Assignmen3 Report

***重要:* 助教抱歉 這次突發狀況 所以我沒有完成... 造成助教困擾 謝謝助教願意給機會**

Implement Explain operantion:

At class LockTable :

首先先加上一個變數 C_LOCK

```
final static int IS_LOCK = 0, IX_LOCK = 1, S_LOCK = 2, SIX_LOCK = 3,
        X_LOCK = 4, C_LOCK = 5;
```

因為一個 item 的 Clock 一次只能被一個 tx 拿到 所以我們單純用數字紀錄即可 NONE 表示沒人拿

```
class Lockers {
    Set<Long> sLockers, ixLockers, isLockers, requestSet;
    // only one tx can hold xLock(sixLock) on single item
    long sixLocker, xLocker, cLocker;
    static final long NONE = -1; // for sixLocker, xLocker

    Lockers() {
        sLockers = new HashSet<Long>();
        ixLockers = new HashSet<Long>();
        isLockers = new HashSet<Long>();
        sixLocker = NONE;
        xLocker = NONE;
        cLocker = NONE;
        requestSet = new HashSet<Long>();
    }
```

這邊只舉改動的其中一個當例子，因為如果有 slock 就不能再拿這三個，所以要把原本 if 裡面的內容，改成符合 2V2PL 規則的形式

```
if (lockType == IX_LOCK || lockType == SIX_LOCK || lockType == C_LOCK) {
    for (Long tx : lks.sLockers) {
        if (tx > txNum) {
            txnsToBeAborted.add(tx);
            if (!toBeNotified.contains(tx))
                toBeNotified.add(tx);
        }
    }
}
```

	S	X	C	IS	IX	SIX
S	O	O	X	O	X	X
X	O	X	X	O	X	X
C	X	X	X	X	X	X
IS	O	O	X	O	O	O
IX	X	X	X	O	O	X
SIX	X	X	X	O	X	X

實作 cLock 跟 xLock 很像，先檢查是不是有 cLock 了，有就直接回傳，沒有的話，就等待拿取 cLock，跳出 while 時，如果還是不能拿，那就跳錯誤訊息，否則就拿

```
void cLock(Object obj, long txNum) {
    Object anchor = getAnchor(obj);
    txWaitMap.put(txNum, anchor);
    synchronized (anchor) {
        Lockers lks = prepareLockers(obj);

        if (hasCLock(lks, txNum))
            return;

        try {
            long timestamp = System.currentTimeMillis();
            while (!cLockable(lks, txNum) && !waitingTooLong(timestamp)) { // 當不允許clock
                avoidDeadlock(lks, txNum, C_LOCK);
                lks.requestSet.add(txNum);

                anchor.wait(MAX_TIME);
                lks.requestSet.remove(txNum);
            }
            if (!cLockable(lks, txNum))
                throw new LockAbortException();
            lks.cLocker = txNum;
            getObjectSet(txNum).add(obj);
```

在 release 裡面，會先去呼叫 releaseLock 把那個 type 的 lock 給 release 掉，這邊都要把 clock 給加進去

```
if (lks != null) {
    releaseLock(lks, anchor, txNum, lockType);

    // Check if this transaction have any other lock on this object
    if (!hasSLock(lks, txNum) && !hasXLock(lks, txNum)
            && !hasSixLock(lks, txNum) && !hasIsLock(lks, txNum)
            && !hasIxLock(lks, txNum) && !hasCLock(lks, txNum)) {
        getObjectSet(txNum).remove(obj);

        // Remove the locker, if there is no other transaction
        // having it
        if (!sLocked(lks) && !xLocked(lks) && !sixLocked(lks)
                && !isLocked(lks) && !ixLocked(lks) && !cLocked(lks)
                && lks.requestSet.isEmpty())
            lockerMap.remove(obj);
    }
}
```

在 releaseAll 裡面，因為 C_LOCK 只有一個，所以直接呼叫 releaseLock 即可，像 IsLock 就必須用 while 迴圈，去把全部都 release 掉才行

```
if (lks != null) {

    if (hasSLock(lks, txNum))
        releaseLock(lks, anchor, txNum, S_LOCK);

    if (hasXLock(lks, txNum) && !sLockOnly)
        releaseLock(lks, anchor, txNum, X_LOCK);

    if (hasSixLock(lks, txNum))
        releaseLock(lks, anchor, txNum, SIX_LOCK);

    if (hasCLock(lks, txNum))
        releaseLock(lks, anchor, txNum, C_LOCK);

    while (hasIsLock(lks, txNum))
        releaseLock(lks, anchor, txNum, IS_LOCK);

    while (hasIxLock(lks, txNum) && !sLockOnly)
        releaseLock(lks, anchor, txNum, IX_LOCK);
```

最後再把沒有用的 locker 給刪掉

```
// Remove the locker, if there is no other transaction
// having it
if (!sLocked(lks) && !xLocked(lks) && !sixLocked(lks)
        && !isLocked(lks) && !ixLocked(lks) && !cLocked(lks)
        && lks.requestSet.isEmpty())
    lockerMap.remove(obj);
```

在實作 releaseLock 裡面，新增一個關於 C_LOCK 的 case 如果有 C_LOCK 就把她設回 -1 代表刪掉他了

```
case C_LOCK:
    if (lks.cLocker == txNum) {
        lks.cLocker = -1;

        anchor.notifyAll();
    }
    return;
```

新增一個判斷現在是不是被 clock 給鎖上的函數

```
private boolean cLocked(Lockers lks) {
    return lks != null && lks.cLocker != -1;
}
```

新增一個判斷 locker 裡面是否有 clock 的函數

```
private boolean hasCLock(Lockers lks, long txNUm) {
    return lks != null && lks.cLocker == txNUm;
}
```

新增一個判斷現在可不可以拿 clock 的函數，因為一旦拿 clock 其他都不能拿，所以要檢查是否有其他的 lock 存在

```
private boolean cLockable(Lockers lks, long txNum) {
    return (!sixLocked(lks) || hasSixLock(lks, txNum))
            && (!ixLocked(lks) || isTheOnlyIxLocker(lks, txNum))
            && (!sLocked(lks) || isTheOnlySLocker(lks, txNum))
            && (!xLocked(lks) || hasXLock(lks, txNum))
            && (!cLocked(lks) || hasCLock(lks, txNum))
            && (!isLocked(lks) || isTheOnlyIsLocker(lks, txNum));
}
```

At class Transaction :

在 tx 裡面新增兩個 hashmap 用來當作 private workspace 一個存 infomation 有 field_name, record_id, constant 另一個存 recordfile

```
// info: field name, record id, constant
public class info {
    public String field_name;
    public RecordId record_id;
    public Constant constant;

    public info(String field_name, RecordId record_id, Constant constant) {
        this.field_name = field_name;
        this.record_id = record_id;
        this.constant = constant;
    }
}

public HashMap<String, info> hashmap;
public HashMap<String, RecordFile> recordfiles;
```

在 commit 的時候，要去檢查每個 info 的 record_id 裡面的 fileName，然後把 RecordFile 移到那個 record_id 之後再呼叫真正的 setVal，這裡要把每個 info 跑過一次

```
public void commit() {
    if(!this.isReadOnly() || this.isReadOnly()) {
        for (Map.Entry<String, info> entry: this.hashmap.entrySet()) {
            info val = entry.getValue();
            RecordFile rf = this.recordfiles_get(val.record_id.block().fileName());
            rf.moveToRecordId(val.record_id);
            rf.real_setVal(val.field_name, val.constant);
        }
    }

    for (TransactionLifecycleListener l : lifecycleListeners)
        l.onTxCommit(this);

    if (logger.isLoggable(Level.FINE))
        logger.fine("transaction " + txNum + " committed");
}
```

新增 get, put 到這兩個 hashmap 的 method

```
public info info_get(String key) {
    return this.hashmap.get(key);
}

public info info_put(String key, info i) {
    return this.hashmap.put(key, i);
}

public RecordFile recordfiles_get(String file_name) {
    return this.recordfiles.get(file_name);
}

public RecordFile recordfiles_put(String file_name, RecordFile rf) {
    return this.recordfiles.put(file_name, rf);
}
```

At class RecordFile :

這裡原本在 setVal 的時候會直接呼叫 rp 的 setVal，因為我在 rp 有新增一個寫進 workspace 的 setVal 所以這邊改成去呼叫這個

```
public void setVal(String fldName, Constant val) {
    if (tx.isReadOnly() && !isTempTable())
        throw new UnsupportedOperationException();
    Type fldType = ti.schema().type(fldName);

    Constant v = val.castTo(fldType);
    if (Page.size(v) > Page.maxSize(fldType))
        throw new SchemaIncompatibleException();

    int position =rp.fieldPos(fldName);
    rp.workspace_setVal(fldName, position, v);

    tx.recordfiles_put(this.fileName, this);
}
```

At class RecordPage :

這裡新增了一個寫進 workspace 的 setval， 用 record_id, filename, val 去新創一個 string 和 info 然後塞到 HashMap info 裡面

```
public void workspace_setVal(String fldName, int offset, Constant val) {
    if (tx.isReadOnly() && !isTempTable())
        throw new UnsupportedOperationException();

    RecordId rid = new RecordId(blk, currentSlot);

    String fieldname_rid = "fieldname:" + fldName + " rid:" + rid;
    Transaction.info value = tx.new info(fldName, rid, val);

    if (tx.info_get(fieldname_rid) == null) {
        if (!isTempTable())
            tx.concurrencyMgr().cmodifyRecord(new RecordId(blk, currentSlot));
        tx.info_put(fieldname_rid, value);
    } else {
        tx.info_put(fieldname_rid, value);
    }
}
```

At class ReadCommittedConcurrencyMgr, RepeatableReadConcurrencyMgr, SerializableConcurrencyMgr :

在這三個 class 裡面 都要新增這個函數，當今天要拿 clock 的時候 最上層 fileName 會拿 ixLock，第二層也是拿 ixLock 最下面是拿 clock，如果不這樣拿 都拿 clock 那其實下面還有其他人是可以拿別的 lock 的只是因為讓面都是 clock 所以就變得不能拿了

```
@Override
public void cmodifyRecord(RecordId recId) {
    lockTbl.ixLock(recId.block().fileName(), txNum);
    lockTbl.ixLock(recId.block(), txNum);
    lockTbl.cLock(recId, txNum);
}
```

結論:

因為我後來發生這個錯誤 然後看了一下助教的解釋 去檢查 getval 可是感覺沒有毛病 找不到錯 在最後那三個 class 要新增的那個函數 我其實不太確定 因為影片中之所以會這樣拿 是因為 ix 代表下面有 x lock 可是 clock 就不知道要怎麼拿了 而且這一版的是變成 S跟X可以相容 所以我不確定他上面的函式是不是也要更動 因為以上原因 所以只有屍體...

```
Exception in thread "main" java.lang.UnsupportedOperationException: Unspported SQL type: 0
        at org.vanilladb.core.sql.Type.newInstance(Type.java:53)
        at org.vanilladb.core.storage.metadata.TableMgr.getTableInfo(TableMgr.java:243)
        at org.vanilladb.core.storage.metadata.CatalogMgr.getTableInfo(CatalogMgr.java:48)
        at org.vanilladb.core.storage.metadata.statistics.StatMgr.refreshStatistics(StatMgr.java:123)
        at org.vanilladb.core.storage.metadata.statistics.StatMgr.initStatistics(StatMgr.java:136)
        at org.vanilladb.core.storage.metadata.statistics.StatMgr.<init>(StatMgr.java:72)
        at org.vanilladb.core.server.VanillaDb.initStatMgr(VanillaDb.java:241)
        at org.vanilladb.core.server.VanillaDb.init(VanillaDb.java:152)
```