LECTURE 1 DBMS

Transaction Properties ACID

- Atomicity : either all of the op. are executed, or none are executed (commit, abort, undo)
- Consistency: a tran. must preserve the consistency of the database after execution
- Isolation: a tran. is protected from the effects of concurrently scheduling other trans. (single user mode)
- Durability: the eff. of a successfully completed trans. should persist even if a system crash occurs before all the changes have been written to the disk (Write-Ahead-Log ch. written to the log and then in the db)

Interleaved Executions - Example

ANOMALIES

- T1,T2 Read Only S.D → NO Conf.
- T1,T2 Read/Write diff data → NO C.
- T1,T2 R/W same data : order of ex is imp!
  - WR conflict : T2 R data prev. written by T1.
  - RW conflict : T2 W data prev. read by T1
  - WW conflict: T2 W data prev. written by T1

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
R(B)	
W(B)	

Dirty reads: occurs when a transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed. (WR)

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	

Unrepeatable read: occurs when two or more read operations of the same trans. read different values of the same variables (RW)

T1	T2
R(A)	
	R(A)
	W(A)
R(A)	
W(A)	

Overwriting uncommitted data: one tran. updates data which is not yet committed perm. to the database ⇒ trans. is rolledback ⇒ data item returned to prev. value. (rolledback trans, WW)

T1	T2
W(A)	
	W(A)
	W(B)
W(B)	

⇒ **Scheduling transactions** (R, W, Commit, Abort)

→ Serial and Non-Serial Schedules

1) serial: actions are not interleaved

T1	T2
	R(V)
	V = V + 50
	W(V), commit
R(V)	
R(S)	
etc	

2) non-serial: —"—

T1	T2
R(V)	
	W(V)
R(V)	
	R(V
etc	

LECTURE 2

Conflict Serializability

S1: T1	S1: T2	S2: T1	S2: T2
R(A)		R(A)	
A = A-100		A = A-100	
W(A)		W(A)	
R(B)			R(A)
B = B+200			A = A*0.2
W(B)			W(A)
	R(A)	R(B)	
	A = A*0.2	B = B+200	
	W(A)	W(B)	
	R(B)		R(B)
	B = B+300		B = B+300
	W(B)		W(B)

conf(S1) = {(R(T_1,A), W(T_2,A)),
- (W(T_1,A), R(T_2,A)),
- (W(T_1,A), W(T2,A)),
- (R(T_1,B), W(T_2,B))
- (W(T_1,B), R(T_2,B))
- (W(T1,B), W(T_2,B))

$\Rightarrow$ $\boxed{S1 \equiv_c S2}$

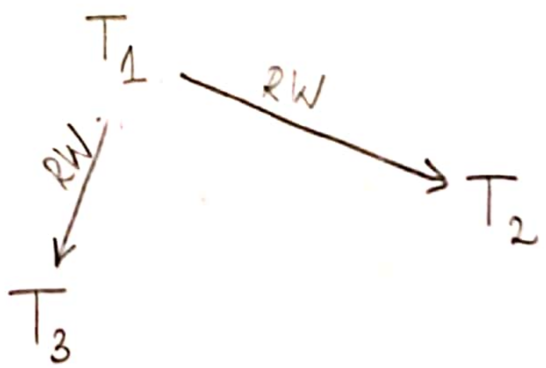
Precedence Graph

$\rightarrow$ arc for every $\boxed{WR, RW, WW}$

S is conf. serializable $\Leftrightarrow$ NO CYCLES

Ex. S_1 over $\{T_1, T_2, T_3\}$

T_1	T_2	T_3
$R_1(x)$		
$R_1(y)$		
	$R_2(y)$	
$W_1(x)$		
	~~$W_2(y)$~~	
	$W_2(y)$	
		$R_3(x)$
		$W_3(x)$



prec. graph.

→ serializable when no delete, ~~update~~ insert, only <u>UPDATE</u>

conflict seriz. ⟹ serializability (sufficient cond. but not necessary)

View Serializability $\boxed{S_1 \equiv_V S_2}$

1) T_i reads initial V in $S_1 \Rightarrow T_1$ also –״– in S_2.
2) T_i reads V written by T_j in $S_1 \Longrightarrow T_i$ –״– in S_2
3) T_i writes final value of V in $S_1 \Rightarrow T_1$ –״– S_2.

- each trans. performs same computation in S_1, S_2.
- S_1, S_2 produce the same final database state

Recoverable Schedules

Lock Based Concurrency Control

serializable, recoverable schedules

- lock : prevents trans. from accessing a data object while another trans. is accessing the object
- trans. protocol: rules, ex. lock before read, write.

Locks : → $\boxed{\text{SLock}}$ (shared or read lock)
↳ T can read the obj, cannot modify it
other T can have SL but not XL

→ $\boxed{\text{XLock}}$ (exclusive or write lock)
↳ T can both read and write object.
other T can't have SL or XL

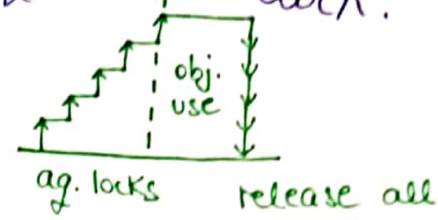
Lock Table → Keeping track of locks (granted)
→ 1) no of trans holding lock on obj
2) lock type
3) pointer to queue of lock req.

LECTURE 3

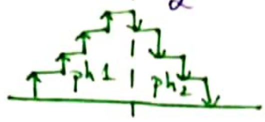
1) Locking protocols

- Strict Two-Phase Locking (A)
- Two-Phase Locking (B)

(A) - before a T can R/W, it must aq. S/X lock.
- locks released after ex.
- only serializable schedules



(B) - bef. a T can R/W, it must aq. S/X lock
- once a T releases a lock, it cannot req. other locks



2) DEADLOCKS

- prevention (Wait-die, Wound-die)
- detection (waits for graph, timeout mech)

cycle of T waiting for one another to release a locked resource

P(T): priority of T

Wait-die: if $P(T_1) > P(T_2) \Rightarrow T_1$ waits
otherwise Abort

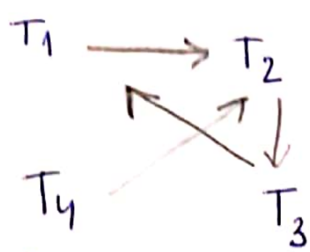
Wound-wait: $P(T_1) > P(T_2) \Rightarrow$ abort T_2
otherwise T_1 waits

Detection

a) waits-for graph.

arc from T_i to T_j if T_i is waiting for T_2 to release a lock

T_1	T_2	T_3	T_4
S(A)			
R(A)			
	X(B)		
	W(B)		
S(B)			
		S(C)	
		R(C)	
	X(C)		
		X(A)	X(B)



CYCLE ⇒ deadlock

b) timeout mechanism : if T has been waiting too long for a lock on an object, a deadlock is assumed to exist
$\Rightarrow$ T is terminated

3) The Phantom Problem

- Schedule is conflict serializable (ACYCLIC prec. gr.)
- if $\exists$ insert op $\Rightarrow$ conflict serial $\not\Rightarrow$ serializability

4) ISOLATION LEVELS : degree to which a trans. is exposed to the op. of other concurrently running trans.

$\boxed{\text{dirty writes}}$ → not allowed under any I.L.
↳ tr. T1 modifies an object prev. written by an ongoing tran. T2

A) READ UNCOMMITTED → T can read data modified by an ongoing T. (uncommitted data)
→ lowest degree of is.
→ No S locks

B) READ COMMITTED → T can only read committed data
→ obj read by T can be changed by another T while T in progress
→ T must aq. X lock → W → end
S lock → R → immed.

C) REPEATABLE READ → T can only read committed data
→ no obj read by T can be ch by anoth

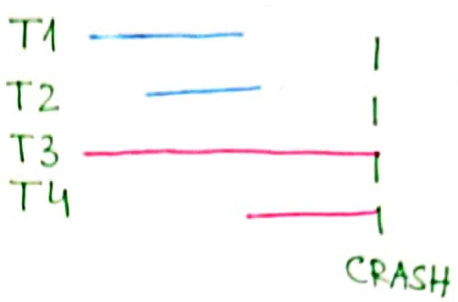
D) SERIALIZABLE → can only read committed data
highest degree → no obj read by T can be changed by another T while T is in prog.

LECTURE 4

Recovery Manager

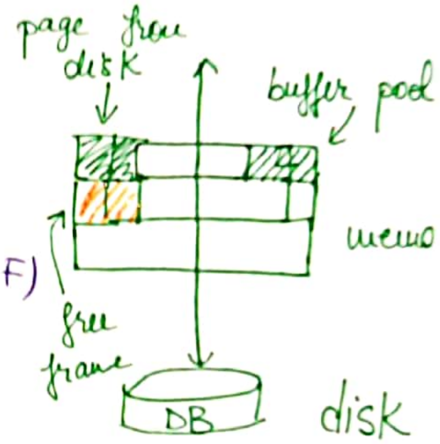
- atomicity: effects of uncommitted T are UNDONE
- durability: eff. of comm. T survive syst. crash

T1, T2 commit before crash
T3, T4 are undone



Buffer Manager

↳ pin_count (no. of current users)
↳ dirty (bool, F has been changed?)
↳ increment pin_count (pin page P in F)
↳ decrement pin_count (unpin —"—)



① pin_count = 0, dirty = off $\forall F \in BP$
L asks for page P the BM.
↓
checks if $\boxed{P \in BM}$ —Yes→ pin_count (F)++ (F contains P)
—No→ a) BM chooses FR for replacement which has pin_count = 0, ++
b) if dirty (FR) = on, BM writes FR to disk.
c) BM reads P in FR.

② BM returns address of BP frame that contains P to L.

Writing Objects

: options: steal / no steal, force / no force

*1) steal: T's changes can be written to disk before it commits
T2 needs page. → BM chooses F while T in prog.
$\boxed{\text{T2 steals frame from T}}$

2) no steal : T's changes cannot be written to the disk before it commits

3) force approach : T's changes are forced to disk when comm.

*4) no force app : —"— are not

ARIES – recovery alg, steal, no-force approach
– sys. restart after a crash : 3 phases :

1) analysis : det → active trans. at the time of crash
→ dirty pages (changes in BP whose ch. haven't been written to disk)

2) redo : reapply all changes (starting from certain record in the log) → bring DB before crash

3) undo : undo changes of uncomm. trans.

change to obj is first recorded in log (LR)

(Ex) • analysis : 1) active T at crash : T1, T3 → to be undone
2) commited : T2 → persist
3) maybe dirty pag : P_1, P_2, P_3
• redo : reapply all changes in order 1,2 ... 6.
• undo : undo changes of T1, T3 in reverse order (6,5,1)
$\underbrace{6,5}_{T_3},\underbrace{1}_{T_1}$

LSN	Log
1	up : T_1 $W(P_1)$
2	up : T_2 $W(P_2)$
3	T_2 comm.
4	T_2 end
5	up : T_3 $W(P_3)$
6	up : T_3 $W(P_2)$
	Crash, restart

Storage Media
• volatile storage
• non-volatile storage
• stable storage

The Log → up, abort, commit, end, ~~del~~
• log tail
• Log seq. Number

• Transaction table, Dirty Page table

LECTURE 5 — Crash Recovery

ARIES

Example:

prevLSN	TID	type	pageID	...
	T10	UP	P100	2
	T15	UP	P2	2
	T15	UP	P100	2
	T10	UP	P10	2
	T15	COM.		
	T10	UP	P11	2

1) Analysis: 1^{st} log: + T10 → TT.
+ P100 → DPT
2^{nd} log: + T15 → TT
+ P2 → DPT
4^{th} log: + P10 → DPT
T10 - active at crash
2) Redo: update not reapp.
3) Undo: undo update T10 + LSN ⇒ ToUndo

SEMINAR 3

1) Lost updates: 2T write same data
2) Dirty reads: T reads uncommitted data, changed by another ongoing trans.
3) Unrepeatable reads: row read by T_1 is changed by another T_2 while reading is in prog.
4) Phantoms: T_1 reads rows while T_2 gen. new row ⇒ T_1 reads again ⇒ another row

Solutions: Write locks: XL, doesn't allow other r/W
Read locks: allows R, not W

! Isolation level det whether read locks are ag. for read op, duration,

LECTURE 6 Security

SQL Injection

↳ user enters code concatenated with SQL statements

1) Changing user's authentication statement

SELECT .... WHERE user="a" AND pass="" OR 1=1=

Steganography, Cryptography → transposition
↘ substitution

ALGORITHM

1) data: disciplina baze de date, secret key: student

a)

	a	b	c	d	e	f	g	h	......	v	w	x	y	z
0	1	2	3	4	5	6	7	8	......	22	23	24	25	26

7 7 7

1) d i s ciplina baze de date student
4 9 19 . . . . 20 5 19 20 21 4 5 14 20

2) 4 09 19 03 09 .... 20 05 +
19 20 21 4 5 .... 19 20

23 02 13 07 14 .... 12 25 ← MOD 27

3) w b m g m .... l y → STRING

for decryption → SUBSTRACT

LECTURE 7

Relational Algebra

Select (σ)
projection (π)
join ($\otimes$)
cross prod ($R1 \times R2$)

set diff. ($R1 - R2$)
union ($R1 \cup R2$)
intersection ($R1 \cap R2$)

LECTURE 8

External Merge Sort: each pass read / process / w each page in the file

N - no of pages
P - no of passess
B - no of buffer pages

- no of passes: $\lceil \log_2 N \rceil + 1$
- total cost: $2 * N * (\lceil \log_2 N \rceil + 1)$ I/Os
  $2 * N * P$
- pass 0: B buffer pages $\longrightarrow \lceil \frac{N}{B} \rceil$
- pass n: B-n

(Ex) $B = 5$, $N = 108$

! pass 0: $\lceil \frac{N}{B} \rceil = \lceil \frac{108}{5} \rceil = 22$ runs
pass 1: $B - 1 = 5 - 1 = 4$ pages for input, 1 for output
↳ 20 pages long (4×5)

cost: total cost: $2 \cdot N \cdot (\lceil \log_{B-1} \lceil \frac{N}{B} \rceil \rceil + 1)$ I/Os
no of passes: $\lceil \log_{B-1} \lceil N/B \rceil \rceil + 1$

Simple Two Way Merge Sort

$\rightarrow$ 3 buffer pages
$\rightarrow$ 1 page run
$\rightarrow$ 2 I/O op per page, per pass

no of passes: $\lceil \log_2 N \rceil + 1$
total cost: $2 * N * P$
$2 * N * (\lceil \log_2 N \rceil + 1)$ I/Os

N - no of pages
P - no of passes
pass 0 $\rightarrow$ N runs

Sort Merge Join

: equality join $E \otimes_{i=j} S$

$E \rightarrow M$ pages
$S \rightarrow N$ pages

cost: sorting E: cost $O(M \log M)$
Sorting S: cost $O(N \log N)$
merging: $M+N$ (scanned 1)
$M*N$ (worst case) same value in join column

(Ex) Exams $\otimes_{\text{Exams.SID} = \text{Students.SID}}$ Students

$B = 300$ (buffer pages)

Sort Exams: 2 passes $\Rightarrow$ cost $2*2*1000 = 4000$ I/Os
Sort Students: 2 passes $\Rightarrow$ cost $2*2*500 = 2000$ I/Os
merging phase: cost: $1000 + 500 = 1500$
total cost: $4000 + 2000 + 1500 = 7500$ I/Os

- $E \rightarrow M$ pages, P_E records/page $\longrightarrow$ $M = 1000$, $P_E = 100$
$S \rightarrow N$, P_S $\longrightarrow$ $N = 500$, $P_S = 80$

Scanned with CamScanner

LECTURE 9

Hash Join

cost: partitioning: cost: $2*(M+N)$ I/Os
probing: cost: $M+N$ I/Os
$\Rightarrow$ total cost: $3*(M+N)$ I/Os

$E \otimes_{i=j} S$

$E \rightarrow M$ pages, p_E records/page
$S \rightarrow N$ pages, p_S records/page

3) b, c
4) b
c → vertical fragm.

Selection

→ cost: no index → M I/Os
no index, sorted → $O(\log_2 M)$

5) b, c,
6) c
7) b

Projection

→ 1) cost: scan E: M I/Os
write temp rel: T I/Os
2) cost: $O(T \log T)$
3) cost: T

T – no of col.

⇒ total cost $O(M \log M)$

1. Page oriented nested loops, Block nested loops join

R, S, R → 10000 records, 10 R records/page ⇒ 1000 pages
S → 2000 records, 10 S records/page ⇒ 200 pages
B → 52 buffer pages
S – outer relation

cost of: SELECT * FROM R INNER JOIN ON R.a = S.b

~~R → 100 re~~

200

- page oriented nested loops join
  - 200 + 200 * 1000 = 200200 I/Os
    outer rel
- block nested loops join = 52 − 1 − 1 → outer buffer p.; → inner
  - block size: 50 ⇒ $\left[\frac{200}{50}\right] = 4$ S blocks
  - 200 + 4 * 1000 = 4200 I/Os

- 200 buffer pages, sort R with external merge sort.

  R – 10000 records ; 10 R/ page ⟶ 1000 pages

  $2 * 1000 * 2 = 4000$ I/Os

  $2 * N * \left(\left\lceil \log_{B-1} \left\lceil \frac{N}{B} \right\rceil \right\rceil + 1 \right)$ I/Os

  B = 200 buffer
  N = 1000 pages

- simple nested loops join (tuple oriented)

  R – 1000 rec ; 10 R/page ⟹ 1000 pages
  S = 2000 rec ; 10 S/ page ⟶ 200 pages
  S - outer rel.
  td time to R/W page from /to disk
  ts time to ship a page

  - $200\, t_d + 2000 * 1000 (t_d + t_s) = 200\, t_d + 2000000 (t_d + t_s)$