

CIS 560 – Database System Concepts

Lecture 18

Transaction Management: Recovery

October 16, 2013

Credits for slides: Chang, Ullman, Whitehead.

Copyright: Caragea, 2013.

Announcements

- Evaluation forms due tonight
- HW5 - SQL/JDBC assignment due Friday by midnight.
- HW6 will be posted on Friday

Two Perspectives of DBMS

- User perspective
 - How to use a database system
 - Conceptual data modeling, database schema design, the SQL query language, transactions from the SQL programmer point of view, embedded SQL.
- System perspective
 - How database systems work
 - Transaction management, data storage and indexing, (relational algebra), query optimization and processing.

Outline

Today:

- Disks 13.2
- Buffer management 15.7
- Issues and models for resilient operations 17.1
- Undo logging 17.2

Next:

- Redo logging 17.3
- Redo/undo 17.4
- Serial and serializable schedules 18.1
- Conflict serializability 18.2
- Locks 18.3

Transaction Definition

- **A transaction** = one or more operations, which reflect a single real-world transition
 - Happens completely or not at all
- Examples
 - Transfer money between accounts
 - Rent a movie; return a rented movie
 - Purchase a group of products
 - Register for a class (either waitlisted or allocated)

5

Transaction Properties ACID

- **Atomic**
 - State shows either all the effects of a transaction, or none of them (commits or aborts)
- **Consistent**
 - A transaction moves from a state where integrity holds, to another where integrity holds
- **Isolated**
 - Effect of transactions is the effect is as if each transaction executes in isolation of the others
- **Durable**
 - Once a transaction has committed, its effects remain in the database (i.e., write data to disk)

6

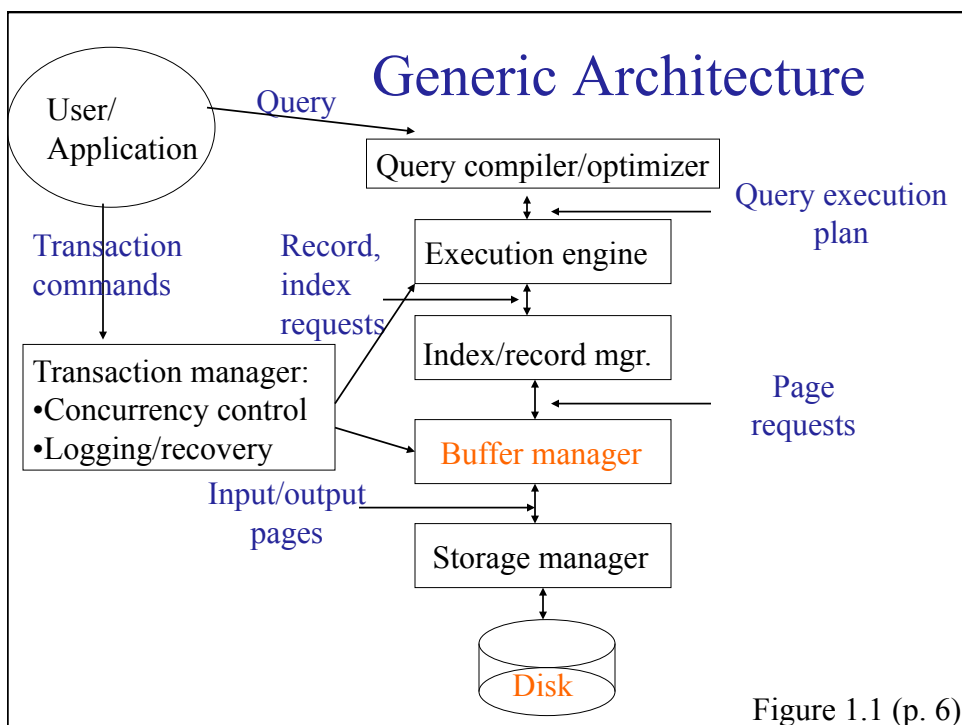
Transaction Management

Two parts:

- Recovery from crashes: ACID
 - Taking action to restore the DB to a consistent state
- Concurrency control: ACID
 - Making sure simultaneous transactions don't interfere with one another

Both operate on the **buffer pool**

7



The Mechanics of Disk

Mechanical characteristics:

- Rotation speed (5400RPM)
- Number of platters (1-30)
- Number of tracks (≤ 10000)
- Number of bytes/track (10^5)

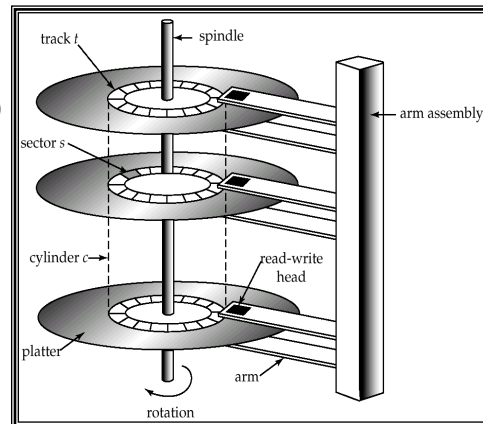
Unit of read or write:

disk block

Once in memory:

page

Typically: 4k or 8k or 16k



Disk Access Characteristics

- **Disk latency** = time between when command is issued and when data is in memory
- Disk latency = seek time + rotational latency + transfer time
 - Seek time = time for the head to reach cylinder
 - 10ms – 40ms
 - Rotational latency = time for the sector to rotate
 - Rotation time = 10ms
 - Average latency = 5ms
 - Transfer time = typically 40MB/s
 - Disks read/write one block at a time

Large gap between disk I/O and memory → Buffer pool

Design Question

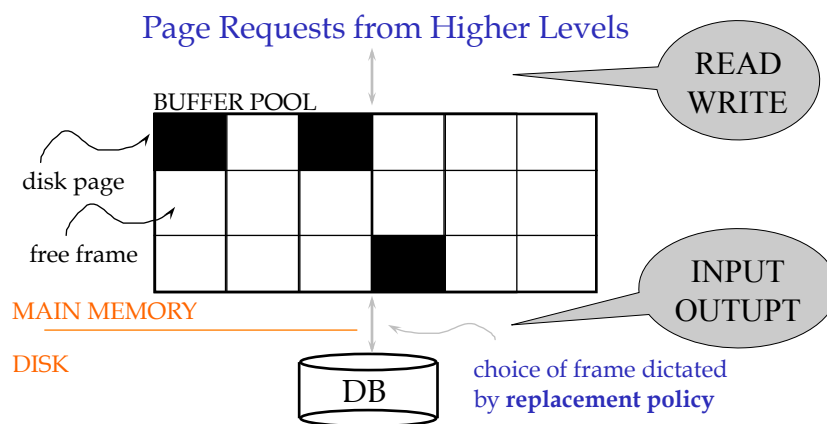
- Consider the following query:

```
SELECT    S1.temp, S2.pressure
FROM      TempSensor S1, PressureSensor S2
WHERE     S1.location = S2.location
AND       S1.time = S2.time
```

- How can the DBMS execute this query given
 - 1 GB of memory
 - 100 GB TempSensor and 10 GB PressureSensor

11

Buffer Management in a DBMS



- Data must be in RAM for DBMS to operate on it!
- Table of <frame#, pageid> pairs is maintained

12

Buffer Manager

DBMS build their own buffer manager and don't rely on the OS

- Reason 1: correctness
 - DBMS needs fine grained control for transactions
 - Needs to force pages to disk for recovery purposes
- Reason 2: performance
 - DBMS may be able to anticipate access patterns
 - Hence, may also be able to perform prefetching
 - May select better page replacement policy

13

Transaction Management and the Buffer Manager

The transaction manager operates on the buffer pool

- **Recovery**: 'log-file write-ahead', then careful policy about which pages to force to disk
- **Concurrency control**: locks at the page level, multiversion concurrency control

14

Recovery

Type of Crash	Prevention
Wrong data entry	Constraints and Data cleaning
Disk crashes	Redundancy: e.g. RAID, archive
Fire, theft ...	Remote backups
System failures: e.g. power	DATABASE RECOVERY

15

Problem Illustration

Client 1:

START TRANSACTION**INSERT INTO** SmallProduct(name, price)**SELECT** pname, price**FROM** Product**WHERE** price <= 0.99

crash

DELETE Product**WHERE** price <=0.99**COMMIT**

16

System Failures

- Each transaction has an internal state
- When system crashes, internal state is lost
 - Don't know which parts executed and which didn't
 - Need ability to undo and redo
- **Remedy: use a log**
 - File that records every single action of each transaction
 - After a crash, transaction manager reads the log and finds out exactly what the transactions did or did not

17

Transactions

- Assumption: the database is composed of *elements*
 - Usually 1 element = 1 block
 - Can be smaller (=1 record) or larger (=1 relation)
- Assumption: each transaction reads/writes some elements

18

Primitive Operations of Transactions

- READ(X,t)
 - copy element X to transaction local variable t
- WRITE(X,t)
 - copy transaction local variable t to element X
- INPUT(X)
 - read element X to memory buffer
- OUTPUT(X)
 - write element X to disk

19

Example

```
START TRANSACTION
READ(A,t);
t := t*2;
WRITE(A,t);
READ(B,t);
t := t*2;
WRITE(B,t)
COMMIT;
```

Atomicity:
BOTH A and B
are multiplied by 2

20

READ(A,t); t := t*2; WRITE(A,t);
READ(B,t); t := t*2; WRITE(B,t)

Action	Transaction	Buffer pool		Disk	
	t	Mem A	Mem B	Disk A	Disk B
INPUT(A)		8		8	8
READ(A,t)	8	8		8	8
t:=t*2	16	8		8	8
WRITE(A,t)	16	16		8	8
INPUT(B)	16	16	8	8	8
READ(B,t)	8	16	8	8	8
t:=t*2	16	16	8	8	8
WRITE(B,t)	16	16	16	8	8
OUTPUT(A)	16	16	16	16	8
OUTPUT(B)	16	16	16	16	16

21

Action	t	Mem A	Mem B	Disk A	Disk B
INPUT(A)		8		8	8
READ(A,t)	8	8		8	8
t:=t*2	16	8		8	8
WRITE(A,t)	16	16		8	8
INPUT(B)	16	16	8	8	8
READ(B,t)	8	16	8	8	8
t:=t*2	16	16	8	8	8
WRITE(B,t)	16	16	16	8	8
OUTPUT(A)	16	16	16	16	8
OUTPUT(B)	16	16	16	16	16


 Crash !

Crash occurs after OUTPUT(A), before OUTPUT(B)
We lose atomicity

22

The Log

- An append-only file containing log records
- Multiple transactions run concurrently, log records are interleaved
- After a system crash, use log to:
 - Redo some transaction that didn't commit
 - Undo other transactions that didn't commit
- Three kinds of logs: undo, redo, undo/redo

23

Undo Logging

Log records

- <START T>
 - transaction T has begun
- <COMMIT T>
 - T has committed
- <ABORT T>
 - T has aborted
- <T,X,v>
 - T has updated element X, and its old value was v

24

Action	T	Mem A	Mem B	Disk A	Disk B	Log
						<START T>
INPUT(A)		8		8	8	
READ(A,t)	8	8		8	8	
$t:=t*2$	16	8		8	8	
WRITE(A,t)	16	16		8	8	<T,A,8>
INPUT(B)	16	16	8	8	8	
READ(B,t)	8	16	8	8	8	
$t:=t*2$	16	16	8	8	8	
WRITE(B,t)	16	16	16	8	8	<T,B,8>
OUTPUT(A)	16	16	16	16	8	
OUTPUT(B)	16	16	16	16	16	
COMMIT						<COMMIT T>

25

Action	T	Mem A	Mem B	Disk A	Disk B	Log
						<START T>
INPUT(A)		8		8	8	
READ(A,t)	8	8		8	8	
$t:=t*2$	16	8		8	8	
WRITE(A,t)	16	16		8	8	<T,A,8>
INPUT(B)	16	16	8	8	8	
READ(B,t)	8	16	8	8	8	
$t:=t*2$	16	16	8	8	8	
WRITE(B,t)	16	16	16	8	8	<T,B,8>
OUTPUT(A)	16	16	16	16	8	
OUTPUT(B)	16	16	16	16	16	
COMMIT						<COMMIT T>



 Crash !

26

WHAT DO WE DO ?

Action	T	Mem A	Mem B	Disk A	Disk B	Log
						<START T>
INPUT(A)		8		8	8	
READ(A,t)	8	8		8	8	
t:=t*2	16	8		8	8	
WRITE(A,t)	16	16		8	8	<T,A,8>
INPUT(B)	16	16	8	8	8	
READ(B,t)	8	16	8	8	8	
t:=t*2	16	16	8	8	8	
WRITE(B,t)	16	16	16	8	8	<T,B,8>
OUTPUT(A)	16	16	16	16	8	
OUTPUT(B)	16	16	16	16	16	
COMMIT						<COMMIT T>

WHAT DO WE DO ?

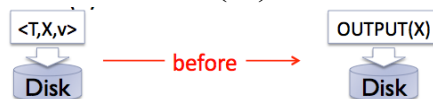


After Crash

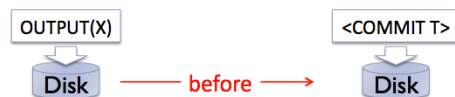
- In the first example:
 - We UNDO both changes: A=8, B=8
 - The transaction is atomic, since none of its actions has been executed
- In the second example
 - We don't undo anything
 - The transaction is atomic, since both its actions have been executed

Undo-Logging Rules

U1: If T modifies X, then $\langle T, X, v \rangle$ must be written to disk before OUTPUT(X)



U2: If T commits, then OUTPUT(X) must be written to disk before $\langle \text{COMMIT } T \rangle$



- Hence: OUTPUTs are done *early*, before the transaction commits

29

Action	T	Mem A	Mem B	Disk A	Disk B	Log
						$\langle \text{START } T \rangle$
INPUT(A)		8		8	8	
READ(A,t)	8	8		8	8	
$t := t * 2$	16	8		8	8	
WRITE(A,t)	16	16		8	8	$\langle T, A, 8 \rangle$
INPUT(B)	16	16	8	8	8	
READ(B,t)	8	16	8	8	8	
$t := t * 2$	16	16	8	8	8	
WRITE(B,t)	16	16	16	8	8	$\langle T, B, 8 \rangle$
OUTPUT(A)	16	16	16	16	8	
OUTPUT(B)	16	16	16	16	16	
COMMIT						$\langle \text{COMMIT } T \rangle$

30

Example

Given this undo log, when can each data item be output to disk?

- A: after 2
- B: after 3
- C: after 5, before 12
- D: after 7
- E: after 8, before 12
- F: after 10
- G: after 11

1	<START T1>
2	<T1, A, a>
3	<T1, B, b>
4	<START T2>
5	<T2, C, c>
6	<START T3>
7	<T3, D, d>
8	<T2, E, e>
9	<START T4>
10	<T4, F, f>
11	<T3, G, g>
12	<COMMIT T2>

1