

# **Gestió i Processament de Transaccions: recuperació**

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

**Oriol Ramos Terrades**

**Carles Sánchez Ramos**

*Departament de Ciències de la Computació*

# Recuperació

Si la BD falla, pot ser que SGBD no pugui garantir el compliment de les transaccions.

El SGBD és el responsable de garantir que totes les transaccions es facin i que les transaccions que actualitzin, esborrin o insereixin les dades ho facin de forma permanent a la BD (disc).

En cas d'errada:

- es busquen cap enrere les transaccions que no hagin acabat (sense `commit`).
- Cal reproduir transaccions confirmades (amb `commit`).

# Recuperació: errades

1. **Caiguda de sistema:** Error hardware, software, xarxa.
2. **Error transacció o de sistema:** Error de programació. Divisió per zero, *overflow*.
3. **Errors locals o condicions d'excepció:** Han de ser previstos pel programador. Dada no trobada, saldo insuficient.
4. **Concurrencia:** *deadlock* entre transaccions.
5. **Errada de disc** (*disk fault*).
6. **Problemes físics i catastròfics:** Errades d'alimentació, aire condicionat, sabotatge.

Errades 1 a 4: Errades típiques, cal recuperar, guardant historial i saber-lo recuperar. Errors 5 ,6: no tants freqüents, cal recuperar prioritàriament.

# Recuperabilitat

- Si una transacció falla, cal desfer els efectes fets per la transacció per assegurar la seva atomicitat i assegurar que tota transacció que depengui d'ella també s'anul·li (`rollback`).
- Planificacions acceptables segons la recuperabilitat:
  - **Planificació recuperable**
  - **Planificació sense cascada** (*cascadeless*)
  - **Planificació estricta**

# Planificació recuperable

Planificació amb transaccions confirmades que ja no es poden anul·lar.

PLANIFICACIÓ P		
t	T'	T
i	write_item(X)	
i+k		read_item(X)
j	commit()	
J+s		commit()

- Si tota transacció T de S no es confirma fins que les transaccions T' que han escrit un element que T llegeix s'han confirmat.

Operacions potencialment conflictives

Planificacions recuperables requereixen procés de recuperació complexa. Si es guarda informació suficient en el registre de sistema, es poden recuperar.

# Planificació recuperable: exemples

PLANIFICACIÓ P		
t	T1	T2
1	read_item(X)	
2		read_item(X)
3	write_item(X)	
4	read_item(Y)	
5		write_item(X)
6	write_item(Y)	
7	commit()	
8		commit()

*P* és una planificació recuperable. Les dues es confirmen.

PLANIFICACIÓ R		
t	T1	T2
1	read_item(X)	
2	write_item(X)	
3		read_item(Y)
4		write_item(X)
5	read_item(Y)	
6	rollback()	

*R* és una planificació recuperable.  $T_1$  s'avorta però no toca el valor de *Y* que llegeix  $T_2$ .

PLANIFICACIÓ S		
t	T1	T2
1	read_item(X)	
2		read_item(X)
3	write_item(X)	
4	read_item(Y)	
5		write_item(X)
6		commit()
7	write_item(Y)	
8	commit()	

*S* és una planificació recuperable, malgrat pèrdua d'actualització de *X* de  $T_2$  ( $w_1(X)$  entremig  $r_2(X) - w_2(X)$ ).

# Planificació recuperable: exemples

PLANIFICACIÓ $W$		
$t$	$T_1$	$T_2$
1	<code>read_item(X)</code>	
2	<code>write_item(X)</code>	
3		<code>read_item(X)</code>
4	<code>read_item(Y)</code>	
5		<code>write_item(Y)</code>
6		<code>commit()</code>
7	<code>rollback()</code>	

$W$  és una planificació **no recuperable**.

- $T_2$  llegeix  $X$  de  $T_1 \rightarrow r_2(X)$
- Es confirma  $T_2$  abans  $T_1 \rightarrow c_2$
- Si  $T_1$  es cancel·la ( $a_1$ ) després de  $c_2$ , el valor  $X$  que  $T_2$  ha llegit no és vàlid. En aquest cas,  $T_2$  s'hauria de cancel·lar després de ser confirmada (planificació no recuperable, no es pot anul·lar).

# Planificació recuperable: exemples

## PLANIFICACIÓ $S_d$

t	T1	T2
1	read_item(X)	
2	write_item(X)	
3		read_item(X)
4	read_item(y)	
5		write_item(Y)
6	write_item(Y)	
7	commit()	
8		commit()

## PLANIFICACIÓ $S_e$

t	T1	T2
1	read_item(X)	
2	write_item(X)	
3		read_item(X)
4	read_item(y)	
5		write_item(Y)
6	write_item(Y)	
7	rollback()	
8		rollback()

- $S_d$  recuperable doncs  $T_1$  es confirma primer ( $c_1$ ) i després  $T_2$  ( $c_2$ ).
- $S_e$  recuperable doncs s'anul·la  $T_1$  ( $a_1$ ) i després  $T_2$  ( $a_2$ ) → **Anul·lació en cascada**:  $T_2$  s'anul·la perquè  $T_1$  s'ha anul·lat.
- Anul·lació en cascada consumeix *overhead* si hi ha moltes transaccions involucrades. Cal distingir les transaccions que no estan involucrades en l'anul·lació en cascada → **Transacció sense cascada**



# Planificació sense cascada

Una transacció  $T$  és una **transacció sense cascada** (*cascadeless*) si la transacció sols llegeix elements escrits per transaccions confirmades o anul·lades.

$P: r_1(X), w_1(X), r_2(X), r_1(Y), w_2(X), w_1(Y), c_1, c_2;$

$P': r_1(X), w_1(X), r_1(Y), w_1(Y), c_1, r_2(X), w_2(X), c_2;$

$S: r_1(X), w_1(X), r_2(X), r_1(Y), w_2(X), w_1(Y), a_1, a_2;$

$S': r_1(X), w_1(X), r_1(Y), w_1(Y), a_1, r_2(X), w_2(X), a_2;$

$P', S'$  conté transacció  $T_2$  sense cascada.

# Planificació sense cascada

Una planificació  $P$  és una **planificació sense cascada** (*cascadeless*) si totes les seves transaccions són sense cascada.

$P' : r_1(X), w_1(X), r_1(Y), w_1(Y), c_1, r_2(X), w_2(X), c_2;$

$S' : r_1(X), w_1(X), r_1(Y), w_1(Y), a_1, r_2(X), w_2(X), a_2;$

$P', S'$  són planificacions sense cascada.

# Planificacions estrictes

Planificació recuperable, sense cascada... tercera opció: planificació estricta.

**Planificació estricta:** Planificació on totes les transaccions no poden llegir ni escriure  $X$  fins haver-se confirmat o cancel·lat la darrera transacció que va escriure  $X$ .

- Simplifica el procés de recuperació.
- Per desfer l'operació d'escriptura  $w(X)$  d'una transacció cancel·lada sols cal recuperar la imatge anterior (valor antic). Això sempre funciona en planificacions estrictes.
- Pot que no funcioni en planificacions de tipus *cascadeless*.

# Planificacions estrictes

Sense cascada no implica estricta. Exemple:

$S_j : w_1(X, 5), w_2(X, 8), a_1; \quad (X \text{ inicial}=9)$

Si  $T_1$  es cancel·la ( $a_1$ ) es restaura  $X$  a 9, però  $T_2$  ja ha canviat el valor a 8 → **INCORRECTE**

$S_j$  no és una planificació estricta, doncs  $T_2$  canvia  $X$  sense que  $T_1$  s'hagi confirmat o cancel·lat.

$S_j$  és *cascadeless* ( $a_1$  no obliga a anul·lar  $T_2$ , doncs no llegeix, sinó que escriu).

# Recuperabilitat

Planificacions classificades segons:

1. Recuperabilitat
2. Sense cascada (*cascadeless*)
3. Rigurositat (estrictes)

Condicions successivament estrictes:

- (2)  $\rightarrow$  (1): Planificacions sense cascada són recuperables.
- (3)  $\rightarrow$  (2)  $\rightarrow$  (1): Planificacions estrictes són sense cascada i per tant són recuperables.

# En resum...

- Els DBAs han de contemplar tot tipus de catàstrofes.
- Els SGBD han de permetre la recuperació dels sistemes i tornar a l'últim estat consistent.
- Les planificacions de les transaccions han de contemplar la recuperació.
- Segons el tipus de recuperació, les planificacions seran més o menys concurrents i requeriran més o menys overhead.