DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

Bases de Datos y Sistemas de Información

Grado en Ingeniería Informática

Unidad Didáctica 3:

Sistemas de Gestión de Bases de Datos

(Doc. UD3)

Curso 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

El contenido de este documento no debe considerarse una aportación novedosa ya que parte se ha realizado utilizando material, literalmente en muchos casos, del libro:

Celma, M.; Casamayor, J. C.; Mota, L.; *Bases de datos relacionales.* Pearson, Prentice Hall, 2003

El permiso explícito de los tres autores ha permitido su realización con el objetivo de facilitar el estudio de la materia de bases de datos para los alumnos de la ETSInf.

Este documento también contiene material libremente adaptado de otros libros, especialmente de:

- Connolly, T.; Begg, C.; Database systems: a practical approach to design, implementation and management, Fifth Edition, 2010.
- Elmasri, R.; Navathe, S.B.; Fundamentos de sistemas de bases de datos, 5ª edición.

Índice

En unidades didácticas anteriores se ha visto la manera de organizar los datos y sus restricciones (modelo de datos). Las herramientas encargadas de esta gestión son los Sistemas de Gestión de Bases de Datos (SGBD).

Existen una serie de aspectos tecnológicos, que determinan cómo los SGBD cumplen con sus funcionalidades en términos de acceso y modificación de los datos por parte de usuarios y aplicaciones, manipulación de los datos, gestión y administración de la base de datos y, fundamentalmente, la *independencia*, *integridad* y *seguridad* de los datos. En esta unidad didáctica se tratarán todos estos aspectos tecnológicos de los SGBD a un nivel básico, centrándonos en lo que es común a la mayoría de SGBD relacionales actuales.

1 ARQUITECTURA ANSI/SPARC

Como se vio en la unidad didáctica 1, un sistema de gestión de bases de datos es una herramienta de *software* que permite la creación y manipulación de bases de datos definidas de acuerdo con las reglas del modelo subyacente al sistema. En concreto, si el modelo de datos es relacional, se habla de *sistemas de gestión de bases de datos relacionales*.

Los sistemas de gestión de bases de datos proporcionan una interfaz entre los programas de aplicación que acceden a los datos y el sistema operativo, caracterizándose principalmente porque permiten una descripción unificada de los datos y la definición de vistas parciales de los mismos para distintos usuarios. Las propiedades que se deben exigir a todo SGBD para que satisfaga las expectativas depositadas en la tecnología de bases de datos son: el mantenimiento de la independencia, la integridad y la seguridad de los datos.

1.1 Esquemas y niveles de abstracción

En 1977, el grupo de estudio ANSI/SPARC[1] realizó una propuesta de arquitectura para los sistemas de gestión de bases de datos con el objetivo de separar las aplicaciones de usuario de las bases de datos físicas, para ello se planteó la definición de la base de datos en tres niveles de abstracción distintos: *conceptual*, *interno*, y *externo*. Las definiciones de la base de datos (esquema) en cada uno de estos niveles, deben tener las siguientes características: el esquema conceptual es la descripción del sistema de información, usualmente de forma gráfica, con independencia del SGBD concreto que se vaya a utilizar, el esquema interno es la descripción del esquema conceptual en términos de su representación física y, por último, los esquemas externos son las distintas vistas parciales que del sistema de información tienen los diferentes usuarios.

Desafortunadamente, en el estado actual de desarrollo de las técnicas de bases de datos, no existe ningún "modelo conceptual" accesible desde cualquier SGBD, es decir desde una representación gráfica del sistema de información es difícil llegar a definir las estructuras en almacenamiento secundario que representan adecuadamente ese sistema. Por este motivo, muchos autores prefieren añadir el esquema *lógico* a la propuesta ANSI/SPARC. Las definiciones de los diferentes esquemas son las siguientes:

- *Esquema conceptual*: descripción del sistema de información desde un punto de vista organizativo independiente del SGBD que se utilice e incluso de que se utilicen o no técnicas de bases de datos. En este esquema se pueden describir la información y las funciones de la organización desde un punto de vista no informático, con lenguaje natural o con lenguajes de modelado gráficos. En la unidad didáctica 4 se estudiará el diagrama de clases del UML para representar esquemas conceptuales.
- *Esquema lógico*: definición de la base de datos expresada en términos del modelo de datos en que se base el SGBD que se vaya a utilizar sin entrar en detalles de su representación física.
- *Esquema interno (físico):* definición de la representación de la base de datos en la memoria secundaria del computador. En este esquema no sólo se define la implementación elegida para las estructuras de datos del esquema lógico, sino que se especifican también otros detalles sobre la organización de la base de datos en el disco.
- *Esquema externo:* definición de los datos de interés para un grupo de usuarios. Cada esquema

[1] American National Standard Institute/System Planning And Requirement Committee.

externo consiste en un conjunto de estructuras (derivadas) definidas a partir de las estructuras del esquema lógico.

Ejemplo 1

El esquema lógico para la base de datos del sistema de información de DOCENCIA usado en la unidad didáctica 1 era el siguiente:

```
Departamento(cod_dep: char(4), nombre: char(50), teléfono: char(8),
              director: char(9))
   CP:{cod_dep}
   VNN:{nombre}
   CAj:{director}→Profesor(dni)
          Borrado a nulos y Modificación en cascada

Asignatura(cod_asg: char(5), nombre: char(50), semestre: char(2),
            cod_dep: char(4), teoría: real, prácticas: real)
   CP:{cod_asg}
   VNN:{nombre,semestre,cod_dep,teoría,prácticas}
   Uni:{nombre}
   CAj:{cod_dep}→Departamento(cod_dep)
          Borrado restrictivo y Modificación en cascada
   RI1:(teoría≤prácticas)
   RI2:(semestre ∈{'1A','1B','2A','2B','3A','3B','4A'4B'})

Profesor(dni: char(9), nombre: char(80), teléfono: char(8), cod_dep: char(4),
          provincia: char(25), edad: entero)
   CP:{dni}
   VNN:{nombre, cod_dep}
   CAj:{cod_dep}→Departamento(cod_dep)
          Borrado restrictivo y Modificación en cascada

Docencia(dni: char(9), cod_asg: char(5), gteo: entero, gpra: entero)
   CP:{dni,cod_asg}
   CAj:{dni}→Profesor(dni)
          Borrado en cascada y Modificación en cascada
   CAj:{cod_asg}→Asignatura(cod_asg)
          Borrado restrictivo y Modificación en cascada
   VNN:{gteo,gpra}

Restricción general:
   RG1: "Todo profesor debe impartir docencia de al menos una asignatura".
```

Por ejemplo, el esquema externo del departamento de código 'DMA', al que sólo le interesa información de sus profesores, de sus asignaturas y de la docencia en estas últimas, impartida por sus profesores podría ser, visto tabularmente el siguiente:

Asignatura-DMA				
cod_asg	**nombre**	**semestre**	**teoría**	**prácticas**
…	…	…	…	…

Profesor_DMA				
dni	**nombre**	**teléfono**	**provincia**	**edad**
…	…	…	…	…

Docencia_DMA			
dni	**cod_asg**	**gteo**	**gpra**
…	…	…	…

Para la definición de este esquema externo se utilizarían *vistas* (consultas almacenadas con un nombre)[2]. Así, las tablas de este esquema externo se definirían mediante las tres vistas siguientes:

```
CREATE VIEW Profesor-DMA
  AS   SELECT dni,nombre,teléfono,provincia,edad
       FROM Profesor
       WHERE cod_dep='DMA';

CREATE VIEW Asignatura-DMA
  AS   SELECT cod_asg,nombre,semestre,teoría,prácticas
       FROM Asignatura A
       WHERE cod_dep='DMA';

CREATE VIEW Docencia-DMA
  AS   SELECT D.dni,D.cod_asg,D.gteo,D.gpra
       FROM Profesor P,Docencia D,Asignatura A
       WHERE  P.cod_dep='DMA' AND A.cod_dep='DMA' AND
              P.dni=D.dni AND A.cod_asg=D.cod_asg;
```

Un esquema externo consiste en un conjunto de relaciones derivadas (vistas) definidas a partir de las relaciones básicas del esquema lógico. Estas relaciones derivadas no tienen existencia real, aunque pueden ser manipuladas por los usuarios del esquema externo como si fueran básicas. El SGBD frente a un requisito definido sobre estas relaciones virtuales traduce la consulta del usuario en una consulta sobre relaciones básicas, utilizando para ello la definición de la correspondiente vista. Por ejemplo, para saber el DNI y el nombre de los profesores del DMA que son de la provincia de Teruel y tienen más de dos grupos de teoría en alguna asignatura (también del DMA), se escribiría la siguiente consulta que accede a las vistas y no a las tablas básicas:

```
SELECT DISTINCT P.dni, P.nombre
FROM Profesor-DMA P, Docencia-DMA D
WHERE P.dni=D.dni AND P.provincia='Teruel' and D.gteo>2
```

Así como las definiciones del esquema lógico y del esquema externo presentados anteriormente son independientes del SGBD que se vaya a utilizar (se definirían en el lenguaje estándar SQL), la definición del esquema interno correspondiente dependerá de cada SGBD particular. La configuración del esquema interno o físico depende de un conocimiento más avanzado del SGBD en particular. Aunque existen comandos estándar en SQL para configurar ciertos aspectos del esquema interno: tipos de ficheros y su ordenación, índices primarios y secundarios, …, que permiten trasladar las decisiones que el diseñador ha realizado durante el diseño físico, otra (gran) parte de la configuración del esquema interno es muy específica del SGBD que se utilice. La configuración del esquema interno a veces es tarea compartida entre el diseñador y el administrador de la base datos (cuando no son la misma persona).

En definitiva, un SGBD que soporte la arquitectura de niveles debe:

- Permitir definir los distintos esquemas de la base de datos y, en cada uno de ellos, los aspectos que interesan de los datos y sus interrelaciones. Como excepción, el esquema conceptual se define exteriormente al SGBD en términos de algún modelo.
- Establecer las correspondencias entre los esquemas de forma que se pueda saber que un determinado dato en un esquema se corresponde con un determinado dato de otro esquema.
- Aislar los esquemas de manera que los cambios en un esquema no afecten a los esquemas de nivel superior y en última instancia a los programas de aplicación. Este es el concepto de *independencia de datos*.

1.2 Funcionamiento básico de un sistema de gestión de bases de datos

De los objetivos perseguidos con las técnicas de bases de datos, expuestos en la unidad didáctica 1, se

[2] Las vistas ya se han estudiado en la unidad didáctica 2.

pueden deducir cuáles son las funciones de un SGBD, y de estas funciones es sencillo inferir cuáles deben ser sus componentes básicos. En la tabla 1 se presenta un resumen de estos tres aspectos.

Las funciones de definición de esquemas y el establecimiento de las correspondencias entre ellos se verán con más detalle a lo largo de este apartado y las funciones de control de la integridad y la seguridad de la base de datos son el objetivo de los apartados 2 y 3.

Objetivos de las técnicas de BD	Funciones del SGBD	Componentes del SGBD
descripción unificada de los datos e independiente de las aplicaciones independencia de las aplicaciones respecto a la representación física de los datos definición de vistas parciales de los datos para distintos usuarios	definición de la base de datos a varios niveles: esquemas – esquema lógico (definición de las estructuras de la base de datos) – esquema interno (implementación de las estructuras del esquema lógico) – esquemas externos (definición de estructuras derivadas) establecer las correspondencias entre los esquemas	lenguajes para la definición de esquemas y los traductores asociados
gestión de la información	manipulación: consulta y actualización	lenguajes de manipulación y traductores asociados
	gestión y administración de la base de datos	herramientas para: - reestructuración - simulación, estadísticas - impresión
integridad y seguridad de los datos	control de: – la integridad semántica – los accesos concurrentes – la reconstrucción de la base de datos en caso de fallo – la seguridad	herramientas para: - control de la integridad - reconstrucción frente a fallos - control de la seguridad

Tabla 1: Objetivos, funciones y componentes de un SGBD

Para conseguir los objetivos anteriores, el SGBD se coloca entre las aplicaciones y los datos, constituyendo en la práctica una capa sobre el sistema operativo del computador.

Debido a las características de la base de datos relativas al volumen de datos y a su previsible existencia en un periodo dilatado de tiempo, ésta debe almacenarse en un dispositivo de memoria secundaria: discos magnéticos u otro tipo de memoria no volátil. Ahí es donde residen físicamente los datos. Pero para que una aplicación o usuario pueda llegar hasta los datos ha de pasar, inexorablemente, a través del SGBD. Éste es el principio general para que el SGBD tenga la certeza que nadie (usuario, proceso o aplicación) puede manipular la organización y contenido de los datos. Éste es el principio básico para poder elaborar la tecnología de independencia, integridad y seguridad que se verá en esta unidad didáctica.

La figura 1 ilustra el esquema de acceso del SGBD a los datos para satisfacer un requisito de consulta de una aplicación o usuario. Como se puede observar, existen muchos pasos y mecanismos, bajo estricto control del SGBD, entre la petición de acceso o modificación de una aplicación hasta la recuperación o modificación de los datos en algún dispositivo de almacenamiento secundario:

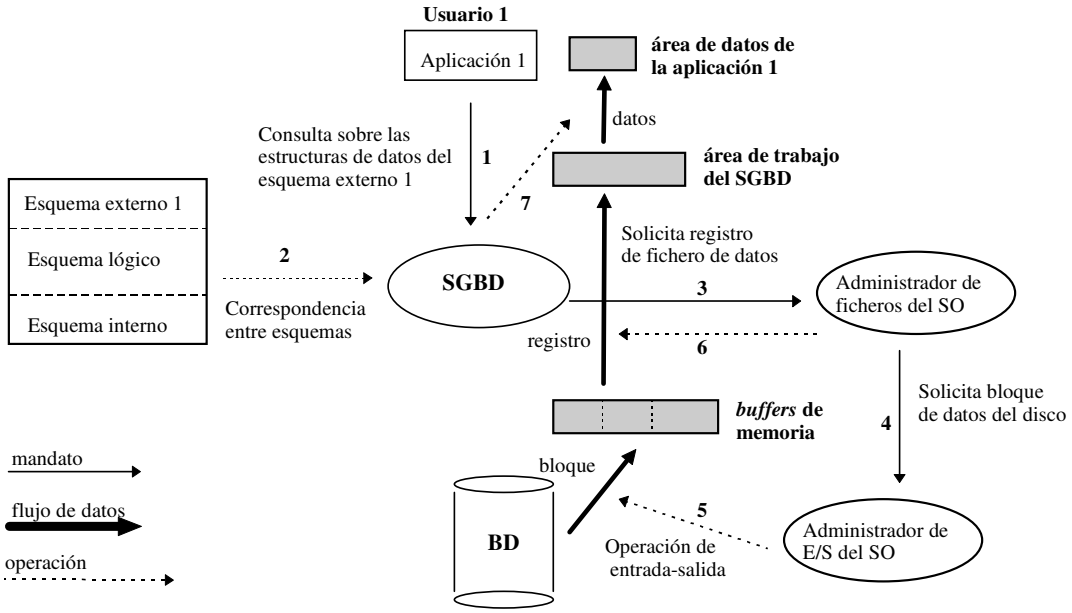


Figura 1: Esquema de acceso del SGBD a los datos.

- Paso 1: el programa de aplicación del usuario 1 realiza una consulta al SGBD.
- Paso 2: el SGBD consulta el esquema externo del usuario 1, el esquema lógico y el esquema físico, realiza la correspondencia entre ellos y traduce la consulta del usuario en una operación de lectura sobre un fichero.
- Paso 3: el SGBD emite órdenes al administrador de ficheros del SO indicando el registro que debe leer y en el fichero en el que está almacenado.
- Paso 4: el SO solicita al administrador de entrada/salida el bloque en el que se encuentra el registro requerido.
- Paso 5: el administrador de entrada/salida del SO recupera el bloque de datos solicitado y lo transfiere de la memoria secundaria a los *buffers* de memoria principal (si el bloque no estuviera ya en la memoria principal).
- Paso 6: el administrador de archivos del SO devuelve al SGBD el registro solicitado.
- Paso 7: el SGBD, comparando el esquema externo del usuario 1 y el esquema lógico, realiza las transformaciones necesarias para determinar los datos requeridos, y transfiere estos datos al área de trabajo del programa que ha realizado la consulta.

Si la consulta del usuario (paso 1) exige la recuperación de varios registros de datos los pasos del 3 al 6 se repetirán para cada uno de los registros necesitados. Recuperados estos registros, el SGBD los examina en memoria principal para ejecutar el paso 7.

1.3 Independencia de datos

Es conveniente empezar con una definición precisa del concepto de independencia de datos:

> En el ámbito de la tecnología de bases de datos, la independencia de datos es la propiedad que asegura que los programas de aplicación escritos por los usuarios sean independientes de los cambios realizados en datos que no usan o en los detalles de representación física (implementación) de los datos a los que acceden.

En la figura 2 se muestra la arquitectura de un SGBD y los niveles de independencia de datos que se comentan a continuación.

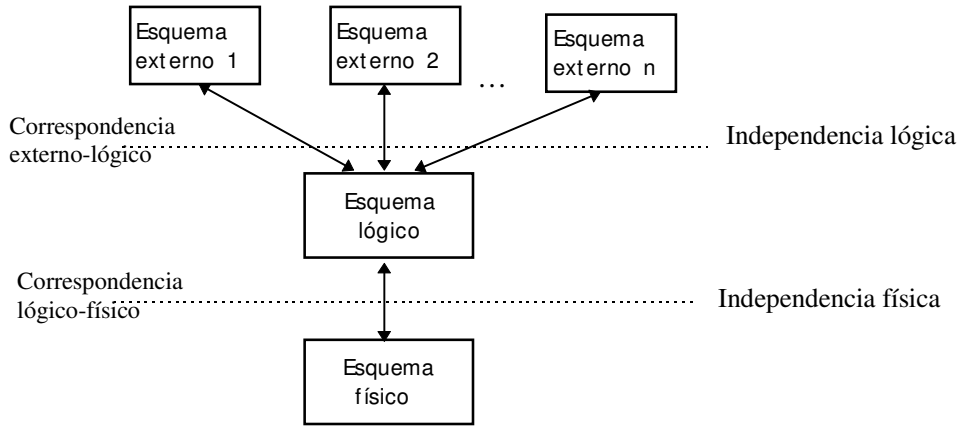


Figura 2: Arquitectura de niveles en un SGBD

- *Independencia lógica* entre el esquema lógico y los esquemas externos: los esquemas externos y los programas de aplicación no deben verse afectados por modificaciones del esquema lógico referentes a datos que no utilizan.
- *Independencia física* entre esquema lógico y esquema interno: el esquema lógico no debe verse afectado por cambios en el esquema interno referentes a la implementación de las estructuras de datos, los modos de acceso, el tamaño de las páginas y otros detalles de representación física.

Dicho de otra forma:

- La independencia lógica es la capacidad de cambiar el esquema lógico sin tener que cambiar los esquemas externos o los programas de aplicación. Es posible cambiar el esquema lógico para expandir la base de datos (añadiendo más tablas o atributos), para cambiar las restricciones o para reducir la base de datos (eliminando tablas o atributos) sin tener que cambiar los esquemas externos que sólo se refieran al resto del esquema lógico.
- La independencia física es la capacidad de cambiar el esquema interno sin que haya que cambiar el esquema lógico ni, por tanto, los esquemas externos. Puede que haya que realizar cambios en el esquema interno porque algunos ficheros físicos necesitan reorganización (cambio de disco, definición de más índices, …) para mejorar el rendimiento de las consultas o actualizaciones. Si en la base de datos permanecen los mismos datos que antes, no será necesario cambiar el esquema lógico.

Por regla general, la independencia física de datos existe en la mayoría de los sistemas ya que las cuestiones de almacenamiento secundario (ubicación de los ficheros en disco, definición de índices, compresión de datos, …) se mantienen ocultas al usuario. La independencia lógica sin embargo es más difícil de conseguir ya que permite cambios estructurales sin afectar a los programas de aplicación.

La independencia de datos ocurre porque cuando cambia un esquema en un nivel, los esquemas de niveles más altos permanecen inalterados, sólo cambia el *mapeado* entre los esquemas (pasos 2 y 3 del acceso a los datos de la figura 1).

Ejemplo 2

A continuación, se presentan algunos ejemplos de modificaciones del esquema lógico que no obligarían a modificar los esquemas externos en un SGBD con independencia lógica; los ejemplos hacen referencia al esquema del Ejemplo 1:

- Se puede ampliar la relación *Departamento* con más atributos, como la fecha de creación, la localización, etc., sin que esto afecte al esquema externo del DMA y por lo tanto a todos los programas que se hayan desarrollado sobre él,
- Si se restructura el esquema lógico reagrupando los datos en estructuras distintas a las originales, las aplicaciones desarrolladas sobre el esquema externo del DMA pueden no verse afectadas siempre que sea posible definir el mismo esquema externo a partir del nuevo esquema lógico, es decir siempre que se puedan definir las mismas relaciones (vistas) que lo componían, en este caso sólo será preciso cambiar la definición de las vistas.

2 TRANSACCIONES, INTEGRIDAD Y CONCURRENCIA

Uno de los objetivos de la tecnología de bases de datos es mantener la integridad de la base de datos, entendida como *calidad de la información* almacenada. Pero ¿qué se entiende por calidad de la información?

En un sistema de bases de datos, *los datos deben estar estructurados de forma que reflejen fielmente los objetos, las relaciones y las restricciones existentes en la parcela del mundo real del cuál la base de datos es una representación. Asimismo, y para que esta representación sea fiable, la base de datos debe ser sensible a los sucesos del mundo real, reflejando los cambios que la ocurrencia de éstos pueda provocar en la parcela del mundo representada.*

Pero ¿cómo se representan las relaciones y restricciones existentes en el mundo real? Esto se hace en el esquema de la base de datos, al definir las estructuras de datos y sus restricciones; y, ¿cómo se reflejan en la base de datos los cambios producidos por la ocurrencia de sucesos en el mundo real? Esto se hace por medio de las actualizaciones de los usuarios sobre la base de datos.

Por ello, con la expresión "calidad de la información almacenada" se pretende hacer referencia a estos dos aspectos de los datos:

a. El SGBD debe asegurar que los datos se almacenan correctamente, de acuerdo con las estructuras definidas, y que cumplan las restricciones especificadas en el esquema de la base de datos.
b. El SGBD debe asegurar que las actualizaciones de los usuarios sobre la base de datos se ejecutan correctamente y que se mantienen permanentemente.

De acuerdo con todo esto, para cumplir el objetivo de mantener la integridad de la base de datos, el SGBD debe disponer de herramientas y técnicas para:

a. Comprobar las restricciones de integridad, definidas en el esquema de la base de datos frente a cualquier actualización del usuario.
b. Controlar la correcta ejecución de las actualizaciones en un entorno concurrente.
c. Recuperar (o reconstruir) la base de datos en el caso de que se pierda la integridad por algún motivo.

Como se verá, la integridad de la base de datos se deteriora principalmente al realizar operaciones de actualización incorrectas sobre los datos, o debido a fallos lógicos o físicos (averías o accidentes). Un concepto fundamental para elaborar mecanismos para gestionar los tres puntos anteriores es el concepto de transacción.

2.1 Concepto de transacción

Como se comentó brevemente en la unidad didáctica 1, en un sistema de bases de datos, las operaciones de acceso a la base de datos, ya sean de lectura o de actualización, se organizan en *transacciones* que se diseñan agrupando operaciones que sólo tienen sentido si se ejecutan conjuntamente y no de forma aislada, por ello antes de estudiar las técnicas de reconstrucción de bases de datos es importante analizar el esquema de procesamiento de transacciones seguido por el SGBD.

> Una transacción es una secuencia de operaciones de acceso a la base de datos que constituye una unidad lógica de ejecución.

Además de las operaciones que constituyen propiamente la transacción, para un correcto procesamiento, el SGBD debe poder detectar el comienzo y el final de la transacción, así como la confirmación o anulación de ésta después de haber finalizado. Estas operaciones se indican explícitamente por medio de sentencias del lenguaje de manipulación o bien implícitamente a través de otras operaciones de acceso al SGBD. La semántica de cada una de estas operaciones es la siguiente (la sintaxis no se corresponde con la de ningún lenguaje real):

- *Principio*: indica el comienzo de la ejecución de la transacción.
- *Anulación* (*rollback de usuario*): indica que el usuario quiere abortar la transacción. El SGBD debe asegurar que ninguno de sus cambios queda grabado sobre la base de datos.

- *Confirmación* (*commit de usuario*): indica que el usuario da por finalizada la transacción. La confirmación de la transacción por parte del usuario no significa que sus cambios vayan a ser grabados definitivamente sobre la base de datos; en ese momento, el SGBD realizará comprobaciones (integridad, concurrencia, etc.) que determinarán si la transacción finaliza:
  - Con éxito (*commit del SGBD*): indica que la transacción ha finalizado con éxito. El SGBD asegurará que los cambios realizados por la transacción son grabados sobre la base de datos.
  - O fracasa (*rollback del SGBD*): la transacción no ha finalizado por algún motivo, o habiendo finalizado no ha cumplido algunas comprobaciones posteriores. El SGBD asegurará que ninguno de sus cambios es grabado sobre la base de datos.

Algunas de estas operaciones se realizan implícitamente, por ejemplo, la conexión al SGBD desde un programa de usuario pude implicar el inicio de una transacción, la terminación normal del mismo puede representar su confirmación, y la terminación anormal su anulación.

Las propiedades que debe poseer el correcto procesamiento de transacciones son:[3]

- *Atomicidad*: una transacción es una unidad atómica de ejecución en la que o se ejecutan todas sus operaciones o no se ejecuta ninguna.
- *Consistencia*: la transacción debe conducir a la base de datos de un estado consistente a otro estado consistente. Un estado *consistente* es aquel en el que se cumplen todas las restricciones de integridad especificadas en el esquema de la base de datos.
- *Aislamiento*: una transacción debe ejecutarse como si estuviera ejecutándose ella sola y no concurrentemente con otras transacciones.
- *Persistencia*: cuando una transacción es confirmada, sus cambios deben ser grabados sobre la base de datos y no deben perderse debido a fallos de otras transacciones o del sistema.

2.2 Integridad semántica

> Una *restricción de integridad* es una propiedad del mundo real del cual la base de datos es una representación. Para que esta representación sea consistente con la realidad, la base de datos debe satisfacer las restricciones en cualquier instante de su historia.

Como ya se ha visto en la unidad didáctica 1, muchas restricciones de integridad se definen en el esquema lógico de la base de datos siendo responsabilidad del SGBD velar por su cumplimiento. Si se considera que la base de datos evoluciona (cambia de estado) al ejecutar una transacción sobre ella, el SGBD debe comprobar las restricciones en el estado posterior a cada transacción, informando al usuario (o al programa) responsable de la misma del resultado de la comprobación.

Puede ocurrir, sin embargo, que haya alguna restricción de integridad que no se pueda definir en el esquema de la base de datos siendo responsabilidad de los programadores el introducir en el código de los programas de acceso a la base de datos, los controles necesarios para su comprobación. Esta situación es, por lo general, inadecuada, ya que pueden existir inconsistencias en las restricciones de integridad, al distribuir y diluir la responsabilidad de mantener las restricciones. Por tanto, para aquellas restricciones asociadas a la información, en general, es preferible, introducir estas restricciones en el SGBD.

Las restricciones que pueden definirse sobre una base de datos se clasifican en función del número de estados de la base de datos que se ven implicados en la propiedad representada por la restricción.

- Restricciones *estáticas*: expresan propiedades que deben satisfacerse en cada estado de la base de datos.
- Restricciones de *transición*: expresan propiedades que se deben cumplir en cada par de estados consecutivos.

[3] A este conjunto de propiedades se le denomina entorno ACID de sus siglas en inglés (*Atomicity, Consistency, Isolation and Durability*).

Ejemplo 3

Las siguientes dos propiedades corresponden a una restricción estática y una restricción de transición[4]:

- Restricción estática: "Los créditos de teoría de una asignatura no pueden superar los créditos de prácticas."
- Restricción de transición: "Los créditos totales de una asignatura no pueden decrecer."

Las restricciones de integridad estáticas pueden expresarse con algún tipo de expresión lógica (p.ej., expresada en SQL). Sin embargo, los sistemas relacionales ofrecen una sintaxis particular para expresar algunos tipos de restricciones muy frecuentes de una manera más sencilla y abreviada. Así, en SQL se pueden expresar restricciones de integridad de varios tipos:

- Restricciones sobre valores posibles de los atributos.
- Restricciones sobre atributos.
- Restricciones sobre relaciones.

Las restricciones que no entran en los apartados anteriores se denominan

- Restricciones generales sobre la base de datos.

En el SQL la definición de las restricciones estáticas es declarativa, es decir en la definición se expresa con una notación (más o menos abreviada) la propiedad en que consiste la restricción, encargándose el SGBD de determinar las operaciones de actualización que pueden ser relevantes para la restricción, así como de determinar la forma más eficiente de comprobarla. Para cualquier tipo de restricción, se puede indicar al sistema el instante en el cual debe comprobarse la restricción, al finalizar la transacción o después de cada operación individual de la misma. En algunos casos (definición de claves ajenas) la sintaxis del SQL permite expresar también algunas acciones compensatorias que el sistema debe realizar, en el caso en que se viole la restricción, para seguir manteniendo la integridad en la base de datos.

Ejemplo 4

En el esquema relacional del Ejemplo 1 visto al principio de esta unidad didáctica, aparecen restricciones de los siguientes tipos:

- Restricciones sobre valores posibles de los atributos: al asociar a un atributo un tipo de datos concreto se limitan los valores posibles que pueden tomar:
  `cod_asg:char(5)`
- Restricciones sobre atributos, por ejemplo, en la definición del atributo *nombre* de la relación *Profesor:*
  `VNN:{nombre}`
- Restricciones sobre relaciones, por ejemplo, las definiciones de clave primaria y clave ajena en la relación *Asignatura*:
  `CP:{cod_asg}`
  `CAj:{cod_dep}` → `Departamento(cod_dep)`
- Restricciones generales sobre la base de datos:
  "`Todo profesor debe impartir docencia de al menos una asignatura`".

  Hoy por hoy, los SGBD no permiten la definición de estas restricciones generales por lo que, como ya se ha dicho, su comprobación debería integrarse en las aplicaciones que manipulan la base de datos.

2.3 Control de accesos concurrentes

Como se ha comentado anteriormente, para mantener la integridad de la base de datos el SGBD debe controlar los accesos concurrentes evitando que los resultados de la ejecución de un programa sean

[4] El estudio de estas restricciones requiere el estudio de los *disparadores* que no se van a presentar en la asignatura.

incorrectos, incoherentes o se pierdan debido a la ejecución concurrente de otro programa que accede a los mismos datos.

De una manera más exhaustiva, algunas de las interferencias que pueden producirse entre programas que actualizan la base de datos son las siguientes:

a. Pérdida de actualizaciones. (Ver Ejemplo 5).
b. Obtención de información incoherente correspondiente a varios estados válidos de la base de datos. (Ver Ejemplo 6).
c. Lectura de datos actualizados (no confirmados) que han sido sometidos a cambios que todavía pueden ser anulados. (Ver el Ejemplo 7).

A continuación, se presentan ejemplos de cada uno de los tres tipos de interferencias.

Ejemplo 5

Ejecución concurrente de los programas *P1* y *P2* que leen y modifican los créditos de teoría de la asignatura de código 11548:

Tiempo	P1	P2
t1	leer(11548, teoría) teoría=4,5	
t2		leer(11548, teoría) teoría=4,5
t3	teoría←teoría+1,5 teoría=6	
t4		teoría←teoría+2 teoría=6,5
t5	escribir(11548, teoría)	
t6		escribir(11548, teoría)

Da como resultado:

Asignatura					
cod_asg	nombre	semestre	cod_dep	teoría	prácticas
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	4,5	1,5

P1 y P2

Asignatura					
cod_asg	nombre	semestre	cod_dep	teoría	prácticas
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	6,5	1,5

En el ejemplo se ilustra cómo el acceso concurrente (no controlado) de dos programas a los mismos datos puede causar la pérdida de actualizaciones. En este caso, después de las dos actualizaciones, el valor del atributo *teoría* de la asignatura 11548 debería valer 8 (*P*1 le suma 1,5 y *P*2 le suma 2), sin embargo, vale 6,5 porque se ha perdido la actualización de *P1*.

Ejemplo 6

Se está ejecutando un programa que lista los créditos impartidos por cada profesor teniendo en cuenta cuántos grupos da de una asignatura y cuántos créditos tiene esa asignatura. Supongamos el siguiente estado de la base de datos:

Profesor					
dni	nombre	teléfono	cod_dep	provincia	edad
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
453	Elisa Rojo Amando	3412	DSIC	Castellón	26
564	Pedro Martí García	3412	DMA	Castellón	27

Docencia			
dni	cod_asg	gteo	gpra
111	11545	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2

Asignatura			
cod_asg	...	teoría	prácticas
11545	...	4,5	1,5
11546	...	4,5	1,5
11547	...	4,5	1,5
11548	...	4,5	1,5

Tiempo	P1	P2
t1	Calcular créditos del profesor 111 Créditos 111= 9 (1×4,5 + 3×1,5)	
t2	Calcular créditos del profesor 123 Créditos 123= 9 (0×4,5 + 2×1,5 + 1×4,5 + 1× 1,5)	
t3		Pasar uno de los grupos de teoría del profesor 564 en la 11545 al profesor 111
t4	Calcular créditos del profesor 453 Créditos 453= 0	
t5	Calcular créditos del profesor 564 Créditos 564= 7,5 (1×4,5 + 2×1,5)	

Docencia

dni	cod_asg	gteo	gpra
111	11545	2	3
123	11545	0	2
123	11547	1	1
564	11545	1	2

En definitiva, el listado obtenido por el proceso *P*1 es el siguiente:

111 Luisa Bos Péres 9 créditos
123 Juana Cerdá Pérez 9 créditos
453 Elisa Rojo Amando 0 créditos
564 Pedro Martí García 7,5 créditos

que como puede observarse es un resultado incorrecto si consideramos el estado previo a la ejecución de *P*2 que devolvería el listado:

111 Luisa Bos Péres 9 créditos
123 Juana Cerdá Pérez 9 créditos
453 Elisa Rojo Amando 0 créditos
564 Pedro Martí García 12 créditos

y también es incorrecto si consideramos el estado posterior a la ejecución de *P*2 que devolvería el listado:

111 Luisa Bos Péres 13,5 créditos
123 Juana Cerdá Pérez 9 créditos
453 Elisa Rojo Amando 0 créditos
564 Pedro Martí García 7,5 créditos

Se ha producido una anomalía ya que se ha obtenido información incoherente obtenida a partir de dos estados válidos (los datos almacenados son correctos).

Ejemplo 7

Ejecución concurrente de los programas *P1* y *P2* que leen y/o modifican los créditos de teoría de la asignatura de código 11548:

Tiempo	P1	P2
t1	leer(11548, teoría)	
t2	teoría←teoría+1,5	
t3	escribir(11548, teoría)	
t4		leer(11548, teoría)
		teoría = 6
t5		usa este valor en sus instrucciones
t6		confirmar
t7	anular	

Al haberse anulado P_1 todas sus actualizaciones deben deshacerse devolviendo la base de datos al estado anterior, sin embargo, P_2 ha leído y utilizado esos datos que aún no estaban confirmados. (Uso de datos actualizados cuyos cambios no han sido confirmados).

Entre las técnicas utilizadas más frecuentemente para evitar los problemas derivados de los accesos

concurrentes están las técnicas basadas en la *reserva de ocurrencias de datos*. Estas técnicas limitan la simultaneidad de los accesos concurrentes. Así, en el caso del Ejemplo 5 y del Ejemplo 7 el programa P_1 debería reservar el registro desde que lo lee hasta después de confirmar la actualización; en el Ejemplo 6, se deberían reservar todas las tablas. Los programas deben solicitar las reservas a un módulo del SGBD que se encarga de llevar el control de las mismas. Lo contrario de la reserva es la liberación que cancela una reserva efectuada.

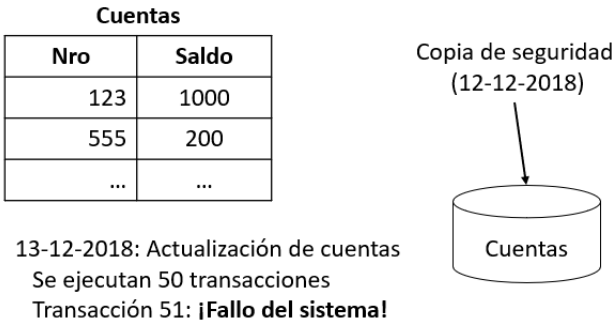
3 RECUPERACIÓN Y SEGURIDAD

En este apartado se examinarán dos aspectos que la tecnología actual de bases de datos considera irrenunciables: una base de datos ha de poderse recuperar ante prácticamente cualquier tipo de fallo y una base de datos no puede tener accesos no autorizados. La primera parte, dedicada a recuperación, entra dentro del ámbito de la "integridad" de datos, pero desde un punto de vista distinto al de la consistencia, visto en el apartado anterior, más centrado en la durabilidad y persistencia de los datos. La segunda parte se dedica al concepto de "seguridad" de datos.

3.1 Reconstrucción de la base de datos

Se puede pensar que una manera adecuada y suficiente de hacer frente a pérdidas de datos puede ser el simple uso de copias de seguridad. Aunque las copias de seguridad cobran un papel importante en la reconstrucción de una base de datos ante pérdidas de datos, no son suficientes (por sí mismas) para las exigencias actuales en este ámbito. El ejemplo siguiente muestra una visión anticuada del proceso de reconstrucción:

Ejemplo 8



- Procedimiento de recuperación: sustituir el fichero de *Cuentas* por su copia de seguridad.
- Efecto negativo: se han perdido las actualizaciones de 50 transacciones.

En el ejemplo se ilustra el procedimiento de recuperación de datos seguido en sistemas de gestión de datos primitivos (sistemas de ficheros). Como se ve, se trata únicamente de recuperar la última copia de seguridad y se asume que se ha perdido un número importante de actualizaciones. Aumentar la frecuencia de copias de seguridad no es una solución viable.

La tecnología de bases de datos debe proporcionar mecanismos de recuperación de la base de datos más potentes, que permitan devolver a la base de datos a un estado íntegro lo más cercano posible en el tiempo al momento en que se produjo el fallo. Este objetivo se complica si se tiene en cuenta que es posible que cambios realizados por transacciones confirmadas no hayan sido llevados a disco, pese a estar confirmados, porque los cambios realizados están en memoria principal y los bloques que los contienen no se han grabado aún en disco. Por otra parte, también es posible que cambios realizados por transacciones no confirmadas sí que se hayan llevado a disco al haber sido grabados los bloques de memoria principal que los contienen. Qué bloques de memoria principal se llevan a disco y cuándo es una tarea que realiza el SGBD y que depende de la política de transferencia de bloques que se le haya definido.

En concreto, hoy en día se exige un concepto de recuperación total, en el sentido de que las propiedades de atomicidad y persistencia de una transacción obligan al SGBD a asegurar que:

- Los cambios producidos por cualquier transacción confirmada sean grabados sobre la base de datos y una vez grabados no se pierdan, y que

- Los cambios producidos por cualquier transacción fallada que ya han sido grabados sobre la base de datos sean deshechos.

Las causas por las que una transacción confirmada no es grabada completamente sobre la base de datos tienen su origen en el desajuste temporal que existe entre el instante en que se confirma la transacción y el instante en que se copian los bloques con los datos actualizados del *buffer* (o *buffers*) de memoria principal al disco ya que la confirmación de una transacción por parte del usuario, como ya se ha dicho anteriormente, no implica que las modificaciones que haya realizado la transacción sean llevadas a disco. En el intervalo que transcurre entre ambos instantes de tiempo pueden producirse diversos fallos del sistema que provoquen la *pérdida de memoria principal* y que causen que algunos de esos bloques no lleguen a ser copiados nunca.

Por otra parte, las causas por las que una transacción confirmada, cuyos cambios ya habían sido grabados sobre la base de datos, se pierde son debidas a la ocurrencia de fallos que provoquen la *pérdida de memoria secundaria*.

Resumiendo, entre las causas externas a la transacción que pueden provocar la pérdida de la integridad de la base de datos, se distinguen:

- *Fallos del sistema que provocan la pérdida de memoria principal*: interrupción del suministro eléctrico, error del software del SGBD, error del sistema operativo, virus informáticos, etc., y
- *Fallos del sistema de almacenamiento que provocan la pérdida de memoria secundaria*: aterrizaje de las cabezas de lectura de los discos, fallos del controlador de discos, accidentes o catástrofes, virus informáticos, etc.

A continuación, se van a presentar de forma simplificada las técnicas utilizadas por el SGBD para reconstruir la base de datos frente a fallos de ambos tipos (al final del apartado se comentarán las simplificaciones asumidas).

3.1.1 Reconstrucción frente a fallos del sistema con pérdida de memoria principal

Para poder recuperar transacciones confirmadas que no han sido grabadas completamente o anular transacciones falladas, el SGBD debe disponer de herramientas que registren las transacciones que tienen lugar sobre la base de datos; asimismo debe disponer de técnicas que, con esa información, devuelvan la base de datos a un estado íntegro en el que se pueda asegurar la calidad de la información almacenada. El módulo encargado de realizar estas tareas es el módulo de reconstrucción. A continuación, se presentan estas herramientas y técnicas.

El método más extendido es el de la utilización de un fichero denominado *diario* (*log* o *journal*). En el diario se guarda información sobre todas las operaciones de actualización de las transacciones. El diario se almacena en disco para evitar su desaparición por una caída del sistema y periódicamente se copia en otro sitio (disco óptico o magnético, cinta magnética, etc.) para prevenir posibles accidentes que deterioren el disco.

Cuando una transacción falla (por causas locales a la transacción o por fallo del sistema) el módulo de reconstrucción puede utilizar la información grabada en el diario para deshacer sus efectos.

De la misma forma, cuando se produce un error del sistema con pérdida de memoria principal, el módulo de reconstrucción debe volver a ejecutar las transacciones que en el diario aparecen como confirmadas, ya que no se tiene la seguridad de que sus cambios hayan sido grabados sobre la base de datos.

Si el diario recoge información sobre todas las transacciones que han tenido lugar en un periodo de tiempo prolongado (por ejemplo, desde que se arrancó el sistema por la mañana) el proceso de recuperación puede ser muy costoso, ya que obliga a volver a ejecutar todas las transacciones confirmadas que estén registradas en el diario, aunque probablemente muchas de ellas ya habrán sido grabadas sobre la base de datos. Para evitar costosos procesos de recuperación se graban en el diario lo que se conoce como *puntos de control* (*checkpoint*).

Los puntos de control (o de verificación) se graban en el diario periódicamente. Grabar un punto de control significa:

- Suspender temporalmente la ejecución de transacciones,

- Grabar en el diario el punto de control, es decir, anotar en qué momento se ha ejecutado el punto de control,
- Forzar la grabación de todas las actualizaciones de las transacciones confirmadas, copiando los correspondientes bloques de los *buffers* de memoria principal al disco, y
- Reanudar la ejecución de las transacciones que han sido suspendidas.

Con esta técnica, el proceso de reconstrucción ante la pérdida de memoria principal se inicia a partir del último punto de control grabado en el diario.

En la figura 3 se representan siete transacciones en estados distintos cuando se produce un error del sistema con pérdida de memoria principal.

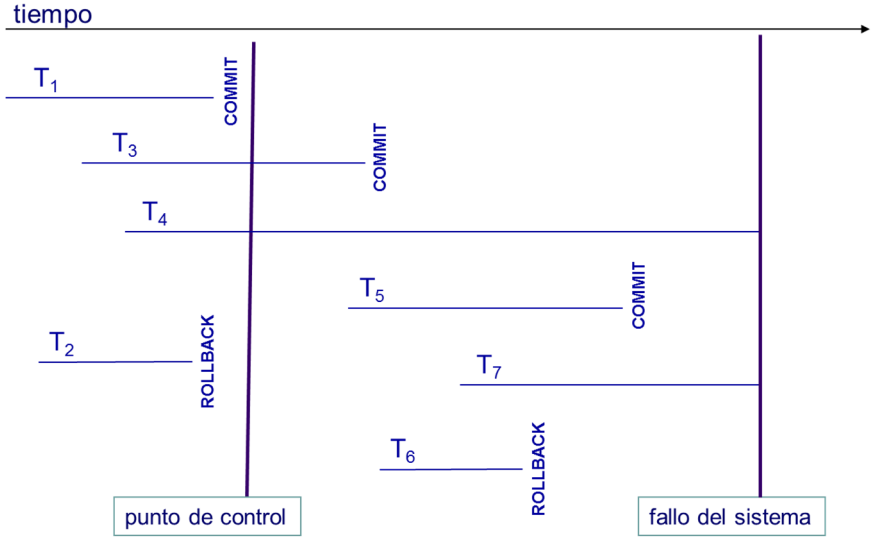


Figura 3: Toma de puntos de control.

Recordando que:

1. Cambios realizados por transacciones confirmadas puede que no hayan sido llevados a disco porque los bloques que los contienen no se han grabado aún en disco.
2. Cambios realizados por transacciones no confirmadas sí que puede que se hayan llevado a disco al haber sido grabados los bloques de memoria principal que los contienen.
3. Cuando se realiza un punto de control, el sistema graba en disco todos los cambios realizados por transacciones confirmadas hasta ese momento.

Entonces, la situación en la que está cada transacción en el momento del fallo del sistema con pérdida de memoria principal es la siguiente:

- La transacción *T*1 había sido confirmada antes del último punto de control, por lo tanto, se tiene la seguridad de que sus cambios han sido grabados permanentemente sobre la base de datos.
- La transacción *T*2 fue anulada antes del fallo del sistema por lo que sus cambios, de haber llegado a disco, fueron anulados en ese momento.
- La transacción *T*3 se inició antes del último punto de control y se confirmó después de él, antes de producirse el fallo del sistema, por lo tanto, no se puede asegurar que todos sus cambios hayan sido grabados en disco.
- La transacción *T*4 se inició también antes del último punto de control y no había finalizado cuando se produjo el fallo del sistema, por lo tanto, deberá ser anulada automáticamente cuando se reinicie el sistema y deberán deshacerse todos sus cambios.
- La transacción *T*5 se inició después del último punto de control y se confirmó antes de producirse el fallo, por lo tanto, no se puede asegurar que sus cambios hayan sido grabados sobre la base de datos.
- La transacción *T*6 fue anulada antes del fallo del sistema por lo que sus cambios, de haber llegado a disco, fueron anulados en ese momento.
- Por último, la transacción *T*7 se inició después del último punto de control y no había finalizado al

producirse el fallo, por lo tanto, deberá ser anulada automáticamente cuando se reinicie el sistema y deberán deshacerse todos sus cambios.

Así pues, la reconstrucción de la base de datos, al volver a iniciarse el sistema, implicará:

- Anular las transacciones *T*4 y *T*7 y deshacer los cambios que pudiesen haber producido.
- Recuperar (volver a ejecutar) las transacciones *T*3 y *T*5.
- Con las transacciones *T*1, *T*2 y *T*6 no hay que hacer nada, en el primer caso porque se tiene la certeza de que sus cambios llegaron a disco (se confirmó antes del punto de control) y en los otros dos porque se tiene la certeza de que sus cambios no están en disco (fueron anuladas antes del fallo del sistema).

Con estas herramientas, diario y puntos de control, se consigue una recuperación de la base de datos mucho más eficiente que utilizando simplemente las copias de seguridad.

3.1.2 Reconstrucción frente a fallos del sistema de almacenamiento

En el caso de que se produzca un fallo del sistema de almacenamiento, con pérdida de memoria secundaria, la base de datos puede quedar dañada total o parcialmente. En estos casos la reconstrucción de la base de datos exige el uso de la copia de seguridad de la base de datos y del diario no siendo útiles los puntos de control.

La técnica de reconstrucción que se sigue consiste en recrear la base de datos actual a partir de la copia de seguridad más reciente y de la información (histórica) grabada en el diario desde el instante de tiempo en que se realizó dicha copia. Para ello se carga la base de datos a partir de la copia de seguridad y a continuación se rehacen todas las transacciones que aparecen confirmadas en el diario desde la fecha de la copia. Es interesante destacar que en este tipo de reconstrucción no es necesario deshacer las transacciones que aparecen anuladas en el diario, ya que sus cambios han desaparecido al utilizar una copia de seguridad anterior que se supone íntegra.

3.1.3 Consideraciones sobre la reconstrucción de las bases de datos

En toda la exposición que se ha hecho sobre la reconstrucción de la base de datos frente a fallos, se han asumido por simplicidad, las siguientes hipótesis de trabajo:

- Las anotaciones en el diario se hacen directamente sobre el disco.
- La técnica de recuperación utilizada permite que las actualizaciones de la transacción se graben en la base de datos antes de que la transacción sea confirmada (*actualización inmediata*).
- La ejecución concurrente de las transacciones se realiza de forma que cada transacción se puede recuperar independientemente de las otras transacciones.

Obviamente, ninguna de estas hipótesis es necesaria.

Respecto a la grabación de las entradas en el diario, éstas pueden permanecer durante algún tiempo en *buffers* de memoria principal, evitando de esta forma los accesos frecuentes al disco. En esta situación el sistema de recuperación debe asegurarse de que cualquier cambio sobre la base de datos (física) es registrado previamente en el diario (físico).

Respecto a las técnicas de reconstrucción, éstas pueden estar basadas en la *actualización inmediata* o en la *actualización diferida*. Las técnicas con actualización diferida no permiten la actualización de la base de datos hasta después de que la transacción haya sido confirmada. En este caso el proceso de recuperación ante un fallo del sistema es distinto; las transacciones que no aparecen confirmadas en el diario no deben ser deshechas ya que se tiene la seguridad de que sus cambios no han sido grabados; respecto a las transacciones que aparecen confirmadas en el diario deberán ser rehechas ya que no se tiene la seguridad de que hayan sido ya grabadas.

Considerando las transacciones de la Figura 3, la reconstrucción de la base de datos en un entorno con actualización diferida supondría:

- Recuperar (volver a ejecutar) las transacciones *T*3 y *T*5.

- Con las transacciones *T*1, *T*2, *T*4, *T*6 y *T*7 no hay que hacer nada, en el primer caso porque se tiene la certeza de que sus cambios llegaron a disco (se confirmó antes del punto de control) y en los restantes porque se tiene la certeza de que sus cambios no están en disco porque al ser la actualización diferida, los cambios sólo pueden empezar a llegar después de la confirmación de la transacción.

Respecto a la recuperación independiente de las transacciones, sólo se puede asegurar cuando el procesamiento concurrente de las transacciones se realiza con ciertas restricciones, así en el Ejemplo 7, en el que el proceso *P*2 leía un dato modificado por el proceso *P*1 que después era anulado, la anulación de *P*1 obligaría no sólo a deshacer los cambios que ese proceso hubiera podido realizar sobre la base de datos sino a anular también *P*2 que ya había sido confirmada (*anulación en cascada*).

Una restricción que se puede imponer al procesamiento concurrente para evitar el problema anterior consiste en no permitir que una transacción lea ni actualice un dato que ha sido actualizado por otra transacción hasta que esta última sea confirmada o anulada (*procesamiento estricto*).

3.2 Seguridad

Este objetivo de los SGBD consiste en asegurar que a la información almacenada sólo pueden acceder las personas autorizadas y en la forma autorizada. Algunas técnicas utilizadas para cubrir este objetivo son las siguientes:

- Identificación de usuario: consiste en determinar quién es la persona que está solicitando acceder a la base de datos como paso previo a la determinación de qué tipos de accesos tiene permitidos. La identificación se realiza normalmente por medio de una clave de acceso.
- Determinación de accesos permitidos: básicamente, existen dos formas de establecer los accesos permitidos para cada usuario de la base de datos:
  - Cada usuario tiene asociada una lista de autorizaciones que denotan los datos accesibles y las operaciones permitidas sobre cada uno de ellos.
  - Definición de distintos niveles de autorización, lo que significa que todos los usuarios con el mismo nivel pueden realizar las mismas operaciones sobre los mismos objetos.

  Claramente, la primera forma es más flexible y permite una mejor especificación de las autorizaciones.
- Gestión de autorizaciones transferibles: un sistema de gestión de autorizaciones de acceso debe satisfacer los siguientes requisitos:
  - Conocimiento de las autorizaciones de accesos de cada usuario; de éstos unas pueden ser transferibles (a terceros) y otras no.
  - Revocación posterior de una autorización de acceso.
  - La revocación de una autorización de acceso transferible implica la revocación automática de todas las autorizaciones que son resultado de una transferencia de la misma.
  - Si un usuario puede recibir de otros más de una autorización por el mismo acceso, cada una de estas autorizaciones puede ser revocada independientemente de las demás.

3.2.1 Privacidad, problemas de seguridad e implicaciones éticas y legales

Generalmente, los errores que afecten a la integridad (p.ej., pérdida de datos) debidos a mala praxis no suelen comportar acciones legales si no hay una negligencia intencionada. Aun así, frecuentemente conllevan consecuencias laborales o económicas para el responsable (cambio de cometidos, reducción de incentivos o, en casos graves, despido). En cambio, cuando los errores afectan a aspectos de seguridad, y en especial en ámbitos sensibles (sanitario, administrativo u otros), comienzan a aparecer problemas éticos y, ocasionalmente, legales. A continuación, se muestran algunos ejemplos de implicaciones legales acarreados por problemas de seguridad:

- Acceso por usuarios no autorizados a datos personales o difusión de datos personales. Aunque la

responsabilidad última recae en quien ha conseguido realizar el acceso no autorizado, puede existir responsabilidad (incluso penal) para el responsable del tratamiento de los datos si éste no demuestra (responsabilidad proactiva) que ha tomado todas las medidas razonables para configurar correctamente la base de datos asignando los permisos adecuados para evitar tal acceso inapropiado. Es muy habitual que en una organización mediana o grande existan muchos usuarios internos (empleados) con acceso a información sensible. No sólo es importante establecer protocolos para dicho acceso sino comunicarlos y asegurarse que empleados y administrativos los asuman.

- Cesión de datos a terceros. La información personal sólo puede utilizarse para los fines para los que el usuario la ha cedido (consentimiento explícito). Eso incluye la no cesión de la información para organizaciones externas, pero tampoco para fines diferentes a los cuales la información se ha cedido, incluso en la misma organización.
- Creación de una base de datos: la mera creación de una pequeña base de datos con información personal (nombres, apellidos, etc.) puede obligar (según la legislación del país) a notificarlo oficialmente a algún organismo regulador.
- Disposición de copias de seguridad, discos duros retirados, etc. Es frecuente que se tenga bastante celo en la base de datos operativa de una organización, pero se tengan a disposición las copias de seguridad o discos duros antiguos de la base de datos. De hecho, estos pueden estar a la vista y sin contraseña, arrojarlos a la basura sin destruirlos previamente, o transportarlos (p.ej., del trabajo a casa) sin el mínimo control de seguridad (una contraseña, en caso de pérdida o robo).
- Celo en el cuidado de la contraseña de administrador. Si esto falla, todo lo demás no tiene sentido. Un número importante de problemas de seguridad son fallos humanos y no resquicios o *bugs* de los SGBD. Existen numerosos casos de administradores de bases de datos que tenían una o más contraseñas sensibles en una nota en su escritorio.

Gran parte de los puntos anteriores están regulados en España por la *Ley Orgánica de Protección de Datos Personal y Garantías de los Derechos Digitales* (LOPDGDD) de 2018 y en toda Europa por *el Reglamento General de Protección de Datos* (RGPD) de 2018. En otros países, existen leyes similares. Como cada día es más habitual que incluso pequeñas empresas ofrezcan servicios multinacionales (p.ej., a través de la web), es importante, por tanto, conocer bien la legislación de cualquier país donde se preste servicio. Por ejemplo, si una base de datos incluye clientes británicos y presta servicios a ese país, hay que conocer la legislación británica al respecto.

Estos y otros aspectos se tratan en asignaturas relacionadas con los aspectos legales y éticos de la profesión informática. Como es de esperar, gran parte de ellos están relacionados con el uso de la información que, en muchos casos, está almacenada en bases de datos.