TEMA 44

TÉCNICAS Y PROCEDIMIENTOS PARA LA SEGURIDAD DE LOS DATOS

ÍNDICE

1. INTRODUCCIÓN

La información almacenada en ordenadores se ha convertido en uno de los soportes principales de nuestra sociedad. De hecho, esta década se denomina a menudo la “era de la información”. A medida que ha ido aumentando nuestra dependencia respecto a la información legible por máquina, lo ha ido haciendo también la necesidad de proteger dicha información. Como la mayor parte de la información almacenada en ordenadores lo está en forma de bases de datos, es importante analizar en este texto el tema de la protección de la información.

Los datos informatizados son vulnerables a muchos tipos de daños y de abusos. Consecuentemente, cuando se planifica el diseño e implementación de una base de datos, resulta esencial el que se prepare la forma de tratar con cualquier posible situación que pudiera dar como resultado una pérdida de datos. Los distintos tipos de circunstancias que pueden conducir a una pérdida inesperada de información pueden incluir fallos del hardware y del software, errores humanos, e incluso acontecimientos catastróficos (terremotos, meteoros, etc.). Una de las áreas más importantes en lo referente a la protección de las bases de datos, de importancia creciente en los últimos años, es la de la confidencialidad de la información que contienen. La información almacenada en ordenadores suele ser muy sensible, en el sentido de que debe ser protegida contra accesos no autorizados. Se han desarrollado diversos métodos para ayudar a asegurarse de que el acceso a la información se encuentra a salvo bajo el control del software de gestión de datos.

Un tema relacionado íntimamente con el área de la protección de bases de datos es el del mantenimiento de una base de datos en un estado consistente. Se pueden implementar muchos tipos de protecciones para intentar asegurarse de que la información que se deposita en una base de datos esté tan libre de errores como sea posible, y sea consistente con los datos previamente almacenados. De hecho, son muchos los errores que se pueden detectar antes de que la información pase a ser parte integrante de la base de datos, llevándose así a su grado máximo la fiabilidad con la que ésta es capaz de reflejar el mundo real.

2. SEGURIDAD DE LAS BASES DE DATOS

La información almacenada en ordenadores es un recurso valioso, y este valor suele estar relacionado directamente con la capacidad para mantener la confidencialidad, y el nivel que debe alcanzar la protección depende hasta cierto punto de cómo sea de sensible. Por ejemplo, la información relativa a los sueldos del personal de una empresa debería mantenerse fuera del alcance de todos, excepto de unas pocas personas. Incluso los números de teléfonos y las direcciones pueden considerarse lo suficientemente privados como para que sean mantenidos relativamente inaccesibles. La mayor parte de la información financiera se considera también confidencial, ya se trate de datos de cuentas bancarias de individuos, o de informes de ganancias y pérdidas de la entidad. En algunos casos, la confidencialidad financiera puede ser vital para el éxito de una operación de una institución.

Otro aspecto del área de seguridad es la necesidad de proteger la información contra intentos malintencionados, por parte de los que a veces se denominan “delincuentes informáticos". Los medios de comunicación han descrito muchos incidentes en los cuales diversas bases de datos han sido violadas, provocando daños de diferente cuantía. La mayoría de estos incidentes han sido el resultado de inofensivos hackers, cuya única intención era la de “vencer al sistema”. De cualquier forma, en ocasiones se produce algún daño de consideración, como resultado de la destrucción o corrupción de datos valiosos, o de robos a gran escala, causados por la manipulación de una base de datos por individuos para su propio provecho. El desarrollo de la prevención de la delincuencia informática se halla aún en su infancia, y parece que está teniendo lugar una batalla a gran escala entre los que intentan proteger las bases de datos valiosas, y aquellos que están determinados a esquivar todos los mecanismos de protección imaginables, cualesquiera que sean.

Una importante función de un SGBD es la capacidad para ofrecer diversos tipos de mecanismos que proporcionen un cierto grado de seguridad a la base de datos. Sólo existen unos cuantos métodos de protección de uso común, y los diferentes sistemas de bases de datos ofrecen asimismo diversos grados de seguridad en dicha protección. Muchos de los SGBD más pequeños, orientados a microordenadores, no ofrecen ningún tipo de protección en absoluto. Por otro lado, los SGBD para las máquinas más grandes suelen dedicar una parte significativa de su software a la seguridad de las bases de datos.

Los **métodos de protección** más comunes para proporcionar seguridad a una base de datos son los siguientes: 1) **protección mediante contraseñas**; 2) **vistas del usuario**, y 3**) programas de aplicación**. En los siguientes apartados se describe cada una de estas técnicas.

2.1. Protección mediante contraseñas

Con este tipo de protección, a cada usuario autorizado se le da una contraseña o palabra clave que le permite el acceso a una base de datos determinada. Además, una contraseña determinada puede restringir la manera en que el usuario puede interactuar con la base de datos. Por ejemplo, se puede aplicar la protección mediante contraseñas a uno o más de los siguientes niveles:

Nivel del ordenador. A menudo resulta conveniente que un usuario posea, una contraseña que le permita acceder a cualquier parte de un sistema informático. Este tipo de contraseña se suele denominar contraseña de entrada al sistema (logon). Sin embargo, pueden necesitarse contraseñas adicionales para acceder a cualquier paquete de software determinado dentro del sistema.

Nivel del SGBD. A veces es necesaria una contraseña para permitir el acceso a un sistema de gestión de bases de datos, residente dentro del ordenador. Así, incluso si un usuario tuviese permiso para conectarse al ordenador, no sería aún capaz de acceder al SGBD.

Nivel de la base de datos. Cada base de datos individual puede tener asignado su propio conjunto de contraseñas válidas.

Nivel de la tabla. Además de todos los precedentes, cada tabla o fichero individual puede disponer de su propio conjunto de contraseñas de acceso. Así, a cada usuario de la base de datos se le puede dar permiso para acceder sólo a partes específicas de una base de datos completa.

Nivel de columna. Un usuario podría tener acceso sólo a columnas seleccionadas de una o más tablas de la base de datos. Esta es otra forma de restringir el acceso a parte de una base de datos.

Nivel de vista. En lugar de disponer de acceso a una tabla o a un conjunto de columnas, un usuario puede poder acceder a una o más vistas determinadas de una base de datos. El uso de las vistas es un mecanismo muy importante para la limitación de los accesos a la base de datos.

La protección mediante contraseñas puede implementarse a cualquiera de estos niveles. Los dos primeros están bajo el control del sistema operativo del ordenador; los restantes, si existen, son mantenidos por el sistema de bases de datos. A veces se emplea el término **granularidad** para describir el grado hasta el cual un sistema de bases de datos ofrece mecanismos de protección mediante contraseñas. Así, la protección al nivel de columna sería un ejemplo de **granularidad fina** mientras que la que alcanza sólo hasta el nivel de la base de datos sería un ejemplo de **granularidad gruesa**. Se ve claramente que, a medida que el grado de granularidad va variando desde grueso hasta fino, la cantidad de software necesaria para soportarlo va aumentando.

Tipos de autorización de contraseñas

A los usuarios se les pueden conceder distintos tipos de acceso, a toda o sólo a parte de una base de datos. Por ejemplo, a un usuario determinado se le podría conceder el acceso sólo para lectura a una o más bases de datos: este usuario podría inspeccionar los datos, pero de ningún modo modificarlos. Por otro lado, otro usuario podría disponer de acceso para lectura/escritura a los mismos datos. Algunos SGBD permiten que un usuario con posibilidad de acceso de lectura/escritura pueda realizar cualquier tipo de modificación en los datos, mientras que otros distinguen entre accesos para operaciones de inserción, modificación, o borrado.

Para los sistemas de bases de datos que ofrecen acceso multiusuario a una base de datos, existen más tipos de contraseñas. Por ejemplo, un usuario que está modificando una tabla puede querer obtener un derecho de acceso exclusivo a dicha tabla: que ningún otro usuario pueda acceder al mismo tiempo a la

tabla. Este tipo de autorización se analiza con mayor detalle en el apartado relativo al control de concurrencia.

Ejemplos de métodos de autorización de contraseñas.

Existen muchos métodos diferentes para implementar la protección mediante contraseñas. En algunos sistemas de bases de datos, se usan comandos simples para asignar los niveles de autorización de las contraseñas. Estos comandos deberían ser introducidos directamente desde un terminal del ordenador. Por ejemplo, considere el conjunto de comandos siguiente:

```
GRANT READ ACCESS TO ESTUDIANTES
    FOR PASSWORD = "EST_LECTURA"

GRANT WRITE ACCESS TO CLASES
    FOR PASSWORD = "CLA_ESCRITURA"
```

El primero de estos comandos permitiría el acceso para lectura a la tabla ESTUDIANTES mediante la contraseña "EST_LECTURA", mientras que el segundo concedería el acceso para escritura a la tabla CLASES mediante la contraseña "CLA_ESCRITURA".

Otro método para implementar la protección mediante contraseñas a niveles tanto de tabla como de columna consiste en incluir las especificaciones de las contraseñas como parte de la definición de la base de datos.

Cuando un sistema de bases de datos soporta un mecanismo de protección mediante contraseñas, se mantiene una lista de las contraseñas válidas para cada tabla y/o columna de la base de datos.

Cuando un usuario introduce un comando del sistema de bases de datos para acceder a una columna determinada dentro de una tabla, se comprueba la contraseña del usuario en la tabla de autorizaciones correspondiente, para verificar si el usuario está autorizado para realizar ese acceso. Resulta claro que si cada operación de la base de datos se acompaña de este tipo de comprobación, puede gastarse una cantidad considerable de tiempo de CPU. A menudo se trata de un precio que hay que pagar, para poder mantener la seguridad de la base de datos.

Existen muchos otros métodos para crear protecciones mediante contraseñas en bases de datos. En general, puede decirse que no existen estándares; cada SGBD tiene su propia forma particular de implementar esta característica. Independientemente del método utilizado, el objetivo es limitar el acceso a la base de datos a los usuarios autorizados.

2.2. Vistas de bases de datos

Como ya se ha tratado en el apartado anterior, una forma conveniente de limitar el acceso de un usuario a partes específicas de una base de datos consiste en la creación de una vista de la base de datos. Se trata de una imagen lógica de una base de datos, que no tiene por qué corresponderse con la estructura física real de las tablas de la base de datos. El efecto que produce una vista consiste en proporcionar acceso a un usuario determinado sólo a la información específica de interés.

Una vista puede consistir en una proyección sobre una tabla específica, en una selección de determinadas filas de una tabla, o en una combinación de ambas. Además, también puede consistir en una combinación de filas y columnas específicas de dos o más tablas. Y, por último, una o más de las columnas de una vista pueden ser virtuales. Una columna virtual contiene valores que no existen en la base de datos. Estos valores son el resultado de la composición de dos o más columnas reales. Esencialmente, una vista es una "tabla virtual" que se construye a partir de fragmentos de las tablas reales de la base de datos.

Uso de las vistas para restringir el acceso

Existen muchas razones por las que utilizar una vista para restringir el acceso de un usuario a una parte específica de la base de datos. Una de ellas es la conveniencia: puede definirse una vista de manera que un usuario vea, de forma automática, sólo aquellos datos que le resulten de interés. El usuario no necesita preocuparse tanto de los detalles relacionados con una búsqueda compleja, ni tampoco tiene por qué ser consciente de la estructura real de la base de datos. La única imagen de la base de datos que debe preocuparle es la de la vista determinada que le interesa, y ésta consiste siempre en una tabla sencilla con unos pocos nombres de columnas.

Otra importante aplicación de las vistas es para el control del acceso a la información, por razones de seguridad. Cuando a un usuario se le restringe a una vista determinada, sólo tiene disponible esa parte de la base de datos. Si se crean diferentes vistas para las distintas clases de usuarios, se consigue automáticamente un alto grado de control de acceso.

Tipos de accesos a vistas

A una vista se le pueden conceder distintos tipos de autorizaciones de acceso. De este modo, un usuario podría tener acceso para lectura: la información accesible mediante la vista sería sólo legible, no modificable. Por otro lado, a otro usuario podría serle concedido el permiso tanto de lectura como de escritura. Estas autorizaciones se efectuarían asignando contraseñas diferentes a una vista.

2.3. Programas de aplicación

El uso de programas de aplicación es una técnica bastante común de proporcionar seguridad a una base de datos. La mayoría de los sistemas orientados a mainframes, así como un reducido número de los sistemas basados en microordenadores, soportan el empleo de programas de aplicación. Estos lenguajes suelen escribirse en lenguajes procedimentales estándar, como COBOL, o FORTRAN, pero algunos sistemas de bases de datos soportan su propio lenguaje procedimental interno. Los programas de aplicación pueden escribirse en cualquiera de estos lenguajes, con el fin de que realicen una gran variedad de tareas, conjuntamente con el uso de la base de datos.

Si un sistema de bases de datos no soporta mecanismos de protección mediante contraseñas, o si la cantidad de protección incorporada es poco adecuada, las deficiencias pueden ser corregidas mediante programas de protección escritos por el usuario. Así, un programa puede actuar como interfaz entre una persona que está introduciendo datos y la base de datos, incorporando al mismo tiempo a ésta una dosis de seguridad adicional.

Esencialmente, un programa de aplicación puede realizar virtualmente cualquier tipo de protección mediante contraseñas. Sin embargo, un programa de este tipo suele ser bastante complejo, y conlleva el mantenimiento de una lista de autorizaciones para cada tabla de la base de datos y la comprobación de todas las peticiones de acceso a la base de datos, para asegurarse de que se dispone de la autorización correcta.

2.4. Encriptado de datos

La protección mediante palabras clave funciona bien hasta un cierto punto, pero para muchos tipos de información almacenada en bases de datos, esta forma de protección resulta poco adecuada. Una palabra clave sirve de protección sólo mientras siga siendo secreta, y la experiencia ha demostrado que prácticamente cualquier método de protección mediante palabra clave puede rebasarse, si se le dedica el suficiente esfuerzo. Para información que es lo suficientemente delicada como para necesitar protección más allá de la ofrecida por el nivel de la palabra clave, existe una segunda línea de defensa, la de encriptado de los datos. Se trata de una técnica en la cual la información de la base de datos se codifica de forma tal que resulte ininteligible sin la clave de desencriptado. Se ha dedicado una gran cantidad de investigación a esta área, y se han desarrollado muchos métodos diferentes. El tipo más simple de sistema corresponde a una sustitución de un carácter por otro, y la cantidad de tiempo de ordenador que precisa es mínima. Por otra parte, este tipo de código es muy fácil de encriptar y, por tanto, rara vez se utiliza.

Algunos tipos de encriptado son virtualmente invulnerables al desencriptado, pero, desgraciadamente, conllevan una enorme cantidad de tiempo de cálculo. Normalmente, se emplean métodos de compromiso entre ambos extremos, que sean razonablemente seguros, y que utilicen cantidades modestas de tiempo de ordenador.

Algunos sistemas de bases de datos incorporan paquetes de encriptado y desencriptado que pueden ser invocados por el usuario cuando lo desee. Para los sistemas que no ofrecen esta capacidad, muchos vendedores independientes ofrecen paquetes que pueden usarse para codificar prácticamente cualquier tipo de fichero de una base de datos con varios niveles de sofisticación y de complejidad.

3. INTEGRIDAD DE LAS BASES DE DATOS

En el apartado anterior hemos visto la protección de bases de datos en lo que se refiere a los accesos no deseados. Pero también es preciso protegerlas contra la corrupción originada por la presencia de información de poca calidad, que puede definirse como datos no válidos o inconscientes. Los errores de datos pueden aparecer en cualquier momento, desde el instante en que se generan por primera vez, hasta el momento en que se introducen en el ordenador, y existe una gran cantidad de errores posibles. Afortunadamente, es posible proteger la base de datos contra muchos de estos errores, mediante técnicas adecuadas de validación. Por ejemplo, si se introdujese accidentalmente un valor de 141 para un campo EDAD, en lugar de 41, se podría detectar el error mediante una comprobación de validación de todos los valores de EDAD, que se cerciorase de que todos ellos se encontraran dentro de unos límites razonables. Por otro lado, si se introdujese un valor erróneo para EDAD de 29, en lugar de 39, el error probablemente no sería detectado. En los apartados siguientes estudiaremos los diversos métodos empleados para ayudar a asegurar la integridad de la base de datos.

3.1. Restricciones

En general, es posible realizar muchos tipos de comprobaciones diferentes de los datos que se introducen, con el fin de asegurarse de que son válidos. Una condición impuesta sobre un conjunto determinado de datos se suele denominar una restricción o control de integridad. Las restricciones pueden aplicarse bien a *columnas individuales*; a la *relación entre dos columnas diferentes*, normalmente en tablas distintas; o a las *filas de una o más tablas*. Cuando se intente introducir una nueva fila de datos que viole las condiciones especificadas por alguna restricción, se negará la entrada de la misma en la base de datos.

Restricciones automáticas y programadas

Es de suponer que la imposición de restricciones sería llevada a cabo por el sistema de bases de datos automáticamente, a medida que se va introduciendo cada nuevo dato. Si se violase alguna restricción, el dato sería rechazado por el sistema. Una restricción que se gestiona de esta manera tiene la ventaja concreta de que es muy fácil de utilizar, ya que sólo requiere que el diseñador introduzca las líneas apropiadas dentro de la definición de la base de datos. El problema de confiarse únicamente a este tipo de verificación automático de restricciones es que la mayoría de los SGBD tienen capacidades muy limitadas en esta área, y muchos carecen totalmente de ellas. Algunos sistemas proporcionan al usuario una enorme flexibilidad en la especificación de restricciones, pero son una minoría muy escasa.

Existe otro tipo de mecanismo para la especificación de restricciones, mucho más flexible, pero que requiere el gasto de considerable cantidad de trabajo por parte de los diseñadores e implementadores de la base de datos. Este método conlleva el uso de programas de aplicación para el control de la entrada de toda la información en una base de datos. A medida que el programa va recibiendo cada nuevo fragmento de datos, éste es examinado en función de las reglas de la restricción que corresponda verificar al programa. Esta aproximación tiene la ventaja de que permite la creación de programas de entrada de datos para virtualmente cualquier tipo de restricción. La desventaja es que la escritura y depuración de estos programas suele ser un proceso que consume un tiempo considerable.

La solución más frecuente suele ser una combinación de los dos tipos de mecanismos de comprobación. Por ejemplo, podría usarse un programa de aplicación que comprobase el ajuste de los datos de entrada a un determinado conjunto de restricciones, tales como restricciones de rango para

determinadas columnas. A continuación los datos se irían pasando al SGBD, que a su vez los comprobaría realizando otros tipos de controles, como los correspondientes a restricciones de las referencias.

La imposición de restricciones es una herramienta de gran importancia para el control de la consistencia y de la validez de una base de datos, y el diseñador suele intentar incluir en la implementación de la misma tantos controles de integridad como sea posible.

3.2. Procesamiento de transacciones

Un concepto que se asocia con frecuencia con los de protección y seguridad de una base de datos es el de **transacción** de la misma, que puede definirse como se expone a continuación. Cuando se realiza una modificación en una base de datos, refleja un cambio en el sistema del mundo real al que representa. A menudo, la modificación puede conllevar la edición o la alteración de varios datos individuales de la base de datos. Este conjunto completo de cambios sufridos por la misma es lo que se conoce como una transacción.

Para que la información de la base de datos conserve la consistencia, es necesario que todos los cambios que sufre la base de datos como parte de una transacción se realicen al mismo tiempo. Si sólo se efectúa parte de estos cambios, y se aborta el proceso de la transacción, la base de datos pasará a encontrarse en un estado inconsistente. Esta suspensión puede haberse debido a un gran número de motivos: fallos del hardware o del software, errores humanos, o incluso a un suceso catastrófico. Desde el punto de vista de la consistencia de la base de datos, la causa es irrelevante: una transacción abortada puede dar como resultado una base de datos inconsistente.

Muchos SGBD permiten que los usuarios definan el comienzo y el final de una transacción completa, de forma que el sistema pueda saber exactamente dónde empieza y acaba. Por tanto, si se aborta una transacción, por una razón cualquiera, el SGBD puede eliminar más tarde los cambios realizados en la base de datos hasta el instante de la suspensión, devolviendo de este modo la base de datos a un estado consistente. De los sistemas de bases de datos que soportan esto se dice que tienen la *capacidad de procesamiento de transacciones.*

4. RECUPERACIÓN DE BASES DE DATOS

La información almacenada en ordenadores es sensible a un amplio rango de sucesos, que puede ocasionar un daño irrecuperable. Este rango va desde los simples errores humanos hasta los fenómenos naturales catastróficos. Como la sociedad actual ha llegado a depender en gran medida de los ordenadores para el mantenimiento de la información, tanto los individuos como las grandes empresas esperan que su información esté a salvo y segura, y que siempre esté disponible de forma fiable cuando se la necesite. Por tanto, resulta vital que existan procedimientos que garanticen el cumplimiento de estas expectativas.

Independientemente de la forma en que pueda sufrir daños una base de datos, deben existir métodos para la recuperación de prácticamente toda la información (más del 99%). Normalmente, es imposible devolver la base de datos al mismo estado exacto en que se encontraba en el instante del fallo. Sin embargo, casi siempre se puede determinar qué actividades estaba realizando el SGBD cuando aquél tuvo lugar. Esta información puede ser usada posteriormente para devolver la base de datos a un estado consistente que se sabe existía un poco antes del fallo.

4.1. Fuentes de fallos

Se ha invertido una gran cantidad de esfuerzo en el diseño e implementación de técnicas que traten los diversos tipos de fallos que pueden sufrir los ordenadores, y que pueden ocasionar daños en la información almacenada. En cierto sentido, los diseñadores de los sistemas de hardware y software deben ser extremadamente pesimistas, e imaginar tantas posibles formas diferentes de que ocurran desastres, imaginando formas de hacer frente a dichas posibilidades. En los siguientes párrafos se resumen algunos de los tipos de circunstancias que pueden ocasionar daños en una base de datos.

Problemas de hardware

Muchos componentes de los ordenadores son enormemente sensibles a los cambios en el entorno y pueden dañarse fácilmente. Algunas partes son sensibles a la temperatura: una subida de 10 grados a menudo es suficiente para causar un fallo. Otros componentes electrónicos tienen tasas de fallos predecibles, debidas a un amplio rango de cambios aleatorios.

Diversos elementos mecánicos, tales como las superficies, y los brazos de lectura/escritura de las unidades de disco, están sujetos al deterioro físico ocasionado por el uso. De hecho, una de las causas más comunes de daños en las bases de datos es lo que se denomina "aterrizaje de la cabeza", en donde una cabeza de lectura/escritura de un disco entra en contacto físico con la superficie, ocasionándole un daño permanente a ésta, así como a los datos, y en ocasiones a la tranquilidad mental del usuario. Los fallos o subidas de tensión de la red pueden causar en ocasiones daños impredecibles en la información almacenada, y muchas instalaciones están equipadas con reguladores de tensión y fuentes de alimentación de reserva para minimizar los riesgos debidos a estas fuentes.

Problemas de software

Los sistemas operativos y los propios sistemas de gestión de bases de datos son aglomeraciones enormemente complejas de programas. Ocasionalmente pueden presentarse circunstancias inesperadas que produzcan la caída de alguno de estos sistemas, posiblemente resultando en un daño imposible de predecir en diversos ficheros de datos. Esto puede parecer un poco sorprendente, ya que sería de esperar que estos sistemas de software estuvieran completamente libres de errores. Desdichadamente, esto rara vez sucede en la práctica; bajo circunstancias especiales (especialmente en entornos multiusuario) es relativamente frecuente que el software presente problemas de este tipo.

Errores humanos

Las personas son considerablemente más propensas a cometer errores que las máquinas, y existen innumerables formas en las que un error humano puede ocasionar un daño en una base de datos. Por ejemplo, el operador de un ordenador podría introducir un comando inapropiado en el sistema operativo, o en el SGBD, lo que podría dar como resultado una acción inesperada, que conduciría tal vez a la destrucción de datos. O bien un usuario de la base de datos podría destruir accidentalmente un fichero, al emplear un conjunto incorrecto de comandos del SGBD (esto es bastante fácil). Además, se puede dañar fácilmente al hardware del ordenador, si se abusa físicamente de él: el contacto de un dedo sucio con la superficie de un disco flexible puede hacer considerables estragos.

4.2. Técnicas de recuperación

Existen diversos métodos para la restauración de una base de datos corrupta a un estado previo libre de daños. El tipo de técnica de recuperación usado en cada situación determinada depende de varios *factores*, incluyendo los siguientes:

La extensión del daño sufrido por la base de datos. Por ejemplo, si se encuentra que ha sido un único registro el que ha sufrido daños, la técnica de recuperación es trivial, en comparación con el procedimiento de restauración necesario después de un choque de una cabeza.

El nivel de actividad de la base de datos. Las técnicas de recuperación son fáciles de implementar en bases de datos que se modifican con escasa frecuencia. Por el contrario, resulta mucho más difícil y caro el diseño de técnicas de recuperación para bases de datos que se están actualizando continuamente. En este último caso, suele tratarse también de bases de datos de gran importancia para sus usuarios, por lo que es de vital importancia que la recuperación sea rápida.

La naturaleza de la información de la base de datos. Para algunos tipos de datos, la pérdida de una pequeña cantidad de información puede no resultar particularmente crítica. En otras situaciones, tales como bases de datos financieras, no es aceptable ninguna pérdida de datos, independientemente de su cuantía. Los dos tipos de circunstancias requieren muy diferentes aproximaciones en lo que se refiere a fiabilidad y recuperación.

Copias de seguridad de la base de datos

Para poder efectuar cualquier tipo de restauración de una base de datos, es necesaria la realización de copias de seguridad (backups) de la base de datos de forma periódica. Este proceso consiste en la escritura de una copia exacta de la base de datos en un dispositivo magnético separado del que contiene a la propia base de datos. En los sistemas más grandes, este dispositivo suele ser una cinta magnética. En los sistemas basados en microordenadores, puede tratarse de un cartucho de cinta de casete, o de uno o más discos flexibles. Habitualmente, mientras se está generando una copia de seguridad es preciso detener todas las demás actividades de la base de datos.

A menudo se realiza más de una única copia, que luego se almacenan en un lugar lejos del ordenador, y alejadas entre sí, con el fin de que si algún tipo de suceso catastrófico produjese la destrucción del ordenador, al menos una de las copias en cinta no resultase dañada por el mismo suceso. Cuando se trata de bases de datos críticas, como las que guardan información bancaria, suele guardarse al menos una copia en un lugar alejado bastantes kilómetros de la instalación del ordenador. Además, no es raro que se mantengan varias generaciones de copias, para añadir un nivel de seguridad adicional.

Un método sencillo de recuperación

El método más simple de recuperación de una base de datos es el expuesto a continuación. Periódicamente, quizá una vez cada día, se realiza una copia de seguridad de la base de datos. Comenzando a partir del momento en el que se hace cada copia, se lleva manualmente una lista física, o diario (log), de todos los cambios subsiguientes que se efectúan en la base de datos. Si la base de datos es dañada o destruida, para recuperarla es preciso seguir la secuencia de pasos siguiente:

- Reparar el problema de hardware o software que causó la caída del sistema.

- Restaurar la base de datos a partir de la copia de seguridad más reciente. Esto no restaura la base de datos a su estado en el instante en el que tuvo lugar el daño.

- Volver a introducir manualmente en la base de datos los cambios realizados desde que se hizo la copia, usando la lista física.

Diarios de transacciones y restauración/reejecución

Una extensión de la técnica anterior consiste en el mantenimiento automático de un fichero de ordenador, que contenga una lista de los cambios hechos en la base de datos entre dos copias de seguridad consecutivas. Esta lista se conoce como *diario de transacciones*, y se mantiene siempre en un dispositivo físico diferente del que almacena a la propia base de datos. Habitualmente se utiliza para este propósito una unidad de cinta magnética, o una unidad de disco diferente. La razón para usar un dispositivo separado es simplemente que si la base de datos resulta dañada, la causa de dicho daño no tiene por qué afectar a los datos almacenados en un dispositivo físico diferente.

La forma de utilizar un diario de transacciones como ayuda para la restauración es idéntica a la que ya se ha descrito, excepto en la última etapa. En este caso, la restauración de las transacciones anotadas en el diario las realiza una utilidad del SGBD, que devuelve la base de datos al estado inmediatamente anterior al momento del fallo. Este proceso se conoce habitualmente como *restauración/reejecución*.

La clave para el uso con éxito de un diario de transacciones radica en la capacidad del SGBD para reconocer el comienzo y el final de cada transacción. Para cada transacción de la base de datos, el diario contiene marcas de “comienzo de transacción” y “final de transacción”, además de una grabación de los cambios individuales realizados en la base de datos para dicha transacción. La marca de “final de transacción” se graba en el diario sólo después de la conclusión con éxito de la transacción. Así, si una caída del sistema interrumpe el procesamiento de una transacción, no aparecerá ninguna marca de “final de transacción” en el diario. Cuando se realice un proceso de restauración/reejecución, sólo se restaurarán a partir del diario las transacciones completadas, y se generará un informe impreso, que indicará qué transacciones no se han completado y, por tanto, no han sido introducidas en la base de datos.

Para bases de datos extremadamente activas, la técnica de restauración/reejecución puede resultar inadecuada, ya que el reprocesamiento del diario puede llevar varias horas, durante las cuales la base de datos no puede ser usada con normalidad. Si una base de datos es muy activa, esta no disponibilidad puede revelarse intolerable, y será preciso emplear otras técnicas de restauración.

Recuperación por retroceso

La recuperación por retroceso resulta útil en situaciones en las que el procesamiento de la base de datos se ve interrumpido, pero la base de datos en sí no resulta dañada de forma alguna. Un ejemplo de esto podría ser algún tipo de fallo que produzca una terminación anormal de la ejecución del SGBD. Las transacciones en marcha podrían ser abortadas antes de su finalización, y los registros asociados a las mismas quedarían en estados desconocidos, aunque el resto de la base de datos no se vería afectada.

La técnica de recuperación por retroceso requiere que el diario de transacciones contenga imágenes iniciales de cada registro de la base de datos que haya sufrido modificaciones desde la última copia de seguridad. Una imagen inicial es una copia de un registro tal como se encontraba inmediatamente antes de ser modificado como parte de una transacción, es decir, justo antes del inicio de dicha transacción.

El procesado de recuperación por retroceso conlleva que después de que se haya colocado nuevamente en funcionamiento el SGBD, con la base de datos correcta, tal como estaba cuando tuvo lugar la interrupción, se pase a procesar el diario de transacciones. Para cada transacción incompleta anotada en el diario se reemplaza la versión actual del registro de la base de datos por la imagen inicial correspondiente. Así, cada registro de la base de datos que ha sufrido modificaciones durante una transacción no completada es devuelto a su estado inicial, antes del comienzo de la transacción. El resultado de este proceso es la eliminación de la base de datos de todas las huellas de transacciones incompletas, es decir, las que estaban en marcha cuando tuvo lugar la caída.

Para que la recuperación por retroceso pueda funcionar, el diario de transacciones debe contener marcas de “comienzo de transacción” y de “final de transacción” para cada transacción. Cuando se realiza un proceso de recuperación, las transacciones incompletas se detectan por la ausencia de una marca de “final de transacción”.

La cantidad de esfuerzo necesaria para efectuar una recuperación por retroceso puede ser mucho menor que la que se necesita para una recuperación por restauración/reejecución. Por ejemplo, supongamos que se han grabado 1000 transacciones en un diario entre el momento en que se hizo la última copia de seguridad y el instante del fallo (un fallo que no dañe a la base de datos). Supongamos asimismo que en el instante del fallo se encuentran en marcha 5 transacciones. Con la técnica de restauración/reejecución, la base de datos debe ser restaurada a partir de la última copia, por lo que habrá que procesar 995 transacciones. Por su parte, una recuperación por retroceso parte de la base de datos tal como se encuentra, limitándose a deshacer los efectos de las 5 transacciones incompletas.

Recuperación por adelanto

El adelanto es otro tipo de mecanismo de recuperación, que se usa a menudo cuando una base de datos ha sido dañada y debe, por tanto, ser restaurada a partir de una copia de seguridad. Se parece a la técnica del retroceso, y comparte con ésta la ventaja de que es mucho más rápida que el método de restauración/reejecución. Requiere que el diario de transacciones contenga una imagen final de cada registro de la base de datos que ha sido modificado desde la última copia. Una imagen final es una copia de un registro, inmediatamente después de haber sido modificado como parte de una transacción, es decir, en el estado en que se encuentra al finalizar dicha transacción.

En su forma más simple, esta técnica consta de dos *etapas*:

1. Después de un fallo que produce un daño en la base de datos, se utiliza la última copia de seguridad para restaurarla.

2. Se procesa el diario, a partir del punto en que se efectuó la última copia de seguridad. Para cada transacción completada anotada en el diario, se sustituye la versión actual del registro de la base de datos por la imagen final correspondiente.

Esta técnica es considerablemente más rápida que la de restauración/reejecución, ya que la sustitución de un registro por su imagen final lleva mucho menos tiempo que el proceso de recreación de la base de datos completa a partir de la copia de seguridad.

Existen variaciones del método de adelanto básico, diseñadas para mejorar aún más la velocidad de la recuperación de la base de datos. Por ejemplo, el conjunto completo de imágenes finales puede ordenarse primero por número de registro. De esta forma, después sólo hace falta escribir en la base de datos la última imagen final de cada registro. Para los registros con varias modificaciones anotadas en el diario, esto puede suponer un considerable ahorro en tiempo de procesamiento.

5. CONTROL DE CONCURRENCIA

Muchos sistemas de bases de datos son capaces de manejar de forma concurrente a múltiples usuarios. En algunas situaciones cada usuario puede estar interactuando con una base de datos diferente, mientras que en otras pueden ser muchos los usuarios que interactúen con la misma base de datos.

Las bases de datos muy grandes, como, por ejemplo, las asociadas con los sistemas de reserva de billetes para líneas aéreas, pueden ser consultadas por millares de usuarios simultáneamente. Incluso en casos extremos como éste, cada usuario individual no suele ser consciente de estar compartiendo los mismos datos con otras personas. La responsabilidad del correcto intercambio de mensajes entre los usuarios y las bases de datos apropiadas es compartida por el sistema operativo y por el SGBD. En este contexto, un usuario podría referirse tanto a una persona sentada a un terminal, como a un programa por lotes que interactúa con una base de datos.

Cuando varios usuarios acceden concurrentemente a una misma base de datos, existen varios tipos de potenciales conflictos relativos al uso de los datos. Estos conflictos podrían ocurrir, por ejemplo, si un usuario estuviese intentando modificar una información que otros usuarios están intentando leer. Otra posibilidad sería el que dos usuarios pudiesen intentar actualizar simultáneamente el mismo registro; o, peor aún, que un usuario pudiese intentar leer un registro mientras otro está intentando borrarlo. El sistema de bases de datos debe asegurarse de que los conflictos de este tipo se resuelvan de forma que satisfagan las necesidades de los diferentes usuarios, sin crear ninguna confusión durante el proceso. Este tipo de tarea del SGBD se denomina **control de concurrencia**.

Para ilustrar con mayor detalle los tipos de conflictos que pueden surgir, supongamos que dos usuarios, LEE y ESCRIBE, están trabajando concurrentemente con la misma base de datos. LEE introduce una petición para una transacción que lee un registro determinado; aproximadamente al mismo tiempo, ESCRIBE solicita una transacción que modifica ese mismo registro. El resultado que el usuario LEE va a ver dependerá de cuál de las dos transacciones realice en primer lugar el sistema de bases de datos. Si se procesa en primer lugar la transacción completa de LEE, se enviará a éste la versión antigua del registro. Por otro lado, si la primera en procesarse completamente es la transacción para ESCRIBE, el registro será modificado primero, y LEE verá la nueva versión del mismo.

Las transacciones de los usuarios no son procesadas necesariamente en el orden cronológico en el que han sido solicitadas. LEE puede haber solicitado una transacción varios segundos antes de que lo haya hecho ESCRIBE, y, sin embargo, la transacción de ESCRIBE puede que se complete antes. Esta posibilidad existe porque una transacción puede componerse de muchas partes diferentes, cada una de las cuales se procesa por separado. Debido a la complejidad del funcionamiento tanto del sistema operativo como del SGBD, es más que posible que la transacción A comience antes que la B, pero que la B se complete antes que la A.

5.1. Bloqueo de recursos

En el ejemplo precedente, parecía que era el azar quien determinaba qué versión del registro se enviaba a LEE, pero esto no es necesariamente cierto. Para demostrar esto, consideremos la situación que exponemos a continuación. Supongamos que las dos transacciones, que denominaremos también LEE y ESCRIBE, son recibidas prácticamente al mismo tiempo por el SGBD, y supongamos que, por azar, el procesamiento de ESCRIBE comenzase primero. Si la primera acción de dicha transacción es un comando de bloqueo del registro, entonces el registro en cuestión se volverá inaccesible para LEE (o

cualquier otra transacción) hasta que ESCRIBE haya terminado. Por otro lado, si ESCRIBE no bloquease el registro, entonces, incluso a pesar de que ESCRIBE hubiese comenzado primero, a LEE le sería posible acceder a la copia antigua del registro.

Continúa siendo cierto el hecho de que es el azar quien determina si ESCRIBE se efectúa o no antes que LEE. Sin embargo, debido al bloqueo del registro, siempre que una transacción del tipo de ESCRIBE comienza primero, la transacción LEE se retrasará lo suficiente como para asegurarse de que lea la última versión posible del registro. Por tanto, el bloqueo de registros puede mejorar significativamente el rendimiento de una base de datos, concretamente si se trata de una base de datos muy activa.

Niveles del bloqueo de recursos

Los recursos de la base de datos pueden bloquearse a diferentes niveles. La extensión con la que pueden bloquearse se denomina **granularidad del bloqueo**. Por ejemplo, una transacción puede bloquear una base de datos completa durante una actualización (granularidad gruesa). Por otro lado, también es posible bloquear sólo los datos concretos que se están actualizando (granularidad fina).

Tipos de bloqueo

A veces se puede especificar la forma precisa en la que una transacción va a producir un bloqueo. Los dos tipos de bloqueo más comunes son los siguientes:

Acceso denegado. Todos los demás usuarios ven bloqueado su acceso a los datos durante la transacción.

Acceso sólo para lectura. Los usuarios pueden leer el dato mientras está siendo modificado. Este último tipo de acceso sería el preferible para bases de datos muy activas, es decir, las que tienen una frecuencia de transacciones de todo tipo muy elevada. En este tipo de situaciones, un bloqueo completo de los registros podría degradar el rendimiento del sistema hasta unos niveles inaceptables.

5.2. Actualización concurrente

En algunos casos, el bloqueo de recursos resulta absolutamente esencial para el mantenimiento de la integridad de la base de datos. Como ejemplo de este tipo de situación, supongamos que los usuarios X e Y realizan cada uno una solicitud de transacción sobre el mismo registro de una cuenta bancaria, y prácticamente al mismo tiempo. Además, supongamos que:

1. Ambas transacciones corresponden a reintegros, el primero de ellos por un total de 10.000 pesetas, y el segundo de 20.000.
2. El campo del registro correspondiente al balance inicial tiene un valor de 100.000 pesetas.
3. Ambas transacciones son recibidas por el SGBD casi en el mismo instante, y el procesamiento de ambas comienza también casi al mismo tiempo.

Si las transacciones tienen lugar lo suficientemente próximas en el tiempo, podría presentarse perfectamente la situación siguiente: la transacción X hace que se efectúe una copia del registro en el espacio de trabajo X. (Un espacio o zona de trabajo es un área del almacenamiento primario donde se efectúan todas las modificaciones relacionadas con una transacción.) Dentro de este espacio de trabajo, el campo BALANCE se reduce de 100.000 a 90.000. Mientras esto tiene lugar, la transacción Y hace que se realice otra copia del mismo registro en el espacio de trabajo. En este espacio de trabajo, el campo BALANCE se disminuye de 100.000 a 80.000. Mientras tanto, la copia final del registro del espacio de trabajo de X se vuelve a escribir en la base de datos, resultando un valor de BALANCE de 90.000. Finalmente, la copia del registro en el espacio de trabajo de Y se escribe en la base de datos, encima del valor 90.000 escrito anteriormente, dejando un valor definitivo de 80.000.

El valor final del campo BALANCE debería haber sido 70.000, pero, como ya hemos dicho, el resultado obtenido es 80.000. La razón de que se produzca este desastre es que se ha permitido proceder a las dos transacciones sin criterio, y el efecto de la transacción X se ha perdido. La forma de prevenir este tipo de situaciones intolerables consiste en hacer que una de las transacciones bloquee el registro contra

cualquier otro tipo de actividad de actualización, mientras está siendo procesada. Así, si la transacción X bloquea el registro, entonces la Y no podrá seguir adelante hasta que la X haya terminado.

Todo esto está muy bien, pero supongamos que ambas transacciones comienzan exactamente al mismo tiempo. En este caso, ¿se bloquearán la una a la otra, o, si no sucede así, qué pasará exactamente? La respuesta a esta cuestión es que, en el mundo real de los ordenadores, sólo puede ejecutarse una instrucción en cada momento. Como un sistema de bases de datos es un programa que funciona ejecutando secuencias de instrucciones, una de las transacciones debe necesariamente preceder a la otra. Es decir, la transacción que primero suceda será la que bloquee a las demás.

El problema del interbloqueo

Una única transacción de una base de datos puede conllevar la actualización de varios registros. Por esta causa, pueden plantearse varios problemas de control de concurrencia, que exigirán de una gestión especial por parte de los sistemas de bases de datos. Un ejemplo típico de este tipo de problemas es el del interbloqueo (deadlock). En esta situación se tienen dos transacciones, X e Y, que comienzan prácticamente al mismo tiempo. Casualmente, ambas actualizan los registros A y B. El SGBD inicia primero la transacción X, y las primeras instrucciones al SGBD de dicha transacción son las siguientes:

X-1. Bloquear el registro A.
X-2. Leer una copia del registro A en el espacio de trabajo X.
X-3. Actualizar la copia del registro en el espacio de trabajo.
X-4. Escribir la versión actualizada de A en la base de datos.
X-5. Bloquear el registro B.
X-6. Leer una copia del registro B en el espacio de trabajo.

Supongamos que la transacción Y se inicia en algún momento durante las etapas anteriores, y también que esta transacción debe acceder al registro B antes que al A. Las primeras instrucciones de esta transacción son las siguientes:

Y-1. Bloquear el registro B.
Y-2. Leer una copia del registro B en el espacio de trabajo Y.
Y-3. Actualizar la copia del registro en el espacio de trabajo.
Y-4. Escribir la versión actualizada de B en la base de datos.
Y-5. Bloquear el registro A.
Y-6. Leer una copia del registro A en el espacio de trabajo.

Supongamos ahora que la instrucción Y-1 tiene lugar después de la instrucción X-1, y (éste es el punto crítico) antes de la X-5. Cuando la transacción X alcance la instrucción X-5, no podrá seguir adelante, ya que el registro B ha sido bloqueado por la transacción Y.

Mientras tanto, la transacción Y avanza hasta que encuentra la instrucción Y-5, momento en el que debe detenerse, ya que la transacción X ha bloqueado el registro A. En este instante, ambas transacciones están suspendidas, esperando que un registro sea desbloqueado por la otra, y ninguna de las dos puede moverse. Esta situación se conoce como interbloqueo, y si no fuera por la intervención del SGBD, ambas transacciones esperarían pacientemente hasta la próxima caída del sistema.

Desbloqueo de un interbloqueo. Se han propuesto varios métodos para resolver el problema del interbloqueo. Aunque resulte extraño, el que se ha demostrado como más eficiente es el que permite que el bloqueo se produzca. Muchos SGBD pueden detectar la existencia de una situación de este tipo, manteniendo un examen constante de los recursos que van siendo solicitados por cada transacción en curso. Cuando se detecta una situación de interbloqueo, el sistema de bases de datos la resuelve terminando una de las transacciones. Para hacerlo debe volverse atrás en los pasos dados por la transacción. Esto puede hacerse mediante un proceso de recuperación por adelanto, que emplee imágenes iniciales, como ya se ha descrito anteriormente en este mismo capítulo.

5.3. Redes locales y concurrencia

Uno de los avances más recientes en el área de los microordenadores es el desarrollo de las redes locales. En este tipo de situaciones, dos o más microordenadores se enlazan mediante una red con unos recursos comunes, tales como una impresora, o una unidad de disco duro.

Dentro de la estructura de una red local, resulta posible que varias copias individuales del mismo SGBD intenten acceder concurrentemente a la misma base de datos. Aquí, cada copia del SGBD se ejecuta en un microordenador separado, independiente de los demás. En este tipo de situaciones, pueden existir los mismos tipos de problemas de control de concurrencia que ya se han analizado con anterioridad, tales como una actualización concurrente, etc. Sin embargo, la solución a estos problemas debe adoptar formas diferentes, debido a la independencia de cada SGBD.

Por ejemplo, el bloqueo de registros sigue siendo una necesidad, pero en este caso el hecho de que una copia del SGBD bloquee un registro dado no va a ser conocido automáticamente por el resto de los SGBD. Es necesario crear algún tipo de mecanismo externo a todos los SGBD que indique qué registros o ficheros están bloqueados en un momento dado.

El rápido desarrollo de las redes locales ha obligado a los SGBD basados en microordenadores a un replanteamiento de sus productos. Como resultado de esto, muchos de estos sistemas han sido diseñados para funcionar adecuadamente en entornos multiusuario. Aún tiene que hacerse una gran cantidad de trabajo en esta área, ya que la mayoría de los SGBD se escriben aún para accesos monousuarios a las bases de datos. Desgraciadamente, aunque estos sistemas parecen funcionar correctamente en principio en entornos multiusuario, a menudo producen resultados desastrosos.

5.4. Entornos de bases de datos distribuidos

El procesamiento distribuido es una situación en la que las diferentes partes de una base de datos se mantienen en diferentes lugares físicos, normalmente bastante lejanos unos de otros. Aunque los sistemas distribuidos son de difícil implementación, cada vez son más populares, debido a las numerosas ventajas que ofrecen. Por ejemplo, cada punto de ubicación (que a veces se denomina nodo) es responsable del mantenimiento de la parte de la base de datos total que resulta de interés para su propio personal. El acceso a esa información local suele ser mucho más rápido de lo que sería si los datos estuviesen almacenados en un lugar alejado. Además, si un ordenador falla, no afecta necesariamente al funcionamiento del resto de los lugares. Un buen ejemplo de esto es el de un banco con muchas sucursales, en donde cada sucursal es responsable del mantenimiento de sus propios datos, además de disponer de acceso a la información almacenada en el resto de los lugares.

Cuando una base de datos está físicamente distribuida entre varios lugares, los problemas de control de concurrencia se complican enormemente, debido a muchos factores, y muchas de las situaciones todavía no se han resuelto totalmente. Consideremos, por ejemplo, las complicaciones que pueden producirse cuando, por conveniencia, se duplican algunas partes de la base de datos en varios lugares diferentes. Dos usuarios pueden estar trabajando concurrentemente con copias diferentes de los mismos datos, cada uno de ellos realizando cambios diferentes en la información. Sólo el problema de mantener las diversas copias sincronizadas está ya lejos de ser trivial.

Además, muchas de las características asociadas a las comunicaciones de datos complican aún más el problema del control de concurrencia. Por ejemplo, los mensajes electrónicos tardan cantidades significativas de tiempo en ir de un lugar a otro, en parte debido a los distintos tipos de actividad de conmutación relacionados con las transferencias. En estas circunstancias, el propio concepto de concurrencia presenta una dimensión completamente nueva. Estos puntos se mencionan aquí por la simple razón de que el procesamiento distribuido se está convirtiendo en un tema importante dentro del campo de la gestión de bases de datos, y es, por tanto, muy importante ser consciente de su existencia.

6. BIBLIOGRAFÍA

Gary W. Hansen
Diseño y Administración de Bases de Datos
Prentice Hall, 1998

Korth
Fundamentos de bases de datos
Mc Graw-Hill, 1993