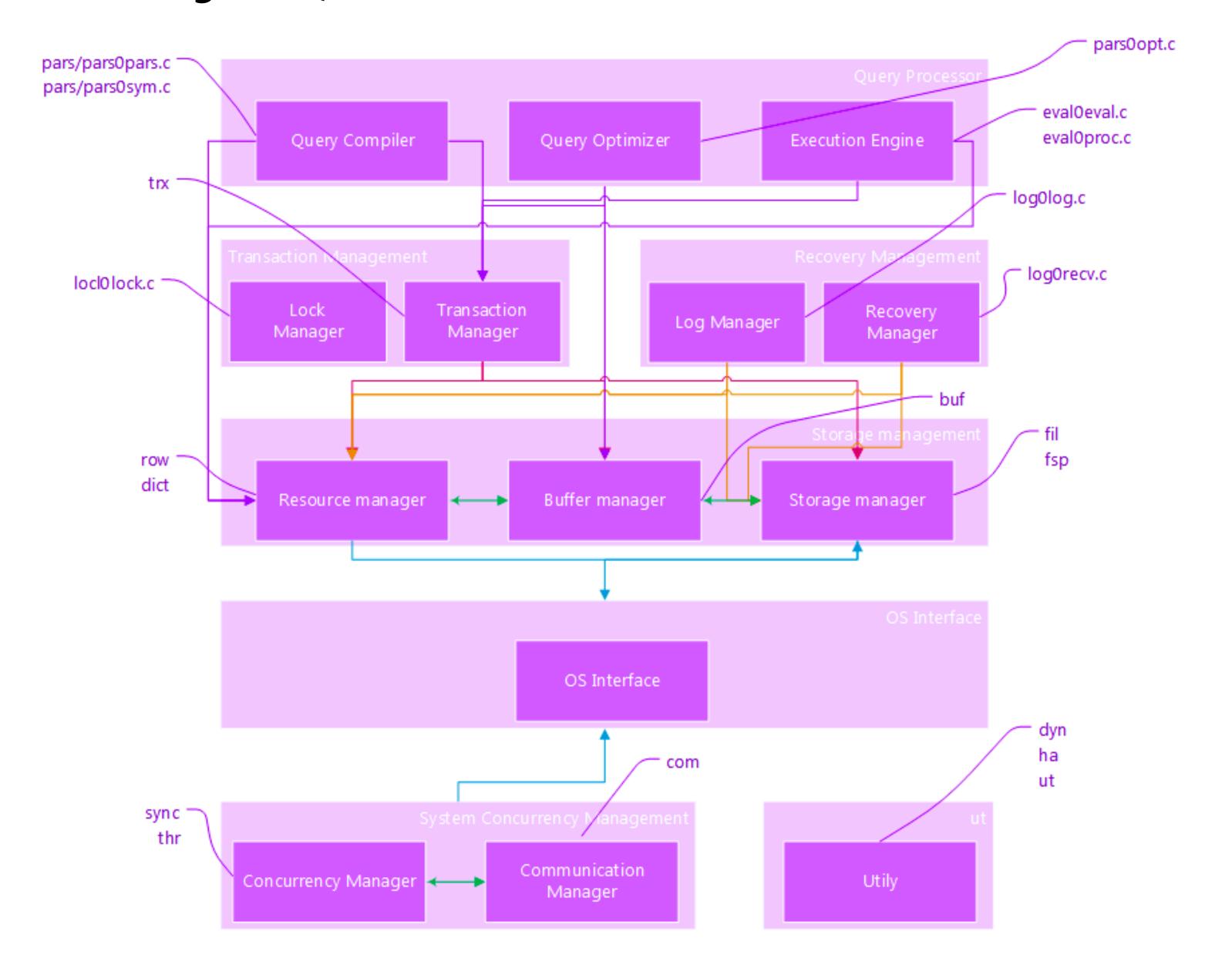
# Chapter 10 Transaction Manager Concept : Mini-Transaction @ MySQL

Jong-Hyeok Park akindo19@gmail.com





## MySQL InnoDB Architecture



## Mini Transaction (MTR)

- Unit guarantees atomic changes to multiple pages
- No rollback; only mtr\_start() and mtr\_commit()
- MTR is not a transaction

#### IF index exists

• (For example) Insert operation, there are 4 (or more) mini-transactions

- 1 Allocate UNDO spaces
- ② Write UNDO logs
- 3 Write data (inserted records)
- 4 Commit

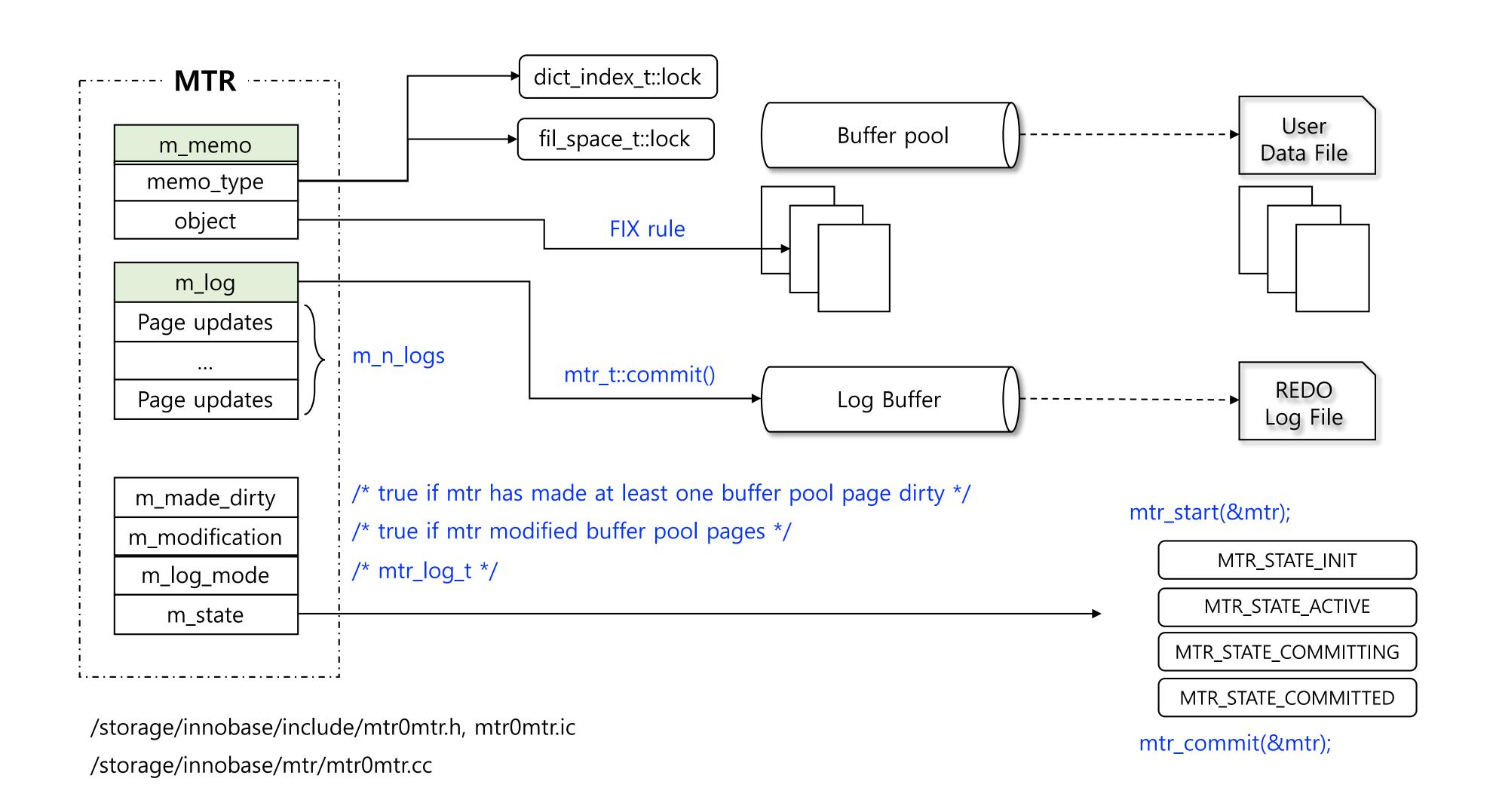
#### **INSERT:**

```
row_ins_clust_index_entry_low
mtr1_start()
    trx_undo_report_row_operation
    mtr2_start()
        trx_undo_assign_undo
        mtr3_start()
            trx_redo_reuse_cached
            mtr3_commit()
        ...
    mtr2_commit()
    page_cur_insert_rec_write_log
mtr1_commit()
```

#### **COMMIT:**

```
trx_commit
mtr4_start()
    trx_commit_low
    ...
mtr4_commit()
```

## Mini Transaction (MTR)



## Insert (Overview)

#### **Transaction INSERT**

```
row_ins_index_entry()

row_ins_clust_index_entry()

row_ins_clust_index_entry_low()

mtr.start(); -- mini trx start --
btr_cur_optimistic_insert();

mtr.commit(); -- mini trx commit --
```

#### mtr.commit()

```
- cmd.execute()
- m_impl->m_log.for_each_block(write_log);
    Append block to WAL log buffer
- add_dirty_blocks_to_flush_list();
    Add_dirty_blocks_to_flush_list()
    buf_flush_note_modification()
    block->page.newest_modification = end_lsn
    Update pageLSN and flush dirty page to flush list
```

```
btr_cur_optimistic_insert();
        btr_cur_ins_lock_and_undo();
                                        ** Part1. check lock & write UNDO logs
         trx_undo_report_row_operation()
           mtr_start(); -- mini trx start --
           trx_undo_assign_undo();
           trx_undo_page_report_insert();
              trx_undo_page_report_insert()
                  Write UNDO logs
                  trx_undo_page_set_next_prev_and_add()
                   trx_undof_page_add_undo_rec_log()
                    Write REDO logs of UNDO logs
                                               ** Flush undo logs + redo of undo logs
           mtr.commit(); -- mini trx commit -
                                               to the WAL log buffer
           build_roll_ptr();
                                         ** Part2. insert records & write REDO logs
        btr_cur_ins_lock_and_undo();
           page_cur_insert_rec_low() ** actual insert
            page_cur_insert_rec_write_log()
            Write REDO logs
```

# Update (Overview)

#### **Transaction UPDATE**

```
- row_upd()
- row_upd_clust_step()
- row_upd_clust_rec()
    mtr.start(); -- mini trx start --
    btr_cur_optimistic_update();
    mtr.commit(); -- mini trx commit --
```

```
btr_cur_optimistic_update();
        btr_cur_update_in_place();
                                          ** Part1. check lock & write UNDO logs
             lock_clust_rec_modify_check_and_lock()
             trx_undo_report_row_operation()
               mtr_start(); -- mini trx start --
               trx_undo_assign_undo();
               trx_undo_page_report_modify();
                   Write UNDO logs
                  trx_undof_page_add_undo_rec_log()
                   Write REDO logs of UNDO logs
                                                  ** Flush undo logs + redo of undo logs
               mtr.commit(); -- mini trx commit -
                                                 to the WAL log buffer
               build_roll_ptr();
                                          ** Part2. update records & write REDO logs
         row_upd_rec_in_place()
                                          ** actual update
       btr_cur_update_in_place_logWrite REDO logs
```

## DELETE (Overview)

#### **Transaction DELETE**

```
row_upd()

row_upd_clust_step()
    mtr.start(); -- mini trx start --

row_upd_del_mark_clust_rec()

row_upd_store_row()

btr_cur_del_mark_set_clust_rec()

mtr.commit(); -- mini trx commit --
```

```
btr_cur_del_mark_set_clust_rec()
       lock_clust_rec_modify_check_and_lock()
       trx_undo_report_row_operation()
                                           ** Part1. check lock & write UNDO logs
          mtr_start(); -- mini trx start --
          trx_undo_assign_undo();
          trx_undo_page_report_modify();
           Write UNDO logs
              trx_undof_page_add_undo_rec_log()
              Write REDO logs of UNDO logs
                                                  ** Flush undo logs + redo of undo logs
          mtr.commit(); -- mini trx commit -
                                                 to the WAL log buffer
          build_roll_ptr();
        btr_rec_set_deleted_flag()
                                            ** Part2. delete records & write REDO logs
       row_upd_rec_sys_fields()
                                                  ** build roll_ptr
       btr_cur_del_mark_set_clust_rec_log()
         Write REDO logs
```

### Reference

- [1] Jim Gray and Andreas Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, San Mateo, CA (1993)
- [2] http://mingxinglai.com/cn/2015/09/Architecture-Of-MySQL/