GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Vorlesung Datenbanksysteme 2

Übung – Recovery Manager Undo Redo Algorithmus

Aufgabe - Recovery

Das Datenbanksystem habe für dieses Schedule ***zwei LRU-Cache-Slots*** reserviert. Der Recovery Manager unterstützt ***Undo/Redo***.

a) Beschreiben Sie die Vorgänge bei der Ausführung des Schedules, indem Sie eine Tabelle vom Typ (1) verwenden.

b) Beschreiben Sie anschließend den Restart-Vorgang. Benutzen Sie hierzu eine Restart-Tabelle vom Typ (2).

Kennzeichnen Sie in den Tabellen einen ***dirty Cache-Slot*** durch ein * und einen ***unbestätigten Wert*** in der Datenbasis durch Einklammern z.B.(<WERT>).

T1	T2	T3
		READ(D) D:=D+15 WRITE(D)
A:=15 WRITE(A)		
	READ(C) C:=2*C WRITE(C)	
READ(B) B:=B+A WRITE(B) COMMIT		
		READ(A) A:=A+15 WRITE(A)
	READ(B) B:=B*C WRITE(B) COMMIT	
		READ(B) READ(E) E:=E+B WRITE(E)
☹ SYSTEMCRASH		

Ausführung - Tabelle vom Typ (1)

Tabelle Typ (1)													
Operation	**Logeintrag**	**Slot 1**		**Slot 2**		**Datenbasis**					**Listen**		
		X	V	X	V	A	B	C	D	E	Aktiv	Commit	Abort
(Init.)	[T0,A,5],[T0,B,10], [T0,C,1],[T0,D,0], [T0,E,1]	-	-	-	-	5	10	1	0	1	---	T0	---
$READ_3(D)$											**---**	**T0**	
$WRITE_3(D)$												**T0**	
$WRITE_1(A)$												**T0**	
$READ_2(C)$												**T0**	
$WRITE_2(C)$												**T0**	
$READ_1(B)$												**T0**	
$WRITE_1(B)$												**T0**	
$COMMIT_1$												**T0**	
$READ_3(A)$												**T0**	
$WRITE_3(A)$												**T0**	
$READ_2(B)$												**T0**	
$WRITE_2(B)$												**T0**	
$COMMIT_2$												**T0**	
$READ_3(B)$												**T0**	
$READ_3(E)$												**T0**	
$WRITE_3(E)$												**T0**	
Stand beim Crash		-	-	-	-								

Ausführung - Tabelle vom Typ (2)

Listen: Aktiv={T3}, Commit={T0,T1,T2}, Abort={}

Tabelle Typ (2)											
Logeintrag	**Slot 1**		**Slot 2**		**Datenbasis**					**Listen**	
	X	V	X	V	A	B	C	D	E	undone	redone
Vor Recovery	-	-	-	-	30	10	2	15	1	---	---
`[T3,E,51]`											
`[T2,B,50]`											
`[T3,A,30]`											
`[T1,B,25]`											
`[T2,C,2]`											
`[T1,A,15]`											
`[T3,D,15]`											
Nach Flush	-	-	-	-						---	---

Undo/Redo Algorithmus

RM-WRITE(t_i,x,v):

1. Füge T_i zur Aktiv-Liste hinzu, falls sie noch nicht dort ist.
2. Wenn *x* noch nicht im Cache ist, fetche *x*.
3. Füge [T_i,x,v] in die Log-Datei ein.
4. Schreibe *v* in den Cache Slot, der von *x* belegt ist.
5. Bestätige (Acknowledge) dem Scheduler die Verarbeitung des RM-WRITE.

Anmerkung zu Schritt 4:

Hier und in allen weiteren Fällen gilt, dass, wenn ein Wert in einen Cache Slot geschrieben wird, das dirty-bit gesetzt wird.

Undo/Redo Algorithmus

RM-READ(T_i,x):

1. Wenn *x* nicht im Cache ist, fetche *x.*
2. Gib den Wert von *x* an den Scheduler.

RM-COMMIT(T_i):

1. Füge T_i der Commit-Liste hinzu.
2. Bestätige dem Scheduler das Commitment von T_i.
3. Lösche T_i aus der Aktiv-Liste.

Undo/Redo Algorithmus

RM-ABORT(T_i):

1. Für jedes Datenobjekt *x,* das durch T_i verändert wurde
   - Wenn *x* nicht im Cache ist, belege einen Slot für *x.*
   - Schreibe das Before-Image von *x,* in Bezug auf T_i , in den Slot von *x*.

Anmerkung: Der Wert ist jetzt nur im Cache wiederhergestellt, Der CM schreibt diesen Wert erst beim Flushen in die stable Database.

2. Füge T_i der Abort-Liste hinzu.
3. Bestätige dem Scheduler die Durchführung des RM-ABORT.
4. Lösche T_i aus der Aktiv-Liste.

Undo/Redo Algorithmus

RESTART:

1. Lösche alle Cache Slots.

- Bei einem Systemfehler ist der Hauptspeicher betroffen, Werten im Cache kann also nicht vertraut werden.

2. Setzte redone = {} & undone = {}

Dies sind lokale Variablen der Restart-Prozedur, sie halten fest, für welche Datenobjekte der letzte committed Wert wiederhergestellt wurde (durch Undo oder Redo).

Undo/Redo Algorithmus

3. Beginne mit dem letzten Eintrag im Log und gehe rückwärts bis zum Anfang des Log.

Wiederhole die folgenden Schritte bis entweder ***redone* ∪ *undone* = *{alle Datenobjekte}*** oder es keine Einträge mehr im Log gibt.

Für alle Einträge [T_i,x,v] im Log, falls $x \notin$ redone ∪ undone, dann

- Falls *x* nicht im Cache ist, fetche *x.*
- wenn T_i in der Commit-Liste ist, kopiere *v* in den Slot von *x* und setzte redone = redone ∪ { *x* }.
  - Der letzte committed Wert von *x* ist jetzt nur im Cache wiederhergestellt.
- sonst (T_i in Abort- oder Aktiv-Liste aber nicht in der Commit-Liste) kopiere das Before-Image von *x* bezüglich T_i (im log), in den Slot von *x* und setzte undone = undone ∪ { *x* }.

Undo/Redo Algorithmus

4. Entferne jede Transaktion T_i die sowohl in der Commit-Liste als auch in der Aktiv-Liste ist, aus der Aktiv-Liste.
5. Bestätige dem Scheduler die Durchführung des RESTART. (Der Scheduler wartet, bis der Restart beendet ist)

Ausführung - Tabelle vom Typ (1)

Tabelle Typ (1)													
Operation	**Logeintrag**	**Slot 1**		**Slot 2**		**Datenbasis**					**Listen**		
		X	V	X	V	A	B	C	D	E	Aktiv	Commit	Abort
(Init.)	[T0,A,5],[T0,B,10], [T0,C,1],[T0,D,0], [T0,E,1]	-	-	-	-	5	10	1	0	1	---	T0	---
$READ_3(D)$		D	0	-	-	5	10	1	0	1	---	T0	
$WRITE_3(D)$	[T3,D,15]	D*	15	-	-	5	10	1	0	1	T3	T0	
$WRITE_1(A)$	[T1,A,15]	D*	15	A*	15	5	10	1	0	1	T3,T1	T0	
$READ_2(C)$		C	1	A*	15	5	10	1	<15>	1	T3,T1	T0	
$WRITE_2(C)$	[T2,C,2]	C*	2	A*	15	5	10	1	<15>	1	T3,T1,T2	T0	
$READ_1(B)$		C*	2	B	10	<15>	10	1	<15>	1	T3,T1,T2	T0	
$WRITE_1(B)$	[T1,B,25]	C*	2	B*	25	<15>	10	1	<15>	1	T3,T1,T2	T0	
$COMMIT_1$		C*	2	B*	25	15	10	1	<15>	1	T3,T2	T0,T1	
$READ_3(A)$		A	15	B*	25	15	10	<2>	<15>	1	T3,T2	T0,T1	
$WRITE_3(A)$	[T3,A,30]	A*	30	B*	25	15	10	<2>	<15>	1	T3,T2	T0,T1	
$READ_2(B)$		A*	30	B*	25	15	10	<2>	<15>	1	T3,T2	T0,T1	
$WRITE_2(B)$	[T2,B,50]	A*	30	B*	50	15	10	<2>	<15>	1	T3,T2	T0,T1	
$COMMIT_2$		A*	30	B*	50	15	10	2	<15>	1	T3	T0,T1,T2	
$READ_3(B)$		A*	30	B*	50	15	10	2	<15>	1	T3	T0,T1,T2	
$READ_3(E)$		E	1	B*	50	<30>	10	2	<15>	1	T3	T0,T1,T2	
$WRITE_3(E)$	[T3,E,51]	E*	51	B*	50	<30>	10	2	<15>	1	T3	T0,T1,T2	
Stand beim Crash		-	-	-	-	<30>	10	2	<15>	1			

Ausführung - Tabelle vom Typ (2)

Listen: Aktiv={T3}, Commit={T0,T1,T2}, Abort={}

Tabelle Typ (2)												
Logeintrag	**Slot 1**		**Slot 2**		**Datenbasis**					**Listen**		**Kommentar**
	X	V	X	V	A	B	C	D	E	undone	redone	
Vor Recovery	-	-	-	-	30	10	2	15	1	---	---	
[T3,E,51]	**E***	**1**	**-**	**-**	**30**	**10**	**2**	**15**	**1**	**E**	**---**	Wert von [T0,E,1]
[T2,B,50]	**E***	**1**	**B***	**50**	**30**	**10**	**2**	**15**	**1**	**E**	**B**	
[T3,A,30]	**A***	**15**	**B***	**50**	**30**	**10**	**2**	**15**	**1**	**E,A**	**B**	Wert von [T1,A,15]
[T1,B,25]	**A***	**15**	**B***	**50**	**30**	**10**	**2**	**15**	**1**	**E,A**	**B**	
[T2,C,2]	**A***	**15**	**C***	**2**	**30**	**50**	**2**	**15**	**1**	**E,A**	**B,C**	Wert von [T2,C,2]
[T1,A,15]	**A***	**15**	**C***	**2**	**30**	**50**	**2**	**15**	**1**	**E,A**	**B,C**	
[T3,D,15]	**D***	**0**	**C***	**2**	**15**	**50**	**2**	**15**	**1**	**E,A,D**	**B,C**	Wert von [T0,D,0]
Nach Flush	-	-	-	-	15	50	2	0	1	---	---	