Índice

Bibliografía

- Fundamentos de la base de datos Sistemas . 6to – Edición
  R. Elmasri y SB Navathe . Addison Wesley, 2010
  Capítulo 23
- Base de Datos : Principios , Programación y Rendimiento, 2ª Edición
  P. O'Neil y E. O'Neil . Morgan Kaufmann, 2000
  Capítulo 10
- Conceptos básicos de bases de datos. 4ta edición
  A. Silberschatz , HF Korth y S. Sudarshan . McGraw-Hill, 2002
  Capítulo 17
- Oracle 10g: Manual del gestor
  K. Solitario . McGraw-Hill, 2005
  Capítulo 7

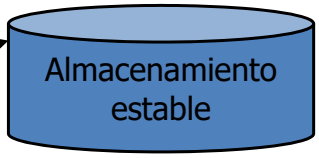
3.2.1 ¿Por qué son necesarias técnicas de recuperación de caídas?

- Un DBMS no está libre de errores ni de situaciones indeseables:
  1. Errores de aplicación : división por cero, excepción no controlada...
  2. Errores del sistema: el sistema operativo se atasca
  3. Excepciones generadas por una transición (violación de la regla de integridad)
  4. Situaciones de interbloqueo en transacciones
  5. Fallo del sistema: fallo en la fuente de alimentación
  6. Rotura de disco
  7. Fallo de red
  8. Error de usuario: eliminación accidental de una tabla...

- Garantizar:
  – Los mecanismos de recuperación garantizan que, después de un fallo, la base de datos volverá a su último estado consistente.

3.2.1 ¿Por qué son necesarias técnicas de recuperación de caídas?

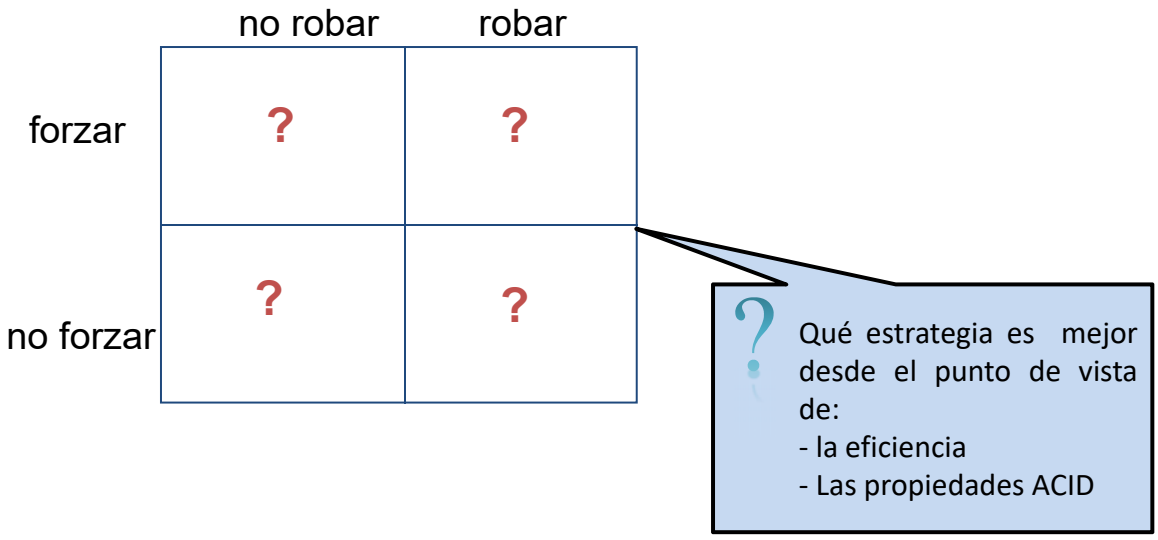
- Según el tipo de fallo:



  - ✓ Archivos de respaldo
    - ✓ lógico (exportaciones)
    - ✓ físico:
      - ✓ completo o incremental
      - ✓ Fuera de línea o en línea
      - ✓ DBMS o sistema operativo
  - ✓ Volver a registrar archivos
    - ✓ mantiene un registro de acciones (actualizaciones) pero que, quizás, ni siquiera forman parte de una copia de seguridad
    - ✓ Una vez configurado, es prácticamente transparente para el usuario.

3.2.2 Gestión del buffer de datos

- **Páginas sucias** : páginas modificadas que aún no se han escrito en el disco.
- ¿Cuándo se debe transferir una página sucia al disco? (por ejemplo, mover un bloque de datos de la memoria al espacio de tabla correspondiente)
  - **estrategia robar :** no se puede escribir una página sucia antes de que se confirme la transacción que modificó dicha página.
  - **estrategia robar** : una página sucia se puede escribir en el disco y eliminar de la memoria cuando sea necesario sin tener en cuenta ningún estado de transacción.
  - **estragia forzar** : cualquier página sucia se escribe en el disco cuando se confirma la transacción que la modificó .
  - **estrategia no forzar** : no es obligatorio forzar la escritura cuando se confirma una transacción determinada.

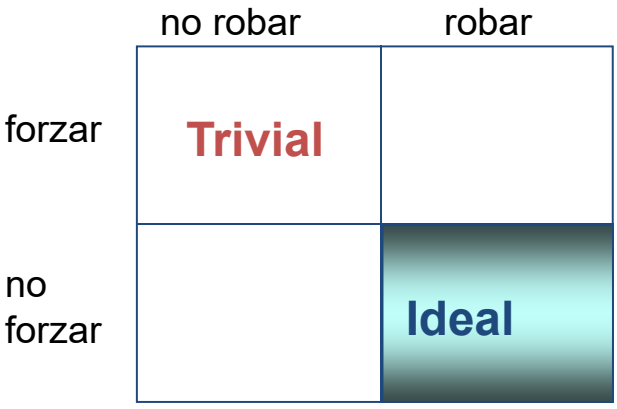


3.2.2 Gestión del buffer de datos

Tema 3.2: Técnicas de recuperación de caídas

- ¿Forzar a escribir a disco páginas sucias al cometer la transacción?:
  - Garantiza durabilidad
  - Daña el rendimiento

- ¿Robar páginas sucias antes de cometer la transacción?
  - ¿Cómo garantizamos la atomicidad?
  - Aumenta el rendimiento



3.2.3 La bitácora

Bitácora (redolog) :
Archivo que mantiene un registro de los datos históricos de cada modificación de datos (traza).

La estructura básica de un archivo bitácora es la siguiente:

`<Ti, COMENZAR>:` Se graba antes de que se empiece a ejecutar la transacción i. Sólo existirá un registro de este tipo por cada transacción..

`<Ti, X, Vo , Vn >:` Vo es el anterior valor para un determinado bloque X de datos y Vn es el nuevo valor . Existirán tantos registros de este tipo como operaciones de `ESCRITURA` en el transacción .

`<Ti, COMMIT>:` Se registra cuando el La transacción pasa al estado de parcialmente cometida. Sólo se escribirá un registro de este tipo para cada transacción .

3.2.3 El bitácora

Registro de escritura anticipada, WAL
Protocolo de escritura anticipada de trazas:

1. Cualquier traza se anota en la bitácora antes de escribir la página sucia correspondiente en el disco.
2. Todos los trazas de una transacción deben escribirse en la bitácora antes de que se confirme la transacción.

#1 garantiza la **atomicidad** .
#2 garantiza la **durabilidad** .

Los algoritmos de recuperación de ARIES se basan en este protocolo.

3.2.3 El bitácora

Registro de escritura anticipada, WAL

Protocolo de escritura anticipada de trazas:

1. Cualquier traza se anota en la bitácora antes de que la página sucia correspondiente pase al disco.
2. Todos los trazas de una transacción deben anotarse en la bitácora antes de que se confirme la transacción.

¿por qué? ¿qué relación mantiene con la estrategia robar ?

#1 garantiza la **atomicidad**
#2 garantiza la **durabilidad**

Los algoritmos de recuperación ARIES se basan en este protocolo.

¿por qué? qué relación mantiene con la estrategia no forzar ?

3.2.4 Algoritmos de recuperación basado en bitácora

- Operaciones de recuperación :
  - `DESHACER (Ti)` :
    - Asigna los valores antiguos, Va, a todos los datos modificados por `Ti`
    - – Necesario para garantizar la atomicidad
  - `REHACER (Ti)` :
    - Asigna los valores nuevos, `Vn`, a todos los datos modificados por `Ti`
    - Necesario para garantizar la durabilidad
- **Actualización inmediata**: Las operaciones de `ESCRITURA` se pueden realizar inmediatamente cuando ha sido grabado `<Ti, X, Vo , Vn >` en la bitácora, no es obligatorio a esperar para el transacción comprometerse .
- **Actualización diferida**: Las operaciones `ESCRIBIR` se realizan cuando ha sido grabado `<Tu, COMMIT>` en la bitácora.

3.2.4 Algoritmos de recuperación basado en bitácora

- Operaciones de recuperación :
  - `DESHACER (Ti)` :
    - Asigna los valores antiguos, Va, a todos los datos modificados por `Ti`
    - – Necesario para garantizar la atomicidad
  - `REHACER (Ti)` :
    - Asigna los valores nuevos, `Vn`, a todos los datos modificados por `Ti`
    - Necesario para garantizar la durabilidad
- **Actualización inmediata**: Las operaciones de `ESCRITURA` se pueden realizar inmediatamente cuando ha sido grabado `<Ti, X, Vo , Vn >` en la bitácora, no es obligatorio a esperar para el transacción comprometerse .
- **Actualización diferida**: Las operaciones `ESCRIBIR` se realizan cuando ha sido grabado `<Tu, COMMIT>` en la bitácora.

¿qué relación guarda DESHACER con el protocolo WAL? ¿y REHACER?

3.2.4 Algoritmos de recuperación basado en bitácora

Tema 3.2: Técnicas de recuperación de caídas

- Operaciones de recuperación :
  - `DESHACER (Ti)` :
    - Asigna los valores antiguos, Va, a todos los datos modificados por `Ti`
    - – Necesario para garantizar la atomicidad
  - `REHACER (Ti)` :
    - Asigna los valores nuevos, `Vn`, a todos los datos modificados por `Ti`
    - Necesario para garantizar la
- **Actualización inmediata**: Las operaci ueden realizar inmediatamente cuando ha sid en la bitácora, no es obligatorio a esperar para el transacción comprometerse .
- **Actualización diferida**: Las operaciones `ESCRIBIR` se realizan cuando ha sido grabado `<Tu, COMMIT>` en la bitácora.

¿estrategia robar y actualización inmediata ?

Computer Science Grade Database Management and Administration

3.2.4 Algoritmos de recuperación basado en bitácora: *undo/redo*

- • Algoritmo de recuperación para bitácora incremental con actualizaciones inmediatas *(deshacer /rehacer)* :

```
SI se ha producido un fallo ENTONCES
    Revisar secuencialmente la bitácora
    SI está <Ti,INICIO> y está <Ti, COMETIDA>
            ENTONCES REHACER(Ti)
    SI está <Ti,INICIO> y NO está <Ti, COMETIDA>
            ENTONCES DESHACER i
```

- Se ajusta a estrategias robar/no forzar.
- Optimiza la normal ejecución de las transacciones.
- Penaliza la eficiencia en caso de recuperación..

3.2.4 Algoritmos de recuperación basado en bitácora: *undo/redo*

- Ejemplo :

```
T0:     LEER (A, a1)
        a1 := a1 - 10000
        ESCRIBIR(A,a1 )
        LEER(B,b1 )
        b1 := b1 + 10000
        ESCRIBIR(B,b1 )

T1:     LEER (C, c1)
        c1 := c1 - 20000
        ESCRIBIR(C,c1 )
```

Se produce un fallo y nos encontramos estas posibles bitácoras:

```
<T0, EMPIEZO>
<T0 , A, 100000, 90000>
<T0 , B, 20000, 30000>
```

Se ejecutaría UNDO (T0)

```
<T0, EMPIEZO>
<T0 , A, 100000, 90000>
<T0 , B, 20000, 30000>
<T0, COMPROMETER>
<T1, EMPIEZO>
<T1 , C, 50000, 30000>
```

Se ejecutaria REHACER (T0) y DESHACER (T1)

3.2.4 Algoritmos de recuperación basado en bitácora: no *undo/redo*

- Algoritmo de recuperación para bitácoras incrementales con actualizaciones diferidas:

```
SI ha ocurrido un fallo ENTONCES
  Verifique secuencialmente la bitácora
  SI está < Ti , INICIO> y está < Ti , COMETIDA> ENTONCES
   REHACER ( Ti )
```

  - No es necesario registrar en el seguimiento de actualización el valor anterior `< Ti , X, Vn >`
  - Se adapta a estrategias no robar.
  - Es muy adecuado para transacciones cortas en las que hay algunas actualizaciones.
  - Optimiza la eficiencia en caso de abortar una transacción.

3.2.4 Algoritmos de recuperación basado en bitácora: no deshacer / *rehacer*

- Ejemplo :

```
T0:     LEER (A, a1)
        a1 := a1 - 10000
        ESCRIBIR(A,a1)
        LEER(B,b1)
        b1 := b1 + 10000
        ESCRIBIR(B,b1)

T1:     LEER (C, c1)
        c1 := c1 - 20000
        ESCRIBIR(C,c1)
```

Se produce un fallo y nos encontramos estas posibles bitácoras:

```
<T0, INICIO>
<T 0 , A, 90000>
<T0 , B, 30000>
```

No se hace nada
(no se ha realizado ninguna operación)

```
<T0, INICIO>
<T 0 , A, 90000>
<T0 , B, 30000>
<T0, COMMIT>
<T1, INICIO>
<T1 , C, 30000>
```

En este momento se hizo:
`ESCRIBIR(A,a1)`
`ESCRIBIR(B,b1)`

Se ejecutaría REHACER(T0)

3.2.4 Algoritmos de recuperación basado en bitácora

- Gestión de la búfer de datos:
  - un robar estrategia marcas difícil el atomicidad → Operador
  - una no forzar estrategia marcas difícil el durabilidad → Operador

	forzar escritura	**no forzar escritura**
robar pagina (actualización Inmediata)	*no redo-undo*	*redo-undo*
No robar página (actualización diferida)	*no redo- no undo*	*redo - no redo*

- Gestión del buffer de bitácora : protocolo WAL :
  - La regla WAL#1 permite implementar el operador DESHACER (undo).
  - La regla WAL#2 permite implementar el operador REHACER (redo).

3.2.5 puntos de verificación

- puntos de verificación *(checkpoints)*:
  - Revisar toda la bitácora tras un fallo es muy lento
  - Se rehacen muchas transacciones que estaban verdaderamente cometidas en la base de datos.
- ¿Cómo se crean los puntos de verificación ?
  - Periódicamente se hace la siguiente secuencia de operaciones:
    1.- Grabar al archivo de datos todas las páginas sucias que se encuentran en el buffer de datos
    2.- Grabar a disco en la bitácora el registro <PUNTO_VERIFICACIÓN> (*checkpoint*)
- Algoritmo de recuperación con puntos de verificación:
  - Se mantiene una lista con las transacciones no cometidas
  - Si se ha producido un fallo entonces
    - Revisar desde el final hacia atrás la bitácora hasta encontrar <PUNTO_VERIFICACIÓN>
    - REHACER todas las transacciones cometidas tras <PUNTO_VERIFICACIÓN>
    - Continuar hacia atrás hasta DESHACER todas las transacciones no cometidas

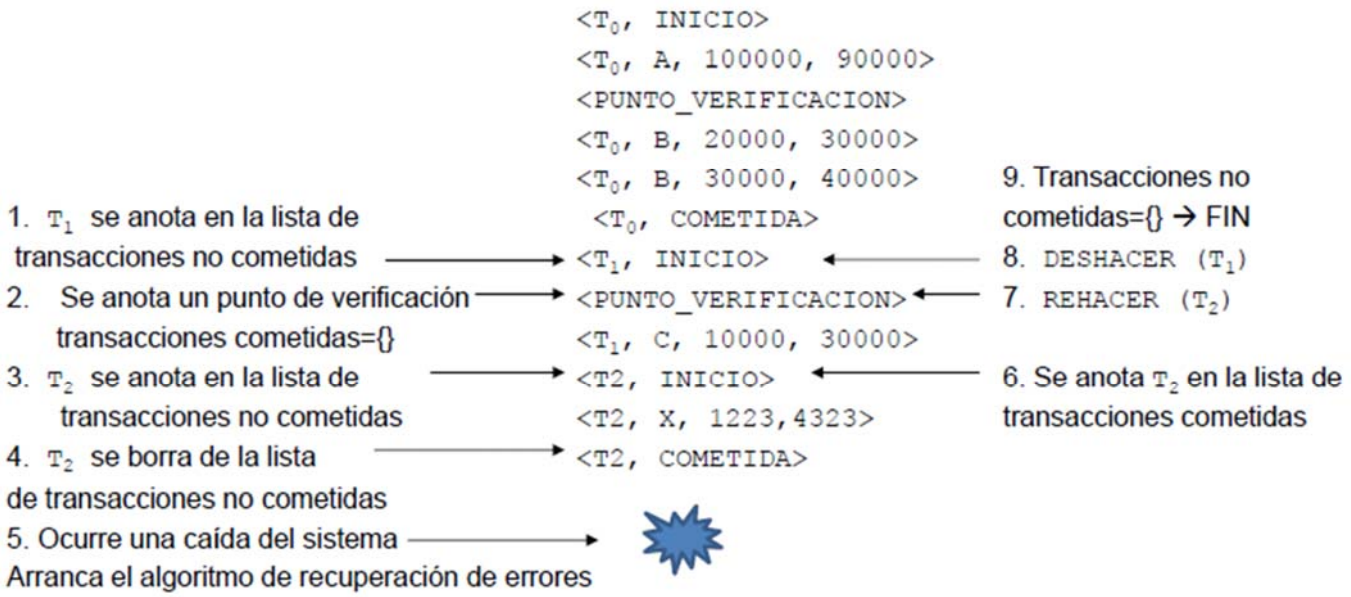
3.2.5 puntos de verificación

- puntos de verificación *(checkpoints)*:
  - Revisar toda la bitácora tras un fallo es muy lento
  - Se rehacen muchas transacciones que estaban verdaderamente cometidas en la base de datos.
- ¿Cómo se crean los puntos de verificación ?
  - Periódicamente se hace la siguiente secuencia de operac
    1.- Grabar al archivo de datos todas las páginas sucias que s os
    2.- Grabar a disco en la bitácora el registro <PUNTO_VERIFIC N> (*checkpoint*)

¿tiene sentido mantener puntos de verificación en estrategias no robar – forzar escritura?

- Algoritmo de recuperación con puntos de verificación:
  - Se mantiene una lista con las transacciones no cometidas
  - Si se ha producido un fallo entonces
    - Revisar desde el final hacia atrás la bitácora hasta encontrar <PUNTO_VERIFICACIÓN>
    - REHACER todas las transacciones cometidas tras <PUNTO_VERIFICACIÓN>
    - Continuar hacia atrás hasta DESHACER todas las transacciones no cometidas

3.2.5 puntos de verificación



3.2.6 Algoritmo de recuperación ARIES

- Algoritmo real que se utiliza en varios DBMS comerciales (DB2, SQL Server, Postgress)
- La gestión del buffer de datos sigue una estrategia de robar/no forzar.
- Garantiza la atomicidad y la durabilidad siguiendo el protocolo WAL.
- El proceso de recuperación está anotado para evitar repetirlo en caso de fallo durante el proceso de recuperación.
- Ventajas de ARIES frente a un *deshacer/rehacer básico* algoritmo con puntos de verificación :
  - En cuanto a la eficiencia:
    - Los puntos de verificación no obligan a guardar las páginas sucias en el archivo de datos.
    - No es necesario ejecutar una búsqueda secuencial en la bitácora hasta localizar el inicio de la transacción para deshacer una transacción.
  - En cuanto a las propiedades del ACID :
    - Anticipa errores en cascada: errores durante el proceso de recuperación
    - Anticipa errores durante la inserción de un punto de control.

3.2.6 Algoritmo de recuperación ARIES

Tema 3.2: Técnicas de recuperación de caídas

- Mantiene tres estructuras:
  – Trazas (una especie de archivo bitácora)

LSN	anterior LSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF

  – tabla de transacciones

TRAN_ID	últimoLSN	Tipo

  – Tabla de páginas sucias

PAG_ID	páginaLSN

- Un punto de control ARIES consta de las siguientes tareas:
  – Anote el punto de control en la bitácora.
  – almacena la tabla de transacciones activas y la tabla de páginas sucias.
  – Guarde el LSN del punto de control en un lugar seguro.
  – No es obligatorio copiar las páginas sucias en el archivo de datos. Cada DBMS implementa una estrategia de reemplazo. Por ejemplo, podría guardar en el disco la página sucia más antigua.

3.2.6 Algoritmo de recuperación ARIES

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	C	....
2	0	T2	Act	B	....
3	1	T1	Conf		....
4	control_ini				
5	fin_control				InfoTables
6	0	T3	Act	A	....
7	2	T2	Act	C	....
8	7	T2	Conf		....

transacciones tabla :

TRAN_ID	últimoLSN	Tipo
T1	3	Conf
T2	2	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	1
B	2

3.2.6 Algoritmo de recuperación ARIES

Tema 3.2: Técnicas de recuperación de caídas

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	C	....
2	0	T2	Act	B	....
3	1	T1	Conf		....
4	control_ini				
5	fin_control				InfoTables
6	0	T3	Act	A	....
7	2	T2	Act	C	....
8	7	T2	Conf		....

Tabla de transacciones activas :

TRAN_ID	últimoLSN	Tipo
T1	3	Conf
T2	2	Act

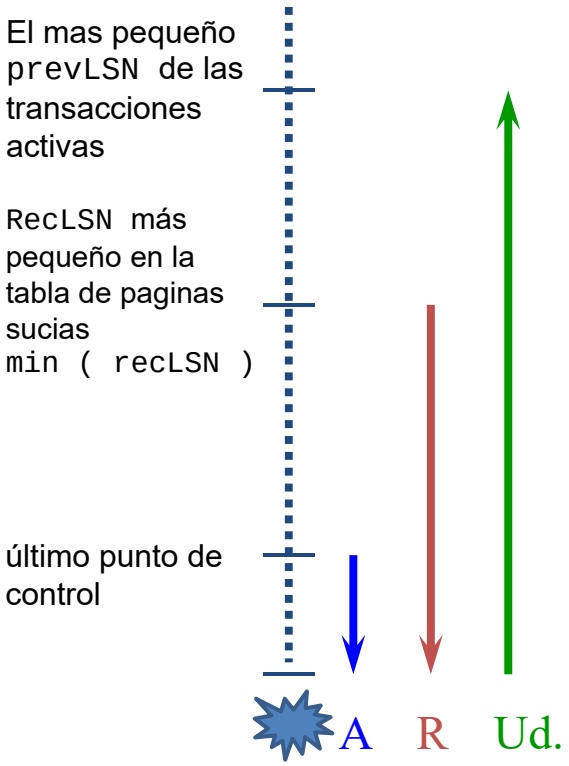
Tabla de páginas sucias:

PAG_ID	recLSN
C	1
B	2

Último LSN de cada transacción activa

LSN del primer evento para que la página se convierta a sucia

3.2.6 Algoritmo de recuperación ARIES



1. Fase de análisis (A)

TRAN_ID	anterior LSN	TIPO
T1	3	Conf
T2	8	Conf
T3	6	Act

PAG_ID	recLSN
C	1
B	2
A	6

~~mín(recLSN)=1~~

`min(recLSN)=1`

2. Fase rehacer (R)

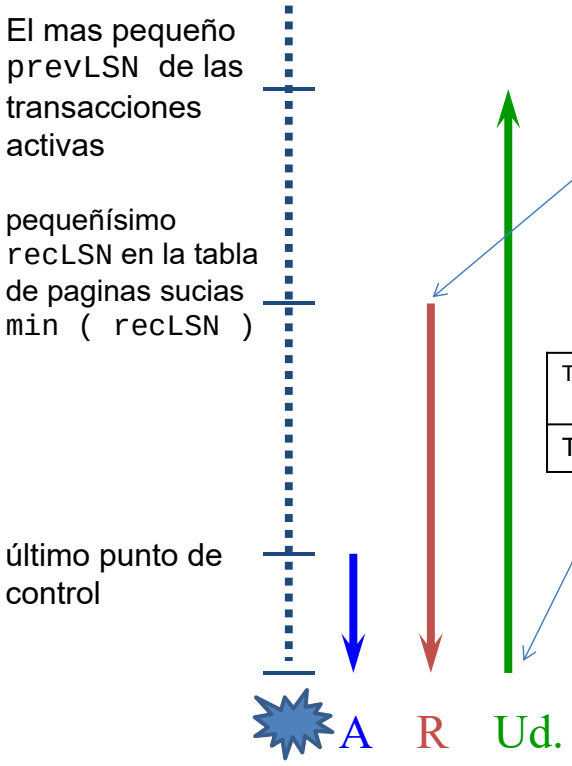
- Se rehace cada transacción posterior a `min(recLSN)`

3. Fase deshacer (U)

- Se deshace todas las transacciones activas hasta llegar al menor `prevLSN` (todas las transacciones están deshechas)

3.2.6 Algoritmo de recuperación ARIES

Tema 3.2: Técnicas de recuperación de caídas



PAG_ID	recLSN
C	1
B	2
A	6

→ no rehacer acciones si :
- la página es no en la tabla
- está pero `recLSN > LSN`

`min(recLSN)=1` → no rehacer acciones dónde `LSN<min(recLSN)`

TRAN_ID	últimoLSN	TIPO
T3	6	Act

→ las transacciones activas se deshacen por medio de `lastLSN`

cada acción *undo* es anotada en la bitácora en un tipo de registro llamado CDR (*Compensation Log Register*)

3.2.6 Algoritmo de recuperación ARIES: ejemplo

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....

Tabla de transacciones:

TRAN_ID	últimoLSN	Tipo
T1	3	Act
T2	2	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	2
B	3

3.2.6 Algoritmo de recuperación ARIES: ejemplo

Tema 3.2: Técnicas de recuperación de caídas

1. Análisis

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....

Tabla de transacciones:

TRAN_ID	últimoLSN	Tipo
T1	6	Conf
T2	9	Act
T3	7	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	2
B	3
mi	7

Mín(recLSN)=2

Computer Science Grade Database Management and Administration

3.2.6 Algoritmo de recuperación ARIES: ejemplo

2.Rehacer

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....

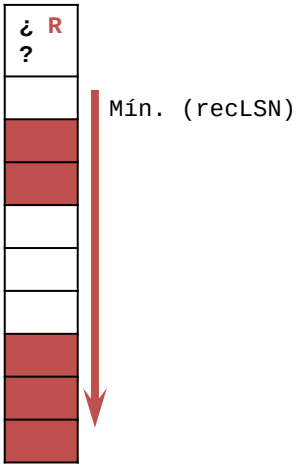


Tabla de transacciones

TRAN_ID	últimoLSN	Tipo
T1	6	Conf
T2	9	Act
T3	7	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	2
B	3
mi	7

Mín(recLSN)=2

3.2.6 Algoritmo de recuperación ARIES: ejemplo

Tema 3.2: Técnicas de recuperación de caídas

2.Deshacer

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....

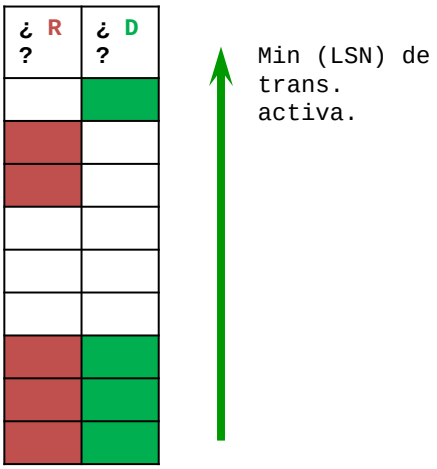


Tabla de transacciones :

TRAN_ID	últimoLSN	Tipo
T1	6	Conf
T2	9	Act
T3	7	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	2
B	3
mi	7

Mín(recLSN)=2

2. Deshacer (T2, paso a paso)

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....
10	9	T2	CLR		8

¿ R ?	¿ D ?

Min (LSN) de trans. activa.

3.2.6 Algoritmo de recuperación ARIES: ejemplo

Tema 3.2: Técnicas de recuperación de caídas

2. Deshacer (T2, paso a paso)

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	ini_control				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....
10	9	T2	CLR		8
11	10	T2	CLR		1

¿ R ?	¿ D ?

Min (LSN) de trans. activa.

Computer Science Grade

Database Management and Administration

3.2.6 Algoritmo de recuperación ARIES: ejemplo

2. Deshacer (T2, paso a paso)

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T2	Act	D	....
2	0	T1	Act	C	....
3	2	T1	Act	B	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T3	Act	mi	....
8	1	T2	Act	C	....
9	8	T2	Act	C	....
10	9	T2	CLR		8
11	10	T2	CLR		1
12	11	T2	Conf		

¿ R ?	¿ D ?

Min (LSN) de trans. activa.

3.2.7 Recuperación de Caídas: Oracle

- Transacción :
  - Conjunto de sentencias SQL
  - Termina cuando :
    - La transacción ejecuta un COMMIT/ROLLBACK
    - Sentencia DDL es ejecutada
    - Usuario es desconectado correctamente : el actual transacción se confirma
    - Se cambia el nivel de aislamiento de la transacción (°)
    - un anormal terminación tiene lugar: el transacción se deshace

- Estructuras físicas para el recuperación
  - Segmento ROLLBACK
    - Segmento especial que almacena los valores antiguos de los datos modificados por la transacción en curso
    - Cuando la transacción se comete se elimina el rollback segment asociado
    - Lo gestiona el proceso DBWR
    - Es usado también para la consistencia en lectura durante las transacciones
  - Bitácora (*redo log*)
    - Guarda todos los cambios (valores nuevos) producidos en cualquier buffer del DBMS (datos, rollback segments, ...)
    - Lo gestiona el proceso LGWR

3.2.7 Recuperación de Caídas: Oracle

- Cuando una transacción entra en estado activo:
  1. Se crea un segmento de *rollback*
  2. El proceso `LGWR` escribe nuevas entradas en la bitácora
  3. La base de datos es modificada por las operaciones de transacción.

- Confirmar una transacción:
  - Las entradas de bitácora de la transacción se guardan en el disco.
  - "Registra" en la tabla de transacciones que esta transacción se ha completado.
  - Se elimina el segmento *de rollback de la transacción.*

- Para deshacer una transacción:
  - Se pueden deshacer las transacciones no confirmadas o parcialmente confirmadas (`punto de guardado`)
  - Los valores almacenados en el segmento *de rollback* se aplican a la base de datos (valores antiguos)

3.2.7 Recuperación de Caídas: Oracle

Tema 3.2: Técnicas de recuperación de caídas

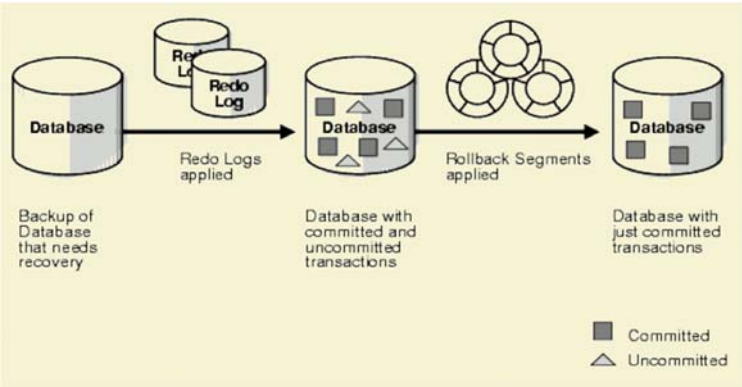
- Problemas a resolver cuando se produce un fallo:
  - Datos perdidos (fuera del espacio de tabla) de transacciones confirmadas
    - estos datos solo aparecerán en la bitácora
  - Datos modificados (son parte del tablespace) de transacciones no confirmadas
    - los valores antiguos de estos datos se almacenan en el bitácora *del segmento de rollback*
- Algoritmo de recuperación:
  - Si se produce un fallo:
    - Paso adelante: los cambios almacenados en la bitácora se rehacen en la base de datos

      1.-se recuperan los datos perdidos de las transacciones confirmadas

      2.-se han aplicado las modificaciones no confirmadas que estaban almacenadas en la bitácora

      3.-se han recuperado los segmentos de rollback de las transacciones no confirmadas
    - Paso atrás: *se aplican los valores antiguos de estos segmentos de retroceso*

      1.-las modificaciones ejecutadas por transacciones no confirmadas se han deshecho



3.2.7 Recuperación de Caídas: Oracle

- Problemas a resolver cuando se produce un fallo:
  - Datos perdidos (fuera del espacio de tabla) de transacciones confirmadas
    - estos datos solo aparecerán en la bitácora
  - Datos modificados (son parte del tablespace) de transacciones no confirmadas
    - los valores antiguos de estos datos se almacenan en el bitácora *del segmento de rollback*

cómo gestiona el búfer de bitácora?

- Algoritmo de recuperación:
  - Si se produce un fallo:
    - Paso adelante: los cambios almacenad… la bitácora se rehacen en la base de datos

      1.-se recuperan los datos perdidos de las tra… confirmadas

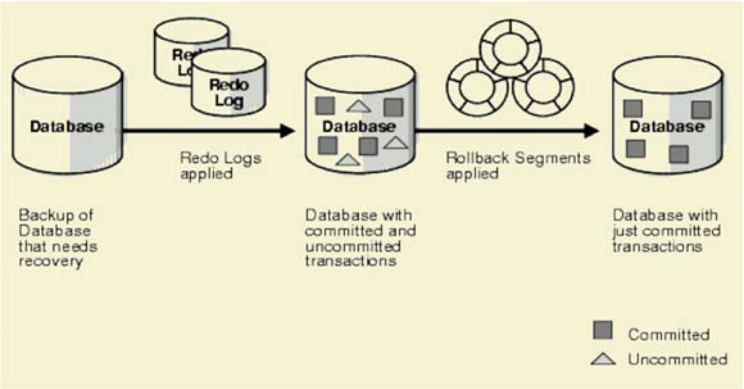
      2.-se han aplicado las modificaciones no cor… itácora

      3.-se han recuperado los segmentos de rollb…

    - Paso atrás: *se aplican los valores antigu…*

      1.-las modificaciones ejecutadas por transacciones no confirmadas se han deshecho

qué algoritmo de recuperación implementa ?



3.2.7 Recuperación de Caídas: Oracle. El espacio de tablas undo

- Oracle 9i: gestión automática de fallos basada en el espacio de tablas *undo*
  → el proceso lo gestiona `dbwr`
- Es la alternativa actual a los segmentos `rollback` . Valor por defecto desde Oracle 11g.
- Simplifica la administración.
- Es más eficiente.
- No se elimina al completar una transacción, pero se mantiene según un período de retención establecido por el administrador.
  → hace posible la llamada tecnología `flashback`

- Algunos parámetros que deben fijarse en el archivo de configuración (`ora.ini spfile`)
  - `DESHACER_GESTIÓN`
  - `DESHACER_TABLESPACE`
  - `DESHACER_RETENCIÓN`

3.2.8 Ejercicios

i. i. El algoritmo de recuperación de caídas de un DBMS debe contar necesariamente con la operación DESHACER si
   a. El algoritmo de gestión del buffer de memoria es de tipo “no robar página”
   b. El algoritmo de gestión del buffer de memoria es de tipo “no forzar escritura”
   c. Se permiten transacciones concurrentes
   d. Se permiten transacciones de larga duración

ii. El algoritmo de recuperación de caídas de un DBMS debe contar necesariamente con la operación REHACER si
   a. El algoritmo de gestión del buffer de memoria es de tipo “no robar página”
   b. El algoritmo de gestión del buffer de memoria es de tipo “no forzar escritura”
   c. Se permiten transacciones concurrentes
   d. Se permiten transacciones de larga duración

iii. En un DBMS donde se sigue un criterio de actualización inmediata de páginas sucias, la primera regla de la escritura anticipada de trazas, WAL#1:
   a. Es necesaria para garantizar la atomicidad
   b. Es necesaria si el algoritmo de gestión del buffer de memoria es de tipo “robar página”
   c. Permite la implementación de la operación DESHACER
   d. No evita que transacciones cometidas puedan perderse

iv. En un DBMS donde se sigue un criterio de actualización inmediata de páginas sucias, la primera regla de la escritura anticipada de trazas, WAL#2:
   a. Es necesaria para garantizar la atomicidad
   b. Es necesaria si el algoritmo de gestión del buffer de memoria es de tipo “robar página”
   c. Permite la implementación de la operación DESHACER
   d. No evita que transacciones cometidas puedan perderse

3.2.8 Ejercicios

Tema 3.2: Técnicas de recuperación de caídas

v. Respecto ARIES se puede afirmar:
   a. En los puntos de verificación se graban en el archivo de datos las páginas sucias
   b. Es un algoritmo de recuperación de caídas adecuado para una estrategia del buffer de datos de tipo forzar
   c. Es un algoritmo de recuperación de caídas adecuado para una estrategia del buffer de datos de tipo robar
   d. En la fase de deshacer se retrocede hasta alcanzar el primer evento que hizo que alguna página cambiara a sucia

vi. La fase de “rehacer” en ARIES, en su formulación original:
   a. Rehace única y exclusivamente aquellos eventos que aún no han sido almacenados en el archivo de datos
   b. Rehace tanto aquellos eventos de transacciones que han sido confirmados como los que no
   c. Utiliza el campo lastLSN almacenado en cada traza para rehacer con mayor eficiencia
   d. Es precedida por la fase “deshacer”

vii. La fase de “deshacer” en ARIES, en su formulación original:
   a. Deshace única y exclusivamente aquellos eventos que aún no han sido almacenados en el archivo de datos
   b. Deshace tanto aquellos eventos de transacciones que han sido confirmados como los que no
   c. Utiliza el campo lastLSN almacenado en cada traza para deshacer con mayor eficiencia
   d. Es precedida por la fase “análisis”

viii. En ARIES se puede afirmar que la base de datos se encuentra exactamente en el estado que se encontraba en el instante de la caída:
   a. Al final de la fase de análisis
   b. Al final de la fase “rehacer”
   c. Al final de la de “deshacer”
   d. Nunca

3.2.8 Ejercicios

ix. Dado el siguiente escenario, indica cómo se recuperaría la BBDD siguiendo ARIES

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	B	....
2	0	T2	Act	A	....
3	2	T1	Act	B	....
4	0	T3	Act	C	....
5	4	T3	Act	C	....
6	ini_control				
7	fin_control				InfoTables
8	3	T1	Conf		....
9	0	T4	Act	mi	....
10	2	T2	Act	C	....
11	10	T2	Act	C	....
12	5	T3	Conf	C	....

Tabla de transacciones:

TRAN_ID	últimoLSN	tipo
T1	3	Act
T2	2	Act
T3	5	Act

Tabla de páginas sucias:

PAG_ID	recLSN
C	4
B	1
A	2

3.2.8 Ejercicios

x. Describe las tablas de transacciones, páginas sucias y min(recLSN) tras la fase de análisis

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	B	....
2	0	T2	Act	A	....
3	2	T1	Act	B	....
4	0	T3	Act	C	....
5	4	T3	Act	C	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T4	Act	mi	....
8	2	T2	Act	C	....
9	8	T2	Act	C	....
10	5	T3	Conf	C	....

Transacción tabla :

TRAN_ID	últimoLSN	tipo

Sucio paginas tabla :

PAG_ID	recLSN

Mín(recLSN)=

3.2.8 Ejercicios

xi. Enumera las trazas que reharían durante la fase de “rehacer”

traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	B	....
2	0	T2	Act	A	....
3	1	T1	Act	B	....
4	0	T3	Act	C	....
5	4	T3	Act	C	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T4	Act	mi	....
8	2	T2	Act	C	....
9	8	T2	Act	C	....
10	5	T3	Conf	C	....

Rehacer trazas

3.2.8 Ejercicios

xii. Enumera las trazas de las transacciones que se reharían durante la fase “deshacer”

Traza:

LSN	anteriorLSN	TRAN_ID	TIPO	PAG_ID	MÁS_INF
1	0	T1	Act	B	....
2	0	T2	Act	A	....
3	2	T1	Act	B	....
4	0	T3	Act	C	....
5	4	T3	Act	C	....
4	control_ini				
5	fin_control				InfoTables
6	3	T1	Conf		....
7	0	T4	Act	mi	....
8	2	T2	Act	C	....
9	8	T2	Act	C	....
10	5	T3	Conf	C	....

Trans .	Trazas a deshacer