2 Main Parts

1. Transaction Semantics (ACID)
2. Multithreading Concurrency Control
Transaction Semantics
Threads, Transactions and Xacts

- **Threads**
  - Transaction
    - Statements or Xacts
      - Write(), Write(), Write(),...
      - Read(), Write(), Read(),...
      - Read(), Read(), Read(),...
      - ...
      - ...
      - ...
      - ...
      - Write(), Write(), Write(),...
      - Read(), Write(), Read(),...
      - Read(), Read(), Read(),...
      - ...
      - ...
      - ...
      - ...

- **Single Thread**
  - Transaction 1
  - Transaction n
ACID: Atomicity

**Some Possibilities:**
1. fail to acquire lock (most common)
2. DB crashes/errors
3. power failures
Multithreading
Concurrency
Control
ACID: Isolation

Issues in Thread Synchronization

DeadLocks

Race Condition

2PL & QueCC

Latch
<table>
<thead>
<tr>
<th></th>
<th>Locks</th>
<th>Latches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate...</td>
<td>User Transaction</td>
<td>Threads</td>
</tr>
<tr>
<td>Protect...</td>
<td>DB Content</td>
<td>In-Memory Data Structure</td>
</tr>
<tr>
<td>During</td>
<td>Entire Transaction</td>
<td>Critical Section</td>
</tr>
<tr>
<td>Kept in...</td>
<td>Lock Manager (Hashmap)</td>
<td>Protected Data Structure</td>
</tr>
</tbody>
</table>
Locks: 2PL

- **Shared**
  - Read()  
  - query(*args == 3)  
  - SELECT(), SUM(), AGGREGATE()

- **Exclusive**
  - Write()  
  - query(*args != 3)  
  - INSERT(), UPDATE(), DELETE()
Lock: 2PL (Shared)

- Is there an exclusive lock?
  - Exclusive lock is present
    - Shared lock not granted
  - No exclusive lock
    - Shared lock granted
Lock: 2PL (Exclusive)

Is there a shared lock?

- No Shared Lock
- Has shared lock

Is there an exclusive lock?

- No Exclusive Lock
- Has Exclusive lock

Who holds the shared lock?

- The one reading is the one writing
- The one/ones reading is NOT the one writing

Upgrade to EL

* will be protected by latch as well

EL granted

EL not granted

EL not granted
Locks: QueCC

class Planner:

  Transaction worker

  = Transactions

Planning Thread #1

Planning Thread #2

Low priority Queue

High Priority Queue
Since Ranking is Arbitrary, Index in Priority List = RID % 10

Eg: RID = 1001, Index = 1001%10 = 1
Locks: QueCC

Planning Thread #1

Transaction worker

Planning Thread #2

Low priority Queue

High Priority Queue
Locks: QueCC

- Low priority Queue
- High Priority Queue

Transaction worker

execute
Additional Implementation
Additional Aggregate Functions

- max(): Get the maximum value
- min(): Get the minimum value
- avg(): Get the average value
- count(): Get the count

*does not support multi-thread implementation
Things to Improve Upon

- Writing the program in a different language to support multithreading
  - C++ or Java
- Code writing style
  - Commenting
- Improving algorithm efficiency within functions
  - Removing nested and repeated loops
Thank You!