Davante

10

GESTIÓN DE BASES DE DATOS

Políticas de bloqueo de tablas

ÍNDICE

ISBN: 978-84-19758-41-5

OBJETIVOS

Comprender los conceptos de simultaneidad y concurrencia en el ámbito de una base de datos.

Entender las situaciones que pueden tener lugar al acceder de forma concurrente a la información.

Conocer las políticas de bloqueo que existen en una base de datos.

Identificar los diferentes tipos de bloqueos de datos que pueden tener lugar en una base de datos.

/ 1. Introducción y contextualización práctica

Es habitual que una base de datos se utilice al mismo tiempo por diferentes usuarios, especialmente en organizaciones de cierto tamaño o entidad.

De esta forma, los usuarios que utilizan la base de datos suelen tener acceso a la misma información (aunque no siempre es así), lo que puede dar lugar a que distintos usuarios deseen modificar los mismos datos al mismo tiempo. Es obvio que este hecho puede ocasionar problemas en la integridad de los datos del sistema, por lo que debe tratarse y gestionarse de alguna manera.

En este tema, se estudiará en qué consiste la concurrencia en el acceso a la información y la manera en que se garantiza la consistencia e integridad de la información que contiene el sistema. Esto es algo que habitualmente se realiza a través de las políticas de bloqueo.

A continuación, vamos a plantear un caso práctico, a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio, en el que planteamos la contextualización práctica de este tema. Encontrarás su resolución en el apartado 'Resumen y resolución del caso práctico'.

Fig.1. Las políticas de bloqueo en una base de datos son fundamentales para impedir ciertos errores y problemas, especialmente en entornos multiusuario.

Audio intro. "Acceso simultáneo a los datos"

https://bit.ly/3ip2Qgt

/ 2. Concepto de concurrencia

En informática, se habla de **concurrencia** cuando existen, de forma simultánea, varios procesos en ejecución. En un sistema informático, la concurrencia se puede encontrar en varios puntos:

- Funcionamiento en paralelo de varios periféricos (teclado, ratón y monitor, por ejemplo).
- Ejecución de distintos procesos en diferentes procesadores (equipos con procesadores múltiples).
- Sistemas distribuidos.
- Ejecución en paralelo de instrucciones.

Decimos que **un sistema es concurrente** cuando permite realizar distintas acciones en paralelo en el mismo instante de tiempo. Por ejemplo, un equipo informático como un ordenador permite, habitualmente, ejecutar varios procesos de forma simultánea. Ejemplos de sistemas concurrentes son:

- Sistemas operativos.
- Sistemas de gestión de bases de datos.
- Sistemas distribuidos.
- Sistemas de tiempo real.

En el caso de las bases de datos, la concurrencia puede ocasionar ciertos problemas, por lo que debe tratarse y gestionarse de forma adecuada, implementando mecanismos de control, como estudiaremos a lo largo de este tema.

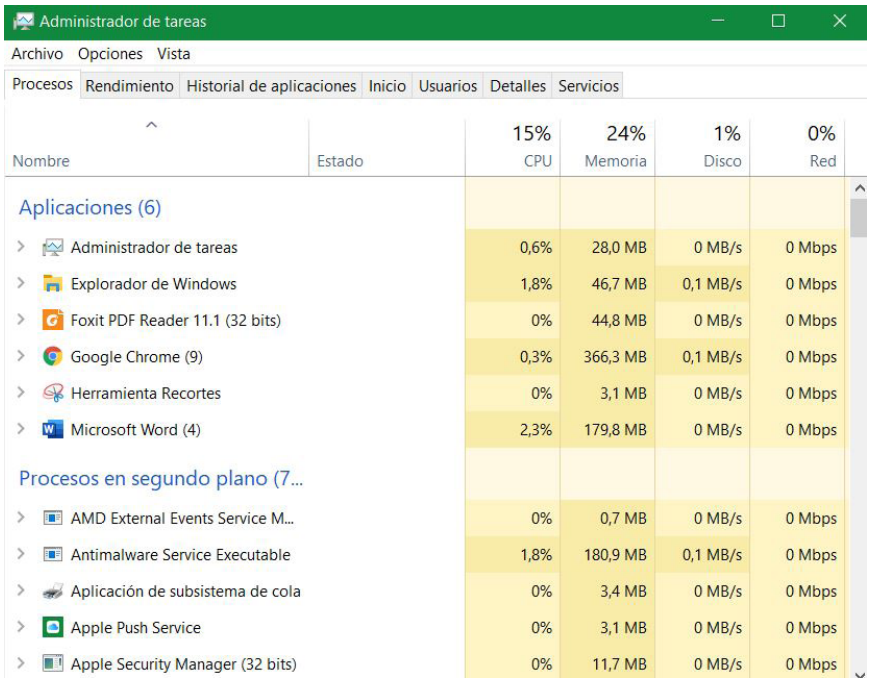

Fig.2. Un equipo informático permite ejecutar varios programas de forma concurrente, algo que puede verificarse desde el administrador de tareas.

Enlaces de interés...

En el siguiente vídeo, podrás ampliar el concepto de concurrencia y entender en qué consiste el concepto de paralelismo, que es un tipo específico de concurrencia. https://www.youtube.com/watch?v=kMr3mF71Kp4

/ 3. Concurrencia en bases de datos

En una base de datos multiusuario, puede darse la situación de que varios usuarios introduzcan transacciones, que se ejecuten de manera concurrente y que pretendan **leer o modificar los mismos datos o elementos de la base de datos**.

En la figura 3 se muestra un problema de concurrencia típico en un sistema multiusuario. Lógicamente, las dos transacciones del ejemplo no pueden acceder al mismo dato al mismo tiempo, ya que la información no sería consistente.

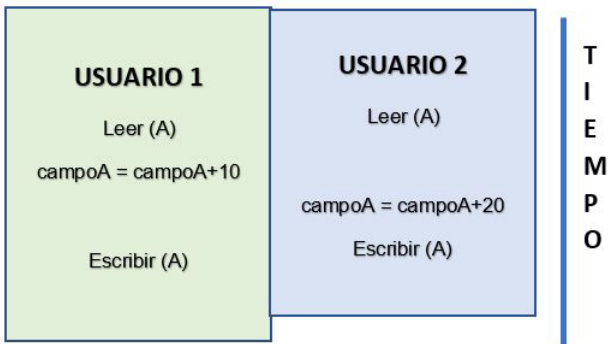

Fig.3. Ejemplo de problema de concurrencia en una base de datos. Las transacciones, en este caso, no pueden modificar el mismo dato al mismo tiempo, ya que la información no sería consistente.

Por lo tanto, pueden aparecer problemas en una base de datos si las transacciones concurrentes se llevan a cabo sin ningún tipo de control. Para una base de datos actual, es fundamental manejar adecuadamente la concurrencia de datos, estableciendo los mecanismos que sean necesarios para garantizar, además, la **integridad y consistencia de la información**.

En la gran mayoría de los SGBD existentes en el mercado está garantizada la concurrencia de los datos, así como la consistencia y la integridad de estos, para lo que se utilizan técnicas de control de concurrencia.

Las **técnicas de control de concurrencia** garantizan que se pueda llevar a cabo, por tanto, la ejecución de un conjunto de transacciones. Se pueden llevar a cabo mediante distintas técnicas:

- Bloqueos.
- Marcas de tiempo.
- Validación.

Estas técnicas pueden clasificarse en pesimistas u optimistas. Escucha el siguiente audio para conocer en qué se diferencian:

Audio 1. "Técnicas pesimistas y optimistas"
https://bit.ly/3KUeg8j

/ 4. Problemas de concurrencia en una base de datos

Los problemas que pueden surgir cuando dos transacciones diferentes intentan acceder a los mismos datos de forma concurrente son los siguientes:

- **Lectura sucia (*Dirty Read*)**: Este problema aparece cuando una segunda transacción lee los datos que están siendo alterados por otra, antes de que se ejecute el 'COMMIT', es decir, antes de que se confirmen los datos.

会话1	会话2
begin	begin
	update table set age = 10 where id =1
select age from table where id =1	
commit	commit

Fig.4. Ejemplo de problema de lectura sucia. Cada columna representa una sesión o usuario.

- **Lectura no repetible (*Non – Repeatable Read*):** Este problema tiene lugar cuando se consulta el mismo dato en dos ocasiones dentro de una misma transacción, verificándose la segunda vez que el valor ha sido modificado o eliminado por otra transacción.

会话1	会话2
begin	begin
select age from table where id =1	
	update table set age = 10 where id =1
	commit
select age from table where id =1	
commit	

Fig.5. Ejemplo de problema de lectura no repetible. Cada columna representa una sesión o usuario.

- **Lectura fantasma (*Phantom Read*):** El problema de la lectura fantasma aparece cuando una transacción está ejecutándose sobre un grupo de datos (filas) y, al llevarse a cabo de nuevo, aparecen nuevas filas dentro de dicho conjunto que no existían cuando se inició la transacción.

- **Problema de la modificación perdida**: Este problema surge cuando varias transacciones modifican el valor de una misma fila, considerando el valor inicial de la misma. De esta forma, la última modificación que se lleve a cabo sobrescribirá las realizadas por las transacciones anteriores.

En el siguiente vídeo, se muestran varios ejemplos de estos problemas que pueden tener lugar en una base de datos:

Vídeo 1. "Ejemplos de problemas de concurrencia"

https://bit.ly/3MZeugn

/ 5. Caso práctico 1: "Consultas a tabla PEDIDOS y PRODUCTOS"

Planteamiento: Un compañero de trabajo te pide ayuda para identificar distintos problemas de concurrencia que se han detectado en el acceso a la información en una base de datos.

Las transacciones que han tenido lugar, en cada caso, se muestran en las tablas 1, 2 y 3.

TRANSACCIÓN 1	TRANSACCIÓN 2
UPDATE cuenta SET saldo = saldo + 1000 WHERE id = 2;	
	SELECT saldo FROM cuenta WHERE id = 2;

Tabla 1. Ejemplo de transacciones concurrentes 1.

TRANSACCIÓN 1	TRANSACCIÓN 2
SELECT saldo FROM cuenta WHERE id = 2;	
	UPDATE cuenta FROM saldo = saldo + 1000 WHERE id=2;
SELECT saldo FROM cuenta WHERE id = 2;	

Tabla 2. Ejemplo de transacciones concurrentes 2.

TRANSACCIÓN 1	TRANSACCIÓN 2
SELECT SUM(saldo) FROM cuenta;	
	INSERT INTO cuenta VALUES (10, 5000);
SELECT SUM(saldo) FROM cuenta;	

Tabla 3. Ejemplo de transacciones concurrentes 3.

Nudo: ¿Podrías identificar el tipo de problema que está teniendo lugar en cada caso?

Desenlace: Según lo estudiado anteriormente, los problemas de concurrencia que están teniendo lugar son los siguientes:

- Tabla 1: Lectura sucia.
- Tabla 2: Lectura no repetible.
- Tabla 3: Lectura fantasma.

/ 6. Niveles de aislamiento

Para poder gestionar adecuadamente los problemas de concurrencia y evitar su aparición, es posible implementar distintos niveles de aislamiento sobre los datos, definidos en el estándar de SQL:

- **Lectura no confirmada (Read Uncommitted):** En este nivel no se realiza ningún aislamiento o bloqueo sobre los datos. Por tanto, todos los cambios hechos por una transacción pueden verse desde otras.
- **Lectura confirmada (Read Committed):** Una transacción sólo puede leer datos que hayan sido confirmados por otras transacciones.
- **Lectura repetible (Repeatable Read):** Ningún registro al que se haya accedido mediante una cláusula 'SELECT' puede ser modificado en otra transacción.
- **Serializable:** Las transacciones están completamente aisladas unas de otras y se ejecutan en serie, unas tras otras. Únicamente se ejecutan de forma concurrente en caso de que el SGBD pueda determinar que no habrá conflictos de acceso a los datos entre ellas.

Estos niveles de aislamiento **controlan el nivel de bloqueo del sistema sobre los datos**. Es decir, los SGBD proporcionan estos niveles de aislamiento mediante técnicas de bloqueo de los datos, que estudiaremos a continuación.

En la tabla 4 se muestran los problemas de lectura que pueden tener lugar en cada nivel de aislamiento, de acuerdo con las características de cada uno, indicadas anteriormente.

NIVEL	LECTURA SUCIA	LECTURA NO REPETIBLE	LECTURA FANTASMA
Lectura no confirmada	SÍ	SÍ	SÍ
Lectura confirmada	NO	SÍ	SÍ
Lectura repetible	NO	NO	SÍ
Serializable	NO	NO	NO

Tabla 4. Problemas que pueden tener lugar en cada nivel de aislamiento.

Enlaces de interés...

En el siguiente vídeo, podrás observar cómo se muestran ejemplos de lo estudiado hasta ahora en el tema:
https://www.youtube.com/watch?v=AoA93mZyEwg

/ 7. Bloqueos. Niveles de bloqueo

Mediante la aplicación de bloqueos, se consigue **implementar el control de concurrencia en las bases de datos**, independientemente del tipo que sean, ni el SGBD que utilicen.

De esta forma, cuando una transacción accede a cualquier objeto o campo de la base de datos, enviará de forma previa una solicitud al sistema para que pueda bloquearlo.

Así, si una segunda transacción desea modificar el objeto o campo, deberá esperar a que la primera transacción lo libere.

Fig.6. Los bloqueos en la base de datos pueden darse a varios niveles.

Los bloqueos pueden realizarse a **varios niveles** dentro de la base de datos:

- **Bloqueo a nivel de fila:** Se trata de realizar bloqueos a nivel de fila, para evitar que sea modificada por otra transacción. Reduce en gran medida los problemas de concurrencia, pero causa gran sobrecarga de bloqueo en el sistema.
- **Bloqueo a nivel de página:** El compromiso entre la reducción de los problemas de concurrencia y la sobrecarga del sistema es medio. No sobrecarga tanto como el bloqueo a nivel de fila, pero también se producen mayores problemas.
- **Bloqueo a nivel de tabla:** En este caso, la probabilidad de que ocurra un problema de bloqueo es mayor, pero, a cambio, se sobrecarga en menor medida el sistema, por lo que los bloqueos se ejecutan de una manera más rápida y ágil. Es el tipo de bloqueo más comúnmente utilizado, aunque hay sistemas que implementan bloqueos a nivel de fila por defecto.
- **Bloqueo a nivel de base de datos:** No resulta operativo realizar un bloqueo completo sobre una base de datos, ya que el sistema resultaría difícilmente manejable. Únicamente sería viable en sistemas con muy pocos usuarios, algo que no suele tener lugar.

8. Políticas de bloqueo de tablas. Comandos de bloqueo

El hecho de bloquear una tabla significa que se impide al resto de transacciones realizar cualquier tipo de operación de lectura y/o escritura sobre la misma, hasta que no se proceda a liberar este bloqueo.

8.1. Comandos utilizados para bloquear tablas

Para llevar a cabo el bloqueo de tablas, se utiliza el comando 'LOCK', con la sintaxis mostrada en la figura 7. Esta sintaxis y sus opciones pueden variar ligeramente, en función de la base de datos que se esté utilizando (Oracle, MySQL, etc.).

```
LOCK TABLES
    nombre_tb [[AS] alias] tipo_bloqueo
    [, nombre_tb2 [[AS] alias] tipo_bloqueo]...;
```

Fig.7. Sintaxis utilizada para bloquear tablas.

Así, para bloquear una tabla, se debe indicar su nombre o alias y el tipo de bloqueo que se desee realizar.

Enlaces de interés...

En el siguiente enlace, podrás observar cómo se utiliza este comando en Oracle:
https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_9015.htm

Para desbloquear tablas, se utiliza la instrucción '**UNLOCK TABLES**', con la que se desbloquearían todas las tablas a la vez.

8.2. Tipos de bloqueo de tablas

El comando 'LOCK TABLES' permite aplicar dos tipos de bloqueo distintos:

- **'READ [LOCAL]':** Si se lleva a cabo este tipo de bloqueo, el usuario que lo ha realizado podrá leer los datos, pero ni él ni ningún otro usuario podrán realizar operaciones de escritura en esa tabla.

  Este tipo de bloqueo puedes ejecutarlo varios usuarios de manera simultánea y permite que cualquier otro pueda leer las tablas que se han bloqueado. El comando 'LOCAL' permite este hecho.

- **'[LOW PRIORITY] WRITE':** En este caso, el usuario que realiza el bloqueo puede realizar operaciones de lectura y de escritura en la tabla. Los bloqueos de escritura tienen mayor prioridad que los de lectura por defecto.

  Si se utiliza la opción 'LOW_PRIORITY' se permite que se lleven a cabo bloqueos de lectura antes que un bloqueo de escritura.

/ 9. Bloqueos y transacciones

Según hemos podido comprobar, **la utilización de bloqueos está directamente asociada a las transacciones**, de manera especial para tablas de tipo transaccional, como pueden ser 'InnoDB' o 'Cluster'.

> **Enlaces de interés...**
>
> Recuerda en lo que consiste 'InnoDB' a través del siguiente enlace: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/que-es-innodb/

9.1. Bloqueos en tablas de tipo InnoDB

En las tablas de este tipo, los bloqueos que se llevan a cabo por defecto son a nivel de fila, por lo que se permite que varias sesiones o usuarios puedan bloquear distintas filas de manera simultánea.

En este tipo de tablas, se puede aseverar que cada operación SQL que se lleve a cabo puede considerarse como una transacción, siempre que la variable 'AUTOCOMMIT' esté activa.

Para ello, se deben utilizar los comandos 'START TRANSACTION' o 'BEGIN' para comenzar la transacción y los comandos 'COMMIT' o 'ROLLBACK' para su finalización, como estudiamos en el tema anterior.

9.2. Tipos de bloqueo en InnoDB

Existen dos tipos básicos de bloqueo, que se muestran en la tabla 5.

TIPO DE BLOQUEO	DESCRIPCIÓN
Bloqueo exclusivo	Mediante la utilización de bloqueos exclusivos, se impide que un recurso en concreto sea compartido. De esta manera, cuando una transacción bloquea un determinado recurso de manera exclusiva, solo puede realizar modificación sobre el mismo dicha transacción, hasta que el recurso se libera. Permite a una transacción bloquear filas para poder modificarlas. En este caso, no se pueden llevar a cabo varios bloqueos exclusivos de manera simultánea.
Bloqueo compartido	De manera contraria al bloqueo exclusivo, en el bloqueo compartido, como su nombre indica, se permite compartir el recurso sobre el que se está actuando, siempre en función del tipo de operación que se está llevando a cabo. Permite a una transacción leer filas, pero no modificarlas. Varias transacciones pueden llevar a cabo bloqueos sobre las mismas filas, pero ninguna podrá modificarla. De esta manera, pueden existir recursos compartidos para varias transacciones en paralelo.

Tabla 5. Tipos de bloqueo en tablas 'InnoDB'.

9.3. Bloqueo de múltiple granularidad

Las tablas de tipo 'InnoDB' también pueden soportar bloqueos de múltiple granularidad. A través de este tipo de bloqueo, **se permite a una transacción indicar que va a bloquear filas de la tabla** para llevar a cabo operaciones de escritura o de lectura.

De esta manera, la gestión de posibles problemas de concurrencia es más sencilla, evitando, además, que aparezcan *deadlocks*. Esto se debe a que permite compartir el bloqueo de una tabla a varias transacciones, ya que se produce a nivel de fila. Si una de ellas lleva a cabo operaciones de modificación, la tabla quedará bloqueada.

Este tipo de bloqueo se puede llevar a cabo mediante distintas sentencias:

- **'SELECT... LOCK IN SHARE MODE':** Para llevar a cabo bloqueos de lectura sobre las filas que se encuentren afectadas por la sentencia 'SELECT'.

  Por ejemplo, si quisiéramos insertar un nuevo jugador en un equipo, resulta útil asegurarse antes de que ese equipo existe y que nadie lo va a borrar. Podríamos bloquearlo mediante el ejemplo mostrado en la figura 8. Así, se consigue un bloqueo de lectura que impedirá modificar el equipo.

```
SELECT * FROM equipos
WHERE nombre = 'ID_EQUIPO'
LOCK IN SHARE MODE;
```

Fig.8. Ejemplo de uso de bloqueo.

- **'SELECT... FOR UPDATE':** Para llevar a cabo bloqueos de escritura, bloqueando aquellas filas afectadas por el 'SELECT'.

  Siguiendo el ejemplo anterior, si deseamos introducir un nuevo jugador, debemos incrementar el campo 'num_jugadores', dentro de la tabla 'contador', que podría ser modificada de manera simultánea por otro usuario. Para ello, bloquearíamos para escritura la tabla 'contador'. De esta forma, hasta que no se libere este bloqueo, no se podrá acceder al valor de este campo, que ya estará actualizado y será correcto.

```
SELECT num_jugadores
FROM contador FOR UPDATE;

UPDATE contador
SET num_jugadores = num_jugadores + 1;
```

Fig.9. Ejemplo de uso de sentencia 'SELECT... FOR UPDATE'.

/ 10. Caso práctico 2: "Padres e hijos"

Planteamiento: Un cliente nos ha solicitado ayuda para realizar una operación en la tabla 'hijo'. Necesita introducir una nueva fila en la tabla, pero desea asegurarse de que el nuevo registro tiene asociado un padre existente en la tabla 'padre'.

Id_padre	Nombre

Tabla 6. Columnas de la tabla 'padre'.

Id_hijo	Nombre	Id_padre

Tabla 7. Columnas de la tabla 'hijo'.

Nudo: Desea asegurarse de que, cuando introduzca el nuevo hijo, nadie habrá borrado el padre asociado de la tabla padre. ¿Cómo se puede asegurar de ello?

Desenlace: En base lo estudiado anteriormente, para asegurarse de que ningún usuario elimina el padre asociado, debe realizar un bloqueo de lectura. Este bloqueo puede llevarse a cabo mediante la siguiente sentencia:

```
SELECT *FROM padre
WHERE nombre = 'Francisco Pérez'
LOCK IN SHARE MODE;
```

De esta forma, hasta que el bloqueo no se libere, ningún otro usuario podrá eliminar los datos de esa fila.

/ 11. Resumen y resolución del caso práctico de la unidad

En este tema, hemos estudiado el concepto de concurrencia y, en particular, de la concurrencia en entornos de bases de datos. Se ha relacionado este asunto con los distintos niveles de aislamiento, que han servido para introducir los bloqueos. Se ha analizado en qué consisten y los niveles de bloqueo existentes, así como las distintas políticas de bloqueo de tablas.

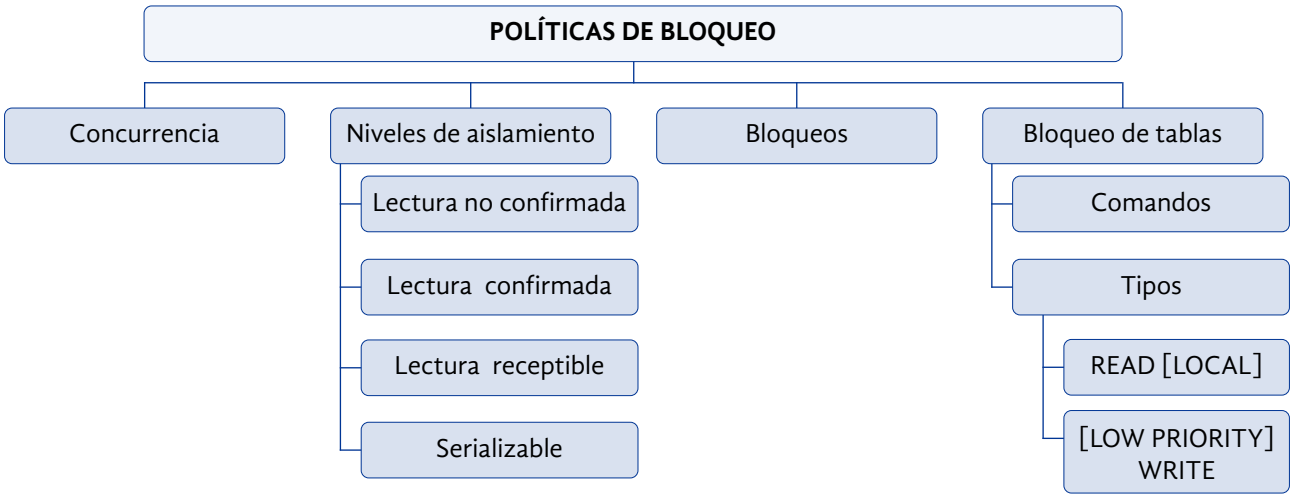

Fig.10. Esquema resumen del tema.

Resolución del caso práctico de la unidad

En este caso y, como hemos visto y estudiado a lo largo del tema, se garantizará la consistencia de la información que contenga la base de datos, para evitar errores y pérdidas de información.

En los casos concretos planteados, se activarán los mecanismos de bloqueo que se deseen para evitar que un mismo dato sea modificado a la vez por dos usuarios distintos.

/ 12. Bibliografía

López, I.; Castellano, M.J. y Ospino, J. (2011). *Bases de datos.* Madrid, España: Garceta.

Oppel, A. (2009). *Databases: A Beginner's Guide.* Madrid, España: McGraw-Hill.

Cabrera, G. (2011). *Sistemas gestores de bases de datos*. Madrid, España: Paraninfo.

Elmasri, R.; Navathe, S. (2007). *Fundamentos de Bases de Datos* (5.a. ed.). Madrid, España: Pearson Addison-Wesley.