Calidad de datos

En varias ocasiones hemos apreciado cómo distintas consideraciones de diseño conllevan distintos tiempos de proceso. Hemos visto que, con frecuencia, una acción propuesta para mejorar las consultas, por ejemplo, empeora las altas. Optamos por ella si el número de altas a realizar es mucho menor que el de las consultas, o si el momento en que se realizan las altas esta menos condicionado a una respuesta rápida. De modo que jugamos balanceando el tiempo. Si conseguimos que los procesos más frecuentes y los más importantes se hagan muy rápidos, el sistema funcionará de modo eficiente aunque los procesos menos frecuentes consuman más recursos. Sin embargo, cuando hablamos de la calidad del contenido no hay balance posible, hemos de poner **todos los medios posibles para que todos los datos almacenados sean los que deben ser el cien por cien de las veces**. En este capítulo vamos a estudiar cuatro aspectos que afectan a la fiabilidad y calidad de los datos almacenados en la base de datos: la integridad, la seguridad, la restauración tras un fallo y el control de los accesos concurrentes. En la introducción comentamos que un sistema gestor de bases de datos está estructurado en componentes que capturan la solicitud del usuario y trabajando en segundo plano (background) resuelven una cuestión concreta. La figura 1, muestra los componentes responsables de cada una de ellas.

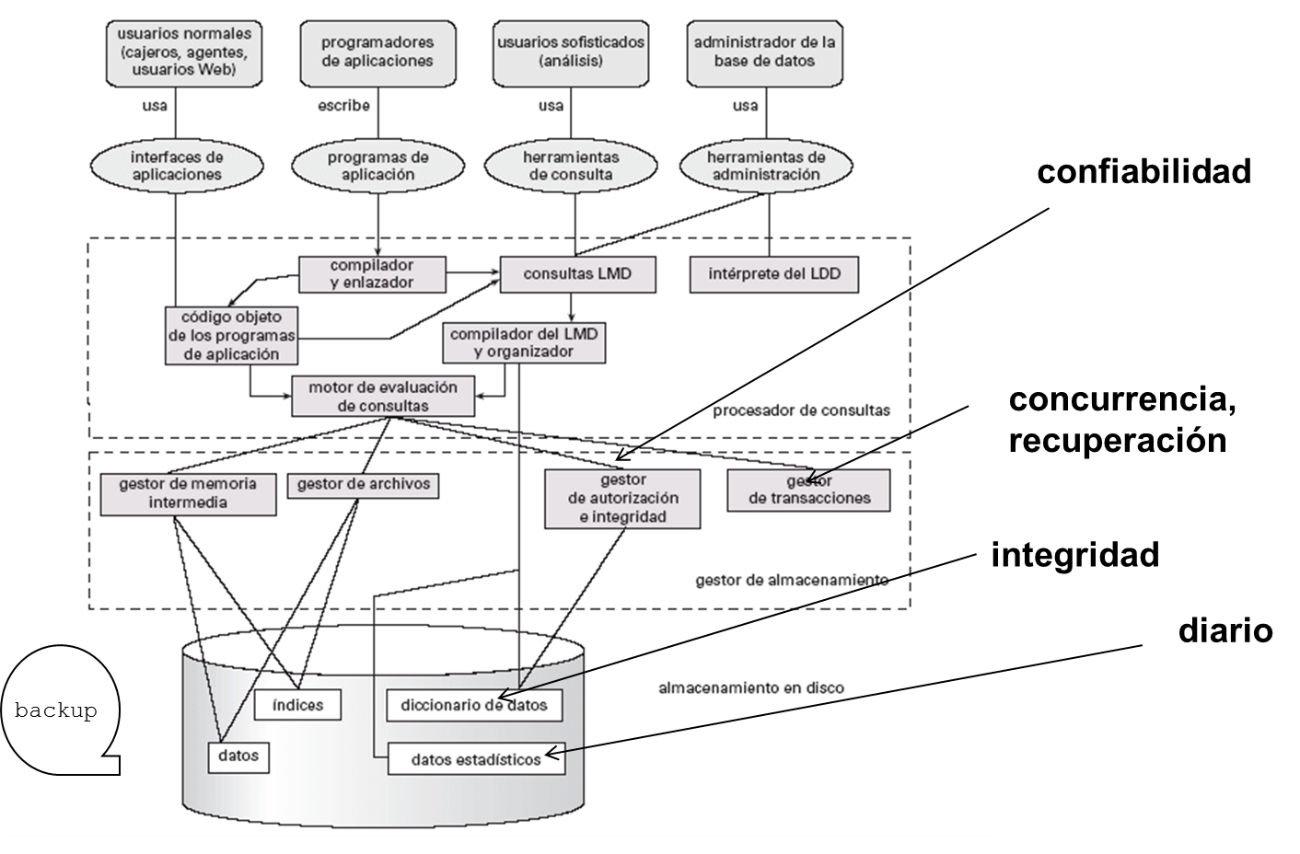


Figura 1. Estructura de un sistema gestor de bases de datos

Terminamos el capítulo con una breve presentación del contexto legal y ético exigible a la gestión de Bases de Datos.

1. Integridad

El primero de los aspectos que abordamos en la calidad de datos es la Integridad. Con este término nos referimos a la garantía de que los datos introducidos en la base de datos sean correctos, es decir que se ha escrito “lo que se debía de escribir”. Realmente es imposible garantizar que no se han producido errores al teclear los datos, pero, al menos, es posible garantizar que los datos son “lógicamente correctos”, impidiendo reflejar en la base de datos situaciones que son imposibles en el mundo real. En los sistemas de ficheros, cada aplicación debía codificar cada una de las restricciones del mundo real, a lo que llamamos restricciones semánticas (por ejemplo: fecha\_fin > fecha\_inicio, un empleado de una empresa no puede tener menos de 16 años…). La codificación de restricciones semánticas puede suponer un 90% del trabajo de desarrollo de la aplicación (ver Piattini2006, pag 121). Pero además, la semántica queda de esta manera dispersa a través de las aplicaciones y redundante. En capítulos anteriores han ido surgiendo las herramientas de que dispone el SGBD para llevar a buen puerto este objetivo, pero las recogemos aquí, como resumen, proporcionando una visión general, debido a que la integridad de datos es parte muy importante de la calidad de datos.

La **definición de dominio** de las estructuras de datos indica qué valores son lógicamente posibles para cada campo concreto (considerado de forma aislada del resto de campos de la base de datos).

Al estudiar la normalización vimos **la regla de integridad de unicidad,** por la que el valor de las claves candidatas de todo registro tiene que ser distinto de los del resto de registros. La **regla de integridad de existencia**, por la que la clave primaria de todo registro tiene que ser conocida (distinta de nulo). Y la **regla de integridad referencial** por la que toda clave ajena tiene que contener uno de los valores existentes en ese momento en el campo al que hace referencia, o ser nula.

Una buena normalización de los datos garantiza también el cumplimiento de las **dependencias funcionales**, por ejemplo, que asociado a un DNI, hay un único nombre y apellidos.

Adicionalmente hemos de atender a **las restricciones entre campos**. Con frecuencia algunos valores de un dominio no son posibles cuando se dan ciertas circunstancias, determinadas por el contenido de otros campos, por ejemplo, si sexo:=Varón, Num\_embarazos:=0.

Almacenar una única vez en el diccionario de datos la semántica, enriquece la comprensión del sistema, ahorra esfuerzo de desarrollo y mantenimiento, pero lo que es más importante, garantizan la coherencia del sistema.

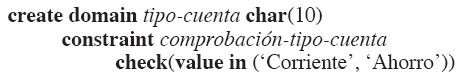
En general, una restricción tiene cinco componentes:

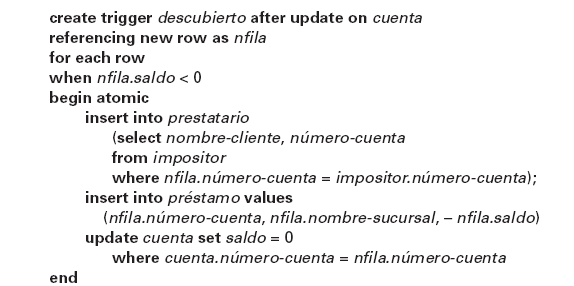
* Operación que da lugar a la comprobación (inserción, borrado, actualización)
* Condición que debe cumplirse
* Acción que debe realizarse en función del resultado de la comprobación
* Momento en que ha de realizarse la comprobación (antes o después de la operación de actualización
* Nombre que identifica la restricción.

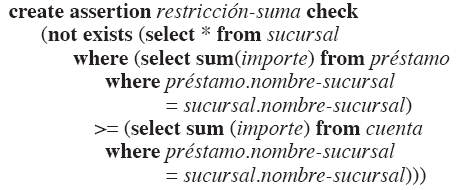
Alguno de los cuales puede no ser necesario (opcional) en algunos casos.

Las restricciones que no soporte el modelo, o el sistema gestor, tendrán que ser implementadas en el código.

A continuación un ejemplo de código de declaración de diferentes tipos de restricciones en SQL:







2. Restauración

Una organización críticamente dependiente de sus datos necesita garantizar la disponibilidad y fiabilidad de los mismos, manteniendo un buen subsistema de restauración que subsane los efectos de cualquier eventualidad. Este subsistema está basado en todos los casos en la duplicidad de la información (redundancia), lo cual conlleva una sobrecarga en el trabajo normal del sistema. Se debe intentar que esta sobrecarga consuma el menor tiempo posible y que los procesos se produzcan de forma automática, con la menor intervención posible de los usuarios, incluidos el operador del sistema, los programadores y los usuarios finales.

2.1 Posibles fallos de un sistema informático

Ningún sistema, dispositivo o programa trabaja siempre como esperamos, pueden existir errores:

● de hardware

‑ en el dispositivo de almacenamiento externo

‑ en la CPU

‑ en el canal de transmisión

● de software

‑ en el sistema operativo

‑ en el sistema gestor de la base de datos

- en nuestro propio programa de aplicación

● del operador

-que monte una cinta equivocada

-que lance el proceso inadecuado...

● cortes de corriente que conllevan pérdida del contenido de la memoria principal, y por tanto del contenido de los buffers de entrada/salida

● catástrofes del tipo de incendio o inundación

● o, incluso, daños provocados intencionadamente, como robo, falsificación de los datos, o difusión de un virus

Ryan (1995) calcula que un sistema con 256 nodos, probablemente, sufre un fallo de hardware cada 6 días. El hardware es mucho más fiable que el software, por tanto, éste último generará más errores. Los fallos imputables al hombre pueden ser aun más frecuentes (por lo menos, mucho más imprevisibles). En resumen, ¡las posibles eventualidades no son tan eventuales como parecía en principio ☺!, y hemos de estar preparados para no perder la información.

2.2 Necesidad de previsión y fundamentos de la restauración

Se puede contratar un mantenimiento y un seguro para el hardware de modo que si se estropea, lo arreglen o traigan uno nuevo. Si el hardware es convencional, está disponible en las tiendas de forma inmediata. En instalaciones críticas con dispositivos especiales, los sistemas informáticos se compran con garantía de reposición, esto es, la casa suministradora se compromete a sustituir los componentes averiados en un determinado plazo.

Es fácil mantener una copia del software estándar guardada en un sitio distinto del de la original. De todos modos la empresa suministradora enviará casi gratuitamente una nueva copia de los programas si ha ocurrido algún problema con el disco en que estaban almacenados. En el peor de los casos, el software estándar se puede volver a comprar. Un imprevisto en el software se puede resolver con un gasto adicional.

Los **datos y los programas** de aplicación específicamente **diseñados para una empresa**, **son** las únicas partes del sistema **absolutamente insustituibles**. No se pueden reponer "ni con todo el oro del mundo". Los programas, incluso los diseñados para la empresa, cambian con menos frecuencia que los datos y, por tanto, es más fácil disponer de copias con "la última versión". Los datos cambian continuamente. Podemos planificar copias de seguridad periódicas, digamos todos los viernes, pero ¿qué pasa con los cambios producidos desde entonces, si el sistema falla el miércoles? Debemos trabajar, cuando no hay problemas, de modo que los datos se puedan reponer si surge alguno.

Todas las eventualidades mencionadas en el apartado 2.1 pueden implicar la pérdida de los datos o error en los mismos. El objetivo del subsistema de recuperación es estar preparados para poder restaurar la base de datos a su estado correcto después de que se haya producido un fallo, bien **reponiendo** (rehaciendo) la información perdida, bien **deshaciendo** las modificaciones inadecuadamente efectuadas.

La única manera de poder asegurar que se está en condiciones de rehacer cualquier porción de información del sistema es que ésta esté repetida en otro punto distinto. De modo que: "la clave de la restauración es la **redundancia**".

2.3 Sistemas tolerantes al fallo

Una opción para garantizar la disponibilidad de los datos consiste en efectuar los procesos por duplicado, es decir, realizar las mismas actualizaciones en dos copias distintas de la base de datos. Esta opción se aplica en algunos sistemas (ej: Encompass de Tandem), con mejores rendimientos de lo que cabría esperar (ya que los procesos se efectúan completamente dos veces), pero también requiere el doble de almacenamiento del mismo tipo. Las dos versiones deben residir sobre distinto dispositivo, utilizar canales distintos, incluso distintas CPUs, para evitar que errores en un componente deterioren ambas copias. En la figura 2 se representa esta situación.

Las copias se deben actualizar con un período de desfase para evitar problemas que afectarían a ambas, como por ejemplo un corte de corriente. Otra posibilidad es disponer de una batería que garantice el suministro de corriente durante el tiempo suficiente para terminar de ejecutar las transacciones que estén activas. Este dispositivo se llama sistema de alimentación ininterumpida (SAI). También es posible plantear que los buffers residan en porciones de memoria principal que sea no volátil.

Con dos máquinas en paralelo, la detección de una discrepancia indica inmediatamente que se ha producido un error, pero no permite determinar cuál de los sistemas lo provocó. En aplicaciones inusualmente críticas, por ejemplo, en lanzamientos espaciales, se emplea una redundancia modular triple, en la que se dispone de 4 ordenadores efectuando los procesos independientemente y 3 de ellos confrontando los resultados continuamente. En cuanto se detecta una discrepancia, el ordenador con resultados distintos se sustituye por el de reserva. Este tipo de sistemas se conoce como **sistemas tolerantes al fallo**. Tal como lo hemos planteado, en estos sistemas aún es necesario resolver el problema de deshacer modificaciones inadecuadas, como por ejemplo que el usuario haya dado una orden que provoque la perdida de algún dato. En los apartados siguientes se ofrecen algunas soluciones a este problema.

CPU2

CPU1

Puesto del usuario

BD2

BD1

Retardador

Figura 2. Sistemas tolerantes al fallo

2.4 Restauración a partir la copia de seguridad y el diario

Otra opción para garantizar la disponibilidad de los datos ante cualquier imprevisto consiste en planificar una **copia de seguridad periódica, acompañada por un diario** (a veces llamado archivo cronológico de transacciones o bitácora de transacciones, journal o log que son los términos ingleses). En el diario se apunta el contenido del registro antes y después de su actualización, de modo que, el proceso de deshacer un cambio en la base de datos consiste en sustituir el contenido actual por el "viejo", y el proceso de reponer consiste en sustituir el contenido actual de la base de datos por el "nuevo", recogido en el diario.

La gestión de la información necesaria para poder restaurar requiere un trabajo extra en el funcionamiento normal del sistema. Como en otros muchos aspectos de esta materia, se ha de encontrar un balance adecuado entre el tiempo requerido para la reposición después de un fallo y el tiempo consumido durante el funcionamiento normal del sistema para almacenar información que permita restaurar, si surge la necesidad. La frecuencia de fallos en nuestro entorno concreto, la importancia de las consecuencias de las posibles pérdidas y los requisitos de disponibilidad del entorno, determinan el coste aceptable para el sistema de restauración.

En el resto del apartado vamos a plantear la restauración a partir de la copia de seguridad y el diario.

2.5 Fallo de una transacción

La manipulación de datos, hoy en día, se suele resolver en un entorno transaccional, y es en él vamos a presentar la problemática de la restauración.

Una transacción es una "conversación" entre el usuario y el ordenador, el primero envía un mensaje, que arranca un proceso en el ordenador, y como consecuencia del mismo, éste "contesta" con otro mensaje al usuario. En la figura 3 se pone un ejemplo procedente del entorno bancario.

¿Saldo de 349?

------------------------

USUARIO ORDENADOR

------------------------

54.000 euros

Figura 3. Transacción = conversación entre usuario y ordenador

La transacción es una unidad de trabajo, debe estar hecha completamente o no hecha en absoluto, es decir, o todo o nada. Si no puede terminarse el proceso de la transacción es necesario deshacer lo que ya se había procesado. En caso de que la transacción esté hecha, debe estarlo una sola vez.

Veamos otro ejemplo: un empleado de banco va a efectuar una transferencia de una cuenta a otra. Enviará un mensaje al ordenador similar a “transferir de la cuenta 304 a la 528 euros 5000”. En la figura 4 se presenta una posible ordenación de los pasos a seguir para realizar la transacción.

TRANSFERIR DE 304 A 528 EUROS 5000

1) comienza la transacción

2) acepta el mensaje

3) ¿existe la cuenta 304?

si no enviar mensaje y acabar

4) ¿saldo suficiente?

si no enviar mensaje y acabar

5)reescribir cuenta 304 con saldo\_nuevo = saldo\_anterior-5000

6) enviar mensaje "descontados 5000 euros de cuenta 304"

7) localizar la cuenta 528

\*

8) reescribir cuenta 528 con saldo\_nuevo=saldo\_anterior+5000

9) enviar mensaje "anotados en cuenta 528"

10) acabar transacción

Figura 4. Ejemplo de algoritmo de una transacción

Este ejemplo se podría haber desarrollado en otro orden, comprobando primeramente que se cumplían los requisitos y solo entonces efectuando las modificaciones. Ha sido planteado así precisamente, para plasmar la problemática con un ejemplo familiar y fácilmente comprensible.

El problema está justamente en 7 \* ¿qué ocurre si no se encuentra la cuenta 528? Habrá que **deshacer** la anotación de la cuenta 304 (paso 5) y enviar un mensaje "no encontrada la cuenta 528". En esta situación, el programa está aún bajo control, de modo que podríamos deshacer los cambios realizados, siguiendo el proceso inverso, es decir, sumar 5000 euros al saldo actual de la cuenta 304. Alternativamente, si en el diario se había anotado el registro de la cuenta 304 antes de efectuar la modificación, se puede utilizar la copia del diario para restaurar todas las anotaciones efectuadas por la presente transacción que no ha podido terminar, garantizando que los datos queden como si nunca se hubiera arrancado la transacción. Es más cómodo contar con esta posibilidad que tener que escribir el programa que deshaga lo hecho en cada momento.

Hay aún otro problema; el mensaje escrito en la línea 6 puede producir confusión al usuario. El usuario leería “"descontados 5000 euros de cuenta 304" y "no encontrada la cuenta 528" que son contradictorios. No debería aparecer ningún mensaje hasta que la transacción se haya realizado. Una solución es generar una cola de mensajes de modo que sólo se presenten al usuario cuando la transacción completa se ha confirmado. Si hay que deshacer la transacción se borra directamente la cola de mensajes.

En otras circunstancias, por ejemplo, en un desbordamiento en un índice, o de una operación aritmética, la transacción abortaría produciendo un error del sistema. En este caso, el programa ya no está bajo control, no es posible que el propio programa deshaga lo que ya había hecho. El subsistema de restauración automáticamente:

a) deshará todos los apuntes en la base de datos recorriendo el diario hacia atrás, escribiendo otra vez en la base de datos la copia del registro tal como estaba antes de iniciar la transacción, que está guardada en el diario

b) retirará todos los mensajes de la cola de salida.

Una transacción comienza siempre con el mensaje de arranque dado por el usuario y termina, o bien **consolidando** las modificaciones efectuadas (COMMIT), o bien eliminándolas, **cancelando** la transacción (ROLLBACK). Una transacción, por tanto, es **una unidad de recuperación además de ser una unidad de trabajo**.

Si una transacción aborta por una causa externa a sí misma, por ejemplo, se produce un corte de corriente mientras se está ejecutando la transacción, se pierde el contenido de la memoria y no se puede "continuar" el proceso. Hay que **deshacerla y rearrancarla**. Si en el diario se apuntó el mensaje de entrada (el comando de arranque de la transacción), el subsistema de restauración puede rearrancarla desde el principio, automáticamente, sin intervención del usuario. Al final del apartado veremos cómo se realiza este proceso.

2.6 Estructura del diario

Las transacciones son relativamente pequeñas y suelen modificar un número reducido de registros, pero pueden modificar varios. En la transferencia entre dos cuentas, que hemos presentado antes, se modifican dos registros. Por cada transacción realizada desde la última copia de seguridad, en el diario habrá:

mensaje de arranque de la transacción

por cada registro modificado

|registro viejo||registro nuevo|

consolidación o cancelación

Donde registro viejo, o imagen anterior contiene una copia de lo que tenía el registro antes de que se efectuara la transacción y registro nuevo, o imagen posterior, contiene una copia de cómo ha quedado el registro después de la transacción.

En un entorno multitarea se pueden estar procesando muchas transacciones simultáneamente. En el diario estarán entremezcladas las anotaciones de todas ellas. Los apuntes correspondientes a una misma transacción podrían estar encadenados, de modo que, los procesos de reponer y deshacer pudieran recorrer selectivamente el diario, y por tanto, proporcionar una restauración rápida. En la figura 5 se representa una porción de un diario con apuntes de dos transacciones numeradas como 01 y 02. La primera línea corresponde al mensaje que arrancó la transacción 01. La segunda contiene la copia de un registro antes y después de ser modificado por la transacción. Las líneas 3 y 4 corresponden a la transacción 02. La línea 5 contiene la copia anterior y posterior de otro registro modificado por la transacción 01. Finalmente la línea 6 indica que la transacción 01 ha abortado. En ese momento se recorre el diario hacia atrás siguiendo los enlaces indicados en el campo “Puntero”, visitando consecutivamente los registros 5,2 y 1.

NºReg │NºTrans Tipo Parametros Puntero

1 │01 A mensaje

2 │01 D rn/rv 1

3 │02 A mensaje

4 │02 D rn/rv 3

5 │01 D rn/rv 2

6 │01 R 5

├─────────────────────────────────────

Tipo:A=Arranque, D=Datos, C=COMMIT, R=ROLLBACK

Figura 5. Apuntes de una transacción para permitir acceso directo

Se aprecia que el **volumen** del diario puede ser **grande y el soporte** en que se almacena debe ser **rápido**, ya que incluye varios apuntes por cada transacción. Estos procesos deben poder ser arrancados por la propia aplicación, sin intervención del usuario, de modo que el diario debe estar **en línea** y se recorre con punteros, luego debe estar sobreun dispositivo de acceso **directo**. Además, el diario debe estar sobre un **dispositivo distinto** a los que soportan la base de datos.Ya que su funcionamiento normal es ir creciendo continuamente en forma secuencial, es conveniente reservar un soporte para él sólo, evitando movimientos de las cabezas lectoras (ya que su escritura es secuencial). Estas características hacen que su gestión sea **cara** y estudiaremos la **forma de reducir su tamaño**, pero antes vamos a plantear la restauración tras actualizaciones incorrectas.

2.7 Restauración tras actualizaciones indebidas

Una problemática muy particular es la de aquellas transacciones que han terminado normalmente, pero que han introducido datos erróneos en la base de datos. Por ejemplo, se debió dar la orden de “transferir de 304 a 528 euros 5000”, pero por error, se tecleó 520 en vez de 528. Como consecuencia de ello, la cuenta 528 quedó en números rojos y se devolvieron los recibos del agua y de la luz.

En esta situación no sólo es necesario restaurar estos datos concretos de la transacción, sino, también, los errores en cascada que se hayan podido propagar en sucesivas operaciones que utilizaban el dato inicialmente erróneo. Sería posible, al menos teóricamente, recorrer el diario hacia atrás, hasta esta transacción, deshaciendo todas las transaciones posteriores, corregir el dato inicial y relanzar todas las transacciones. Es una tarea con un consumo exponencial. Con el fin de minimizar las transacciones a repetir, una vez terminado el proceso de una, se pueden comparar los resultados anteriores y actuales, determinando si verdaderamente el error se propagó por ese camino, y, por tanto, es necesario revisar los movimientos posteriores sobre este registro, o por el contrario, el proceso recientemente examinado no contaminó la base de datos en esta dirección y no hace falta continuar por esta línea.

Aunque teóricamente esto es posible, el número de operaciones a realizar es tan alto que resulta impracticable. No se ha determinado un algoritmo eficiente para abordar la restauración de los errores propagados. Está claro que cuanto antes se detecte el error, menor riesgo de propagación existe. Por regla general, los sistemas disponibles actualmente no contemplan esta necesidad y el proceso se realiza manualmente.

2.8 Simplificación del Diario

El volumen del diario, tal como lo hemos planteado, puede ser elevado, y el coste de mantenerlo continuamente accesible puede resultar prohibitivo, de modo que buscamos alguna alternativa. Realmente, si no soportamos la restauración de errores propagados, las únicas transacciones que necesitan estar en línea son las que están activas ya que son las únicas que pueden sufrir una cancelación. Las transacciones que han sido consolidadas pueden ser necesarias para reconstruir la base de datos junto con la copia de seguridad, pero, para este proceso pueden estar almacenadas en un dispositivo secuencial.

Como las transacciones consolidadas nunca van a ser deshechas, no será necesario conservar la versión vieja del registro ni los mensajes que las iniciaron, de modo que esta información se puede eliminar del diario, conservando sólo la copia del registro nuevo. Las transacciones que han sido deshechas no han producido cambios, luego tampoco es necesario conservarlas. Las que finalmente no modificaron datos tampoco hemos de conservarlas, aunque terminaran normalmente.

Podemos disponer de un fichero sobre dispositivo de acceso directo y en línea que llamaremos diario "activo" o diario de las transacciones activas, y un fichero secuencial (bitácora archivo) con la información necesaria para reponer. La transacción va escribiendo sus apuntes en el diario activo. Cuando una transacción que ha modificado datos se ha consolidado, se pasa la copia del registro nuevo al diario archivo. Cuando una transacción ha terminado, independientemente de cómo haya acabado, se libera el espacio que ocupaba en el diario activo. La figura 6 presenta esta gestión.

Transacciones consolidadas

que modificaron datos

Activo

rn1

mensaje1

rv1/rn1

mensaje2

comit1

rv2/rn2

rollback

Archivo

Figura 6. Minimización del volumen del diario: paso de diario activo a archivo

En vez de liberar el espacio cuando acaba una transacción, podemos plantear disponer de dos áreas de diario activo, como se indica en la figura 7, de modo que cuando el activo 1 tenga un determinado grado de ocupación (p.e. 80%), el primer mensaje de una nueva transacción se escribirá en el activo 2. Estando el sistema un tiempo escribiendo en ambos activos, los apuntes de transacciones nuevas en el área 2 y terminando las antiguas en el 1, hasta que:

a) todas las transacciones del 1 están acabadas (hayan o no efectuado cambios en los ficheros de la base), en cuyo caso se copian al diario archivo los datos necesarios para rehacer y se inicia el área 1.

b) se haya acabado el espacio del activo 1, de modo que algún apunte de las transacciones activas que se estaban escribiendo en él no quepa. Se abortan las transacciones sin terminar y, por tanto, se deshacen y relanzan automáticamente, comenzando sus apuntes, esta vez, en el activo 2. De esta manera, todas las transacciones en el activo1 están terminadas (caso a), por tanto, se copian al diario archivo los datos necesarios para rehacer y se inicia el área 1.

Activo1

Activo2

puntero

puntero

Bitácora

Figura 7. Minimización del volumen del diario: Dos áreas de activo

Otra alternativa frecuente es mantener dos diarios distintos, uno para las transacciones activas que contiene los mensajes y las imágenes previas y en el otro las imágenes posteriores, de modo que en cuanto una transacción termina se pueden retirar sus apuntes del diario anterior y el diario posterior no requiere ninguna gestión adicional. Si la transacción aborta habrá que eliminar los apuntes del diario posterior. En este caso ambos soportes deben ser de acceso directo. La situación se refleja en la figura 8.

Diario Previo Diario Posterior

ID- PARAMETROS ID-TRANS PARAMETROS

01 Mensaje arranque 01 rn

... ....

01 rv

Figura 8. Diario previo y diario posterior

Otra estructura alternativa del diario es que no se modifique el dato en la base de datos, sino que se añada la nueva copia y se encadenen las sucesivas versiones de un determinado dato, apuntando en el diario sólo la versión nueva, para poder rehacer en caso de pérdida. El proceso de deshacer sería simplemente retrasar el puntero de versión actual (Ver figura 9).

┌──────────┬──────────┬──────────┬──────────-┐

│R1 f p│R2 f p│R1 f p│R1 f p│

└──────┴─┴─┴──────┴─┴─┴──────┴─┴-┴──────┴─┴──┘

Ri:identificador registro lógico, f:fecha, p:puntero a ocurrencia anterior

Figura 9. Base de datos histórica

Esta opción presenta la ventaja de proporcionar directamente una base de datos histórica. Si junto a cada versión de un registro se almacena la fecha de su inclusión, se puede preguntar a la base de datos, por ejemplo, cuál era el sueldo de un empleado en noviembre de 2000. Lógicamente esta opción incrementa el consumo de almacenamiento de la base de datos y se debe estudiar para cada aplicación si es necesario mantener las distintas versiones (POSTGRES basa en esta alternativa su gestión del diario. Stonebraker91).

2.9 Tamaño de las imágenes

Una unidad frecuente para el tamaño de las imágenes anterior y posterior es el bloque o registro físico. En este caso los apuntes del diario archivo podrían ordenarse por direcciones físicas, de modo que el proceso de restauración consistiría en la escritura secuencial de los bloques y resultaría muy rápido. Se puede minimizar el volumen del diario manteniendo sólo el último apunte correspondiente a registros que han sufrido sucesivas modificaciones. Ambas mejoras en el proceso de restauración conllevan un procesamiento adicional, es decir, más trabajo en el mantenimiento normal de la base de datos y habrá que estudiar su conveniencia o no en cada organización concreta.

Con el fin de minimizar el volumen del diario, a veces, se almacenan la imagen del registro lógico, de modo que el proceso de restauración supondrá leer el bloque de la base de datos, reconstruirlo sustituyendo el contenido del registro y volverlo a escribir. En este caso no se pueden eliminar las modificaciones intermedias, sobre todo cuando trabajamos con registros de longitud variable, en que una modificación puede suponer un cambio en la posición física del registro.

Como tantas veces en nuestra materia, nada es gratis y nada es "lo" ideal, todo tiene su precio, sus ventajas e inconvenientes. Nos enfrentamos al dilema entre eficiencia en el proceso de restauración o en el trabajo normal. Las técnicas de recuperación rápida imponen mayor sobrecarga al sistema durante el procesamiento de las transacciones.

2.10 Mantenimiento del diario

Como se ha comentado, puede producirse un fallo del sistema que implique pérdida del contenido de la memoria principal y, por tanto, de los buffers de entrada/salida. Las modificaciones de algunas transacciones (las que no habían acabado) tendrán que ser deshechas y éstas relanzadas, por otra parte, cabe la posibilidad de que algunas modificaciones de transacciones acabadas estuvieran aún en la memoria esperando a ser transmitidas a la base de datos, de modo que habrá que reponer la copia de la versión nueva en la base de datos. En esta situación tenemos tres problemas fundamentales:

a) no sabemos en qué situación se encuentran los datos de nuestra base de datos, qué modificaciones se habían escrito físicamente en el dispositivo y cuáles estaban esperando en el buffer para ser registradas. Luego tendremos que estar preparados para rehacer algo que ya está hecho, o deshacer algo que nunca se hizo. Por ello las operaciones deben ser idempotentes.

b) toda la información físicamente almacenada por una transacción en la base de datos, tiene que poder ser deshecha mientras no haya terminado la transacción, y toda modificación de transacciones terminadas tiene que poder ser rehecha. Esto condiciona el orden de escritura entre el diario y la base de datos.

c) no sabemos qué transacciones habían realmente terminado y cuáles estaban activas. Para resolver este problema veremos posteriormente los puntos de chequeo.

2.10.1 Operaciones idempotentes

Se dice que una operación es idempotente cuando el resultado de aplicarla una vez es el mismo que el de aplicarla varias veces. Como realmente no sabemos si el contenido de los buffers se había llevado o no a memoria, tenemos que poder deshacer lo que quizá aún no estaba hecho, e igualmente rehacer lo que quizá ya estaba almacenado en la base de datos.

Para deshacer una transacción se leen, en el diario, las copias de los registros afectados tal como estaban antes de iniciarse la transacción y se rescriben en la base de datos. Si las modificaciones efectuadas aún estaban en el buffer de la base de datos y no se habían escrito, esta operación sería innecesaria, pero, en caso de duda, se rescribe, garantizando que no se han generado errores.

Un intento de reducir el volumen del diario podría habernos llevado a no almacenar las imágenes posteriores y calcularlas cuando fuera necesario, a partir del mensaje de entrada. Esta forma de rehacer no es válida, pues no es idempotente. No es lo mismo descontar una vez de una cuenta los 500 euros sacados de un cajero, que descontarlos dos veces (sobre todo si el cliente sólo lo ha obtenido una ☺). Podemos tener que rehacer cambios que ya estaban hechos. Disponiendo en el diario de la copia del registro tal como quedó después de la modificación, la operación de rehacer consiste en copiarla a la base de datos. El resultado de copiar a la base de datos la imagen del registro viejo (o el registro nuevo) una vez, es el mismo que el de copiarlo varias veces, no es necesario atender a si estaba ya hecho.

2.10.2 Orden de escritura

Siempre que hay que realizar varios procesos se plantea la duda de cuál se ha de realizar primero. Todos los apuntes del diario y de la base de datos se están efectuando en sus respectivos buffers, pero hemos de asegurar que toda modificación reflejada físicamente en la base de datos tiene que poderse deshacer si cae el sistema antes de consolidar la transacción. Esto implica que el sistema de recuperación debe asegurarse de:

● no efectuar ninguna modificación en la base hasta que el apunte del diario correspondiente al registro viejo esté, de hecho, físicamente almacenado en el diario, para lo cual debe forzarse el volcado del buffer del diario antes de la escritura en la base.

● no terminar (consolidar) una transacción hasta que la información necesaria para rehacer (imagen posterior) esté físicamente escrita en el diario.

Este proceso aparece frecuentemente reflejado en la bibliografía anglosajona como “Write-ahead logging”, escritura del diario primero o proceso de escritura en dos fases.

2.10.3 Paradas de chequeo

A estas alturas está claro que si ocurre una irregularidad en el sistema hay que deshacer las transacciones que estaban activas. El problema ahora es: ¿cuáles estaban activas cuando se produjo el problema (corte de luz, por ejemplo)? La 1ª opción para responder esta pregunta sería recorrer el diario desde el principio e ir apuntando las transacciones para las que encontremos el comienzo y suprimirlas cuando encontremos su final. Si el diario es muy grande este proceso puede consumir mucho tiempo, de modo que puede ser inaceptable para una organización.

Otra posibilidad es efectuar periódicamente **asentamientos** del estado del sistema. También se llaman paradas de chequeo. Un asentamiento supondría:

● forzar la escritura de los buffers en el diario

● forzar la escritura de los buffers en la base de datos

● escribir en el diario un registro de asentamiento conteniendo, para cada transacción activa, la posición de su último registro en el diario

● escribir la dirección del registro de asentamiento en el fichero de arranque.

El rearranque del sistema se efectúa desde el fichero de arranque. Se localiza la posición del último asentamiento que proporciona información de las transacciones que estaban activas en la última parada de asentamiento, y se recorre el diario hacia adelante construyendo dos listados, uno de transacciones a deshacer y otro transacciones a rehacer. En principio todas las transacciones activas en el momento del asentamiento se apuntan en la lista de deshacer y se recorre el diario hacia adelante buscando los tres tipos de apuntes que van a hacer modificar estas listas:

● un apunte de inicio de una nueva transacción hará que ésta sea incluida en la lista de deshacer

● la cancelación de una transacción permitirá suprimirla de la lista de deshacer

● un apunte confirmando la terminación de la transacción la pasará de la lista de deshacer a la de rehacer.

Una vez examinado el último registro del diario las listas están terminadas. Se rehacen las transacciones terminadas desde el punto de asentamiento hasta su terminación y se recorre el diario hacía atrás, deshaciendo las transacciones de la lista de deshacer hasta encontrar su punto de comienzo, relanzándolas a partir de su mensaje de entrada.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_tc\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_tf\_\_

Línea de tiempos

T1

T2

T3

T4

T5

Figura 10. Estado de las transacciones en el momento del fallo

La figura 10 representa una línea de tiempo, indicando tc el punto de asentamiento o chequeo, y tf el momento del fallo. Se representan en ella transacciones en las distintas situaciones posibles en los momentos tc y tf.

La transacción T1 había acabado ya cuando se efectuó el chequeo, de modo que, tenemos la seguridad de que sus anotaciones están efectuadas físicamente en los ficheros de la base de datos.

En el momento del chequeo estaban activas T2 y T3 por lo que están apuntadas en el registro de asentamiento, y con ellas se inicia la lista de deshacer. Efectuando un recorrido del diario hacia delante, encontramos el comienzo de T4 que es incluida en esta lista. Encontramos después el fin de T2 que, por tanto, se pasa de la lista de deshacer a la de rehacer, y a continuación T5 es incluida en deshacer, nuevamente acaba una transacción, se borra T4 de la lista de deshacer y se apunta en la de rehacer, quedando las listas como se recoge en la figura 11

Deshacer Rehacer

T3 T2

T5 T4

Figura 11. Listas de deshacer y rehacer

Una vez deshechas T3 y T5 se relanzan automáticamente. T2 hay que rehacerla solo a partir de tc, sin embargo, T3 hay que deshacerla hasta alcanzar su principio, más allá del punto de asentamiento.

Con esta exposición, queda patente el ahorro de tiempo que proporcionan las paradas de chequeo en el caso de que sea necesario restaurar el sistema, ya que sólo hay que recorrer la última parte del diario. Sin embargo el esfuerzo adicional que supone su realización es relativamente pequeño. El fichero de arranque es muy pequeño y frecuentemente se mantienen varias copias en el sistema, que se actualizan alternativamente, de modo que si se pierde la última, sigue estando disponible la anterior.

2.11 Restauración cuando los datos son inaccesibles

Si el dispositivo que soporta a la base de datos está físicamente dañado, parte de la información (o toda) está perdida, habrá que sustituir el disco por otro. Las transacciones en curso que intenten escribir en la zona dañada abortarán y el operador recibirá un mensaje de que el disco debe ser sustituido. En esta situación la recuperación deberá efectuarse restaurando la copia de seguridad y una vez terminada ésta, habrá que rehacer la base de datos desde el diario, primero desde el archivo, apuntando todas las modificaciones realizadas por las transacciones consolidadas desde la última copia de seguridad. Posteriormente, consultando el diario activo, se relanzarán aquellas transacciones que no habían acabado. En este proceso de recuperación no hay que deshacer nada (partimos de un dispositivo nuevo) de modo que sólo hay dos posibles situaciones:

● **transacción terminada‑rehacer**

● **transacción abortada‑relanzar**.

2.12 Copias de seguridad

Periódicamente se realizará una copia del estado actual de la base de datos. El proceso de restauración a partir de la copia de seguridad es más rápido que la reconstrucción a partir del diario porque es secuencial. Cada vez que se realice una copia de seguridad se puede inicializar el diario.

La creación de la copia de seguridad es sencilla, supone ir leyendo secuencialmente toda la base de datos y copiarla sobre otro soporte, que puede ser de acceso secuencial. La restauración supone leer en secuencia la copia de seguridad e ir volcándola al soporte de la base de datos. La copia de seguridad también se suele llamar copia de respaldo, o volcado de la base de datos. Los términos ingleses para estos dos procesos son “back-up” y “restore”. Aunque el proceso es sencillo, involucra muchos registros, por tanto, requiere muchas operaciones de entrada salida. Es un proceso lento que requiere además que el sistema esté en un estado estático (en el que no se estén produciendo modificaciones), de otro modo, la copia podría contener modificaciones posteriormente canceladas o inconsistencias. Volviendo al ejemplo bancario, si se estaba haciendo una trasferencia entre dos cuentas a la vez que la copia de seguridad, podría ocurrir que se copie una cuenta antes de modificarla y la otra ya modificada, el balance del banco sería incorrecto, faltaría dinero en caja. Al restaurar desde la copia de seguridad es imposible lanzar sólo la parte de la transacción que está sin hacer en la copia de seguridad. Normalmente las copias de seguridad se efectúan cuando no hay usuarios conectados. Como copiar la base de datos completa requiere su tiempo, se debe elegir cuidadosamente la frecuencia de copias y el momento de las mismas.

Para aminorar este problema, la mayoría de los sistemas proveen facilidades para efectuar copias incrementales, que contienen, solamente, los registros modificados desde la última copia de seguridad, de modo que se reduce la duración del proceso. Frecuentemente se programa la realización de varias copias de seguridad **incrementales** intercaladas entre copias de seguridad completas. El proceso de restauración de la base de datos supone cargar la última copia de seguridad completa disponible y sucesivamente, en orden, todas las incrementales. Para poder realizar copias incrementales es necesario que el sistema lleve la cuenta de los registros que va modificando, lo cual nuevamente supone un esfuerzo adicional durante el proceso normal de trabajo para reducir el tiempo dedicado a la copia de seguridad.

En una base de datos, posiblemente algunos de los registros sufren más actualizaciones que otros, y/o resultan más imprescindibles que otros, de modo que todos los datos no necesitan las mismas frecuencias para la realización de copias de seguridad. Es adecuado plantear copias de la base de datos **selectivas** de determinados tipos de datos, de modo que no todos los usuarios están inactivos todo el tiempo en que se realiza la copia de seguridad.

Como el proceso de generar (o recuperar a partir de) una copia de seguridad es secuencial y, una vez terminado el proceso, la información almacenada no tiene que estar en línea, los soportes para las copias de seguridad resultan relativamente baratos. Se suele disponer de varios soportes que se van rotando alternativamente, de modo que si se estropea la última copia de seguridad se dispone de la anterior. En la figura 12 se pone un ejemplo de programación de la copia de seguridad alternando copias incrementales y completas en ocho soportes que se van rotando.

Completa incremental incremental incremental

1

3

semana1 semana2 semana3 semana4

4

8

2

6

5

7

semana 5 semana6 semana7 semana8

1

2

............

semana9 semana10

Figura 12. Programación de copias de seguridad

2.13 Procedimiento de inicio de un sistema transaccional

Existen tres maneras de iniciar el trabajo en un sistema transaccional:

● Inicio de **emergencia** es el presentado para recuperación tras fallos que producen datos accesibles, pero inseguros. El sistema busca el fichero de arranque, se dirige al último punto de asentamiento, rehace las transacciones que han terminado desde entonces, y deshace y relanza automáticamente las transacciones activas en el momento del fallo.

● Inicio **caliente** o normal, que es el que se efectúa después de haber "apagado normalmente" el sistema. El inicio normal es tratado por el sistema como un inicio de emergencia. La diferencia está en cómo han acabado las tareas en la anterior sesión. Los pasos de cierre de sesión son los siguientes, el operador da la orden de cierre, a partir de ese instante el sistema no deja iniciar ninguna nueva transacción y espera a que terminen las que tiene activas, efectuando entonces un asentamiento y escritura de su posición en el fichero de arranque, de modo que el registro de asentamiento es el último registro del diario. Al dar la orden de arranque, se localiza el último registro de asentamiento que no tiene transacciones activas, y se comprueba que no hay más registros por delante en el diario.

● Inicio en **frío**, este término es comúnmente usado para referirse al proceso de arrancar después de algún fallo desastroso que hace imposible la inicialización con los métodos anteriores. Teóricamente, el arranque en frío sólo es necesario en la instalación del sistema o cuando se acopla un nuevo soporte. En el apartado 2.11 hemos visto como restaurar la base de datos después de un fallo del soporte que contenía la base de datos.

2.14 Fallos en el propio sistema de recuperación

Básicamente tenemos tres alternativas cuando el propio sistema de seguridad es el que cae, pensemos, por ejemplo, una rotura del dispositivo que alberga el diario:

a.- Si la organización lo puede resistir, pararemos el sistema y efectuaremos una copia de seguridad de la base de datos, iniciando el diario a partir de ese momento en un nuevo soporte.

b.- Algunos sistemas, que necesitan un alto grado de fiabilidad y disponibilidad, gestionan por duplicado la información necesaria para la restauración (diario) o de la propia base de datos. Ya que los periféricos de almacenamiento son, posiblemente, la parte del hardware más propensa a fallos, algunos sistemas mantienen dos copias de los discos de la base de datos. Es una situación similar a la comentada en los sistemas tolerantes al fallo, pero sólo está duplicado el disco. En la terminología inglesa se llama “disk mirroring” (mirror significa espejo). De este modo, si el diario cae aún disponemos de la copia y podemos inicializar un nuevo diario. Este aumento en la seguridad conlleva nuevamente una mayor sobrecarga de trabajo durante el procesamiento normal.

c.- Si esta sobrecarga de trabajo es inadmisible para la organización, y no es posible en el momento del fallo efectuar una parada del sistema para hacer una copia de seguridad, tendremos que iniciar el diario y ser optimistas pensando que no vaya a producirse ningún fallo en la base de datos, que en esta circunstancia supondría una pérdida automáticamente irrecuperable de la información.

En muchos entornos las entradas de datos al ordenador se realizan desde impresos o formularios, por ejemplo, el impreso de matrícula o las actas en la gestión del alumnado, los albaranes de pedido en un almacén, etc. En estas situaciones siempre queda el recurso de volver a teclear los datos si se produjera un problema adicional. Es conveniente mantener los documentos provisionalmente en el orden en que se han introducido en el sistema, y anotar cualquier otro proceso realizado hasta que haya posibilidad de hacer una copia de seguridad. (Adicionalmente ponerse a rezar para que no falle la base de datos. Esta opción no tiene, desde luego, unos fundamentos muy científicos, pero, a veces es necesario recurrir a ella ☺).

2. 15 Beneficios de los mecanismos de restauración en otras áreas del sistema

La gestión del diario puede tener implicaciones beneficiosas adicionales en las siguientes áreas del sistema:

a.- Gestión de recursos humanos y productividad.

Si el diario planteado con fines de recuperación incluye, además, nombre de usuario, fecha y hora en que se hizo cada operación, es una herramienta adecuada para ayudar a mejorar la gestión y el rendimiento del personal a cargo del sistema. Es posible hacer evaluaciones de productividad y se ha demostrado que, el simple hecho de que los empleados sepan que su trabajo está siendo monitorizado aumenta el rendimiento y disminuye los errores debidos a falta de atención. Es interesante en este punto que el lector se haga una reflexión sobre las implicaciones negativas que éste trazado puede tener sobre el derecho a la privacidad de los trabajadores del sistema. Llevado a extremos indeseables este trazado puede aportar cuántos descansos hace en su trabajo, si hoy tiene un día malo, etc.

b.- Administración de la base de datos

Un estudio estadístico del diario proporciona al administrador de la base de datos información fidedigna de los patrones de uso del sistema, lo que le permitirá optimizar la ubicación de los ficheros, la organización de los datos y los métodos de acceso más adecuados.

c.- Cumplimiento de exigencias legales

El Reglamento de Medidas de Seguridad para la protección de ficheros que contienen datos personales (RD994/99. BOE 25/06/99), exige las siguientes medidas de seguridad en función del nivel de riesgo de los datos almacenados en la base de datos (ver apartado 5.3 para interpretación de niveles de seguridad):

●copia de respaldo semanal excepto quietud total (nivel bajo)

●política que garantice la restauración a tiempo de fallo (nivel bajo)

●diario incluyendo: día, hora, quién accedió a qué y qué hizo (nivel bajo). Mantener 2 años (nivel alto)

●Auditoría de seguridad cada dos años. El diario será una herramienta clave para esta auditoria (nivel medio)

2.16 Colofón

El funcionamiento de las organizaciones suele ser críticamente dependiente de sus datos. Los datos de la organización son insustituibles y sólo la organización los puede aportar, por tanto, se debe garantizar su disponibilidad ante posibles eventualidades.

Observamos que una modificación de la base de datos requiere, para garantizar su recuperación, escribir en el diario el mensaje de arranque, las imágenes previa y posterior y el apunte de que se ha completado con éxito la actualización, además de escribir la imagen posterior en la base de datos. Es decir, una modificación supone cinco operaciones de entrada / salida si los buffers son transferidos al soporte cada vez que se modifican. El protocolo de escritura en dos fases fuerza, de hecho, el volcado frecuente de los buffers. La implantación de los mecanismos de restauración supone una gran penalización de los tiempos de respuesta del sistema.

Como el costo de asegurar la disponibilidad es elevado, el administrador debe estudiar cuidadosamente a qué datos aplicarlo. Un fichero de índices puede ser reconstruido por un programa especial a partir de los propios datos y no es necesario mantener un diario para su reconstrucción, igualmente, un fichero almacenando resultados intermedios de unos cálculos, puede ser más económico crearlo nuevamente que reconstruirlo a partir del diario.

☺Esta adaptación de una conocida canción de los Beetles puede ayudar a recordar la importancia de mantener una adecuada política de restauración.

\*\* Yesterday\*\*

---------------------

Yesterday,

All those backups seemed a waste of pay.

Now my database has gone away.

Oh I believe in yesterday.

Suddenly,

There's not half the files there used to be,

And there's a milestone hanging over me

The system crashed so suddenly.

I pushed something wrong

What it was I could not say.

Now all my data's gone

and I long for yesterday-ay-ay-ay.

Yesterday,

The need for back-ups seemed so far away.

I knew my data was all here to stay,

Now I believe in yesterday.

Ejercicio 1 ¿Qué información en el diario debe residir necesariamente en el dispositivo de acceso directo?. Justifíquelo brevemente.

Ejercicio 2 Una transacción que ha finalizado se podría rehacer completamente, a partir del mensaje que la arrancó, esto ocuparía menos que la lista de las imágenes nuevas. ¿Por qué no es está la información en el diario archivo?

Ejercicio 3 Explique brevemente el proceso de recuperación de un sistema cuando se ha producido un fallo de corriente, suponiendo que no se dispone de paradas de control.

Ejercicio 4 Indique cuál es el proceso de iniciar una sesión de trabajo en un sistema transaccional que dispone de paradas de control.

Ejercicio 5 Criterios que influyen en la frecuencia con que se efectúan las copias de seguridad.

Ejercicio 6 Compare la ocupación del diario si guarda copias del registro modificado, y si lo hace del bloque. Compare el tiempo que tardará en realizarse la restauración en cada caso.

3. Seguridad

En este apartado vamos a considerar otra faceta de la calidad de datos que está relacionada con el uso que de los datos hacen los usuarios. El objetivo es garantizar que los datos sean modificados, e incluso accedidos sólo por las personas que deban hacerlo y en el modo en que tengan derecho a hacerlo.

En el tratamiento actual de datos, la base de datos es única para toda la organización, aportando una gestión integrada de los datos de la misma, esto implica que hay distintos tipos de usuarios que tienen derecho a acceder a porciones de la base de datos. Vamos a presentar los mecanismos que permiten controlar a qué porciones de datos acceden los usuarios legales del sistema y qué derechos tienen sobre ellos. De otra parte, los datos de una organización son valiosos y puede que personas externas pretendan entrar al sistema, bien para beneficio personal, bien con el ánimo de dañar los datos disponibles. Cuando los sistemas están conectados a redes de ordenadores, como suele ocurrir hoy en día, resulta mucho más difícil mantener el acceso externo bajo control, por lo que los temas de seguridad están cobrando cada vez mayor relevancia. Vamos a presentar algunas herramientas para proteger el sistema. Encontramos dos situaciones distintas: el control del acceso de usuarios legales, y el control del acceso externo.

3.1 Control de accesos de usuarios del sistema

Cuando la base de datos contiene la información de una organización, las necesidades de conceder acceso a los datos pueden ser muy variadas. Para centrarnos, recogemos a continuación algunas de las situaciones en que pueden encontrarse usuarios legales del sistema respecto a un fichero concreto, el de los empleados de la compañía. Un usuario del sistema puede tener:

● acceso libre a cualquier campo y registro del fichero, para realizar cualquier operación (altas, bajas, modificaciones, consultas e incluso borrar el fichero completo)

● acceso prohibido a toda la relación para cualquier operación

● acceso a ver toda la relación, pero no cambiar ningún dato

● permitido ver solamente su propio registro, no los del resto de empleados, pero no autorizado a cambiar ningún valor

● permitido ver su propio registro y cambiar la dirección y el teléfono, pero no autorizado a cambiar el resto de sus datos (por ejemplo el sueldo ☺)

● permitido a ver el nombre, dirección, teléfono y departamento de todos los empleados, y cambiar dirección y teléfono

● autorizado a ver y cambiar el sueldo de todos los empleados, sólo de 9 a 17 horas en días laborables y sólo desde terminales de departamento de nóminas (alguien, fuera de su horario de trabajo, está “altruistamente” cambiando sueldos... sospechoso ¿no?☺)

● autorizado a ver y cambiar sueldos menores de 10.000 euros, pero no los superiores

● autorizado a ver nombre o sueldos de todos los empleados, pero no simultáneamente el nombre y el sueldo de un empleado concreto

● permito ver los registros y cambiar el campo de “anotación\_del\_jefe” sólo de empleados que son sus subordinados.

Para conseguir esta gran flexibilidad el subsistema de seguridad debe ser capaz de identificar al usuario, proporcionar mecanismos para recibir especificaciones de los derechos que cada usuario tiene sobre cada dato, almacenar esta información, que lo hará en la matriz de autorización, y un programa árbitro que intercepte cualquier solicitud de acceso a la base de datos y consultando la matriz de autorización, deniegue o conceda la solicitud. Estos son los cuatro apartados que vemos a continuación.

3.1.1 Identificación y autenticación de usuarios

El primer proceso para conceder los derechos de acceso que correspondan a cada usuario, es poder **identificar** a la persona que hay sentada detrás de un terminal de ordenador. Los sistemas operativos y también los sistemas gestores de bases de datos suelen iniciar una sesión de trabajo preguntando el nombre o identificativo de usuario, y a continuación efectúan alguna acción que **acredite** que el usuario es quien dice ser. Este proceso se llama **autenticación o autentificación**. La forma más sencilla y más extendida de esta acreditación es **solicitar una palabra de paso** o clave de acceso (password en terminología inglesa). Frecuentemente el sistema no replica en la pantalla la palabra tecleada por el usuario, a fin de facilitar su confidencialidad, en alternativa, escribe una \* por cada carácter pulsado por el usuario, para que éste sepa por donde va.

Esta forma de autenticación presenta muchos problemas, de los que hablaremos a continuación, pero es sencilla de implementar y barata de mantener. Su fiabilidad depende, en gran medida, del cuidado que cada usuario tenga sobre su propia clave. Se está trabajando mucho en la actualidad en otro tipo de acreditaciones que usan **identificadores biométricos**, es decir, que miden determinadas características físicas del usuario comparándolas con las que se tienen almacenadas en el sistema. Esto es posible, por ejemplo a partir de las huellas dactilares, el timbre de voz o características del ojo. Estos sistemas son mucho más caros que la gestión de palabras de paso y presentan el problema de que, pueden no acreditar al usuario legal, impidiéndole el acceso, al no reconocerlo por alguna circunstancia, porque está constipado, había viento y tiene los ojos irritados etc. Que un sistema de acreditación no reconozca al usuario legal es un grave problema para el buen funcionamiento del sistema. Estos sistemas se están usando ya en determinadas circunstancias que requieren alta seguridad, pero no están muy extendidos, porque para ser fiables son muy caros. Frecuentemente nos enfrentaremos a la acreditación por palabra de paso.

La primera consideración a hacer sobre la gestión de palabras de paso es que el personal debe ser consciente de que su clave identifica su trabajo en el ordenador, y que él es **responsable** de las tareas realizadas en una sesión arrancada con ella. ☺ Un post-it amarillo pegado en la pantalla del ordenador con la clave de acceso, hace completamente inútil cualquier gestión del subsistema de seguridad. Cuando los terminales están ubicados en una zona pública, si el usuario abandona su puesto de trabajo con una sesión abierta (es decir habiendo introducido su clave de acceso), el sistema puede ser operado por cualquiera, suplantándole, y disfrutando de sus derechos de acceso. La **formación** de los usuarios legales del sistema resulta imprescindible para el buen funcionamiento de un subsistema de seguridad.

La clave debe poder ser definida por el usuario para que le sea **fácil de recordar** y desconocida por cualquier otro. No es buena idea que el usuario tenga que confiar en el distribuidor de claves. El usuario debe **cambiar frecuentemente** su clave.

Una buena clave debe tener 6-8 caracteres, mezclando números y letras. Un hipotético agresor podrá ir introduciendo claves para ver si acierta. Claves menores o de sólo números reducen mucho las combinaciones posibles, facilitando la tarea del intruso. La clave no debe ser el nombre del usuario, ni el de un familiar cercano, ni su fecha de nacimiento, pues todos estos datos pueden ser conocidos por muchas personas. Tampoco son adecuados nombres mitológicos o comerciales que resultaría fácil descubrir a un agresor haciendo pruebas.

Si la clave debe ser tecleada en público es conveniente usar caracteres de todas las partes del teclado, pues el agresor puede obtener pistas por la posición de los dedos en el teclado. En este caso es conveniente cambiarla con mayor frecuencia.

Aun eligiendo una buena palabra de paso, se presentan otros problemas. Un agresor puede incluir en el sistema un programa que registre todo lo que se teclea y por tanto la clave, o bien, escribir un programa que simula la pantalla de identificación, envía la clave al agresor, simula el mensaje de palabra de paso incorrecta y enlaza con la pantalla de identificación del sistema, el usuario pensará que se ha equivocado sin sospechar que estaba siendo espiado.

Otro problema es que el sistema tiene que almacenar las palabras de paso de sus usuarios legales, manteniendo en el diccionario palabras de paso. Este almacén debe estar muy bien protegido, pues el agresor intentará atacarlo de distintas maneras.

Una opción que aumenta la seguridad de la palabra de paso es coordinarla con la presentación de un soporte físico, como ocurre en los cajeros con las tarjetas de crédito. Así el agresor debe conocer la clave y disponer de la tarjeta sin que el usuario legal se haya percatado del robo (por supuesto tampoco es buena idea almacenar la clave de acceso junto a la tarjeta).

Tampoco es buena idea mantener la misma palabra de paso para acceder a distintos sistemas, por ejemplo para la cuenta de correo electrónico, el banco y la base de datos. Estos entornos pueden requerir distinto nivel de seguridad y quizá unos sean más vulnerables que otros.

Ejercicio 7 ¿Cuántas palabras de paso distintas de 6 caracteres de longitud se pueden construir con un teclado que tiene 48 teclas diferentes? ¿Cuánto tiempo (min, max y medio) necesita un agresor para adivinar una clave, si no tiene ninguna información adicional y tarda 3 segundos en probar cada clave.

3.1.2 Reglas de autorización

Es necesario poder comunicar al sistema qué usuarios tienen derechos sobre los datos y qué derechos tienen. El lenguaje de definición de datos dispondrá de unas instrucciones que permitan conceder y retirar permisos. En SQL la instrucción para conceder permiso a realizar unas operaciones sobre un conjunto definido de datos a un conjunto de usuarios es:

GRANT operaciones ON datos TO usuarios

Y para retirar permisos previamente concedidos la instrucción es

REVOKE operaciones ON datos FROM usuarios.

Las operaciones que se pueden autorizar al usuario son:

●SELECT derecho solamente a consultar

●UPDATE derecho a consultar y modificar

●DELETE derecho a consultar y borrar registros

●INSERT derecho a consultar y crear nuevos registros

●ALL derecho a cualquiera de las operaciones anteriores

●NONE el usuario no tiene ningún derecho sobre el conjunto de datos

●RUN derecho a ejecutar los programas indicados y ha hacer las operaciones que ellos permitan, pero autoriza a acceder a los datos directamente

●RESOURCE autoriza a crear nuevos ficheros

●COPY autoriza a copiar cualquier parte o todo el conjunto de datos indicado

●GRANT autoriza a conceder derechos a otros usuarios sobre los datos indicados

●DBA derechos de administrador, la mayor concesión que se puede hacer.

La autorización o retirada de derechos puede ser sobre cualquier conjunto de datos que se pueda definir con una instrucción SELECT de SQL, es decir, cualquier subconjunto de la base de datos, tanto sean tablas base o vistas montadas sobre partes de varias tablas base.

El conjunto de usuarios en una autorización o retirada de derechos es, bien una lista de identificadores de usuarios legales (o roles), bien PUBLIC, que significa que cualquier usuario legal del sistema, actual o futuro, tiene los derechos indicados sobre el conjunto de datos definido.

Inicialmente el único usuario del sistema es el administrador y éste va concediendo los derechos a los distintos usuarios según va haciendo falta, o delegando la administración de partes de la base de datos en otros usuarios (con la autorización DBA).

3.1.3 Matriz de autorización

El subsistema de autorización necesita mantener la información sobre los derechos actualmente en vigor. Conceptualmente se suele presentar el almacén de estos datos como un arreglo de dos dimensiones, que contiene en las filas los usuarios (o roles) del sistema, en columnas los distintos conjuntos de datos y en la celda de intersección de cada fila y columna se indicarán los derechos de acceso que tiene ese usuario para ese subconjunto de datos. En la figura 13 se representa la estructura de este arreglo, que recibe el nombre de matriz de autorización.

datos

usuario conj1 conj2 conj3

us1 ALL NONE SELECT

us2 INSERT+DELETE RUN GRANT+ALL

us3 NONE DBA SELECT

Figura 13. Matriz de autorización de accesos

Aunque conceptualmente estemos visualizando el almacenamiento de los derechos como una matriz, a la que corresponde un tipo de dato arreglo, posiblemente la forma de almacenamiento en el ordenador será distinta, ya que es una matriz dispersa. Es decir, la información que contiene es muy redundante. Almacenarla como arreglo consumiría mucho más espacio del necesario. Decimos que es muy redundante porque para muchos de los conjuntos de datos habrá un conjunto reducido de usuarios que tienen la mayoría de los derechos y el resto de usuarios no tienen ninguno, como ocurre por ejemplo en el fichero de empleados, en el que los trabajadores del departamento de nóminas tienen unos derechos completamente distintos al del resto de usuarios. En el mismo sentido, en un fichero definido como público, todos los usuarios tienen todos los derechos.

Se deben cumplir las siguientes condiciones en las autorizaciones concedidas a un usuario:

● Si el usuario i tiene derechos sobre un conjunto de datos, los derechos que mantiene sobre cualquier subconjunto de este, son iguales o menos restrictivos.

Ejemplo: si acceso ALL a la relación empleados, es imposible que tenga acceso restringido a sueldos >1000.000.

● Si el usuario i tiene derecho a actualizar un conjunto de datos, también tiene derecho a verlo.

Ejemplo: si i puede actualizar el sueldo de empleados con sueldo ≤1000.000, tiene que poder ver empleados con sueldo ≤ 1000.000.

● Si i no tiene derecho a leer un conjunto de datos, no puede tener acceso a dar altas, modificaciones, ni bajas. En caso contrario, no podría ver datos que acaba de introducir.

3.1.4 Arbitro

El subsistema de seguridad dispone de un programa que se está ejecutando siempre en segundo plano, que intercepta todas las solicitudes de acceso a la base de datos, analiza la solicitud determinando que el usuario i quiere acceder a los datos j para realizar la operación o, lee en la matriz los derechos que i tiene sobre j. Si o está permitida, autoriza el proceso y la base de datos responde la solicitud del usuario. Si la operación o no está autorizada para el usuario sobre estos datos, el árbitro para el proceso, no permitiendo que se realice la operación del usuario.

Hay que tener algún cuidado con los mensajes que proporciona el árbitro porque, a veces, un acceso denegado puede proporcionar información. Si Juan tiene derecho a ver los registros de todos los empleados excepto los de aquellos cuyo su oficio es espía, y Juan pregunta a la base de datos por Pepe Pérez, y por Xyz Xyz y obtiene en el primer caso “acceso denegado” y en el segundo “no existen datos de Xyz Xyz”, es muy probable que Pepe Pérez sea un espía y que no hay ningún empleado que se llame Xyz. La respuesta del sistema debería ser siempre la misma, “(para Vd.) no existen datos con estas características”.

Para agilizar la decisión, el árbitro podría ir analizando los procesos en orden creciente de complejidad. En la figura 14 se presenta la decisión del árbitro al contrastar la solicitud del usuario i sobre los datos j, S(i,j) con los derechos que figuran en la matriz de acceso. Las situaciones posibles se han ordenado de forma que en cuanto se satisface una, el árbitro toma la decisión, sin necesidad de examinar las situaciones posteriores.

**Situación**  **Acción**

a) Todos los ficheros en S(i,j) incondicionalmente accesibles Autorizar

b) Hay un fichero en S(i,j) incondicionalmente prohibido Denegar

c) Todos los atributos en S(i,j) incondicionalmente accesibles Autorizar

d) Hay un atributo en S(i,j) incondicionalmente prohibido Denegar

e) Todas las combinaciones de atributos son incondicionalmente accesibles Autorizar

f) Hay una combinación de atributos incondicionalm. prohibida Denegar

g) El valor del dato está en el rango permitido Autorizar

h) El valor del dato está fuera del rango permitido Denegar

Figura 14. Acción del árbitro ante los derechos mantenidos por el usuario

Resaltamos el hecho de que las situaciones comprendidas entre la a y la f son independientes del valor concreto involucrado en la solicitud del usuario, luego el árbitro puede tomar la decisión de autorizar o denegar conociendo la solicitud del usuario y la información en la matriz de acceso, por tanto, puede decidir antes de que se inicie el trabajo en los ficheros. Por el contrario, la decisión en las situaciones nombradas con las letras g y h tienen que realizarse en tiempo de ejecución.

La ley de protección de datos personales fuerza la gestión de derechos de accesos para las bases de datos afectadas.

3.2 Control del acceso externo

En este apartado queremos presentar las herramientas disponibles para proteger los datos en el sistema del acceso por otras vías distintas del acceso normal. La mayoría de los ataques que reciben los sistemas, o al menos los más perjudiciales, son efectuados por personas de la propia organización, que conocen bien el sistema y sus puntos débiles. A la hora de organizar un subsistema de seguridad hemos de ser conscientes de que la seguridad del sistema en su conjunto, es menor o igual a la de su punto más inseguro. Lo vemos claramente con un ejemplo de la vida diaria: de nada sirve poner una puerta blindada con 27 candados si está abierta una ventana en la planta baja. El potencial ladrón entrará por la ventana directamente sin importarle los candados de la puerta. Por tanto, hay que ir examinando los distintos puntos donde la información puede ser accedida.

Los datos en nuestra base de datos tienen que ser protegidos no solo contra modificaciones que atenten contra la integridad de los datos, sino también de los que vulneren su confidencialidad. Es decir, con frecuencia, no sólo hay que intentar evitar la pérdida de los datos, o que un empleado se suba unilateralmente su propio sueldo, sino que es necesario impedir que personas no autorizadas lean la información almacenada, no solo porque a veces porque a la organización no le interesa que se sepa, sino porque incluso hay una obligación deontológica o legal de mantener el secreto sobre los datos almacenados. Por ejemplo, un paciente de un hospital tiene derecho a que su historia sea confidencial.

Un punto que a veces se olvida es destruir la información en papel. Los listados salidos del ordenador pueden contener información que hay que mantener confidencial. Igualmente, hay que custodiar las copias de seguridad y confirmar que se reescriben los ficheros temporales o las posiciones de memoria utilizadas en un proceso que maneja datos confidenciales, de modo que no sean recuperables por otro programa.

Otro problema es que la matriz de acceso puede estar gestionada por el sistema gestor de la base de datos, un usuario que no tenga acceso a un fichero a través de la base de datos, puede copiarlo mediante el sistema operativo a otro fichero sobre el que él tenga todos los derechos y desde su sistema podrá acceder a la información contenida. O bien sustraer el soporte, llevarlo a otra máquina en la que él puede ser el administrador y desde allí tiene acceso a todos los datos. Resulta más difícil pensar que en estas situaciones pueda modificar los datos y reponer la información sin que sea detectada la intrusión, pero como mínimo hay un problema para la confidencialidad.

Otro punto particularmente problemático es la información que viaja a través de una línea de comunicaciones. Cuando usuarios legales del sistema realizan procesos de forma remota conectados a una red de ordenadores, el agresor puede interceptar la información e incluso modificarla mientras viaja por la red. Por tanto los datos en el soporte y mientras viajan por la red de ordenadores tendrán que estar cifrados, es decir, codificados de forma que la información no sea legible.

Hemos comentado que el fichero de palabras de paso es uno de los puntos más críticos respecto a la seguridad del sistema. Por supuesto se almacenará también cifrado. Frecuentemente, se cifra de modo que no pueda ser descifrado. El usuario introduce su palabra de paso, el sistema la cifra y la almacena en el ordenador. Cuando un usuario quiere iniciar una sesión de trabajo introduce su palabra de paso, el sistema la cifra y la compara con el valor almacenado.

El agresor puede copiar el fichero de palabras de paso para hacer pruebas con él más cómodamente en otro equipo. En vez de intentar arrancar sesiones con el nombre de un usuario actual del sistema, puede ir dando de alta usuarios con distintas palabras de paso y analizar después el diario para ver si alguna de las que ha introducido coincide con alguna de las que hay en el diccionario. Si el sistema tiene muchos usuarios será relativamente fácil encontrar la clave de alguno de ellos. El agresor que pretende entrar en el sistema le da igual entrar como uno u otro usuario. Por esto insistíamos en no usar nombres de personas, o de cosas, nombres mitológicos o de marcas como palabras de paso.

El diario del sistema debe llevar control de los intentos de conexiones que han fallado por un error en la clave. Los usuarios legales pueden tener un determinado número de fallos al teclear la clave y bloquearle tras ellos. Incluso, el sistema puede evaluar el porcentaje de fallos global de los usuarios. Si en un determinado momento este porcentaje aumenta considerablemente puede significar que alguien está intentando violar el sistema, localizando la clave de un usuario legal. En estas circunstancias es adecuado que los usuarios legales modifiquen sus claves, y posiblemente reforzar otras medidas de seguridad.

4. Accesos Concurrentes

Los aspectos de la calidad de datos de que hemos hablado hasta ahora consideran los procesos aisladamente, garantizando que cada uno se realiza adecuadamente según distintos criterios (validación, disponibilidad de datos y confidencialidad), pero en los sistemas multitarea aparece otra fuente de error que es la que vamos a estudiar ahora.

En un sistema multitarea se realizan varias transacciones, simultáneamente (si hay varios procesadores), o solapadamente en el tiempo, de modo que hay varias transacciones activas repartiéndose fracciones de tiempo del procesador. Si las transacciones solamente consultan datos no se generan problemas. El problema surge cuando dos transacciones pretenden modificar simultáneamente el mismo dato. Vamos a verlo con un caso práctico. Como en apartados anteriores, nos apoyamos en un ejemplo de un entorno bancario, porque al hablar de dinero queda muy clara la gravedad del problema de que se reflejen errores en los datos ☺. Varias personas tienen tarjetas de débito contra una cuenta bancaria ¿Qué pasaría si quedan todos en sacar, a una hora predeterminada y desde distintos cajeros, todo el saldo de la cuenta? El banco tiene que estar preparado para que esto no pueda ocurrir.

Veamos otro ejemplo, supongamos que, en el momento actual, las cuentas 301, 302 y 303 tienen un saldo de 10.000 euros cada una. Trabajamos en un ordenador multitarea que reparte fracciones de tiempo entre distintas transacciones. El sistema recibe las siguientes órdenes de trasferencia:

T1: transferir 5000 euros de la cuenta 300 a la 301 y

T2: transferir 3000 euros de la cuenta 301 a la 302

Los saldos esperados al final del proceso son: 300:5000; 301:12.000 y 302:13.000 euros respectivamente.

En la figura 15 se presenta una de las posibles secuencias en que se pueden haber ido resolviendo los procesos relevantes de ambas transacciones sobre una línea de tiempos

tiempo **T1 T2**

t1 inicio

t2 lee 300

t3 actualiza 300

t4 lee 301

t5 inicio

t6 lee 301

t7 actualiza 301

t8 lee 302

t9 actualiza 301

t10 termina

t11 actualiza 302

t12 termina

Figura 15. Una posible distribución de tiempos para dos transacciones

En la figura 16 se repite la misma tabla pero indicando los valores leídos o actualizados en cada caso. Aparece en negrita la última vez que se ha escrito cada cuenta.

**T1 valor leído/escrito T2 valor leído/escrito**

t1 inicio

t2 lee 300 Saldo.300=10.000

t3 actualiza 300 **Saldo.300=5000**

t4 lee 301 Saldo.301=10.000

t5 inicio

t6 lee 301 Saldo.301=1.0000

t7 actualiza 301 Saldo.301=7000

t8 lee 302 Saldo.302=10.000

t9 actualiza 301 **Saldo.301=15.000**

t10 termina

t11 actualiza 302 **saldo.302=13.000**

t12 termina

Figura 16. Distribución de tiempos, indicando resultado de las acciones

En la figura 16 se aprecia que la actualización de la cuenta 301 que se hizo en el tiempo t7 se ha perdido. La transacción T1 leyó el saldo de la cuenta 301 antes de que lo actualizara T2 y no se enteró de que esto había sucedido. Como consecuencia el saldo de la cuenta 301 es incorrecto. En el balance de la oficina bancaria faltarían 3000 euros.

4.1 Candados

Un sistema informático no puede permitir que “algunas veces” se produzcan errores, puede permitir que algunas veces (las menos posibles) el sistema no sea eficiente y consuma muchos recursos en un proceso determinado (tiempo o memoria), pero los resultados siempre, absolutamente **siempre tienen que ser correctos**. En el balance de todos los procesos que realiza, además, tiene que ser eficiente de forma global. Prevenir el problema que acabamos de plantear va a suponer nuevamente un trabajo extra del sistema.

Una herramienta para manejar adecuadamente los accesos concurrentes es que cada transacción ponga candados (locks) a los recursos que vaya necesitando para actualizar. En los tiempos t2 y t4, cuando T1 lee las cuentas 300 y 301, respectivamente, les pone un candado reservando estos registros para sí, avisando al resto de transacciones que ella los tiene, con la intención de modificarlos. En t5, la transacción T2 solicita acceder a la cuenta 301, pero como tiene un candado, T2 entra en espera y solo se reanudara cuando T1 libere el registro.

4.2 Cierre y liberación de candados

En principio, cuanto menor sea el tiempo en que se reserva un recurso para una transacción, mayor posibilidad de compartir recursos y el sistema funcionará mejor, de modo que los candados se pondrán lo más tarde posible y se liberarán en cuanto no sean necesarios. Los candados se irán poniendo conforme la transacción va necesitando los recursos. En el ejemplo anterior, la transacción T1, en el tiempo t2, pone un candado en la cuenta 300 porque la lee con intención de modificarla. En el tiempo t3 lo actualiza y ya no va a volver a utilizarlo, luego podíamos pensar que es buen momento para liberar ese candado. Pero, como hemos visto al estudiar los mecanismos adecuados para garantizar la restauración de datos ante eventualidades, una transacción puede ser deshecha en cualquier momento mientras está activa. Si el candado se libera estando la transacción activa, y luego se cancela, podría ocurrir que otra transacción vea valores que “nunca han existido”.

Veámoslo nuevamente con un ejemplo: modificamos el punto de partida anterior, de modo que los mensajes de inicio son:

T1: transferir 5000 euros de la cuenta 300 a la 301 y

T2: transferir 3000 euros de la cuenta 301 a la 300

En la figura 17 se recoge una posible distribución de tiempos para ambas transacciones. Se ha indicado con un pequeño candado cerrado o abierto cuando una transacción reserva un registro para sí y cuando lo libera. En el tiempo t9 se produce una situación por la que se cancela la transacción (por ejemplo su diario activo está lleno, por tanto se deshace y se relanza). En el tiempo t7, la transacción T2 ha leído un valor para la cuenta 300, que “nunca ha llegado a existir” lo que también produciría errores en los saldos finales esperados. Por tanto, una transacción **no puede liberar** los recursos que tiene reservados **hasta que no está consolidada**.

**T1 valor leído/escrito T2 valor leído/escrito**

t1 inicio

t2 🔒 lee 300 Saldo.300=10.000

t3 actualiza 300🔓Saldo.300=5000

t4 inicio

t5 🔒lee 301 Saldo.301=1.0000

t6 actualiza 301🔓 Saldo.301=7000

t7 🔒lee 300 Saldo.300=5000

t8 actualiza 300🔓 **Saldo.300=8.000**

t9 DESHACER

t10 termina

Figura 17. La transacción no pueden liberar recursos hasta que está consolidada

4.3 Abrazo mortal (Deadlock)

La reserva de recursos, por parte de una transacción, genera un problema adicional. En la figura 18 presentamos otra posible línea de tiempos cuando se lanzan al sistema los siguientes mensajes de arranque de transacciones:

T1: transferir 5000 euros de la cuenta 300 a la 301 y

T2: transferir 3000 euros de la cuenta 301 a la 300

T1 T2

t1 inicio

t2 🔒 lee 300

t3 actualiza 300

t4 inicio

t5 🔒lee 301

t6 actualiza 301

t7 intenta leer 300 y **entra en espera** de fin T1

t8 intenta leer 301 y **entra en espera** de fin de T2

t9

t10

Figura 18. Abrazo mortal. Las transacciones quedarán infinitamente en espera

Apreciamos que T2 no puede proseguir hasta que T1 haya terminado. T1 no puede progresar porque está esperando que termine T2. Y así quedarían eternamente. Esta situación se llama **abrazo mortal**.

Hay varias alternativas para solucionar este problema. El sistema podría, de tiempo en tiempo, lanzar un proceso que viera qué transacciones están esperando a cuáles y cancelar (deshacer y relanzar) una de ellas. Pensemos que la situación puede ser más compleja que la que aquí hemos pintado en que T1 espera a T2 y T2 a T1. Puede que T1 espere a T2, T2 a T3,.. y finalmente Tn espere a T1, de modo que hay varias transacciones involucradas. Detectar la existencia del abrazo mortal supondría construir un grafo con las esperas de todos los procesos activos y comprobar si se cierra por algún punto. Otra posibilidad es que cuando una transacción lleva un determinado tiempo en espera, el sistema la cancela (deshace y relanza), por si estaba en un abrazo mortal. Si era así, al cancelarla se liberan sus candados y la transacción que estaba en espera continuará. Si no estaba en espera se ha desperdiciado un trabajo.

Ejercicio 8. El sistema de gestión de información Knosys permite actualizar la base de datos en red, pero las actualizaciones que se producen en cada terminal no son visibles por el resto de terminales hasta que no se ejecuta la opción “Actualizar bases de datos”. Haga un diseño posible de cómo se estará realizando este proceso.

5. LEGISLACION APLICABLE A LA GESTION DE BD

La generalización de los sistemas informáticos y las redes de comunicación para el almacenamiento, gestión y acceso a la información supone una amenaza potencial para el derecho de los individuos a controlar la imagen que los demás tengan sobre su personalidad. Muchas bases de datos contienen datos relativos a personas. Esta información puede ser accedida, si no es debidamente protegida incluso por personas que nada tienen que ver con la organización que mantiene los datos, y adecuadamente tratada puede permitir que otros violen este derecho individual.

5.1 La privacidad

Realmente la palabra privacidad no estaba recogida en el diccionario de la lengua española hasta 2002, sin embargo es el término que utiliza el legislador en 1992 y no es equivalente a intimidad. Parece adecuado detenerse un momento en especificar a qué se refiere concretamente.

El término intimidad se refiere al conjunto de las facetas más reservadas de la vida de un individuo, sus actos en solitario o entre las paredes de su casa. La privacidad cubre un ámbito más amplio que la intimidad de la persona, ya que abarca todo el conjunto de datos que, aunque considerados aisladamente carezcan de significación, debidamente enlazados pueden proporcionar un retrato del individuo.

Enfrentado al derecho a la privacidad se encuentra el derecho a estar informados de la actualidad y a veces los datos personales pueden contener un interés histórico que también deben estar públicamente disponibles. Hay que dilucidar, en cada caso, cual es el interés que prevalece.

Existen al menos cinco componentes que hacen más peligroso el tratamiento automatizado de datos personales que el manual:

1) **En cuanto al tiempo**: uno de los principios por los que se rige nuestra sociedad es el derecho al olvido, que conlleva la creencia en la capacidad de reinserción y rehabilitación del individuo. El tratamiento automatizado disminuye las posibilidades de disfrutar de este principio, ya que es más fácil almacenar y conservar la información digital y, por tanto, “recordar” los datos por más tiempo. Resulta mucho más rápido y barato localizar la información concreta de interés en un momento determinado. También lo es distribuir la información digitalizada.

2) **En cuanto al espacio**: una posibilidad para rehacer la vida tras una situación inaceptado socialmente ha sido y es cambiar de provincia o país. Pero a través de redes de ordenadores o copias digitales, la información está disponible en cuestión de segundos en cualquier parte del mundo, minorando las ventajas del cambio de domicilio. Adicionalmente, se pueden almacenar muchos más datos en menor volumen y con bajos requisitos de conservación.

3) **Vulnerabilidad**: Los datos contenidos en ordenadores, especialmente, los trasmitidos a través de redes de ordenadores, si no están debidamente protegidos, podrían ser accedidos por personas no autorizadas, que incluso nada tengan que ver con la organización que está manteniéndolos.

4) **Capacidad de síntesis**: valores globales, promedios, totales, frecuencias, etc., pueden ser más significativos que los datos aislados. El sistema automatizado puede proporcionar los datos con el grado de detalle con que se le requiera en cada circunstancia. Por ejemplo, el sistema de información de una biblioteca que almacena los documentos accedidos por sus usuarios o tomados en depósito. A partir de esta información se puede inducir, posiblemente información sobre su dedicación, formación o capacitación, sus gustos (en algunos casos, aspectos especialmente privados como ideología y tendencias sexuales), responsabilidad y seriedad (¿cumple los plazos de entrega?), su capacidad de mantener el interés ante adversidad (tiempos de espera ante obras solicitadas).

5) **Capacidad de inferencia**: cruzando adecuadamente datos pueden deducirse conocimientos que, en principio, nada tienen que ver con el contenido de la base de datos. Así, por ejemplo, si habitualmente un usuario tomaba prestados dos libros al mes y los devolvía en plazo, pero ahora no lo hace y se retrasa, puede ser indicativo de que está bajo una presión que le desborda y si, además, ha cambiado el tipo de libros que consultaba puede inducirse el motivo que provoca en él esa presión. La base de datos de la biblioteca no incluye los estados de ánimo de sus usuarios, pero éstos se pueden deducir. La amenaza a la privacidad puede resultar aún más grave cuando se relacionan bases de datos, elaboradas para distintos fines ya que pueden arrojar facetas realmente sensibles. Por tanto, resultan especialmente preocupantes las cesiones o comunicaciones de datos entre empresas o instituciones o el acceso a información disponible en distintos puntos de la red.

Los tres primeros aspectos de peligrosidad que hemos presentado son cuantitativos, es decir, suponen un agravamiento de las amenazas ya padecidas por el individuo cuando el almacenamiento era manual. Los dos segundos son cualitativos, es decir, suponen un nuevo tipo de amenaza prácticamente impensable en el tratamiento manual de la información.

5.2 Legislación de protección de datos personales

Haciéndose eco de esta problemática muchos países desarrollaron leyes que garantizan el derecho a la privacidad, entre ellas cabe destacar la Directiva Comunitaria relativa a la protección de las personas físicas y a la libre circulación de sus datos personales 95/46/CE. La Constitución Española, que data de 1978, preveía el desarrollo de una ley a estos fines, que se materializó en la LORTAD (Ley Orgánica para la Regulación del Tratamiento Automático de Datos personales) de 1992. En 1999, la Ley Orgánica de protección de Datos de carácter Personal 15/99 (LODP) 3 vino a sustituir a la LORTAD, ampliando el ámbito de la protección a ficheros no automatizados, ajustándose a los requisitos de la Directiva Comunitaria y rebajando los niveles de exigencia establecidos por la anterior en algunos aspectos, a fin no dificultar innecesariamente el comercio internacional.

Los principios de protección que aplica pueden agruparse en los siguientes cinco epígrafes:

1) **Calidad de los datos**: establece que los datos almacenados en ficheros informatizados relativos a personas deben ser: adecuados, pertinentes a la finalidad explícita con que se crea el archivo, veraces y mantenidos sólo el tiempo en que son realmente necesarios. Considera expresamente compatibles con la finalidad del fichero el tratamiento posterior de los datos con fines históricos, estadísticos o científicos.

2) **Datos especialmente protegidos**: detecta como especialmente sensibles y, por tanto, merecedores de una protección especial datos relativos a ideología, religión, afiliación sindical, creencias, origen racial o étnico, salud y vida sexual, así como los relativos a infracciones penales o administrativas.

3) **Información a los interesados**: en el impreso de captura de datos para la creación de una base de datos se debe informar de la finalidad para la que se solicitan, destinatarios, cesiones o comunicaciones previstas, derecho del propietario de los datos a acceder a estos y a exigir la rectificación, cancelación o de oposición al tratamiento de los mismos, el responsable del fichero al que dirigirse para la satisfacción de los derechos que acabamos de mencionar. Cuando los datos no hayan sido recabados del interesado (por ejemplo, obtenidos de fuentes accesibles al público), se le informará del origen de los datos y de la identidad del responsable del tratamiento así como de los derechos que le asisten.

4) **Seguimiento**: se crea un organismo (La Agencia de Protección de Datos) que ejerce el seguimiento del tratamiento, velando por el cumplimento de la Ley. Posee facultades de inspección con funciones sancionadoras y potestad para inmovilizar los ficheros que no cumplan la normativa. Atiende a los ciudadanos en sus reclamaciones y ejerce funciones formativas e informativas.

5) **Calidad en el tratamiento**: exige la integridad y corrección de los datos, el establecimiento de medidas de seguridad que impidan el uso ilegal de los mismos. Exige el secreto profesional a las personas implicadas en el tratamiento de datos y regula la cesión de datos entre empresas e instituciones, especialmente cuando se realiza entre distintos países.

5.3 Niveles y medidas de seguridad

El reglamento español clasifica los ficheros en tres niveles, en función de la sensibilidad de los datos que incluyen, especificando las medidas técnicas y organizativas a adoptar en cada caso:

RMS: Niveles de seguridad

BAJO

Identificación, sociales, academicos, caracteristicas, transacciones (poco volumen)

MEDIO

Transacciones (gran volumen) hacienda, penales

Alto : ideología, creencias, afiliación sindical, religión, raza o etnia, salud, comportamiento sexual y los ficheros con fines policiales.

**•Nivel básico**: los que contienen solamente alguno o todos los conceptos: Identificación, circunstancias sociales, datos académicos y profesionales, transacciones, características personales, información comercial, siempre y cuando estos datos no permitan inferir un perfil de personalidad.

**•Nivel medio**: los que incluyendo alguno de los datos anteriores, por el volumen de estos u otras circunstancias permiten inferir un perfil individual. Además, los que contienen infracciones penales y administrativas, los relativos a la Hacienda Pública, los financieros o de solvencia económica.

**•Nivel alto**: los que contienen datos relativos a ideología, creencias, afiliación sindical, religión, raza o etnia, salud, comportamiento sexual y los ficheros con fines policiales.

Por centrar la magnitud del problema diremos que el sistema de información de una biblioteca ordinaria que almacene un histórico de consultas alcanza el nivel medio[[1]](#footnote-1).

La figura 19 detalla las medidas contempladas por el reglamento atendiendo a su clasificación como medidas técnicas/organizativas.



Figura 19. Medidas de seguridad técnicas y organizativas

En la figura 20 se recogen, de forma esquemática, las medidas exigibles en cada uno de los niveles de seguridad. Los tres recuadros, dibujados superpuestos en cada esquema, corresponden de menor a mayor a los niveles bajo, medio y alto, respectivamente, y contienen las medidas exigidas para el correspondiente nivel.

Figura 20. Medidas de seguridad exigibles al sistema en función de la sensibilidad de los datos que contiene

5.4 Legislación para bases de datos de entornos específicos

Hay datos especialmente sensibles que están sujetos a legislaciones específicas adicionales a la LOPD. Por ejemplo, para prevenir el blanqueo de capitales, cuando la BD contiene datos económicos[[2]](#footnote-2) hay que conservarlos durante 6 años y exige borrarlos en un determinado plazo. Afecta a casinos, entidades bancarias, compañías aseguradoras, promoción inmobiliaria, asesores fiscales, notarios…

La sanidad está transferida a las comunidades autónomas, lo que dificulta el seguimiento de la legislación que afecta a bases de datos que contienen datos sanitarios[[3]](#footnote-3), como las de hospitales, consultas privadas, farmacias. Hay obligación de mantener el historial a disposición del paciente durante largos periodos y a las autoridades sanitarias para estudios epidemiológicos, y simultáneamente protegida a confidencialidad (incluso de familiares).

Por su parte, en un balance entre el derecho a la privacidad y la seguridad pública, las empresas que dan servicios de comunicaciones[[4]](#footnote-4) electrónicas al públicos (ya sean de imagen, voz, datos) tienen obligación de conservar y poner a disposición judicial los datos relativos a las circunstancias (tipo de comunicación: chat, email, llamada…origen, destino, es decir, número de teléfono o dirección IP del ordenador. Localización física del equipo, cliente que contrató el servicio, fecha, hora de inicio y fin) de cada una de las comunicaciones interpersonales prestadas. En ningún caso se almacena el contenido de la conversación.

Para aquilatar la relevancia de estar alertas de la existencia de legislación aplicable a la BD que estamos gestionando, indicaremos que las penalizaciones por incumplimiento pueden suponer multas para la entidad comprendidas entre 150.000€ y 1500.000€ y entre 60.000€ y 600.000€ para el empleado, además de poder retirar autorización empresa y amonestación pública con inhabilitación cargo durante 10 años para el empleado.

Solución a los ejercicios propuestos

1 Debe residir en un dispositivo de acceso directo toda la información necesaria para deshacer una transacción activa, pues esta operación tiene que poderse realizar sin intervención humana.

2 Porque relanzar no es idempotente

5 Posibilidad de parar el sistema para realizarla. Volumen de modificaciones realizadas en el sistema en cada periodo de tiempo. Existencia del diario y velocidad de restauración a partir de éste. Dependencia de la organización de que funcione su sistema de información. Que exista constancia impresa de los movimientos realizados. Que exista posibilidad de rehacer los datos por otro mecanismo (por ejemplo datos resultados de ejecutar un determinado programa).

7 Se pueden construir 486 claves distintas, esto es 12\*109. Si un agresor no tiene ninguna información adicional sobre la clave, y tarda 3 segundos en probar una clave, tardará 1163 años en probarlas todas.

8 Para cada terminal utilizando el sistema se crea un fichero que alberga las modificaciones, de modo que este trabaja contra la base de datos y contra su propio archivo. Al solicitar la actualización de la base de datos, se procesan los ficheros temporales, unificándolos con la base de datos preexistente. Durante este periodo no se podrán realizar nuevas modificaciones.

1. R. Peña Privacidad en Sistemas de Información Bibliotecarios. Anabad. 2001 [↑](#footnote-ref-1)
2. Reglamento medidas de prevención del blanqueo de capitales Ley 10/2010

   https://www.boe.es/buscar/act.php?id=BOE-A-2010-6737 [↑](#footnote-ref-2)
3. <http://www.protegemostusdatos.com/data/NOTICIAS/Microsoft%20Word%20-> %2012\_Plazos\_conservacion\_datos.pdf [↑](#footnote-ref-3)
4. Ley 121/000128 BOE de 2007 <http://xribas.typepad.com/xavier_ribas/files/PLCD.pdf> Ley general telecomunicaciones 2003. Artículo 33. Reglamento telecomunicaciones 424/2005 [↑](#footnote-ref-4)