6 Transaction Processing

The correctness principle

- Hvis transaction udføres alene på en konsistent database, så er databasen konsistent bagefter.

⇒ Transaction er atomic
Alt eller intet udføres

⇒ Hvis der er flere transactioner der udføres samtidigt, skal de styres for ikke at lave inkonsistent DB

Undo/redo log

- Forhindrer system chrash i at give inkonsistent DB

Regel) Log på disk før databaseændring på disk

#	action	T	A	AD	Log
1					<start T>
2	Read(A)	8	8	8	
3		16			
4	write(A)		16		<T, A, 8, 16>
5	Flush Log				
6	OUTPUT(A)		16	16	
7					<commit T_1>

<start T_1>
<T_1, A, 8, 16>
<start T_2>
<T_2, B, 8, 10>
<T_2 commit>
<T_1 commit>

Checkpointing
1) <chpt T_1, T_2> Flush
2) Flush dirty pages
3) <end chpt> Flush

Schedules

Sekvens af operationer fra forskellige transactioner
- Read(X)
- Write(X)

Eksempel

Konsistens : A=B

T_1 1 A=A+1 T_2 3 A=A·2
2 B=B+1 4 B=B·2

Problem schedule: 1 3 4 2 ⇒ inkonsistent

Serial schedule

T_1, T_2
T_2, T_1

Serializable schedules
- samme effekt som en eller anden serial schedule

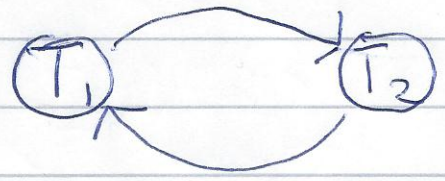
Conflict serializable schedule

Schedule er conflict sirializable hvis vi kan swappe operationer så vi kommer til en serial schedule uden at få en konflikt.

Konflikt: $r_1(X)\ w_2(X)$ $w_1(X)\ w_2(X)$

Bevis grafalgoritme til check

$r_1(X)\ r_2(X)\ w_1(X)\ w_2(X)$



Cykel = konflikt!

Databaser benytter locks. 2 phase locking:

1)- Får locks

2)- Frigiver locks