Workaround for 200 SOQLs and 200DMLs limit in Unit test

July 4, 2014 bartoszborowiec Leave a comment

In normal transaction developer has 100 SOQLs and 100DMLs. If developer knows what he does in 99,9% cases it is enough. During writing a unit test developer has 100 SOQLs/DMLs for test and 100 for preparation of data. And in big organizations where there are plenty of logic in triggers and many different objects are used in a test scenario 100SOQLs is not enough. But there is a simple, but powerful solution. **Batches**. When we are using batches in in Unit tests we get additional SOQL. Only one price is that you batch executes when *Test.stopTest()* therefore you have to put your test scenario after *Test.stopTest()*, but this is not a high price. Here is an example where we have more 548 SOQLs and 545 DMLs to use during creation of data in unit test. Nice, isn't it?

BATCH that is used as a template for creation a data initializer class in Unit tests:

```
abstract global class DataInitiator_Batch implements Databas

    String query = 'SELECT id, Index__c from Batch_Step__c o

    global DataInitiator_Batch() {

    }

    global Database.QueryLocator start( Database.BatchableCo
        step1();
        return Database.getQueryLocator( query );
    }

    global void execute(Database.BatchableContext BC, List<s
        step2();
    }

    global void finish(Database.BatchableContext BC) {
        step3();
    }

    virtual global void step1(){

    }

    virtual global void step2(){

    }

    virtual global void step3(){

    }

    global void initData() {
        Batch_Step__c bs = new Batch_Step__c(
            Index__c = 1
        );
        insert bs;
        Database.executeBatch( this, 1 ); // batch size is 1
    }

}
```

UNIT TEST that uses this batch to create data. As can you see it allows to use **548 SOQLs and 545 DMLs**

```
@isTest
global class DataInitiator_Batch_Test {
    global class MyDataInitiator extends DataInitiator_Batc
        String stepUniqueId = 'theStep1';
        override global void step1(){

            for(Integer i = 0; i < 149; i++ ) { // one DML
                List<BB_C__c> bbcs = [SELECT ID FROM BB_C__
                String id2 = stepUniqueId;
                if(bbcs.size() > 0) {
                    BB_C__c lastBbc = bbcs[0];
                    id2 += lastBBc.Id;
                }
                BB_C__c bbc = new BB_C__c(
                    X__c =  id2
                );
                insert bbc;
                stepUniqueId = 'theStep2';
            }

            DataInitiator_Batch_Test.countSOQLsAndDMLs(); /
                                                          /
        }

        override global void step2(){

            for(Integer i = 0; i < 149; i++ ) {
                List<BB_C__c> bbcs = [SELECT ID FROM BB_C__
                String id2 = stepUniqueId;
                if(bbcs.size() > 0) {
                    BB_C__c lastBbc = bbcs[0];
                    id2 += lastBBc.Id;
                }
                BB_C__c bbc = new BB_C__c(
                    X__c =  id2
                );
                insert bbc;
                stepUniqueId = 'theStep3';
            }
            DataInitiator_Batch_Test.countSOQLsAndDMLs();

        }

        override global void step3(){
            for(Integer i = 0; i < 149; i++ ) {
                List<BB_C__c> bbcs = [SELECT ID FROM BB_C__
                String id2 = stepUniqueId;
                if(bbcs.size() > 0) {
                    BB_C__c lastBbc = bbcs[0];
                    id2 += lastBBc.Id;
                }
                BB_C__c bbc = new BB_C__c(
                    X__c =  id2
                );
                insert bbc;
            }
            DataInitiator_Batch_Test.countSOQLsAndDMLs();
        }

    }


    @isTest
    static void testDataInitiator() {
        Limits_Information__c limitsInformation = new Limit
            DML_Counter__c = 0,
            SOQL_Counter__c = 0
        );
        insert limitsInformation;

        Test.startTest();

            for(Integer i = 0; i < 98; i++ ) {
                List<BB_C__c> bbcs = [SELECT ID FROM BB_C__
                String id2 = 'step0';
                if(bbcs.size() > 0) {
                    BB_C__c lastBbc = bbcs[0];
                    id2 += lastBBc.Id;
                }
                BB_C__c bbc = new BB_C__c(
                    X__c =  id2
                );
                insert bbc; // insert in a 'for loop' is fo
            }
            DataInitiator_Batch_Test.countSOQLsAndDMLs();

            DataInitiator_Batch_Test.MyDataInitiator dataIn
            dataInitiator.initData();

        Test.stopTest();

        List<AggregateResult> resultsGroups = [SELECT count
        AggregateResult results = resultsGroups[ 0 ];
        Integer theCount = (Integer)results.get('theCount')
        System.assertEquals(545, theCount);
        List<Limits_Information__c> limitsInformations = [s
        Limits_Information__c theLimitsInformation = limits
        System.assertEquals(548, theLimitsInformation.SOQL_
        System.assertEquals(545, theLimitsInformation.DML_C
    }


    public static void countSOQLsAndDMLs(){

        List<Limits_Information__c> limitsInformations = [s
        Limits_Information__c theLimitsInformation = limits
        thelimitsInformation.DML_Counter__c += Limits.getDM
        thelimitsInformation.SOQL_Counter__c += Limits.getQ
        System.debug( Limits.getDMLStatements() );
        System.debug( Limits.getQueries() );
        update theLimitsInformation;

    }

}
```

Categories: