1 SMALLBANK BENCHMARK

We add the ***VID*** field, which indicates the version changes at the tuple granularity. The SQL statement pseudocode of SmallBank benchmark for reference.

```
DepositChecking(CNAME, AMOUNT):
      SELECT CustomerID into :CID
        FROM Accounts
        WHERE CustomerName = :CNAME;

      UPDATE Checking
        SET BAL = BAL + :AMOUNT, VID = VID + 1
        WHERE CustomerID = :CID
        RETURNING VID;
      COMMIT;

WriteCheck(CNAME, AMOUNT):
      SELECT CustomerID into :CID
        FROM Accounts
        WHERE CustomerName = :CNAME;

      SELECT BAL into :a, VID
        FROM Savings
        WHERE CustomerID = :CID;

      SELECT BAL into :b, VID
        FROM Checking
        WHERE CustomerID = :CID;

      IF :a < :AMOUNT
        UPDATE
           Checking
           SET BAL = BAL - (:AMOUNT - 1), VID = VID + 1
           WHERE CustomerID = :CID;
      ELSE
        UPDATE
           Checking
           SET BAL = BAL - :AMOUNT, VID = VID + 1
           WHERE CustomerID = :CID;
      END IF;
      COMMIT;

TransactSavings(CNAME, AMOUNT):
      SELECT CustomerID into :CID
        FROM Accounts
        WHERE CustomerName = :CNAME;

      UPDATE Savings
        SET BAL = BAL + :AMOUNT, VID = VID + 1
        WHERE CustomerID = :CID;
        RETURNING VID;
      COMMIT;
```

```
Amalgamate(CID1, CID2):
      SELECT *
        FROM Account
        WHERE CustomerID = :CID1;

      SELECT *
        FROM Accounts
        WHERE CustomerID = :CID2;

      UPDATE Savings
        AS new SET BAL = 0, VID = VID + 1
        FROM Savings AS old
        WHERE new.CustomerId = :CID1;
              AND old.CustomerID = new.CustomerID
        RETURNING old.BAL into :a AND VID;

      UPDATE Checking
        AS new SET BAL = 0, VID = VID + 1
        FROM Checking AS old
        WHERE new.CustomerId = :CID1;
              AND old.CustomerID = new.CustomerID
        RETURNING old.BAL into :b AND VID;

      UPDATE Savings
        SET BAL = BAL + (:a + :b), VID = VID + 1
        WHERE CustomerId = :CID2;
        RETURNING VID;
      COMMIT;

Balance(CNAME):
      SELECT CustomerID into :CID
        FROM Accounts
        WHERE CustomerName = :CNAME;

      SELECT BAL, VID into :a, VID
        FROM Savings
        WHERE CustomerID = :CID;

      SELECT BAL + :a, VID
        FROM Checking
        WHERE CustomerID = :CID;
      COMMIT;
```

2 YCSB-T BENCHMARK

We add the ***VID*** field, which indicates the version changes at the tuple granularity. The SQL statement pseudocode of YCSB benchmark for reference.

```
DeleteRecord(KEYNAME):
      DELETE FROM Usertable
        WHERE YCSB_KEY = :KEYNAME;
      COMMIT;

InsertRecord(KEYNAME, VALS):
      INSERT INTO Usertable
        VALUES (:KEYNAME, :VALS[1], :VALS[2],
                :VALS[3], :VALS[4], :VALS[5],
                :VALS[6], :VALS[7], :VALS[8],
                :VALS[9], :VALS[10]);
      COMMIT;

ReadRecord(KEYNAME, VALS):
      SELECT *
        FROM Usertable
        WHERE YCSB_KEY = :KEYNAME;
      COMMIT;

ReadWriteRecord(KEYNAMES, VALLISTS, i, j):
      IF :i == 1:
          FOR :KEYNAME in :KEYNAMES {
              SELECT VID, FIELD1
                FROM Usertable
                WHERE YCSB_KEY = :KEYNAME;
          }
      ELSE IF :i == 2:
          FOR :VALS in :VALLISTS, :KEYNAME in :KEYNAMES{
              UPDATE Usertable
                SET VID = VID + 1,
                    FIELD1 = :VALS[1], FIELD2 = :VALS[2],
                    FIELD3 = :VALS[3], FIELD4 = :VALS[4],
                    FIELD5 = :VALS[5], FIELD6 = :VALS[6],
                    FIELD7 = :VALS[7], FIELD8 = :VALS[8],
                    FIELD9 = :VALS[9], FIELD10 = :VALS[10]
                WHERE YCSB_KEY = :KEYNAME;
          }
      END IF;
      COMMIT;
```

```
ScanRecord(KEYNAME, START, COUNT):
      SELECT *
        FROM Usertable
        WHERE YCSB_KEY > :START
            AND YCSB_KEY < :START + :COUNT
      COMMIT;

UpdateRecord(KEYNAME, VALS):
      UPDATE Usertable
        SET FIELD1 = :VALS[0], FIELD2 = :VALS[1],
            FIELD3 = :VALS[2], FIELD4 = :VALS[3],
            FIELD5 = :VALS[4], FIELD6 = :VALS[5],
            FIELD7 = :VALS[6], FIELD8 = :VALS[7],
            FIELD9 = :VALS[8], FIELD10 = :VALS[9]
        WHERE YCSB_KEY = :VALS[10];
      COMMIT;

ReadModifyWriteRecord(KEYNAME, VALS):
      SELECT *
        FROM Usertable
        WHERE YCSB_KEY = :KEYNAME FOR UPDATE;

      UPDATE Usertable
        SET FIELD1 = :VALS[1], FIELD2 = :VALS[2],
            FIELD3 = :VALS[3], FIELD4 = :VALS[4],
            FIELD5 = :VALS[5], FIELD6 = :VALS[6],
            FIELD7 = :VALS[7], FIELD8 = :VALS[8],
            FIELD9 = :VALS[9], FIELD10 = :VALS[10]
        WHERE YCSB_KEY = :KEYNAME;
      COMMIT;
```

3 TPC-CKV BENCHMARK

We add the ***VID*** field, which indicates the version changes at the tuple granularity. The SQL statement pseudocode of TPC-Ckv benchmark for reference.

```
OrderStatus(WID, DID, CID, OID):
      SELECT C_INFO, C_BALANCE, VID
        FROM Customer
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND CustomerID = :CID;

      SELECT O_STATUS, VID
        FROM Orders
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND OrderID = :OID;

      SELECT OL_DELIVERY_INFO, OL_QUANTITY, VID
        FROM OrderLine
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND OrderID = :OID;
      COMMIT;

Payment(WID, DID, CID, AMOUNT):
      UPDATE Warehouse
        SET W_YTD = W_YTD + :AMOUNT, VID = VID + 1
        WHERE WarehouseID = :WID
        RETURNING VID;

      UPDATE District
        SET D_YTD = D_YTD + :AMOUNT
         WHERE WarehouseID = :WID AND DistrictID = :DID;

      UPDATE Customer
        SET C_BALANCE = C_BALANCE + :AMOUNT, VID = VID + 1
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND CustomerID = :CID
        RETURNING VID;
      COMMIT;

Delivery(WID, DID, OID, CID, PRICE):
      UPDATE Orders
        SET OSTATUS = 'delivered', VID = VID + 1
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND OrderID = :OID
        RETURNINIG VID

      UPDATE OrderLine
        SET OL_DEL_INFO = 'delivered', VID = VID + 1
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND OrderID = :OID
        RETURNING VID

      UPDATE Customer
        SET C_BALANCE = C_BALANCE - :PRICE, VID = VID + 1
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND CustomerID = :CID
        RETURNING VID;
      COMMIT;
```

```
NewOrder(WID, DID, CID, ITEMS):
      SELECT W_INFO, VID
        FROM Warehouse
        WHERE WarehouseID = :WID;

      UPDATE District
        SET NextOrderID = NextOrderID + 1
        WHERE WarehouseID = :WID AND DistrictID = :DID
        RETURNING NextOrderID into :OID;

      SELECT C_INFO, VID
        FROM Customer
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND CustomerID = :CID;

      UPDATE Orders
        SET OrderStatus := 'renewed'
        WHERE WarehouseID = :WID AND DistrictID = :DID
              AND OrderID = :OID;

      :OLID = 1
      FOR :IID, :QUANTITY in :ITEMS{
            UPDATE Stock
            SET Quantity = Quantity - :quantity
            WHERE WarehouseID = :WID AND ItemID = :IId

          UPDATE Orderline
            SET INFO = 'renewed'
            WHERE WarehouseID = :WID
                  AND DistrictID = :DID
                  AND OrderID = :OID;
                  AND OrderLineID = :OLID;
      }
      COMMIT;

StockLevel(WID, IID):
      SELECT S_QUANTITY
        SFROM Stock
        WHERE WarehouseID = :WID AND ItemID = :IID;
      COMMIT;
```