Tree-Based Index Structure Problem

- Consider the following tree-based index structure, in which the keys are a person's last name:

- Is this a B-tree or a B+tree?

- If the tree has an order of 2, what would it look like after inserting a key/value pair for a person whose last name is Partridge?
Two-Phase Locking Problem

• Consider the following schedule:

<table>
<thead>
<tr>
<th>T₁</th>
<th>T₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>r(A)</td>
<td>w(A)</td>
</tr>
<tr>
<td></td>
<td>r(C) w(C)</td>
</tr>
</tbody>
</table>

• Is it possible under regular two-phase locking? If so, insert lock and unlock operations as needed.

• Is it possible under strict two-phase locking?

<table>
<thead>
<tr>
<th></th>
<th>regular</th>
<th>strict</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>B.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>C.</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>D.</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Logging and Recovery Problem, Part I

• At the start of recovery, what are the possible on-disk values under undo-redo logging?

<table>
<thead>
<tr>
<th>LSN</th>
<th>record contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>bxn: 1; BEGIN</td>
</tr>
<tr>
<td>250</td>
<td>bxn: 1; item: D1; old: 15; new: 25; olsn: 0</td>
</tr>
<tr>
<td>300</td>
<td>bxn: 2; BEGIN</td>
</tr>
<tr>
<td>400</td>
<td>bxn: 2; item: D2; old: 50; new: 35; olsn: 0</td>
</tr>
<tr>
<td>450</td>
<td>CHECKPOINT (active txns = 1, 2)</td>
</tr>
<tr>
<td>500</td>
<td>bxn: 1; item: D3; old: 40; new: 20; olsn: 0</td>
</tr>
<tr>
<td>550</td>
<td>bxn: 1; item: D1; old: 25; new: 75; olsn: 250</td>
</tr>
<tr>
<td>600</td>
<td>bxn: 1; COMMIT</td>
</tr>
<tr>
<td>650</td>
<td>bxn: 2; item: D3; old: 20; new: 10; olsn: 500</td>
</tr>
<tr>
<td>700</td>
<td>bxn: 2; item: D2; old: 35; new: 40; olsn: 400</td>
</tr>
</tbody>
</table>
Logging and Recovery Problem, Part II

- What steps are taken during recovery if we use the LSNs to ensure that actions aren't taken unnecessarily?

<table>
<thead>
<tr>
<th>LSN</th>
<th>record contents</th>
<th>backward pass</th>
<th>forward pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>txn: 1; BEGIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>txn: 1; item: D1; old: 15; new: 25; olsn: 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>txn: 2; BEGIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>txn: 2; item: D2; old: 50; new: 35; olsn: 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>CHECKPOINT (active txns = 1, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>txn: 1; item: D3; old: 40; new: 20; olsn: 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>txn: 1; item: D1; old: 25; new: 75; olsn: 250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>txn: 1; COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>650</td>
<td>txn: 2; item: D3; old: 20; new: 10; olsn: 500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>txn: 2; item: D2; old: 35; new: 40; olsn: 400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

datum LSNs
D1: 550
D2: 400
D3: 0

ACID Properties Problem

- Recall that a transaction has the following “ACID” properties:
  - Atomicity: either all of its changes take effect or none do
  - Consistency preservation: its operations take the database from one consistent state to another
  - Isolation: it is not affected by other concurrent transactions
  - Durability: once it completes, its changes survive failures

- If you disable locking in your DBMS, which of these properties do you lose?

- What if you disable logging?

- What other properties do we need to ensure full isolation?
SQL Problem

• Recall our movie database from PS 2:
  Movie(id, name, year, rating, runtime, genre, earnings_rank)
  Person(id, name, dob, pob)
  Actor(actor_id, movie_id)
  Director(director_id, movie_id)
  Oscar(movie_id, person_id, type, year)

• Write a query to retrieve the winners (name and type) of the acting and directing Oscars in the year 2000.

Deadlock Detection Problem

• Would the following schedule produce a deadlock under rigorous 2PL? Assume that no commits occur during the sequence of operations shown.
  \( \text{r}_4(B); \text{r}_1(B); \text{r}_3(A); \text{r}_5(C); \text{r}_2(C); \text{w}_6(B); \text{r}_1(A); \text{w}_5(A); \text{w}_6(C); \text{w}_3(D); \ldots \)
ER Diagram Problem

- Consider the following ER diagram:

```
A
  ↓
  aid

R
  ↓
  name

B
  ↓
  bid

name
```

- What constraints does this diagram specify?

  **uniqueness constraints:**

  **cardinality constraints:**

- What attributes would be needed for the primary key of the relation used to capture R?

XML Problem

- Consider an XML document that looks like this:

```
<students>
  <student>
    <name>Alan Turing</name>
    <course>CSCI E-119</course>
    <course>CSCI E-268</course>
    <course>CSCI E-50a</course>
    <course>CSCI E-50b</course>
    <course>CSCI E-160</course>
    <course>CSCI E-215</course>
    <course>CSCI E-113</course>
  </student>
  ...
</students>
```

- Give two different ways of finding the names of all students taking CSCI E-268.
**MongoDB Problem**

```json
{   _id: "0499549",
    name: "Avatar",
    year: 2009,
    rating: "PG-13",
    runtime: 162,
    genre: "AVYS",
    earnings_rank: 1,
    actors: [{ id:"0000244", name: "Sigourney Weaver" }, ... ],
    directors: [{ id: "0000116", name: "James Cameron" } ]
}
```

- Write a query to find, for each director of a top-200 movie, the number of top-grossing movies by that director.

**Timestamp-Based Concurrency Control Problem**

- How will this schedule be executed (without commit bits)?
  
  $w_1(B); r_2(A); w_2(A); r_1(A); r_3(C); w_3(B); w_2(B); w_2(C); r_3(A); w_3(C)$

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>RTS = WTS = 0</td>
<td>RTS = WTS = 0</td>
<td>RTS = WTS = 0</td>
</tr>
</tbody>
</table>
Replication Problem

A database is being replicated across 20 sites, and a client wants to update one of the data items.

• If the database is being managed by a system that uses synchronous replication and fully distributed locking, what is the minimum number of copies that a client would need to update?

• What if the system were using synchronous replication and primary-copy locking?

Two-Phase Commit (2PC) Problem

A transaction T requires subtxns at sites A, B, and C. Site A is the coordinator.

Consider the following sequence of events from the 2PC of T:

1. After the last operation in T, site A:
   • sends prepare messages to B and C
   • puts its own subtxn in the ready state
2. Sites B and C:
   • put their subtxns in the ready state
   • force-write ready records and send ready messages to A
3. Site A force-writes its commit record and sends commit messages to B and C.
4. Site B receives the commit message from A and force-writes its own commit record.
5. Site C receives the commit message from A and force-writes its own commit record. (continued)
Two-Phase Commit (2PC) Problem (cont.)

Now consider the following variations on the above scenario:

1. If C crashes \textit{before} force-writing its commit record, what should happen during its recovery?

2. If B crashes \textit{after} force-writing its commit record, what should happen during its recovery?

3. If B crashes \textit{before} force-writing its \textit{ready} record:
   • what should happen during its recovery?
   • what should A and C do?

4. If A crashes \textit{before} force-writing its commit record, what should B and C do?