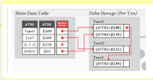
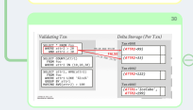
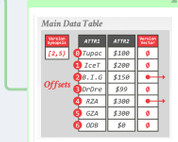
06-mvcc2

- Microsoft Hekaton
  - Incubator project started in 2008 to create new in-memory OLTP engine for ***Microsoft SQL Server***.
    - Led by top database experts Paul Larson and Mike Zwilling
  - It was important that the new engine integrated with ***MSSQL*** ecosystem.
  - It was also necessary that the engine support all possible OLTP workloads with predictable performance.
    - Single-threaded partitioning (e.g., H-Store) works well for some applications but terrible for others.
  - The Hekaton concurrency control relies on compare-and-swap to install new versions
    - Reference
      P.-A. Larson, S. Blanas, C. Diaconu, C. Freedman, J. M. Patel, and M. Zwilling. High-performance concurrency control mechanisms for main-memory databases. In VLDB '11: Proceedings of the VLDB Endowment, volume 5, pages 298-309, December 2011. URL https://dl.acm.org/citation.cfm?id=2095686.2095689.
    - Each transaction is assigned a timestamp when they begin (BeginTS) and when they commit (EndTS)
    - Each tuple contains two timestamps that represents their visibility and current state
      - BEGIN
        - The BeginTS of the active transaction or the EndTS of the committed transaction that created it
      - END
        - The BeginTS of the active transaction that created the next version or infinity or the EndTS of the committed transaction that created it
    - Speculative Reads
      - Hekaton allows transactions to read versions of transactions of uncommitted trans- actions. Then checks in validations if the transactions that it read uncommitted data committed.
      - 
      - 交易committed: 重新赋值
        
    - The DBMS does not allow speculative write; first transaction to write a new version succeeds, second gets aborted
      - 不能更新uncommitted的交易
        
  - Transaction State Map
    - Global map of all transactions states in the system. Transactions have to consult this map to determine the state of a transaction
    - ACTIVE
      - The transaction is executing read/write operations.
    - VALIDATING
      - The transaction has invoked commit and the DBMS is checking whether it is valid
    - COMMITTED
      - The transaction is finished, but may not have updated its version's TS.
    - TERMINATED
      - The transaction has updated the TS for all of the versions that it created.
  - Transaction Meta-Data
    - Read Set
      - Pointers to every version that the transaction read.

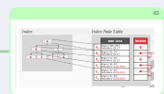
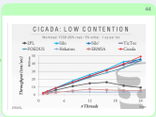
        "Invidual tuples that I read"
    - Write Set
      - Pointers to the versions that the transaction updated (old and new), deleted (old), and inserted (new).
    - Scan Set
      - Stores enough information needed to perform each scan operation
    - Commit Dependencies
      - List of transactions that are waiting for this transaction to finish.
  - Transaction Validation
    - Read Stability
      - Check that each version read is still visible as of the end of the transaction.
    - Phantom Avoidance
      - Repeat each scan to check whether new versions have become visible since the transaction began.
    - Extend of validation depends on isolation level
      - SERIALIZABLE
        - Read Stability + Phantom Avoidance
      - REPEATABLE READS
        - Read Stability
      - SNAPSHOT ISOLATION
        - None
      - READ COMMITTED
        - None
  - 乐观和悲观
    
    - Optimistic Txns
      - Check whether a version read is still visible at the end of the txn.
      - Repeat all index scans to check for phantoms
    - Pessimistic Txns
      - Use shared & exclusive locks on records and buckets.
      - No validation is needed
      - Separate background thread to detect deadlocks
  - Key lessons
    - Use only lock-free data structures. This means no latches, spin locks, or critical sections for indexes, transaction map, memory allocator, garbage collector.
    - Only one single serialization point in the DBMS to get the transaction's begin and commit timestamp using an atomic addition (CAS).
  - 发现
    - Read/scan set validations are expensive if the txns access a lot of data
    - Appending new versions hurts the performance of OLAP scans due to pointer chasing & branching.
    - Record-level conflict checks may be too coarse- grained and incur false positives.

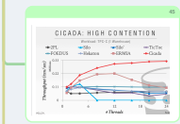
- HyPer MVCC
  - Column-store with delta record versioning.
    HyPer uses a MVCC implementation that is designed for HTAP workloads

    T. Neumann, T. Mhlbauer, and A. Kemper. Fast serializable multi-version concurrency control for main- memory database systems. In SIGMOD '15: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pages 677-689, 2015. doi: https://dl.acm.org/citation.cfm?id=2749436.
    - In-Place updates for non-indexed attributes
    - Delete/Insert updates for indexed attributes.
    - Newest-to-Oldest Version Chains
    - No Predicate Locks / No Scan Checks
  - Avoids write-write conflicts by aborting txns that try to update an uncommitted object
  - Designed for HTAP workloads.
  - 存储架构
    
  - Transaction Validation
    - First-Writer Wins
      - The version vector always points to the last committed version. Do not need to check whether write-sets overlap.
    - Check the undo buffers (i.e., delta records) of transactions that committed after the validation trans- action started.
    - Compare the committed transaction's write set for phantoms using ***Precision Locks***
      - Reference
        J. R. Jordan, J. Banerjee, and R. B. Batman. Precision locks. In Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data, SIGMOD '81, pages 143-147, 1981. doi: 10.1145/582318.582340. URL http://doi.acm.org/10.1145/582318.582340.
    - Only need to store the transaction's read predicates and not its entire read set.
    - 
      
      
      
  - Version Synopsis
    - Store a separate column that tracks the position of the first and last versioned tuple in a block of tuples
    - When scanning tuples, the DBMS can check for strides of tuples without older versions and execute more efficiently.
    - 

      2和4之间的才有version，之外的没有，所以可以很快处理
  - Best-effort in-lining

    - Record meta-data is stored in a fixed location.
    - Threads will attempt to inline read-mostly version within this meta-data to reduce version chain traversals
  - Loosely Synchronized Clocks
  - Fast Validation
    - Contention Aware validation

      contention-aware 调度的意思是，根据事务整体的锁竞争会对系统造成的影响，将事务分配优先级。调度是根据优先级进行的，而不是按FIFO 方式
      - Validate access to recently modified (high contention) records first.
    - Early Consistency Check
      - Pre-validated access set before making global writes.
    - If the DBMS knows that most of the recently executed transactions committed successfully, then it can skip Contention-Aware Validation and Early Consistency check
    - Incremental Version Search:
      - Resume from last search location in version list
  - Index Nodes Stored in Tables
    - 
  - 

  - 

  - Andy considers HyPer and Cicada to be state-of- the-art as of January 2018.

- CMU Cicada
  In-memory OLTP engine based on optimistic MVCC with append-only storage (N2O) [3]. Designed to be scalable for both low and high contention workloads.

  H. Lim, M. Kaminsky, and D. G. Andersen. Cicada: Dependably fast multi-core in-memory transac- tions. In SIGMOD '17: Proceedings of the 2008 ACM SIGMOD International Conference on Manage- ment of Data, pages 21-35, 2017. doi: https://dl.acm.org/citation.cfm?id=2749436.