SEP INSTITUTO TECNOLOGICO de la laguna

Base de Datos II

Notas del Curso

Ing. José Ruiz Ayala

Enero del 2002

Programa del Curso

I Recuperación

II Integridad

III Concurrencia

IV Seguridad

V B. de Datos Distribuidas

Bibliografía

- Base de Datos C.J. Date (vol. II)

- Base de Datos Henry F. Korth

- B. DE D. Distribuidas Ceri y Pelagatti.

Unidad I

Recuperación de un sistema de Base de Datos después de una falla

Tipos de Fallas

- Falla en suministro de energía
- Falla de Software (aplicaciones)
- Falla de Hardware (equipo)

Plan emergente

A) Operación del equipo
- Instalaciones
- Encendido Apagado
- Restauración

B) Operación de la empresa
- Seguir operando
- Actualización

C) Soporte del equipo
- Equipo similar
- Pruebas

D) Respaldo de información
- Preparación, establecer el horario de respaldo
- Resguardo para prever un daño mayor
- Restauración

1 Bitácora incremental con actualizaciones diferidas

Supongamos un proceso (I), mediante el cual se retiraran $ 50.00 de una cuenta A(A=100).

La bitácora o archivo temporal, contendrá un registro para marcar el inicio de cada transacción, y otro para marcar el final. El formato de los registros del detalle de la transacción tendrá el siguiente formato:

Nombre de la transacción Ti
Nombre del Registro que se actualiza Nom-Reg
Valor final(después de la transacción)

< Ti, Nomreg , valfinal >

Bitácora

<T1,inicio>
<T1,A,50>
<T1,Final>

Esta bitácora es un archivo que se debe de guardar en un medio **NO-VOLATIL**

Si no hay registros en bitácora todo esta bien, Si los hay, no se hizo la actualización, y se manda a un proceso de recuperación.

Caso 1: (Bitácora Incompleta)

Como no existe un registro Ti final, y eso quiere decir que la transacción no se termino de registrar en la bitácora, no podemos concluir el proceso, se enviara un mensaje al usuario para indicarle que debe realizarla nuevamente ; luego se procederá a limpiar la bitácora.

Caso 2: (Bitácora Completa)

La rutina de recuperación detecta un registro Ti Final, procede a vaciar el contenido de la bitácora a la Base de Datos, posteriormente, le envía un mensaje al usuario que le dice que la ultima transacción que estaba realizando, se ha recuperado exitosamente, por ultimo limpia la bitácora.

Ejemplo 2 :

Considerar un proceso de 2 transacciones, en donde la 1ª de ellas, consiste en retirar 50 pesos de la cuenta A (A=150) y una transacción 2, mediante la cual, vamos a transferir 100 pesos de una cuenta B (B=200) a una cuenta C(C=120)

Bitácora	Base de Datos
<T1,Inicio>	
<T1,A,100>	
<T1,Final>	A = 100
<T2,Inicio>	
<T2,B,100>	
<T2,C,220>	
<T2,Final>	B=100 C=220

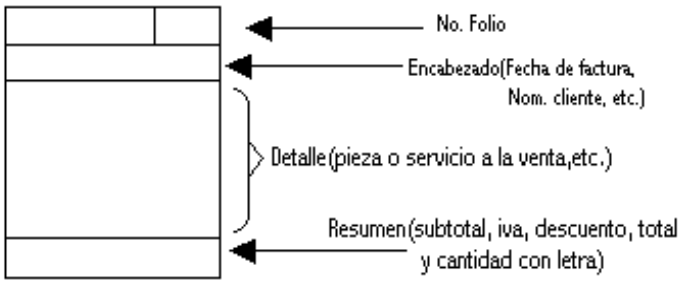
Se puede actualizar independientemente las 2 transacciones (como el ej.) así si existe una falla puede ser que no se tengan que pedir al usuario las 2 transacciones o se puede actualizar todo al final.

Consideraciones

La bitácora de actualizaciones diferidas es relativamente simple de implementar, pero esta condicionada a sistemas muy bien estructurados, o bien sistemas no muy complejos.

FLUSH: Este estatuto le indica al sistema que vacíe los buffers de memoria, esto ayuda a minimizar el riesgo de no completar la transacción en caso de una falla.

Considerar un sistema típico de facturación de cualquier empresa que venda productos



1er Tabla de la factura "Encabezado de la factura"

FAC-ENC

Num_Fac	Llave primaria
Fec_fac	
Cve_Cte	

FAC-DET

Num_Fact	Llave primaria
Parte	clave de la pieza en venta
Cantidad	
Costo	
Precio	

INVENTARIO

Parte	Llave primaria
Desc	
Existencia	
Costo	
Precio	

Ejemplo

FAC_ENC

Num_Fac	Fec_Fac	Cve_cte
501	9/02/99	2050

Fac_Det

Num_Fac	Parte	Cantidad	Costo	Precio
501	F	5	100	150
501	C	10	150	175
501	D	2	200	210
501	A	20	110	140

Inventario

Parte	Desc	Exist	Costo	Precio
A	Lampara Fluorescente	100	110	140
C	Balastra	80	150	175
D	Foco 100 W	40	200	210
F	Cable	60	100	150

Procedure Actualiza

```
1)      SELECT FAC_DET
        SEEK MFACT
        DO WHILE NUM_FAC = MFACT .AND. .NOT. EOF()
                SELECT INVENTARIO
                SEEK FAC_DET.PARTE
2)       REPLACE EXISTENCIA WITH EXISTENCIA-FAC_DET.CANTIDAD
                SELE FAC_DET
                SKIP
        ENDDO

3)      FLUSH
```

1. Crear registro inicio en bitácora
2. Crear registro de detalle en bitácora
3. Crear registro final

Implementación de la bitácora de actualizaciones diferidas

Al inicio de la aplicación (antes de que el usuario emprenda alguna acción), verificamos como terminó el sistema en la sesión anterior.

```
USE BITACORA
IF RECCOUNT() > 0
        DO REORGANIZA
        DO RECUPERA
ENDIF

DO CREA_BIT
DO ACTUALIZA
```

BITACORA

NOM_TRA	C2
TIPO	C1
NOM_REG	C15
DESPUES	N10

PROC CREA_BIT

1)

```
SELE BITACORA
APPE BLANK
REPL NOM_TRA WITH "T1", TIPO WITH "i"
SELE FACT_DET

SEEK MFACT
DO WHILE NUM_FACT = MFACT .AND. .NOT. EOF()
        SELECT INVENTARIO
        SEEK FACT_DET.PARTE
```

2)

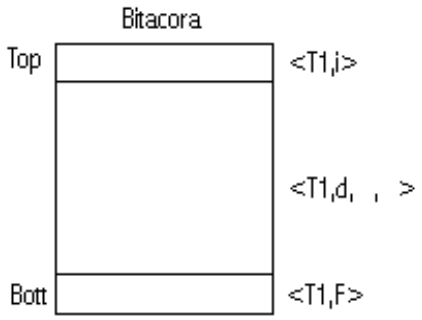
```
        MRES = EXISTENCIA-FACT_DET.CANTIDAD
        SELE BITACORA
        APPE BLANK
        REPL NOM_TRA WITH "T1", TIPO WITH "d", ;
        NOM_REG WITH FAC_DET.PARTE, DESPUES WITH MRES
        SELE FACT_DET
        SKIP
ENDDO
```

3)

```
SELE BITACORA
APPE BLANK
REPL NOM_TRA WITH "T1", TIPO WITH "f"
FLUSH
```

Al final del procedimiento actualiza, limpiamos la bitácora.

```
SELE BITACORA
ZAP                    *
```



*ZAP requiere que el archivo este abierto en forma exclusiva(monousuario)

PROC RECUPERA

```
SELE BITACORA
GO BOTTOM
IF TIPO #  "f"
        WAIT "ULTIMA TRANSACCION NO SE REALIZO" + ;
                "<VOLVER A EJECUTARLA   <ENTER>"
ELSE
        **VACIA BITACORA A B. DE D.
        SELE BITACORA
        GO TOP
        SKIP
        DO WHILE TIPO = "d"
                SELE INVENTARIO
                SEEK BITACORA.NOM_REG
                REPL EXISTENCIA WITH BITACORA.DESPUES
                SELE BITACORA
                SKIP
        ENDDO
        FLUSH
        WAIT "TRANSACCION RECUPERADA EXITOSAMENTE"
ENDIF
SELE BITACORA
ZAP
** FIN RECUPERA
```

El proceso rehacer la transacción Ti REDO(Ti), debe de ser idempotente, es decir, poder realizarse N veces con el mismo resultado.

Si el procedimiento recupera falla en el ciclo es decir, se cae el sistema cuando el procedimiento vaya en el ciclo, al restaurarse el sistema, entra al procedimiento real pero y como ve que la bitácora esta llena, trata de actualizar la B. de D. y los datos que ya había actualizado simplemente los reescribe.

Bitácora Incremental con actualizaciones inmediatas

Formato del registro < Ti, Registro, Antes, Después > considerando el mismo ejemplo en donde tenemos una transaccion1 que retira $ 50 pesos de la cuenta A (A = 100). Una transacción 2 que transfiere $ 100 pesos de B a C (B = 200 y C = 150).

Bitácora	B. de D.
< T1, Inicio>	
< T1, A , 100 , 50 >	A = 50
< T1, Final>	
< T2, inicio>	
< T2, B , 200 , 100 >	B = 100
< T2, C , 150 , 250 >	C = 250
< T2, Final>	

Caso 1

Después de una falla el sistema revisa la bitácora y encuentra el registro final de la ultima transacción, es decir la bitácora esta completa, en este caso ejecutara un REHACER (REDO(Ti)), lo cual consiste en vaciar los valores de después, contenidos en la bitácora, a la B. de D.

Se envía un mensaje al usuario donde se le confirma que la transacción se llevo a cabo exitosamente y borramos bitácora.

Caso 2

El sistema determina si la bitácora no esta completa pudiendo a la vez presentarse 2 situaciones:

A) El ultimo registro de la bitácora no es un registro de final. En estas condiciones se opta por deshacer la totalidad de las transacciones (Deshacer o UNDO(Ti)), es decir, vaciar la información de antes a la B. de D. se le envía un mensaje al usuario para notificarle que las ultimas transacciones que realizaba no se pudieron llevar a cabo, por lo cual deberá realizarlas nuevamente.

B) El sistema si encuentra un registro de final de una transacción intermedia, se envía un mensaje al usuario para que el determine en base a la independencia de las transacciones si puede completar las transacciones

faltantes, en este caso, el sistema aplica rehacer hasta la ultima transacción completa, borra la bitácora y cede el control al usuario para que complete las transacciones. En caso contrario, si el usuario determina que no puede completar las transacciones faltantes, el sistema aplica un deshacer (UNDO(Ti)) como en el caso **A)** , para luego indicar al usuario que vuelva a empezar.

Implementar Bitácora Actualizaciones Inmediatas

BITACORA	
NOM_TRA	C2
TIPO	C1
NOM_REG	C15
ANTES	N10
DESPUES	N10

```
PROC ACTUALIZA
SELE BITACORA
APPE BLANK
REPL NOM_TRA WITH “T1” , TIPO WITH “i”
SELE FAC_DET
SEEK MFACT
DO WHILE NUM_FAC = MFACT .AND. .NOT. EOF( )
        SELE INVENTARIO
        SEEK FAC_DET.PARTE
        MRES = EXISTENCIA - FAC_DET.CANTIDAD
        SELE BITACORA
        APPE BLANK
        REPL NOM_TRA WITH “T1”, ;
                    TIPO WITH “d”, ;
                    NOM_REG WITH FAC_DET.PARTE, ;
                ANTES WITH INVENTARIO.EXISTENCIA, ;
                    DESPUES WITH MRES
        SELE INVENTARIO
        REPL EXISTENCIA WITH MRES
        SELE FAC_DET
        SKIP
ENDDO
SELE BITACORA
APPE BLANK
REPL NOM_TRA WITH “T1”, TIPO WITH “f”
FLUSH
WAIT “ACTUALIZACION TERMINADA”
SELE BITACORA
ZAP
```

Considerando el caso mas simple de una sola transacción y solamente las opciones principales tenemos :

```
PROC RECUPERA
SELE BITACORA
GO BOTT
IF TIPO = “f”
        ** BITACORA COMPLETA, APLICAR REHACER
        GO TOP
        SKIP
        DO WHILE TIPO = “d”
                SELE INVENTARIO
                SEEK BITACORA.NOM_REG
                REPL EXISTENCIA WITH BITACORA.DESPUES
                SELE BITACORA
                SKIP
        ENDDO
        FLUSH
        WAIT “TRANSACCION VERIFICADA <ENTER>”
ELSE
        ** BITACORA INCOMPLETA, APLICAR DESHACER
        GO TOP
        SKIP
        DO WHILE TIPO = “d”
                SELE INVENTARIO
                SEEK BITACORA.NOM_REG
                REPL EXISTENCIA WITH BITACORA.ANTES
                SELE BITACORA
                SKIP
        ENDDO
        FLUSH
        WAIT “ TRANSACCION DESHECHA <ENTER>”
ENDIF
ZAP
** FIN RECUPERA
```

Antes de llamar a un proceso de recuperación (DO RECUPERA) es necesario verificar la situación de nuestras tablas, después de que ocurra la falla.

A) Daño mínimo a los índices

PROC REORGANIZA

```
SELECT A
USE ARCH1 EXCL
REINDEX
USE ARCH2 EXCL
REINDEX
.
.
.
USE ARCHN EXCL
REINDEX
```

Para todos los archivos del sistema

El daño puede resultar no tan leve, de tal forma que el sistema manejador de la base de datos no lo puede abrir (menos indexar), podemos optar por lo siguiente.

B) De un respaldo recuperamos el ARCH1.cdx

- Borrar .cdx
- Recuperar del respaldo el mismo .cdx
- reindexar

C) No hay respaldo o no queremos recurrir a el

- Borrar .cdx
- Del manual técnico reconstruir los índices

```
INDEX ON <EXPR1> TAG <NOM1>
INDEX ON <EXPR2> TAG <NOM2>
                .               .
                .               .
                .               .
INDEX ON <EXPRN> TAG <NOMN>
```

D) Daño a los datos (dbf).

- Se recurre al respaldo más reciente.