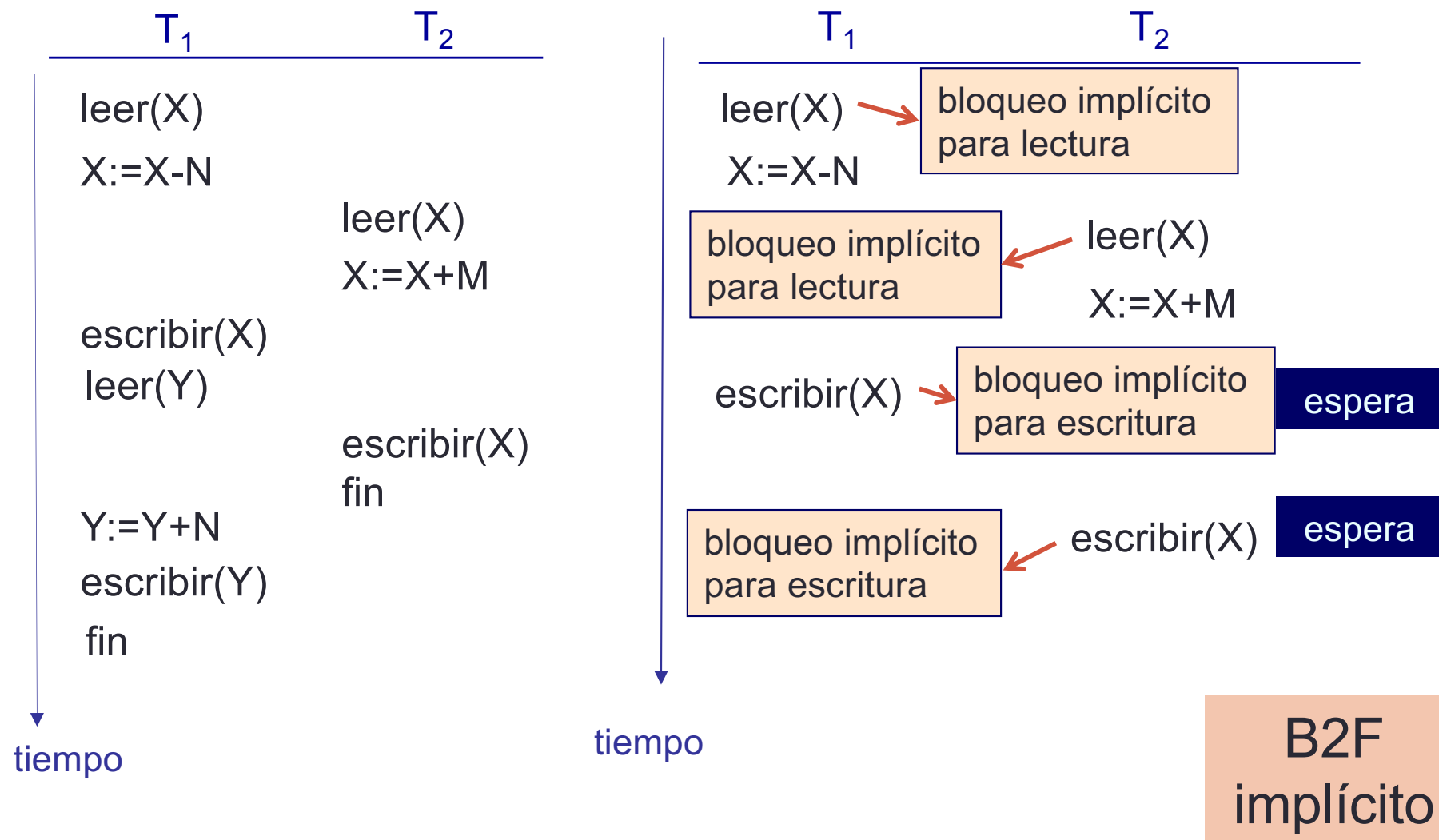
Bloqueo implícito.

El subsistema de control de la concurrencia del SGBD se encarga de generar implícitamente los bloqueos de lectura y de escritura y los desbloques sobre elementos de datos:

- ✓ leer(X) genera un bloqueo para lectura sobre X
- ✓ escribir(X) genera un bloqueo para escritura sobre X
- ✓ la finalización de la transacción (anulación o confirmación) genera el desbloqueo de todos los elementos de datos bloqueados por la transacción.

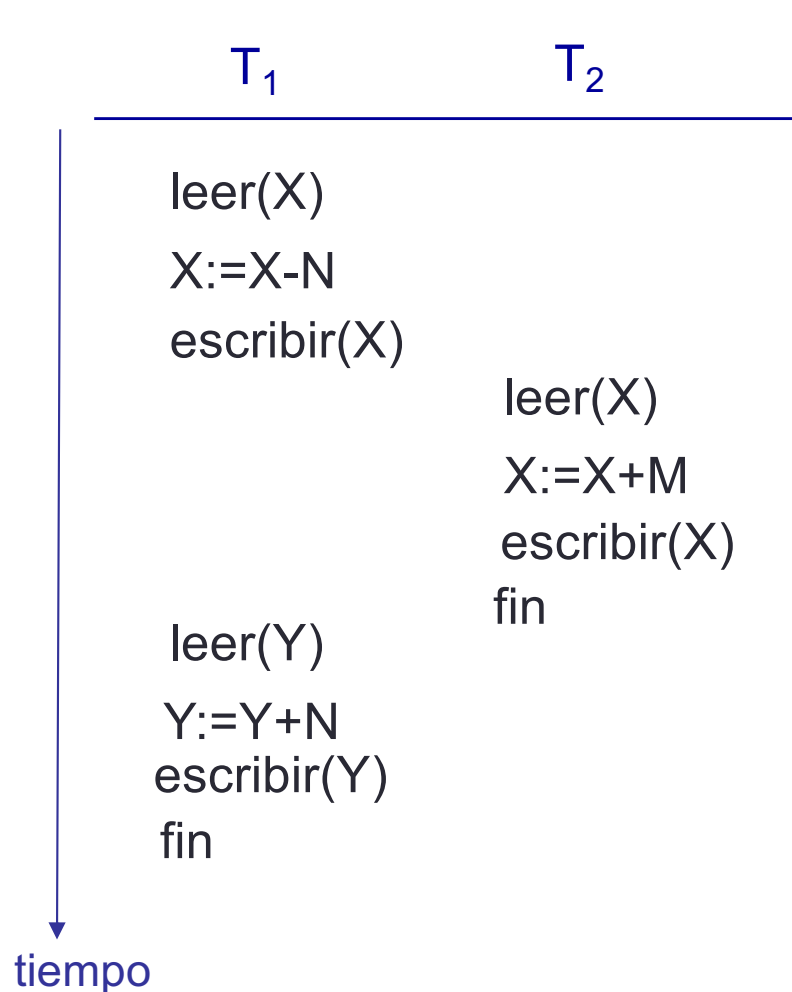
3. Integridad de los datos

Protocolo de bloqueo B2F.

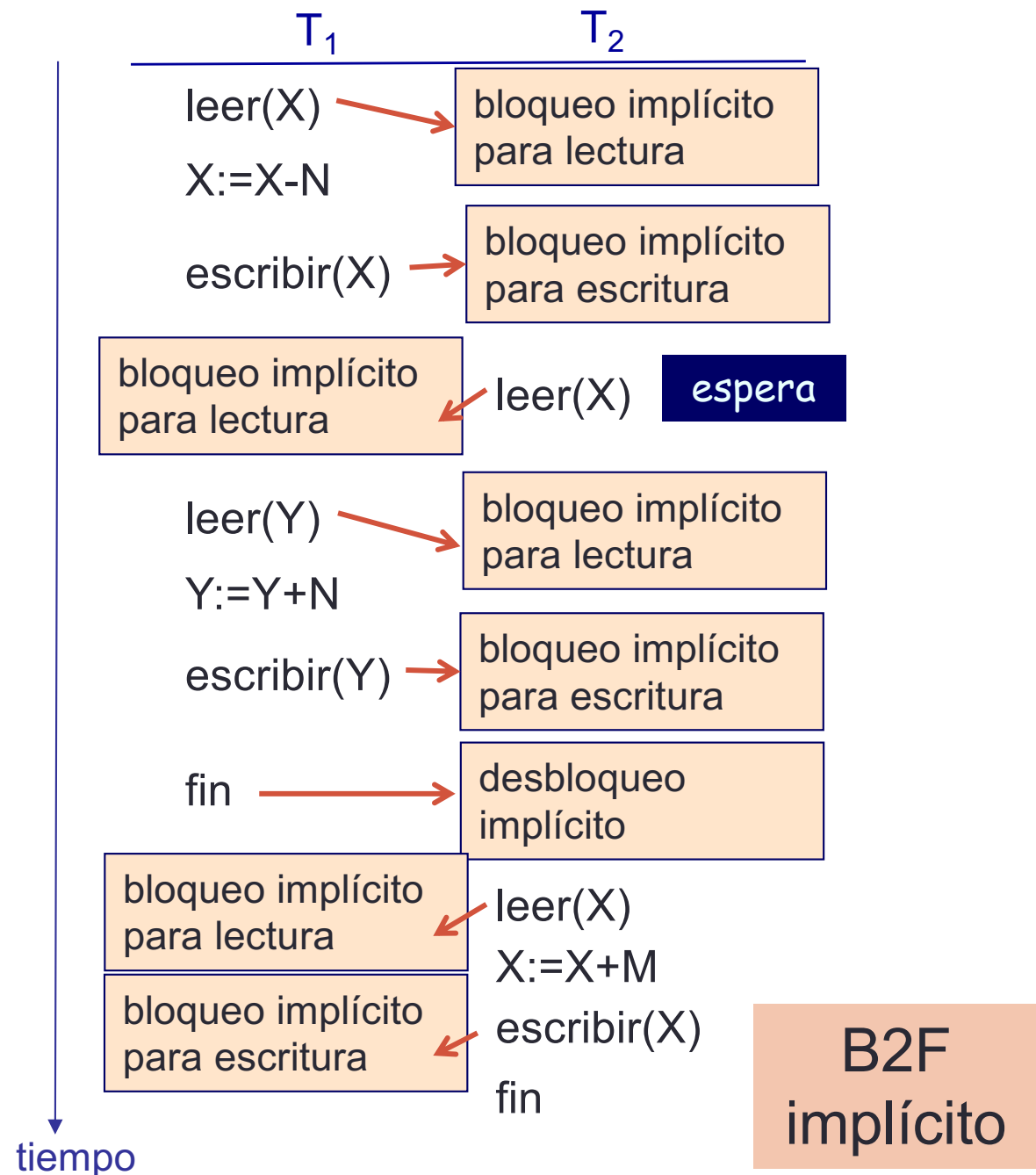


C: Plan concurrente para T_1 y T_2

Protocolo de bloqueo B2F.



D: Plan concurrente para T_1 y T_2



3. Integridad de los datos

Aspectos relacionados con el mantenimiento de la integridad de los datos:

- Recuperación de transacciones.
- Control de la ejecución concurrente de transacciones
- Comprobación de restricciones de integridad

3. Integridad de los datos

Comprobación de R.I en SQL:

Modo de comprobación de la integridad (se define para cada restricción del esquema):

- ✓ después de cada operación SQL (modo inmediato)
- ✓ al final de la transacción (modo diferido)

El modo de comprobación de una restricción puede ser fijo (restricción no diferible) o se puede cambiar dinámicamente durante la ejecución de una transacción (restricción diferible).

3. Integridad de los datos

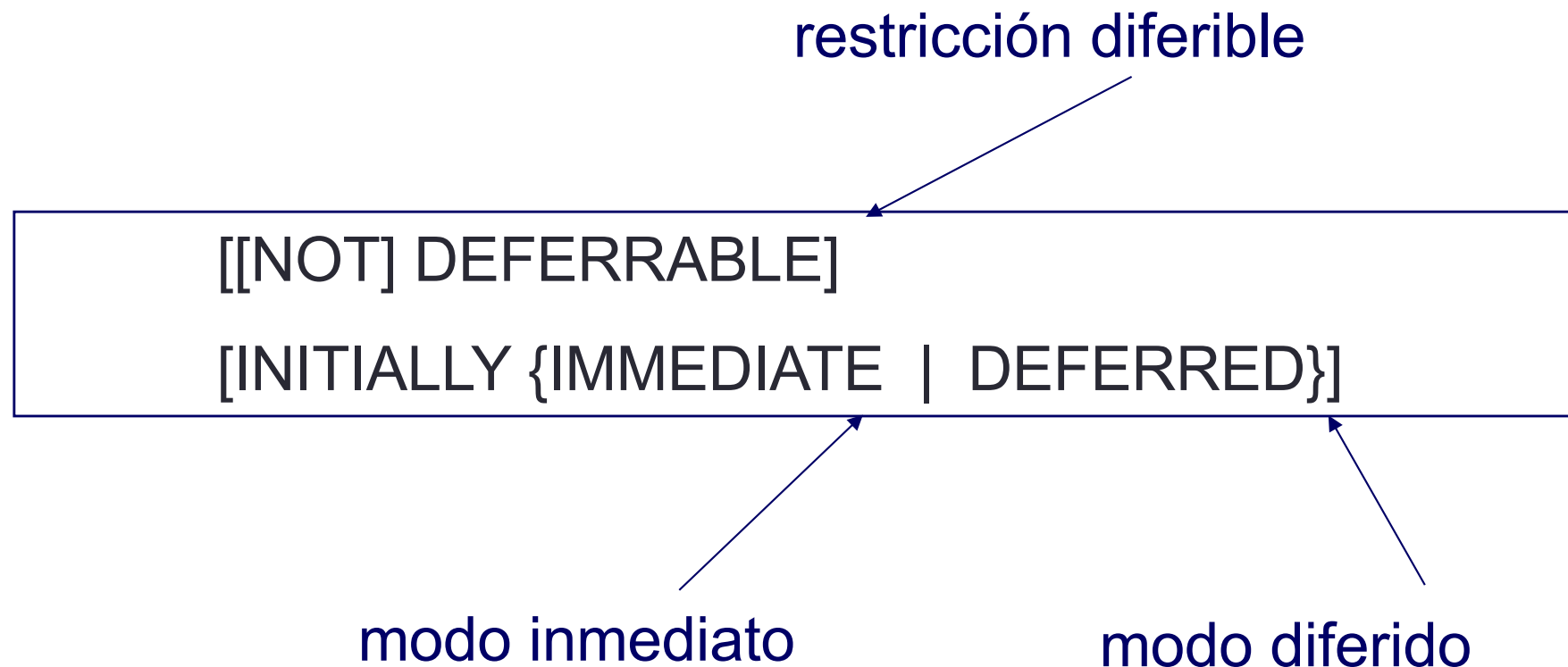
Comprobación de R.I en SQL:

Acción del SGBD frente a la violación de una RI:


- ✓ RI con modo inmediato: el SGBD anula la instrucción SQL que ha provocado la violación y la transacción continúa.
- ✓ RI con modo diferido: el SGBD anula la transacción.

3. Integridad de los datos

Comprobación de R.I en SQL:



Tecnología Relacional

1. Modelo relacional de datos.
2. Independencia de datos: arquitectura ANSI/SPARC.
3. Integridad de los datos.
4. Evolución de la tecnología relacional. 
5. Implementaciones

4. Evolución de la tecnología relacional

Limitaciones del modelo relacional

- ✓ atributos simples sobre dominios escalares.
 - ✓ identificación de las entidades por valor (contenido)
 - ✓ manipulación de las relaciones explícitamente
 - ✓ ausencia de primitivas para representar entidades complejas (agregación)
 - ✓ ausencia de primitivas para representar jerarquías de entidades (generalización)
 - ✓ lenguaje de manipulación (SQL3) computacionalmente incompleto
 - ✓ operadores predefinidos y genéricos (INSERT, DELETE, UPDATE)
-
- ✓ limitaciones en la definición de información implícita: vistas recursivas
-
- ✓ limitaciones en la definición de comportamiento activo: disparadores

1

BD objeto-
relacionales

2

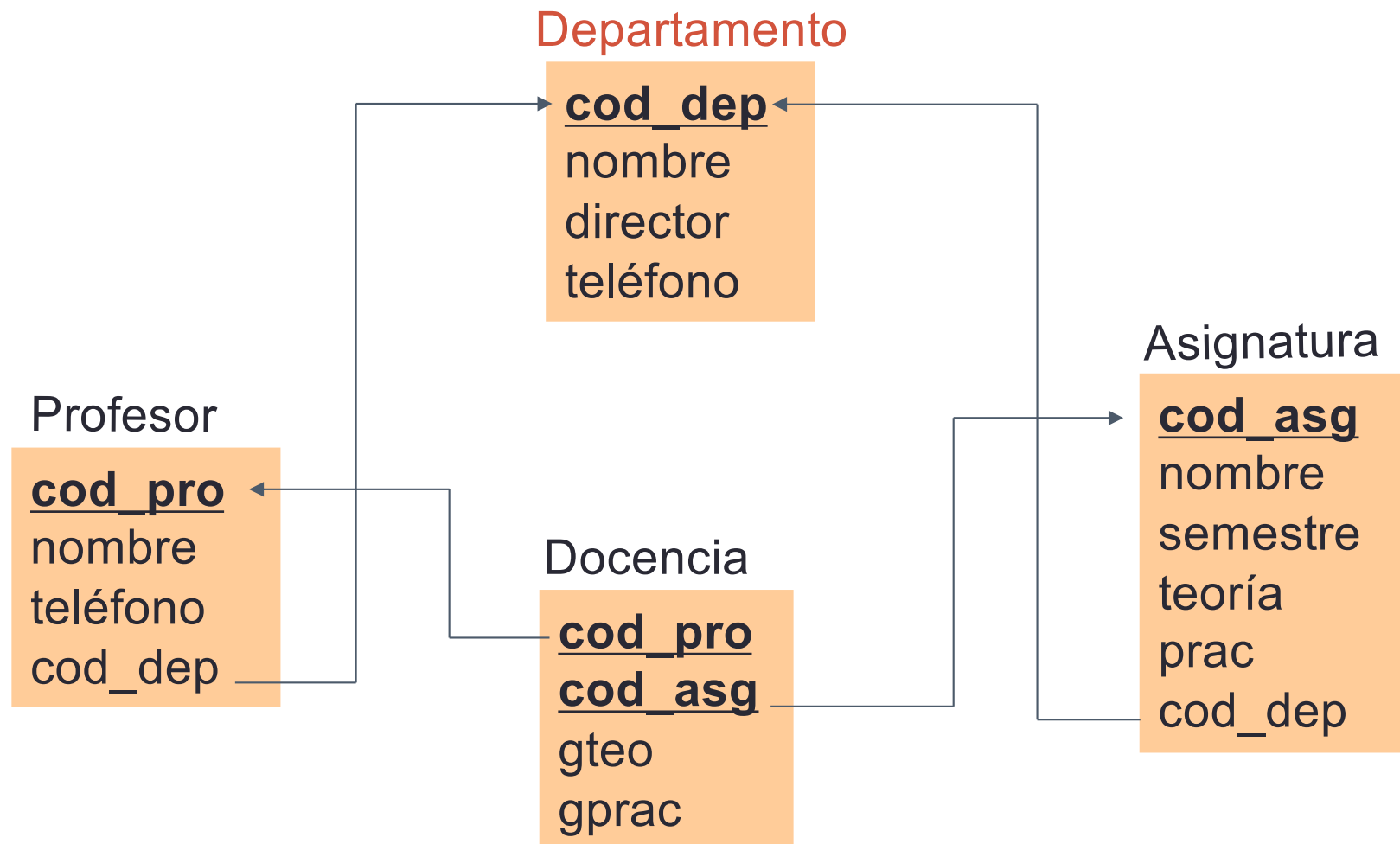
BD
deductivas

3

BD
activas

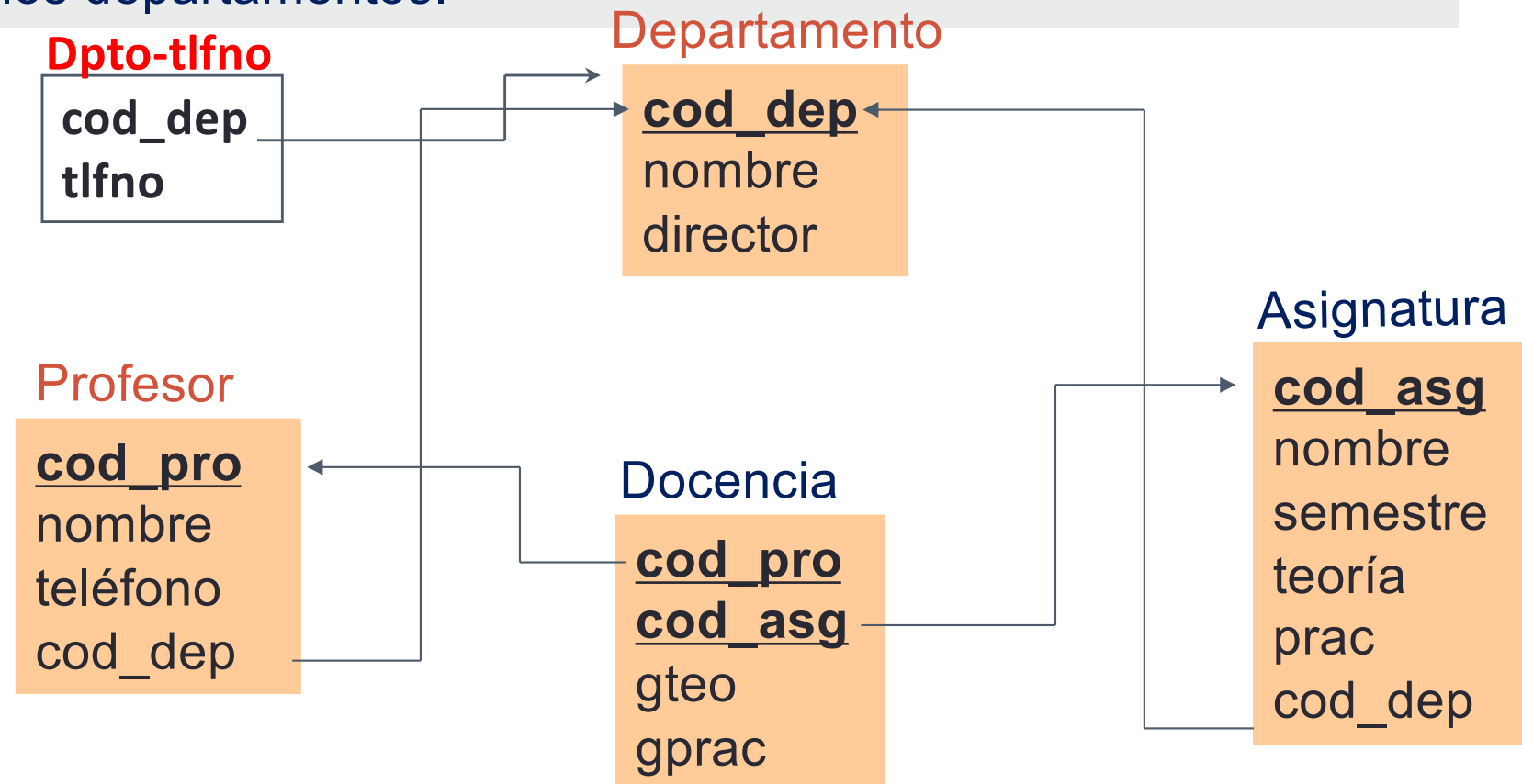
4. Evolución de la tecnología relacional

Ejemplo: base de datos “Docencia”



4. Evolución de la tecnología relacional

- 1 Se desea poder registrar varios teléfonos de contacto para los departamentos.



Limitaciones del modelo relacional (SQL92) en la definición de información compleja.



- dominios escalares
- ausencia de primitivas para definir objetos complejos

4. Evolución de la tecnología relacional

1

Consulta: “Obtener las asignaturas impartidas por el profesor de código ‘LBP’ ”:

```
SELECT Asignatura.nombre  
FROM Asignatura, Docencia  
WHERE Asignatura.cod_asg = Docencia.cod_asg  
and  
Docencia.cod_pro = 'LBP'
```

Limitaciones del
modelo relacional
en la consulta de la
base de datos



- manipulación explícita de las
relaciones (JOIN) para reconstruir
la información de un objeto.

4. Evolución de la tecnología relacional

2

Se desea registrar en la base de datos información sobre los prerrequisitos (incompatibilidades) entre las asignaturas.

La propiedad de ser prerrequisito entre dos asignaturas es una propiedad que se transmite transitivamente.

A partir de un información básica sobre la relación de prerrequisitos entre las asignaturas a un primer nivel (prerrequisitos directos), se puede derivar información sobre la relación de prerrequisitos a cualquier nivel (prerrequisitos indirectos), es decir a través de otras asignaturas.

“Si la asignatura X tiene como prerrequisito a la asignatura Y e Y tiene como prerrequisito a la asignatura Z, entonces X tiene como prerrequisito a Z”.

4. Evolución de la tecnología relacional

2

Se desea registrar en la base de datos información sobre los prerrequisitos (incompatibilidades) entre las asignaturas.

Definición declarativa de la propiedad “ser prerrequisito”:

Regla 1: $\text{Prerrequisito}(x,y) \leftarrow \text{Prer}(x,y)$

Regla 2: $\text{Prerrequisito}(x,y) \leftarrow \text{Prer}(x,z) \wedge \text{Prerrequisito}(z,y)$

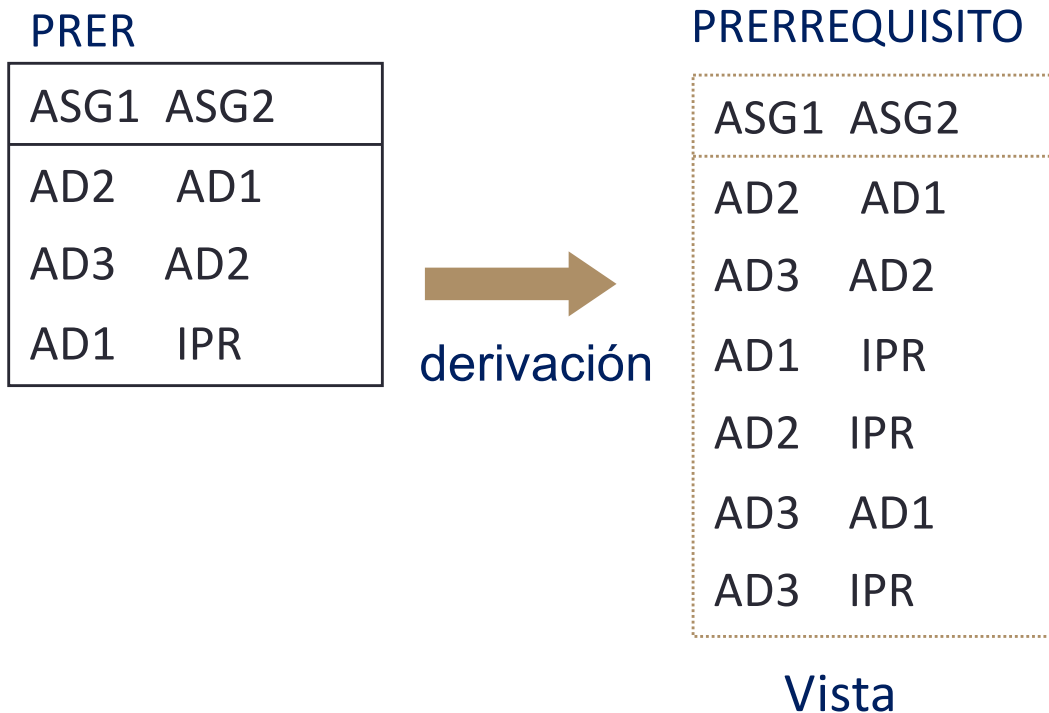
Prer: representa la propiedad “ser prerrequisito directo”

Prerrequisito: representa la propiedad “ser prerrequisito directa o indirectamente”

4. Evolución de la tecnología relacional

2

Se desea registrar en la base de datos información sobre los prerrequisitos (incompatibilidades) entre las asignaturas.

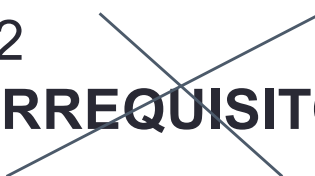


4. Evolución de la tecnología relacional

2

Relación derivada PRERREQUISITO  **VISTA**

```
SQL: CREATE VIEW PRERREQUISITO AS
      SELECT asg1, asg2
      FROM PRER
      UNION
      SELECT asg1, asg2
      FROM PRER, PRERREQUISITO
      .....
```



Limitaciones del modelo relacional (SQL92) en la definición de información implícita:



definición de vistas (no recursivas).

4. Evolución de la tecnología relacional

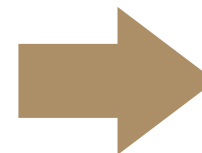
- 3 Se desea incluir un atributo en la tabla Departamento con el número total de profesores del departamento.

Departamento

cod_dep
nombre
director
teléfono
profesores

información derivada que
debe ser mantenida por el
SGBD

INSERT Profesor
DELETE Profesor
UPDATE (cod_dep) ON Profesor



SGBD

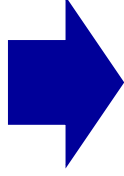
UPDATE (profesores)
ON Departamento

Limitaciones del modelo
relacional (SQL92) en la
definición de comportamiento
activo



- ausencia de disparadores
(triggers).

4. Evolución de la tecnología relacional

SQL3
(1999)  Modelo de datos objeto-relacional,
deductivo y activo.

Objeto-relacional: extensión del modelo con conceptos de la orientación a objetos.

Deductivo: extensión del modelo para definir información implícita (vistas recursivas)

Activo: extensión del modelo para definir comportamiento activo (disparadores).

4. Evolución de la tecnología relacional

Modelo objeto-relacional.

Objetivo de la orientación a objetos: mantener una correspondencia “directa” entre los objetos del mundo real y su representación en la base de datos.

Principios de la orientación a objetos:

- a) Identidad de objetos
- b) Abstracción de datos
- c) Herencia en jerarquía de objetos

4. Evolución de la tecnología relacional

a) Identidad de objetos:

Modelo Relacional
de Datos



estructura de datos \equiv relación

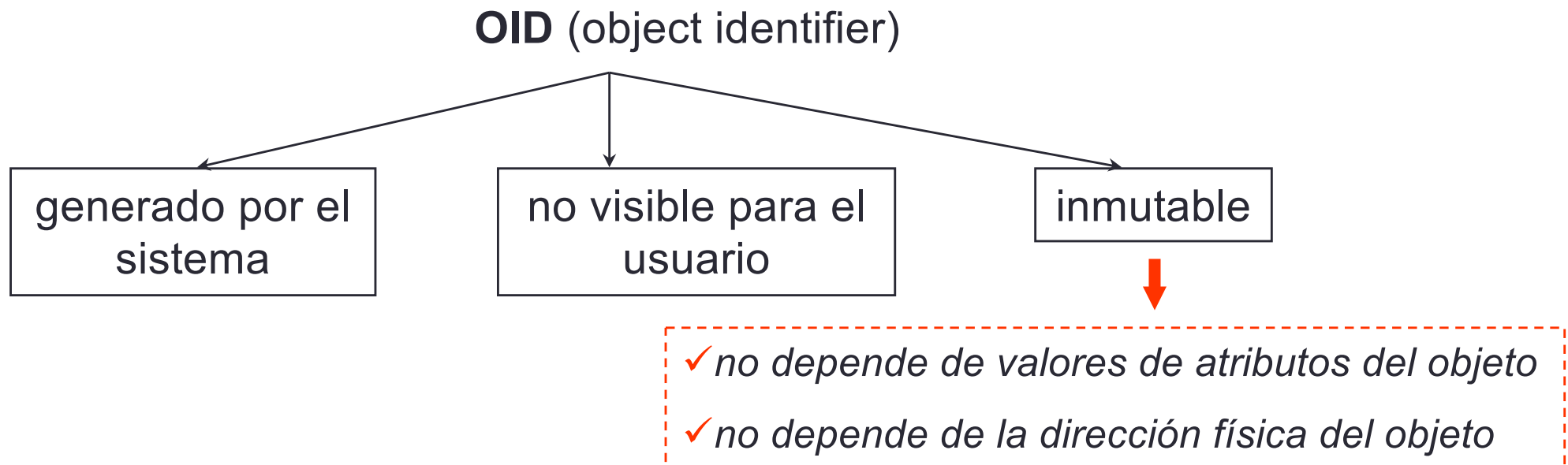
- los objetos del mundo real se representan como tuplas de relaciones
- las tuplas se identifican *por contenido*: por el valor de los atributos que constituyen la clave primaria de la relación



- si cambia el valor de la clave de una tupla entonces cambia la identidad del objeto representado
- las referencias al objeto desde otros objetos de la base de datos (claves ajenas) no son válidas y deben ser actualizadas

4. Evolución de la tecnología relacional

En el modelo orientado a objetos se proporciona una identidad única a cada objeto de la base de datos en el momento de su creación



El sistema utiliza internamente el OID para identificar unívocamente cada objeto en la creación y manejo de relaciones (referencias) entre objetos.

4. Evolución de la tecnología relacional

b) Abstracción de datos

Modelo Relacional
de Datos



estructura de datos \equiv relación

- los objetos del mundo real se representan como tuplas de relaciones
- los dominios de los atributos de las relaciones deben ser escalares



- la información del objeto está dispersa en varias relaciones
- el valor de un objeto (valor de sus atributos) debe ser reconstruido por el usuario manejando las referencias (claves ajenas) entre las relaciones de la base de datos durante la consulta de ésta: operaciones de JOIN

4. Evolución de la tecnología relacional

En el modelo orientado a objetos los objetos se definen con una estructura de cualquier complejidad por medio de los constructores de tipos.

Clase **Departamento**

cod_dep tira(4)	nombre tira(20)
teléfonos CONJUNTO (tira(10))	director Profesor
profesores LISTA (Profesor)	asignaturas LISTA (Asignatura)

Clase **Profesor**

cod_pro tira(4)	nombre tira(20)
director Departamento	teléfono tira(10)
departamento Departamento	
docencia LISTA (TUPLA (Asg: Asignatura, gteo: entero, gprac: entero))	

Clase **Asignatura**

cod_asg tira(4)	nombre tira(20)
semestre tira(3)	teoría entero prac entero
departamento Departamento	
docencia LISTA (TUPLA (Prof: Profesor, gteo: entero, gprac: entero))	

4. Evolución de la tecnología relacional

Modelo Relacional
de Datos



estructura de datos \equiv relación



- los objetos del mundo real se representan como tuplas de relaciones
- la estructura de los objetos es visible para el usuario
- las relaciones se manipulan con un conjunto de operadores genéricos: INSERT, DELETE, UPDATE, SELECT
- las restricciones de integridad se definen en el esquema de la base de datos, siendo responsabilidad del SGBD su comprobación



Si cambia la estructura de la BD deben cambiarse los programas de acceso

4. Evolución de la tecnología relacional

En el modelo orientado a objetos se separa la especificación de las propiedades del objeto de su implementación:

- ocultamiento de la estructura del objeto
- manipulación del objeto a través de operadores específicos definidos para el tipo de objeto (métodos)



**Encapsulamiento
del objeto**



Definición de un tipo de objeto por medio de los operadores (consultores, constructores) que se pueden aplicar sobre los objetos del tipo.

4. Evolución de la tecnología relacional

Definición de un tipo de objeto

Interfaz del objeto

Especificación de los operadores: nombre, tipo y argumentos

Implementación del objeto

Estructura interna del objeto: definida con los constructores de tipo

Implementación de los operadores (métodos): en función de la estructura del objeto

4. Evolución de la tecnología relacional

c) Herencia en jerarquía de objetos

Herencia: definición de un nuevo tipo de objeto (subtipo) a partir de otro tipo ya definido (supertipo)



- el subtipo hereda todas las propiedades (atributos y operadores) del supertipo y su implementación
- el subtipo puede tener atributos y operadores propios
- en el subtipo se puede cambiar la implementación de alguna de las propiedades heredadas (*overriding*)

4. Evolución de la tecnología relacional

El estándar SQL3: modelo objeto-relacional

Características:

- **nuevos tipos de datos:** BOOLEAN, LOB
- **constructores de tipos predefinidos:** escalares y estructurados
- **concepto de objeto** (definición de tipos de datos de usuario)
 - identidad de objetos (tablas de un tipo de objeto)
 - encapsulación (definición de métodos)
- **herencia** en jerarquía de tipos de objetos y en jerarquía de tablas

4. Evolución de la tecnología relacional

Tipos de datos

✓ **nuevos tipos de datos:** BOOLEAN, LOB

✓ **constructores de tipos predefinidos:**

(utilizables en la definición del tipo de los atributos de una tabla, o en la definición de tipos de usuario)

- escalares: REF (referencia a objeto)
- estructurados: ROW (tupla), ARRAY (vector)

(operadores (consultores y constructores) predefinidos)

✓ **tipos de datos definidos por el usuario:** CREATE TYPE

- escalares (DISTINCT TYPES)
- estructurados (**tipos de objetos**) (utilizables en cualquier punto donde se pueda usar otro tipo de datos y como tipo de las tuplas de una tabla)
(operadores definidos por el usuario (métodos))

4. Evolución de la tecnología relacional

Nuevos tipos de datos:

BOOLEAN: lógica trivaluada (TRUE, FALSE, UNKNOWN)

LOB: (Large Object):

BLOB (Binary Large Object): almacenamiento de sonido, imagen

CLOB (Character Large Object): almacenamiento de texto

- operadores y funciones predefinidas

- atributos de tipo LOB: consultados y actualizados de la forma usual

4. Evolución de la tecnología relacional

```
CREATE TABLE Libro  
(título          VARCHAR(100),  
id               INTEGER,  
resumen         CLOB(32K),  
texto           CLOB(20M),  
video           BLOB(2G))
```

```
SELECT position ('Capítulo 1' IN texto) INTO :variable  
  
FROM Libro  
  
WHERE título = 'Moby Dick'
```

4. Evolución de la tecnología relacional

Tipos de datos

Tipos de datos definidos por el usuario:

CREATE TYPE ...

(operadores definidos por el usuario (métodos))

- **escalares** (DISTINCT TYPES)
- **estructurados** (tipos de objetos)

4. Evolución de la tecnología relacional

Tipos de datos definidos por el usuario:

Tipos escalares (DISTINCT TYPES)

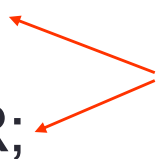
CREATE TYPE *nombre_tipo* AS *tipo_predefinido*

- están basados en un tipo de datos (escalar) predefinido (INTEGER, CHAR, ...)
- definición propia de operadores (métodos)
- fuertemente tipados
- no admiten subtipos
- heredan la representación interna del tipo de datos de origen

4. Evolución de la tecnología relacional

```
CREATE TYPE metros AS INTEGER;  
CREATE TYPE metros2 AS INTEGER;  
  
CREATE TABLE Habitación  
(ID            integer,  
altura         metros,  
superficie     metros2)
```

**DISTINCT
TYPES**



```
SELECT *  
FROM Habitación  
WHERE altura > superficie
```

¡error!

4. Evolución de la tecnología relacional

Tipos de datos definidos por el usuario:

Tipos estructurados: tipos de objetos

especificación

```
CREATE TYPE nombre_tipo AS (atributo1 tipo1, ..., atributon tipon)  
[INSTANTIABLE | NOT INSTANTIABLE]  
[FINAL | NOT FINAL]  
[METHOD nombre_método (param1, ..., paramn) RETURN tipo ] ...
```

implementación

```
CREATE INSTANCE METHOD nombre_método (param1, ..., paramn)  
FOR nombre_tipo BEGIN ... END
```

4. Evolución de la tecnología relacional

- ✓ **los métodos son funciones con un parámetro implícito** del tipo estructurado (tipo de objeto) al que está asociado el método.
- ✓ **especificación de métodos separada de su implementación.**
- ✓ **selección de atributos e invocación de métodos** con notación de punto.
- ✓ **constructores:** de instancia nula y de valor del tipo (NEW).

4. Evolución de la tecnología relacional

```
CREATE TYPE dirección_t AS  
(calle CHAR(30),  
ciudad CHAR(20),  
código INTEGER)
```

Tipos de objetos
definidos por el
usuario

```
CREATE TYPE datos_laborales AS  
(trienios INTEGER,  
prima INTEGER)
```

4. Evolución de la tecnología relacional

```
CREATE TYPE Empleado AS  
(DNI      INTEGER,  
nombre   VARCHAR(60),  
dirección dirección_t,  
méritos  datos_laborales)  
METHOD complemento () RETURNS REAL  
REF (DNI)
```

Especificación

Tipo de objeto
definido por el usuario

```
CREATE INSTANCE METHOD complemento() RETURNS REAL FOR Empleado  
BEGIN  
RETURN  
    SELF.méritos.trienios x 30 + SELF.méritos.prima  
END
```

Implementación

```
CREATE TABLE Empleados OF Empleado
```

← Tabla (relacional)
de objetos

4. Evolución de la tecnología relacional

constructor de instancia nula

constructor de valor del tipo

Dirección_t ()

NEW dirección_t (calle: Jesús 13, ciudad: Valencia, código: 12345)

UPDATE Empleados

SET dirección =

NEW dirección_t (calle: Jesús 13, ciudad: Valencia, código: 12345)

WHERE DNI=8765439

4. Evolución de la tecnología relacional

Tipos de datos

Constructores de tipos predefinidos:

(operadores (constructores y consultores) predefinidos)

Escalares:

REF: referencia a un objeto a través de su oid

(un valor de un tipo de objeto (tipo estructurado definido por el usuario) tiene oid cuando es una tupla de una tabla)

Definición: REF (*tipo_objeto*)

La representación interna de los valores del tipo REF asociado a un tipo de objeto puede elegirse:

- generado por el sistema: REF IS SYSTEM GENERATED
- derivado de valores de otros atributos: REF (*atributo*)
- generado por el usuario: REF USING <tipo de datos predefinido>

4. Evolución de la tecnología relacional

Constructores de tipos:

Estructurados:

ARRAY: tipo vector (unidimensional)

- definición : *tipo_base* ARRAY [*dimensión*]
- tipo base: cualquier tipo de datos excepto ARRAY
- selector de elemento por ordinal: *expresión* [i]
- constructor: ARRAY [valor₁, valor₂,]
- operadores: cardinalidad, comparación, concatenación,...
- facilidades para seleccionar elementos por contenido

4. Evolución de la tecnología relacional

```
CREATE TYPE Persona AS  
(DNI    INTEGER,  
nombre VARCHAR(60) )  
REF (DNI)
```

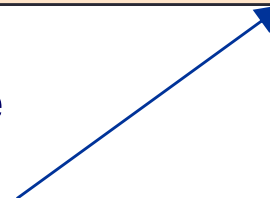


```
CREATE TABLE Personas OF Persona
```



```
CREATE TABLE Informes  
(id      INTEGER,  
título   VARCHAR(100),  
autor    REF (Persona),  
claves   VARCHAR(10) ARRAY [5 ],  
formato  ROW (páginas  INTEGER,  
              tipo_letra CHAR(10),  
              tamaño   INTEGER) )  
(autor WITH OPTIONS SCOPE Personas)
```

Los objetos de tipo *Persona* referidos desde el atributo *autor* en la tabla *Informes* deben localizarse en la tabla *Personas*



4. Evolución de la tecnología relacional

*selector de campo en tipo
ROW*

```
SELECT título, formato.páginas  
FROM Informes I  
WHERE I.autor-> nombre ='Juan García' AND claves[1]='Física'
```

*manipulación de tipo
REF*

*selector de elemento en tipo
ARRAY*

```
UPDATE Informes SET claves = ARRAY ['Física', 'Química']  
WHERE id = 123
```

*constructor de tipo
ARRAY*

4. Evolución de la tecnología relacional

Herencia en jerarquía de tipos

- los tipos estructurados definidos por el usuario (tipos de objeto) pueden clasificarse en una jerarquía de tipos
- un subtipo hereda los atributos y métodos de su supertipo
- un subtipo puede incluir nuevos atributos y métodos
- se admite sólo herencia simple
- se admite sobrecarga (*overload*) y reescritura (*overriding*) de métodos a través de tipos
- se respeta el principio de sustitutabilidad: en el lugar de un valor del supertipo se puede incluir un valor de cualquier subtipo

4. Evolución de la tecnología relacional

```
CREATE TYPE Empleado AS  
(DNI    INTEGER,  
nombre VARCHAR(60),  
dirección dirección,  
méritos datos_laborales)  
METHOD complemento () RETURNS REAL  
REF (DNI)
```

Especificación
del tipo



```
CREATE TYPE Vendedor UNDER Empleado  
(zona    CHAR(4))  
OVERRIDING METHOD complemento () RETURNS REAL
```

Especificación
del subtipo

4. Evolución de la tecnología relacional

```
CREATE INSTANCE METHOD complemento() RETURNS REAL FOR Empleado  
BEGIN  
RETURN  
    SELF.méritos.trienios x 30 + SELF.méritos.prima  
END
```

Implementación
del tipo

```
CREATE INSTANCE METHOD complemento() RETURNS REAL FOR Vendedor  
BEGIN  
RETURN  
    SELF.méritos.trienios x 40 + SELF.méritos.prima  
END
```

Implementación
del subtipo

4. Evolución de la tecnología relacional

Herencia en jerarquía de tablas:

- una subtabla hereda de la supertabla las columnas, restricciones, disparadores, ...
- las consultas sobre la supertabla tienen como alcance las tuplas de todas las subtablas en la jerarquía

```
CREATE TABLE Empleados OF Empleado  
PRIMARY KEY (DNI)
```

```
CREATE TABLE Vendedores OF Vendedor UNDER Empleados
```

4. Evolución de la tecnología relacional

SQL3
(1999)



Modelo de datos objeto-relacional,
deductivo y activo.

Objeto-relacional: extensión del modelo con conceptos de la orientación a objetos.

Deductivo: extensión del modelo para definir información implícita (vistas recursivas)

Activo: extensión del modelo para definir comportamiento activo (disparadores).

4. Evolución de la tecnología relacional

Se desea registrar en la base de datos información sobre los prerrequisitos (incompatibilidades) entre las asignaturas.

Definición declarativa de la propiedad “ser prerrequisito”:

Regla 1: $\text{Prerrequisito}(x,y) \leftarrow \text{Prer}(x,y)$

Regla 2: $\text{Prerrequisito}(x,y) \leftarrow \text{Prer}(x,z) \wedge \text{Prerrequisito}(z,y)$

Prer: representa la propiedad “ser prerrequisito directo”

Prerrequisito: representa la propiedad “ser prerrequisito directa o indirectamente”


4. Evolución de la tecnología relacional

El estándar SQL3: modelo deductivo

```
CREATE [ RECURSIVE ] VIEW nombre_vista  
    [(nombre_atr1 [{,nombre_atr2}...]) ]  
    AS sentencia_SELECT
```

```
CREATE RECURSIVE VIEW Prerrequisito (asg1, asg2)  
    AS (SELECT * FROM Prer  
        UNION  
        SELECT P.asg1, PP.asg2  
        FROM Prer P, Prerrequisito PP  
        WHERE P.asg2 = PP.asg1)
```

*SELECT recursiva
que define la vista
(relación derivada)*



4. Evolución de la tecnología relacional

SQL3
(1999)



Modelo de datos objeto-relacional,
deductivo y activo.

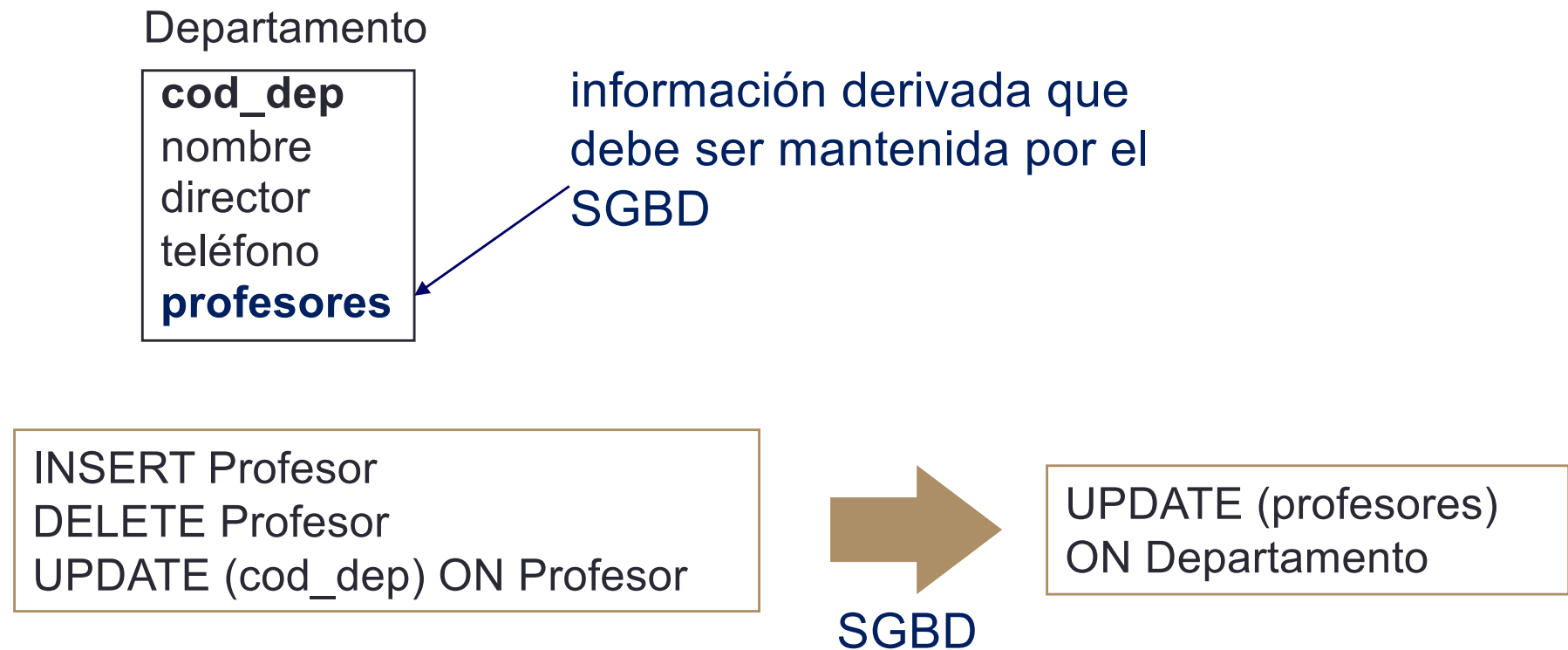
Objeto-relacional: extensión del modelo con conceptos de la orientación a objetos.

Deductivo: extensión del modelo para definir información implícita (vistas recursivas)

Activo: extensión del modelo para definir comportamiento activo (disparadores).

4. Evolución de la tecnología relacional

Limitaciones del modelo relacional (SQL92)
en la definición de comportamiento activo:



- limitaciones en la definición de comportamiento activo: disparadores.

4. Evolución de la tecnología relacional

Regla de comportamiento → evento - condición - acción

evento: especifica el suceso a cuya ocurrencia debe responder el sistema .

condición: especifica el contexto en el cual la regla cuyo evento se ha producido debe ser ejecutada.

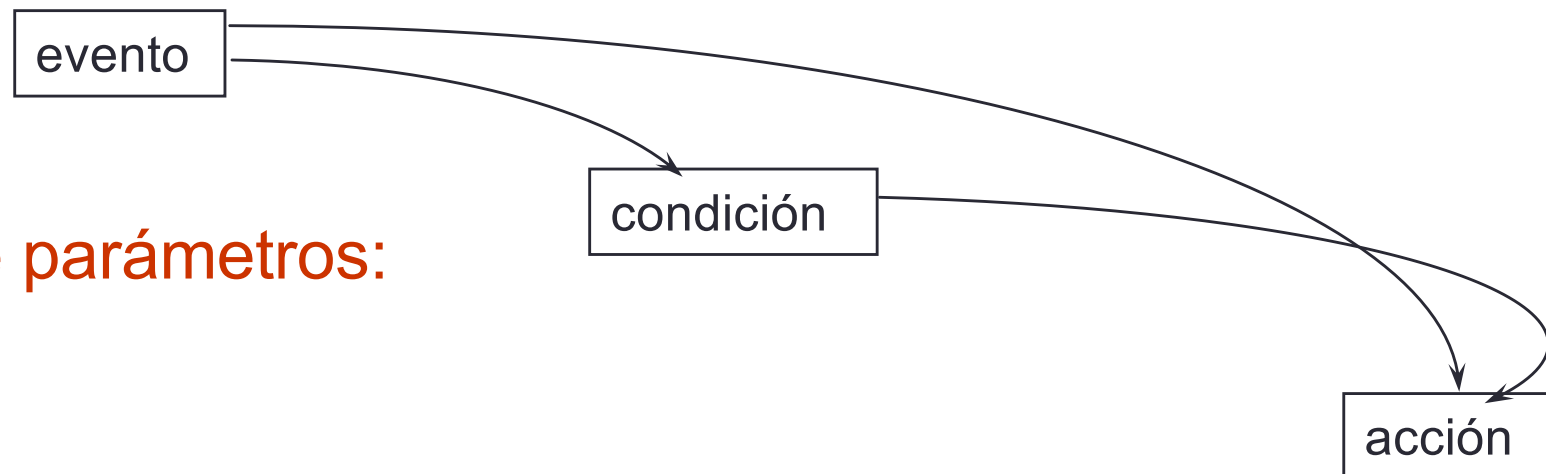
acción: especifica las acciones que deben ser ejecutadas por el sistema como respuesta a la ocurrencia del evento cuando la condición es cierta .

4. Evolución de la tecnología relacional

evento: especifica la causa que activa la regla.

condición: se evalúa cuando una regla activada es seleccionada por el SGBD para su ejecución.

acción: se ejecuta cuando una regla ha sido seleccionada por el SGBD y su condición se ha evaluado a cierto.



Paso de parámetros:

4. Evolución de la tecnología relacional

El estándar SQL3: modelo activo.

definición_regla :=

CREATE TRIGGER nombre_regla

{BEFORE | AFTER }

{INSERT | DELETE | UPDATE [OF lista_atributos]}

ON nombre_relación

[REFERENCING *alias1*, *alias2*, ...]

[FOR EACH {ROW | STATEMENT}]

[WHEN (*condición*)]

{sentencia_SQL | *bloque SQL/PSM* | *procedimiento_SQL*}

punto de ejecución

evento

parámetros

granularidad

acción

condición

4. Evolución de la tecnología relacional

Evento

Tipo de eventos: operaciones de DML (INSERT, DELETE, UPDATE)

Parametrización de eventos:

- parametrización implícita
- parámetros de *tupla*: OLD, NEW: tupla afectada por el evento
- parámetros de *tabla*: OLD_TABLE, NEW_TABLE: conjunto de tuplas afectadas por el evento
- nombres para los parámetros (*obligatorios si se va a hacer referencia a ellos*):

alias:= {OLD [ROW] [AS] nombre_referencia
 | NEW [ROW] [AS] nombre_referencia
 | OLD_TABLE [AS] nombre_referencia
 | NEW_TABLE [AS] nombre_referencia }

parámetros de tabla →

← *parámetros de tupla*

4. Evolución de la tecnología relacional

Condición

Tipo de condición: expresión lógica escrita en la sintaxis de SQL.
(no existen restricciones en la definición de condiciones)

Instanciación de la condición: si en la condición se hace referencia a parámetros del evento, ésta se instancia cuando la regla es activada.

Parametrización de la condición: la condición no puede incluir parámetros propios.

4. Evolución de la tecnología relacional

Acción

Sentencias SQL:

- sentencia simple: *operación_SQL*
- sentencia compuesta:

```
BEGIN [ATOMIC] {operación_SQL} {; operación_SQL} ... END
```

Bloque SQL/PSM:

```
BEGIN [ATOMIC]  
    ...  
    estructuras de programación en SQL/PSM  
    ...  
    operación-SQL | CALL procedimiento-SQL  
    ...  
END
```

4. Evolución de la tecnología relacional

Operaciones SQL permitidas:

- Reglas BEFORE:

- ⇒ sentencias de DML. Excepciones: INSERT, DELETE, UPDATE, sentencias de control de transacciones y sentencias de conexión a la BD.
- ⇒ sentencias de asignación sobre el parámetro de tupla NEW.
- ⇒ sentencia de control de errores (SIGNAL SQLSTATE*).

(una regla BEFORE no puede actualizar la base de datos)

- Reglas AFTER:

- ⇒ sentencias de DML. Excepciones: sentencias de control de transacciones sentencias de conexión a la BD.
- ⇒ sentencia de control de errores (SIGNAL SQLSTATE).

*SIGNAL SQLSTATE: aborta la ejecución del disparador y visualiza un mensaje de error.

4. Evolución de la tecnología relacional

Modelo de ejecución de reglas en SQL3

Granularidad*: la granularidad se elige para cada regla:

- orientada a la instancia (tupla): FOR EACH ROW
- orientada al conjunto (sentencia): FOR EACH STATEMENT

Punto de ejecución: el punto de ejecución es en cada instrucción SQL (acoplo evento-condición inmediato). Se puede elegir para cada regla:

- BEFORE: antes de la ejecución del evento
- AFTER: después de la ejecución del evento.

Tipo de procesamiento: recursivo.

Resolución de conflictos: se eligen las reglas en orden ascendente de su fecha de creación.

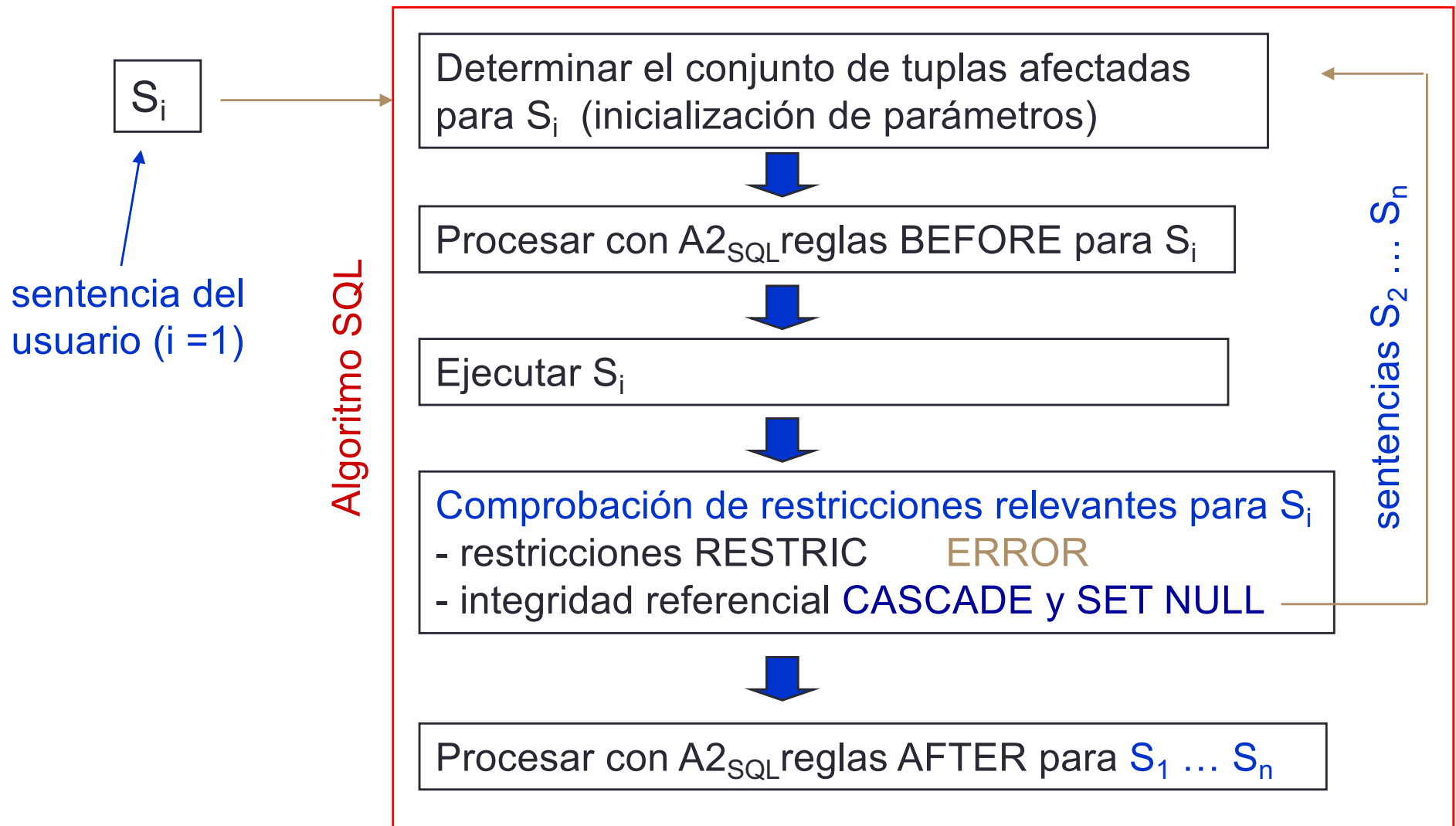
4. Evolución de la tecnología relacional

Granularidad	Punto de ejecución	Evento	Parámetros de tupla	Parámetros de tabla
ROW	BEFORE	INSERT	NEW	NO
		UPDATE	OLD, NEW	
		DELETE	OLD	
	AFTER	INSERT	NEW	NEW_TABLE
		UPDATE	OLD, NEW	OLD_TABLE, NEW_TABLE
		DELETE	OLD	OLD_TABLE
STATEMENT	BEFORE	INSERT	NO	NO
		UPDATE		
		DELETE		
	AFTER	INSERT	NO	NEW_TABLE
		UPDATE		OLD_TABLE, NEW_TABLE
		DELETE		OLD_TABLE

4. Evolución de la tecnología relacional

Modelo de ejecución de reglas en SQL3

(Algoritmo SQL para la ejecución de una sentencia S_1)



4. Evolución de la tecnología relacional

A2_{SQL}

mientras *existan reglas activadas*

1. seleccionar una regla activada R

2. si R es de tipo FOR EACH STATEMENT

- evaluar la condición de R

- si la condición es cierta

ejecutar secuencialmente las sentencias de la acción de R con el algoritmo **Algoritmo SQL**

sino

- para cada tupla afectada por el evento de R

- evaluar la condición de R

- si la condición es cierta

ejecutar secuencialmente las sentencias de la acción de R con el algoritmo **Algoritmo SQL**

fin_mientras

recursivo

La acción de una regla de tipo BEFORE no puede actualizar la base de datos.

4. Evolución de la tecnología relacional

Departamento


cod_dep nombre director teléfono profesores

información derivada que
debe ser mantenida por el
SGBD



```
CREATE TRIGGER r1
AFTER INSERT ON Profesor
FOR EACH ROW
BEGIN
    UPDATE Departamento
    SET profesores := profesores + 1;
END
```

Tecnología Relacional

1. Modelo relacional de datos.
2. Independencia de datos: arquitectura ANSI/SPARC.
3. Integridad de los datos.
4. Evolución de la tecnología relacional.
5. Implementaciones 

Anexo I:

Estructuras de almacenamiento

Fichero disperso

Direccionamiento calculado

Función de dispersión: f

Dominio del campo de dispersión K : D

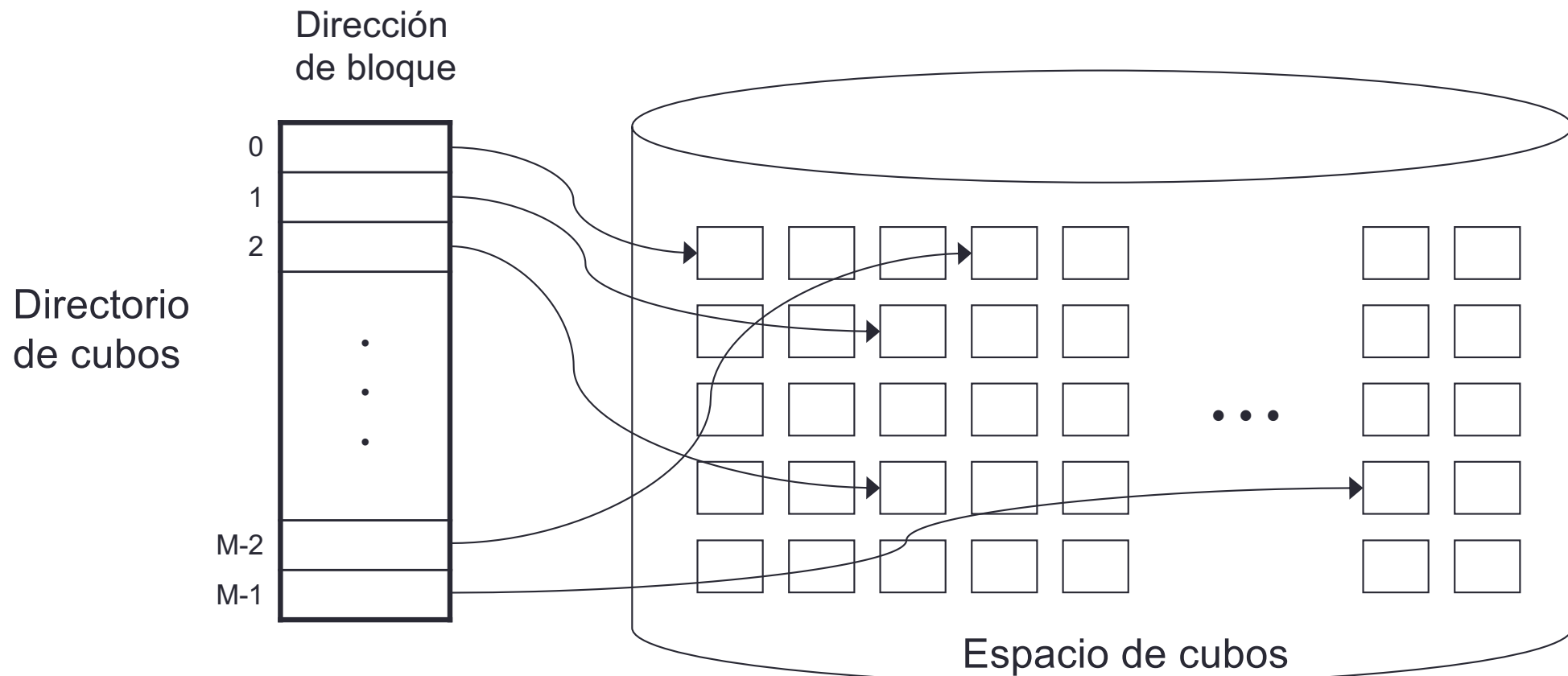
Espacio de direcciones: $0 \dots N-1$

$$f : D \rightarrow 0 \dots N-1$$

- Espacio de direcciones: cubos
 - El espacio en disco (bloques) se divide en cubos.
 - Un cubo se compone de uno o varios bloques de disco contiguos
- La función de dispersión aplicada al campo de dispersión del registro devuelve el número de cubo del fichero donde se almacena.
- Un tabla en la cabecera del fichero convierte el número de cubo en una dirección de bloque de disco.

Fichero disperso

Direccionamiento calculado



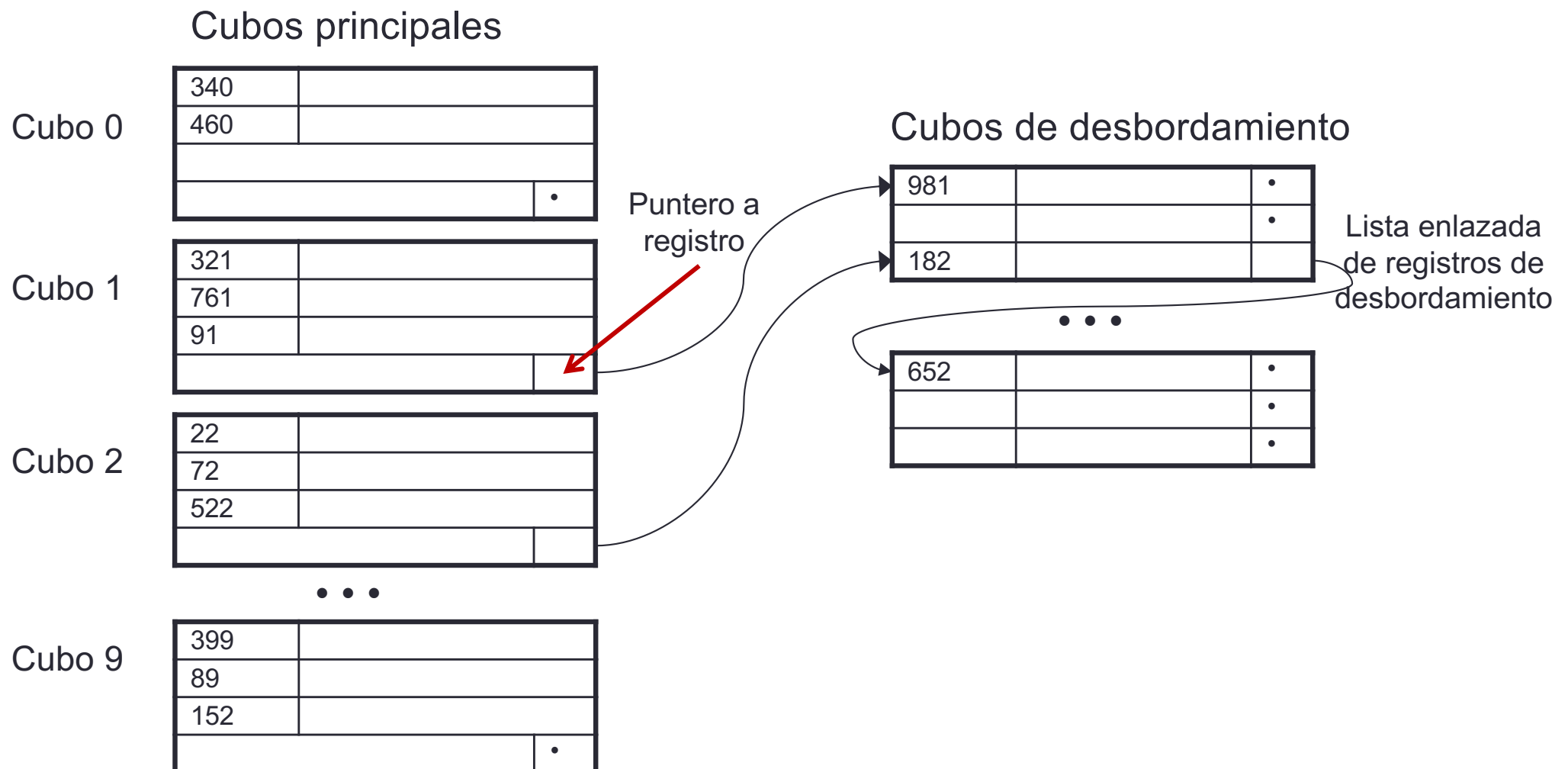
Fichero disperso

Direccionamiento calculado

- **Colisión:** la función de direccionamiento aplicada al campo de direccionamiento de un nuevo registro devuelve un número de cubo que ya está lleno
- Son menos frecuentes que en el direccionamiento interno, en un cubo pueden caber muchos registros
- Solución: variante del encadenamiento
 - Se añade a los registros un nuevo campo: puntero a registro
 - Se mantiene en cada cubo un puntero a una lista enlazada de registros de desbordamiento
 - Los registros de desbordamiento se almacenan en cubos de desbordamiento
 - Los punteros de la lista enlazada deben ser punteros a registros (dirección de bloque + posición relativa del registro en el bloque)

Fichero disperso

Direccionamiento calculado



Índices en árbol B

- Se construyen como árboles de búsqueda: árbol B y árbol B⁺
- Los árboles B y B⁺ son árboles n-arios, de búsqueda, equilibrados, que garantizan una ocupación eficiente del espacio en los nodos.

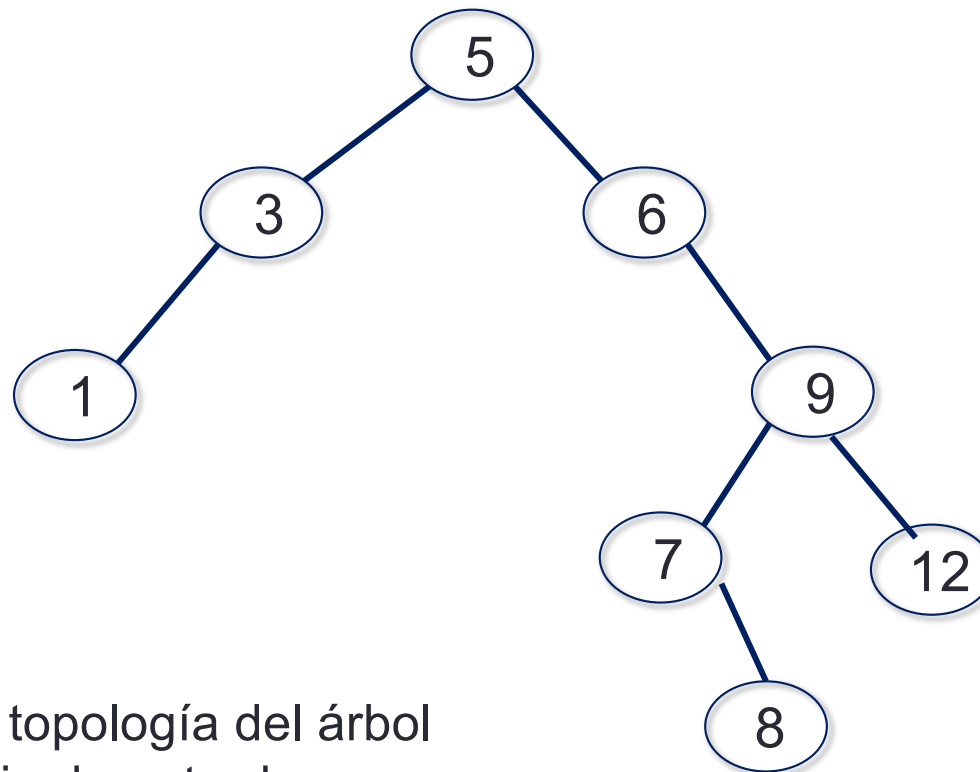
Índices en árbol B

Árbol:

- Nodos: raíz, nodo hoja, nodo interno, nodo padre y nodo hijo.
- Orden del árbol: número máximo de hijos de un nodo
- El nivel de un nodo es una unidad superior al de su padre (raíz: nivel 1)
- Subárbol de un nodo: un nodo hijo y todos sus descendientes
- Árbol de búsqueda: árbol organizado de forma que facilita la búsqueda de un valor entre un conjunto de valores (almacenados en los nodos)

Índices en árbol B

Árbol binario de búsqueda: 5, 3, 6, 1, 9, 7, 12, 8



- ✓ Árbol no equilibrado: la topología del árbol depende de la secuencia de entrada.
- ✓ Búsqueda de un valor no uniforme.
- ✓ Ocupación de los nodos: un valor en cada nodo.

Índices en árbol B

Árbol de búsqueda de orden p:

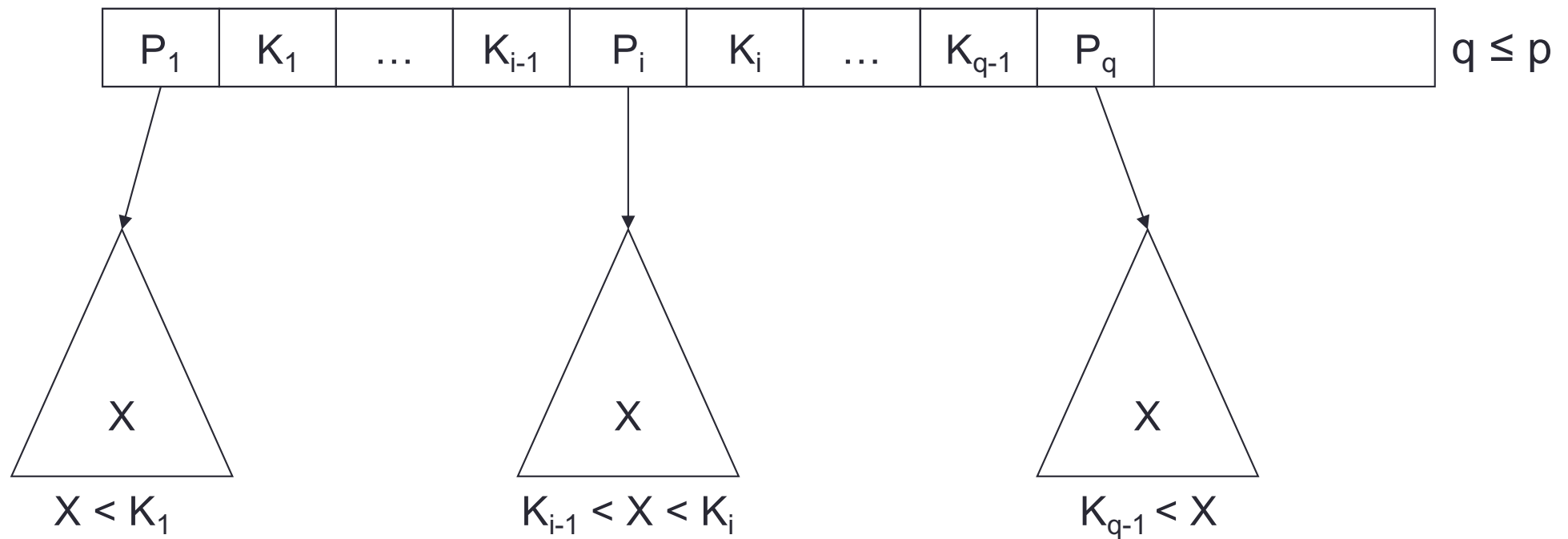
- Cada nodo tiene como máximo p hijos y p-1 valores: estructura

$$\langle P_1, K_1, P_2, K_2, \dots, P_{q-1}, K_{q-1}, P_q \rangle$$

- $q \leq p$
 - P_i es un puntero a un nodo hijo o un puntero nulo
 - K_i es un valor de búsqueda (único).
- Restricciones:
 1. En cada nodo: $K_1 < K_2 < \dots < K_{q-2} < K_{q-1}$
 2. Para todos los valores X del subárbol al que apunta P_i , se cumple
 - $K_{i-1} < X < K_i$ ($1 < i < q$)
 - $X < K_1$, ($i=1$)
 - $K_{q-1} < X$, ($i=q$)

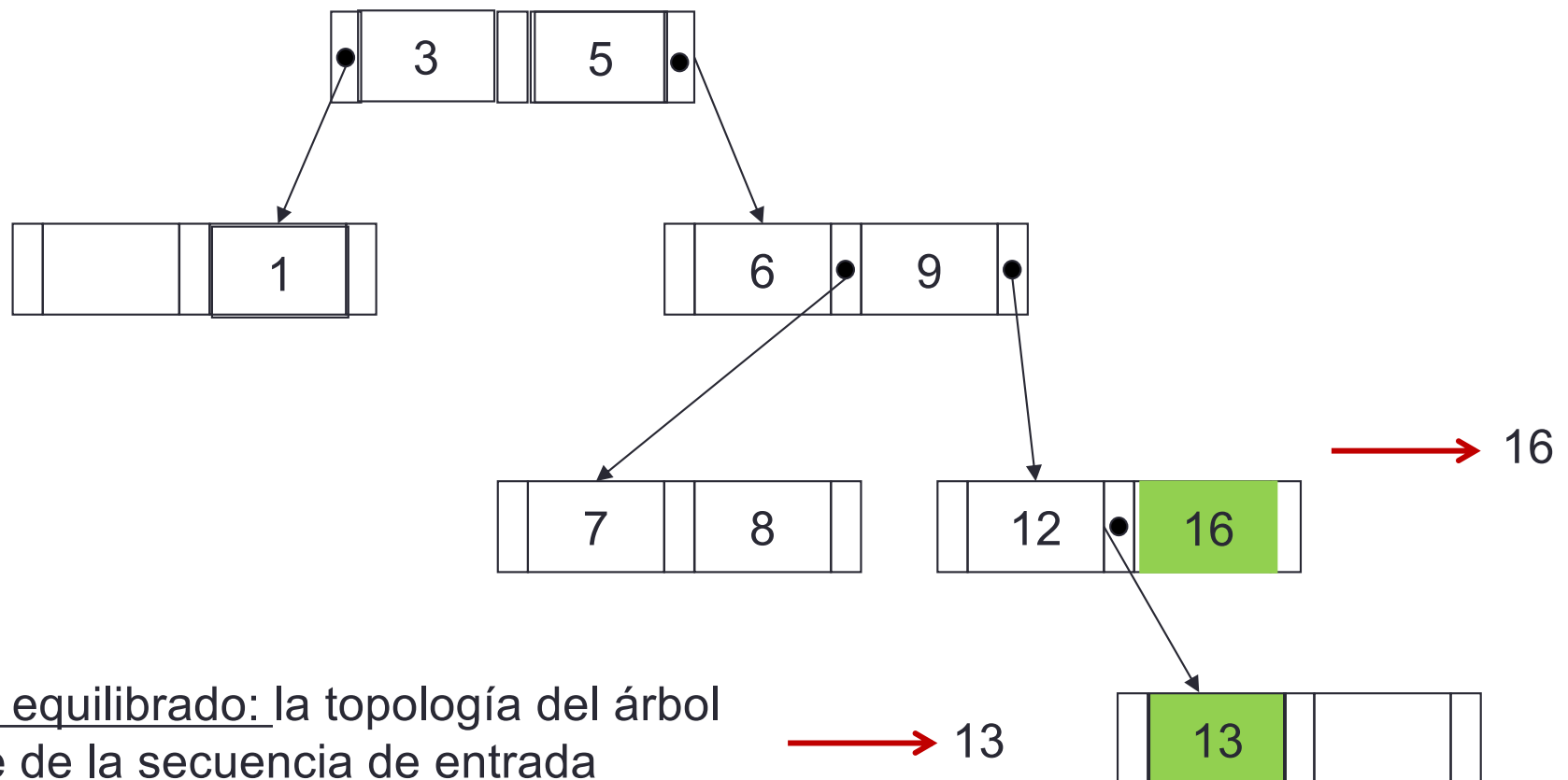
Índices en árbol B

Árbol de búsqueda de orden p : estructura de nodo



Índices en árbol B

Árbol de búsqueda de orden 3: 5, 3, 6, 1, 9, 7, 12, 8



- ✓ Árbol no equilibrado: la topología del árbol depende de la secuencia de entrada
- ✓ Búsqueda de un valor no uniforme.
- ✓ Ocupación no uniforme de los nodos

Índices en árbol B

Estructura de índice \equiv árbol de búsqueda de orden p

- campo de indexación \equiv valor de búsqueda del árbol
- En los nodos se almacenan **entradas de índice** de la forma $\langle \text{valor}, \text{puntero} \rangle$: valor del campo de indexación y puntero a registro (o bloque) del fichero de datos.
- Almacenamiento en disco del índice:
 - Cada nodo es un bloque de disco.
 - El orden del árbol dependerá del tamaño del bloque, del tamaño de los punteros y del tamaño del campo de indexación.

Índices en árbol B

Árbol B: árbol de búsqueda de orden p

Restricciones adicionales:

- Árbol equilibrado: todos los nodos hoja están al mismo nivel.
- Gestión eficiente del espacio en los nodos: los nodos tienen una ocupación entre el 50% y el 100%.

Índices en árbol B

Índice en árbol B de orden p (definido sobre un campo clave):

- Cada nodo tiene como máximo p hijos y p-1 valores: estructura

$$\langle P_1, \langle K_1, Pr_1 \rangle, P_2, \langle K_2, Pr_2 \rangle, \dots, \langle K_{q-1}, Pr_{q-1} \rangle, P_q \rangle$$

- $q \leq p$
- P_i es un puntero a un nodo hijo o un puntero nulo
- $\langle K_i, Pr_i \rangle$: entrada del índice
 - K_i es un valor del campo de indexación
 - Pr_i es un puntero al registro cuyo valor del campo de indexación es igual a K_i .

Índices en árbol B

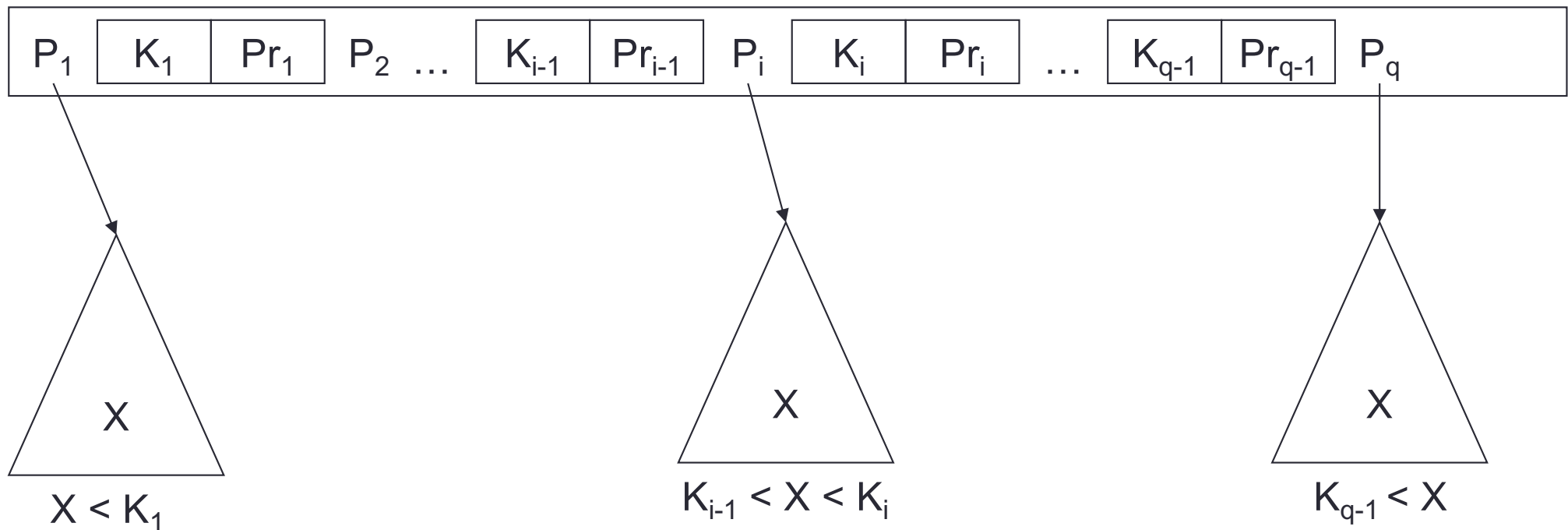
Índice en árbol B de orden p (definido sobre un campo clave):

➤ Restricciones:

1. En cada nodo: $K_1 < K_2 < \dots < K_{q-2} < K_{q-1}$
2. Para todos los valores X del subárbol al que apunta P_i , se cumple
 - $K_{i-1} < X < K_i$ ($1 < i < q$)
 - $X < K_1$ ($i = 1$)
 - $K_{q-1} < X$ ($i = q$)
3. Cada nodo, excepto el raíz y los nodos hoja, tiene como mínimo $p/2$ punteros de árbol. El nodo raíz tiene como mínimo dos punteros de árbol a no ser que sea el único nodo del árbol
4. Todos los nodos hoja están al mismo nivel y tienen la misma estructura que los internos, pero sus punteros de árbol son nulos

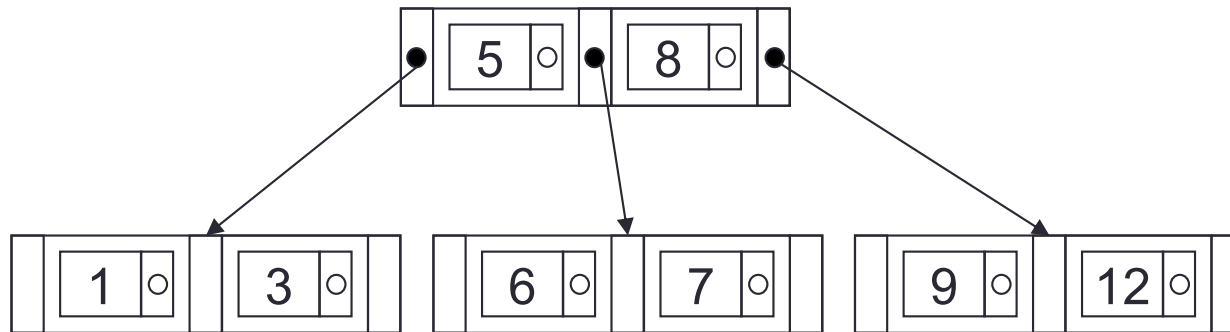
Índices en árbol B

Árbol B de orden p : estructura de un nodo



Índices en árbol B

Índice en árbol B de orden 3: 8, 5, 1, 7, 3, 12, 9, 6

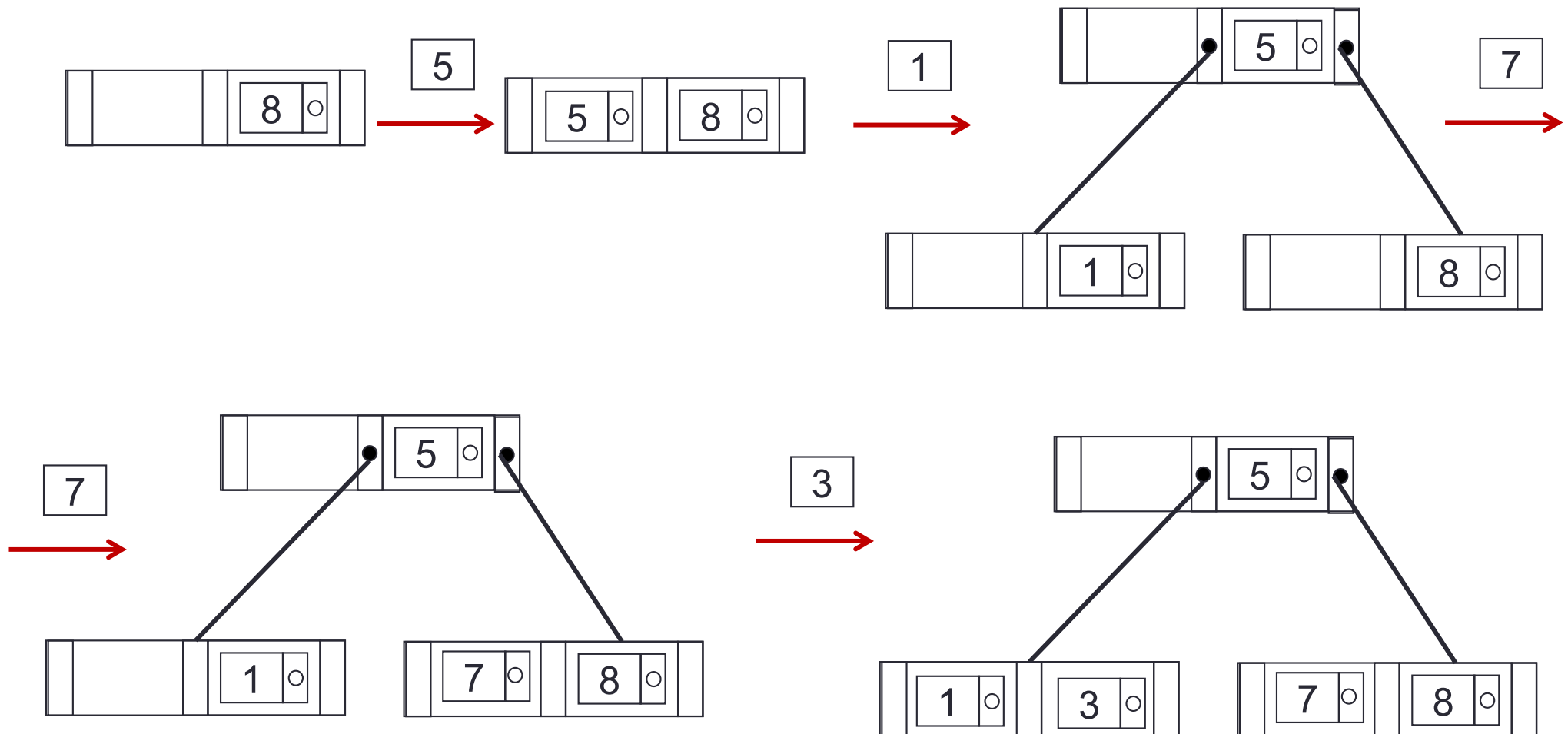


- Puntero a nodo de árbol
- Puntero a nodo de árbol nulo
- Puntero a registro del fichero de datos

- ✓ Árbol equilibrado
- ✓ Ocupación uniforme de los nodos
- ✓ Búsqueda de un valor no uniforme

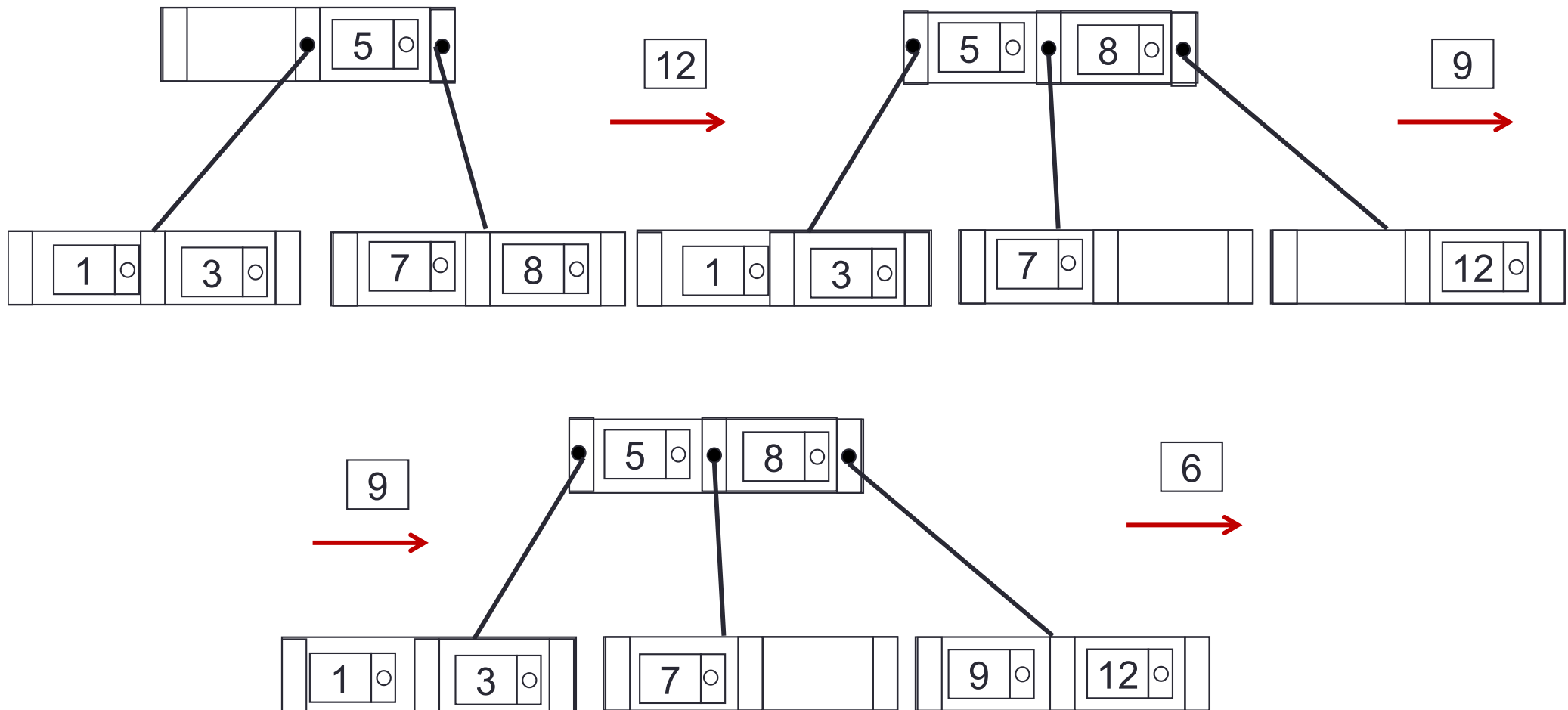
Índices en árbol B

Índice en árbol B de orden 3: 8, 5, 1, 7, 3, 12, 9, 6



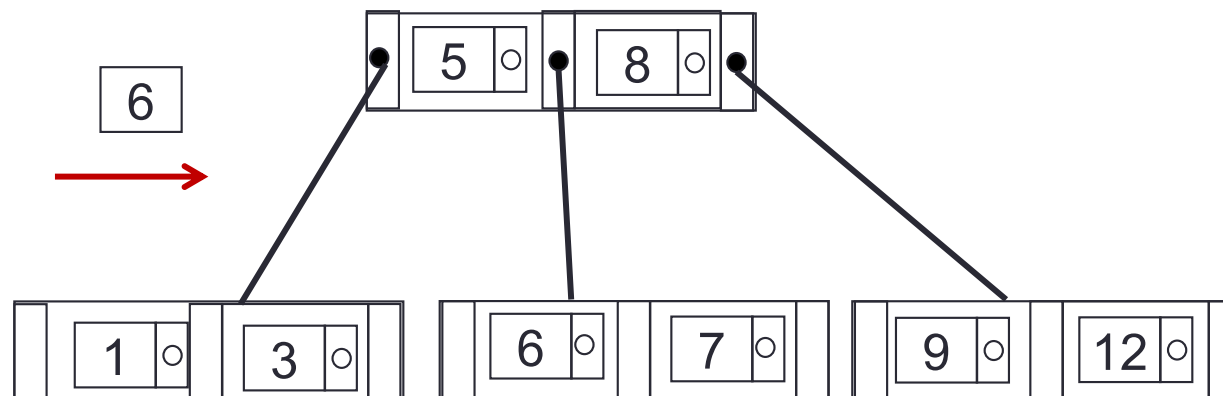
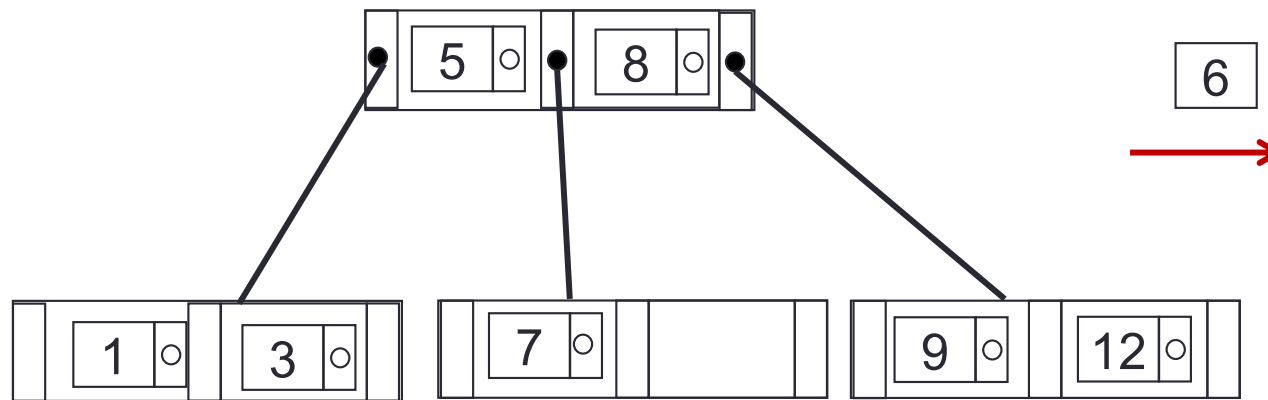
Índices en árbol B

Índice en árbol B de orden 3: 8, 5, 1, 7, 3, 12, 9, 6



Índices en árbol B

Índice en árbol B de orden 3: 8, 5, 1, 7, 3, 12, 9, 6



Índices en árbol B

Árbol B⁺: variante del árbol B

- Orden p: los nodos tienen como máximo p hijos y p-1 valores.
- Equilibrado: todos los nodos hoja están al mismo nivel.
- Espacio en los nodos: todos los nodos tienen una ocupación entre el 50% y el 100%.
- Nodos con estructura distinta: nodos internos y nodos hoja
- Las entradas del índice sólo se almacenan en los nodos hoja.
- Los valores en los nodos internos sirven para organizar la búsqueda.
- Los nodos hoja están enlazados para favorecer la búsqueda ordenada por el campo de indexación.

BB⁺

Índices en árbol B

Índice en árbol B⁺ de orden p (definido sobre un campo clave)

- Cada nodo interno tiene como máximo p hijos p-1 y valores: estructura:

$$\langle P_1, K_1, P_2, K_2, \dots, P_{q-1}, K_{q-1}, P_q \rangle$$

- $q \leq p$
- P_i es un puntero a un nodo hijo o un puntero nulo
- K_i es un valor del campo de indexación

Índices en árbol B

Índice en árbol B⁺ de orden p (definido sobre una clave)

Restricciones:

1. En cada nodo interno: $K_1 < K_2 < \dots < K_{q-2} < K_{q-1}$
2. Para todos los valores X del subárbol al que apunta P_i , se cumple
 - $K_{i-1} < X \leq K_i$ ($1 < i < q$)
 - $X \leq K_1$ ($i = 1$)
 - $K_{q-1} < X$ ($i = q$)
3. Cada nodo interno tiene como mínimo $p/2$ punteros de árbol. El nodo raíz tiene al menos dos punteros de árbol si es un nodo interno

Índices en árbol B

Índice en árbol B⁺ de orden p (definido sobre una clave)

➤ Cada nodo hoja tiene la forma:

$\langle \langle K_1, Pr_1 \rangle, \langle K_2, Pr_2 \rangle, \dots, \langle K_{q-1}, Pr_{q-1} \rangle, P_{\text{siguiente}} \rangle \quad q \leq p$

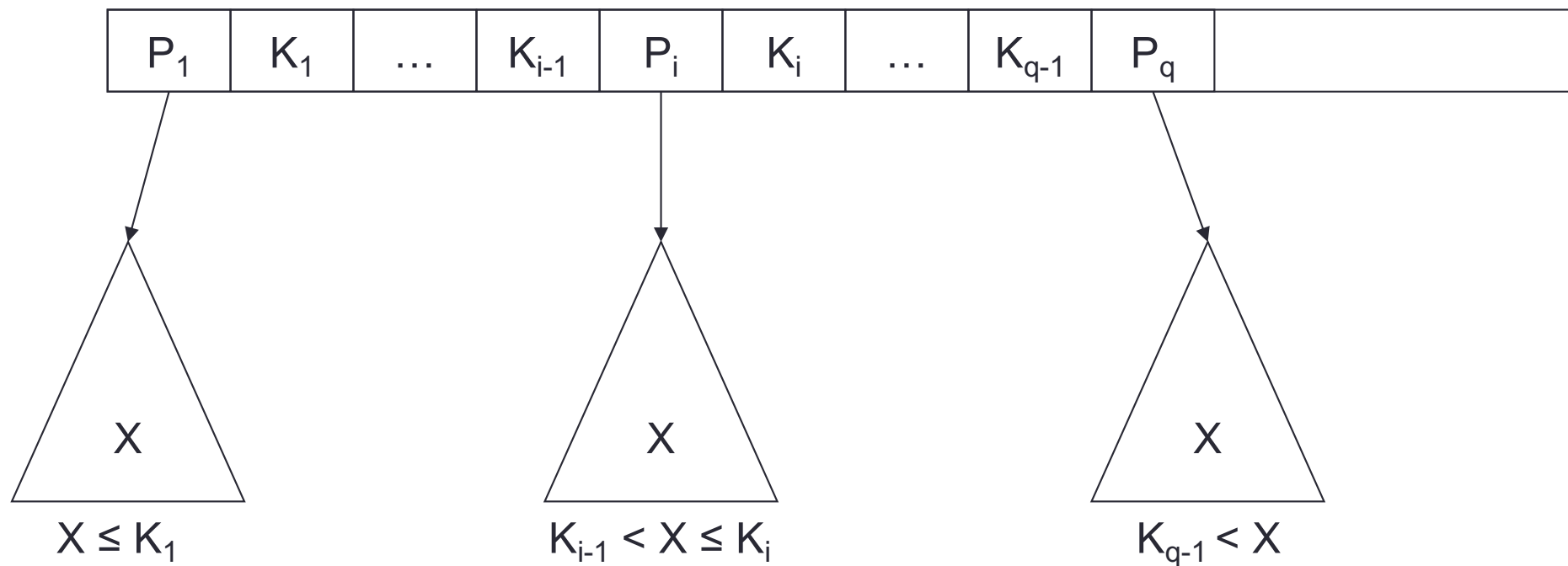
- $\langle K_1, Pr_1 \rangle$: entrada del índice
 - Pr_i es un puntero a registro
 - K_i es un valor del campo de indexación
- $P_{\text{siguiente}}$ puntero al **siguiente** nodo hoja del árbol

Restricciones:

2. En cada nodo hoja: $K_1 < K_2 < \dots < K_{q-2} < K_{q-1}$
3. Cada nodo hoja tiene como mínimo $p/2$ entradas
4. Todos los nodos hoja están al mismo nivel

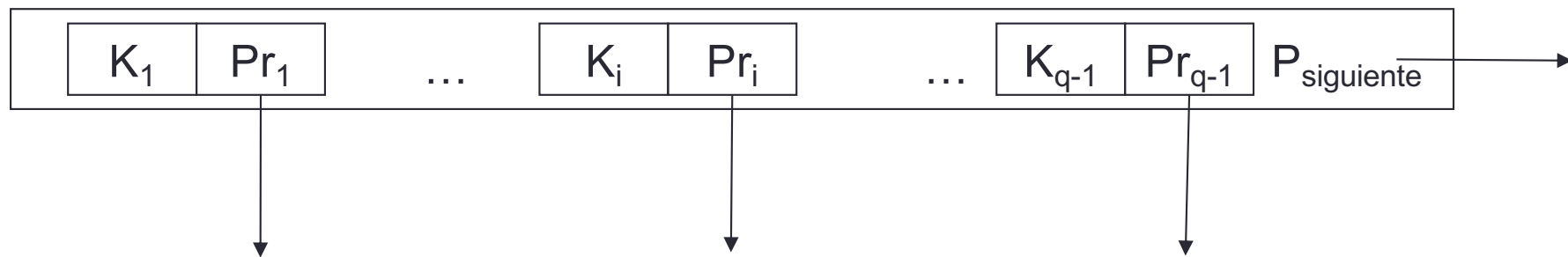
Índices en árbol B

Nodo interno de árbol B^+ de orden p :



Índices en árbol B

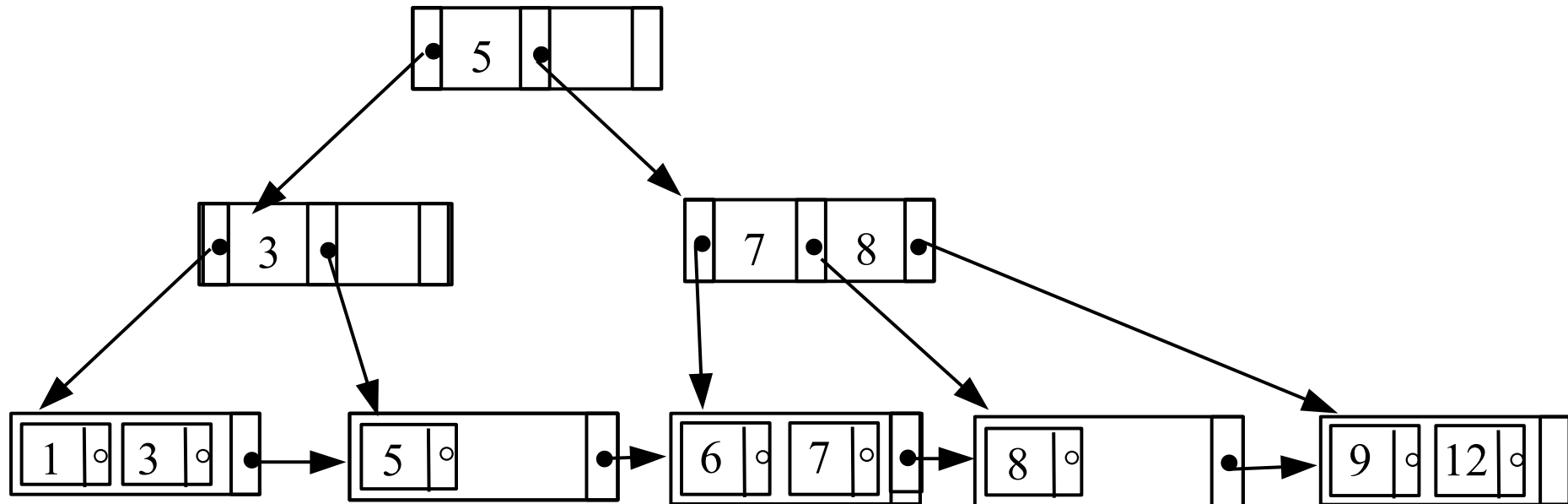
Nodo hoja de árbol B⁺ de orden p:



Índices en árbol B

Árbol B⁺ de orden 3

SECUENCIA DE INSERCIÓN : 8, 5, 1, 7, 3, 12, 9, 6



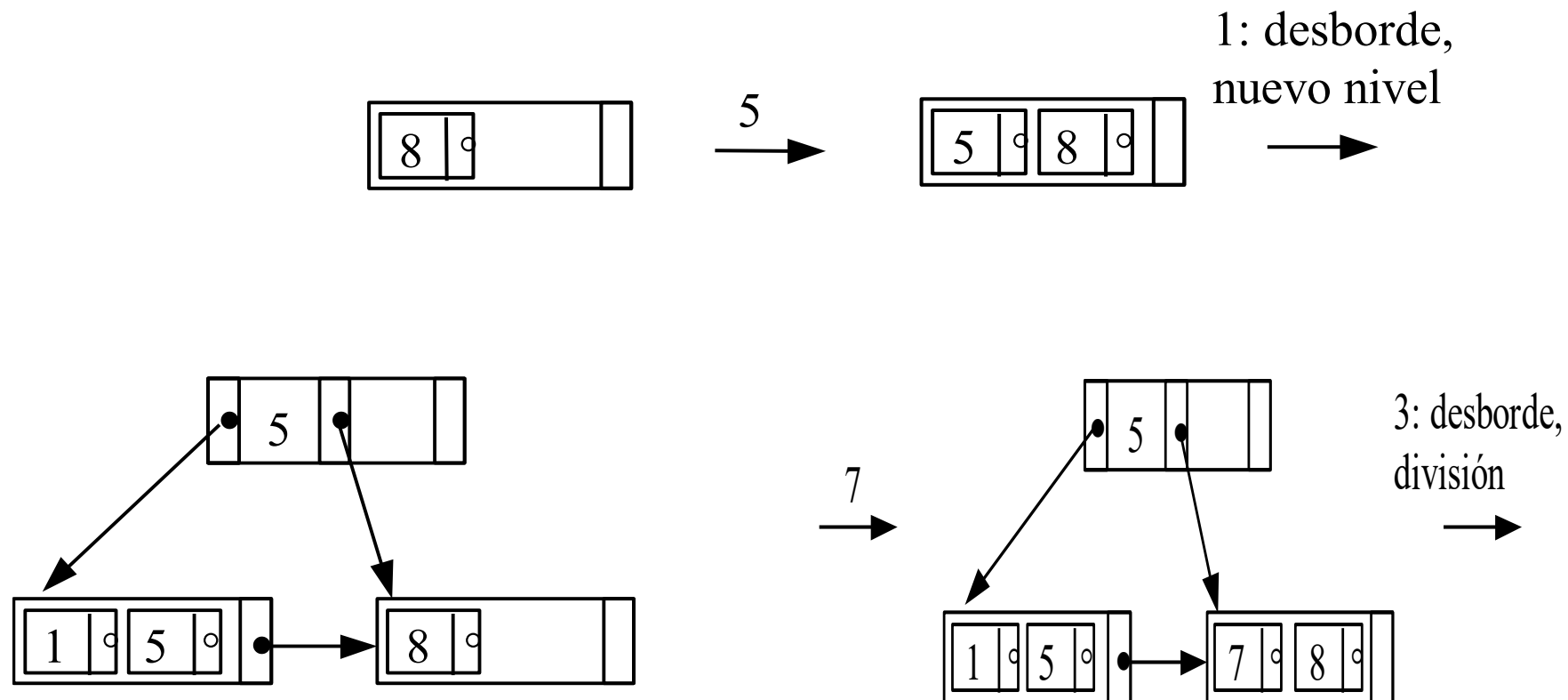
- ✓ Árbol equilibrado
- ✓ Ocupación uniforme de los nodos
- ✓ Búsqueda de un valor uniforme

- puntero de árbol
- puntero de registro
- puntero de árbol nulo

Índices en árbol B

Árbol B⁺ de orden 3

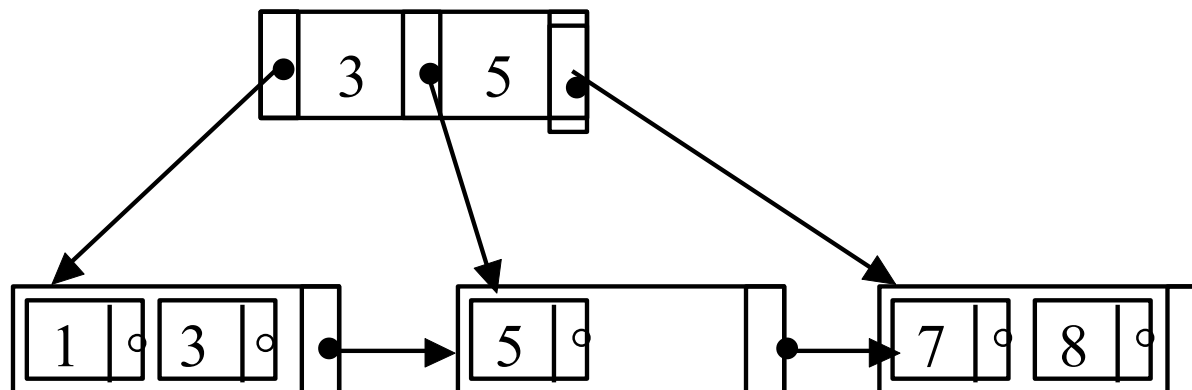
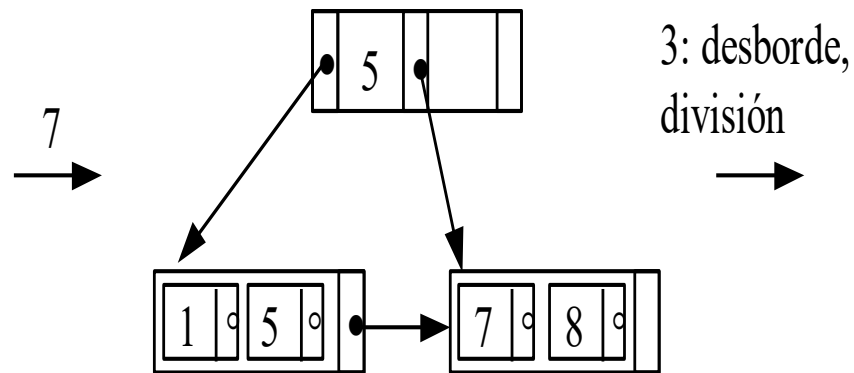
SECUENCIA DE INSERCIÓN : 8, 5, 1, 7, 3, 12, 9, 6



Índices en árbol B

Árbol B⁺ de orden 3

SECUENCIA DE INSERCIÓN : 8, 5, 1, 7, 3, 12, 9, 6

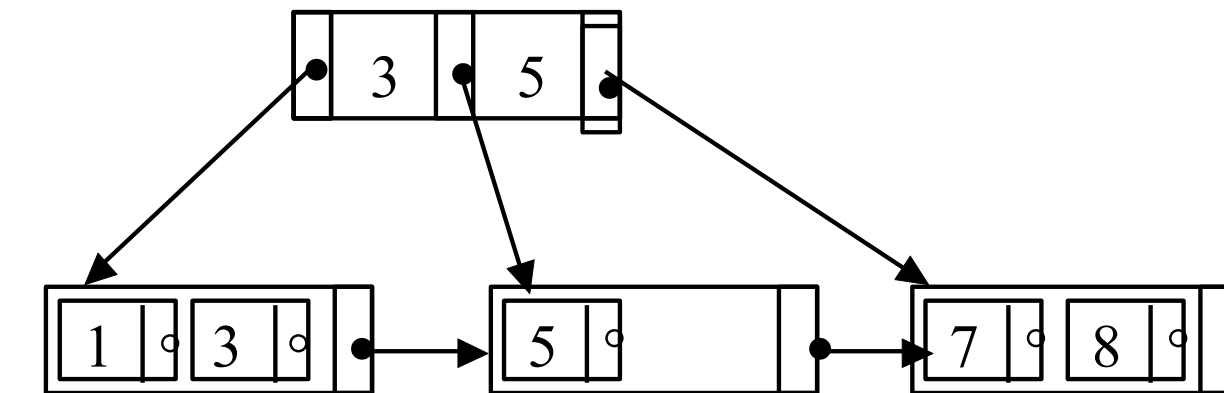


12: desborde, división,
nuevo nivel

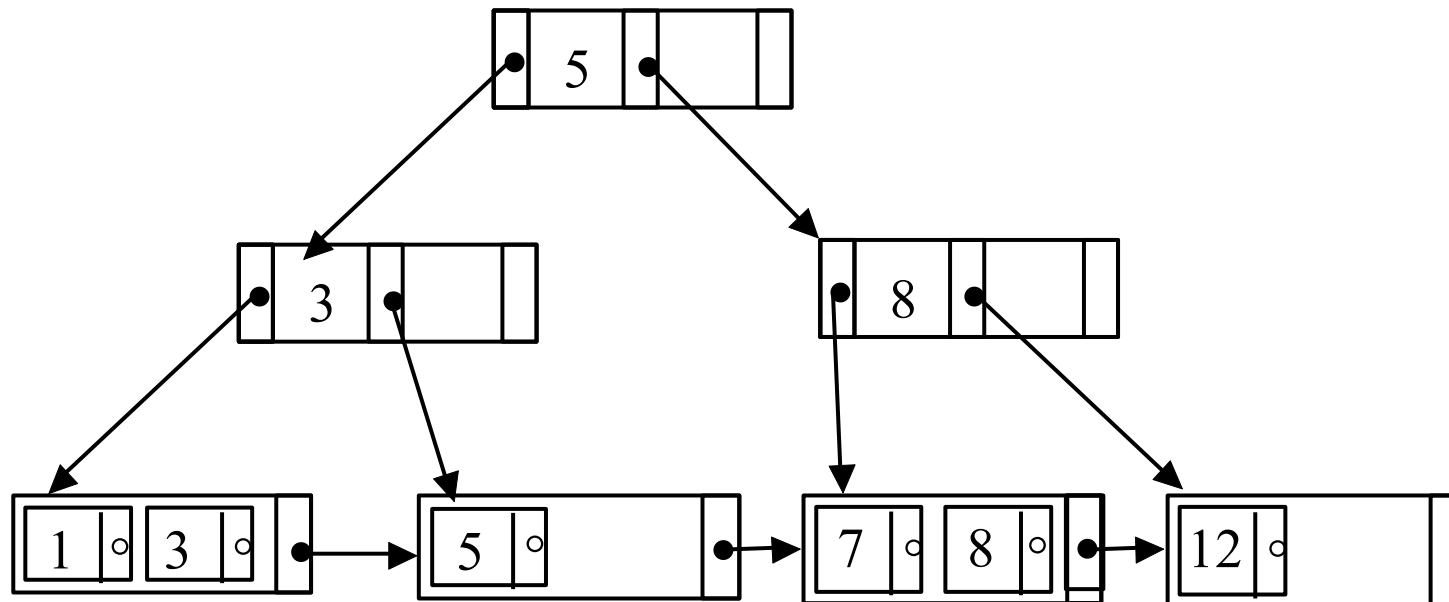
Índices en árbol B

Árbol B⁺ de orden 3

SECUENCIA DE INSERCIÓN : 8, 5, 1, 7, 3, 12, 9, 6

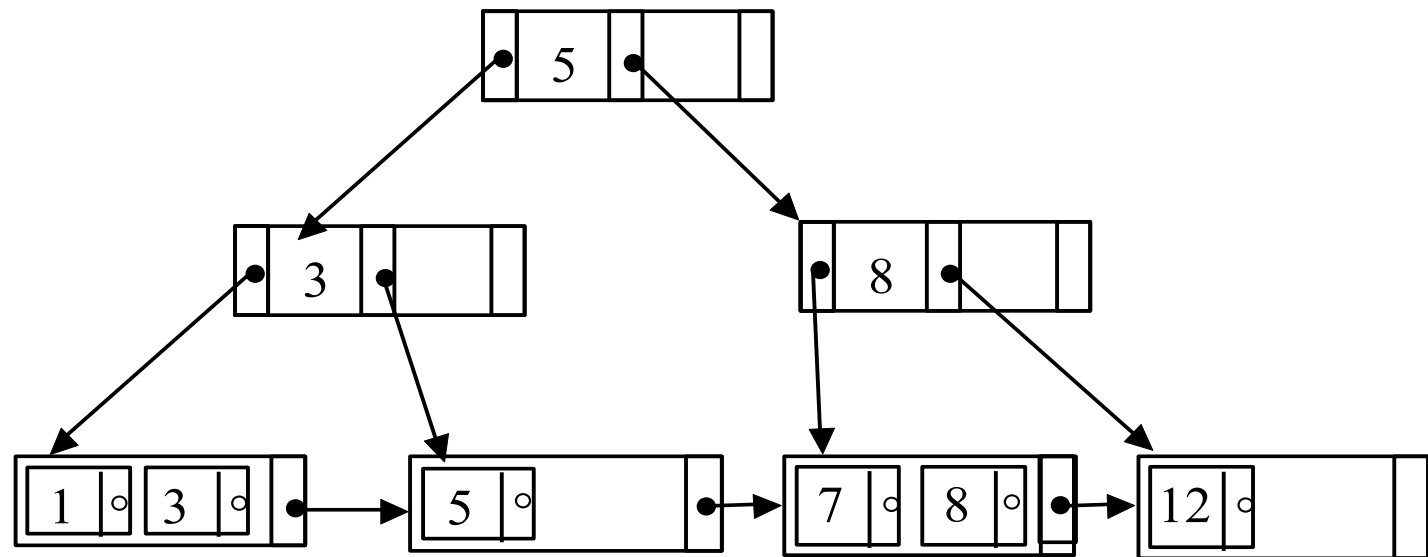


12: desborde, división,
nuevo nivel

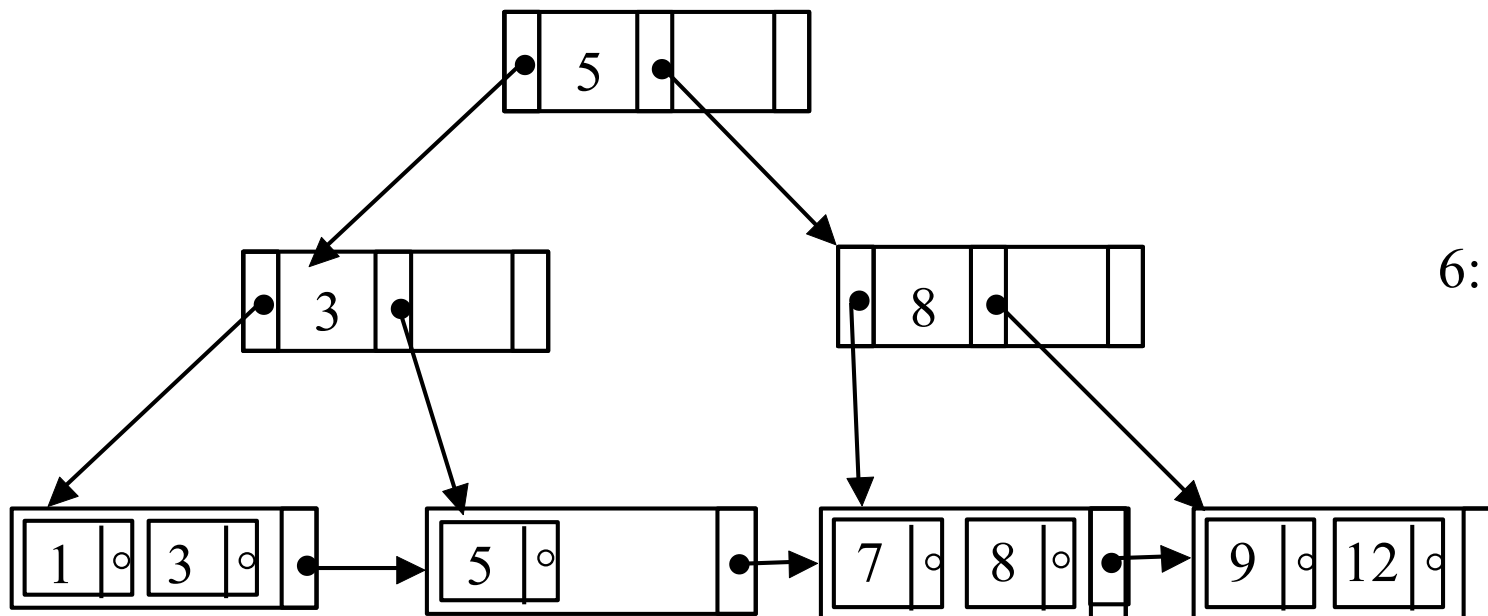


9
→

Secuencia de inserción: 8, 5, 1, 7, 3, 12, 9, 6

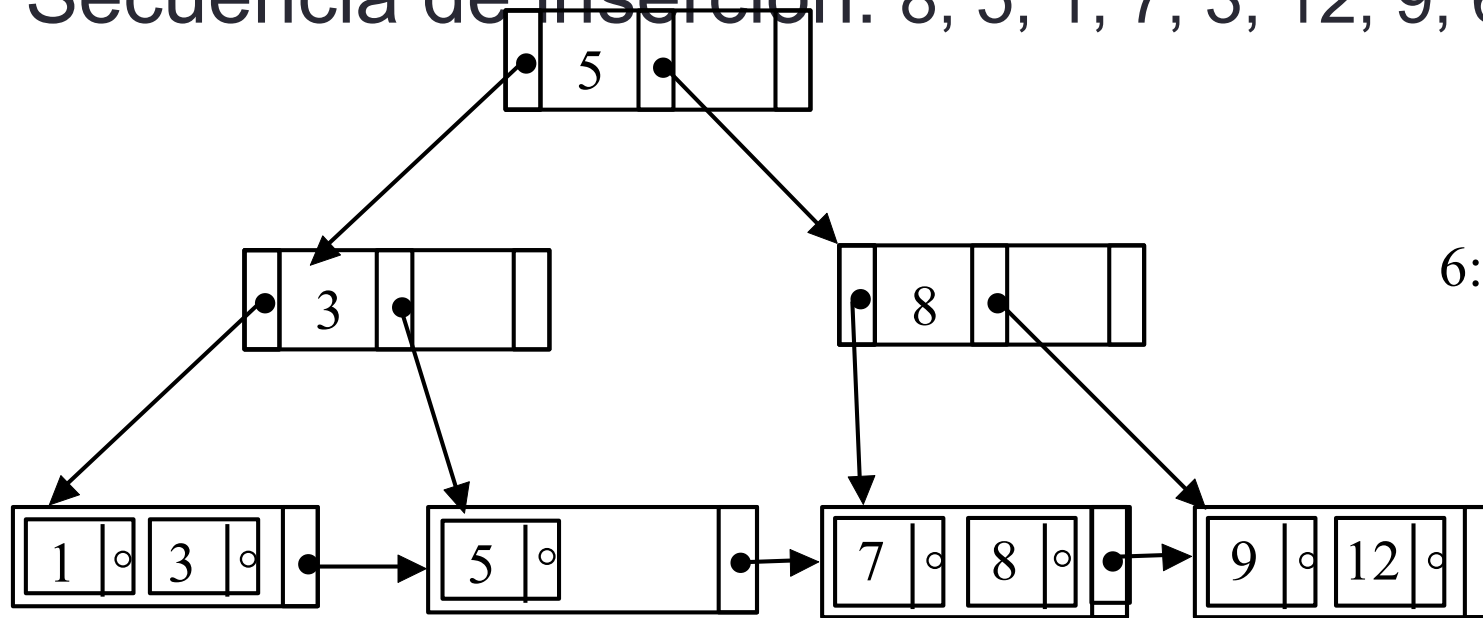


9
→

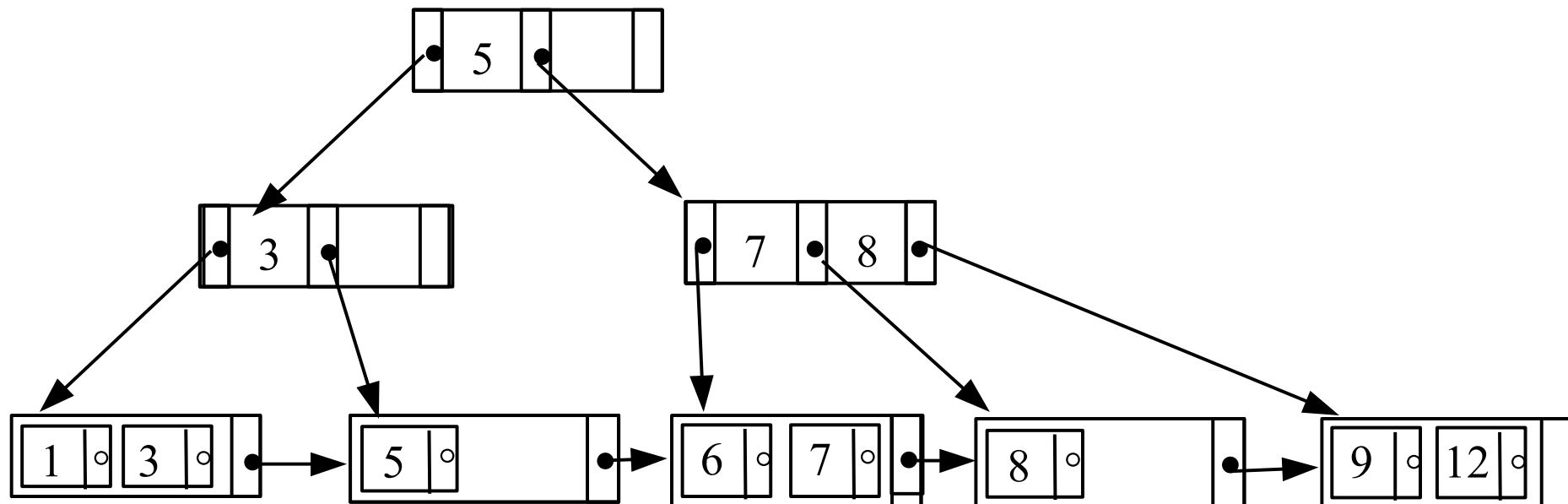


6: desborde, división
→

Secuencia de inserción: 8, 5, 1, 7, 3, 12, 9, 6



6: desborde, división



Anexo II:

Protocolo para el control de la
conurrencia B2F.

Protocolo de bloqueo B2F

Protocolos de bloqueo de elementos de datos:

Los protocolos de bloqueo de elementos de datos se basan en la idea de restringir el acceso al mismo elemento de datos por varias transacciones.

Las transacciones solicitan el bloqueo de un elemento de datos para acceder a él, y lo liberan posteriormente.

Cuando una transacción solicita el bloqueo de un elemento de datos y las reglas de bloqueo se lo impiden, la transacción queda en espera.

Bloqueo: estado de un elemento de datos, provocado por una transacción, que determina las operaciones que otras transacciones pueden realizar sobre él.

Protocolo de bloqueo B2F

Bloqueo de lectura/escritura (compartido/exclusivo):

En el bloqueo de lectura/escritura un elemento de datos puede estar en tres estados: 'bloqueado para lectura', 'bloqueado para escritura' o 'desbloqueado'.

- Una transacción debe '**bloquear para lectura**' el elemento de datos X antes de realizar una operación **leer(X)**.
- Una transacción debe '**bloquear para escritura**' el elemento de datos X antes de realizar una operación **escribir(X)**. En este caso también podrá realizar la operación **leer(X)**.
- Si **bloqueo(X)='bloqueado para lectura'**, otras transacciones distintas a la que realizó el bloqueo de X pueden 'bloquear para lectura' el elemento de datos (**bloqueo compartido**).
- Si **bloqueo(X)='bloqueado para escritura'**, ninguna otra transacción distinta a la que realizó el bloqueo de X puede bloquear el elemento de datos (**bloqueo exclusivo**).

Protocolo de bloqueo B2F

Operaciones para usar bloqueo de lectura/escritura:

Bloquear_escritura (X):

B: IF bloqueo(X)='desbloqueado' (el elemento está desbloqueado)

THEN bloqueo(X) ← 'bloqueado para escritura' (bloqueado para escritura)
añadir T a la lista de transacciones del elemento X

ELSE WAIT (hasta que bloqueo(X)='desbloqueado' y el
gestor de bloqueos considere a T)

GOTO B

END IF.

Protocolo de bloqueo B2F

Bloquear_lectura (X):

B: IF bloqueo(X) = 'desbloqueado' (el elemento está desbloqueado)

THEN bloqueo(X) \leftarrow 'bloqueado para lectura' (bloqueado para leer)

nro_lecturas \leftarrow 1

añadir T a la lista de transacciones del elemento X

ELSE

IF bloqueo(X) = 'bloqueado para lectura'

THEN nro_lecturas \leftarrow nro_lecturas + 1

añadir T a la lista de transacciones del elemento X

ELSE WAIT (hasta que bloqueo(X) = 'desbloqueado' y el
gestor de bloqueos considere a T)

GOTO B

END IF

END IF.

Protocolo de bloqueo B2F

Desbloquear (X):

IF bloqueo(X)='bloqueado para escritura'

THEN bloqueo(X) ← 'desbloqueado'

 eliminar T de la lista de transacciones de X

 --considerar alguna transacción que espera para bloquear X--

ELSE

 IF bloqueo(X) = 'bloqueado para lectura'

 THEN nro_lecturas ← nro_lecturas – 1

 eliminar T de la lista de transacciones del elemento X

 IF nro_lecturas=0

 THEN bloqueo(X) ← 'desbloqueado'

 --considerar alguna transacción que espera para bloquear X--

 END IF

 END IF

END IF

Protocolo de bloqueo B2F

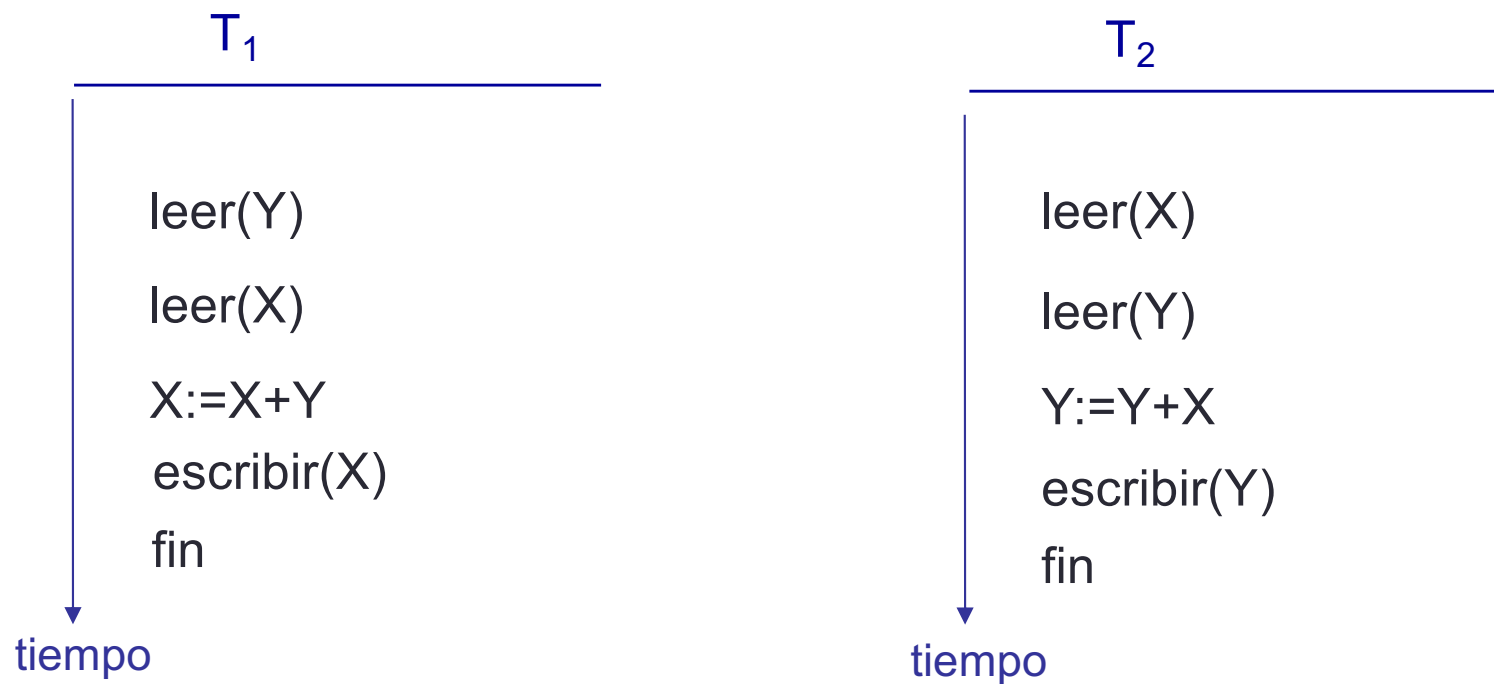
Conversión de bloqueos:

- ✓ Una transacción T que tiene un bloqueo de lectura sobre el elemento X puede solicitar bloquear_escritura (X): si T es la única transacción con bloqueo de lectura sobre X, es posible **promover el bloqueo**, de lo contrario T debe esperar.
- ✓ Una transacción T que tiene un bloqueo de escritura sobre el elemento X puede solicitar bloquear_lectura (X) y **degradar el bloqueo**.

Protocolo de bloqueo B2F

X=20

Y=30

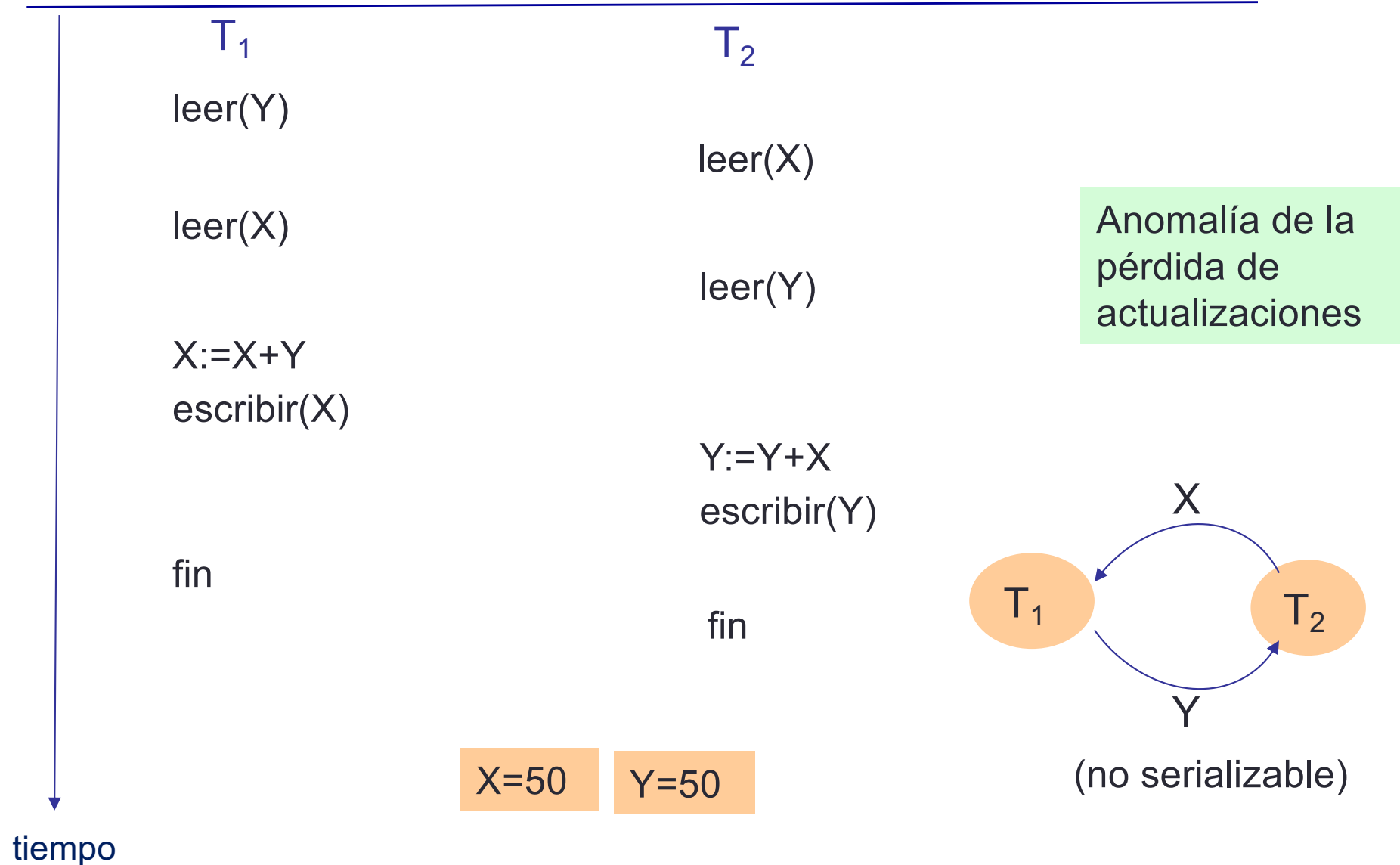


$T_1 - T_2$ X=50, Y=80

$T_2 - T_1$ X=70, Y=50

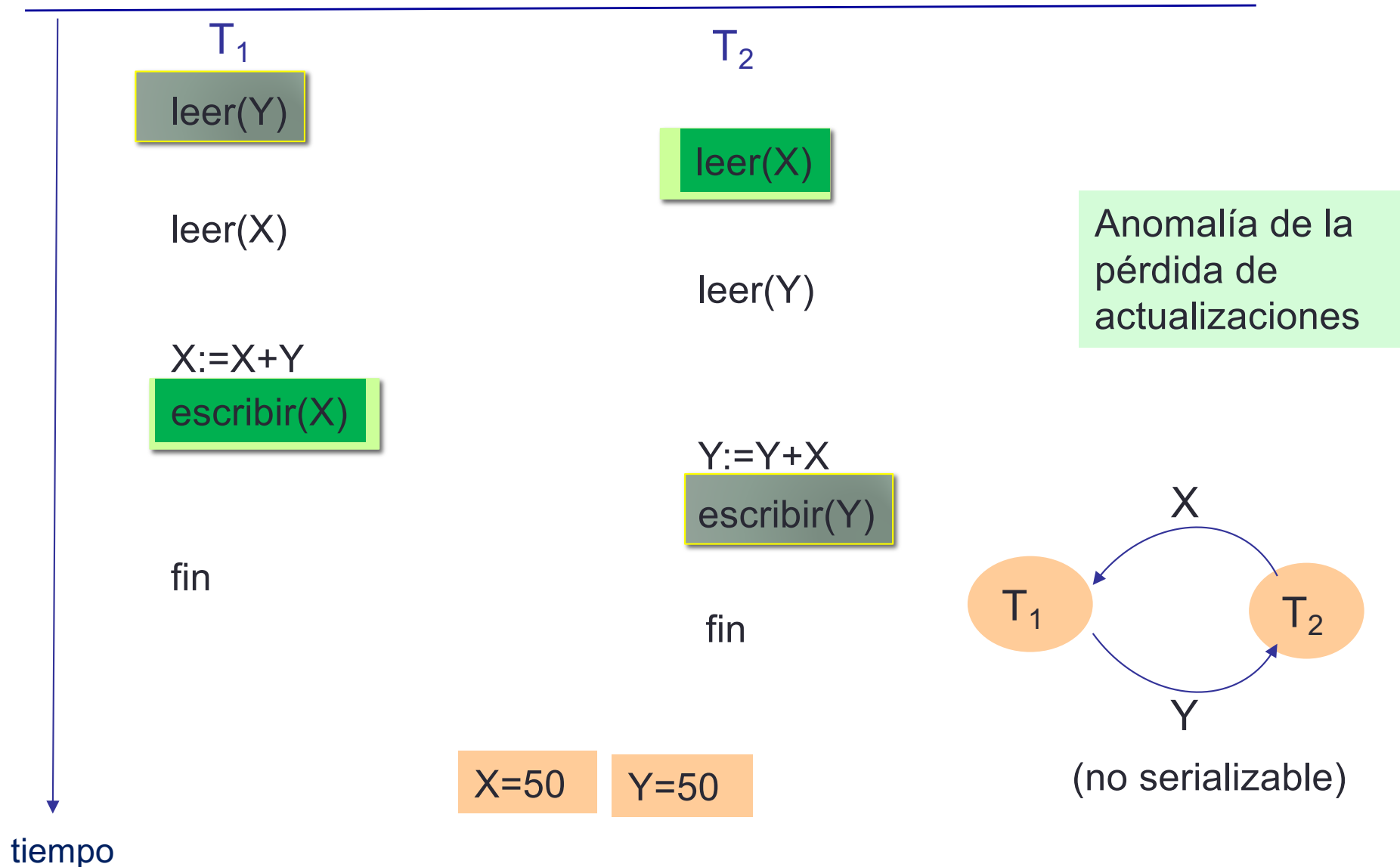
Protocolo de bloqueo B2F

Plan concurrente para T_1 y T_2



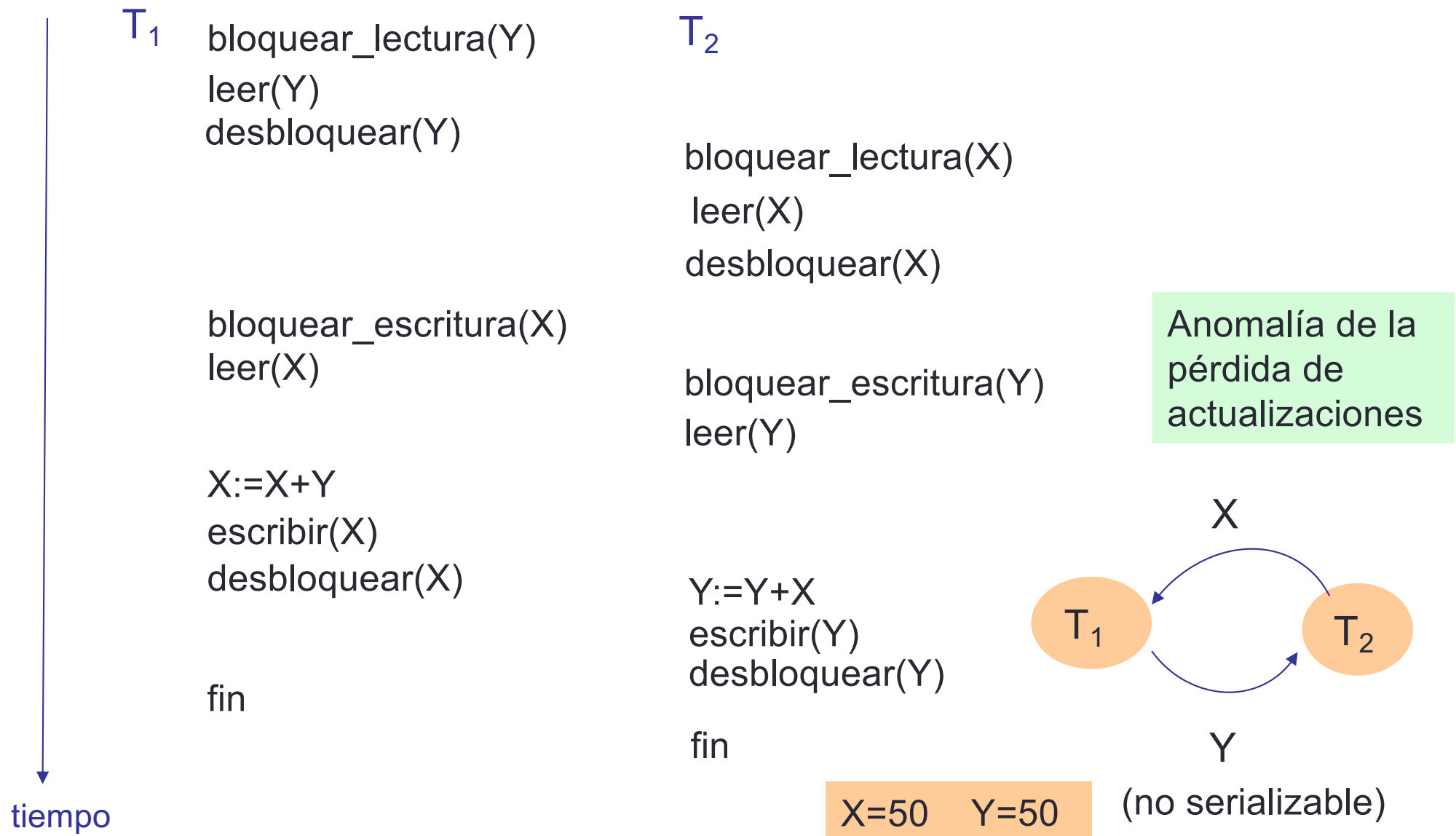
Protocolo de bloqueo B2F

Plan concurrente para T_1 y T_2



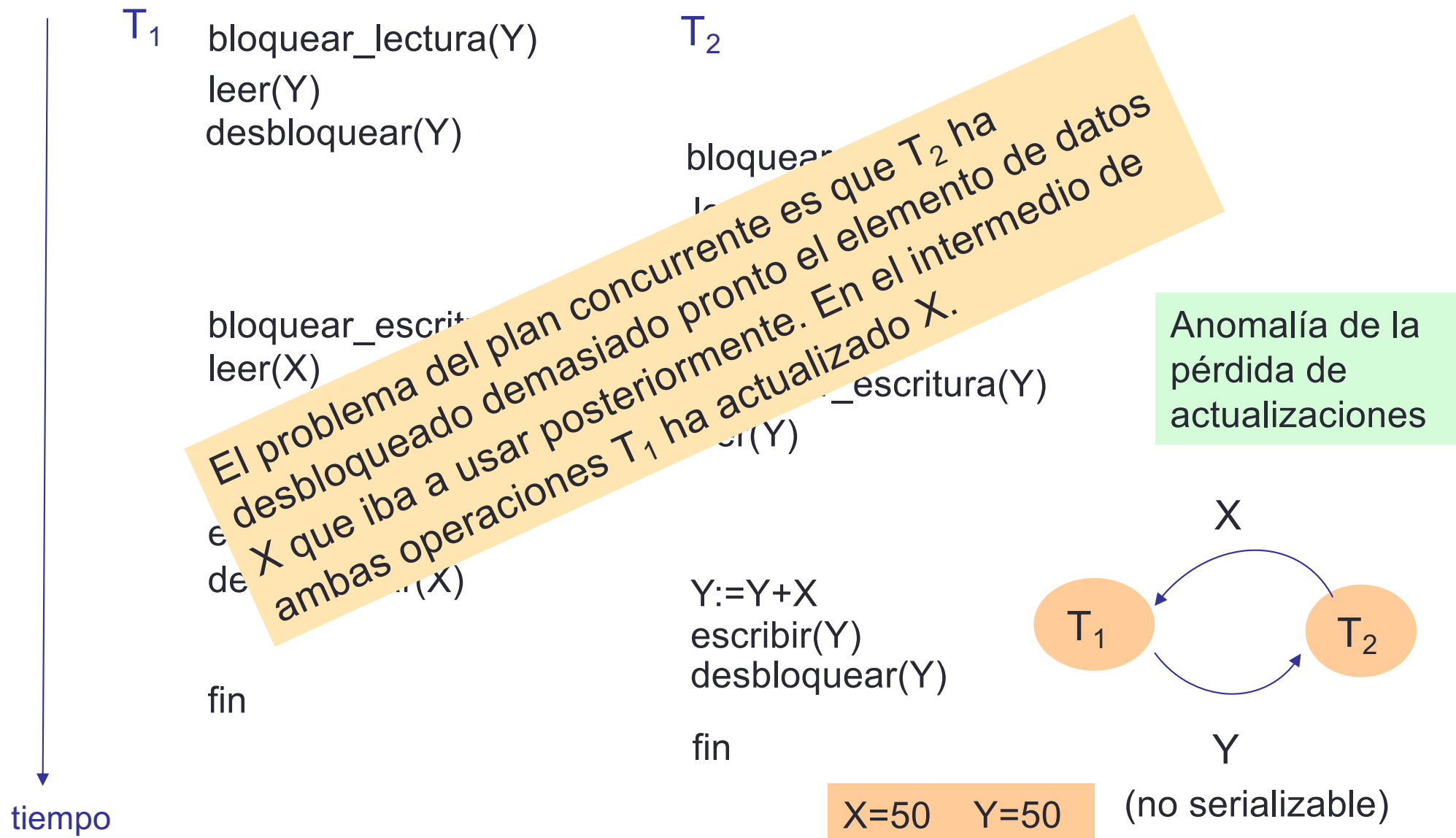
Protocolo de bloqueo B2F

Plan concurrente para T_1 y T_2 (con bloqueo de lectura/escritura)



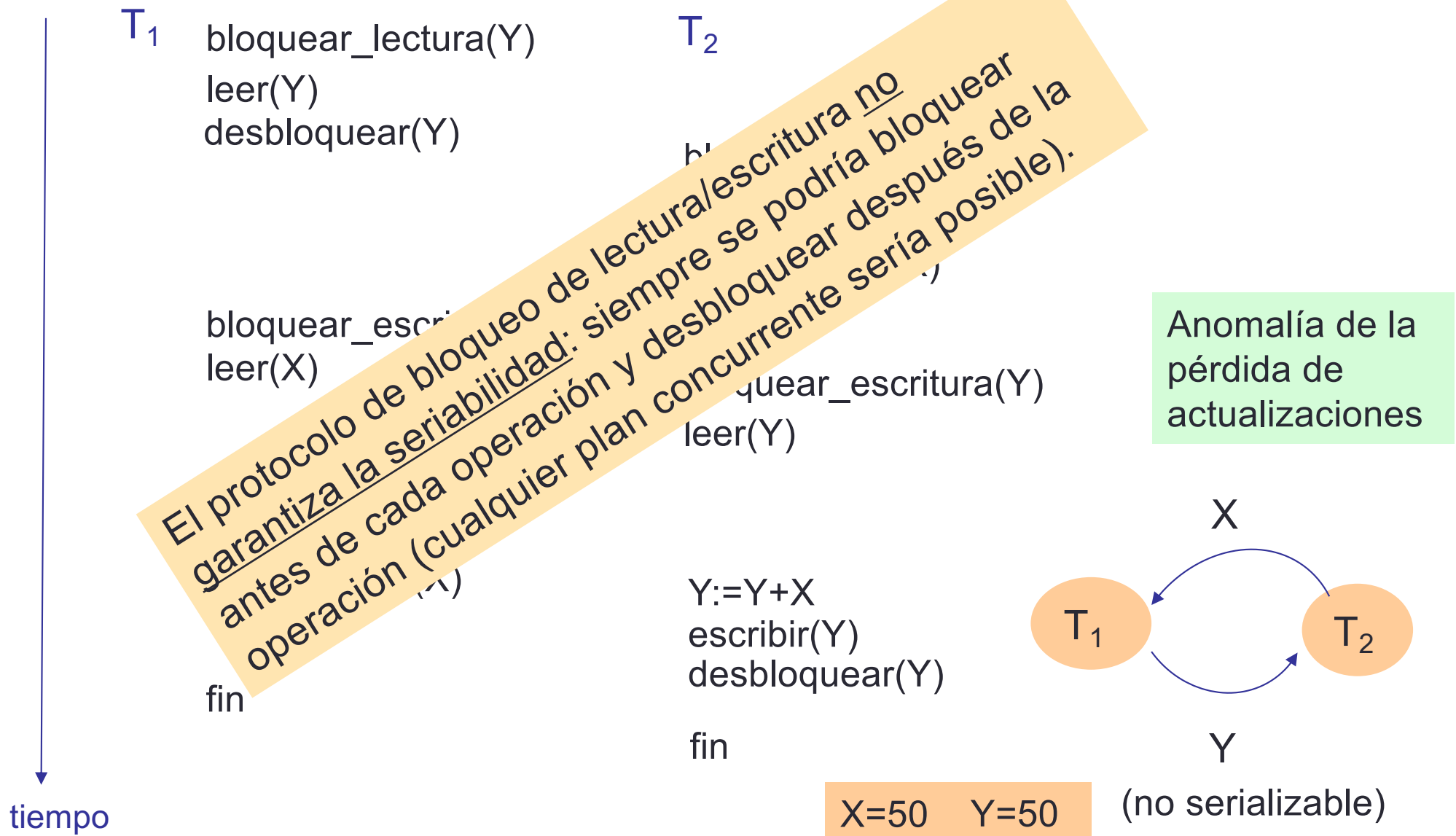
Protocolo de bloqueo B2F

Plan concurrente para T_1 y T_2 (con bloqueo de lectura/escritura)



Protocolo de bloqueo B2F

Plan concurrente para T_1 y T_2 (con bloqueo de lectura/escritura)



Protocolo de bloqueo B2F

El protocolo de bloqueo de lectura/escritura no garantiza la seriabilidad (por conflictos) de los planes

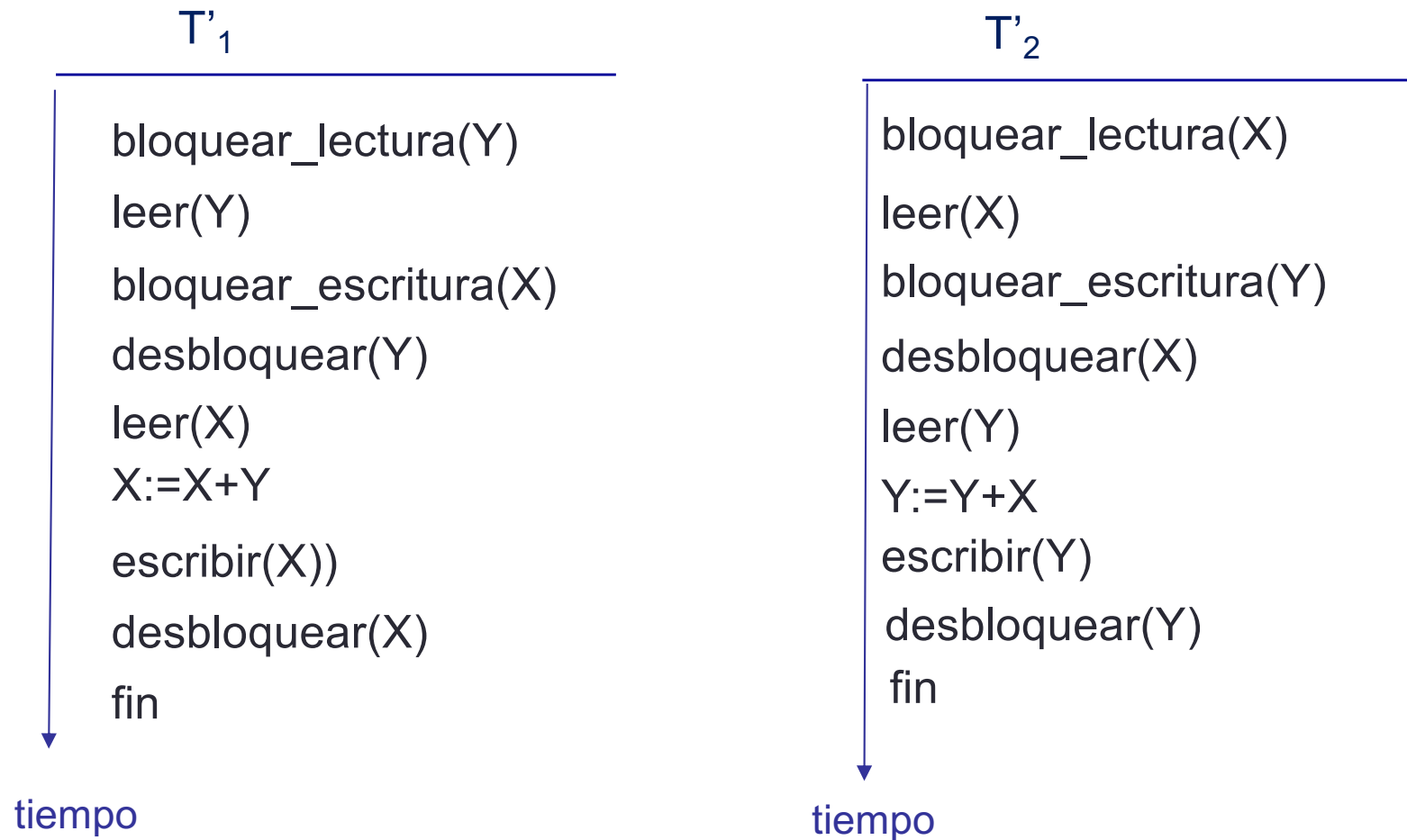


Bloqueo en dos fases (B2F)



En una transacción todas las operaciones de bloqueo (bloquear_lectura, bloquear_escritura) preceden a la primera operación de desbloqueo (desbloquear) de la transacción.

Protocolo de bloqueo B2F



T'_1 y T'_2 cumplen la regla de bloqueo en dos fases (B2F)

Protocolo de bloqueo B2F

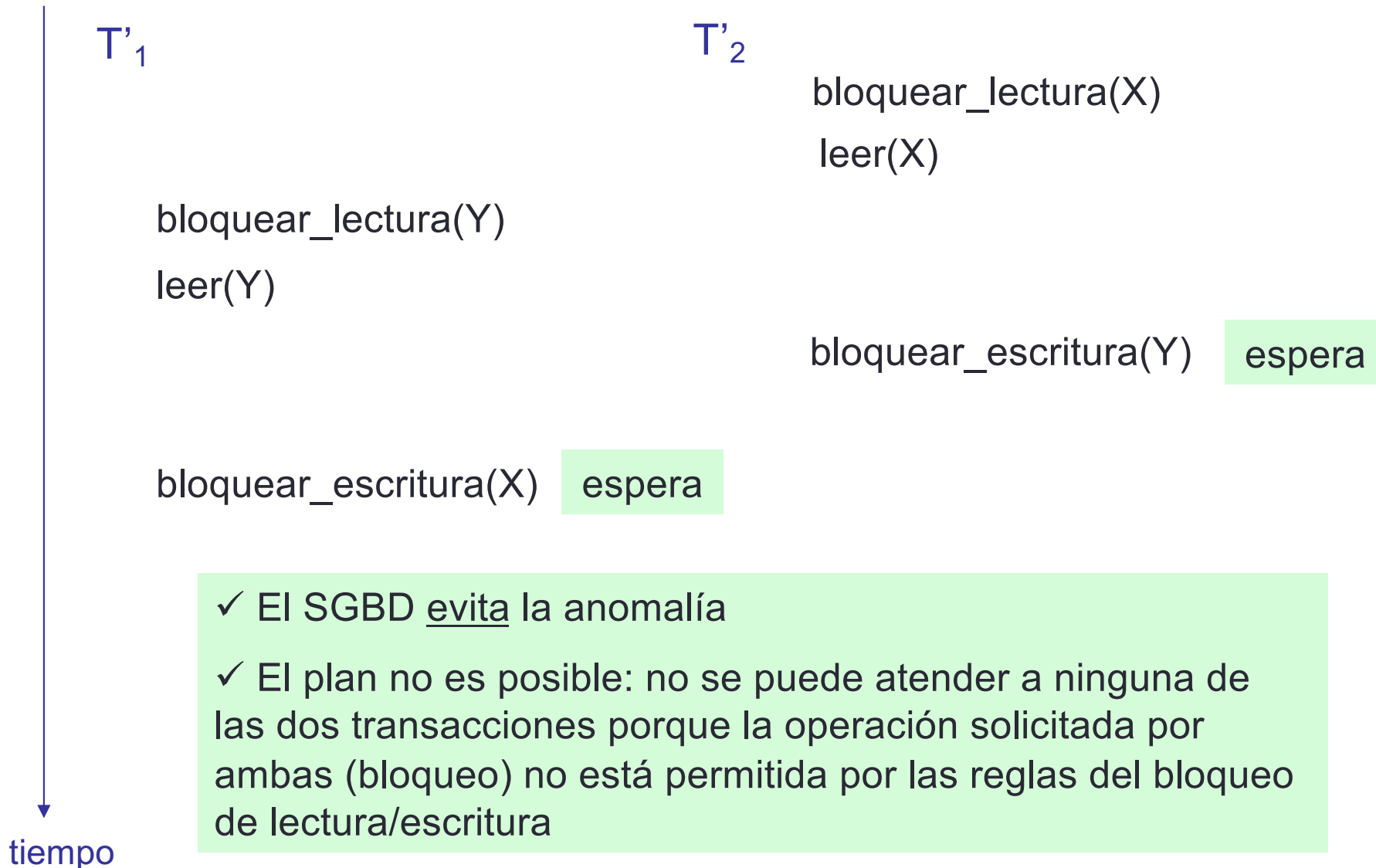
Puede demostrarse que si todas las transacciones que participan en un plan cumplen la regla de bloqueo en dos fases (B2F), entonces el protocolo de bloqueo lectura/escritura garantiza la seriabilidad del plan.



El protocolo de bloqueo lectura/escritura en dos fases (B2F) limita la concurrencia, pero asegura la seriabilidad de los planes sin tener que examinarlos.

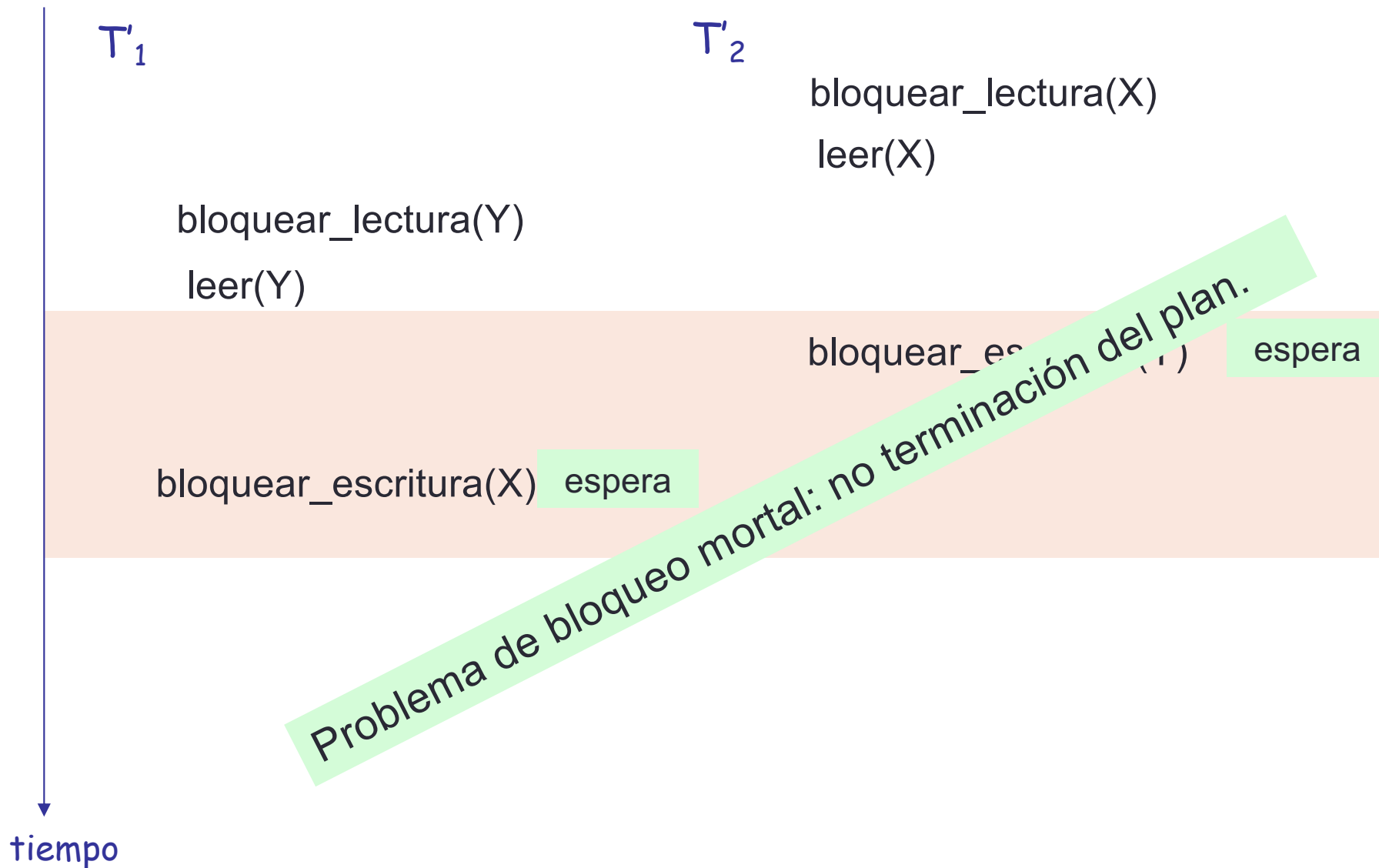
Protocolo de bloqueo B2F

Plan concurrente para T'_1 y T'_2 (con B2F)



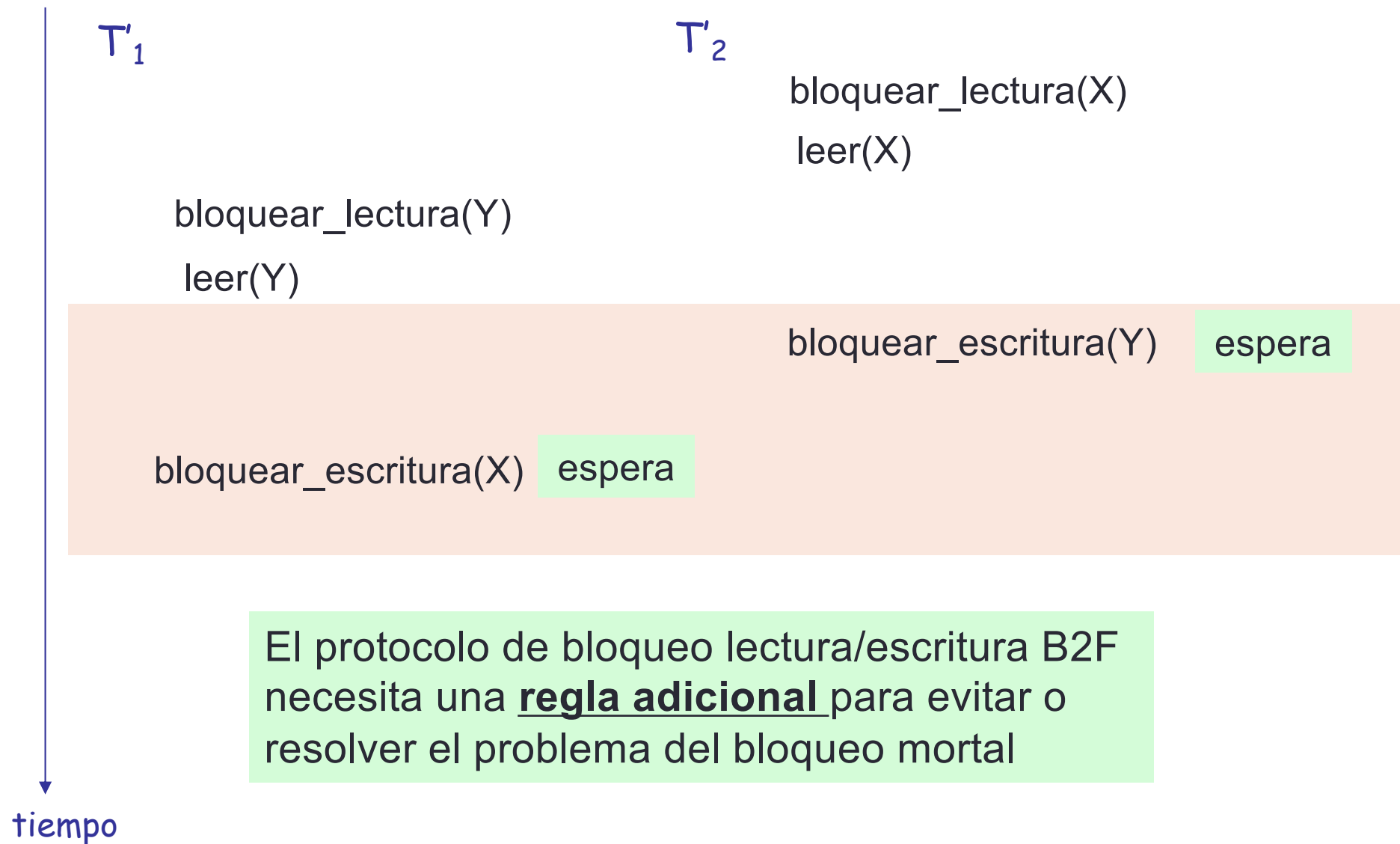
Protocolo de bloqueo B2F

Plan concurrente para T'_1 y T'_2 (con B2F)



Protocolo de bloqueo B2F

Plan concurrente para T'_1 y T'_2 (con B2F)



Protocolo de bloqueo B2F

BD:

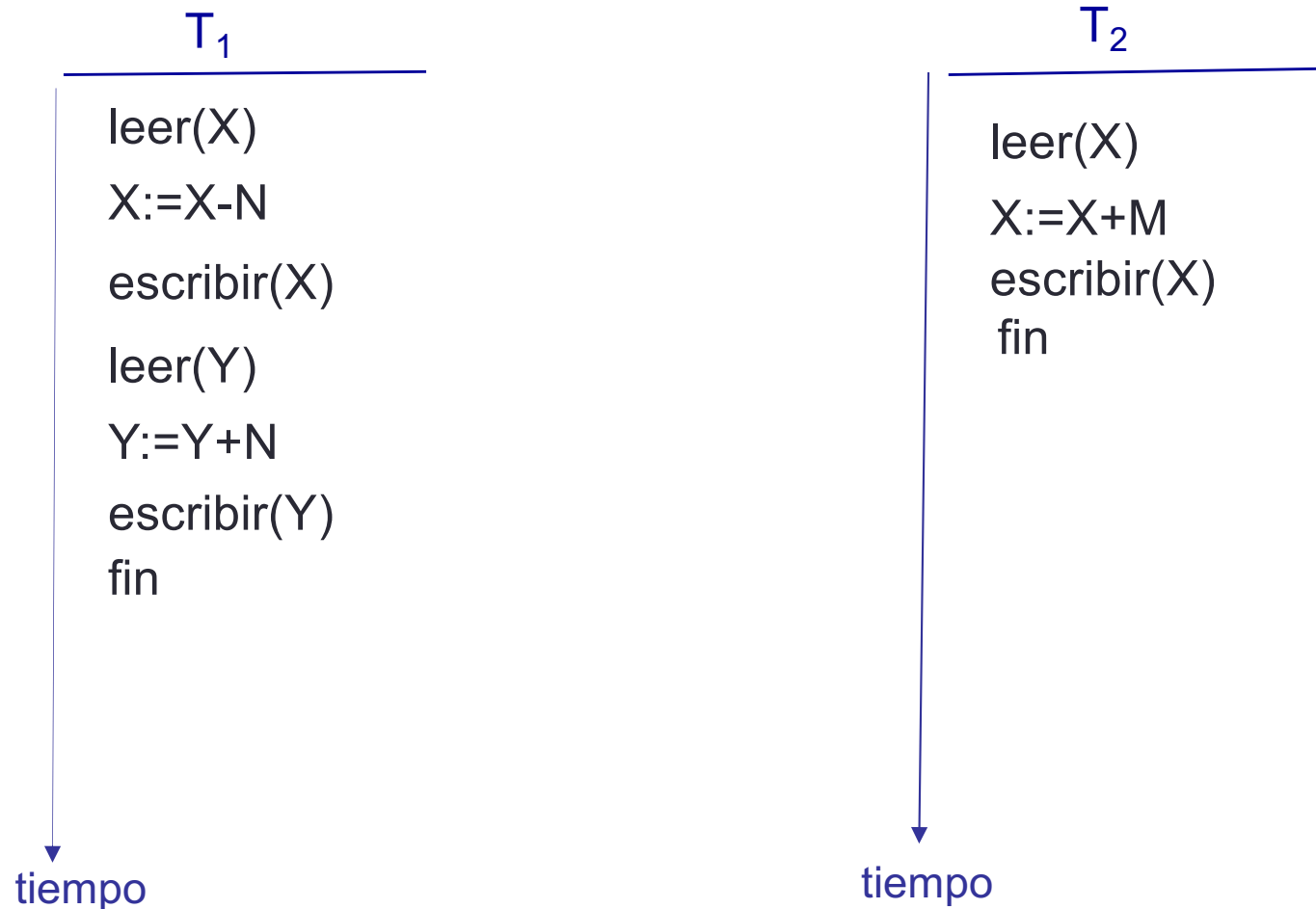
X=90

Y=90

Constantes:

M=2

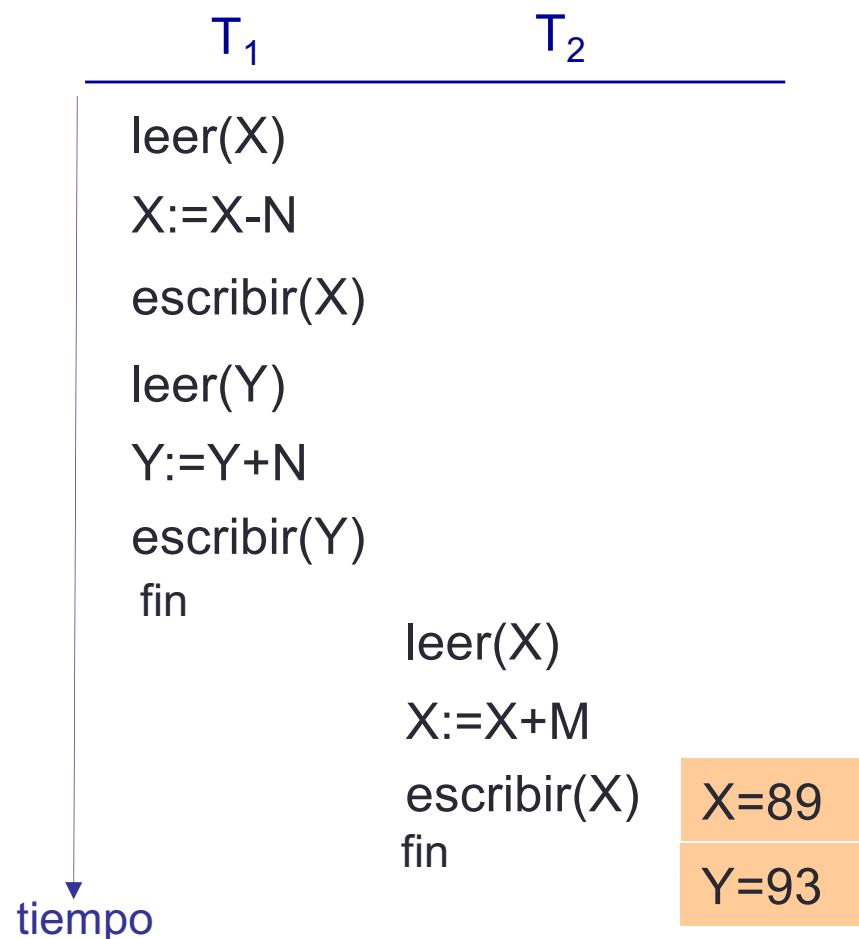
N=3



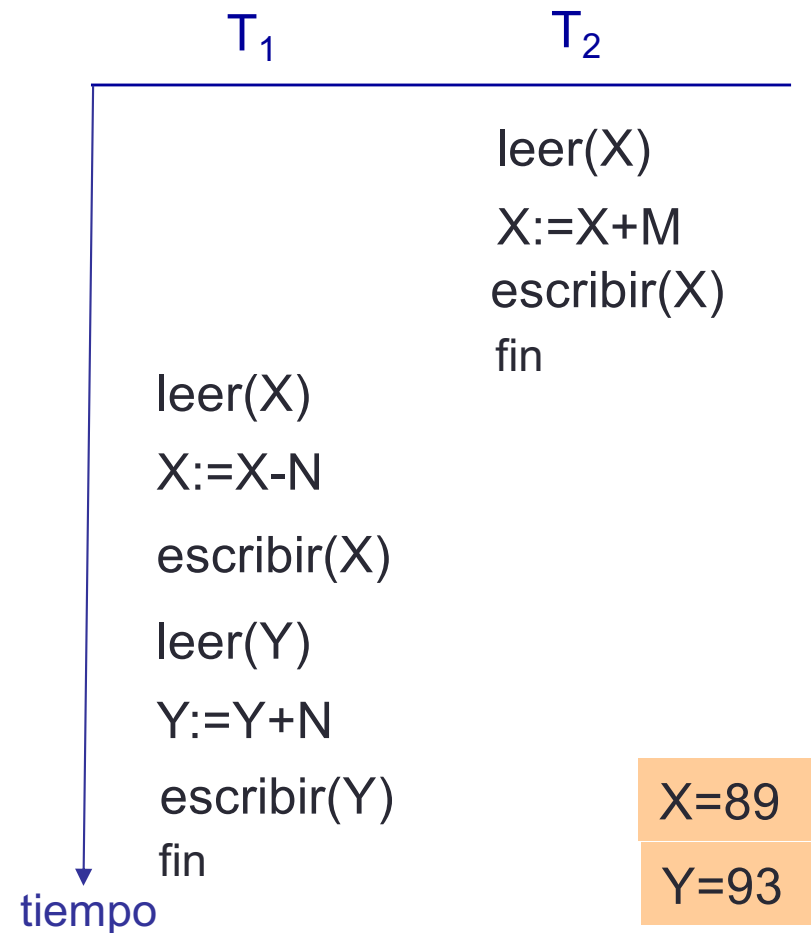
Protocolo de bloqueo B2F

BD: X=90 Y=90

Constantes: M=2 N=3

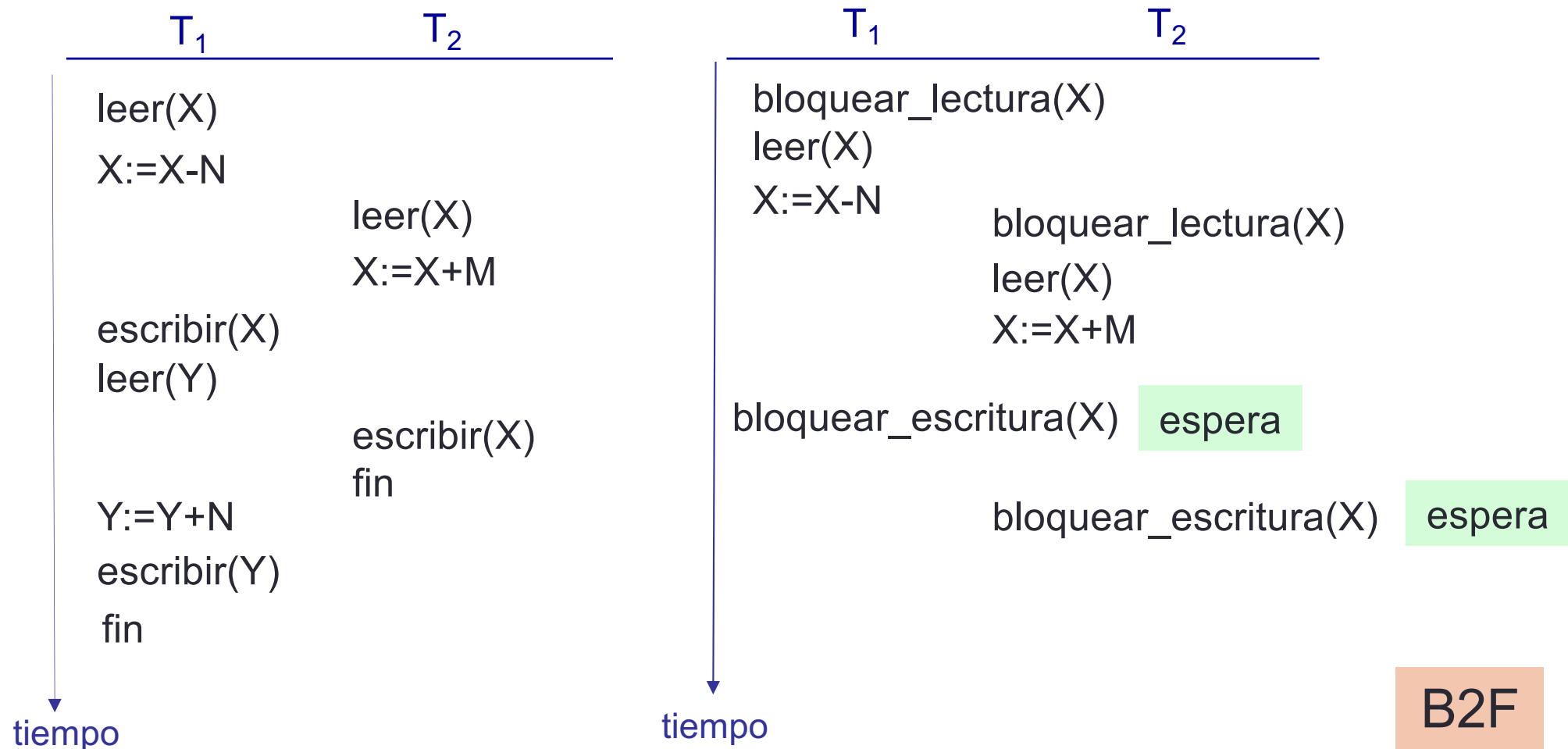


A: Plan en serie para T₁ y T₂



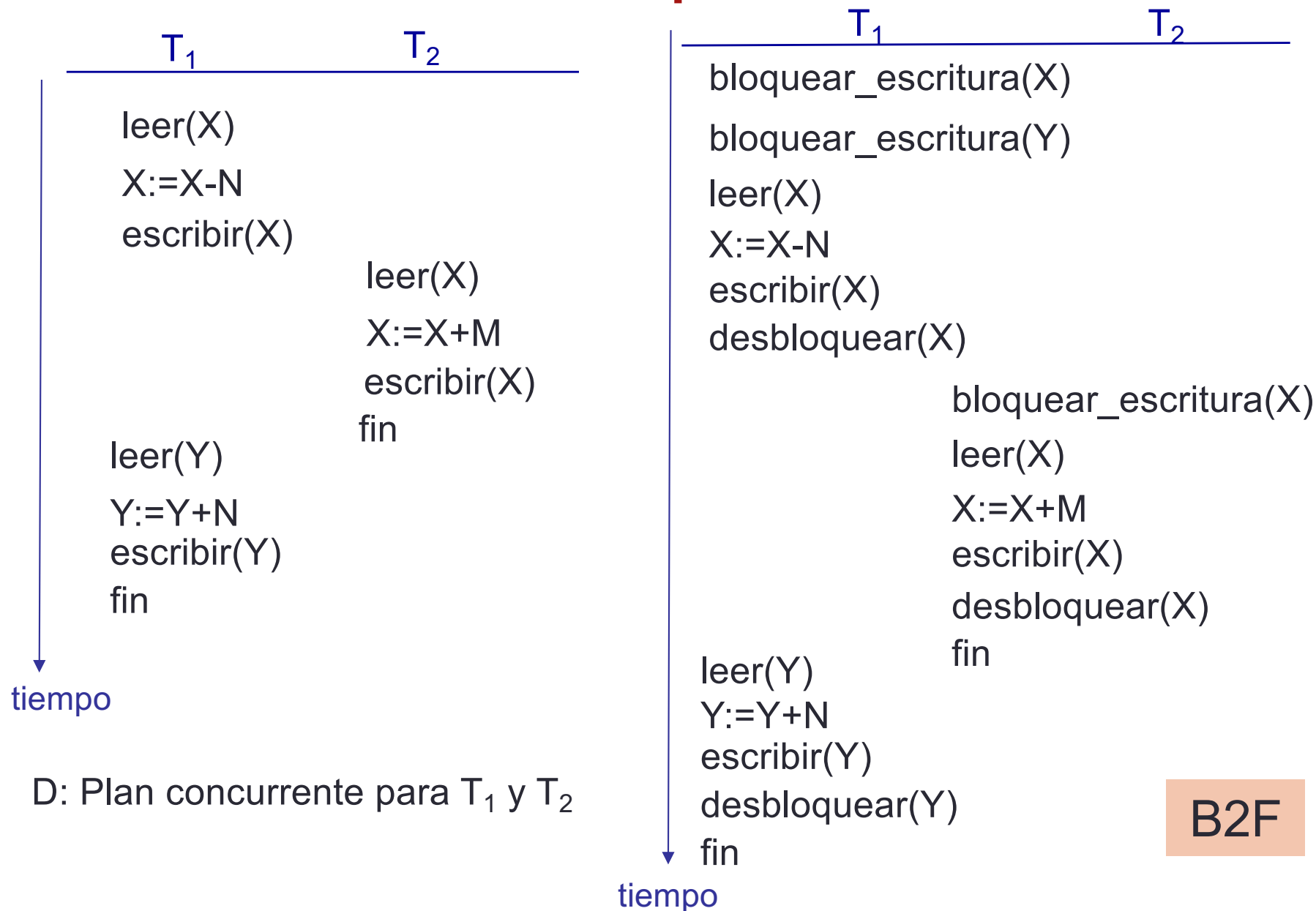
B: Plan en serie para T₁ y T₂

Protocolo de bloqueo B2F



C: Plan concurrente para T_1 y T_2

Protocolo de bloqueo B2F



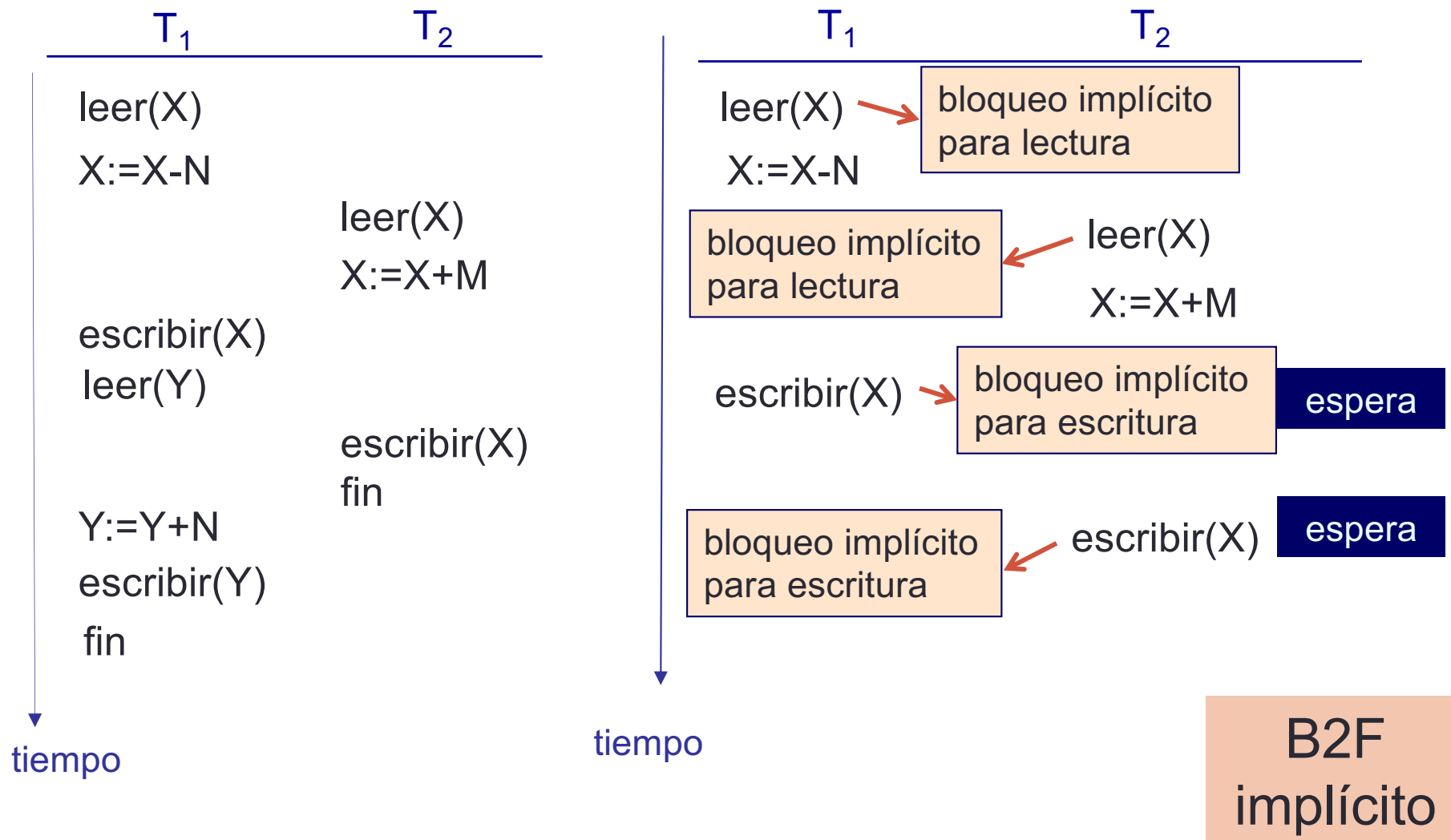
Protocolo de bloqueo B2F

Bloqueo implícito

El subsistema de control de la concurrencia del SGBD se encarga de generar implícitamente los bloqueos de lectura y de escritura y los desbloques sobre elementos de datos:

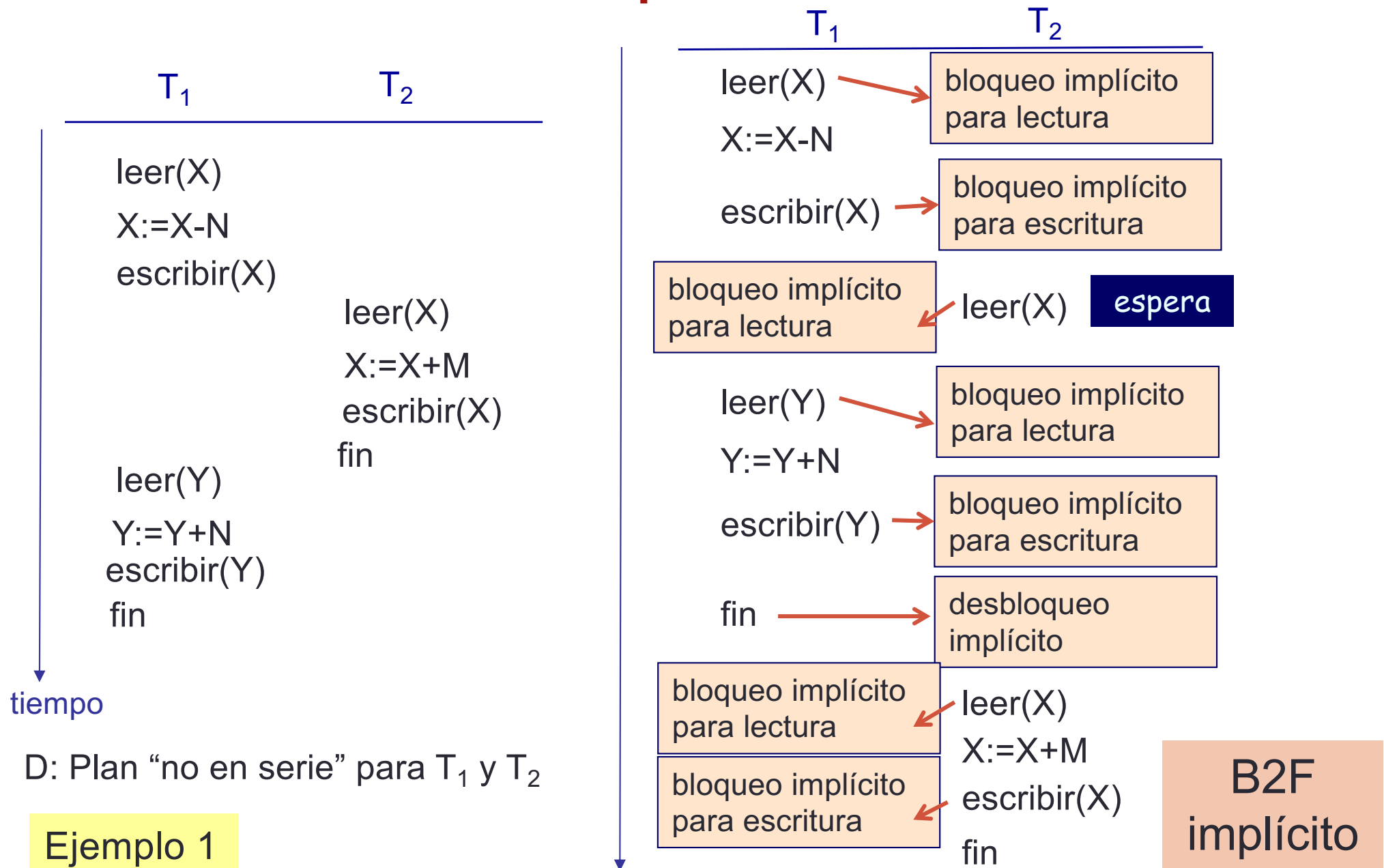
- ✓ leer(X) genera un bloqueo para lectura sobre X
- ✓ escribir(X) genera un bloqueo para escritura sobre X
- ✓ la finalización de la transacción (anulación o confirmación) genera el desbloqueo de todos los elementos de datos bloqueados por la transacción.

Protocolo de bloqueo B2F



C: Plan “no en serie” para T_1 y T_2

Protocolo de bloqueo B2F



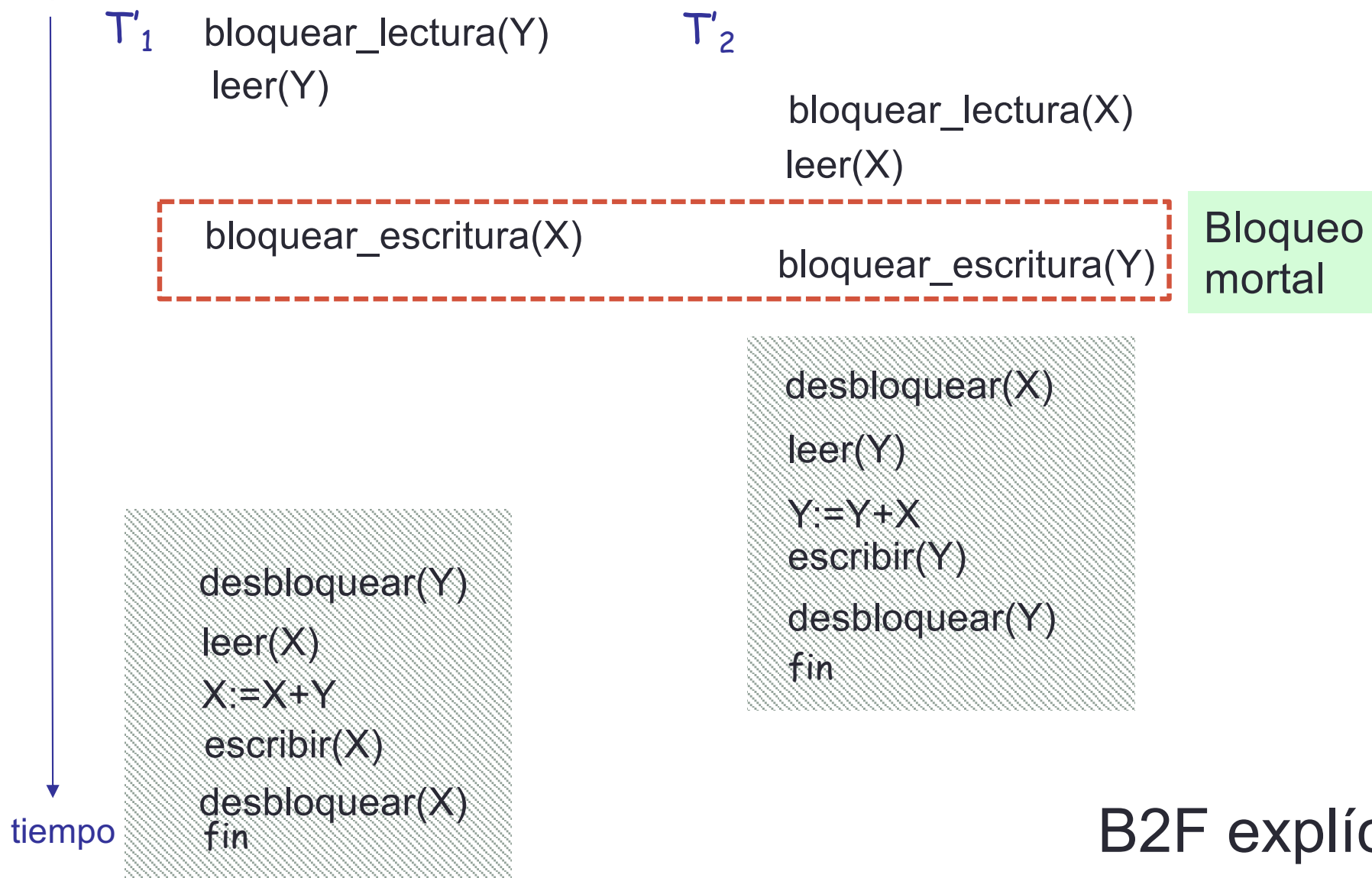
Protocolo de bloqueo B2F

Bloqueo mortal: cada transacción T_i en un conjunto de dos o mas transacciones está esperando un elemento de datos que está bloqueado por otra transacción T_j de dicho conjunto.

Algoritmos de detección del bloqueo mortal: el sistema verifica si el plan está en un estado de bloqueo mortal.

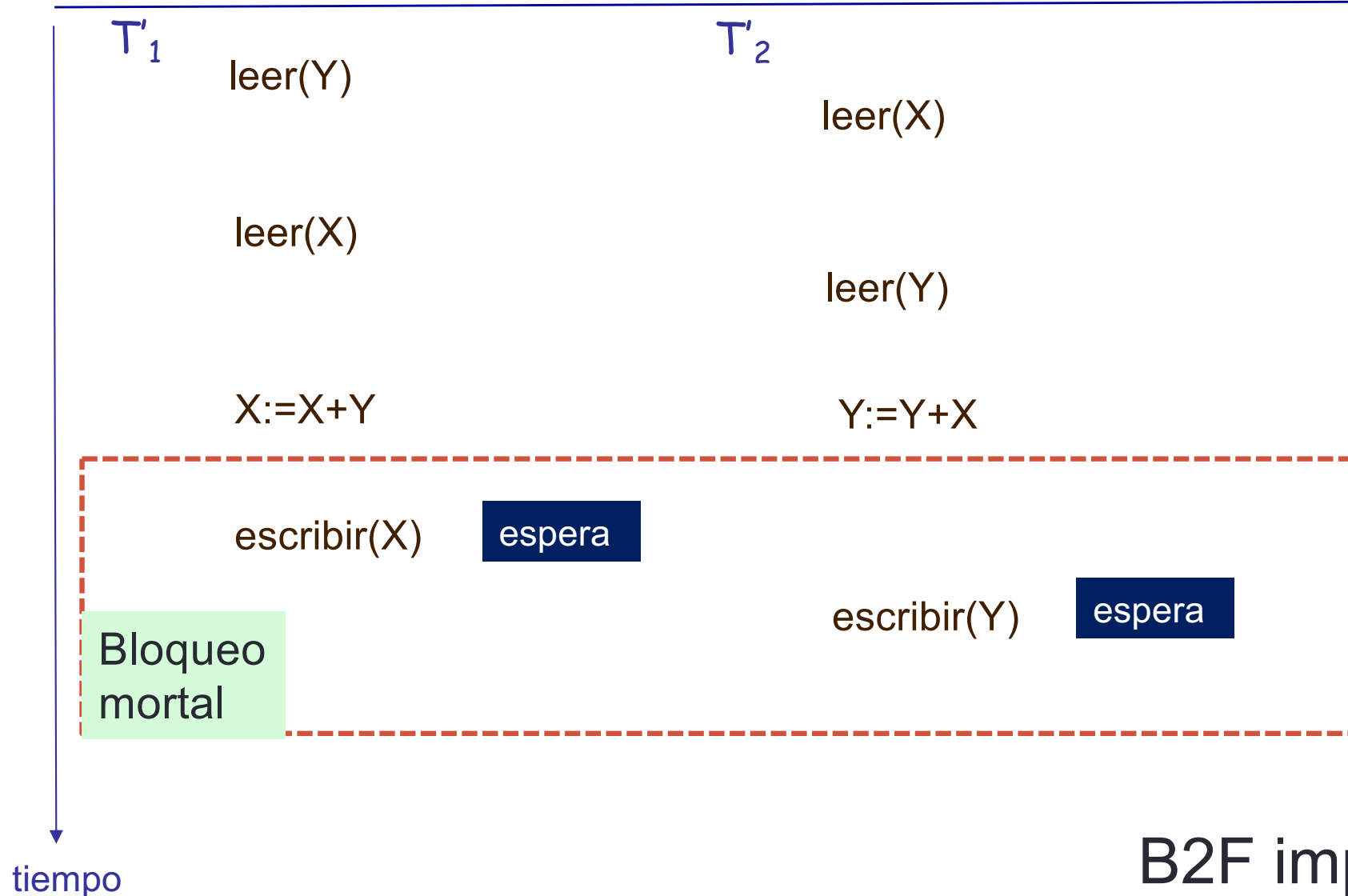
Protocolo de bloqueo B2F

Plan para T'_1 y T'_2 (con bloqueo mortal)



Protocolo de bloqueo B2F

Plan para T'_1 y T'_2 (con bloqueo mortal)



Protocolo de bloqueo B2F

Uso del grafo de espera.

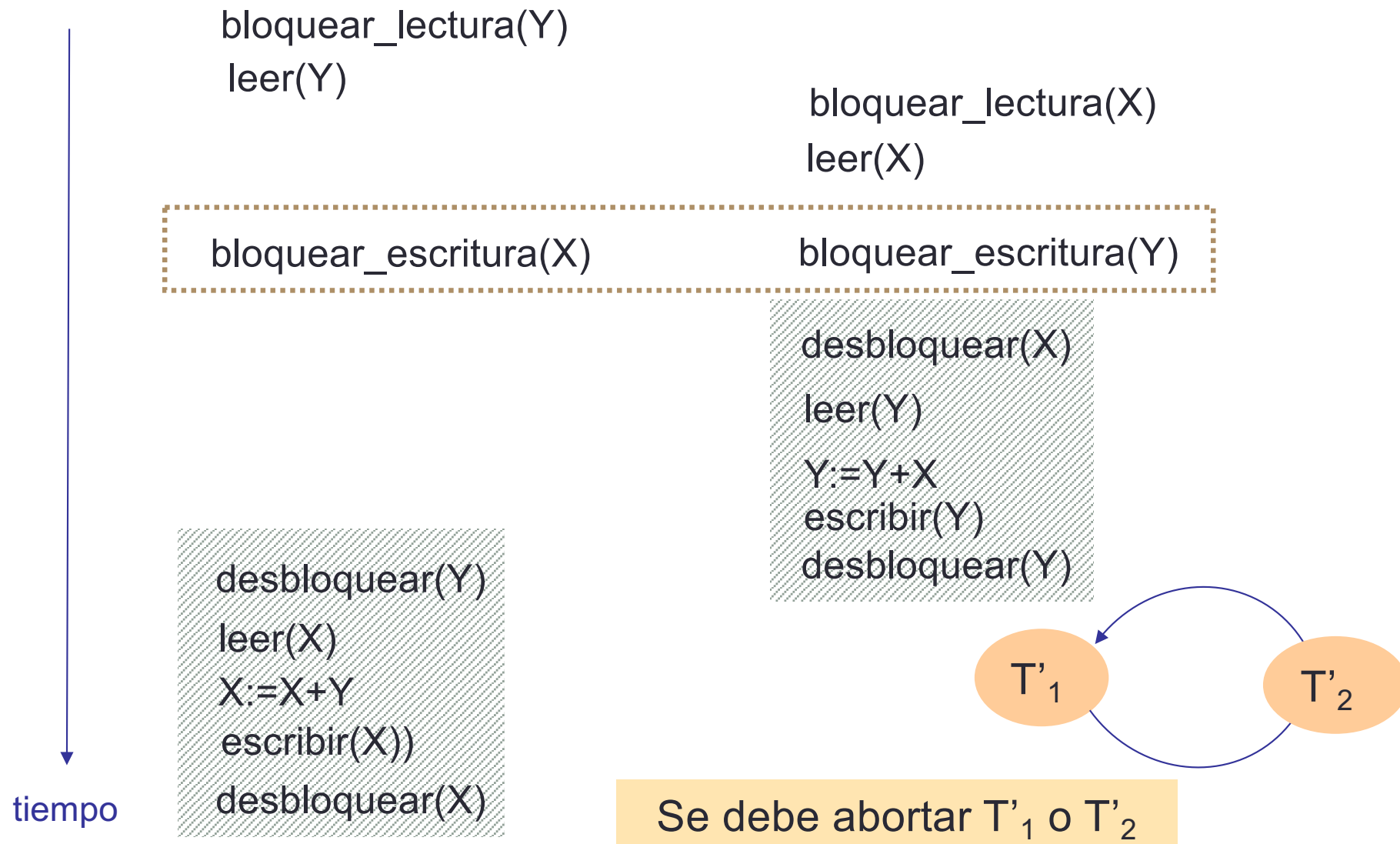
Grafo de espera: grafo dirigido $G=(N,A)$, que consiste en un conjunto de nodos $N=\{T_1, T_2, \dots, T_n\}$ y un conjunto de arcos $A=\{a_1, a_2, \dots, a_m\}$.

- El grafo contiene un nodo por cada transacción que se está ejecutando actualmente.
- El arco $a_i = (T_i \rightarrow T_j)$ se crea si la transacción T_i está esperando bloquear un elemento X que está bloqueado (en modo exclusivo) por la transacción T_j . Cuando T_j libera el elemento X se borra el arco.
- Existe un estado de bloqueo mortal si y sólo si el grafo de espera tiene un ciclo.

Si el sistema está en un estado de bloqueo mortal será preciso abortar alguna de las transacciones que lo están provocando.

Protocolo de bloqueo B2F

Plan para T'_1 y T'_2 (con bloqueo mortal)



Anexo III:

Gestión de un fichero de diario.

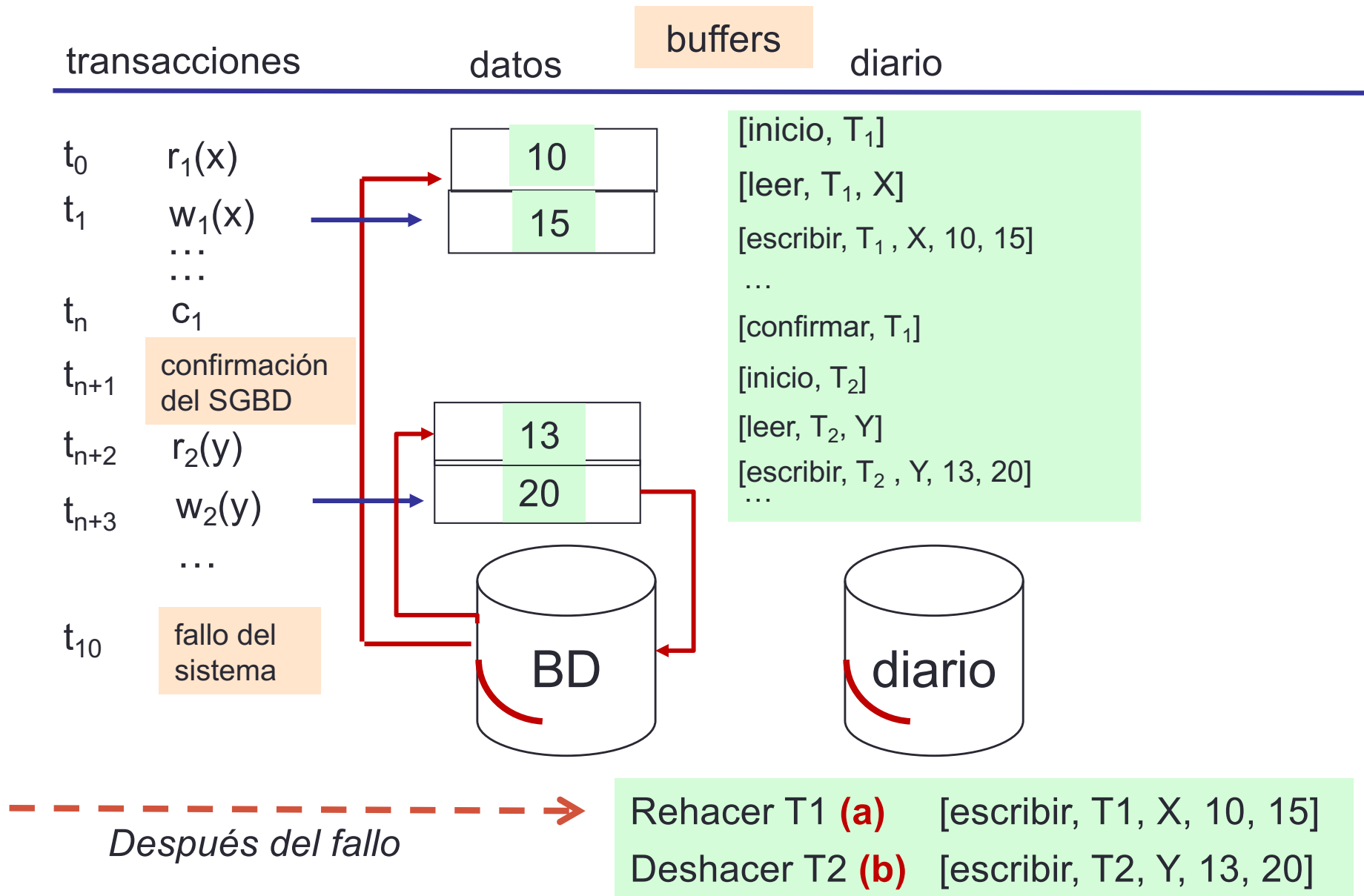
Notación simplificada para representar las operaciones de una transacción T_i :

leer(x): $r_i(x)$

escribir(x): $w_i(x)$

fin (con confirmación): c_i

anulación: a_i

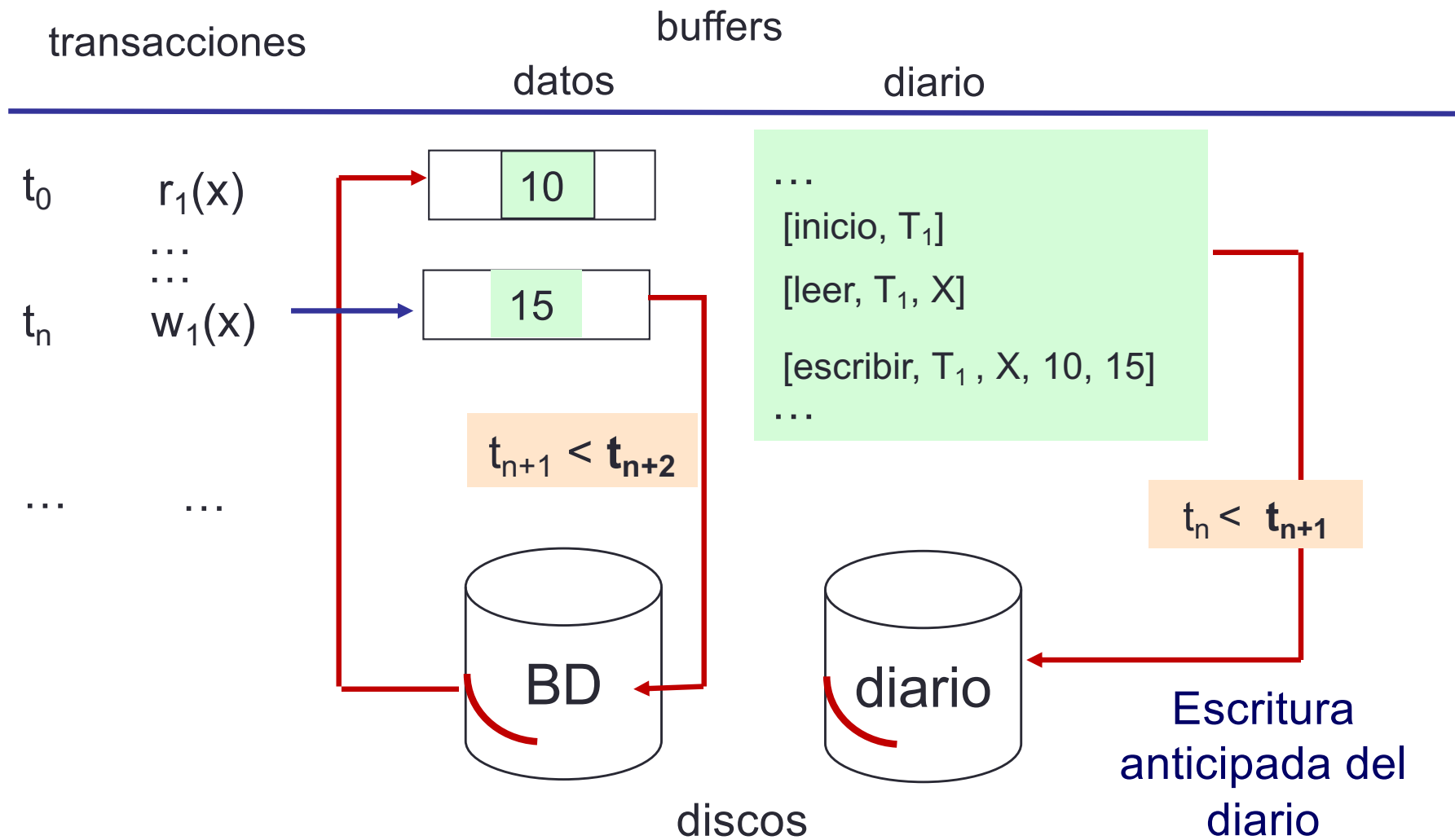


¿Será posible la recuperación?: NO (se ha perdido el buffer de diario)

Diario: principios de gestión.

Escritura anticipada en el diario: las entradas de diario correspondientes a actualizaciones ([escribir, T, X, valor_antes, valor_después]) deben haber sido grabadas en el fichero de diario en disco antes de que los bloques de datos con dichas actualizaciones sean grabados en el disco.

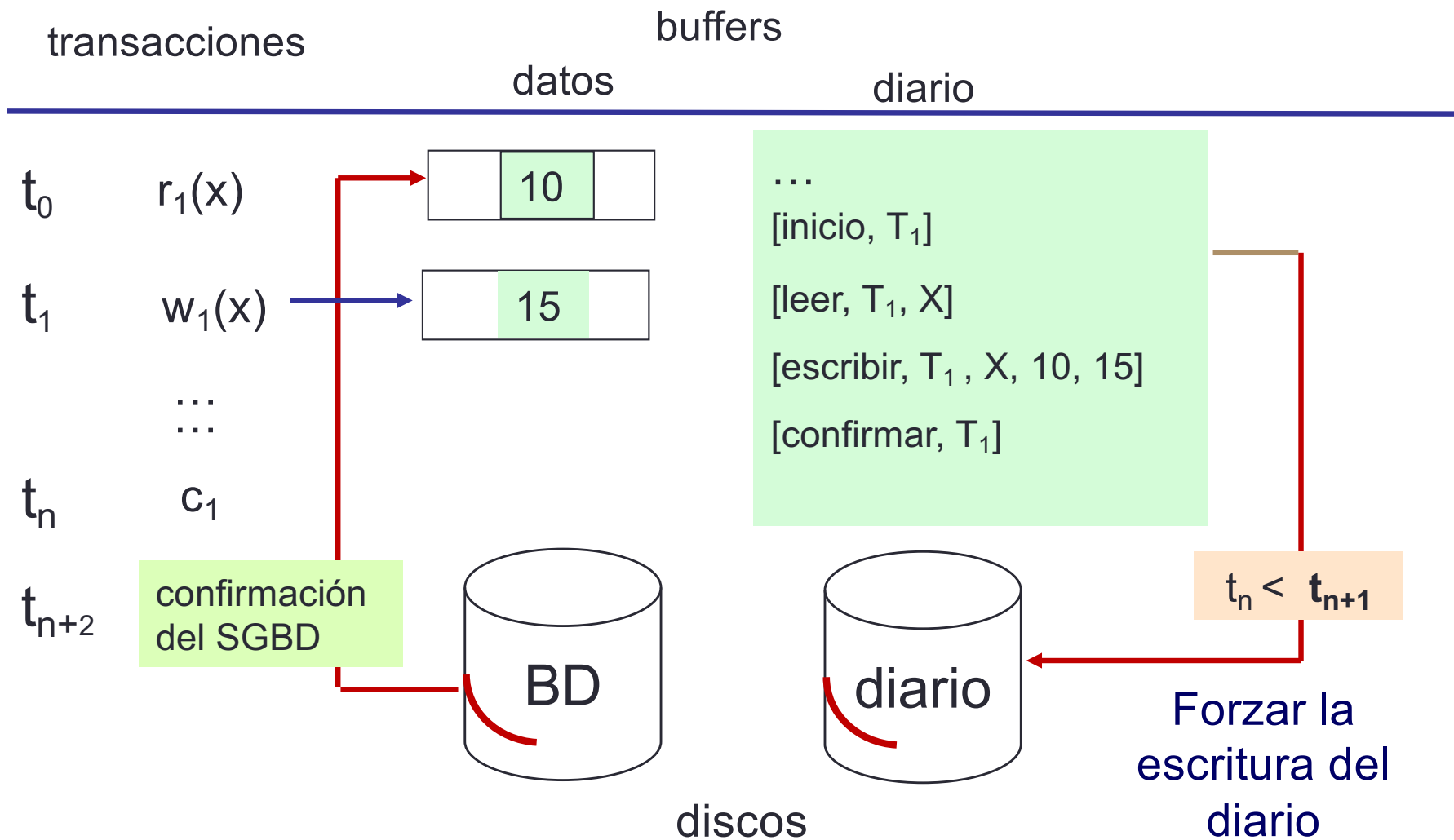
Asegura la recuperación de las transacciones anuladas e interrumpidas.

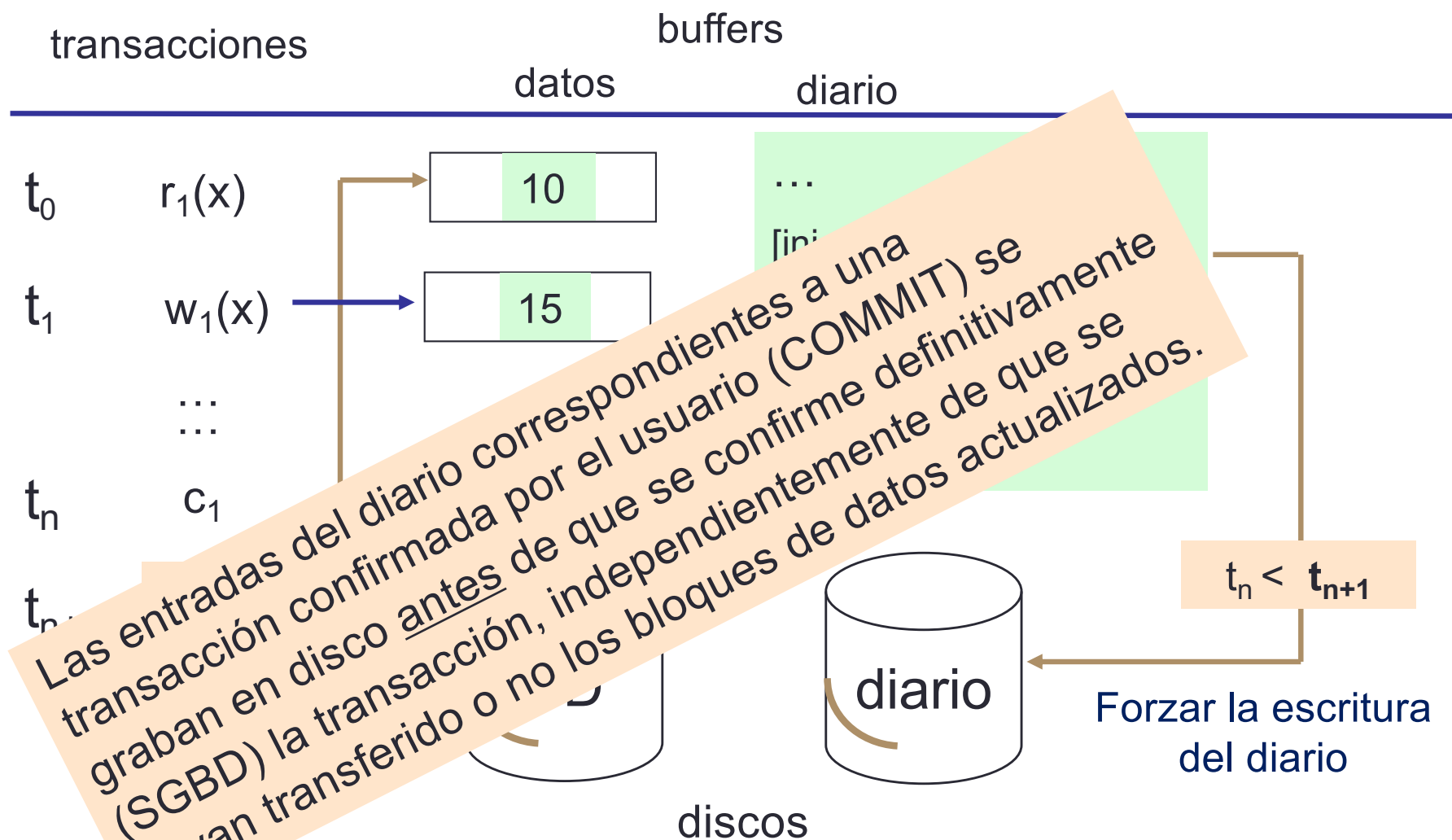


Diario: principios de gestión.

Forzar la escritura del diario: todas las entradas de diario correspondientes a una transacción, deben haber sido grabadas en el fichero de diario en disco antes de que la transacción se confirme (SGBD).

Asegura la recuperación de las transacciones confirmadas.





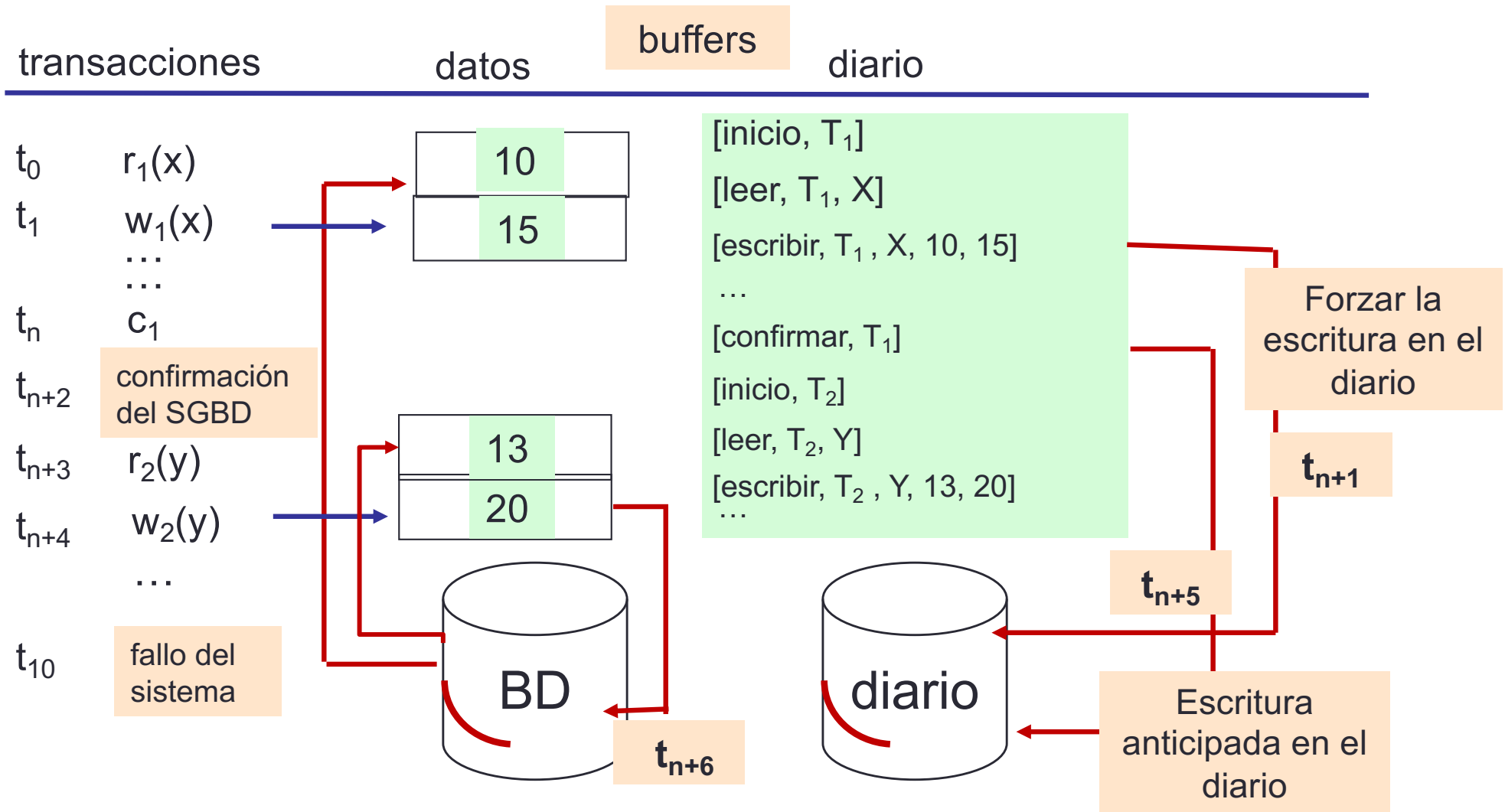
Diario: principios de gestión.

Para que sea posible la recuperación (con actualización en el lugar), en el fichero de diario en disco se deben grabar todas las entradas necesarias para la recuperación de las transacciones.

Protocolo de escritura en el diario

(WAL: Write-Ahead Logging):

1. Antes de que un bloque de datos actualizado por una transacción sea transferido a disco, las entradas de diario correspondientes a dichas actualizaciones deben haber sido grabadas en el fichero de diario en disco. **(escritura anticipada del diario)**
2. Antes de que se confirme definitivamente una transacción (SGBD), todas las entradas de diario correspondientes a dicha transacción deben haber sido grabadas en el fichero de diario en disco. **(forzar la escritura del diario).**



¿Será posible la recuperación?: SÍ (el buffer de diario ha sido salvado a disco en los momentos oportunos)

Diario: principios de gestión.

Puntos de control en el diario: marcas que el SGBD registra en el diario indicando que en ese momento todas las actualizaciones de transacciones confirmadas, es decir con una entrada en el diario [confirmar, T], han sido grabadas en disco.

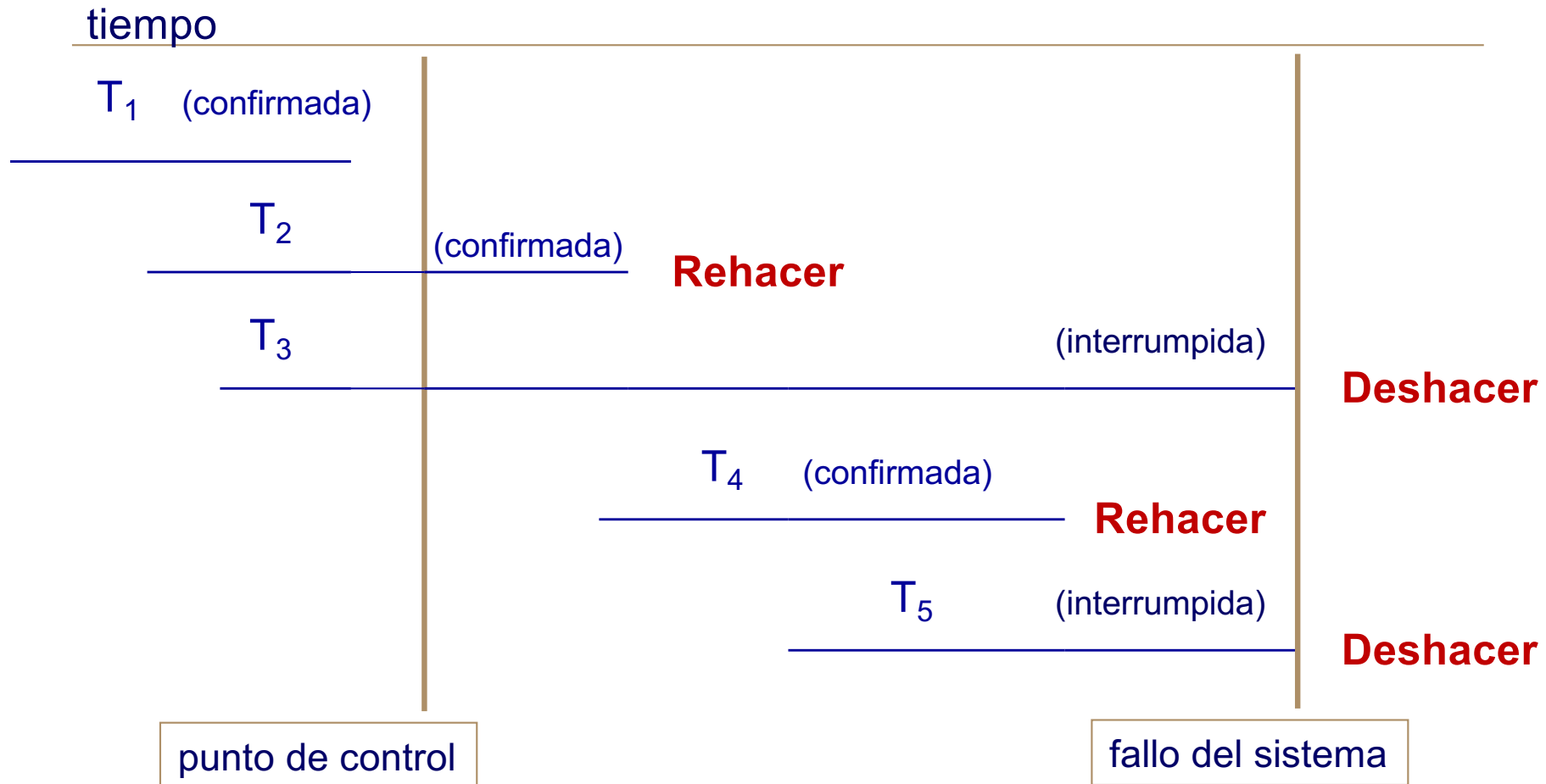
Los puntos de control en el diario:

- ✓ Simplifican el proceso de recuperación: las transacciones con una entrada de confirmación anterior al último punto de control no se deben rehacer durante la recuperación.
- ✓ El SGBD decide la frecuencia con la que se registran los puntos de control.

Registrar un punto de control en el diario significa:

- ✓ Suspenden temporalmente las transacciones activas.
- ✓ Transferir a disco todos los bloques actualizados por transacciones confirmadas después del último punto de control. (implica una escritura anticipada del diario)
- ✓ Registrar el punto de control en el buffer de diario y forzar la escritura del diario.
- ✓ Reactivar las transacciones suspendidas.

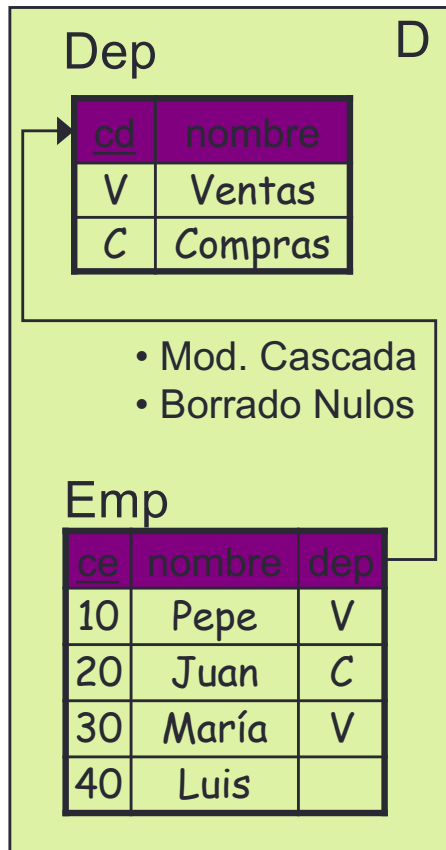
Ejemplo:



Anexo IV

Algoritmo de ejecución de reglas de actividad (disparadores) en SQL3

Ejemplo 1: modelo de ejecución de reglas en SQL3



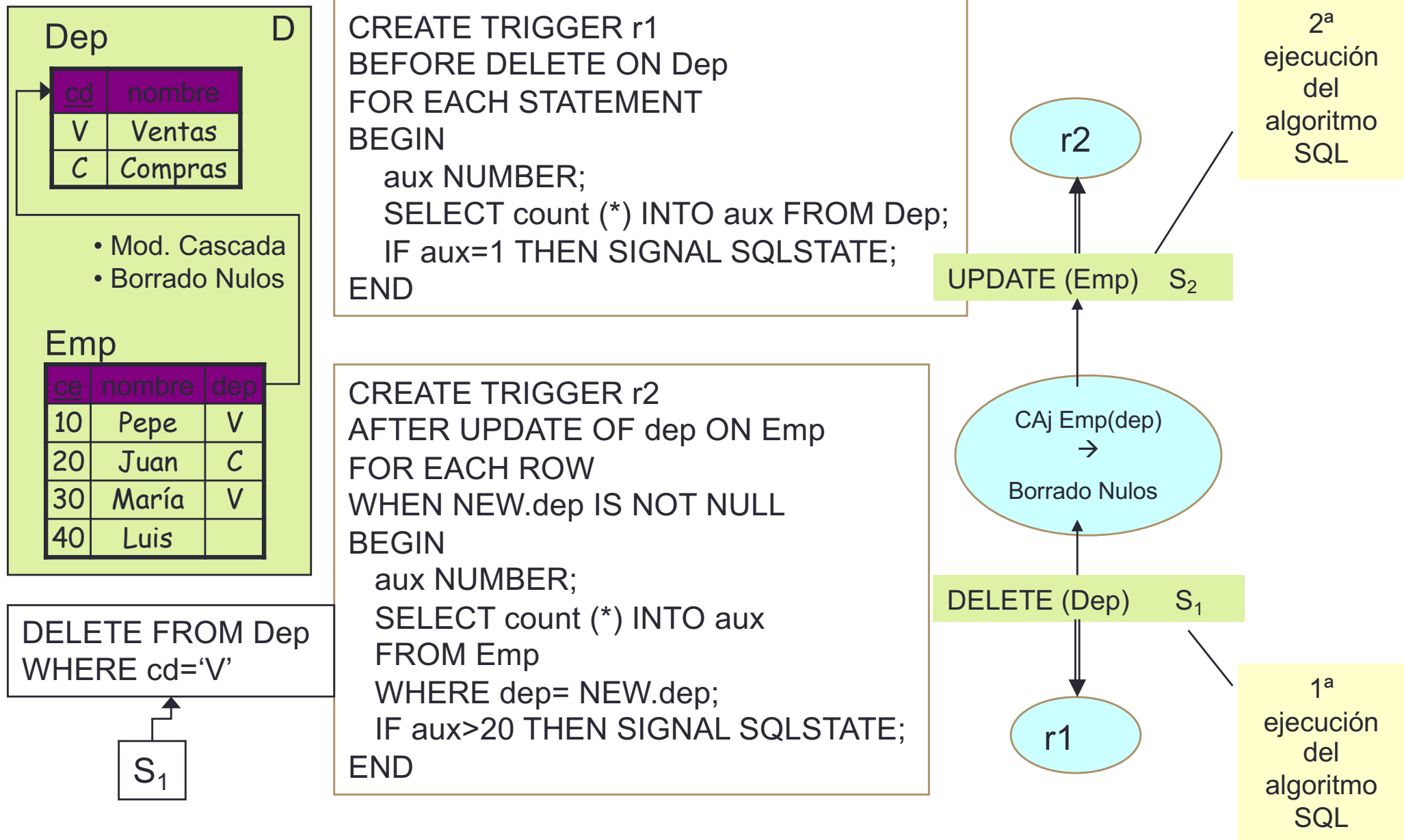
```
CREATE TRIGGER r1
BEFORE DELETE ON Dep
FOR EACH STATEMENT
BEGIN
    aux NUMBER;
    SELECT count (*) INTO aux FROM Dep;
    IF aux=1 THEN SIGNAL SQLSTATE;
END
```

No se puede
borrar un
departamento si
éste es el último

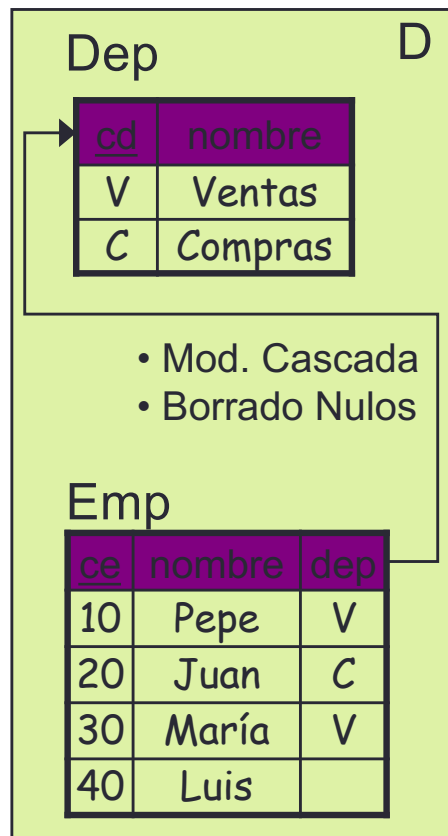
```
CREATE TRIGGER r2
AFTER UPDATE OF dep ON Emp
FOR EACH ROW
WHEN NEW.dep IS NOT NULL
BEGIN
    aux NUMBER;
    SELECT count (*) INTO aux
    FROM Emp
    WHERE dep= NEW.dep;
    IF aux>20 THEN SIGNAL SQLSTATE;
END
```

En un
departamento
no puede haber
más de 20
empleados

Ejemplo 1: modelo de ejecución de reglas en SQL3



Ejemplo 1: modelo de ejecución de reglas en SQL3



Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_1 (inicialización de parámetros)

OLD

V	Ventas
---	--------

NEW

--	--

OLD_TABLE

V	Ventas
---	--------

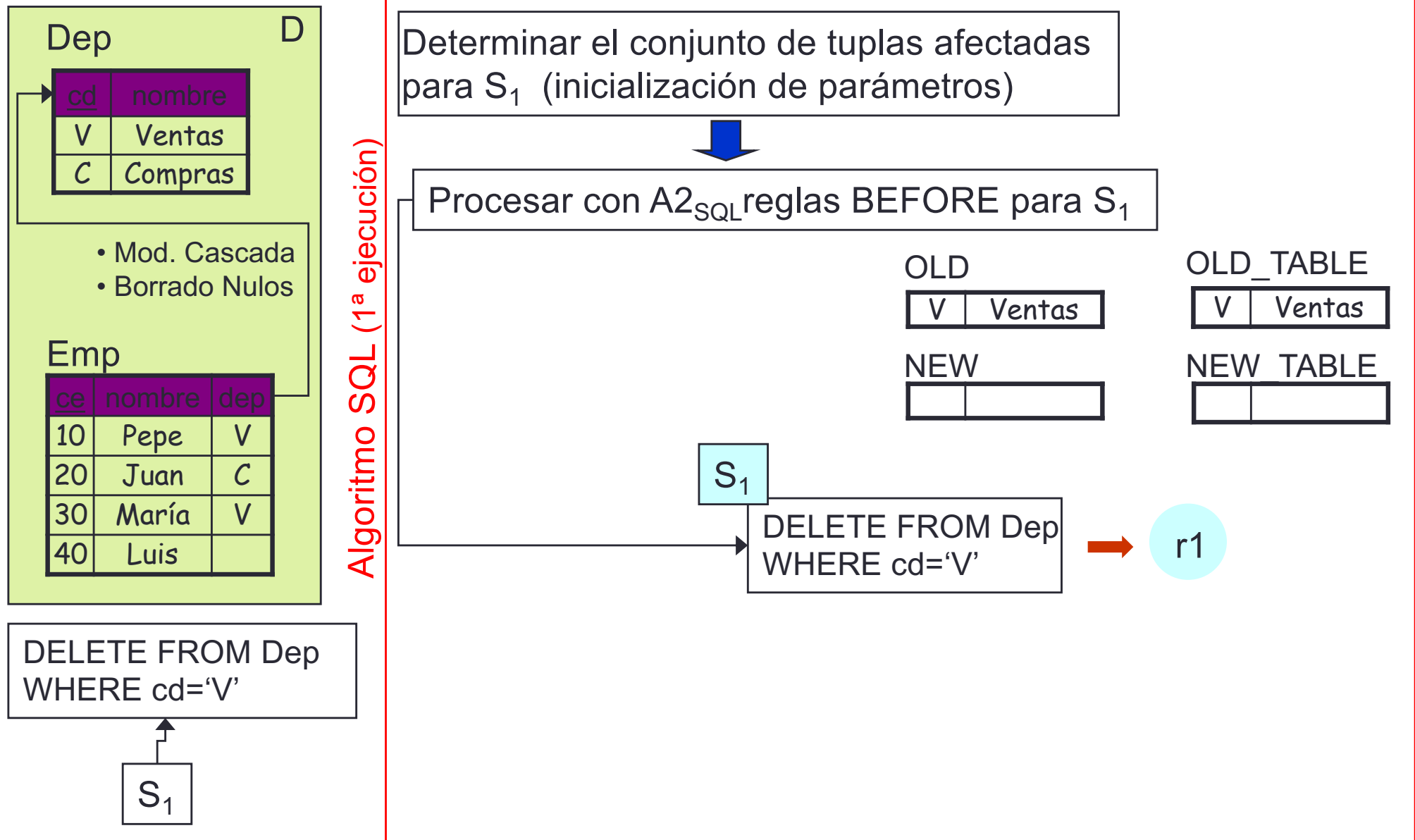
NEW TABLE

--	--

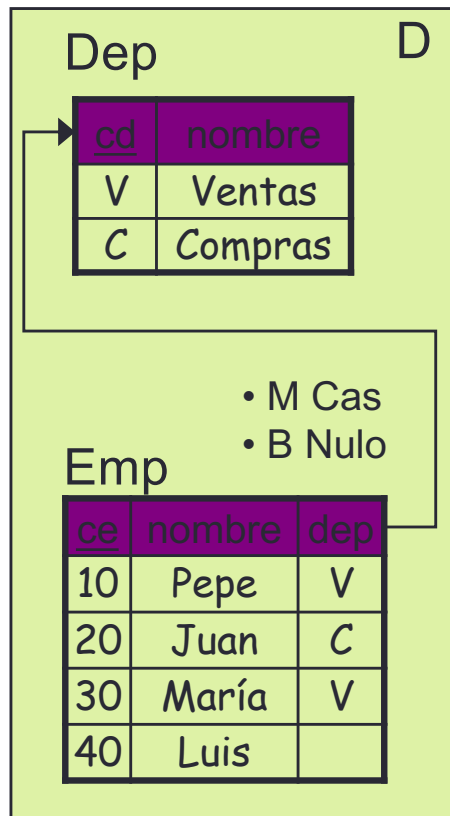
DELETE FROM Dep
WHERE cd='V'

S_1

Ejemplo 1: modelo de ejecución de reglas en SQL3



Ejemplo 1: modelo de ejecución de reglas en SQL3



A2SQL (1ª ejecución)

DELETE FROM Dep
WHERE cd='V'

S₁

Reglas BEFORE para S₁

mientras {r1} ≠ ∅

1. seleccionar una regla activada: r1

2. si r1 es de tipo FOR EACH STATEMENT

si la condición de r1 es cierta

ejecutar la acción de r1 con **Algoritmo SQL**

sino

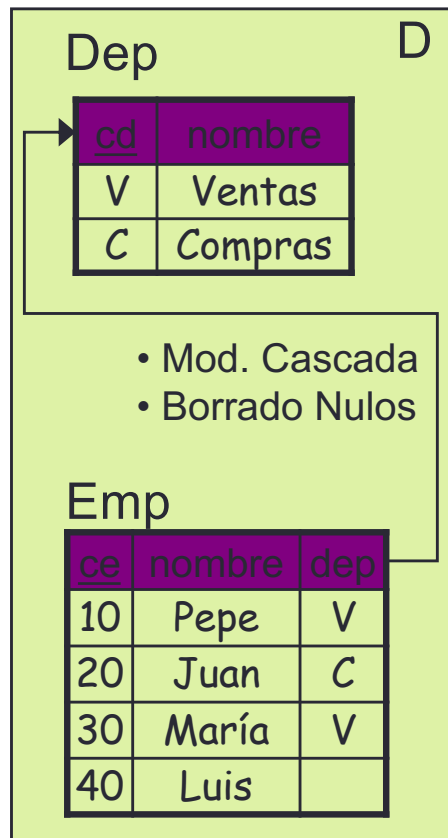
...

fin_mientras

BEGIN
aux NUMBER;
SELECT count(*) INTO aux
FROM Dep;
IF aux=1 THEN SIGNAL SQLSTATE;
END

2

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2_{SQL} (1ª ejecución)

DELETE FROM Dep
WHERE cd='V'

S₁

Reglas BEFORE para S₁

mientras $\{\} \neq \emptyset$

1. seleccionar una regla activada:

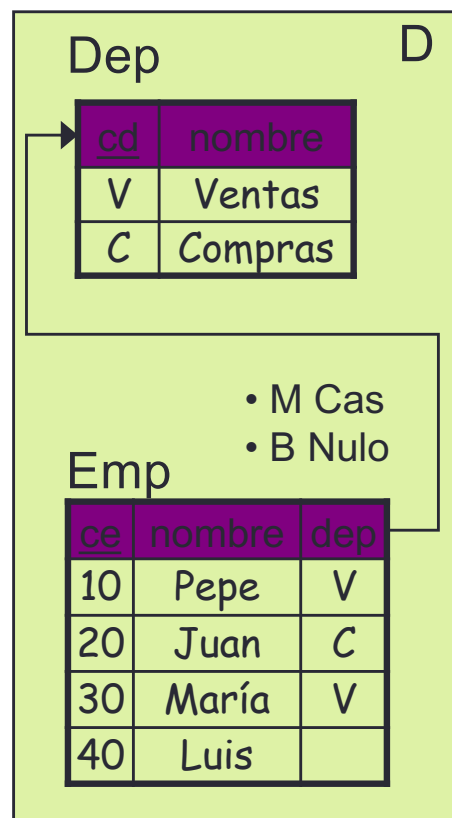
2. si

sino

fin_mientras

FIN de la 1ª ejecución
del Algoritmo A2_{SQL}

Ejemplo 1: modelo de ejecución de reglas en SQL3



DELETE FROM Dep
WHERE cd='V'

S₁

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₁



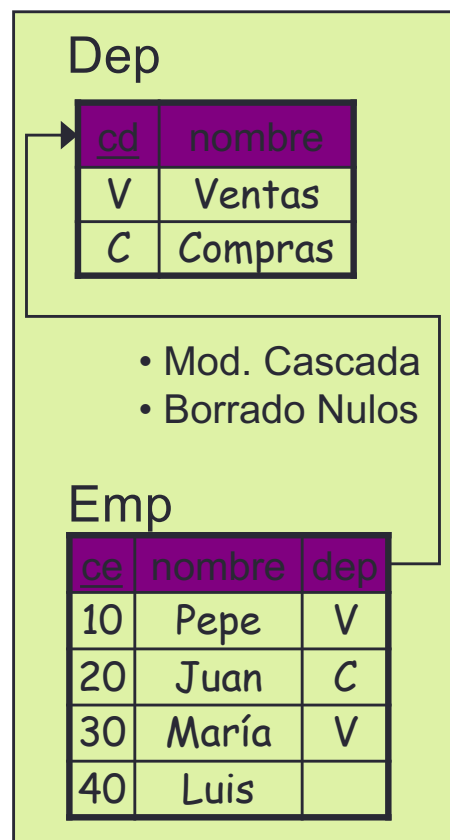
Ejecutar S₁

DELETE FROM Dep
WHERE cd='V'

Dep

cd	nombre
V	Ventas
C	Compras

Ejemplo 1: modelo de ejecución de reglas en SQL3



DELETE FROM Dep
WHERE cd='V'

S₁

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)

Procesar con A2_{SQL} reglas BEFORE para S₁

Ejecutar S₁

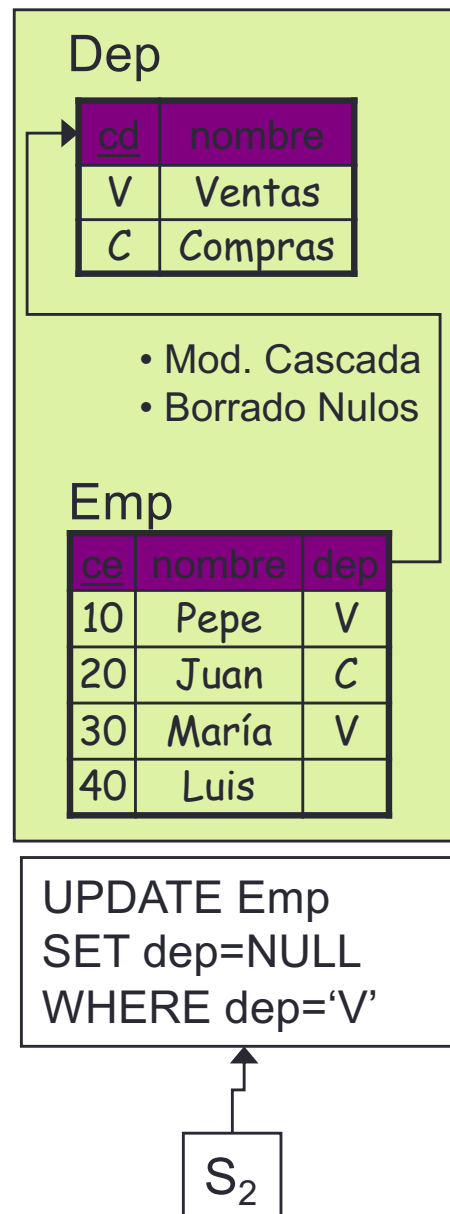
Comprobación de restricciones relevantes para S₁

- restricciones RESTRIC ERROR
- integridad referencial CASCADE y SET NULL

UPDATE Emp
SET dep=NULL
WHERE dep='V'

S₂

Ejemplo 1: modelo de ejecución de reglas en SQL3



Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_2 (inicialización de parámetros)

OLD

10	Pepe	V
----	------	---

OLD_TABLE

10	Pepe	V
30	María	V

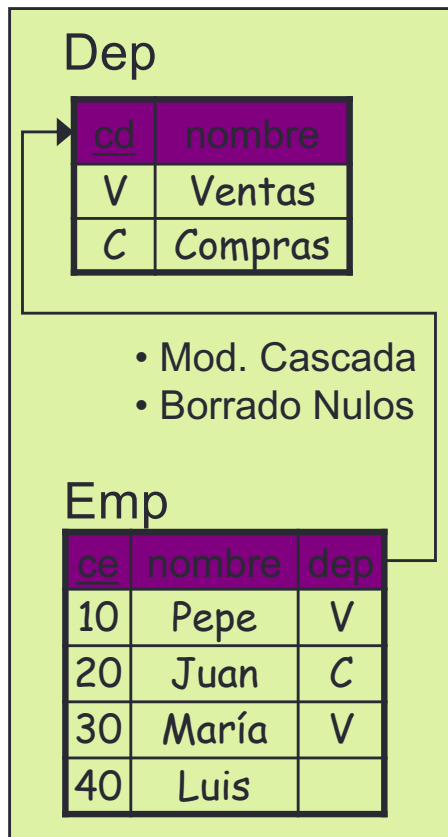
NEW

10	Pepe	
----	------	--

NEW_TABLE

10	Pepe	
30	María	

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S₂

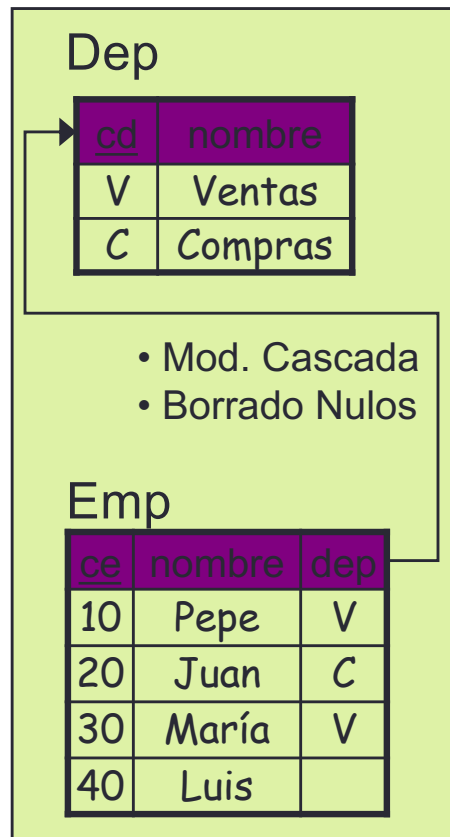
Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₂ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₂

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S₂

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₂ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₂



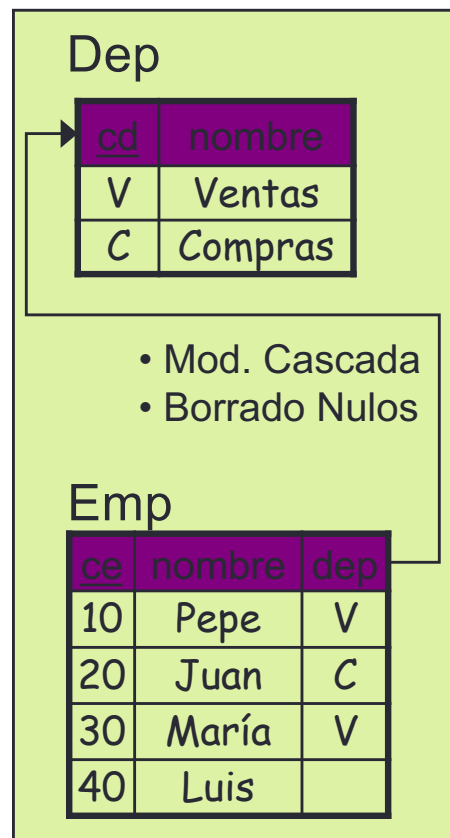
Ejecutar S₂

```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

Emp

<u>ce</u>	nombre	dep
10	Pepe	
20	Juan	C
30	María	
40	Luis	

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S_2

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_2 (inicialización de parámetros)



Procesar con $A2_{SQL}$ reglas BEFORE para S_2



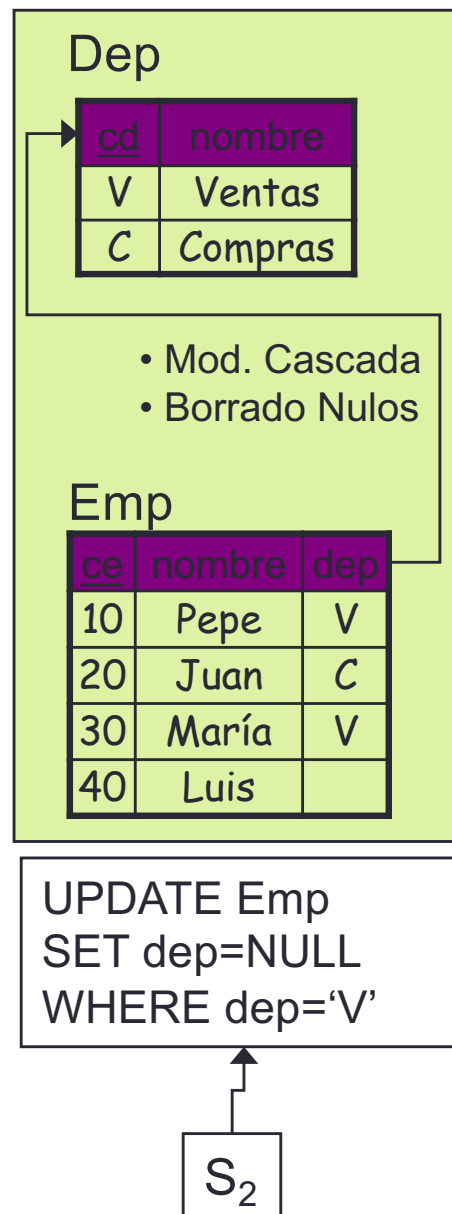
Ejecutar S_2



Comprobación de restricciones relevantes para S_2

- restricciones RESTRIC **ERROR**
- integridad referencial **CASCADE** y **SET NULL**

Ejemplo 1: modelo de ejecución de reglas en SQL3



Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₂ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₂



Ejecutar S₂

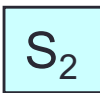


Comprobación de restricciones relevantes para S₂

- restricciones RESTRIC **ERROR**
- integridad referencial **CASCADE** y SET NULL



Procesar con A2_{SQL} reglas AFTER para S₁, S₂

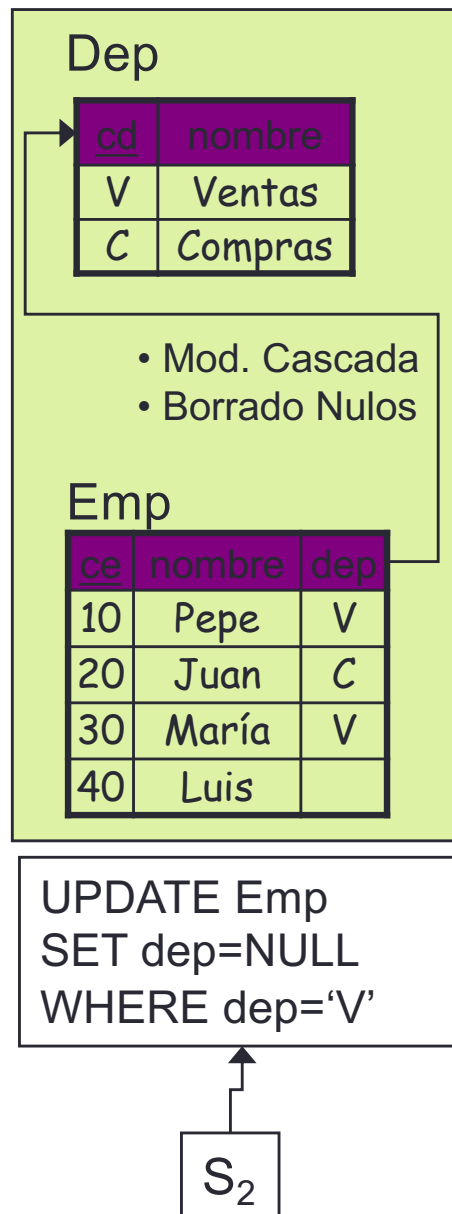


UPDATE Emp
SET dep=NULL
WHERE dep='V'



r2

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

Reglas AFTER para S_1 y S_2

mientras $\{r2\} \neq \emptyset$

1. seleccionar una regla activada: $r2$

2. si $r2$ es de tipo FOR EACH STATEMENT

...

sino

- para cada tupla afectada por el evento de $r2$

- evaluar la condición de $r2$

- si la condición es cierta

ejecutar la acción de $r2$ con **Algoritmo SQL**

fin_mientras

10 Pepe V

OLD

10	Pepe	V
----	------	---

OLD_TABLE

10	Pepe	V
30	María	V

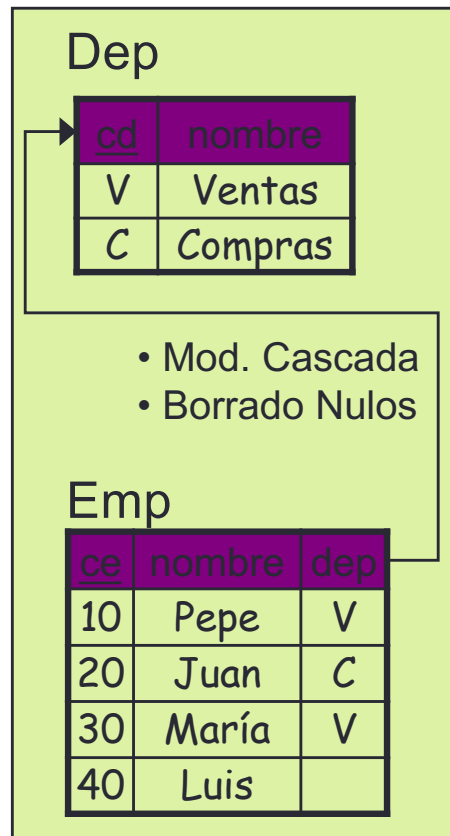
NEW

10	Pepe	
----	------	--

NEW_TABLE

10	Pepe	
30	María	

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S₂

A2_{SQL} (2ª ejecución)

Reglas AFTER para S₁ y S₂

mientras {r2} ≠ ∅

1. seleccionar una regla activada: r2
2. si r2 es de tipo FOR EACH STATEMENT
sino

- para cada tupla afectada por el evento de r2
 - evaluar la condición de r2
 - si la condición es cierta

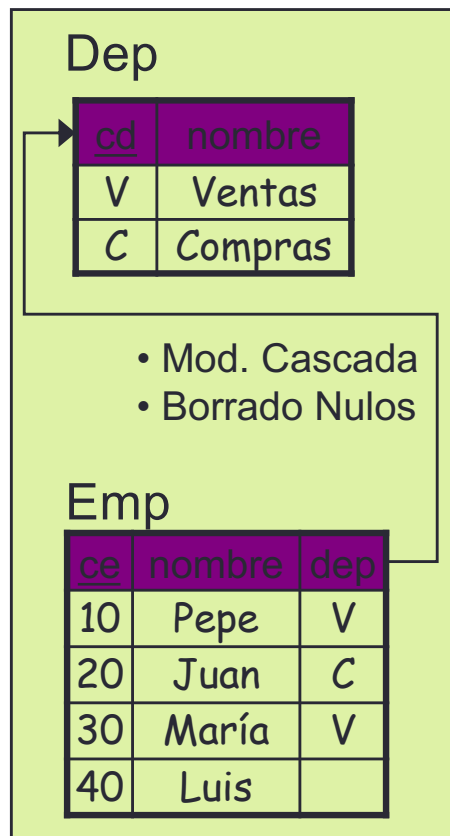
ejecutar la acción de r2 con **Algoritmo SQL**

fin_mientras

10 Pepe NULL

WHEN NEW.dep IS NOT NULL

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S₂

A2_{SQL} (2ª ejecución)

Reglas AFTER para S₁ y S₂

mientras {r2} ≠ ∅

1. seleccionar una regla activada: r2
2. si r2 es de tipo FOR EACH STATEMENT
sino

- para cada tupla afectada por el evento de r2
- evaluar la condición de r2
- si la condición es cierta

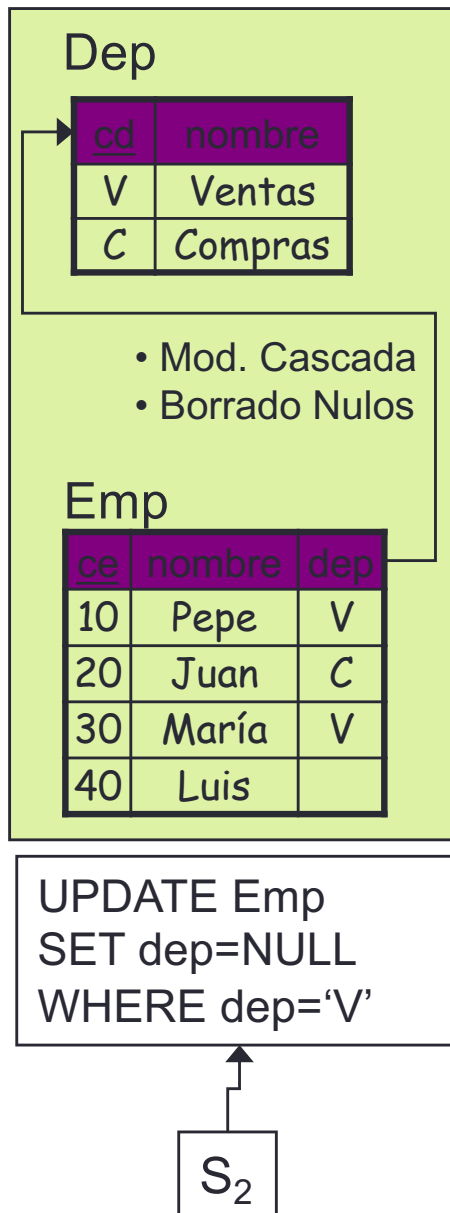
ejecutar la acción de r2 con **Algoritmo SQL**

fin_mientras

10 Pepe NULL

WHEN **NULL** IS NOT NULL

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

Reglas AFTER para S₁ y S₂

mientras {r2} ≠ ∅

1. seleccionar una regla activada: r2

2. si r2 es de tipo FOR EACH STATEMENT

Si

...

sino

- para cada tupla afectada por el evento de r2

- evaluar la condición de r2

- si la condición es cierta

ejecutar la acción de r2 con **Algoritmo SQL**

fin_mientras

30 María V

OLD

30	María	V
----	-------	---

OLD_TABLE

10	Pepe	V
30	María	V

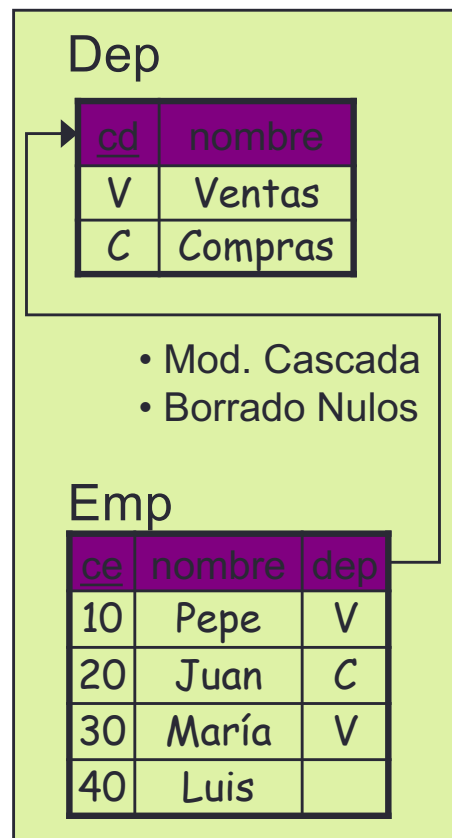
NEW

30	María	
----	-------	--

NEW_TABLE

10	Pepe	
30	María	

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2SQL (2ª ejecución)

```
UPDATE Emp
SET dep=NULL
WHERE dep='V'
```

S₂

Reglas AFTER para S₁ y S₂

mientras {r2} ≠ ∅

1. seleccionar una regla activada: r2

2. si r2 es de tipo FOR EACH STATEMENT

Si

...

sino

- para cada tupla afectada por el evento de r2

- evaluar la condición de r2

- si la condición es cierta

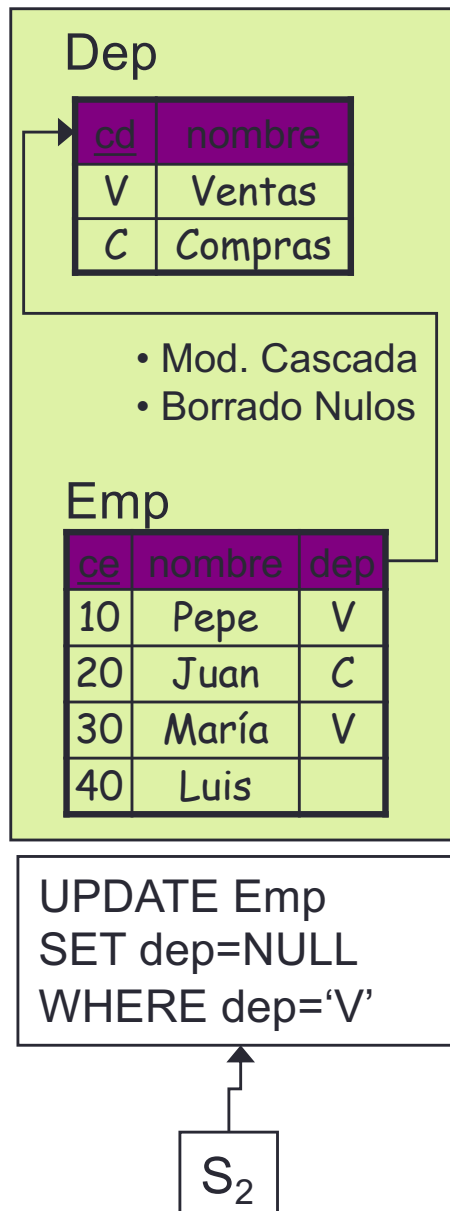
ejecutar la acción de r2 con **Algoritmo SQL**

fin_mientras

30 María NULL

WHEN NEW.dep IS NOT NULL

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

Reglas AFTER para S₁ y S₂

mientras {r2} ≠ ∅

1. seleccionar una regla activada: r2

2. si r2 es de tipo FOR EACH STATEMENT

Si

...

sino

- para cada tupla afectada por el evento de r2

- evaluar la condición de r2 ——— 30 María NULL

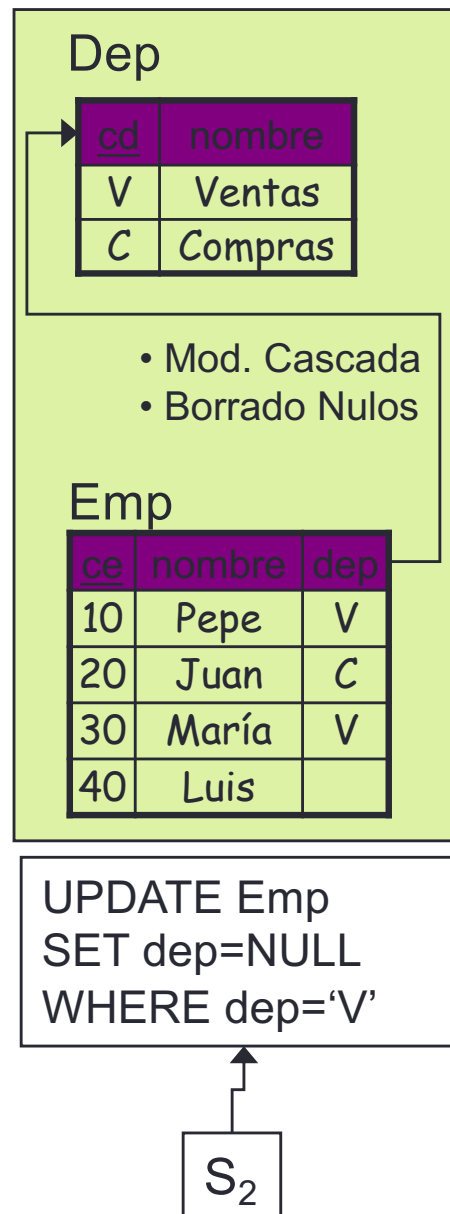
- si la condición es cierta

ejecutar la acción de r2 con **Algoritmo SQL**

fin_mientras

WHEN **NULL** IS NOT NULL

Ejemplo 1: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

Reglas AFTER para S_1 y S_2

mientras $\{\} \neq \emptyset$

1. seleccionar una regla activada:

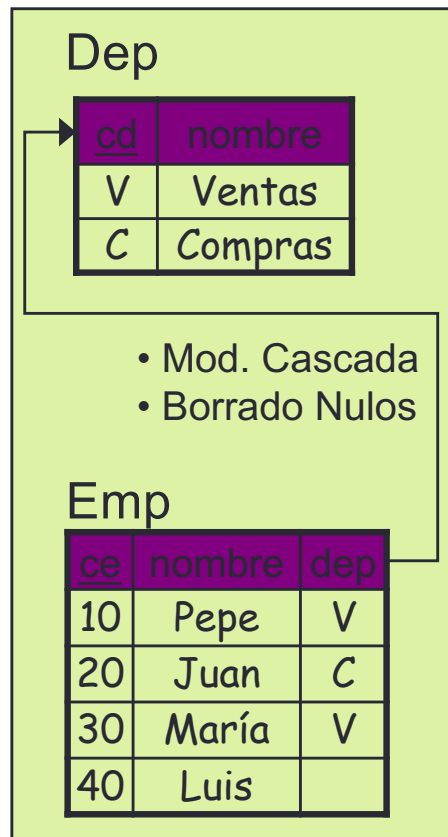
2. si

sino

fin_mientras

FIN de la 2ª ejecución
de Algoritmo A2_{SQL}

Ejemplo 1: modelo de ejecución de reglas en SQL3



```
DELETE FROM
Dep
WHERE cd='V'
```

S₁

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_i (inicialización de parámetros)

Procesar con A2_{SQL} reglas BEFORE para S_i

Ejecutar S_i

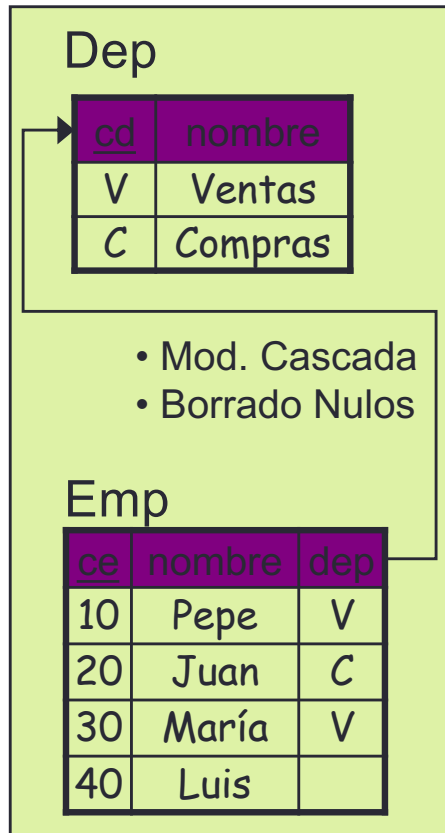
Comprobación de restricciones relevantes para S_i

- restricciones PRIMARY KEY ERROR
- integridad referencial CASCADE y SET NULL

Procesar con A2_{SQL} reglas AFTER para S₁, S₂

FIN de la 1ª ejecución del Algoritmo SQL

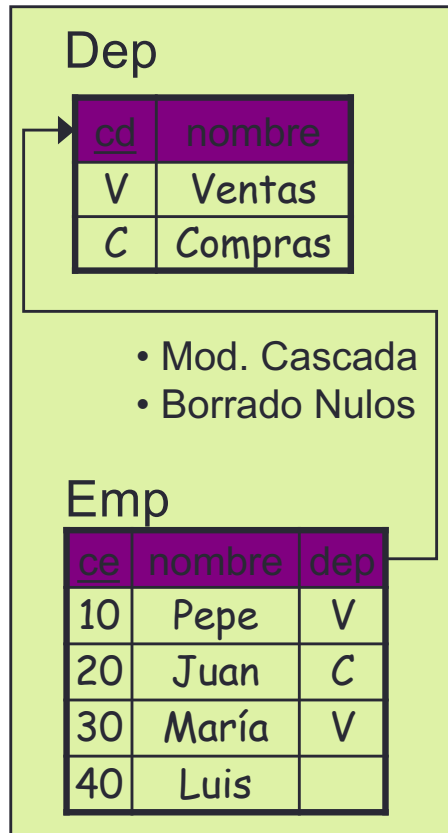
Ejemplo 2: modelo de ejecución de reglas en SQL3



```
CREATE TRIGGER r3
AFTER INSERT ON Dep
FOR EACH ROW
BEGIN
    aux NUMBER;
    SELECT ce INTO aux FROM Emp
    WHERE ce=(SELECT MIN(ce) FROM Emp WHERE dep IS NULL);
    UPDATE Emp SET dep=NEW.cd WHERE ce=aux;
END
```

```
CREATE TRIGGER r4
BEFORE UPDATE OF dep ON Emp
FOR EACH ROW
WHEN NEW.dep IS NOT NULL
BEGIN
    aux NUMBER;
    SELECT count(*) INTO aux FROM Emp WHERE dep= NEW.dep;
    IF aux=20 THEN SIGNAL SQLSTATE;
END
```

Ejemplo 2: modelo de ejecución de reglas en SQL3



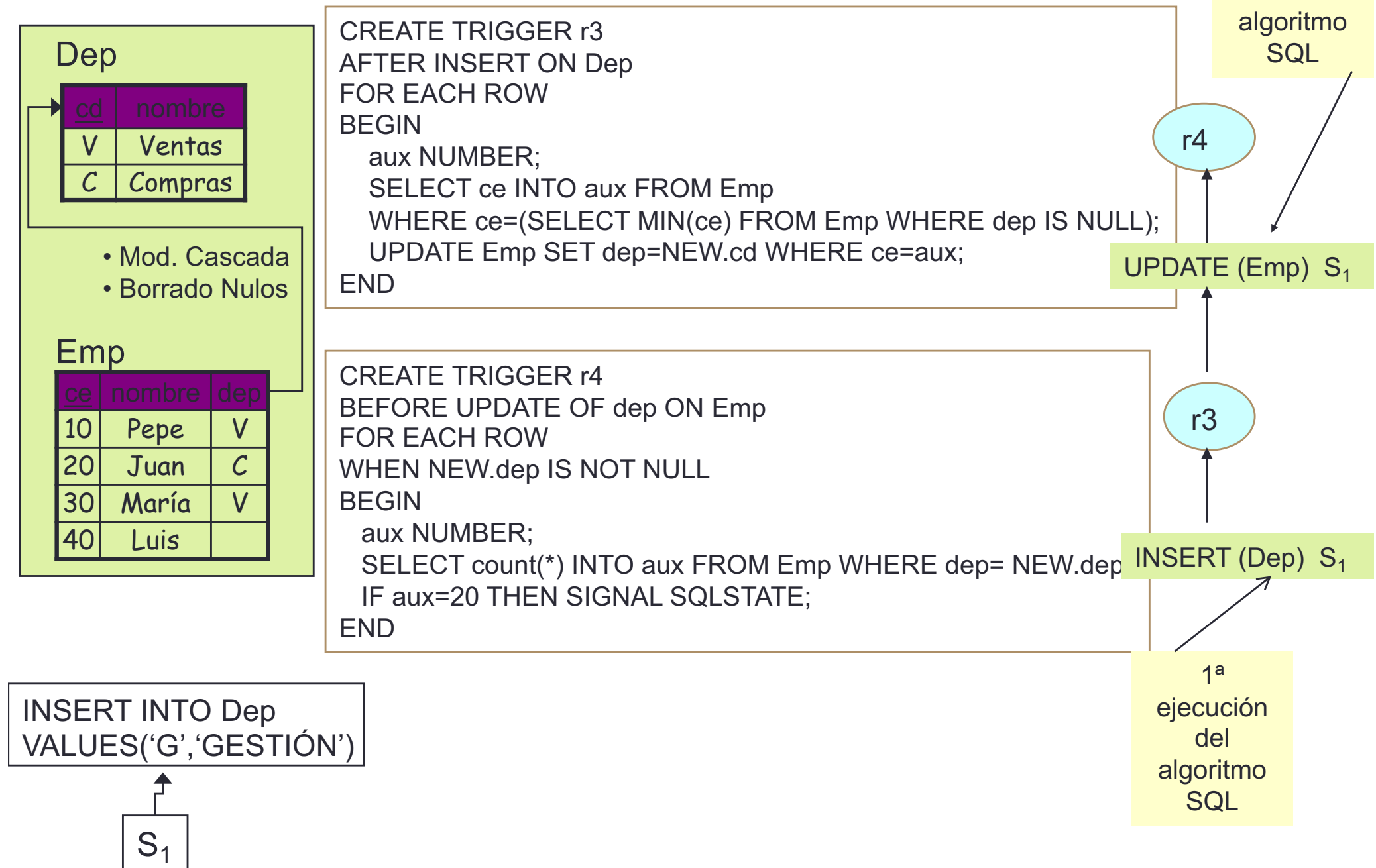
```
CREATE TRIGGER r3
AFTER INSERT ON Dep
FOR EACH ROW
BEGIN
    aux NUMBER;
    SELECT ce INTO aux FROM Emp
    WHERE ce=(SELECT MIN(ce) FROM Emp WHERE dep IS NULL);
    UPDATE Emp SET dep=NEW.cd WHERE ce=aux;
END
```

Cuando se inserta un nuevo dpto. se le asigna el empleado sin departamento con código menor

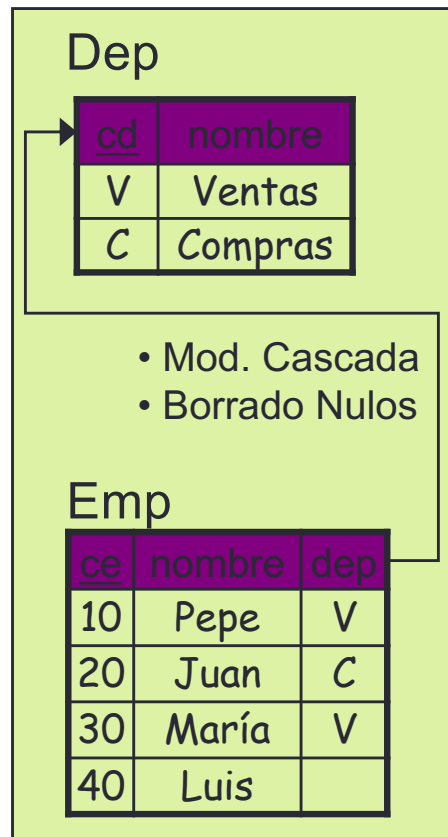
```
CREATE TRIGGER r4
BEFORE UPDATE OF dep ON Emp
FOR EACH ROW
WHEN NEW.dep IS NOT NULL
BEGIN
    aux NUMBER;
    SELECT count(*) INTO aux FROM Emp WHERE dep= NEW.dep;
    IF aux=20 THEN SIGNAL SQLSTATE;
END
```

En un departamento no puede haber mas de 20 empleados

Ejemplo 2: modelo de ejecución de reglas en SQL3



Ejemplo 2: modelo de ejecución de reglas en SQL3



Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_1 (inicialización de parámetros)

OLD

--	--

NEW

G	Gestión
---	---------

OLD_TABLE

--	--

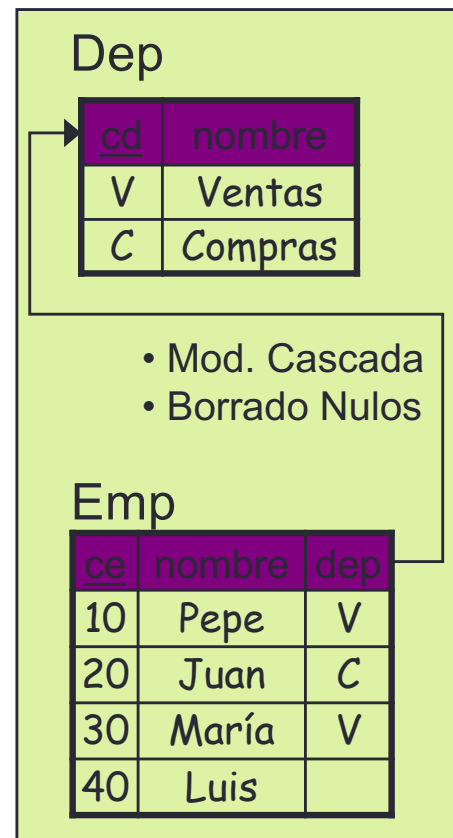
NEW TABLE

G	Gestión
---	---------

INSERT INTO Dep
VALUES('G','GESTIÓN')

S_1

Ejemplo 2: modelo de ejecución de reglas en SQL3



INSERT INTO Dep
VALUES('G','GESTIÓN')

S₁

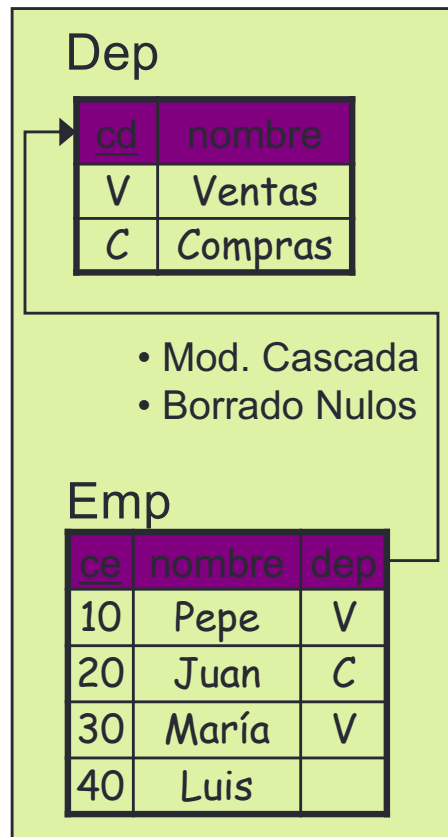
Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₁

Ejemplo 2: modelo de ejecución de reglas en SQL3



INSERT INTO Dep
VALUES('G','GESTIÓN')

S₁

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)

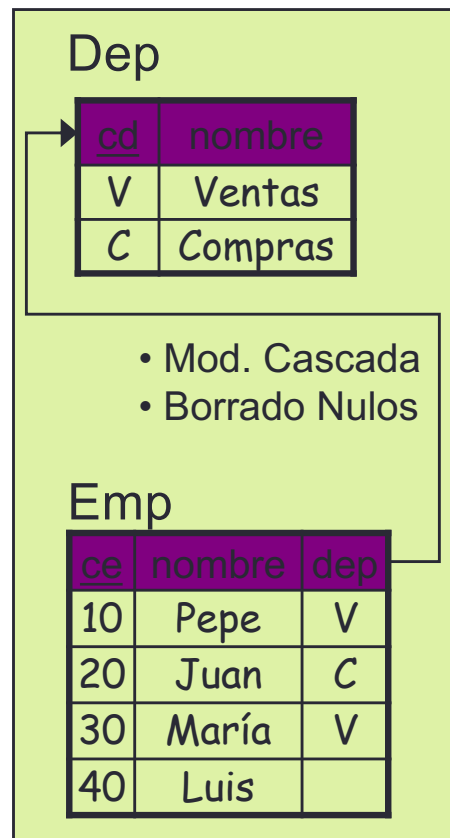


Procesar con A2_{SQL} reglas BEFORE para S₁



Ejecutar S₁

Ejemplo 2: modelo de ejecución de reglas en SQL3



INSERT INTO Dep
VALUES('G','GESTIÓN')

S_1

Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_1 (inicialización de parámetros)



Procesar con $A2_{SQL}$ reglas BEFORE para S_1



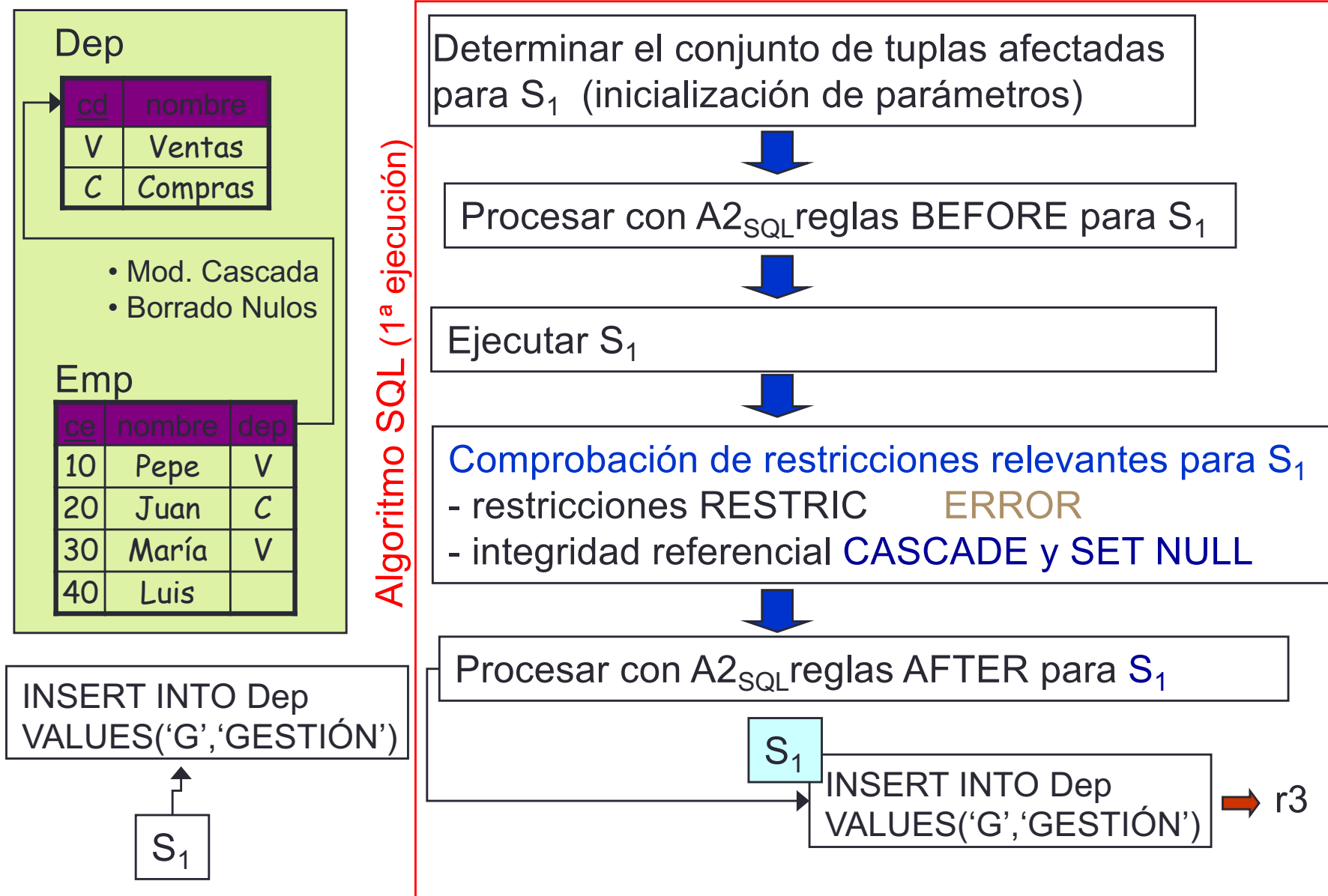
Ejecutar S_1



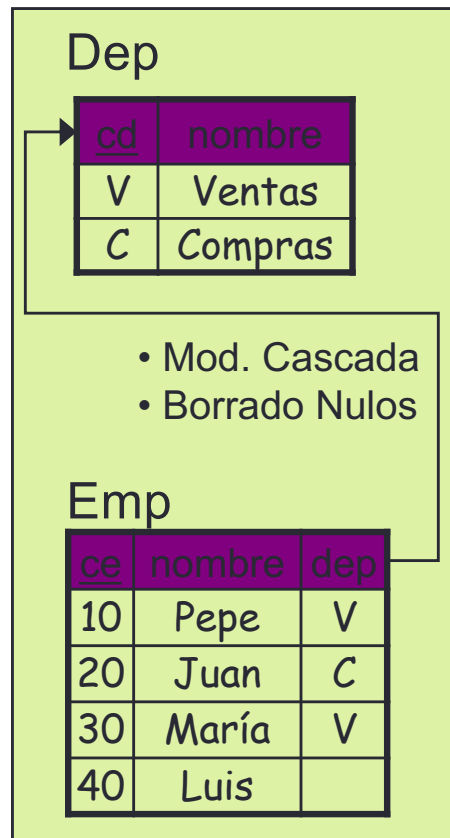
Comprobación de restricciones relevantes para S_1

- restricciones RESTRIC **ERROR**
- integridad referencial **CASCADE** y **SET NULL**

Ejemplo 2: modelo de ejecución de reglas en SQL3



Ejemplo 2: modelo de ejecución de reglas en SQL3



A2_{SQL} (1ª ejecución)

INSERT INTO Dep
VALUES('G','GESTIÓN')

S₁

Reglas AFTER para INSERT

mientras {r3} ≠ ∅

1. seleccionar una regla activada: r3
2. si r3 es de tipo FOR EACH STATEMENT
sino

- para cada tupla afectada por el evento de r3
- evaluar la condición de r3
- si la condición es cierta
ejecutar la acción de r3 con **Algoritmo SQL**

G Gestión

fin_mientras

OLD

--	--

NEW

G	Gestión
---	---------

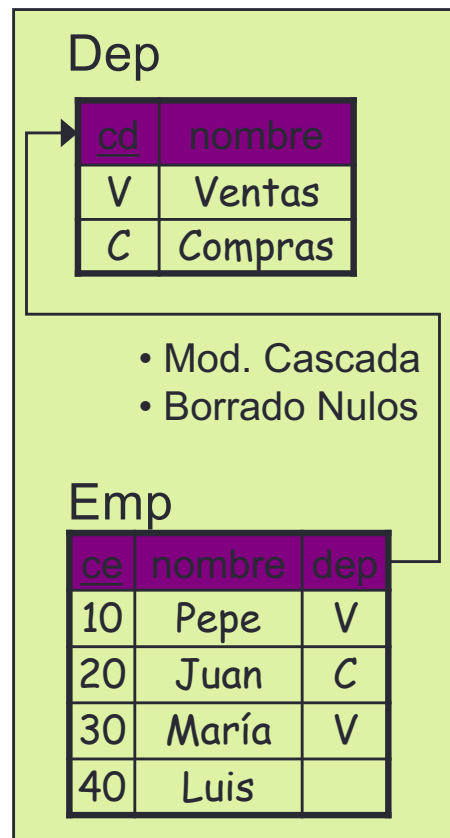
OLD_TABLE

--	--

NEW TABLE

G	Gestión
---	---------

Ejemplo 2: modelo de ejecución de reglas en SQL3



A2_{SQL} (1ª ejecución)

INSERT INTO Dep
VALUES('G','GESTIÓN')



Reglas AFTER para INSERT

mientras {r3} ≠ ∅

1. seleccionar una regla activada: r3
2. si r3 es de tipo FOR EACH STATEMENT
sino

- para cada tupla afectada por el evento de r3

- evaluar la condición de r3

- si la condición es cierta

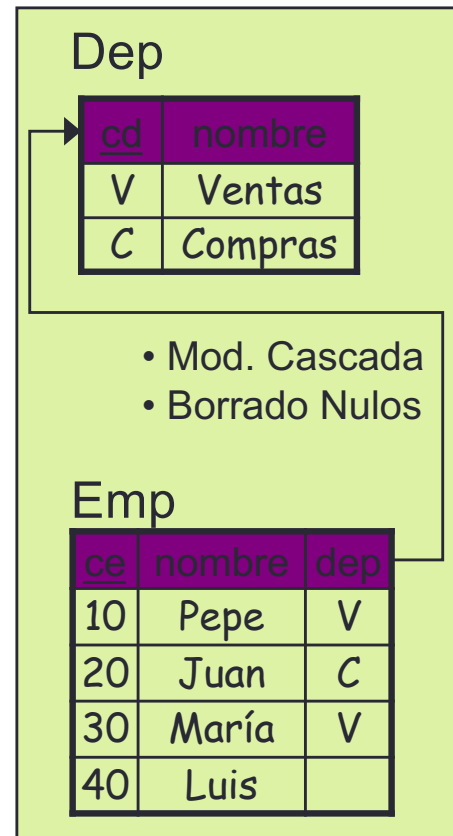
ejecutar la acción de r3 con **Algoritmo SQL**

G Gestión

fin_mientras



Ejemplo 2: modelo de ejecución de reglas en SQL3



A2_{SQL} (1ª ejecución)

INSERT INTO Dep
VALUES('G','GESTIÓN')

S₁

Reglas AFTER para INSERT

mientras {r3} ≠ ∅

1. seleccionar una regla activada: r3

2. si r3 es de tipo FOR EACH STATEMENT

sino

- para cada tupla afectada por el evento de r3
- evaluar la condición de r3
- si la condición es cierta

ejecutar la acción de r3 con **Algoritmo SQL**

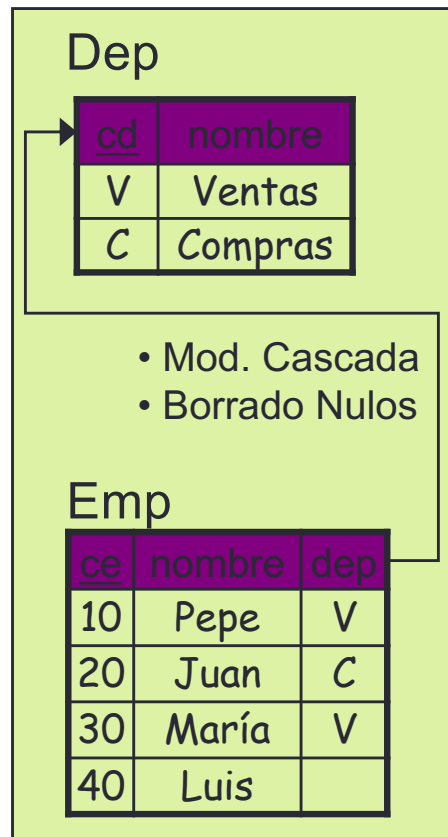
fin_mientras

G Gestión

aux NUMBER;
SELECT ce INTO aux FROM Emp
WHERE ce=(SELECT MIN(ce) FROM Emp WHERE dep IS NULL);
UPDATE Emp SET dep='G' WHERE ce=aux;
END

40

Ejemplo 2: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

S₁

Algoritmo SQL (2ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)

OLD

40	Luis	
----	------	--

NEW

40	Luis	G
----	------	---

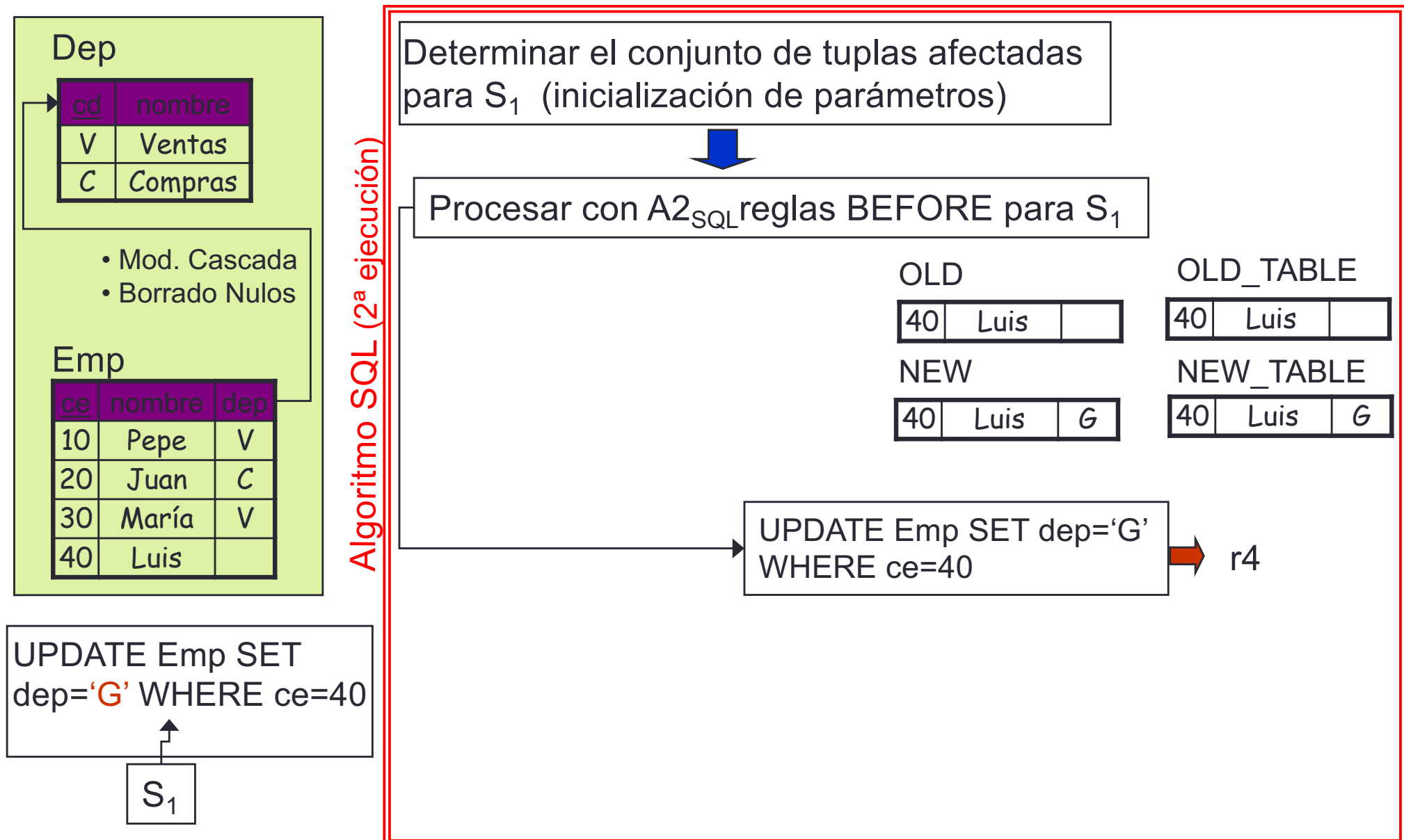
OLD_TABLE

40	Luis	
----	------	--

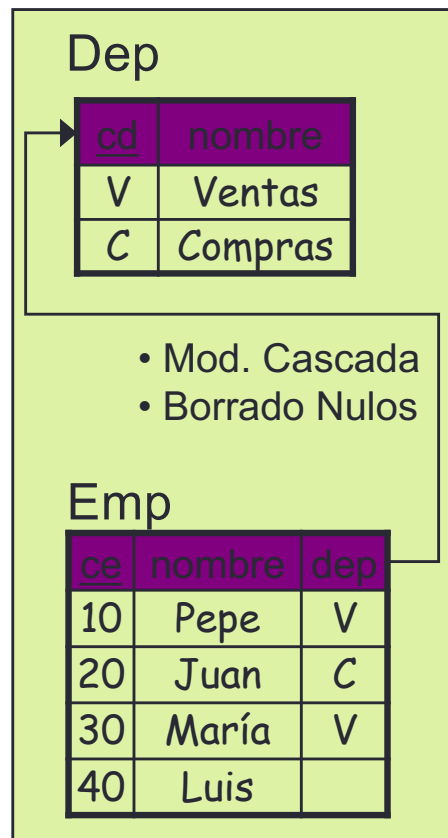
NEW_TABLE

40	Luis	G
----	------	---

Ejemplo 2: modelo de ejecución de reglas en SQL3



Ejemplo 2: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

S₁

A2_{SQL} (2ª ejecución)

Reglas BEFORE para UPDATE

mientras {r4} ≠ ∅

1. seleccionar una regla activada: r4
2. si r4 es de tipo FOR EACH STATEMENT
sino

- para cada tupla afectada por el evento de r4
- evaluar la condición de r4
- si la condición es cierta

ejecutar la acción de r4 con **Algoritmo SQL**

fin_mientras

40 Luis G

OLD

40	Luis	
----	------	--

NEW

40	Luis	G
----	------	---

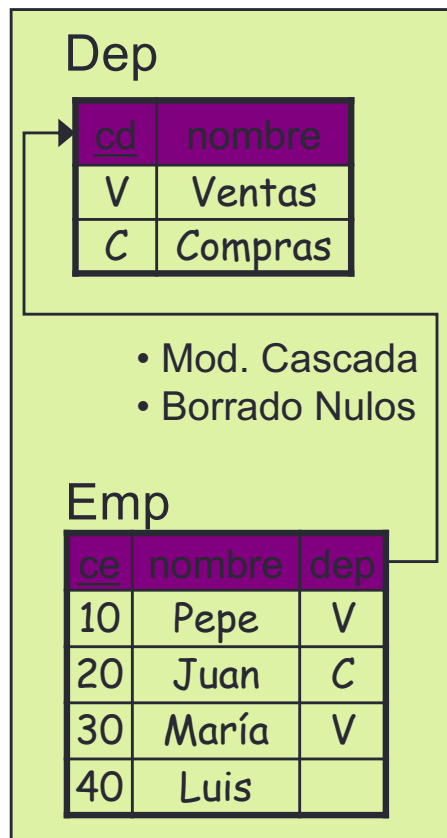
OLD_TABLE

40	Luis	
----	------	--

NEW_TABLE

40	Luis	G
----	------	---

Ejemplo 2: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

mientras {r4} ≠ ∅

1. seleccionar una regla activada: r4

2. si r4 es de tipo FOR EACH STATEMENT

sino

- para cada tupla afectada por el evento de r4

- evaluar la condición de r4

- si la condición es cierta

ejecutar la acción de r4 con **Algoritmo SQL**

fin_mientras

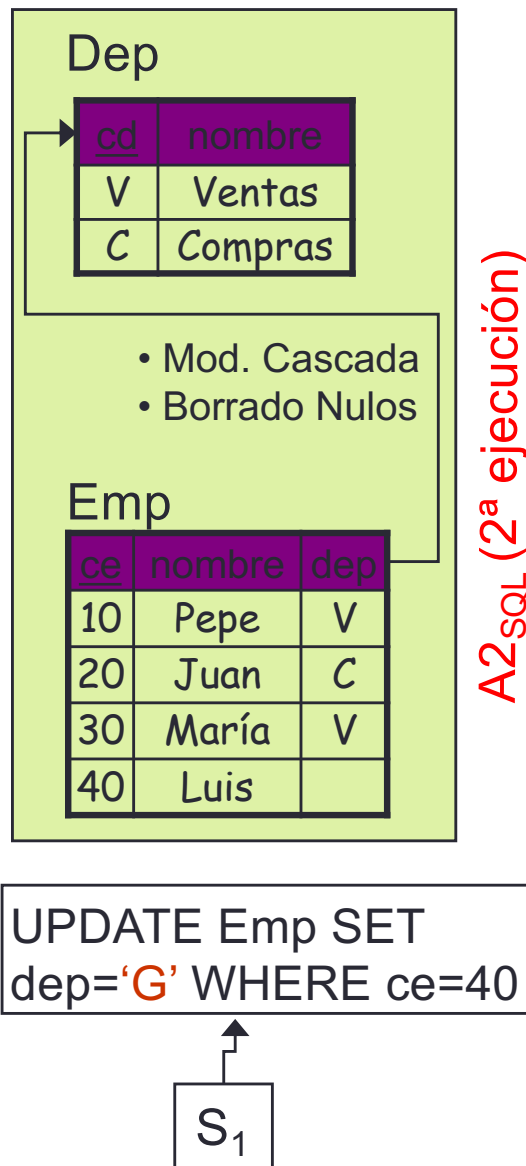
40 Luis G

WHEN 'G' IS NOT NULL

UPDATE Emp SET
dep='G' WHERE ce=40

S₁

Ejemplo 2: modelo de ejecución de reglas en SQL3



mientras {r4} ≠ ∅

1. seleccionar una regla activada: r4

2. si r4 es de tipo FOR EACH STATEMENT

sino

- para cada tupla afectada por el evento de r4

- evaluar la condición de r4

- si la condición es cierta

ejecutar la acción de r4 con **Algoritmo SQL**

fin_mientras

0

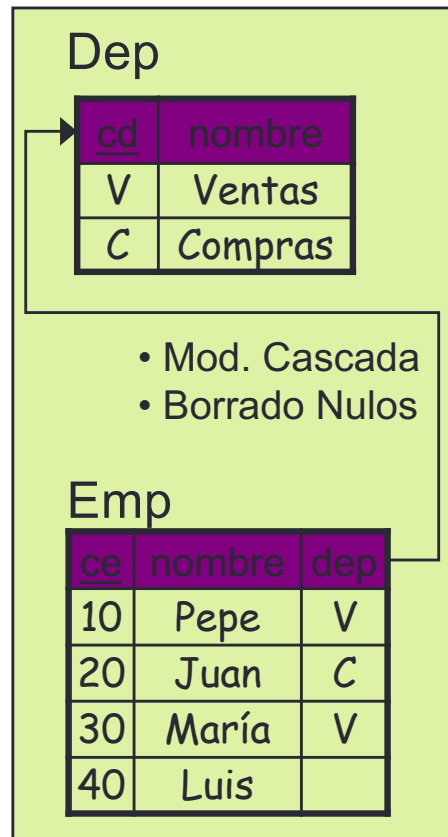
BEGIN

SELECT count(*) INTO aux FROM Emp WHERE dep= 'G';

IF aux=20 THEN SIGNAL SQLSTATE;

END

Ejemplo 2: modelo de ejecución de reglas en SQL3



A2_{SQL} (2ª ejecución)

```
UPDATE Emp SET  
dep='G' WHERE ce=40
```

S₁

mientras $\{\neq \emptyset\}$

1. seleccionar una regla activada:

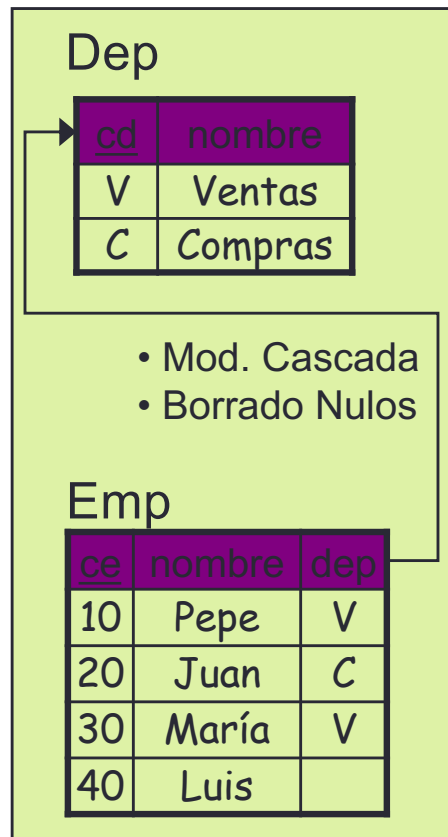
2. si

sino

fin_mientras

FIN de la 2ª ejecución
de Algoritmo A2_{SQL}

Ejemplo 2: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

S₂

Algoritmo SQL (2ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)

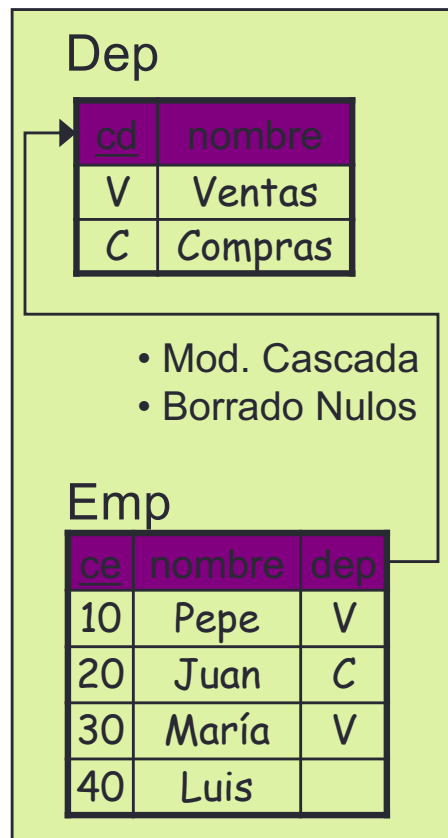


Procesar con A2_{SQL} reglas BEFORE para S₁



Ejecutar S₁

Ejemplo 2: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

S₂

Algoritmo SQL (2ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)



Procesar con A2_{SQL} reglas BEFORE para S₁



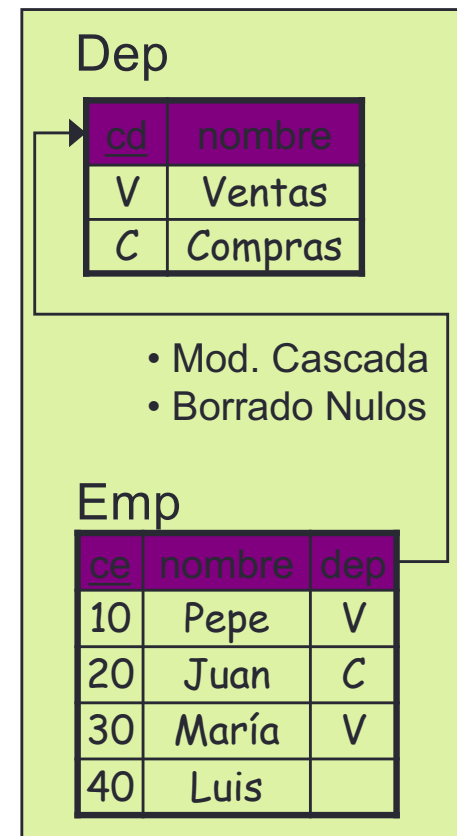
Ejecutar S₁



Comprobación de restricciones relevantes para S₁

- restricciones RESTRIC **ERROR**
- integridad referencial **CASCADE** y **SET NULL**

Ejemplo 2: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

S₁

Algoritmo SQL (2ª ejecución)

Determinar el conjunto de tuplas afectadas para S₁ (inicialización de parámetros)

Procesar con A2_{SQL} reglas BEFORE para S₁

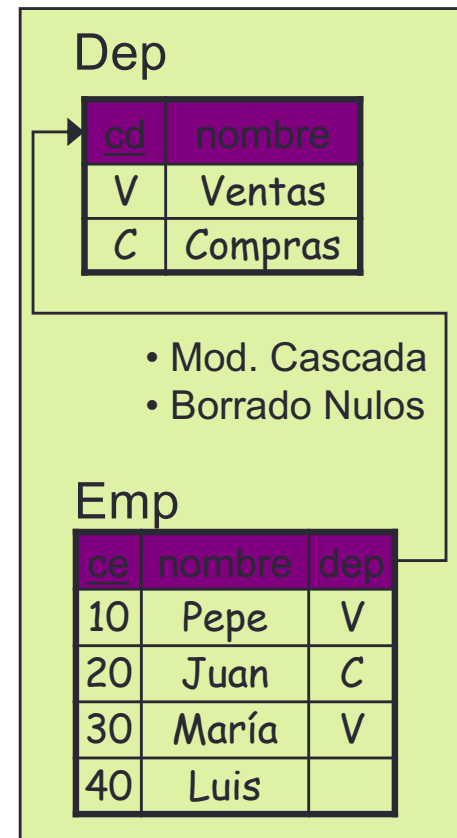
Ejecutar S₁

Comprobación de restricciones relevantes para S₁

- restricciones RESTRIC **ERROR**
- integridad referencial **CASCADE** y **SET NULL**

Procesar con A2_{SQL} reglas AFTER para S₁

Ejemplo 1: modelo de ejecución de reglas en SQL3



UPDATE Emp SET
dep='G' WHERE ce=40

↑

S₁

Algoritmo SQL (2ª ejecución)

Determinar el conjunto de tuplas afectadas para S_i (inicialización de parámetros)

Procesar con A2_{SQL} reglas BEFORE para S_i

Ejecutar S_i

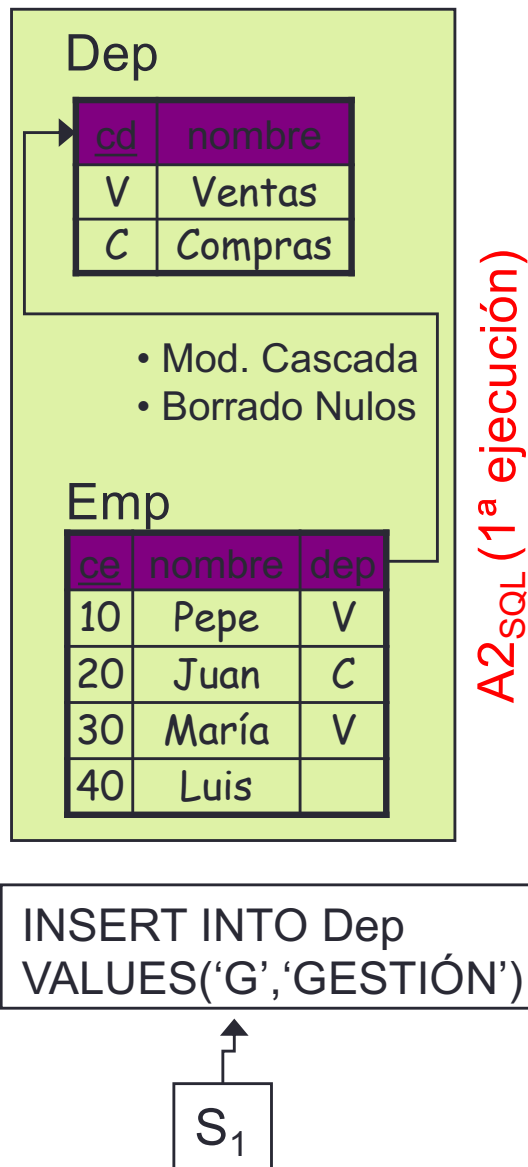
Comprobación de restricciones relevantes para S_i

- restricciones PRIMARY KEY ERROR
- integridad referencial CASCADE y SET NULL

Procesar con A2_{SQL} reglas AFTER para S₁

FIN de la 2ª ejecución
de Algoritmo SQL

Ejemplo 2: modelo de ejecución de reglas en SQL3



mientras $\{\} \neq \emptyset$

1. seleccionar una regla activada:

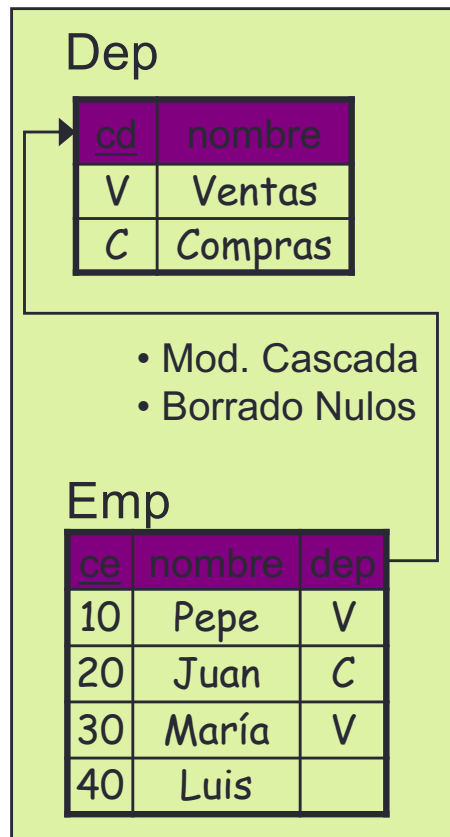
2. si

sino

fin_mientras

FIN de la 1ª ejecución
de Algoritmo A2_{SQL}

Ejemplo 2: modelo de ejecución de reglas en SQL3



INSERT INTO Dep
VALUES('G','GESTIÓN')



Algoritmo SQL (1ª ejecución)

Determinar el conjunto de tuplas afectadas para S_i (inicialización de parámetros)

Procesar con $A2_{SQL}$ reglas BEFORE para S_i

Ejecutar S_i

Comprobación de restricciones relevantes para S_i

- restricciones PRIMARY KEY
- integridad referencial

ERROR

CASCADE y SET NULL

Procesar con $A2_{SQL}$ reglas AFTER para S_1

FIN de la 1ª ejecución
de Algoritmo SQL