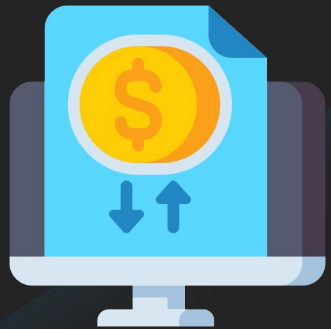
Milestone 3

The Data Dragons: Nick Fitzpatrick, Michael Liu, Joshua Lee, Saif Khan, Meghana Manepalli



ECS 165A, March 15th 2024

Overview

Transaction Semantics:

- Transaction Class
- Transaction-worker Class
- ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability)

Concurrency Control:

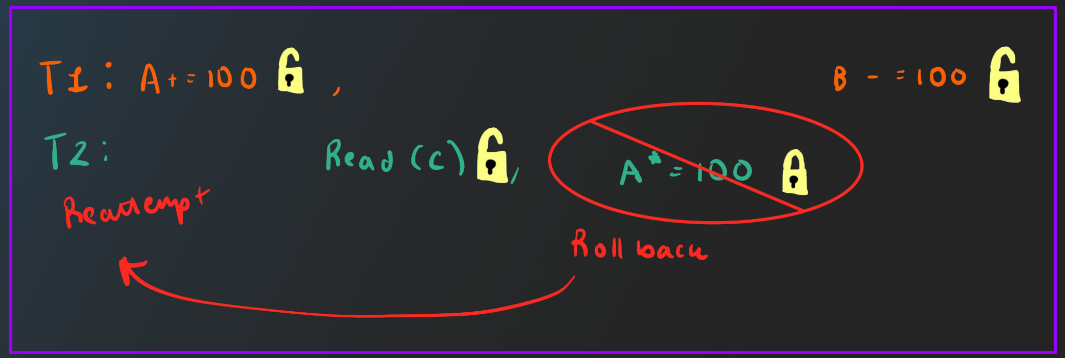
- Multi-Threading
- Locks
- Aborting

Final:

- Achievements/Improvements

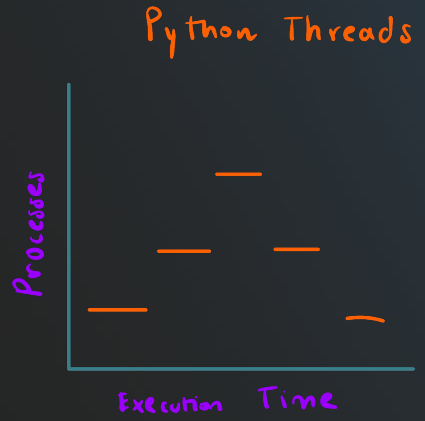
Transaction Semantics

- **A**tomicity: Each transaction is treated as a single unit
  - All statements succeed completely or entire transaction fails
- A successful transaction is committed
- Failure in the transaction will result in rollback
  - Thread will keep attempting execution



Concurrency Control

- Multiple transactions will occur concurrently to fully utilize resources
  - Optimal performance
- Note: Our implementation is in Python, so it is concurrent but not truly parallel
- **I**solation: Intermediate states from any transaction will not be visible to other transactions
  - Avoid Dirty Reads, Non Repeatable Reads

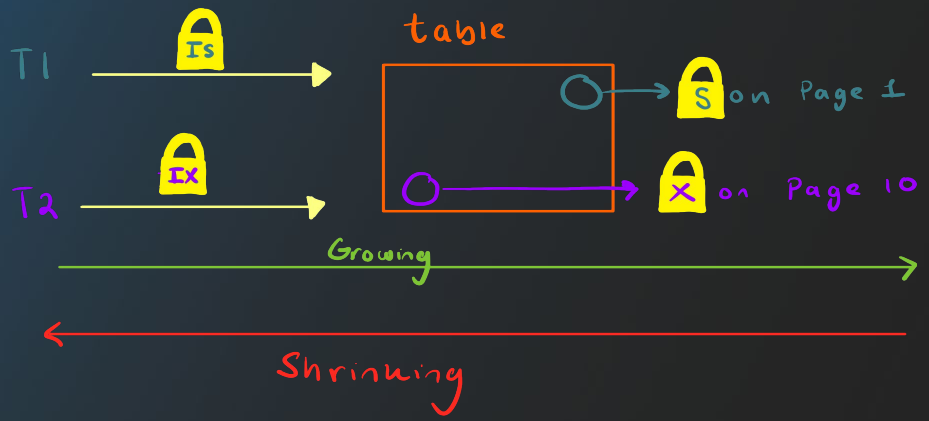


Strict 2PL Locking

- Growing Phase: Acquiring locks on relevant objects
- Shrinking Phase: Releasing locks after committing
- Shared Locks are used for reading
- Exclusive Locks are used for writing
- No wait property eliminates deadlocks
- Why? Highest level of Isolation
  - Schedule is serializable → No Dirty Reads or Non Repeatable Reads
  - Strong guarantee of data integrity and consistency
- Trade Offs
  - Reduced Concurrency
  - Locking Overhead
  - Overall Reduce Performance

Lock Manager

- Multi-Granularity Locks (MGL)
  - Obtained from Table level to Page level
  - Released from Page level to Table level
- Intention locks acquired at higher granularity
- Ensures compatibility of lock types requested on same objects



	IS	IX	SIX	S	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
SIX	✓	X	X	X	X
S	✓	X	X	✓	X
X	X	X	X	X	X

Transaction Procedure

1. Fetch objects that will be accessed/modified in the transaction
2. Acquire locks from lock manager

Successful Transaction

1. All actions are executed - each logged beforehand by Write Ahead Logger
2. Modified data committed whenever necessary
3. All locks are released

Aborted Transaction

1. Certain number of actions executed
2. Invalid action occurs, enter exception handler
3. All locks are released, **Transaction not committed**

What we achieved and moving forward

- Improvements we made from M2
  - Concurrency achieved - improvement on resource utilization and efficiency
  - Consistency - Locks for data protection and logging for system recovery
- Next Steps..
  - Crash Recovery
  - Increased Granularity for improved resource utilization
  - Language Exploration – Efficiency and Parallelism
  - Predicate Locking to prevent Phantom Reads – Data integrity
  - Support range queries, indexing strategies, various bufferpool policies etc. (many ways to improve!)

Thank you!!!