CS411
Database Systems

15: Final Review

Kazuhiro Minami

Final exam

7:00pm-10:00pm on Thursday, December 17
1404 Siebel Center

- **Bring UIUC photoid, pen, pencil, and eraser**
- **Closed book, no notes**
- **Don't cheat**

Final exam (Subject to changes)

- 10 - 15 True-False questions
- Data Storage
- Indexes
- Query execution
- Query optimization
- Logging & Recovery
- Concurrency control

Concurrency Control

Concurrency Control – Basic concepts

- What is a transaction?
- Which actions do we consider in a transaction?
- How to represent a transaction?
- What is a schedule?
- What is the goal of concurrency control?
- What is a serial schedule?
- What is a serializable schedule?
- What is a conflict-serializable schedule?
- What are conflicting swaps?
- How to determine whether a schedule is conflict-serializable?

How to draw a precedence graph? Why the precedence-graph test works?

- $r_1(A); r_2(A);r_3(B);w_1(A);r_2(C);r_2(B);w_2(B);w_1(C)$

- $r_1(A); w_1(B);r_2(B);w_2(C);r_3(C);w_3(A);$

Enforcing Serializability by Locks

- What is a lock?
- What is a lock table? What kind of information is stored there?
- What is the consistency of transactions?
- What is the legality of transactions?
- What is the notations for actions of locking and unlocking?
- What is the job of the locking scheduler?
- What is the two-phase locking (2PL) condition? What type of serializable schedules are produced with this approach?

Two-phase locked schedule

- T_1: $r_1(A);w_1(B);$
- T_2: $r_2(A);w_2(A);w_2(B);$

- Q1: Make T_1 consistent and 2PL by adding lock and unlock actions
- Q2: Do the same for T_2
- Q3: Give an example of legal schedule of the above modified T_1 and T_2

Enforcing Serializability with Timestamps

- What is a timestamp?
- When is a timestamp assigned to a transaction?
- What's the notation for transaction T's timestamp?
- What information do we maintain for each database element X?
- What type of serializable schedules are produced with this approach?
- How the timestamp-based approach solve the problem of "dirty" read?

Transaction T is aborted if

1. T tries to read data written by transaction U, which started later than T
   - Q: Why?
   - Q: How to detect this situation?

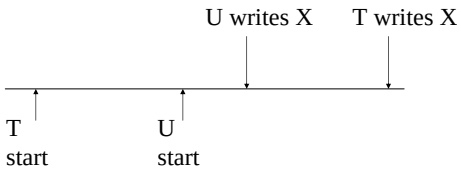
1. T tries to write data to database element X, which is already read by transaction U, which started later than T
   - Q: Why?
   - Q: How to detect this situation?

Prevention of Dirty Read

- How to prevent transaction T from reading data written by uncommitted transaction U?

Thomas Write Rule

- Why can we skip transaction T's write on element X, which is already modified by transaction U?
- What if there is another transaction V starting after T but before U?
- What if V starts after U?



Concurrency Control by Timestamps

- Tell me what happens as each executes
  - st_1; st_2; $r_1(A)$; $r_2(B)$; $w_2(A)$; $w_1(B)$

T1	T2	A	B
100	200	RT=0 WT=0	RT=0 WT=0
$r_1(A)$		RT=100	
	$r_2(B)$		RT=200
	$w_2(A)$	WT=200	
$w_1(B)$			

Write too late; need to roll back.

Concurrency Control by Timestamps

- Tell me what happens as each executes
  - st_1; $r_1(A)$; st_2; $w_2(B)$; $r_2(A)$; $w_1(B)$

T1	T2	A	B
100	200	RT=0 WT=0	RT=0 WT=0
$r_1(A)$		RT=100	
	$w_2(B)$		WT=200
	$r_2(A)$	RT=200	
$w_1(B)$			

OK, but needs to wait until T2 commits

Concurrency Control by Validation

Concurrency Control by Validation

- Another type of optimistic concurrency control
- Maintain a record of what active transactions are doing
- Just before a transaction starts to write, it goes through a “validation phase”
- If a there is a risk of physically unrealizable behavior, the transaction is rolled back

Validation-based Scheduler

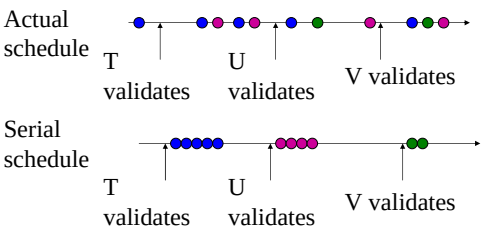
- Keep track of each transaction T's
  - Read set RS(T): the set of elements T read
  - Write set WS(T): the set of elements T write
- Execute transactions in three phases:
  1. *Read*. T reads all the elements in RS(T)
  2. *Validate*. Validate T by comparing its RS(T) an WS(T) with those in other transactions. If the validation fails, T is rolled back
  3. *Write*. T writes its values for the elements in WS(T)

Scheduler Maintains Information Sets

- START: the set of transactions that have started, but not yet completed validation. For each T, maintain (T, START(T))
- VAL: the set of transactions that have been validated, but not yet finished. For each T, maintain (T, START(T), VAL(T))
- FIN: the set of transaction that have completed. For each T, maintain (T, START(T), VAL(T), FIN(T))

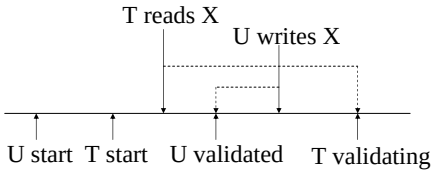
Assumed Serial Schedule for Validation

- We may think of each transaction that successfully validates as executing at the moment that it validates



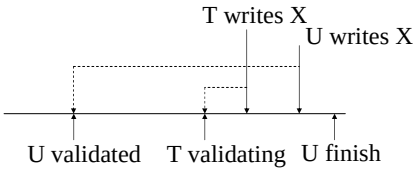
Potential Violation of the Serial Order

- Transactions T and U such that
  - U has validated
  - START(T) < FIN(U)
  - RS(T) ∩ WS(U) is not empty



Another Potential Violation of the Serial Order

- Two transactions T and U such that
  - U is in VAL
  - VAL(T) < FIN(U)
  - WS(T) ∩ WS(U) is not empty



Validation Rules

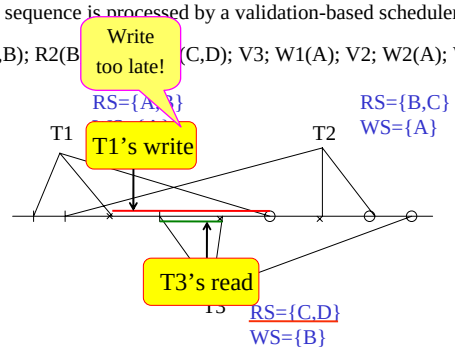
To validate a transaction T,

1. Check that RS(T) ∩ WS(U) is an empty set for any *validated* U and START(T) < FIN(U)
2. Check that WS(T) ∩ WS(U) is an empty set for any *validated* U that did not finish before T validated, i.e., if VAL(T) < FIN(U)

Example Problem

In the following sequence of events, tell what happens when each sequence is processed by a validation-based scheduler.

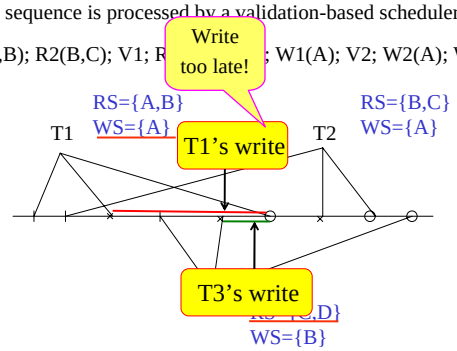
R1(A,B); R2(B (C,D); V3; W1(A); V2; W2(A); W3(B);



Example Problem

In the following sequence of events, tell what happens when each sequence is processed by a validation-based scheduler.

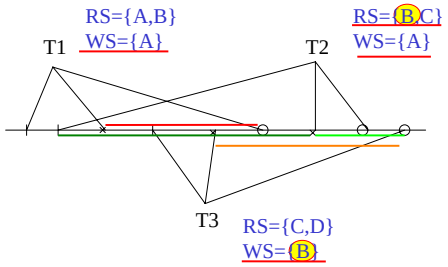
R1(A,B); R2(B,C); V1; R ; W1(A); V2; W2(A); W3(B);



Example Problem

In the following sequence of events, tell what happens when each sequence is processed by a validation-based scheduler.

R1(A,B); R2(B,C); V1; R3(C,D); V3; W1(A); V2; W2(A); W3(B);



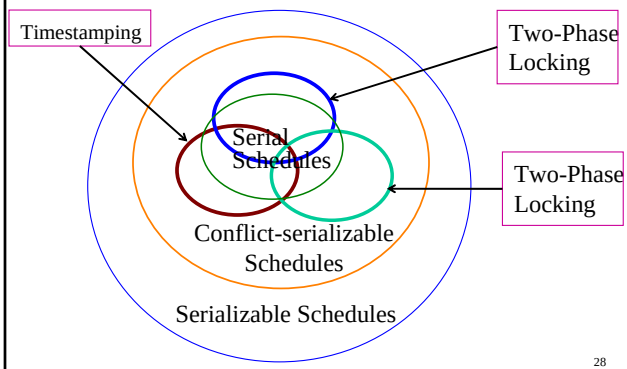
Comparison of Three Mechanisms

- Storage utilization
  - Locks: space in the lock table is proportional to the number of database elements locked
  - Timestamps: Read and write times for recently accessed database elements
  - Validation: timestamps and read/write sets for each active transaction, plus a few more transactions that finished after some currently active transaction began

Comparison of Three Mechanisms

- Delay
  - Locking delays transactions but avoids rollbacks, even when interaction is high
  - If interference is low, neither timestamps nor validation will cause many transactions
  - When a rollback is necessary, timestamps catch some problems earlier than validation

Summary



Storage: T/F Questions

- Secondary storage is volatile
- Spanned records refer to records that are longer than a single block and therefore are broken into fragments
- When we store multiple fixed-length records in a block, we do not need to have a record header for each record.
- A block header always contains pointers to each record in the block
- When we insert a new record into a relation, we sometimes need to create an overflow block even if records in that relation are not sorted.
- When a client requests a record that contains a BLOB, the database server returns the entire record at a time

Index: T/F Questions

- We can only use a dense index if the data file is sorted by the search key.
- When we add a second level of index, the second-level index must be dense.
- B-trees sometimes need overflow blocks.
- B-trees can be used to perform ranged queries.
- Extensible hash tables sometimes have overflow blocks.
- In a linear hash table, each data block has the "nub" indicating how many bits of the hash function's sequence is used.

Query execution

- Cost parameters
- Sort methods
- Hash methods
- Index method
- 1 phase vs. 2 phase
- Pipelining vs. materialization
- Estimation

How to do joins

Know the major ways to do joins

- hash join
- simple-sort-based join
- sort join (merge join)
- indexed join
- nested loop join

and when to pick one over the other

Make sure to understand how each algorithm works, its required memory, and I/O cost

True/False questions

- Duplicate elimination is a physical operator.
- The cost of scanning a unclustered relation R is B(R)
- A selection operator can be always executed with a one-pass algorithm.
- We can sort any number of blocks with 2-way merge sort using memory buffer of size M.
- When we sort two relations R and S with sort-merge-join, its total cost would be 5B(R) + 5B(S).
- Two-pass hash-based join algorithm has a size requirement only on the smaller of the two input relations

Query Optimization

- Do you know how to apply algebraic laws to get a better logical plan?
- Do you know how to perform a cost-based optimization on a logical plan?
- Do you know how to convert a logical plan to a physical plan?
- Do you know how to estimate the size of an intermediate operation?

Logging & Recovery

- What's the correctness principle?
- What are primitive four operations of transactions?
- How does undo logging work?
  - Log records
  - Undo-logging rules
  - Recovery procedure
- How does redo logging work?
- How does undo/redo logging work?
- What is a checkpoint?
- What is a nonquiescent checkpoint?
- Do you know when <END CKPT> is added in each method?
- Do you know recovery procedures with a checkpointed log?
- What are advantages and disadvantages of each method?