Database Assignment 5 Report 1

TEAM 6 107062318 李俊逸 107062202 陳敬和 107062237 張濬洋

- How to implement
  1. 讓執行sql指令前能使用到ConservativeConcurrencyMgr
     把TxnMgr Serializable Isolation Level case中的SerializableConcurrencyMgr，換成ConservativeConcurrencyMgr，把benchmark中使用的ConcurrencyMgr換成我們的實作。

```
switch (isolationLevel) {
case Connection.TRANSACTION_SERIALIZABLE:
    try {
        Class<?> partypes[] = new Class[1];
        partypes[0] = Long.TYPE;
        Constructor<?> ct = ccConcurMgrCls.getConstructor(partypes);
        concurMgr = (ConcurrencyMgr) ct.newInstance(new Long(txNum));
    } catch (Exception e) {
        e.printStackTrace();
    }
    break;
```

  2. ConservativeConcurrencyMgr實作
     繼承ConcurrencyMgr，除了rollback和commit要release lock之外，其他function都不做事（ConservativeConcurrency的特性就是開始執行之前拿好所有lock，直到tx結束）。此外，我們新增一個getReadWriteLock的function，用以SP在執行SQL前先拿到所需的Lock。

```
public synchronized void lockReadWriteRecordIds(ArrayList<RecordId> readRecordId
                                                ArrayList<RecordId> writeRecordIds) throws In
```

  3. MicroTxnProc實作
     (a.) 建立item_id對應record_id的map(ItemIdToRecordId)
     executesql()之前需要拿到所需資源相對應的Lock，而每個tx在執行前已先給定所需record的i_id。我們讓第一個tx執行executesql()前，拿到item table的RecordFile並建立一個map，讓每個i_id對應到其recordId。之後每個tx就可以依照給定的i_id建立對應所需的Read(Write)RecordIdList（如第三張圖所示），在實際執行前再根據這兩個List拿Lock。

```
lic class MicroTxnProc extends StoredProcedure<MicroTxnProcParamHelper> {

 private static HashMap<Integer, RecordId> ItemIdToRecordId = new HashMap<In
 private static Boolean isBulitMapping = false;
 private ArrayList<RecordId> readRecordIdList = new ArrayList<RecordId>();
 private ArrayList<RecordId> writeRecordIdList = new ArrayList<RecordId>();
```

```
TableInfo tblInfo = VanillaDb.catalogMgr().getTableInfo(tblName: "item", tx);
RecordFile recordFile = tblInfo.open(tx, doLog: false);
recordFile.beforeFirst();
while(recordFile.next()) {
    int itemId = (int) recordFile.getVal(fldName: "i_id").asJavaVal();
    ItemIdToRecordId.put(itemId, recordFile.currentRecordId());
}
```

```
public void getReadWriteItemRecordIds() { // TODO: tx # 4 cannot get its corr
    // while (!isBulitMapping) ; // busy waiting until the mapping is built
    try {
        MicroTxnProcParamHelper paramHelper = getParamHelper();
        int[] readItemIdList = paramHelper.getReadItemIdList();
        int[] writeItemIdList = paramHelper.getWriteItemIdList();
        for(int i=0; i<readItemIdList.length; i++) {
            readRecordIdList.add(ItemIdToRecordId.get(readItemIdList[i]));
        }
        for(int i=0; i<writeItemIdList.length; i++) {
            writeRecordIdList.add(ItemIdToRecordId.get(writeItemIdList[i]));
        }
```

(b.) tx執行SQL前，用RecordIdList向ConcurrencyMgr拿lock(lockReadWriteRecordIds)

```
@Override
protected void executeSql() {
    MicroTxnProcParamHelper paramHelper = getParamHelper();
    Transaction tx = getTransaction();
    ConservativeConcurrencyMgr conserConcurMgr =
        (ConservativeConcurrencyMgr) tx.concurrencyMgr();
    bulidItemIdToRecordIdMapping();
    getReadWriteItemRecordIds();
    // get all locks
    try {
        conserConcurMgr.lockReadWriteRecordIds
            (readRecordIdList, writeRecordIdList);
    } catch (Exception e) {
        e.printStackTrace();
        tx.rollback();
    }
```

- The challenge of implementing conservative locking for TPC-C benchmark
    1. 在執行TPC-C時，需要存取各個table的id，如wid, did等，來取得對應的recordId。而針對每一種id都要額外建立lock機制，以及id與disk recordId之間的map
    2. TPC-C的資料量遠比micro benchmark大，建maps所需空間可能無法同時放入記憶體中，需要實作每個map在記憶體與硬碟之間的swap功能
    3. TPC-C執行的SQL指令較micro benchmark複雜，如predicate同時有三個table的id、有INSERT指令等等。

- Experiments

  Environment : MacBook Air (M1, 2020), Apple M1 chip, 16 GB RAM, 256 GB SSD, macOS Big Sur Version 11.2

1. 比較conservativeLock實作前後的效能差異:
   a. Settings

#RTE	RW_txn_rate	Read count
20	0.5	1000

   b. Results

	Throughtput
Origin code	1707
Our code	3613

   c. Origin Code Experiment Screenshot

```
[karta1502545@lijunyideMacBook-Air benchmark_results % cat 20220517-185443-microb]
ench.txt
# of txns (including aborted) during benchmark period: 5701
MICRO_TXN - committed: 1707, aborted: 3994, avg latency: 259 ms
```

   d. Our Code Experiment Screenshot

```
[karta1502545@lijunyideMacBook-Air benchmark_results % cat 20220517-190110-microb]
ench.txt
# of txns (including aborted) during benchmark period: 3613
MICRO_TXN - committed: 3613, aborted: 0, avg latency: 328 ms
```

   e. Explanation

      由於conservative lock的機制在一個tx需要拿大量的lock的情況下表現更佳，所以我們將readCount調成1000，以凸顯conservative lock會避免serializable會發生deadlock的狀況。

2. Observe and discuss the impact of buffer pool size to your new system
   a. Settings

#RTE	RW_txn_rate	Read count
20	0.5	1000

   b. Results

BufferPoolSize	throught
128	3465
256	3559
1024	3613

c. Experiment Screenshot (128)

```
karta1502545@lijunyideMacBook-Air benchmark_results % cat 20220517-195703-microb
ench.txt
# of txns (including aborted) during benchmark period: 3465
MICRO_TXN - committed: 3465, aborted: 0, avg latency: 342 ms
TOTAL - committed: 3465, aborted: 0, avg latency: 342 ms%
```

d. Experiment Screenshot (256)

```
karta1502545@lijunyideMacBook-Air benchmark_results % cat ourCode_20RTE_1000read
Count_halfRWrate_256bufferPoolSize.txt
# of txns (including aborted) during benchmark period: 3559
MICRO_TXN - committed: 3559, aborted: 0, avg latency: 331 ms
TOTAL - committed: 3559, aborted: 0, avg latency: 332 ms%
```

e. Experiment Screenshot (1024)

```
karta1502545@lijunyideMacBook-Air benchmark_results % cat ourCode_20RTE_1000read
Count_halfRWrate.txt
# of txns (including aborted) during benchmark period: 3613
MICRO_TXN - committed: 3613, aborted: 0, avg latency: 328 ms
TOTAL - committed: 3613, aborted: 0, avg latency: 328 ms%
```

f. Explanation

BufferPoolSize越大，代表同一時刻，系統可以接受更多的Block在記憶體中，減少I/O次數，所以系統運行效能會更好。