



Diseño y Gestión de Bases de Datos

Tema 2

**Procesamiento de transacciones y
mantenimiento de la integridad**

Profesoras: Laura Mota y Pedro Valderas

Procesamiento de transacciones y mantenimiento de la integridad.

Objetivos:

- ✓ Revisar el concepto de transacción en bases de datos.
- ✓ Revisar la definición de transacciones en SQL.
- ✓ Revisar las estrategias de comprobación de restricciones de integridad durante el procesamiento de transacciones.
- ✓ Revisar el tratamiento de la integridad en SQL.

Procesamiento de transacciones y mantenimiento de la integridad.

1 Concepto de transacción.

2 Operaciones y estados de una transacción.

3 Propiedades del procesamiento de transacciones.

4 Definición de transacciones en SQL.

5 Concepto de restricción de integridad.

6 Comprobación de restricciones en SQL.

1 Concepto de transacción.

Acceso a bases de datos relacionales (consulta o actualización): lenguaje SQL

- ✓ Consultar el valor de las filas de una o varias tablas: **SELECT**
- ✓ Insertar filas en una tabla: **INSERT**
- ✓ Borrar filas de una tabla: **DELETE**
- ✓ Modificar el valor de las filas de una tabla: **UPDATE**

4

La manipulación (consulta y actualización) de bases de datos relacionales se realiza con el lenguaje estándar **SQL**.

La consulta se realiza con la instrucción **SELECT** (seleccionar filas de una o varias tablas), y la actualización con las instrucciones **INSERT** (insertar filas en una tabla), **DELETE** (borrar filas de una tabla) y **UPDATE** (modificar filas de una tabla).

La operación de consulta actúa sobre una o varias tablas, las operaciones de actualización actúan sobre una única tabla.

1 Concepto de transacción.

Empleado (dni: char(10), nombre: char(60), salario: real, dpto: char(5))

CP: {dni} VNN: {dpto}

CAj: {dpto} → Departamento f(dpto)=código

Departamento (código: char(5), nombre: char(50), ubicación: char(15))

CP: {código} VNN: {nombre}

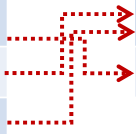
R1: "A todo departamento pertenece al menos un empleado"

Empleado

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruíz	1500	d1
37418739	Luis Cerdá	1000	d1

Departamento

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2



Las *restricciones de integridad* (RI): CP, VNN, CAj, R1, se satisfacen en la base de datos.

5

En el ejemplo se presenta el esquema lógico de una base de datos relacional con dos tablas: una tabla de empleados (*Empleado*) y una tabla de departamentos (*Departamento*). La clave ajena en la tabla *Empleado* expresa que todo empleado debe pertenecer a un departamento de la organización, y la restricción de integridad R1 (se verá más adelante su definición en SQL) expresa que a todo departamento de la organización debe pertenecer al menos un empleado. Se observa que, en la extensión actual (estado) de la base de datos, estas restricciones de integridad se cumplen (se satisfacen).

Comentario: las flechas rojas incluidas en la transparencia simplemente denotan con qué fila de la tabla *Departamento* se relaciona cada fila de la table *Empleado*, no debe confundirse con la existencia de punteros.

1 Concepto de transacción.

Empleado (dni: char(10), nombre: char(60), salario: real, dpto: char(5))

CP: {dni} VNN: {dpto}

CAj: {dpto} → Departamento f(dpto)=código

Departamento (código: char(5), nombre: char(50), ubicación: char(15))

CP: {código} VNN: {nombre}

R1: "A todo departamento pertenece al menos un empleado"

Actualización de la BD:

"Crear un nuevo departamento, <d3, Personal, Planta3>, y asignarle el empleado de DNI 65743983 (ya existente en la base de datos)"

Empleado				Departamento		
dni	nombre	salario	dpto	código	nombre	ubicación
23125679	Juan García	2000	d2	d1	Compras	Planta1
65743983	Pedro Ruíz	1500	d3	d2	Ventas	Planta2
37418739	Luis Cerdá	1000	d1	d3	Personal	Planta3

6

Supóngase que se desea crear (dar de alta) un nuevo departamento en la organización, tal como expresa el requerimiento de actualización de la base de datos que plantean los usuarios. La base de datos, una vez actualizada, quedaría como se muestra en la transparencia.

1 Concepto de transacción.

1) Inserción en Departamento: <d3, Personal, Planta3>

ERROR: la restricción R1 no se cumple
Hay un departamento sin profesores

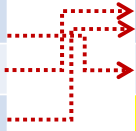
2) **Modificación en Empleado:** modificación del atributo *dpto* del empleado con DNI 65743983 (nuevo valor: d3)

Empleado

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruíz	1500	d1
37418739	Luis Cerdá	1000	d1

Departamento

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2
d3	Personal	Planta3



1 Concepto de transacción.

- 1) **Modificación en Empleado:** modificación del atributo *dpto* del empleado con DNI 65743983 (nuevo valor: d3)

ERROR: la restricción de CAj no se cumple
Hay un empleado cuyo departamento no existe

- 2) **Inserción en Departamento:** <d3, Personal, Planta3>

Empleado

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruíz	1500	d3
37418739	Luis Cerdá	1000	d1

Departamento

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2

1 Concepto de transacción.

- 1) **Inserción en Departamento:** <d3, Personal, Planta3>

ERROR: la restricción R1 no se cumple

- 2) **Modificación en Empleado:** modificación del atributo *dpto* del empleado con DNI 65743983 (nuevo valor: d3)

-
- 1) **Modificación en Empleado:** modificación del atributo *dpto* del empleado con DNI 65743983 (nuevo valor: d3)

ERROR: la restricción de CAj no se cumple

- 2) **Inserción en Departamento:** <d3, Personal, Planta3>

¡Necesidad del concepto de transacción!

9

Es obvio que esta actualización de la base de datos exige operaciones de actualización sobre las dos tablas: la inserción de una nueva fila en *Departamento* y la modificación de una fila en *Empleado*.

Es importante recordar que, aunque las tablas son estructuras de datos independientes, están relacionadas entre sí por restricciones de integridad (clave ajena y restricción R1), y que el SGBD debe asegurar en todo momento el cumplimiento de estas restricciones.

Cualquier intento de ejecutar las dos operaciones de actualización (en cualquier orden) fracasa porque las relaciones entre las dos tablas, establecidas por las restricciones de integridad, lo impide.

Surge de esta forma el concepto de TRANSACCIÓN como unidad de ejecución (ejecución conjunta) de varias operaciones sobre las tablas de la base de datos: interesa el efecto global del conjunto de operaciones, no el efecto individual de cada una de ellas.

Para utilizar este concepto, será necesario poder indicar al SGBD qué conjunto de operaciones constituyen una transacción, para que, de esta forma, el SGBD pueda considerar su efecto global antes de comprobar las restricciones de integridad definidas en el esquema de la base de datos.

1 Concepto de transacción.

INICIO

```
INSERT INTO Departamento VALUES ('d3', 'Personal', 'Planta3');
```

```
UPDATE Empleado SET dpto='d3' WHERE dni='65743983');
```

FIN



- ✓ La transacción se procesa como una operación atómica.
- ✓ Las restricciones se comprueban al final de la transacción.
- ✓ La transacción se rechaza si alguna restricción se viola.

10

Si el SGBD ejecuta las dos operaciones de actualización como una unidad de ejecución (transacción) y comprueba las restricciones de integridad después de observar su efecto global, la base de datos puede ser actualizada sin ningún problema.

1 Concepto de transacción.

INICIO

```
INSERT INTO Departamento VALUES ('d3', 'Personal', 'Planta3');
```

```
UPDATE Empleado SET dpto='d3' WHERE dni='65743983';
```

FIN

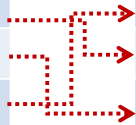


Empleado

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruíz	1500	d3
37418739	Luis Cerdá	1000	d1

Departamento

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2
d3	Personal	Planta3




Las RI se satisfacen en la base de datos: CP, VNN, CAj, R1

11

En el nuevo estado de la base de datos, después de ejecutar la transacción, se cumplen las dos restricciones de integridad (clave ajena y restricción R1).

1 Concepto de transacción.

Las operaciones de acceso a una BD se organizan en transacciones

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

Procesar correctamente una transacción significa:

(a) todas las operaciones de la transacción se ejecutan con éxito y sus cambios (actualizaciones) quedan grabados permanentemente en la base de datos,

o bien

(b) la transacción no tiene ningún efecto en la base de datos.

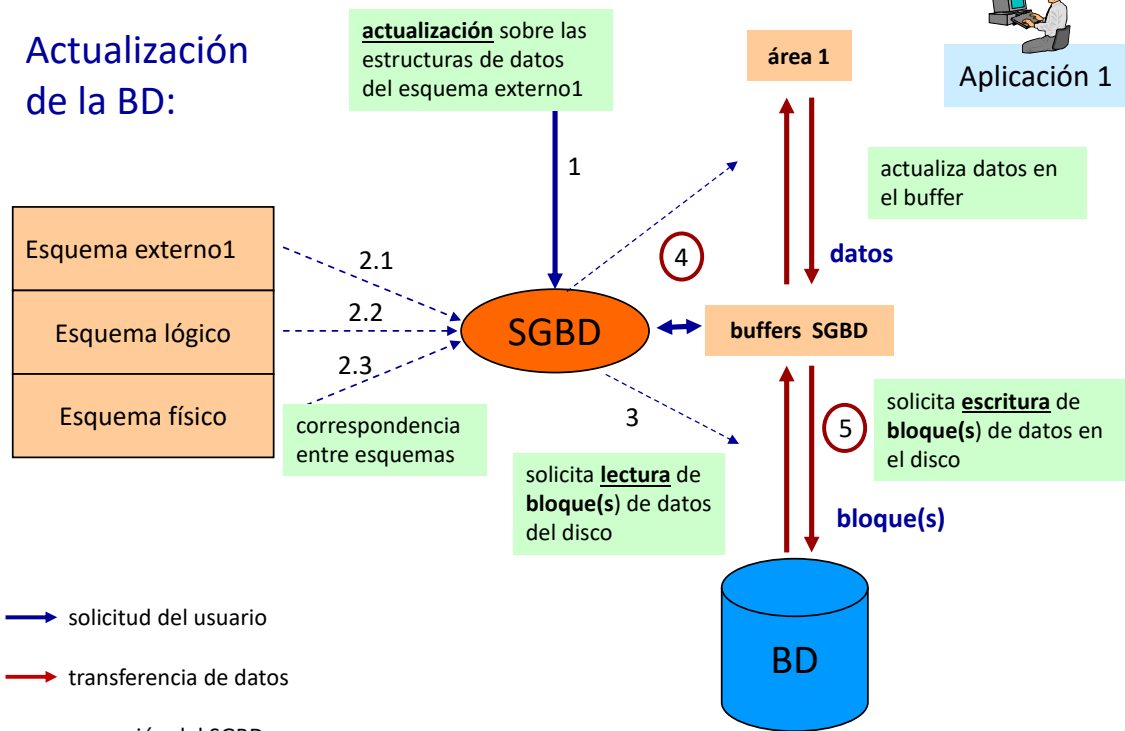
12

Si definimos el concepto de transacción como “**unidad de ejecución**”, la ejecución de transacciones debe respetar este significado, es decir, ejecutar una transacción significará o ejecutar todas sus operaciones o no ejecutar ninguna de ellas.

Hay que observar que cuando se dice “ejecutar todas sus operaciones” se refiere a dejar los cambios de esas operaciones grabados en la BD **en disco**. Recuérdese a este respecto, el desajuste temporal entre la ejecución de una operación de actualización (en memoria principal) y la actualización de la base de datos (en memoria secundaria), como se muestra en el diagrama siguiente (pasos 4 y 5). Este desajuste temporal entre los pasos 4 y 5 estará en el centro del funcionamiento del SGBD, como se verá en temas posteriores.

1 Concepto de transacción.

Actualización de la BD:



13

Recuérdese el desajuste temporal entre los pasos 4 y 5.

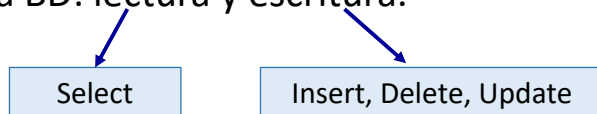
Procesamiento de transacciones y mantenimiento de la integridad.

- 1 Concepto de transacción.
- 2 Operaciones y estados de una transacción.
- 3 Propiedades del procesamiento de transacciones.
- 4 Definición de transacciones en SQL.
- 5 Concepto de restricción de integridad.
- 6 Comprobación de restricciones en SQL.

2 Operaciones y estados de una transacción.

Modelo simplificado para estudiar el procesamiento de transacciones:

- ✓ Base de datos: colección de elementos de datos con nombre.
- ✓ Elemento de datos: atributo, fila, tabla, etc. (**granularidad del elemento**).
- ✓ Operaciones de acceso a la BD: lectura y escritura.



Estas simplificaciones no quitan generalidad al estudio del procesamiento de transacciones en un SGBD.

15

El concepto de transacción es universal en bases de datos, es decir es independiente del tipo de bases de datos (modelo de datos) que se esté utilizando.

En una base de datos relacional, las estructuras de datos son tablas y las operaciones de manipulación tienen la sintaxis precisa del lenguaje SQL: SELECT, INSERT, DELETE y UPDATE.

Para simplificar el estudio de la ejecución de transacciones y sus efectos en la base de datos es interesante hacer una simplificación e independizarse de la estructura tabla y del lenguaje SQL. Esto no significa que los ejemplos que aparecerán en las explicaciones no puedan ser sobre bases de datos relacionales.

Para no hablar de estructuras de datos concretas (como la tabla del modelo relacional) se hablará de elementos de datos (por ejemplo, una fila de una tabla); en cada modelo de datos estos elementos cambian.

Para no hablar de un lenguaje concreto (como el lenguaje relacional SQL), se hablará de operaciones generales de acceso a bases de datos. Las operaciones de acceso a una base de datos pueden ser operaciones de **consulta** (lectura) u operaciones de **actualización** (escritura). Las operaciones de consulta no cambian el contenido de la base de datos, sólo leen datos. Las operaciones de actualización cambian el contenido de la base de datos.

Las operaciones de actualización pueden ser, a su vez, operaciones de inserción de nuevos datos, u operaciones de eliminación o de modificación de datos existentes. Es decir cualquier operación de actualización significa escribir en la base de datos.

Con la idea de simplificar el estudio de la ejecución de transacciones, se pueden considerar sólo dos tipos de operaciones sobre la base de datos: **lectura** de datos, y **escritura** de datos.

2 Operaciones y estados de una transacción.

Operaciones de acceso a datos en una transacción:

(1) leer(X):

- (2) determinar la dirección del bloque que contiene el elemento X.
- (3) copiar el bloque del disco a un buffer de memoria principal (si el bloque no está ya en memoria principal).
- (4) copiar el elemento de datos X del buffer a la variable X del programa del usuario.

(1) escribir(X):

- (2) determinar la dirección del bloque que contiene o debe contener el elemento X.
- (3) copiar el bloque del disco a un buffer de memoria principal (si el bloque no está ya en la memoria principal).
- (4) copiar la variable X del programa del usuario al elemento de datos X en el buffer.
- (5) copiar el bloque actualizado del buffer al disco.

16

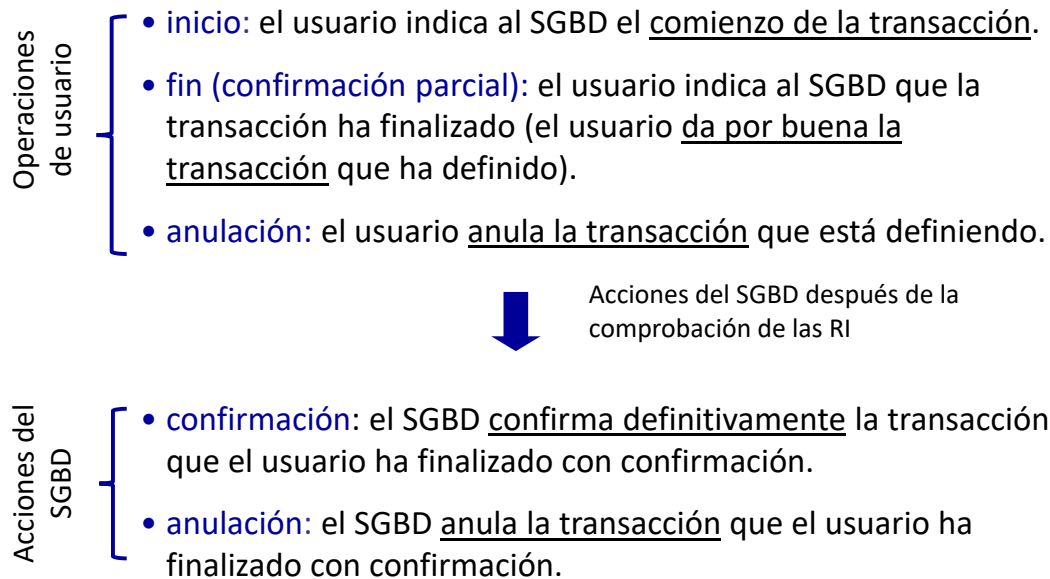
En la transparencia aparecen los pasos que sigue el SGBD para la ejecución de una operación de consulta (**leer**) y una operación de actualización (**escribir**) a la base de datos.

Estos pasos coinciden con los pasos que se vieron en los diagramas de las operaciones de lectura y actualización en el Tema 1.

De nuevo, es importante fijarse en los pasos 3, 4 y 5 de ambas operaciones.

2 Operaciones y estados de una transacción.

Operaciones adicionales en una transacción:



17

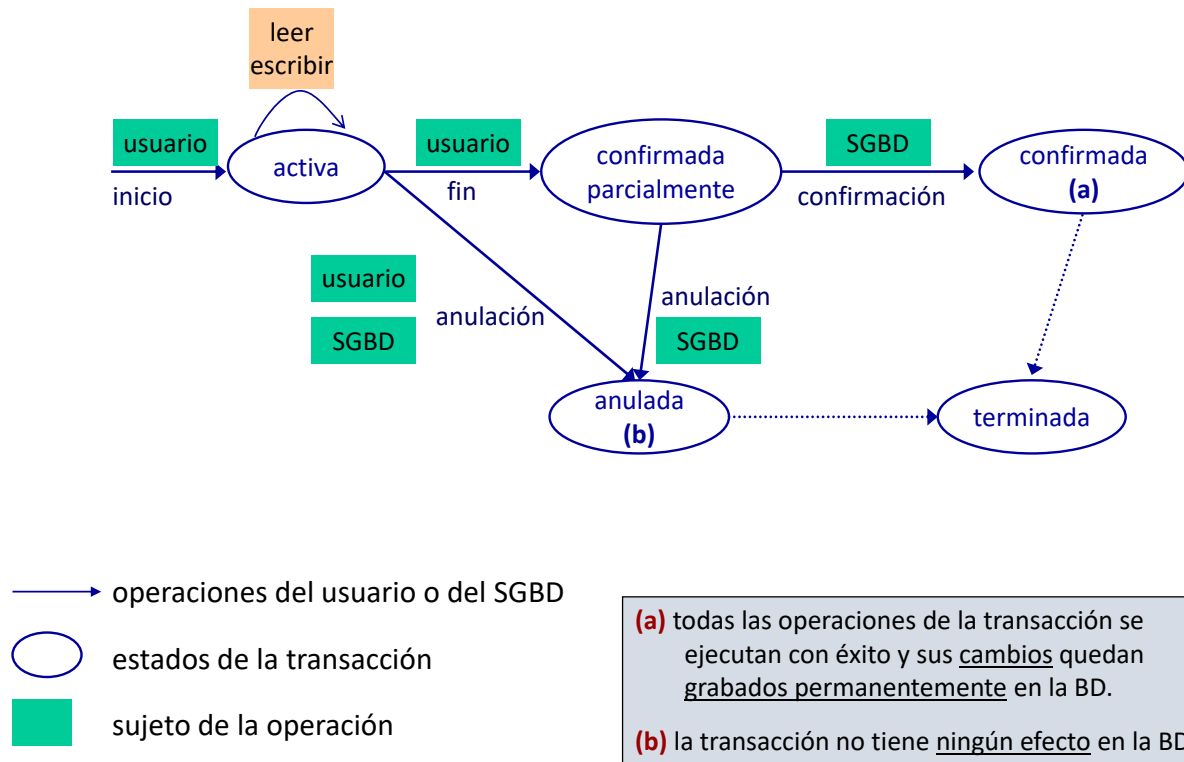
Las operaciones leer(X) y escribir(X) constituyen el contenido de las transacciones del usuario. Para definir una transacción harán falta también dos operaciones con las que el usuario pueda indicar al SGBD el inicio y el final de la transacción ya que es preciso delimitar claramente el conjunto de operaciones que constituyen la transacción.

Para flexibilizar la definición de transacciones se contemplan dos formas de finalizar una transacción por parte del usuario: finalización con **confirmación parcial** (el usuario da por buena la transacción que ha ejecutado), o finalización con **anulación** (el usuario rechaza la transacción que está ejecutado).

Por otro lado, es necesario recordar que el SGBD considera el efecto global de la transacción confirmada (por el usuario) antes de comprobar las restricciones de integridad del esquema. Según el resultado de esta comprobación, el **SGBD** puede **confirmar definitivamente** o **anular** (rechazar) la transacción del usuario.

Una transacción confirmada definitivamente por el SGBD debe quedar grabada en la base de datos en disco. Una transacción anulada por el usuario o por el SGBD no debe tener ningún efecto sobre la base de datos. (Estos aspectos se estudiarán con más detalle en el Tema 4).

2 Operaciones y estados de una transacción.



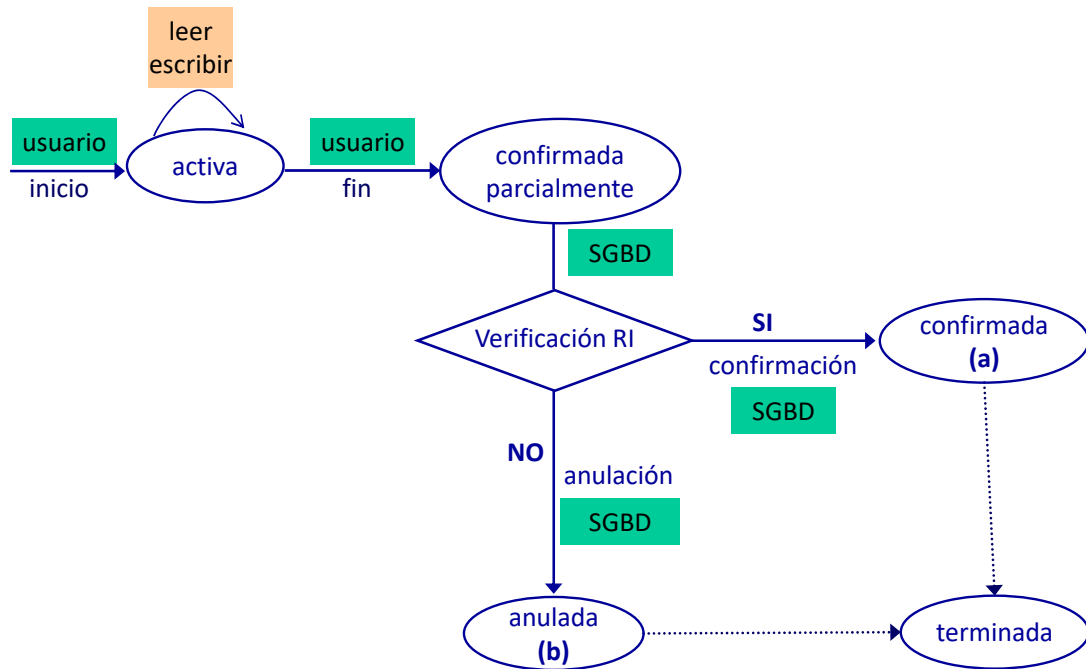
18

En este diagrama se muestran los estados y las transiciones de estado por las que puede pasar una transacción durante su ejecución.

Una transacción puede estar en los siguientes estados:

- **Activa:** el usuario inicia la transacción (operación **inicio**) y solicita operaciones de lectura o escritura sobre la base de datos
- **Confirmada parcialmente:** el usuario finaliza la transacción (operación **fin**) dándola por buena (transacción confirmada parcialmente por el usuario).
- **Anulada:** el usuario finaliza la transacción anulándola (operación **anulación**); o bien el SGBD anula la transacción que está activa (por la ocurrencia de errores) o anula la transacción que ya ha sido confirmada por el usuario (por la violación de alguna restricción de integridad).
- **Confirmada:** el SGBD confirma definitivamente una transacción finalizada con confirmación por el usuario (comprobación válida de restricciones de integridad).

2 Operaciones y estados de una transacción.



(a) todas las operaciones de la transacción se ejecutan con éxito y sus cambios quedan grabados permanentemente en la BD.

(b) la transacción no tiene ningún efecto en la BD.

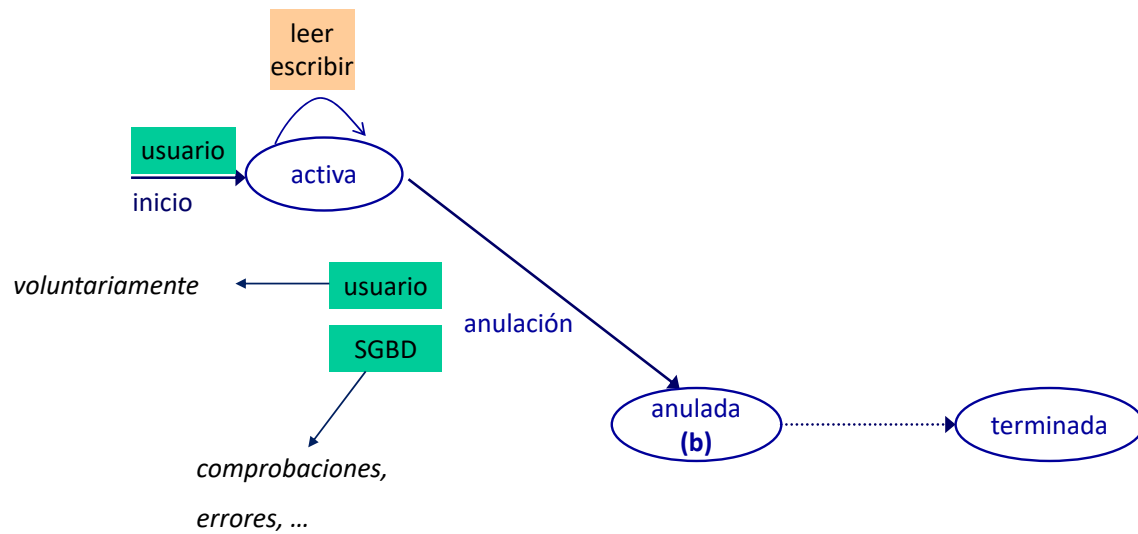
19

Después que ha finalizado una transacción con confirmación del usuario (confirmada parcialmente), el SGBD comprueba las restricciones de integridad del esquema sobre la base de datos resultante de su ejecución.

Si el resultado de esta comprobación es afirmativo (se satisfacen todas las restricciones de integridad) el SGBD confirma definitivamente la transacción. En este caso el SGBD debe asegurar que los cambios de la transacción son (o serán) grabados en la base de datos en disco (objetivo (a)).

Si el resultado de esta comprobación es negativo (se viola alguna de las restricciones de integridad) el SGBD anula la transacción. En este caso el SGBD debe asegurar que la transacción no deja ningún efecto sobre la base de datos (objetivo (b)).

2 Operaciones y estados de una transacción.



(a) todas las operaciones de la transacción se ejecutan con éxito y sus cambios quedan grabados permanentemente en la BD.

(b) la transacción no tiene ningún efecto en la BD.

20

Si la transacción está activa y el usuario o el SGBD anula la transacción, el SGBD debe asegurar que la transacción no deja ningún efecto sobre la base de datos (objetivo (b)).

Procesamiento de transacciones y mantenimiento de la integridad.

- 1 Concepto de transacción.
- 2 Operaciones y estados de una transacción.
- 3 **Propiedades del procesamiento de transacciones.**
- 4 Definición de transacciones en SQL.
- 5 Concepto de restricción de integridad.
- 6 Comprobación de restricciones en SQL.

3 Propiedades del procesamiento de transacciones.

Propiedades que debe cumplir la ejecución de transacciones:

- ✓ **Atomicidad:** una transacción es una unidad de ejecución (o se ejecutan todas sus operaciones o no se ejecuta ninguna de ellas). Tema 2
- ✓ **Consistencia:** una transacción debe conducir la BD de un estado consistente a otro estado consistente (*estado consistente*: se cumplen todas las restricciones de integridad del esquema). Tema 2
- ✓ **Aislamiento:** una transacción debe ejecutarse como si se ejecutase de forma aislada (en solitario). Tema 4
- ✓ **Persistencia:** los cambios de una transacción confirmada por el SGBD deben quedar grabados permanentemente en la BD. Tema 3



Propiedades ACID= Atomicity+Consistency+Isolation+Durability

22

La ejecución correcta de transacciones en un SGBD debe respetar el concepto de transacción como “**unidad de ejecución**”.

Esto significa el cumplimiento de cuatro propiedades:

- **Atomicidad:** es la esencia del concepto de transacción, o se ejecutan todas sus operaciones o no se ejecuta ninguna (la transacción tiene un carácter atómico).
- **Consistencia:** sólo se deben admitir transacciones que conduzcan a la base de datos a un estado consistente, es decir en el que se cumplen todas las restricciones de integridad del esquema. (Tema 2)
- **Aislamiento:** las transacciones de usuario se deben ejecutar como si se ejecutasen de forma aislada, sin la interferencia de otras transacciones de usuario. Los protocolos de control de la concurrencia de los SGBD aseguran este objetivo. (Tema 4)
- **Persistencia:** el SGBD debe asegurar (definición de transacción) que los efectos de una transacción confirmada definitivamente por el sistema quedarán grabados en la base de datos en disco. (Tema 3)

Procesamiento de transacciones y mantenimiento de la integridad.

- 1 Concepto de transacción.
- 2 Operaciones y estados de una transacción.
- 3 Propiedades del procesamiento de transacciones.
- 4 Definición de transacciones en SQL.
- 5 Concepto de restricción de integridad.
- 6 Comprobación de restricciones en SQL.

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

- INICIO: `START TRANSACTION*` (o inicio implícito).
- FIN (confirmación del usuario): `COMMIT [WORK]`
(transacción confirmada parcialmente)
- ANULACIÓN (anulación del usuario): `ROLLBACK [WORK]`
(transacción anulada)

En SQL no se pueden anidar las transacciones.

El inicio implícito de una transacción se realiza cuando se ejecuta una instrucción SQL (de DML) y no está ninguna transacción activa en ese momento.

*En SQL92 no existía la instrucción `START`, el inicio siempre era implícito.

24

El lenguaje SQL es el lenguaje estándar para los SGBD relacionales.

En la transparencia se presenta la sintaxis que SQL ofrece para las operaciones de usuario en la definición de transacciones: **inicio**, **fin** y **anulación**.

El inicio implícito de una transacción tiene lugar cuando después del inicio de la sesión de usuario, o después del final de una transacción, el usuario solicita una operación de manipulación (DML) sobre la base de datos, esta operación le indica al SGBD el inicio de la siguiente transacción.

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

- **ANULACION:** `ROLLBACK [WORK] [TO SAVEPOINT marca_savepoint]`
 - ✓ Las *marcas de savepoint* permiten el establecimiento de partes opcionales dentro de una transacción que podrán ser posteriormente (dependiendo de la lógica de la transacción) deshechas sin deshacer la transacción completa.
 - ✓ Una marca de *savepoint* se establece en una transacción con la sentencia `SAVEPOINT marca_savepoint`.
 - ✓ Se pueden establecer varias marcas de *savepoint* dentro de una transacción.

25

En SQL existe una variante de la operación de **anulación** de una transacción (ROLLBACK):
`ROLLBACK TO SAVEPOINT marca_savepoint`.

Las marcas de *savepoint* las define el usuario estratégicamente a lo largo de la transacción con la instrucción `SAVEPOINT`.

Estas marcas permiten al usuario hacer anulaciones parciales de una transacción con la instrucción `ROLLBACK TO SAVEPOINT marca_savepoint` (anular hasta una marca de *savepoint*). Esta variante tiene utilidad cuando las transacciones se ejecutan embebidas en programas donde la lógica del programa (estructuras condicionales) puede aconsejar deshacer parte de una transacción.

Las marcas de *savepoint* y la instrucción `ROLLBACK TO SAVEPOINT` permiten definir varias transacciones en una, la ejecución de cada una de ellas dependerá de las condiciones del contexto de ejecución. Puede verse como la definición de partes opcionales en una transacción.

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

- **ROLLBACK [WORK]:** finaliza la transacción y se deshacen todas las operaciones que se hayan ejecutado desde su inicio.
- **ROLLBACK TO SAVEPOINT *marca_savepoint*:** se deshacen todas las operaciones ejecutadas desde el establecimiento de *marca_savepoint* y la marca de savepoint es borrada. La transacción continúa en la instrucción siguiente al ROLLBACK ejecutado (no finaliza la transacción).

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

```
START TRANSACTION
```

```
...
```

```
...
```

```
SAVEPOINT marca1
```

```
...
```

```
...
```

```
SAVEPOINT marca2
```

```
...
```

```
...
```

```
IF ... THEN ROLLBACK TO SAVEPOINT marca2
```

```
...
```

```
...
```

```
IF ... THEN ROLLBACK TO SAVEPOINT marca1
```

```
...
```

```
COMMIT
```

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

- SET TRANSACTION modo [,modo] ...
modo:= nivel de aislamiento
| modo de acceso
| área de diagnóstico

Con la instrucción SET TRANSACTION el usuario puede dar al SGBD directrices sobre el procesamiento de transacciones durante su sesión de usuario.

Sesión de usuario: espacio de tiempo comprendido entre la conexión del usuario al SGBD y la desconexión.

28

La instrucción SET tiene distintas variantes en SQL y sirve para que el usuario pueda dar al SGBD directrices de funcionamiento durante su sesión de usuario.

Una sesión de usuario es el espacio de tiempo comprendido entre la conexión del usuario al SGBD y su desconexión.

La variante SET TRANSACTION sirve para dar al SGBD directrices relativas a la ejecución de las transacciones. Esta instrucción se debe ejecutar entre transacciones y el alcance de la directriz es la transacción siguiente.

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

SET TRANSACTION modo [,modo] ...

- ✓ El nivel de aislamiento especifica el nivel de control de la concurrencia que debe realizar el SGBD.
- ✓ El área de diagnóstico especifica el *número* máximo de condiciones de diagnóstico (errores o excepciones) que pueden registrarse relativas a la ejecución de las últimas instrucciones SQL:

SET TRANSACTION DIAGNOSTICS SIZE *número*

- ✓ El modo de acceso especifica el tipo de operaciones que se pueden ejecutar en una transacción: READ ONLY prohíbe operaciones de actualización de la base de datos:

SET TRANSACTION {READ ONLY | READ WRITE}

29

La instrucción SET TRANSACTION tiene tres argumentos (opcionales) cuyo significado se explica en la transparencia.

El argumento *nivel de aislamiento* se estudiará con más detalle en el Tema 10.

4 Definición de transacciones en SQL.

Definición de transacciones en SQL (operaciones de usuario):

SET TRANSACTION modo [,modo] ...

- ✓ La instrucción SET TRANSACTION no se puede ejecutar dentro de una transacción.
- ✓ La instrucción SET TRANSACTION debe ejecutarse antes del inicio de la transacción.
- ✓ **Alcance:** la ejecución de la instrucción SET TRANSACTION establece el modo de ejecución de la siguiente transacción.
- ✓ Existen unos valores por defecto para los tres modos: **READ WRITE**, **SERIALIZABLE**, y un tamaño del área de diagnóstico indeterminada.

Procesamiento de transacciones y mantenimiento de la integridad.

- 1 Concepto de transacción.
- 2 Operaciones y estados de una transacción.
- 3 Propiedades del procesamiento de transacciones.
- 4 Definición de transacciones en SQL.
- 5 Concepto de restricción de integridad.
- 6 Comprobación de restricciones en SQL.

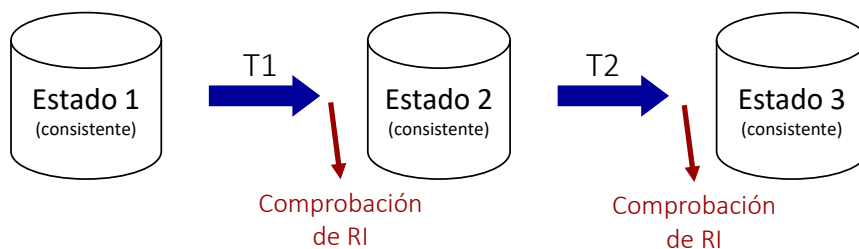
5 Concepto de restricción de integridad.

Restricción de Integridad



Propiedad que la base de datos debe satisfacer en cualquier instante de su historia.

- ✓ La BD evoluciona por la ejecución de transacciones de usuario.
- ✓ Las transacciones se consideran unidades de ejecución (**atomicidad**).
- ✓ Las restricciones "se deben" comprobar después de la ejecución de cada transacción (**consistencia**).



32

Como se comentó en el Tema 1, el sistema de información de una organización contiene toda la información relacionada con su actividad. Con la tecnología actual, el núcleo de cualquier sistema de información es una (o varias) bases de datos. Es decir una base de datos reúne (de forma estructurada) todos los datos producidos durante la actividad de la organización.

En cualquier organización existen reglas de funcionamiento que se deben cumplir en cualquier instante de tiempo. Estas reglas se traducen, en la base de datos, en forma de restricciones de integridad.

Una restricción de integridad es una regla (o norma) de la organización que la base de datos debe cumplir en cualquier instante de tiempo.

(Por ejemplo, la regla: *"todo empleado debe pertenecer a un departamento de la organización"* se representa en el ejemplo, al principio de este tema, con una definición de clave ajena en la tabla Empleado).

Como la base de datos sólo evoluciona por la ejecución de transacciones de usuario, las restricciones de integridad "deberán" ser comprobadas en cada nuevo estado de la base de datos, es decir, después de la ejecución de cada transacción.

5 Concepto de restricción de integridad.

Propiedades ACID:

- ✓ **Atomicidad:** una transacción es una unidad de ejecución (o se ejecutan todas sus operaciones o no se ejecuta ninguna de ellas).
- ✓ **Consistencia:** una transacción debe conducir la BD de un estado consistente a otro estado consistente (estado consistente: se cumplen todas las restricciones de integridad del esquema).
- ✓ **Aislamiento:** una transacción debe ejecutarse como si se ejecutase de forma aislada (en solitario).
- ✓ **Persistencia:** los cambios de una transacción confirmada por el SGBD deben quedar grabados permanentemente en la BD.

33

La comprobación de restricciones de integridad tiene que ver con las propiedades de **Atomicidad** y **Consistencia** del principio ACID de la ejecución de transacciones.

Procesamiento de transacciones y mantenimiento de la integridad.

- 1 Concepto de transacción.
- 2 Operaciones y estados de una transacción.
- 3 Propiedades del procesamiento de transacciones.
- 4 Definición de transacciones en SQL.
- 5 Concepto de restricción de integridad.
- 6 Comprobación de restricciones.

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL:

Desde un punto de vista teórico, las restricciones de integridad se deben comprobar cuando termina una transacción, en la práctica esta comprobación se puede relajar. Para ello, cada restricción de integridad del esquema tiene dos propiedades:

- ✓ **Modo:** define cuándo se comprueba la restricción y qué se hace si se viola.
 - Inmediato.
 - Diferido.
- ✓ **Propiedad de cambio:** determina la posibilidad de cambiar o no el **modo** definido.
 - Diferible.
 - No diferible.

35

Estas propiedades se definen para cada restricción pudiendo ser diferentes de una a otra y se especifican en la creación de las tablas junto a la definición de cada restricción de integridad en una cláusula que ya se introdujo en Bases de Datos (UD2.3) y que se denominó *cuándo_comprobar*. Recordamos la definición de una restricción de integridad (a nivel de atributo) en la notación BNF para ilustra dónde se incluiría la cláusula.

```
[CONSTRAINT nombre_restricción]
{NOT NULL | UNIQUE | PRIMARY KEY | CHECK(condición_búsqueda) |
  REFERENCES nom_relación [(nom_atributo)]
  [ON DELETE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}]
  [ON UPDATE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}]
[cuándo_comprobar]}
```

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL:

Modo inmediato:

- ✓ *¿Cuándo se comprueba?* Después de cada operación SQL que pueda violar la restricción.
- ✓ *¿Qué hace el SGBD si se viola?* Anula la instrucción SQL que ha provocado la violación y la transacción continúa.

Modo diferido:

- ✓ *¿Cuándo se comprueba?* Después de cada transacción que contenga una operación SQL que pueda violar la restricción.
- ✓ *¿Qué hace el SGBD si se viola?* Anula la transacción entera.

36

Existen dos modos de comprobación: inmediato y diferido con el significados que se expone en la transparencia.

Según el concepto de transacción, la comprobación de las restricciones de integridad se debería hacer al final de la transacción (modo diferido). La transacción es una unidad de ejecución y lo que importa es su efecto global.

La propuesta de dos modos de comprobación de la integridad responde a la búsqueda de una mayor flexibilidad en la ejecución de transacciones. Anular operaciones individuales de una transacción ofrece al usuario la oportunidad de corregir el error sin tener que esperar a la finalización de la transacción y su posible anulación.

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL:

Propiedad de cambio **diferible**:

- ✓ El modo de una restricción (inmediato o diferido) se puede cambiar dinámicamente durante la ejecución de una transacción. El cambio es local a la transacción, no modifica la definición de la restricción.

Propiedad de cambio **no diferible**:

- ✓ El modo de una restricción no se puede cambiar dinámicamente durante la ejecución de una transacción.

37

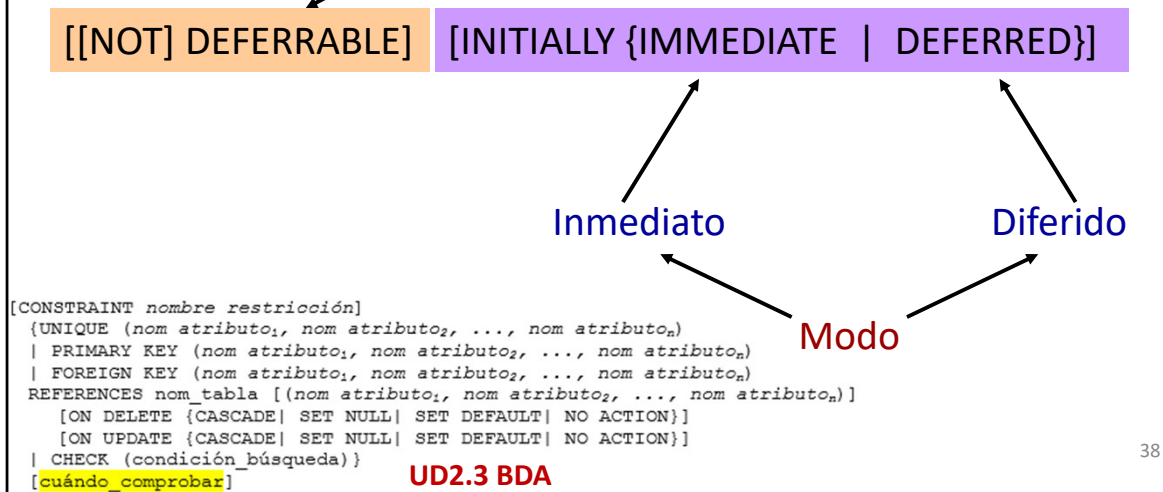
Cada restricción de integridad tiene un modo de comprobación inicial definido en el esquema de la base de datos. Este modo se puede cambiar dinámicamente o no durante la ejecución de las transacciones. Para poder cambiar el modo, la restricción debe haber recibido esa **propiedad de cambio** en su definición.

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL: sintaxis

Cláusula *cuándo_comprobar*

Propiedad del cambio: diferible/no diferible



38

En esta transparencia se presenta la cláusula *cuándo_comprobar* con la notación BNF que se usa para definir instrucciones SQL.

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL: sintaxis

La semántica de cada una de las versiones de la cláusula es:

1. Sin cláusula: si no se especifica nada en la definición, la restricción se define como no diferible y con modo inmediato. (Valor por defecto).
2. **DEFERRABLE INITIALLY IMMEDIATE**: define la restricción como diferible y con modo inmediato.
3. **DEFERRABLE INITIALLY DEFERRED**: define la restricción como diferible y con modo diferido.
4. **NOT DEFERRABLE INITIALLY IMMEDIATE**: define la restricción como no diferible y con modo inmediato. (Coincide con los valores por defecto).
5. **NOT DEFERRABLE INITIALLY DEFERRED**: esta versión está **prohibida**.
6. **DEFERRABLE**: define la restricción como diferible y con modo inmediato.
7. **NOT DEFERRABLE**: define la restricción como no diferible y con modo inmediato.
8. **INITIALLY IMMEDIATE**: define la restricción como no diferible y con modo inmediato.
9. **INITIALLY DEFERRED**: define la restricción como diferible y con modo inicial diferido.

39

Teniendo en cuenta que en la sintaxis BNF:

- los corchetes ([]) denotan partes opcionales,
- la llaves ({ }) elementos de los que se debe elegir uno, y
- la barra vertical (|) separa las distintas opciones posibles,

la cláusula *cuándo_comprobar* de una restricción de integridad se puede escribir de nueve formas posibles. En la transparencia se incluyen todas ellas y se explica el comportamiento que implican.

6 Comprobación de restricciones en SQL.

Comprobación de restricciones en SQL: cambio de modo

La instrucción SQL que permite cambiar, localmente en una transacción, el modo de una restricción definida como diferible (**DEFERRABLE**) es:

SET CONSTRAINT {restricción1, ... | ALL} {IMMEDIATE | DEFERRED}

- ✓ Cada restricción especificada en la lista debe ser diferible y la opción ALL hace referencia a todas las restricciones diferibles del esquema de la base de datos.
- ✓ El **alcance** del cambio producido por la instrucción SET CONSTRAINT es **la transacción** en la que se incluye o el fragmento de transacción hasta la siguiente aparición de la instrucción.
- ✓ Si se incluye la instrucción en medio de la transacción con la opción IMMEDIATE, las restricciones afectadas por la instrucción son comprobadas cuando se ejecuta ésta, si alguna de estas restricciones falla, la instrucción SET falla y el modo de las restricciones permanece sin modificar.

40

La variante de la instrucción SET de la forma **SET CONSTRAINT** permite cambiar dinámicamente (durante la ejecución de la transacción) el modo de comprobación de una (o varias) restricción de integridad, siempre que ésta se haya definido con la propiedad DEFERRABLE.

6 Comprobación de restricciones en SQL.

Ejemplo:

Empleado (dni: char(10), nombre: char(60), salario: real, dpto: char(5))

CP: {dni} VNN: {dpto}

CAj: {dpto} \rightarrow Departamento f(dpto)=código

Departamento (código: char(5), nombre: char(50), ubicación: char(15))

CP: {código} VNN: {nombre}

R1: “A todo departamento pertenece al menos un empleado”

41

Sea el esquema lógico de la base de datos relacional del ejemplo inicial del tema.

6 Comprobación de restricciones en SQL.

Ejemplo:

```
CREATE TABLE Empleado
(dni char(10) CONSTRAINT cp_emp PRIMARY KEY
    NOT DEFERRABLE INITIALLY IMMEDIATE,
nombre varchar2 (60),
salario number,
dpto char(5)  CONSTRAINT dpto_nulo NOT NULL
    NOT DEFERRABLE INITIALLY IMMEDIATE,
    CONSTRAINT caj_emp_dpto REFERENCES Departamento (codigo)
    DEFERRABLE INITIALLY DEFERRED);

CREATE TABLE Departamento
(codigo char(5) CONSTRAINT cp_dpto PRIMARY KEY
    NOT DEFERRABLE INITIALLY IMMEDIATE,
nombre varchar2(50) CONSTRAINT nombre_dep_nulo NOT NULL
    NOT DEFERRABLE INITIALLY IMMEDIATE,
ubicacion varchar2 (15));

CREATE ASSERTION R1 CHECK (NOT EXISTS
    (SELECT * FROM Departamento D
      WHERE NOT EXISTS (SELECT * FROM Empleado E
                        WHERE E.dpto=codigo)))
    DEFERRABLE INITIALLY DEFERRED;
```

42

Esta es la definición del esquema lógico anterior en el lenguaje SQL.

En los sistemas de gestión de bases de datos actuales no es posible la definición de asertos por lo que la restricción R1 debería ser controlada por programa.

6 Comprobación de restricciones en SQL.

Ejemplo:

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruíz	1500	d1
37418739	Luis Cerdá	1000	d1

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2

```
INSERT INTO Departamento VALUES (' d3 ','Personal','Planta3');
```

SGBD: comprobación de las RI: cp_dpto, nombre_dep_nulo

```
UPDATE Empleado SET dpto = 'd3' WHERE dni = ' 65743983';
```

SGBD: comprobación de la RI: dpto_nulo

```
COMMIT WORK;
```

SGBD: comprobación de las RI: caj_emp_dpto, R1

SGBD → CONFIRMAR (transacción)

RI en modo inmediato

RI en modo diferido



Transacción confirmada por el SGBD: sus cambios **serán grabados** en la BD **(a)**

43

En las transparencias siguientes se presentan ejemplos de ejecución de transacciones (SQL) sobre la base de datos de ejemplo, con la indicación de la comprobación de la integridad. Obsérvese el **modo** de comprobación de cada restricción y su efecto en la ejecución de la transacción.

6 Comprobación de restricciones en SQL.

Ejemplo:

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruiz	1500	d1
37418739	Luis Cerdá	1000	d1

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2

```
INSERT INTO Departamento (codigo, ubicacion) VALUES ('d5 ', 'Planta2')
```

SGBD: comprobación de la RI: cp_dpto,
comprobación de la RI: nombre_dep_nulo

RI en modo
inmediato

SGBD → ANULAR (instrucción)

...



Instrucción anulada por el SGBD (si los cambios ya han sido grabados **son deshechos**): la transacción puede continuar.

6 Comprobación de restricciones en SQL.

Ejemplo:

dni	nombre	salario	dpto
23125679	Juan García	2000	d2
65743983	Pedro Ruiz	1500	d1
37418739	Luis Cerdá	1000	d1

código	nombre	ubicación
d1	Compras	Planta1
d2	Ventas	Planta2

```
INSERT INTO Departamento VALUES ('d8 ', 'Proyectos', 'Planta1');
```

SGBD: comprobación de las RI: cp_dpto, nombre_dep_nulo

RI en modo inmediato

...

```
COMMIT WORK;
```

SGBD: comprobación de las RI: R1

RI en modo diferido

SGBD → ANULAR (transacción)



Transacción anulada por el SGBD: si los cambios ya han sido grabados **son deshechos (b)**.

6 Comprobación de restricciones en SQL.

Las restricciones **caj_emp_dpto** y **R1** son diferibles y con modo diferido. Si se cambiase su modo a inmediato no se podría actualizar la base de datos.

Ejemplo:

```
CREATE TABLE Empleado
(dni char(10) CONSTRAINT cp_emp PRIMARY KEY
    NOT DEFERRABLE INITIALLY IMMEDIATE,
nombre varchar2 (60) ,
salario number,
dpto char(5)  CONSTRAINT dpto_nulo NOT NULL
    NOT DEFERRABLE INITIALLY IMMEDIATE,
    CONSTRAINT caj_emp_dpto REFERENCES Departamento (codigo)
    DEFERRABLE INITIALLY DEFERRED);

CREATE TABLE Departamento
(codigo char(5) CONSTRAINT cp_dpto PRIMARY KEY
    NOT DEFERRABLE INITIALLY IMMEDIATE,
nombre varchar2(50) CONSTRAINT nombre_dep_nulo NOT NULL
    NOT DEFERRABLE INITIALLY IMMEDIATE,
ubicacion varchar2 (15));

CREATE ASSERTION R1 CHECK (NOT EXISTS
    (SELECT * FROM Departamento D
      WHERE NOT EXISTS (SELECT * FROM Empleado E
                        WHERE E.dpto=codigo)))
    DEFERRABLE INITIALLY DEFERRED;
```

46

Las dos restricciones del esquema que se han definido como DEFERRED (y por lo tanto con la opción de DEFERRABLE) son la restricción de clave ajena y la restricción R1. El modo DEFERRED para ambas restricciones asegura que se pueda actualizar la base de datos, y dar de alta un nuevo departamento.

6 Comprobación de restricciones en SQL.

Restricciones diferibles:

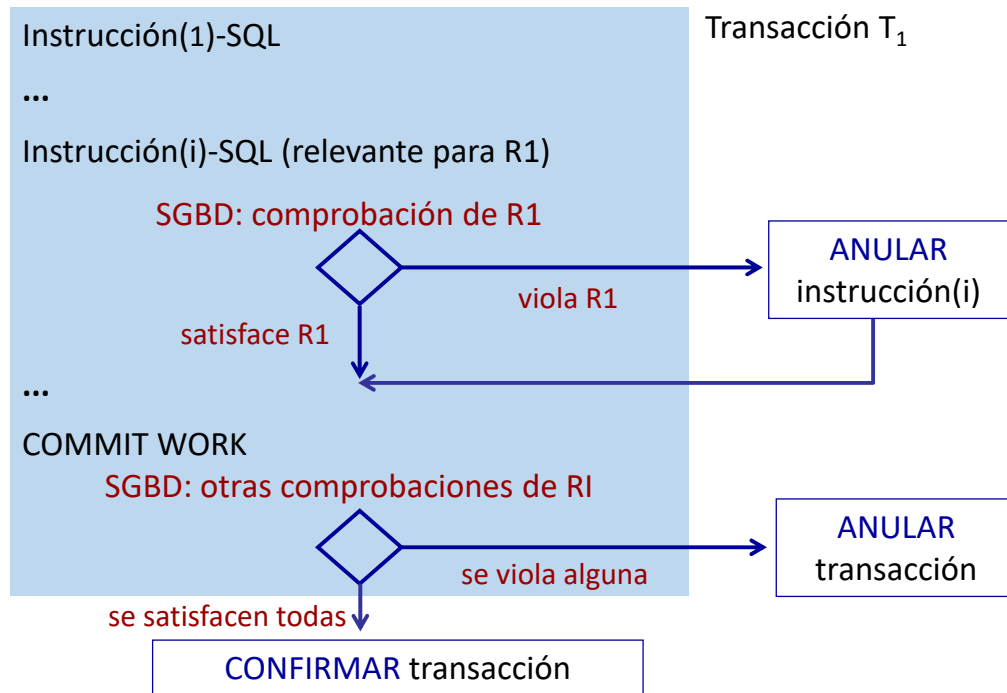
```
CREATE SCHEMA esquema1
...
CREATE ASSERTION R1 CHECK ( ... )
        DEFERRABLE INITIALLY IMMEDIATE
...
```

47

Para ilustrar el uso de la propiedad DEFERRABLE, se va a usar un ejemplo genérico, donde aparece definida una restricción R1, con modo de comprobación IMMEDIATE y con la propiedad DEFERRABLE.

6 Comprobación de restricciones en SQL.

Restricciones diferibles:

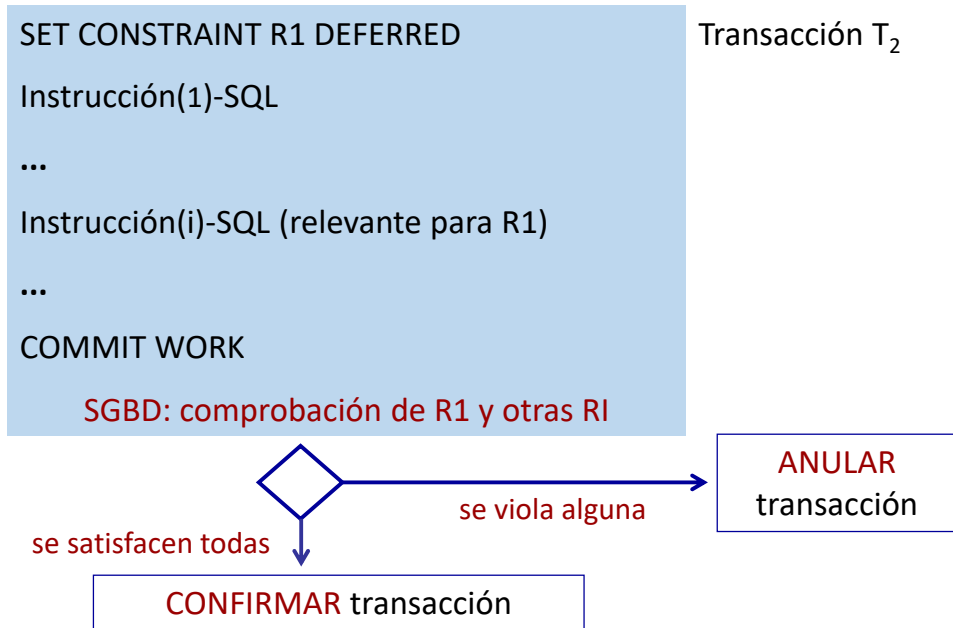


48

En la transparencia, se muestra que, como el modo inicial de R1 es IMMEDIATE, cuando se ejecuta una instrucción SQL, relevante para R1, el sistema comprueba la restricción y, en caso de violación, anula la instrucción (deshaciendo sus cambios) y la transacción puede continuar. Una operación es relevante para una restricción de integridad si su ejecución puede violar la restricción.

6 Comprobación de restricciones en SQL.

Restricciones diferibles:



49

En la transparencia, se muestra como el usuario cambia el modo de comprobación de R1 (instrucción SET). Cuando se ejecuta la instrucción SQL, relevante para R1, el sistema no hace ninguna comprobación, la restricción R1 se comprueba al final de la transacción, y, en caso de violación, se rechaza la transacción completa.

Los SGBDs hacen una comprobación de la integridad inteligente, sólo comprueban aquellas restricciones que son relevantes para las operaciones que se ejecutan en la transacción.