Disappearing soql queries

November 16, 2014 bartoszborowiec 1 comment

Today I am going to write about dispersing soql queries. As you know there is an governor limit that only 100 queries can be used in one transaction. In proper written system that is usually enough but sometimes you will reach this limit. And you going to investigate code to find a place that can be optimized. Here is a very tricky example:

We have two objects:

1. **Alfa__c** – it does not have any custom field
2. **Beta__c** – it has two fields:
    1. **AlfaCounter__c** – count of alfa objects is written to this object
    2. **CounterOfChanges__c** – count of changes of given Beta__c object

>**AlfaCounter__c** is updated by a following trigger:

```
trigger Beta on Beta__c (before insert, before update,  befc

    if ( Trigger.isBefore && Trigger.isInsert ) {
        beforeInsert();
    } else if ( Trigger.isBefore && Trigger.isUpdate ){
        beforeUpdate();
    }

    private static void beforeInsert(){
        execute();
    }

    public static void beforeUpdate(){
        execute();
    }

    private static void execute() {
        List<Alfa__c> alfas = [
            SELECT Id
            FROM Alfa__c
        ];
        Integer count = alfas.size();
        for( Beta__c theBeta : Trigger.new ){
            theBeta.alfaCounter__c = count;
        }
    }
}
```

CounterOfChanges__c is updated by a following workflow:

```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow xmlns="http://soap.sforce.com/2006/04/metadata">
    <fieldUpdates>
        <fullName>UpdateCounter</fullName>
        <field>CounterOfChanges__c</field>
        <formula>CounterOfChanges__c  + 1</formula>
        <name>UpdateCounter</name>
        <notifyAssignee>false</notifyAssignee>
        <operation>Formula</operation>
        <protected>false</protected>
    </fieldUpdates>
    <rules>
        <fullName>OnChange</fullName>
        <actions>
            <name>UpdateCounter</name>
            <type>FieldUpdate</type>
        </actions>
        <active>true</active>
        <formula>TRUE</formula>
        <triggerType>onCreateOrTriggeringUpdate</triggerType
    </rules>
</Workflow>
```

Lets execute following unit test:

```
@isTest
private class BetaTest {

    @isTest
    static void checkHowManySOQL() {
        Alfa__c[] alfas = new List<Alfa__c>();
        for( Integer i=0; i<3; i++ ){
            Alfa__c  theAlfa = new Alfa__c( Name='Alfa' + i
            alfas.add( theAlfa );
        }
        insert alfas;

        Test.startTest();

            Beta__c beta = new Beta__c( Name = 'Beta');
            insert beta;
            System.assertEquals( 1, Limits.getQueries() );

        Test.stopTest();
    }

}
```

Unfortunately assert is not fulfilled. Function **Limits.getQueries()** returns **2**.

Why??????? Answer will be very simple, after you read following documentation.
https://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers_order_of_execution.htm
Take a look on point **13**.
If the record was updated with workflow field updates, fires before update triggers and after update triggers one more time (and only one more time), in addition to standard validations. Custom validation rules are not run again.

Worflow fired trigger second time. In general using workfolow and triggers for the same object has to be done with a extremely caution.