

Recuperació de BD

Gestió i Administració de Bases de Dades.
Grau en Enginyeria Informàtica

Oriol Ramos Terrades

Carles Sánchez Ramos

Departament de Ciències de la Computació

Continguts

- Classificació de caigudes del sistema
- Estructures d'emmagatzematge
- Recuperació basada en registre de sistema
- Paginació a l'ombra (*shadowing*)
- Concurrència i recuperació
- Errada d'emmagatzematge no volàtil
- Algorisme de recuperació ARIES

Introducció

Els computadors estan subjecte a errades, produïdes per diferents causes i es pot perdre informació.

El **SGBD** realitza accions per a garantir que les propietats d'atomicitat i durabilitat de les transaccions es preserven enfront d'errades del sistema.

L'esquema de recuperació:

- És el responsable de la restauració de la BD a l'estat consistent previ a l'errada.
- Ha de mantenir alta disponibilitat, minimitzant el temps en que la BD no es pugui utilitzar després de l'errada.

Classificació de caigudes de sistema

Errada en la transacció: Dos tipus d'errors que fan que la transacció falli:

- **Error lògic:** Error d'execució. Entrada incorrecte, *divide by zero*, *stack overflow*. Alguns evitables per programador.
- **Error de sistema:** BD està en estat no desitjat. Per exemple, *deadlock*. Transacció no pot continuar, es reinicialitza més tard.

Caiguda de sistema: Mal funcionament hardware o error del SGBD. Es perd contingut de RAM i s'avorta transacció. **Errada - parada:** SGBD fa comprovacions internes que paren el sistema quan detecten un error.

Errada de disc: Bloc de disc es corromp. S'utilitzen còpies de les dades en altres discos o cintes per a restaurar.

Classificació de caigudes de sistema

Els algorismes per a garantir consistència de la BD i durabilitat de les transaccions després d'errades tenen dos passos:

1. Accions per a recopilar tota la informació necessària per a la recuperació de les transaccions.
2. Accions per a restablir el contingut de la BD a un estat consistent.

Estructures d'emmagatzematge

La preservació de les propietats d'atomicitat (A) i durabilitat (D) de les transaccions està condicionada pel següents factors:

- els tipus d'emmagatzematge
- els tipus d'errors i la recuperació de transferència de dades
- l'accés a les dades

Tipus d'emmagatzematge

Classificables segons velocitat, capacitat, resistència a errades. Classificació segons volatilitat:

- **Emmagatzematge volàtil:** Informació no sobreviu a caigudes de sistema (RAM). Ràpid accés, circuits integrats.
- **Emmagatzematge no volàtil:** Informació sobreviu a caigudes de sistema. Dispositius electromagnètics. Disc i cintes, accés lent. Estat sòlid, més ràpid i compacte.
- **Emmagatzematge estable:** Informació quasi impossible que es perdi (sistemes replicats locals o remots, RAID).

Sistemes SAI: Mantenen fonts d'alimentació de reserva per a mantenir RAM estable.

Errors i recuperació de transferència de dades

Transferència entre memòria i disc pot acabar en tres formes:

- **Èxit:** tota la informació transferida.
- **Errada parcial:** Errada a mitja transferència, la informació en el bloc de destí és incorrecte.
- **Errada total:** Errada massa aviat, el bloc de destí està intacte, i la informació és correcte (encara que potser antiga).

Errors i recuperació de transferència de dades

En cas de transferència correcte (**Èxit**), es realitzen els passos:

- Transferir el bloc lògic al primer disc de *backup*.
- Si correcte, transferir el bloc lògic al segon disc de *backup*.
- Si correcte, senyal emetre senyal de confirmació.

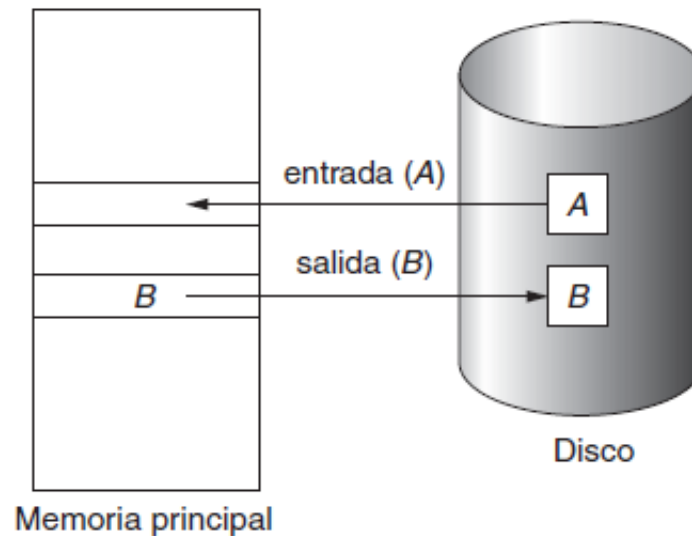
Si és **Errada Parcial**, cal recuperar el bloc físic original. S'examinen els dos blocs de *backup*:

- Si iguals, agafar un i restaurar la informació.
- Si error detectable en un bloc corregir-lo i restaurar la informació.

Accés a dades

Transferència entre disc i RAM en blocs (o pàgines).

Operació d'entrada (A), sortida (B), essent A i B blocs a disc.



Cada transacció té una quota de p pàgines a RAM, que utilitza per a manipular dades.

Accés a dades

Pàgines: Blocs de memòria (RAM o cache) on es guarden les pàgines llegides de disc.

- **Bit d'ús d'escriptura (BE):** Si la pàgina en buffer s'ha modificat (1) o no (0). Guardar a disc només pàgines amb BE=1.
- **Bit d'accés (BA):** Si la pàgina s'està utilitzant (1) i per tant no es pot guardar a disc.

Es guarden a cache les pàgines de dades i els fitxers índex altament utilitzats.

Accés a dades

Dues estratègies per a escriure a disc buffers modificats:

Actualització *in place*: la pàgina de buffer es guarda al mateix lloc que el bloc original sense modificar. Una sola còpia de les dades.

Paginació a l'ombra (*Shadowing*): Pàgina del buffer es guarda en un lloc de disc diferent a la pàgina original, guardant diverses còpies de la pàgina en el temps.

- Es guarden valors BFIM, AFIM de cada dada modificada, no cal guardar-les en registre de sistema.

BFIM (*BeFore IMage*) d'un objecte: Valor abans actualitzar.

AFIM (*AFter IMage*) d'un objecte: Valor després actualitzar.

Accés a dades. Paginació a demanda:

A mida que transacció necessita dades va carregant pàgines a RAM de disc (entrada).

Quan necessita més pàgines i té la quota ocupada, allibera una guardant el contingut a disc (sortida). Algorisme per a decidir quina: LIFO, FIFO, LRU, etc.

Quan apareix punt de sincronització (*CheckPoint*) o finalitza una transacció (`commit`): Es força la sortida de tots els blocs de RAM a disc.

Anul·lació en cascada

Anul·lació de transaccions: Si la transacció T falla després d'actualitzar la BD, cal anul·lar T i restaurar el valor anterior (BFIM) cercant registres en el registre de sistema.

- Si T anul·lada, cal anul·lar transacció S que hagi llegit algun valor escrit per T .
- Si S anul·lada, cal anul·lar transacció R que hagi llegit algun valor escrit per S ... **anul·lació en cascada.**

Anul·lació en cascada :

Complexa i amb molt *overhead*. Passa si protocols de control de concurrència no garanteixen almenys recuperació sense casca (*cascadeless*).

Anul·lació en cascada

Exemple:

T_1
leer_elemento(A)
leer_elemento(D)
escribir_elemento(D)

T_2
leer_elemento(B)
escribir_elemento(B)
leer_elemento(D)
escribir_elemento(D)

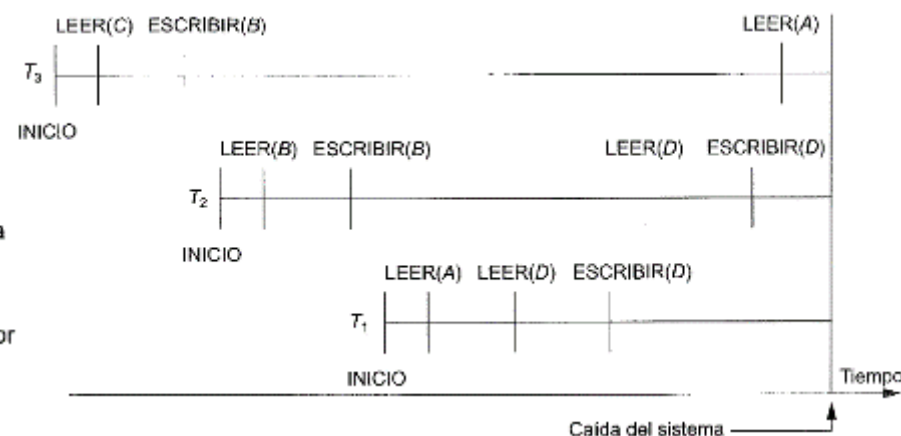
T_3
leer_elemento(C)
escribir_elemento(B)
leer_elemento(A)
escribir_elemento(A)

	A	B	C	D
	30	15	40	20
[iniciar transacción, T_3]				
[leer_elemento, T_3 , C]				
* [escribir_elemento, T_3 , B, 15, 12]		12		
[iniciar transacción, T_2]				
[leer_elemento, T_2 , B]				
** [escribir_elemento, T_2 , B, 12, 18]		18		
[iniciar transacción, T_1]				
[leer_elemento, T_1 , A]				
[leer_elemento, T_1 , D]				
[escribir_elemento, T_1 , D, 20, 25]				25
[leer_elemento, T_2 , D]				
** [escribir_elemento, T_2 , D, 25, 26]				26
[leer_elemento, T_3 , A]				

← Caída del sistema

* T_3 se anula porque no alcanza su punto de confirmación.

** T_2 se anula porque lee el valor del elemento B escrito por T_3 .



Anul·lació en cascada

En la pràctica, anul·lació en cascada no es produeix perquè els SGBD garanteixen planificacions lliures d'anul·lació en cascada.

Per tant, no cal guardar operació *llegir element* en el registre de sistema, operació necessària en cas de restaurar l'anul·lació en cascada.

Recuperació i atomicitat dades

Donada la transacció T_3 , en que es produeix una errada a $t=4$.

t	T_3	X	Y
1	read_item(X)	200	100
2	$X=X-50$	200	100
3	write_item(X)	150	100
4	ERRADA DE SISTEMA		
5	read_item(Y)		
6	$Y=Y+50$		
7	write_item(Y)		

En moment de l'errada no sabem si bloc de RAM BX s'ha guardat a disc o no. Dos procediments de recuperació simple:

- **Tornar a executar T_3 :** X té el valor de 100 (150 - 50). Sistema entra a un estat inconsistent, on està.
- **No tornar a executar T_3 :** X, Y tenen valors 150 i 100, no ha sortit de l'estat inconsistent.

Recuperació i atomicitat dades

Ambdós procediments porten a un estat inconsistent. S'han realitzat canvis sense que la transacció els hagi pogut fer tots:
Pèrdua d'atomicitat.

Per aconseguir atomicitat, estratègia **Registre abans escriptura (WAL – Write Ahead Login)**: Abans d'una actualització, cal escriure en el **registre de sistema** la informació suficient per a recuperar els canvis realitzats per la transacció.

Amb aquesta estratègia, tècniques de **recuperació basada en registre de sistema**.

Recuperació basada en registre de sistema

Estratègia de recuperació més utilitzada en motors de BD.

Suposem **execució seqüencial de transaccions**: només una transacció està activa en cada moment.

Recuperació basada en registre de sistema

Registre del sistema (*fitxer log, registre històric*): Seqüència de fitxers que manté un registre de tots els canvis realitzats en les dades de la BD i en l'ordre en que es produeixen.

Entrades del registre de sistema:

```
[start transaction T]
[write_item T,X,<old_value>,<new_value>]
[commit T]
```

Recuperació basada en registre de sistema

Per a garantir un correcte registre de totes les operacions, el registre de sistema s'actualitza abans d'escriure el bloc a disc.

Registre de sistema:

- Es guarda en emmagatzematge estable.
- És de grans dimensions.
- La seva informació és esborrable sota condicions.

Recuperació basada en registre de sistema

Dos mètodes de recuperació:

- Modificació diferida de la BD
- Modificació immediata de la BD

Modificació diferida de la BD

Garanteix la atomicitat de les transaccions amb el registre de sistema, però retardant l'execució de les operacions d'escriptura fins que la transacció es confirma.

Si el sistema cau abans que la transacció T es confirmi o avorti, els registres del sistema de T s'ignoren.

Quan T fa `commit`:

- Registres de sistema de T es guarden en emmagatzematge estable.
- Registres de sistema s'utilitzen per a fer les escriptures a disc.

Modificació diferida de la BD

Execució d'una transacció T_i :

1. S'insereix registre `<start transaction T_i >` a registre de sistema abans d'iniciar transacció.
2. Qualsevol escriptura a un element Q s'insereix abans un registre `[write_item T_i , Q , <new_value>]`
3. S'insereix registre `[commit T_i]` si la transacció es confirma.

Modificació diferida només necessita `<new_value>` en el registre `write_item()`, doncs mai haurà de desfer cap actualització a disc.

Modificació diferida de la BD

Exemple:

Dues transaccions T_3, T_4 .

T_3	T_4
read_item(X)	read_item(Z)
X=X-50	Z=Z-100
write_item(X)	write_item(Z)
read_item(Y)	
Y=Y+50	
write_item(Y)	

Si es produeix execució seqüencial T_3, T_4 amb valors inicials $X=100$, $Y=2000$, $Z=700$, en el registre de sistema es guarden actualitzacions BD:

t	Registre de Sistema	Transferència RAM -> Disc	X	Y	Z
1	start transaction T3		1000	2000	700
2	write_item T3,X,950		1000	2000	700
3	write_item,T3,Y,2050		1000	2000	700
4	commit T3		1000	2000	700
5		X (RAM) -> X (disc)	950	2000	700
6		Y (RAM) -> Y (disc)	950	2050	700
7	start transaction T4		950	2050	700
8	write_item,T4,Z,600		950	2050	700
9	commit T4		950	2050	700
10		Z (RAM) -> Z (disc)	950	2050	600

Modificació diferida de la BD

Mitjançant registre de sistema (o registre històric), el SGBD pot recuperar qualsevol errada en que es perdi informació en RAM. L'esquema de recuperació utilitza el següent **procediment de recuperació**:

REFER(T_i): Fixa el valor de tots els elements de dades actualitzats per T_i als nous valors especificats en el registre de sistema.

L'operació **refer** és **idempotent**. Garanteix el resultat correcte, encara que hi hagi una errada durant la recuperació del sistema.

Modificació diferida de la BD

En cas d'errada, les transaccions on cal aplicar el procediment refer són:

- aquelles que tinguin en el registre de sistema un `[start transaction T]` i un `[commit T]`.
- Si una transacció T té un `[start transaction T]` però no té un `[commit T]` no ha estat confirmada. No cal recuperar-la, sinó reiniciar-la.

Modificació diferida de BD: Ideal per a transaccions curtes, que ocupin pocs buffers. Per a transaccions llargues, provoca ocupació alta de buffers fins arribar a punt de confirmació.

Modificació diferida de la BD

Exemple 1: Donades dues transaccions T_1, T_2 .

T_1	T_2
leer_elemento(A)	leer_elemento(B)
leer_elemento(D)	escribir_elemento(B)
escribir_elemento(D)	leer_elemento(D)
	escribir_elemento(D)

[iniciar_transacción, T_1]
[escribir_elemento, $T_1, D, 20$]
[confirmar, T_1]
[iniciar_transacción, T_2]
[escribir_elemento, $T_2, B, 10$]
[escribir_elemento, $T_2, D, 25$]

← Caída del sistema

Entrades de T_1 es refan.

Entrades de T_2 són ignorades.

Modificació diferida de la BD

Exemple 2: Donades dues transaccions T_3, T_4 :

<i>t</i>	<i>T3</i>	<i>T4</i>	<i>Registre de Sistema</i>	<i>X</i>	<i>Y</i>	<i>Z</i>	<i>Punts d'errada de sistema</i>	
1	read_item(X)		start transaction T3	1000	2000	700		
2	X=X-50			1000	2000	700		
3	write_item(X)		write_item T3,X,950	1000	2000	700		
4	read_item(Y)			1000	2000	700		
5	Y=Y+50			1000	2000	700		
6	write_item(Y)		write_item,T3,Y,2250	1000	2000	700	ERRADA (1)	
7	commit T3		commit T3	950	2050	700		
8		read_item(Z)	start transaction T4	950	2050	700		
9		Z=Z-100		950	2050	700		
10		write_item(Z)	write_item,T4,Z,600	950	2050	700	ERRADA (2)	ERRADA (2.1)
11		commit T4	commit T4	950	2050	600		
12				950	2050	600	ERRADA (3)	

Modificació diferida de la BD

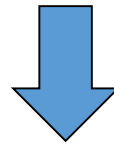
Punts d'errada i procediments de recuperació:

- **Errada (1):** No hi ha cap transacció iniciada - finalitzada en el registre històric, pel que no cal refer cap transacció. Totes es perden i cal reiniciar-les.
- **Errada (2):** $T3$ iniciada [`start`] i finalitzada [`commit`] en el registre històric. Cal *refer*($T3$) per a guardar canvis a disc.
- **Errada (2.1):** Errada durant *refer*($T3$). Com registre històric és el mateix es torna a iniciar *refer*($T3$).
- **Errada (3):** $T3, T4$ tenen inici i finalització, pel que cal *refer*($T3$) i *refer*($T4$) per a restaurar contingut.

Modificació immediata de la BD

Permet fer modificacions de la BD abans de confirmar.

En cas d'errada, cal desfer els canvis fets per la transacció cancel·lada, utilitzant el valor antic en el registre `[write_item]` del registre històric.



Operació desfer.

Modificació immediata de la BD

Execució d'una transacció T_i :

1. S'insereix registre [start transaction T_i] a registre de sistema abans d'iniciar transacció.
2. Qualsevol escriptura a un element Q s'inserta abans un registre [write_item T_i , Q , <old_value>, <new_value>]
3. S'insereix registre [commit T_i] si transacció es confirma.

Modificació immediata necessita <old_value> (per operació desfer) i <new_value> (per operació refer) en el registre [write_item].

Modificació immediata de la BD

Exemple:

Dues transaccions T_3, T_4 :

T_3	T_4
read_item(X)	read_item(Z)
X=X-50	Z=Z-100
write_item(X)	write_item(Z)
read_item(Y)	
Y=Y+50	
write_item(Y)	

Si execució seqüencial T_3, T_4 amb valors inicials $X=100$, $Y=2000$, $Z=700$, en el registre de sistema es guarden actualitzacions de la BD:

t	Registre de Sistema	Transferència RAM -> Disc	X	Y	Z
1	start transaction T_3		1000	2000	700
2	write_item $T_3, X, 1000, 950$	X (RAM) -> X (disc)	950	2000	700
3	write_item, $T_3, Y, 2000, 2250$	Y (RAM) -> Y (disc)	950	2050	700
4	commit T_3		950	2050	700
5	start transaction T_4		950	2050	700
6	write_item, $T_4, Z, 700, 600$	Z (RAM) -> Z (disc)	950	2050	600
7	commit T_4		950	2050	600

Modificació immediata de la BD

Utilitza el següent **procediment de recuperació**:

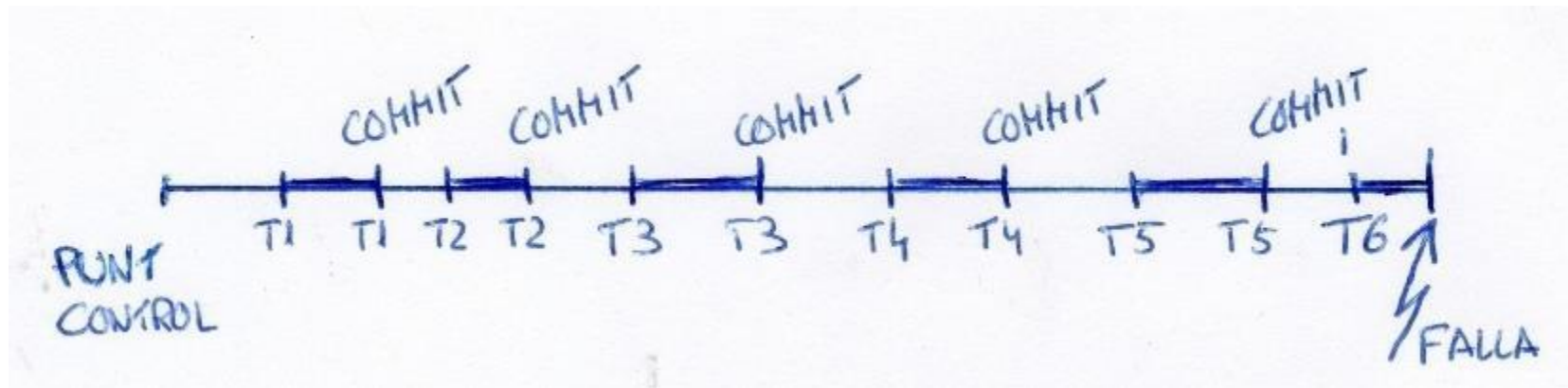
- **DESFER(T_i)**: Restaura el valor de tots els elements de dades actualitzats per T_i als valors anteriors.
- **REFER(T_i)**: Fixa el valor de tots els elements de dades actualitzats per T_i als nous valors especificats en el registre de sistema.

Les operacions **desfer** i **refer** són **idempotents**. Garanteix resultat correcte, inclús si hi ha una errada durant la recuperació del sistema.

Modificació immediata de la BD

Anul·la totes les operacions d'escriptura de la transacció activa amb procediment **DEFER**.

Torna a executar les operacions d'escriptura de les transaccions confirmades a partir del punt de control en el mateix ordre en que es van executar amb procediment **REFER**.



Modificació immediata de la BD

Procediment **DEFER**:

- Desfà l'operació d'escriptura examinant l'entrada del registre [write_item, T, Q, <old_value>, <new_value>] i posa a Q el <old_value> (valor BFIM).
- Realitza les operacions de recuperació en l'ordre invers al que es van executar i escriure en el registre del sistema.

Procediment **REFER**:

- Reescriu l'element examinant l'entrada del registre [write_item, T, Q, <old_value>, <new_value>] i posar a Q el <new_value> (valor AFIM).
- Realitza les operacions de recuperació en el mateix ordre que es van executar i escriure en el registre del sistema.

Modificació immediata de la BD

Quan succeeix una errada, mira a quines transaccions cal aplicar els procediments desfer i refer:

- Es farà **desfer**(T) a aquelles que tinguin en el registre de sistema un `[start transaction T]` i no tinguin un `[commit T]`. Transacció no finalitzada ni confirmada.
- Es farà **refer**(T) si una transacció T té un `[start transaction T]` i un `[commit T]`. Transacció finalitzada i confirmada.

Modificació immediata de la BD

Exemple 1: Donades dues transaccions T_3 , T_4 :

t	T_3	T_4	Registre de Sistema	X	Y	Z	Punts d'errada de sistema	
1	read_item(X)		start transaction T3	1000	2000	700		
2	X=X-50			1000	2000	700		
3	write_item(X)		write_item T3,X,1000,950	950	2000	700		
4	read_item(Y)			950	2000	700		
5	Y=Y+50			950	2000	700		
6	write_item(Y)		write_item,T3,Y,2000,2250	950	2050	700	ERRADA (1)	
7	commit T3		commit T3	950	2050	700		
8		read_item(Z)	start transaction T4	950	2050	700		
9		Z=Z-100		950	2050	700		
10		write_item(Z)	write_item,T4,Z,700,600	950	2050	600	ERRADA (2)	ERRADA (2.1)
11		commit T4	commit T4	950	2050	600		
12				950	2050	600	ERRADA (3)	

Modificació immediata de la BD

Punts d'errada i procediments de recuperació:

- **Errada (1):** T_3 iniciada [`start`] però no confirmada en històric: *desfer*(T_3). Saldos X , Y restituïts a 1000 i 2000 euros.
- **Errada (2):**
 1. T_4 iniciada [`start`] i no confirmada en registre històric: *desfer*(T_4). Saldo Z restituït a 700.
 2. T_3 iniciada i confirmada [`commit`] en registre històric: *refer*(T_3). Saldos X, Y actualitzats a 950 i 2050 euros.
- **Errada (2.1):** Errada durant recuperació. Reiniciar recuperació.
- **Errada (3):** T_3 , T_4 iniciades i confirmades: *refer*(T_3) i *refer*(T_4) per a restaurar contingut X , Y , Z a 950, 2050 i 600 euros.

Punts de revisió: *CheckPoints* (CP)

Quan es produeix una errada, cal recórrer tot el registre de sistema (d'inici a fi) per a determinar quines transaccions cal desfer i/o refer.

Inconvenients:

- Recorregut de l'històric costós (*overhead*).
- Si l'històric és llarg, la majoria de transaccions a refer antigues ja es deuen haver actualitzat, pel que procediment refer afegeix *overhead* innecessari.

Per a reduir *overhead*, nova entrada al registre de sistema →

***CP - CheckPoint* (punt de revisió, punt de control)**

Punts de revisió: *CheckPoints* (CP)

Accions a fer quan es produeix *CP*:

1. Suspendre temporalment l'execució de transaccions.
2. Escriptura en emmagatzematge estable de tots els registres de l'històric en RAM.
3. Escriptura a disc de tots els blocs en RAM modificats per les transaccions.
4. Escriptura en emmagatzematge estable del registre [CheckPoint] en l'històric.
5. Tornar a permetre l'execució de transaccions.

Punts de revisió: *CheckPoints* (CP)

Presència de *CheckPoint* en registre històric fa més eficient el procés de recuperació: **Tota transacció confirmada abans de CP ja està actualitzada a disc, i per tant no cal refer.**

Freqüència del CP la decideix el DBA en funció del nombre de transaccions que s'executen en el temps.

Punts de revisió: *CheckPoints* (CP)

Procediment de recuperació amb *Checkpoint* (CP) si la modificació és immediata:

- Busca la darrera transacció T_i començada abans del CP. Per a trobar-la, es recorre el registre històric cap enrere fins el primer CP. A partir d'aquí cap enrere fins a trobar el primer [start transaction T_i].

Recórrer l'històric des de [start transaction T_i] fins al final, endavant. Per a transacció T_k iniciada després T_i ,

- **DESFER**(T_k) a totes les T_k que no hi hagi [commit T_k].
- **REFER**(T_i): a totes les T_k que hi hagi [commit T_k].

Per a modificació diferida no cal fer operació desfer.

Punts de revisió: *CheckPoints* (CP)

Exemple: Donat el conjunt de transaccions

$$\{ T_0, T_1, \dots, T_{100} \}$$

sobre la planificació seqüencial

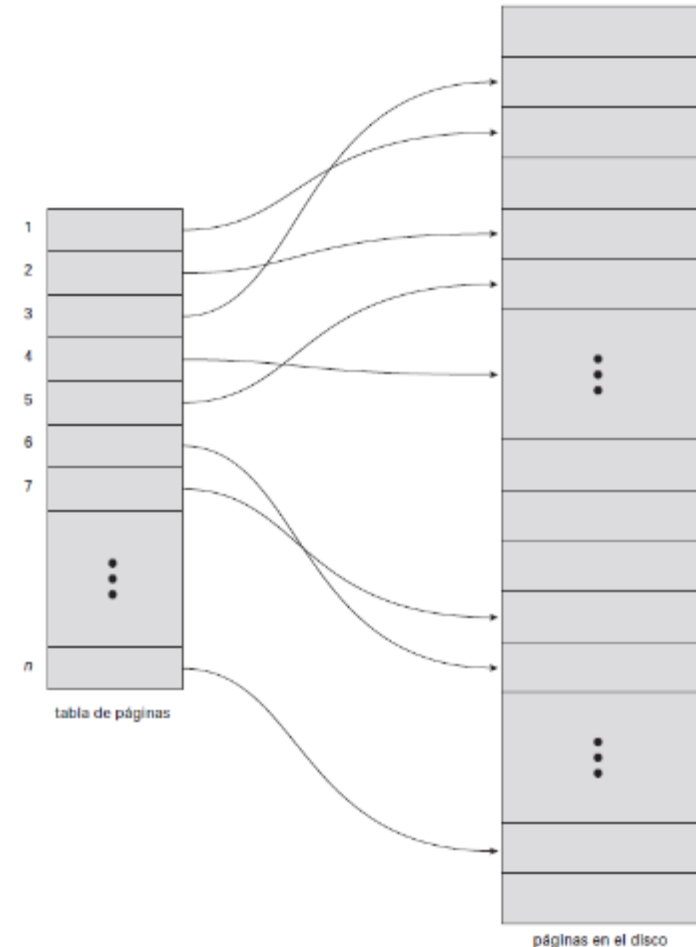
$$S = T_0, T_1, \dots, T_{67}, \text{ CP }, T_{68}, \dots, T_{100}$$

En la recuperació, donat el *CP*, només s'analitzen per a desfer/refer en el procés de recuperació les transaccions $T_{67}, T_{68}, \dots, T_{100}$.

Paginació a l'ombra (*Shadowing*)

Tècnica de recuperació alternativa a les basades en registre històric.

Taula de pàgines: Taula de n entrades, una per pàgina. Cada entrada conté un punter a disc on es troba una pàgina.



Paginació en l'ombra (*Shadowing*)

Paginació a l'ombra. Mantenir dues taules de pàgines:

- Una **Taula de Pàgines Activa** (TPA), on es guarden les pàgines actualitzades.
- Una **Taula de Pàgines a l'Ombra** (TPO), on es guarden les pàgines originals, abans d'actualitzar.

Procediment:

- A l'inici de la transacció T , $TPA = TPO$.
- Quan T fa operació `write_item(Q)`, TPA s'actualitza. Per a operacions de lectura s'utilitza TPO.

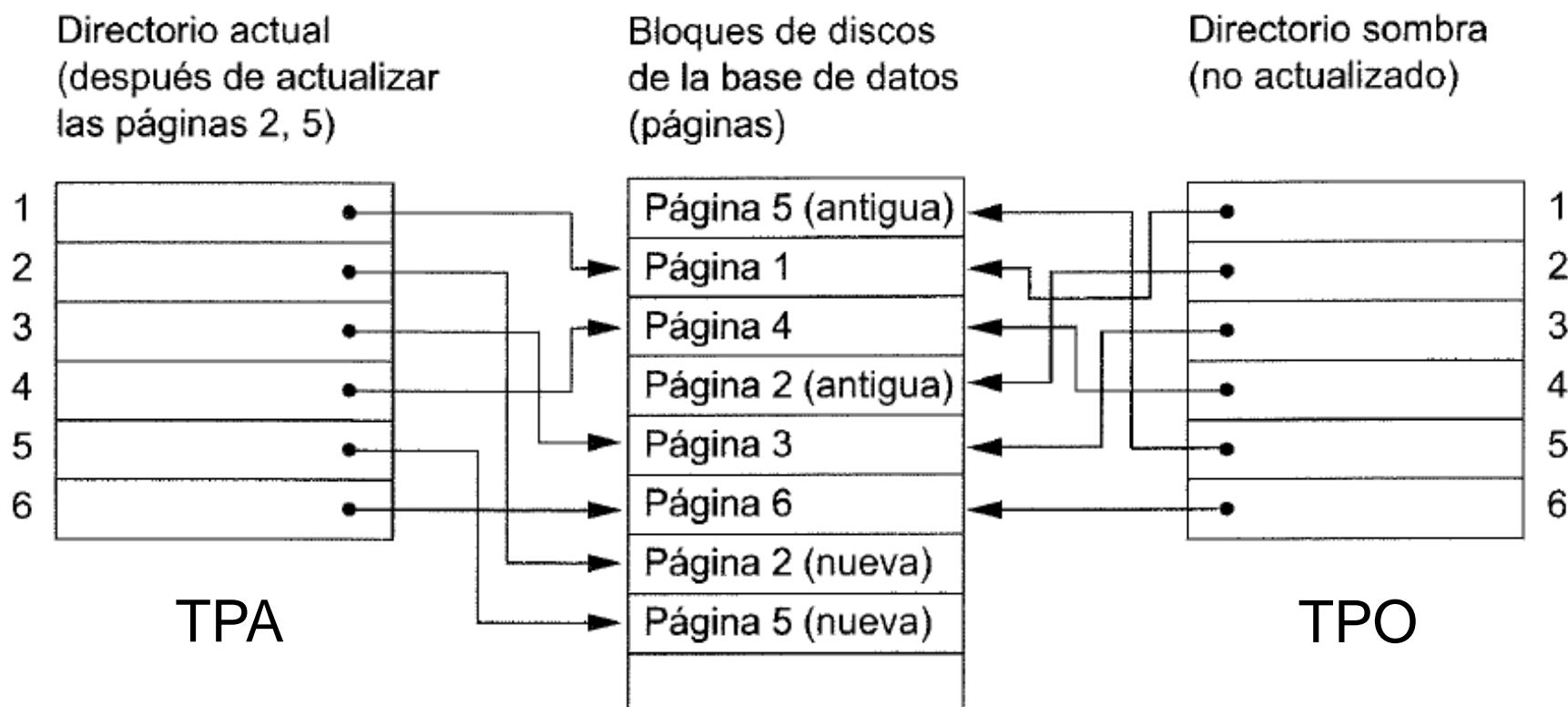
Paginació en l'ombra (*Shadowing*)

Transacció T fa l'operació $write_item(Q)$, i Q es troba a la pàgina P_i . L'operació es fa amb els passos:

1. $read_item(Q)$ si la pàgina no es troba en RAM.
2. Si és la primera escriptura que es fa a P_i , es modifica la TPA amb els passos:
 - Es busca una pàgina lliure a disc (P')
 - Es marca P' com a no lliure
 - Es modifica la TPA per a que l'entrada i -èssima que apunti P' .
3. $write_item(Q)$ sobre la pàgina P' .

Paginació en l'ombra (*Shadowing*)

Exemple sobre una transacció que modifica les pàgines 2 i 5 de disc.



Paginació a l'ombra (*Shadowing*)

Estratègia:

- La TPO conté les pàgines abans de la modificació i es guarda en un emmagatzematge no volàtil per a poder recuperar-la en cas d'errada de sistema o que la transacció avorti.
- La TPA es pot guardar en RAM mentre s'executa la transacció.
- És important poder recuperar la TPO de disc de forma ràpida per a recuperar les dades.

Paginació a l'ombra (*Shadowing*)

Accions a fer quan es confirma una transacció:

- Es transfereix les pàgines de RAM a disc, actualitzant la TPA, i no la TPO.
- Es transfereix la TPA a emmagatzematge estable.
- $TPO = TPA$ (sobreescriure TPO com a TPA).

Possibles incidències:

- Si errada de sistema abans d'acabar el pas 3, es recupera TPO.
- Si errada de sistema després d'acabar el pas 3 es conserven els canvis, no cal l'operació refer.

Paginació a l'ombra (*Shadowing*): Avantatges

- S'elimina el temps d'*overhead* en escriure en el registre de sistema.
- Menys accessos a disc. La recuperació és més ràpida, sense accessos a disc per a operacions de desfer i/o refer.

Paginació a l'ombra (*Shadowing*): Inconvenients

- Masses blocs a disc per a una transacció. En canvi, el registre de sistema ocupa poc més d'una pàgina.
- **Fragmentació de dades:** Pèrdua de localitat (dispersió) de les pàgines si s'actualitzen.
- Les pàgines antigues es tornen inaccessibles al modificar l'apuntador de la TPA (ni lliures ni accessibles). Procés **Garbage Collection** detecta i allibera aquestes pàgines.
- Difícil extensió del mètode per a execució concurrent de transaccions. Per a concurrència, més fàcil estendre mètodes basats en registre històric.
- *System R*: registre històric + paginació a l'ombra.

Concurrencia i recuperació

Fins ara, recuperació en entorns d'una sola transacció activa en un instant de temps.

Extensió de l'esquema de recuperació amb registre de sistema a executió concurrent.

Condicions:

- Un únic registre històric per a totes les transaccions.
- Blocs de RAM compartits per a una o més transaccions.
- Modificacions diferides o immediates.

Concurrencia i recuperació

Aspectes a tenir en compte:

- Interacció amb control de concurrencia
- Retrocés de transaccions
- *CheckPoints* (punts de revisió)
- Recuperació al reiniciar

Interacció amb control de concurrència

Esquema de recuperació depèn del protocol de concurrència utilitzat.

Exemple: Transacció T_5 modifica Q i després ha de retrocedir. Segons registre històric podem recuperar valor antic de Q . Però transacció T_6 ha modificat Q abans del retrocés de T_5 .

t	T_5	T_6	Registre de Sistema	$Q(T_5)$	$Q(T_6)$
1		read_item(Q)	start transaction T6		100
2	read_item(Q)		start transaction T5	100	100
3	$Q=Q+10$			110	100
4	write_item(Q)		write_item T5,Q,100,110	110	100
5		$Q=Q-10$		110	100
6		write_item(Q)	write_item,T6,Q,100,110	110	90
7	rollback T5			110	90
8		commit T6	commit T6	110	90

Si T_5 modifica Q , cap altra transacció pot modificar Q fins que T_5 hagi confirmat o retrocedit.

Retrocés de transaccions

Quan transacció T està en retrocés, es recorre cap enrere el registre de sistema desfent els canvis que ha fet T fins el registre [start transaction T].

Exemple: Transacció T_7 modifica Q dos cops durant la transacció.

Recorregut enrere del registre de sistema restitueix valor original.

t	T_7	Registre de Sistema	Q
1	read_item(Q)	start transaction T_7	100
2	$Q=Q+10$		100
3	write_item(Q)	write_item $T_5, Q, 100, 110$	110
4	...		110
5	read_item(Q)		110
6	$Q=Q+10$		110
7	write_item(Q)	write_item, $T_6, Q, 110, 120$	120
8	rollback(T_7)		120

Checkpoint

Els *Checkpoint* (*CP*) redueixen el nombre de registres a analitzar en el procés de recuperació.

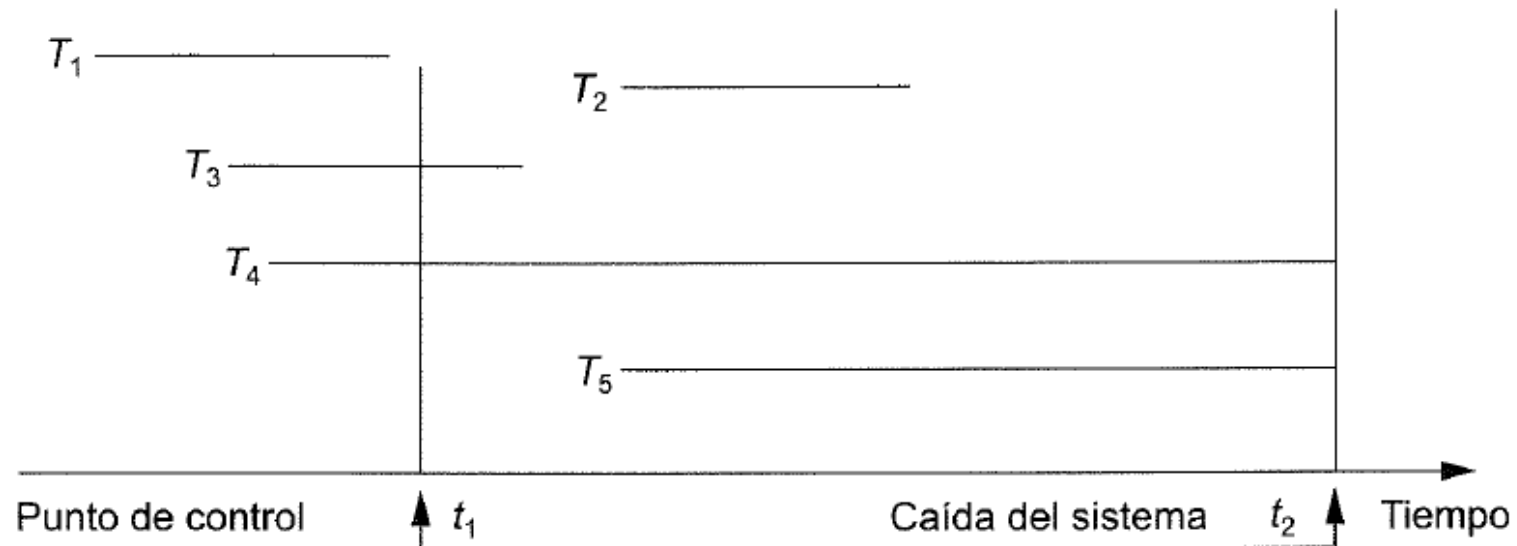
Amb una sola transacció activa en un instant de temps, la recuperació per *CP* només considerava:

- Transaccions començades després de *CP*.
- La transacció activa quan es grava el *CP* en el registre històric.

En execució concurrent, el nombre de transaccions actives quan es grava el *CP* pot ser n .

CheckPoint

Exemple: En el punt d'errada de sistema hi ha dues transaccions actives: T_4 i T_5 .



Cal afegir al registre *CP* del registre històric la llista de transaccions actives al moment de fer-se el *CP*: [CheckPoint L] L: llista de transaccions actives (T_3 i T_4).

CheckPoint

Exemple: Amb modificació diferida, transaccions T_1, T_2, T_3, T_4

Registre històric:

T_1	T_2	T_3	T_4
leer_elemento(A)	leer_elemento(B)	leer_elemento(A)	leer_elemento(B)
leer_elemento(D)	escribir_elemento(B)	escribir_elemento(A)	escribir_elemento(B)
escribir_elemento(D)	leer_elemento(D)	leer_elemento(C)	leer_elemento(A)
	escribir_elemento(D)	escribir_elemento(C)	escribir_elemento(A)

[iniciar_transacción, T_1]
[escribir_elemento, T_1 , D, 20]
[confirmar, T_1]
[punto_de_control]
[iniciar_transacción, T_4]
[escribir_elemento, T_4 , B, 15]
[escribir_elemento, T_4 , A, 20]
[confirmar, T_4]
[iniciar_transacción, T_2]
[escribir_elemento, T_2 , B, 12]
[iniciar_transacción, T_3]
[escribir_elemento, T_3 , A, 30]
[escribir_elemento, T_2 , D, 25]

← $[CP \neq \emptyset]$

- T_2, T_3 són ignorades, doncs no arriben a confirmació.
- $Refer(T_4)$, doncs confirmació està després punt de control.

← Caída del sistema

CheckPoint

Durant la realització del CP no es poden executar transaccions d'actualització de dades.

Punt de revisió difús (*CheckPoint* difús): Procés de CP que permet executar transaccions d'actualització en paral·lel.

CheckPoint difús

Permet realitzar modificacions a disc de transaccions després de gravar registre *CP* a registre històric i mentre es realitza el *CP*.

CP difús permet guanyar temps i no haver d'esperar a acabar l'escriptura de pàgines per a executar transaccions.

Estratègia: Anar guardant pàgines del *CP* a disc mentre transaccions accedeixen a altres pàgines no guardades encara pel *CP*.

CheckPoint difús: Procediment

1. Adreça de l'últim *CP* complet es guarda en una posició fix en disc.
2. Llistar tots els blocs de RAM modificats $\{B_1 \dots B_n\}$.
3. *CP* actual es guarda en històric a disc.
4. Cada B_i guardat a disc no pot ser actualitzat per cap transacció.
Blocs no actualitzats accessibles a transaccions.
5. Actualitzats tots els blocs, es guarda *CP* actual a la posició fix en disc.

Recuperació al reiniciar

Al recuperar-se d'una errada en execució concurrent, el sistema crea dues llistes en modificació immediata:

- **Llista Desfer (LD):** Transaccions a aplicar l'operació desfer.
- **Llista Refer (LR):** Transaccions a aplicar l'operació refer.

En modificació diferida només es crea LR.

Procediment de recuperació, dos fases:

1. Es defineixen les llistes LD, LR
2. Es recupera les transaccions

Recuperació al reiniciar: llistes LD, LR

1. Inicialment, LD i LR buides.

2. Actualització de LD i LR amb l'anàlisi del registre històric cap enrere:

- Per a cada registre `[commit Ti]`, s'afegeix T_i a la llista LR.
- Per a cada registre `[start transaction Ti]`,
 - a) Si T_i no està en LR (`[commit]` no trobat), afegir T_i a LD.
 - b) Si T_i està a LR, no fer res.

3. Anàlisi de L, llista de transaccions actives en el moment de fer el CP.

Per a cada transacció T_i en L ,

- Si T_i no està en LR (`[commit]` no trobat), afegir T_i a LD.
- Si T_i està a LR, no fer res.

Recuperació al reiniciar: recuperació

1. Es recorre el registre històric cap enrere desfent les operacions de les transaccions en LD. Per a cada transacció T_i desfer fins que es trobi `[start transaction T_i]`.
2. Localització del darrer registre de *CP* en registre històric.
3. Es recorre el registre històric des de *CP* cap endavant i es refà totes les operacions de les transaccions en la llista LR. Es refà cada transacció T_i des de `[start transaction T_i]` fins que es troba `[commit T_i]`.

Finalitzada la recuperació, es continua el processament de les transaccions.

Recuperació al reiniciar

Important: Primer DESFER, després REFER



Exemple: Transaccions T_{12} i T_{13} .

<i>t</i>	<i>T12</i>	<i>T13</i>	<i>Registre de Sistema</i>	<i>Q</i>	
1	read_item(Q)		start transaction T12	10	
2	Q=Q+10			10	
3	write_item(Q)		write_item T12,Q,10,20	20	
4	rollback T12			20	
5		read_item(Q)	start transaction T13	20	
6		Q=30		30	
7		write_item(Q)	write_item,T13,Q,20,30	30	
8		commit T13	commit T13	30	
9					ERRADA DE SISTEMA

Recuperació al reiniciar

Primera fase recuperació: $LR = \{ T_{13} \}$ i $LD = \{ T_{12} \}$.



Segona fase recuperació: 1) REFER, 2) DESFER

<i>t</i>	<i>Registre de Sistema</i>	<i>Q</i>	1) <i>REFER</i>	
1	start transaction T12		INICI	
2	write_item T12,Q,10,20			
3	start transaction T13			
4	write_item,T13,Q,20,30	30		
5	commit T13	30	FI	
<i>t</i>	<i>Registre de Sistema</i>	<i>Q</i>	2) <i>DESFER</i>	
6	commit T13	30	INICI	
7	write_item,T13,Q,20,30	30		
8	start transaction T13	30		
10	write_item T12,Q,10,20	10		
11	start transaction T12	10	FI	Q val 10 quan hauria de valer 30, doncs T13 ha posat valor 13

Recuperació al reiniciar

Primera fase recuperació: $LR = \{ T_{13} \}$ i $LD = \{ T_{12} \}$.

Segona fase recuperació: 1) DESFER, 2) REFER

<i>t</i>	<i>Registre de Sistema</i>	<i>Q</i>	1) DESFER	
1	start transaction T12		INICI	
2	write_item T12,Q,10,20	10		
3	start transaction T13	10		
4	write_item,T13,Q,20,30	10		
5	commit T13	10	FI	
<i>t</i>	<i>Registre de Sistema</i>	<i>Q</i>	2) REFER	
6	commit T13	10	INICI	
7	write_item,T13,Q,20,30	30		
8	start transaction T13	30		
10	write_item T12,Q,10,20	30		
11	start transaction T12	30	FI	CORRECTE

Errada d'emmagatzematge no volàtil

Errada a sistema de disc amb pèrdua.

Estratègia: Bolcar periòdicament (un cop al dia) tot el contingut de la BD en emmagatzematge estable (cinta, disc) amb còpia absoluta o incremental.

Procediment de bolcat:

1. Escriure en emmagatzematge estable tots els registres de l'històric en RAM.
2. Escriure a disc tots els blocs de RAM actualitzats.
3. Copiar el contingut de la BD en emmagatzematge estable.
4. Inserir en el registre històric el registre *CP*.

Errada d'emmagatzematge no volàtil

Procediment de recuperació:

1. Restituir a disc el darrer bolcat de la BD.
2. Refer les transaccions confirmades del registre històric des del CP. No caldrà desfer cap transacció.

Bolcat BD: Costos del bolcat en arxius:

1. Transferència de moltes dades.
2. CPU no pot executar transaccions: *overhead*.

Usualment bolcat es fa en hores de baixa activitat.

Bolcat difús: Bolcat que permet l'execució de transaccions mentre es realitza.

Algorisme recuperació ARIES

Mètode actual de recuperació en els SGBD. Certa complexitat.

Utilitza varies tècniques per a reduir el temps de recuperació i reduir sobrecàrrega de *CP*. Evita desfer moltes transaccions ja fetes.

Algorisme recuperació ARIES: conceptes

1. **Registre abans d'escriptura.**
2. **Repetició de l'històric durant el procediment REFER:** ARIES traça totes les accions del sistema anteriors a la caiguda per a reconstruir l'estat BD en el moment de la caiguda. Les transaccions no confirmades s'eliminen.
3. **Registre de canvis durant procediment DESFER:** Evita que ARIES repeteixi operacions de recuperació si durant aquesta apareix una nova errada.

Algorisme recuperació ARIES

Aspectes a tractar:

- Estructures de dades
- Algorisme
- Característiques

ARIES: Estructures de dades

Utilitza les següents estructures de dades:

1. Registre de sistema
2. Pàgina de disc
3. Taula de transaccions
4. Taula de pàgines brutes
5. Punts de control

ARIES: Registre de sistema

- Cada un dels registres té un **número de seqüència de registre (LSN)** que identifica de forma única la seva posició a disc.

$$LSN(R_i) < LSN(R_{i+1}) < LSN(R_{i+2})$$

essent R_{i+2} posterior a R_{i+1} i aquest posterior a R_i .

- ARIES divideix el registre històric en varis arxius, cadascun amb un número d'arxiu. La mida dels arxius és configurable pel DBA.
- Quan un arxiu s'emplena es crea un de nou. El codi *LSN* conté el numero d'arxiu i el desplaçament dins l'arxiu.

ARIES: registre de sistema

Camps comuns dels registres:

- LSN
- ID transacció
- Tipus d'entrada del registre.
- LSN del registre anterior de la mateixa transacció (encadenament d'operacions en una transacció).

Per a una actualització cal afegir:

- La ID pàgina que inclou element a actualitzar.
- La longitud de l'element actualitzat.
- El desplaçament respecte l'inici de pàgina.
- Els valors anterior i posterior a l'actualització de l'element.
- Anterior LSN de la mateixa transacció.

ARIES: Pàgina de disc

Cada una de les pàgines de disc conté el LSNP, el LSN del registre que ha realitzat el darrer canvi en la pàgina.

Si durant la recuperació es tracta un registre d'actualització R_i a una pàgina P_i tal que $LSN(R_i) < LSNP(P_i)$ no cal executar l'actualització, doncs la pàgina ha estat canviada per una actualització més recent → **menys operacions de recuperació.**

ARIES: taula de transaccions (TT)

La manté el gestor de transaccions. Una entrada per a cada transacció:

- ID de la transacció.
- LSN de la seva entrada més recent en el registre de sistema.
- Estat transacció (confirmada o activa).

ARIES: taula de pàgines brutes (TPB)

Una entrada per a cada pàgina bruta:

- ID de la pàgina.
- LSN del registre que ha fet l'actualització més recent a la pàgina.

ARIES: Punts de control

Consten de:

- Registre `Inici_CP` en registre de sistema.
- Registre `Fi_CP` en registre de sistema.
- Escriure `LSN` del registre `Inici_CP` en **fitxer especial**, que s'accedeix durant la recuperació per a localitzar informació del darrer punt de control.

Amb el registre `Fi_CP` s'afegeix al final del registre de sistema el següent:

- Taula de Transaccions (TT)
- Taula de Pàgines Brutes (TPB)
- Per a cada transacció activa, el `LSN` del darrer registre en el registre històric.

ARIES: Algorisme

3 fases:

1. **Fase Anàlisi:** Identificar pàgines actualitzades (brutes), transaccions actives i punt del registre on iniciar l'operació refer.
2. **Fase DESFER:** Desfer cap enrere operacions de les transaccions actives.
3. **Fase REFER:** Refer transaccions confirmades necessàries.

ARIES: Recuperació davant caiguda

1. S'accedeix a la informació (TT, TPB) del darrer *checkpoint* a través del fitxer especial.
2. **Fase d'Anàlisi:** s'analitzen els registres de sistema cap enrere actualitzant TT i TPB, transaccions a desfer:
 - Registre [`commit T`] → confirmar T a la TT.
 - Registre [`start transaction T`] → afegir T a la TT.
 - Si la pàgina P és modificada pel registre R de la transacció T' → afegir la pàgina P a TPB, actualitzant el LSNP associat a la pàgina amb valor $LSN(P) = R$.

ARIES: Recuperació davant caiguda

- 3. Fase Desfer:** S'explora cap enrere la TT i es desfan les accions realitzades per transaccions actives no confirmades.

La recuperació de l'estat de la BD per a continuar els modus d'operació normals de les transaccions.

- 4. Fase Refer:** Es començar a refer en el punt de registre on els canvis en pàgines brutes ja s'han aplicat a la BD.

Cal trobar el LSN mínim (M) de totes les pàgines brutes de la TPB. M indica la posició del registre a partir del qual ARIES ha de començar a recuperar.

ARIES: Recuperació davant caiguda

4. Fase Refer (Cont.): s'examina registres des de $LSN=M$ fins a final. Per a cada canvi de registre, en aquesta fase es verifica si cal fer canvi segons els següents criteris:

- Si canvi es fa a una pàgina P que no és a la TPB, no es fa canvi.
- Si canvi de pàgina P en registre $LSN=N$, que és a TPB, però $LSNP(P) > M$, canvi és present i cal refer.
- Si canvi de pàgina P en registre $LSN=N$ que és a TPB, però $LSNP(P) < M$, no cal fer l'actualització.

ARIES: Exemples

T_1 actualitza pàgina C, T_2 actualitza B, C i T_3 la A.

Lsn	Último_lsn	Id_tran	Tipo	Id_página	Otra_información
1	0	T_1	actualizada	C	...
2	0	T_2	actualizada	B	...
3	1	T_1	confirmada		...
4	inicio_punto_control				
5	fin_punto_control				
6	0	T_3	actualizada	A	...
7	2	T_2	actualizada	C	...
8	7	T_2	confirmada		...

Registre del sistema.

TABLA DE TRANSACCIONES

Id_transacción	Último_lsn	Estado
T_1	3	confirmada
T_2	2	en curso

TABLA DE PÁGINAS SUCIAS

Id_página	Lsn
C	1
B	2

Taules en el punt de control.

TABLA DE TRANSACCIONES

Id_transacción	Último_lsn	Estado
T_1	3	confirmada
T_2	8	confirmada
T_3	6	en curso

TABLA DE PÁGINAS SUCIAS

Id_página	Lsn
C	7
B	2
A	6

Taules després de l'anàlisi.

ARIES: recuperació de l'exemple

1. Es cerca el *CP* des de l'últim registre cap enrere. Trobat $LSN=4$, per tant cal analitzar des de LSN 4 fins a 8. El registre $LSN=5$ conté Taula de Transaccions Actives (TT) i la Taula de Pàgines Brutes (TPB).

2. Anàlisi dels registres de 4 a 8:

- Anàlisi registre 6 (T_3 actualitza pàgina A):
 - Crear nova entrada T_3 a TT.
 - Crear nova entrada pàgina A a TPB.
- Anàlisi registre 7 (T_2 actualitza pàgina C):
 - $LSNP=7$ en pàgina C.
- Anàlisi registre 8 (T_2 confirmat):
 - Canviar estat de T_2 a confirmat en TT.

ARIES: Recuperació de l'exemple

3. **Fase Desfer:** En registre, la transacció activa T_3 no està confirmada ($LSN=6$), pel que cal desfer-la.
4. **Fase Refer:** Buscar el LSN mínim a la TPB: $\min(LSN)=2$. Es comença per registre 2:
 - $LSNP(B)=2 \rightarrow$ Entrada registre que actualitza B és 2:
 - $LSNP(B)=2$: Refer actualització registre $LSN=2$.
 - $LSNP(A)=6 \rightarrow$ Entrada registre que actualitza A és 6.
 - $LSNP(A)=6$: Refer actualització registre $LSN=6$.
 - $LSNP(C)=7 \rightarrow$ Entrada registre que actualitza C és 7:
 - $LSNP(C)=7$: Refer actualització registre $LSN=7$.

ARIES: Característiques

- **Independència de recuperació:** Algunes pàgines es poden recuperar independentment d'altres, pel que es permet executar transaccions durant la recuperació.
- **Punts d'emmagatzematge:** les transaccions poden registrar punts d'emmagatzematge i retrocedir fins aquests punts. Això és útil per a casos d'interbloqueigs.
- **Bloqueig fi:** L'algorisme es pot utilitzar amb algorismes de control de concurrència que permeten el bloqueig per tuples enloc de pàgines, augmentant concurrència.
- **ARIES:** L'algorisme de recuperació amb optimitzacions per a millorar concurrència, reduir càrrega del registre de sistema i reduir temps de recuperació.

En resum...

Cal saber restaurar el contingut d'una BD cas el sistema falli o hi hagi un desastre (inundació, sabotatge, etc.).

2 operacions bàsiques:

- Desfer – No Refer transaccions no confirmades
- Refer – No Refer transaccions confirmades

Mètodes recuperació:

- Modificació diferida. Algorisme Refer
- Modificació immediata. Algorismes Desfer i Refer
- Paginació a l'ombra
- Algorisme ARIES

Recuperació davant desastres: Recuperació emmagatzematge no volàtil.