TRANSACCIONES,

CONCURRENCIA

Y

RECUPERACIÓN

EN

BASES DE DATOS

Índice

1- Control de Concurrencia

En las siguientes lineas se explicarán las diferentes técnicas para el control de la concurrencia. Pero antes de entrar en eso se dejará claro el concepto de Concurrencia, según el diccionario de la Real Academia de la Lengua Española. Concurrencia en su tercera acepción se dice lo siguiente , *Coincidencia, concurso simultáneo de varias circunstancias.* Pero claro esta definición es en el sentido general de la palabra. Pero en el caso de las Bases de Datos y los sistemas de gestión de bases de datos, básicamente se podría decir, que la concurrencia aparece cuando un elemento X, de una base de datos es manipulada por dos agentes , en nuestro caso Transacciones, de forma simultánea. La verdad es que dicho de esta manera no se ven muchos problemas en esto, pero la concurrencia trae asociado el problema de que un elemento X, se le obtiene su valor por dos "transacciones", y luego estas dos actualizan a la vez el valor de X . Esto podría provocar que el valor de X , no sea el que deseamos y produzca problemas de de consistencia y demás .
Ahora comenzamos a describir las técnicas que explicaremos.
Tenemos técnicas de bloqueo para el control de concurrencia, Control de concurrencia basado en ordenamiento por marca de tiempo , Técnica para el control de concurrencia de multiversión y Técnica para el control de concurrencia de validación (optimista)

1.1- Técnicas de bloqueo para el control de concurrencia.

En esta sección se plantearán todas aquellas técnicas que se basan en el bloqueo de los elementos de información. Utilizando para ello el concepto de candado, que puede ser una variable asociada a un elemento X, que indicará el estado en que se encuentra. En nuestro caso nos referiremos siempre como candado(X), indicando que es el candado asociado al elemento X.

Pero en esta parte no sólo haremos esto, sino también se estudiará los problemas asociados a este conjunto de técnicas.

Este punto se divide en tres apartados, el primero estudia los tipos de candados y sus características, el segundo versará sobre sobre los protocolos que garantizan que mediante estas técnicas las transacciones se produzcan de forma correcta. Y por último se hablará sobre los problemas asociados a esta técnica.

1.2- Tipos de Candados

Primero estudiaremos los candados binarios y luego los compartidos y exclusivos.

1.2.1- Candado Binario,

es aquél que sólo tiene dos posibles estados, bloqueado o no. Para lo cual se le asocia a cada elemento X de la base de datos, de una variable que almacene el valor de este. Y este candado tiene asociado que un elemento X tiene su candado en modo bloqueado, ese elemento no puede ser accedido por ninguna razón, es decir, ni para lectura ni escritura.

Por eso una transacción debe implementar la acción bloquear y desbloquear, operaciones que deben ser operaciones indivisibles. Por eso una transacción lo primero que hace es pedir bloquear el elemento, así si el elemento está bloqueado, esta se queda esperando hasta que el elemento cambio de estado. Y cuando la transacción anterior libere el elemento, la que estaba en espera continua bloqueando el elemento.

1.2.2- Candado compartidos y exclusivos,

en este candado se implementa más estados, en este caso se tiene: lectura, que bloque para lectura, escritura , que bloquea para escritura, y desbloquear . Así un elemento puede estar bloqueado para lectura y puede ser accedido por otra transacción para lectura también . Mientras que si está bloqueado para escritura este elemento no será accedido hasta que no se pase al estado de desbloqueo.

1.3- Bloqueo de dos fases

Un protocolo es de dos fases siempre y cuando una transacción siempre a su inicio se realizan todas las operaciones de bloqueo, y luego se realizan las de desbloqueo. Pero nunca mezclando estas dos operaciones de forma simultánea.

Estas se pueden dividir en tres subclases : **básico,** donde se liberan los elementos ya no necesarios tras haber bloqueado a los siguientes necesarios para esta transacción. **Conservador,** donde la transacción tiene que saber claramente que elementos tiene que bloquear antes de iniciar la operación y como debe bloquearlos. Y si un sólo elemento no se pudiera bloquear la transacción no tendría lugar hasta que se pueda bloquear. **Estricto ,** se basa en que un elemento bloqueado no es desbloqueado hasta la transacción termina o es abortada.

1.4- Resolución del bloqueo mortal y de la espera indefinida.

Hay **Bloqueo Mortal** cuando existen dos o más transacciones que se están esperando mutuamente para realizar operaciones sobre un elementos usados por ellas. Por ejemplo:

T1	*T2*
bloquea_lectura (Y);	
leer_elemento (Y);	
	bloquear_lectura (X);
	leer_elemento (X);
bloquear_escritura (X);	
	bloquear_escritura (Y);

Así T1 para poder bloquear para escritura debe esperar a que T2 libere de lectura. Y T2 también espera por T1. Así esos elementos se quedan “congelados” mientras no se liberen.

Para solucionar este problema, se puede usar un **protocolo de prevención del bloqueo.** Un ejemplo de este protocolo es el de dos fases anteriormente descrito pero esto limita más la concurrencia

sobre los elementos. Otra técnica se basa en ordenar los elementos y asegurarse de que una transacción que necesite varios elementos los bloqueará según ese orden.

Otras técnicas propuestas se basan en el uso de marcas de tiempos y la decisión si una transacción provocará un bloqueo o no. Y si esta debe esperar o ser abortada.

También existe otro grupo de técnicas donde la transacción no espera, simplemente si no puede esperar se aborta y tras esperar un cierto tiempo se reinicia la transacción.

Otra forma es la de estableces tiempos de espera máximos donde si una transacción espera más de un cierto tiempo, esta debe abortar.

Otra técnica es que el sistema cada cierto tiempo, compruebe si hay alguna situación de bloqueo, esta técnica es adecuada en sistemas donde se prevea que el bloque no suele aparecer muy a menudo. Una forma de detectar esto puede ser el uso de un **grafo de bloque mortal**, donde se crea un vértice por cada transacción que se está ejecutando, y una arista por cada espera por elemento bloqueado desde quien espera, al que bloquea. Así cuando se halla un ciclo en el grafo, tendremos un bloqueo mortal. Abortando el los "vértices" que incidan sobre él.

Otro problema que se puede presentar puede ser la **Espera indefinida,** que aparece cuando una transacción se queda esperando para bloquear, mientras otras realizan sus operaciones sin problema. Esto puede ser debido a un esquema de espera injusto. La solución más directa es el uso de una técnica justa de tiempos de espera, como pueden ser el primero que llega, o primero que se atiende.

1.5- Control de concurrencia basado en ordenamiento por marca de Tiempo

En este apartado se expone otro mecanismo de control, este se basa en el uso de marcas de tiempos, para ordenar la ejecución de las transacciones, que permita una ejecución serial.

1.6- Marcas de Tiempo

Una marca de tiempo es un identificador único asignado a una transacción. Que se suelen asignar según el orden en que entran las transacciones al sistema, y esta marca se puede considerar como el tiempo de inicio de una transacción. Las técnicas que usan marcas de tiempo, no usan bloqueo , por eso no se corre riesgo de bloqueos mortales.

1.7- El Algoritmo de ordenamiento por marca de tiempo.

Este algoritmo se basa en ordenar en base a las marcas de tiempo. Para lo que cada elemento tendrá dos marcas asociadas. Una **marca lectura del elemento X**, será la más grande de todas las marcas de tiempo de las transacciones que han leído con éxito el elemento X . Y también usará una **marca de tiempo de escritura del elemento X,** será la más grande de todas las marcas de tiempo de las transacciones que han escrito con éxito.

Y el funcionamiento se basa en que una una transacción T, que quiere acceder a un elemento X, primero se mira que la marca de tiempo de T y la marca de X. No violen el plan de ejecución de transacciones. Y en el caso que se viole esto la transacción T, es reintegrada al sistema con una marca de tiempo nueva. Pero implica que cualquier otras transacciones que su marca dependiera de T, deberá ser revertida. Que trae el problema de la reversión en cascada.

1.8- Técnicas de Control de Concurrencia de multiversión.

En este apartado se habla sobre técnicas donde se almacenan versiones de un elemento X, y luego en el momento de acceso a este, se elige alguna de sus versiones para mantener la secuencia de las operaciones.

1.9- Técnica de multiversión basada en ordenamiento por marca de tiempo.

Se basa en que cada versión de un elemento X, almacena la marca de tiempo **lectura,** valor más alto de acceso a este con éxito por una transacción. Y también se almacena una marca de tiempo **escritura,** que es la marca de tiempo de la transacción que escribió el valor de la versión del elemento X.

Así para el manejo del elemento X, se aplican las siguientes reglas:

1. Si la transacción T emite una operación escribir_elemento (X), y la versión i de X tiene la Marca De Tiempo_escritura(X) más alta de todas las versiones de X que también es menor que MT(T) o igual a ella, y MT (T) < MT_lectura (X), entonces se debe abordar y revertir la transacción Ti en caso contrario, se crea una nueva versión de X de X con MT_lectura (X) = MT_escritura(X) = MT(T).
2. Si la transacción T emite una operación leer_elemento(X), se busca la versión i de X que tiene la MT_escritura(X) más alta de todas las versiones de X que también es menor que MT(T) o igual a ella; luego se devuelve el valor de X a la transacción T y se asigna a MT_lectura (X) el más alto de estos dos valores : MT(T) y el MT_lectura(X) actual.

1.10- Bloqueo de dos fases Multiversión.

Se basa en tener tres estados posibles bloqueos para cada elemento X, que serían lectura , escritura y certificación . Y se basa en que un mientras un elemento X , es bloqueado para escritura por una Transacción T, de este modo otra transacción T´ podrá acceder a la versión de X, que está en estado certificación. Y mientras las T´ acceden a esta versión, T puede manipular X sin ningún tipo de problemas, y al finalizar T sus acciones con éxito. Se establece el candado de Certificación para

todos aquellos bloqueados por T.

1.11- Técnica para el control de concurrencia de validación (optimista)

En esta técnica no se no se efectúa ninguna comprobación durante la ejecución de una transacciones, sino que todas las operaciones son realizadas en un "temporal", y tras finalizar la transacción en todas sus operaciones, se realizaran las operaciones.

Para esto el proceso se divide en tres fases:

1. **Fase de lectura,** las lecturas se realizan sobre la bases de datos directamente, pero las de actualización se harán sobre copias locales.
2. **Fase de validación,** se valida si no se viola la secuencia de actualizaciones y entonces se aplican los cambios a la base de datos.
3. **Fase de escritura,** si la validación tubo éxito, se realiza la escritura del temporal a la base de datos. Si no fuera así, se desechan las acciones y se reinicia el proceso.

2- Transacciones

Una **transacción** consiste en una interacción con una estructura de datos que, aún siendo compleja y estar compuesta por varios procesos que se han de aplicar uno después del otro, queremos que sea equivalente a una interacción *atómica*. Es decir, que se realice de una sola vez y que la estructura a medio manipular no sea jamás alcanzable por el resto del sistema.

Las operaciones que pueden incluir una transacción son :

-Leer un elemento.- Un elemento de la base de datos es leído y guardado en una variable de programa.

-Escribir un elemento.- Una variable es se escribe en un elemento de la base de datos.

Las transacciones pueden ser introducidas por distintos usuarios de forma concurrente y podrían leer y actualizar los mismos elementos de una base de datos.

Este hecho puede provocar problemas cuando las transacciones concurrentes se ejecutan de manera no controlada. Los tipos de problemas causados pueden ser:

- La actualización perdida.- Esto ocurre cuando dos transacciones que tienen acceso a los mismos elementos de la base de datos tienen sus operaciones intercaladas de modo que hacen incorrecto el valor de algún elemento.
- La actualización temporal o lectura sucia.- Esto ocurre cuando una transacción actualiza un elemento de la base de datos y luego la transacción falla por alguna razón.
- El resumen incorrecto.- Si una transacción está calculando una función agregada de residen sobre varios registros mientras otras transacciones están actualizando algunos de ellos, puede ser que la función agregada calcule algunos valores antes de que se actualicen y otros después de actualizarse.
- Otro problema que podría presentarse es el de la lectura no repetible, en el que una transacción T lee un elemento dos veces y otra transacción T' modifica el elemento entre las dos lecturas.

En un sistema es necesario asegurarse de que siempre que se introduce una transacción:

1) Todas las operaciones de la transacción se completen con éxito y su efecto quede asentado en la base de datos.
2) O bien que la transacción no tenga efecto alguno sobre la base de datos ni sobre cualquier otra transacción.

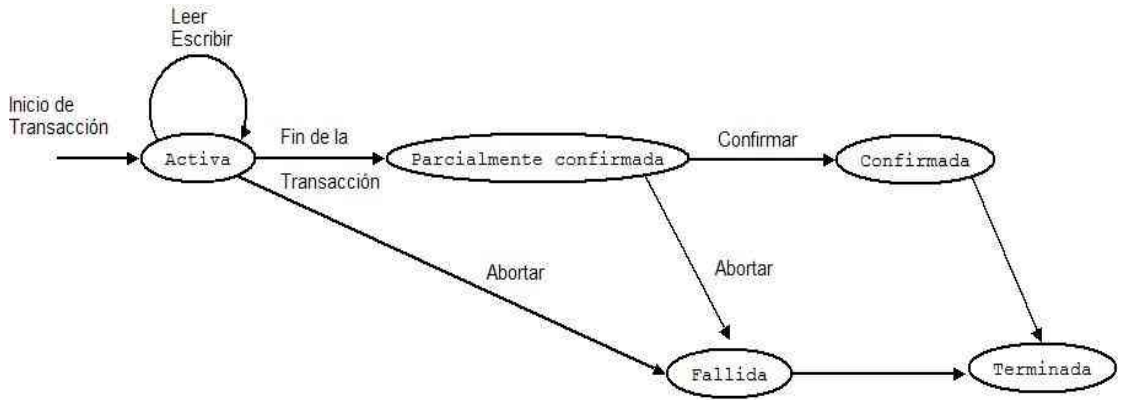
Los tipos de fallo de transacción que pueden darse, ordenados del más posible a menos, son los siguientes:

1. **Fallo del computador.** Un fallo de Hardware o software que ocurre en el sistema de cómputo durante la ejecución de una transacción.
2. **Error de la transacción o sistema.** Alguna operación de una transacción hace que ésta falle.
3. **Errores locales o condiciones de excepción detectadas por la transacción.** Durante la ejecución de transacciones pueden presentarse ciertas condiciones que requieran la cancelación de la transacción.
4. **Imposición del control de concurrencia.** El método de control de concurrencia puede decidir que se aborte una transacción.

5. **Fallo del disco.** Que bloques del disco pierdan su información por una mal funcionamiento.
6. **Problemas y catástrofes físicas.** Esto contiene una incontable lista de problemas que incluyen desde fallos en la lineas de energias, explosiones nucleares y meteoritos que caen a la tierra.

Los problemas del 1 al 4 son los más frecuentes y siempre que ocurren el sistema debe mantener suficiente información para poder recuperarse del fallo. Mientras que los fallos 5 y 6 la recuperación puede ser bastante complicada.

Una transacción es una unidad atómica de trabajo que se realiza por completo o bien no se efectúa en absoluto. Pare poder hacer recuperación el sistema se mantiene al tanto de cuando se inicia la transacción, cuando se acaba y si se confirma o se aborta.



Para que el sistema se pueda recuperar de los fallos de transacciones, el sistema necesita mantener un registro llamado bitácora o también diario, que sigue las pista a todas las operaciones de transacciones que afectan a los valores de la base de datos.
Se llama punto de confirmación cuando todas las operaciones que tienen acseso a la base de datos de una transacción T, se han ejecutado con éxito y el efecto de todas estas operaciones se han asentado en la bitácora.(Más allá del punto de confirmación, se dice que la transacción está confirmada).

Las propiedades que se desean que tenga una transacción son ACID:

1. Atomocidad.- Una transacción es una unidad atómica de procesamientos.
2. Conservación de la consistencia.- Una ejecución correcta de la transacción debe llevar a la base de datos de un estado consistente a otro.
3. Aislamiento.- Una transacción no debe dejar que otras transacciones puedan ver sus actualizaciones antes de que sea confirmada.
4. Durabilidad o permanencia.- Una vez que una transacción ha modificado la base de datos y las modificaciones se han confirmado, éstas nunca deben perderse por un fallo subsecuente.

Cuando varias transacciones se ejecutan concurrente mente de forma intercalada, el orden en el cual se ejecutan sus operaciones es lo que se llama **plan o historia** de las transacciones. Un plan de P de n transacciones $T_1,T_2,...T_n$ es un ordenamiento para las operaciones de las transacciones sujeto a la restricción de que, para cada transacción T_i que participe en P, las operaciones de T_i en P devén aparecer en el mismo orden en que ocurren en T_i. Se dice que un plan P de n transacciones es completo si cumple:

1. Las operaciones de P son exactamente las operaciones de $T_1,T_2,...T_n$ incluidas una operación de confirma o abortar como ultima operación de cada transacción en el plan.
2. Para cualquier par de operaciones de la misma transacción T_i su orden de aparición en P es el mimos que en T_i.
3. Para cualquier dos operaciones en conflicto, una de ellas debe ocurrir antes que la otra en el plan.

Se dice que dos operaciones están en conflicto si perteneciendo a distintas transacciones tienen el acceso a un elemento, llamado X, y una de las dos operaciones es escribir elemento.
Un plan P se llama recuperable cuando ninguna transacción de T de P se confirma antes de que se halla confirmado todas las transacciones T' que hallan escrito algún elemento que lea T. Un fenómeno que puede darse es la reversión en cascada, en el que una transacción no confirmada tiene que revertirse por que leyó un elemento de una transacción fallida. Un plan P evita la reversión en cascada cuando toda transacción del plan sólo lee elementos escritos por transacciones confirmadas. Existe además un plan más restrictivo llamado estricto en el que las transacciones no

pueden leer ni escribir datos de un elemento hasta que la ultima transacción que halla escrito ese elemento no se halla confirmado o abortado.

Para determinar que planes son correcto y cuales no, se usa la llamada teoría de seriabilidad, para poder desarrollar técnicas que sólo permitan planes correctos.

P es un plan en serie si por cada transacción T que participa en el plan, todas las operaciones de T se ejecutan consecutivamente en el plan, de lo contrario P no es plan en serie. Si un plan en serie de n transacciones es equivalente a un plan P con esas mismas n transacciones, se dice que P es seriable. Los planes que no son en serie se pueden dividir en dos grupos lo que son equivalente a otro plan en serie (seriables) y lo que no (no seriables). Si un plan P es equivalente a un plan en serie correcto, es considera que el plan P es correcto.

Para poder ver que dos planes son equivalentes existen varias definiciones:

1. Se dice que dos planes son equivalentes por resultados si producen el mismo estado final en la base de datos.
2. Para que dos planes sean equivalentes, las operaciones que se aplican a cada uno de los elementos de información afectados por los planes se deben aplicar a ese elemento en el mismo orden en ambos planes.
3. Se dice que dos planes son equivalentes por conflictos si el orden de cualquier dos operaciones en conflicto es el mismo en ambos planes.

3- Técnicas de Recuperación

En esta sección se estudiará la forma algunos puntos importantes sobre la Recuperación, dividiéndolo en una primera parte donde se trata conceptos de la recuperación, luego se hablará sobre algunas de la técnicas existentes para la recuperación de errores . Y finalmente se estudiará como actuar ante un error catastrófico.

3.1- Conceptos

3.1.1- Bosquejo de la Recuperación

Normalmente cuando aparece un fallo en una base de datos, se suele recuperar la base de datos, restaurándola a algún estado anterior, que suele estar almacenado en un medio secundario. Lo más

cerca la momento del fallo. Para realizar algo así, se deberá tener un registro en el cual se almacena todas las modificaciones realizadas por la ejecución de transacciones . Y esto se suele realizar en una **bitácora del sistema.**

Las estrategias de acción en la recuperación se pueden resumir en los siguiente pasos:

1. Si se produce un fallo extenso en la base de datos, se restaurará el sistema con una copia anterior almacenada en un medio secundario. Y tras esto apoyándose en los registros de transacciones de la Bitácora, se termina la restauración, en un punto lo más cercano posible al momento del fallo
2. Si el fallo no es físico, sino problemas de inconsistencia en los datos. Se desaceran todas aquellas transacciones hasta llegar a un punto, en el cual aparezca de nuevo la consistencia en el sistema

En cuanto a técnicas de recuperación, se pueden dividir en dos grandes grupos:

Actualización diferida, que se basa en que las transacciones no se realizaran directamente sobre la base de datos. Sino primero se realiza en la bitácora y luego se se confirma la transacción, entonces se traslada a la base de datos.

Actualización Inmediata, se basa en que las transacciones tienen lugar directamente sobre la base de datos, pero forzando que los pasos dados se anoten en la bitácora antes. Pero todo esto sin confirmar la transacción.

3.2- Conceptos de sistema para la recuperación

En esta sección se expondrá las características de los sistemas y su relación con la recuperación de errores.

El manejo de las transacciones no se realizan normalmente directamente sobre el elemento

almacenado en el medio primario de almacenamiento. Sino se realiza siempre en un medio intermedio como puede ser la memoria principal. Para lo que se suele usar un caché, el cual en muchas ocasiones es manejado por el Sistema Gestor de la Base de Datos. Y en el uso de este cache se encuentra algunas de las técnicas de recuperación.

Se suele tener un buffer donde en cada ocasión en la cual se necesita acceder a un elemento se observa si este está o no el buffer. Debido a que si este no se encuentra se deberá copiar al buffer, o incluso eliminarlo para tener espacio para tener más espacio. Pero antes de eliminar este elemento se observará si el **bit de modificación,** nos indica si ese elemento se deberá o no copiar a elemento primario.

En el control de este bit se distinguen dos estrategias. La primera, **actualización en el lugar**, escribe el elemento en la misma posición en la que estaba el elemento, en el almacenamiento primario, sobre escribiendo el valor anterior del elemento. La segunda se denomina, **creación de sombras,** se besa en escribir en otra posición este elemento, de esta manera se mantiene el valor anterior y el valor actual. Debido a esto no es necesario tener una bitácora del sistema.

3.3- Reversión de transacciones

Si una transacción falla después de actualizar la base de datos, puede ser necesario revertir la transacción. Y es en este momento cuando apoyándose en la bitácora se inicia la reversión, pero esto puede traer aparejado la reversión de todas las transacciones que halla manejado cualquier elemento, que fuera actualizado por la primera en revertir. Y esto puede hacer que entremos en un estado de **reversión en cascada.** Y como este fenómeno tarda mucho los sistemas con diseñados con la intención de evitar este efecto, a no ser que sea totalmente necesario.

3.4- Técnicas de recuperación basadas en actualización diferida.

Son el conjunto de técnicas donde la actualización en la base de datos se realizará, sólo cuando se halla confirmado con éxito la transacción. De esta manera si una transacción no llega a su punto de confirmación , no hará falta la realización de ninguna acción, debido a que la transacción no ha afectado a la base de datos.

En estos sistemas se conocen como **NO REHACER,** debido a que no rehacen ninguna operación. Y sólo sería necesario REHACER si el error se produce después de la confirmación de la transacción. Y estas se rehacen a partir de las entradas de la bitácora.

3.5- Recuperación por actualización diferida

3.5.1- En Sistemas Mono usuario,

se basa en realizar la operación de REHACER de una forma **idempotente,** que es que ejecutarla una vez u otra vez, no cambiará nada. Debido a que si hubieran diferencias esto podría repercutir en que si se produce un error durante la operación, se podría a realizar operaciones que debían ser revertidas.

Y se basa en tener una lista con las transacciones confirmadas desde el último punto de control y las transacciones activas. Y a partir de la bitácora REHACER todas las operaciones de escritura que estén en estado activo, en el orden indicado por la bitácora.

3.5.2- En sistemas Multi Usuario,

en un sistema así es importante también el sistema de control de concurrencia. Para poder garantizar la seriabilidad de las transacciones. Pero básicamente el algoritmo de recuperación es igual al del sistema mono usuario.

3.6- Técnicas de recuperación basadas en actualización inmediata

Se basan en entornos donde los cambios a la bases de datos se realizaran directamente, es decir, antes de que se confirme la transacción . Pero se debería tener sistemas para deshacer los efectos de las operaciones de actualización sobre la base de datos, debido a que una transacción puede fallar tras realizar una actualización en la base de datos. Y se distinguen dos categorías : en las que se asegura que se asiente en la base de datos antes de la confirmación de la transacción. Y nunca se deberá realizar operaciones de REHACER, y estos algoritmos se denominan **DESHACER/NO REHACER**. Y si se permite la confirmación de la transacción antes de la actualización estamos ante los algoritmo más complicados y se llaman **DESHACER/REHACER.**

3.7- Paginación de Sombra

Se basa en tener considerar que la base de datos está compuesta por páginas, o bloques, de tamaño fijo para fines de recuperación. Y luego se mantienen listas de páginas, donde en cada transacción se crea una nueva lista, sin eliminar la anterior. Y haciendo que los nuevos datos se actualicen en nuevas ubicaciones. así cuando se produce un error lo único que se hace es poner como tabla actual, la más próxima.

3.8- Respaldo de bases de datos y recuperación de fallos catastróficos.

Aquí se trata los fallos que se pueden ocasionar debido a fallos en los medios físicos de almacenamiento, y en este caso se usa el **Respaldo,** que no es más que el uso de algún medio externo de almacenamiento, en el cual se copiará de forma periódica la base de datos y su bitácora.

De tal modo si se produce un error grabe, se deberá restaurar la base de datos con su última copia, y dejarla lo más actualizada mediante la copia de bitácora más cercana.