CS437/537
Review Session for Final Exam

Ennan Zhai
ennan.zhai@yale.edu
Final Exam

- Location: Our classroom
- NOT open-book
- Time: 9-10:15am Thu (Dec 8)
- Final is 30% of your total grade
Final Exam Structure

- Module 10: Physical Storage Systems
- Module 11: Data Storage Structures
- Module 12: Indexing
- Module 15: Transactions
- Module 16: Intro to Recovery System
- Module 17: Concurrency Control
  (up to Multiversion 2-phase locking)
- Module 21: Distributed Databases
  (up to Two phase commit)
Final Exam Structure

- Three parts (25, 40, and 35 points)
  - 7 short questions
  - 20 multiple choice (2x20 = 40 points)
  - 7 longer questions
Final Exam Structure

- Three parts (25, 40, and 35 points)
  - 7 short questions
  - 20 multiple choice (2x20 = 40 points)
  - 7 longer questions

- All the knowledge is based on lecture slides
- Ignore the slides after “End of Module”
Short Questions

• One of 7 short questions is related to one of the papers (CryptDB and DJoin) assigned in hwk5

• No concept and definition question

• Short questions mainly test “difference” and “why”
Short Questions

• One of 7 short questions is related to one of the papers (CryptDB and DJoin) assigned in hwk5

Examples:
- Which one of the papers in our homework is about a file system (not a database system)?
- Which one is a pure key-value DB?
- BigTable uses B+tree or B-tree?
Short Question Examples

- What’s the difference between B+tree and B-tree?
- Why we need concurrency control?
- Why we need multilevel index? Disadvantage?
- Why we have deadlock?
- Difference between deadlock and starvation?
- Why 2-phase locking is bad?
- What’s the potential problem in homogeneous distributed databases?
- Advantages and disadvantages of replications
Deadlock Conditions

- Mutual exclusion
  - Only one transaction at a time can use a resource

- Hold and wait
  - A transaction holding one resource needs to wait to acquire additional resources held by other transaction

- No preemption
  - A resource cannot be forced to release

- Circular wait (loop)
Why two-phase locking is bad?

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WL(A)</td>
<td>WL(A) Not granted</td>
</tr>
<tr>
<td>R(A)</td>
<td></td>
</tr>
<tr>
<td>W(A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WL(B)</td>
</tr>
<tr>
<td></td>
<td>R(B)</td>
</tr>
<tr>
<td></td>
<td>W(B)</td>
</tr>
<tr>
<td></td>
<td>UL(A)</td>
</tr>
<tr>
<td></td>
<td>UL(B)</td>
</tr>
<tr>
<td></td>
<td>Commit</td>
</tr>
<tr>
<td>Commit</td>
<td>Commit</td>
</tr>
</tbody>
</table>
Multiple Choice

• Only **ONE** choice is correct for each question

• All the questions and choices are only based on our lecture slides
Multiple Choice

- Only **ONE** choice is correct for each question

- All the questions and choices are only based on our lecture slides

If you are not sure which choice is correct, randomly choose one, say C.
Example

Which one of the following operations does not belong to the first phase in two-phase commit of distributed DB?
Which one of the following operations does not belong to the first phase in two-phase commit of distributed DB?

a. Coordinator $C_i$ asks all the participants to prepare to commit transaction $T_i$

b. Upon receiving message, transaction manager at site determines if it can commit the transaction
c. Coordinator receives a $<\text{ready } T>$ message from all the sites
d. Each manager adds a $<\text{ready } T>$ to the log
Example

Which one of the following operations does not belong to the first phase in two-phase commit of distributed DB?

a. Coordinator $C_i$ asks all the participants to prepare to commit transaction $T_i$
b. Upon receiving message, transaction manager at site determines if it can commit the transaction
c. Coordinator receives a $\langle\text{ready} \ T\rangle$ message from all the sites
d. Each manager adds a $\langle\text{ready} \ T\rangle$ to the log
Longer Questions

- Mainly test how to use knowledge to answer questions
Example

A partial schedule for data items for transactions with timestamps 1, 2, 3, 4, and 5

Q: Which transactions will be rolled back, in order to ensure serializability?

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>read (X)</td>
<td>read (Y)</td>
<td></td>
<td>write (Y)</td>
<td></td>
<td>read (X)</td>
</tr>
<tr>
<td>read (X)</td>
<td></td>
<td>read (Z)</td>
<td>write (Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>read (X)</td>
<td></td>
<td></td>
<td>write (W)</td>
<td></td>
<td>write (Y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>read (W)</td>
<td>write (Z)</td>
</tr>
</tbody>
</table>
Knowledge for this question

- We need to use timestamp-based protocol to ensure serializability.

- What conditions a transaction will be rolled back? (mod-17, pages 27-29)

\[
\begin{array}{|c|c|c|c|c|}
\hline
T_1 & T_2 & T_3 & T_4 & T_5 \\
\hline
\text{read (Y)} & \text{read (Y)} & \text{write (Y)} & \text{read (X)} & \\
\text{read (Z)} & \text{write (Z)} & \text{write (W)} & \text{read (Z)} & \\
\text{read (X)} & & \text{write (W)} & & \text{write (Y)} \text{ write (Z)} \\
\hline
\end{array}
\]
Knowledge for this question

- We need to use timestamp-based protocol to ensure serializability.

- What conditions a transaction will be rolled back? (mod-17, pages 27-29)

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>read ($Y$)</td>
<td>read ($Y$)</td>
<td>write ($Y$)</td>
<td>write ($Z$)</td>
<td>read ($X$)</td>
</tr>
<tr>
<td>$T_2$</td>
<td>read ($Y$)</td>
<td>read ($Z$)</td>
<td>write ($W$)</td>
<td>abort</td>
<td>read ($Z$)</td>
</tr>
<tr>
<td>$T_3$</td>
<td>read ($X$)</td>
<td>abort</td>
<td>read ($W$)</td>
<td>abort</td>
<td>write ($Y$)</td>
</tr>
<tr>
<td>$T_4$</td>
<td>write ($W$)</td>
<td></td>
<td>write ($Y$)</td>
<td></td>
<td>write ($Z$)</td>
</tr>
</tbody>
</table>
Project Advertisement

- We have several novel systems to develop:
  - Query language and engine for system reliability auditing
  - Realtime storage system behaviors emulation
  - Vulnerability scoring database system
Project Advertisement

• We have several novel systems to develop:
  - Query language and engine for system reliability auditing
  - Realtime storage system behaviors emulation
  - Vulnerability scoring database system

• You can develop a system that has not existed in this world
• You can learn a lot of skills, e.g., coding and research
• You will not waste time on profiling project because we already had very clear idea and direction
• You will have chance to become co-author in a top-tier paper
Project Advertisement

- We have several novel systems to develop:
  - Query language and engine for system reliability auditing
  - Realtime storage system behaviors emulation
  - Vulnerability scoring database system

- You can develop a system that has not existed in this world
- You can learn a lot of skills, e.g., coding and research
- You will not waste time on profiling project because we already had very clear idea and direction
- You will have chance to become co-author in a top-tier paper

Undergraduate students can use this project for senior project, and master students can use this project for research course to instead of one regular course.

Send email to Avi or me, if you are interested.
Questions?