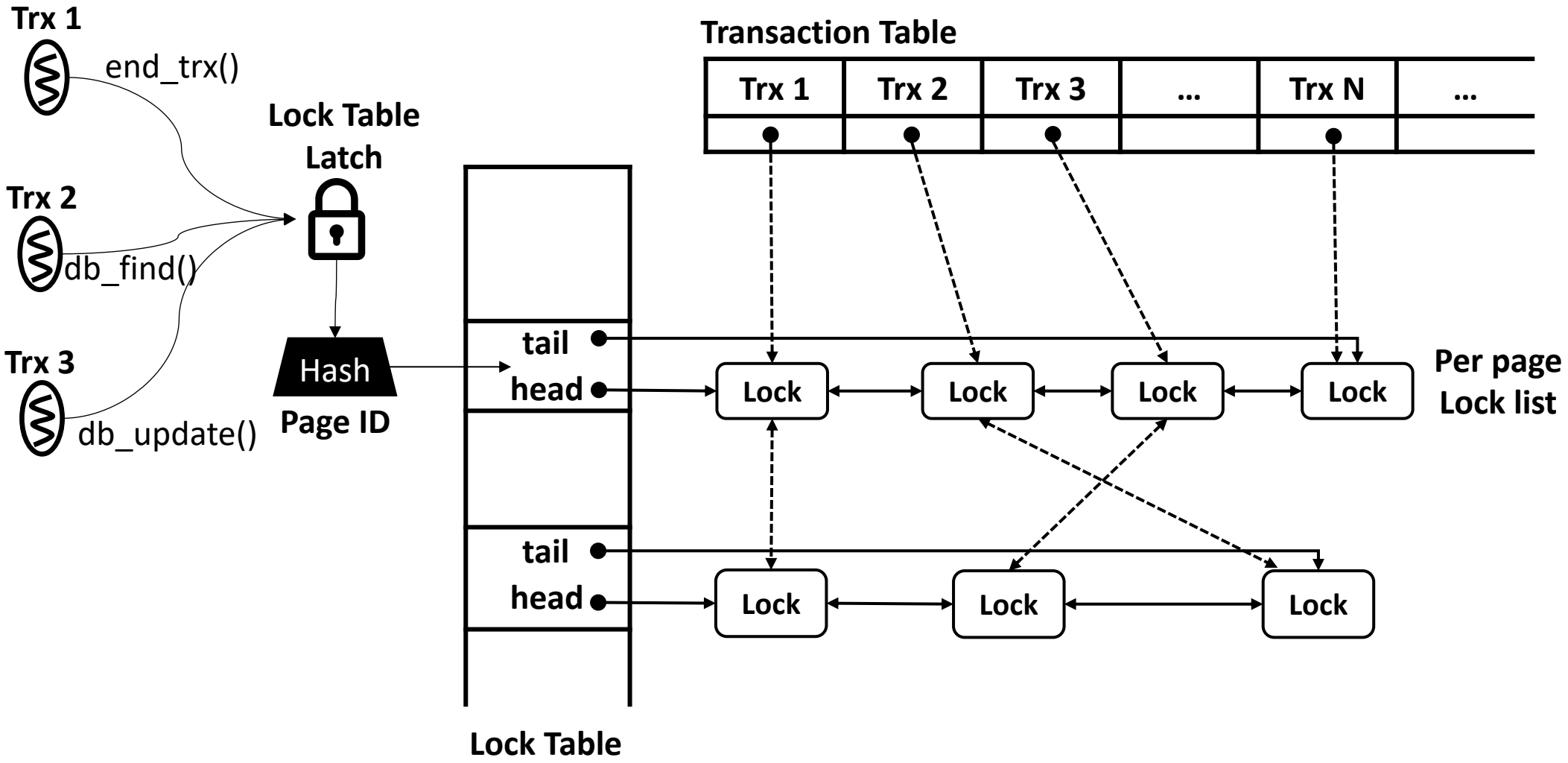
Database System
Project 5 Supplement

Concurrency Control

Scalable Computing Systems Laboratory
Hanyang University

Overview



Latch (Mutex)

- Buffer Pool Latch
  - Prevent race condition on buffer pool modification(LRU list).
- Buffer Page Latch
  - Prevent eviction of a buffer page by another thread.
  - Prevent race condition on access(read/write) a buffer page.
- Transaction System Latch
  - Prevent race condition on generating transaction id and accessing transaction table.
- Lock Table Latch
  - Prevent race condition on accessing lock table.

Begin Transaction

① Acquire the transaction system latch.

② Get a new transaction id.

③ Insert a new transaction structure in the transaction table.

④ Release the transaction system latch.

⑤ Return the new transaction id.

Update / Find

① Acquire the buffer pool latch.

② Find a leaf page containing the given record(key).

③ Try to acquire the buffer page latch.

  ① If fail to acquire, release the buffer pool latch and go to (1).

④ Release the buffer pool latch.

⑤ Try to acquire record lock.

  ① If fail due to deadlock, abort transaction and release buffer page latch. Return FAIL.

  ② If fail due to lock conflict, release the buffer page latch and wait(sleep) until another thread wake me up. After waken up, go to (1).

⑥ Do update / find.

⑦ Release the buffer page latch.

⑧ Return SUCCESS.

Acquire Record Lock

① Acquire the lock table latch.

② Find the linked list with identical page id in lock table.

③ Find lock nodes of given key in the list. If there is not a lock node of given key, insert a new lock node.

  ① If no conflict, return SUCCESS.

  ② If conflict, return CONFLICT.

  ③ If deadlock is detected, return DEADLOCK.

End Transaction

① Acquire the lock table latch.

② Release all acquired locks and wake up threads who wait on this transaction(lock node).

③ Release the lock table latch.

④ Acquire the transaction system latch.

⑤ Delete the transaction from the transaction table.

⑥ Release the transaction system latch.

⑦ Return the transaction id.

Abort Transaction

① Iterate reversely undo logs of the transaction.
  ① Acquire the buffer pool latch.
  ② Find a leaf page containing the given record(key).
  ③ Try to acquire the buffer page latch.
    ① If fail to acquire, release the buffer pool latch and go to (1)-(1).
  ④ Release the buffer pool latch.
  ⑤ Rollback one undo log.
  ⑥ Release the buffer page latch.
② End transaction.

Transaction Manager

```
/* Struct for managing whole transaction system. */
typedef struct {
    /* Mapping transaction id to transaction structure. */
    TransactionTable    trx_table;

    /* Next transaction id for will-be-generated-transaction. */
    TransactionId       next_trx_id;

    /* Latch to protect transaction manager. */
    pthread_mutex_t     trx_sys_mutex;
} TransactionManager;
```

Transaction

```
/* Struct for a transaction. */
typedef struct {
    /* Transaction id. */
    TransactionId       trx_id;

    /* Transaction state. */
    TransactionState    trx_state;

    /* Lock list which this transaction acquire. */
    LockPtrList         acquired_locks;

    /* Mutex & condition variable to wait & wakeup. */
    pthread_mutex_t     trx_mutex;
    pthread_cond_t      trx_cond;

    /* A lock node which this transaction wait for. */
    LockPtr             wait_lock;

    /* Undo log list. */
    UndoLogList         undo_log_list;
} Transaction;
```

Undo Log

```
/* Struct for a undo log. */
typedef struct {
    TableId         table_id;
    Key             key;
    Value           old_value;
} UndoLog;
```

Lock Node

```
/* Struct for lock node. */
typedef struct lock_t {
    TableId         table_id;
    TransactionPtr  trx;

    /* Page id where the record is. */
    PageId          page_id;

    Key             key;

    /* Whether this lock is acquired or not(wait). */
    bool            acquired;

    LockMode        lock_mode;

    /* Linked with the lock nodes
     * contained in the same hash bucket. */
    lock_t*         prev;
    lock_t*         next;
} Lock;
```

Lock Manager

```
/* Struct for lock manager. */
typedef struct {
    /* Mapping page id to lock list. */
    LockTable           lock_table;

    /* Latch to protect lock manager. */
    pthread_mutex_t     lock_sys_mutex;
} LockManager;
```

Thank you