Reducing Salesforce SOQL queries by using static variables

by Patrick Connelly posted on April 04, 2012

The more moving pieces you have with triggers and classes the more you want to reduce the number of SOQL queries. One way to do this is to have Utility classes that do a lot of the heavy lifting. The problem with this is that you don't want to call a utility method that does a query every time, because if you call it from different triggers you'll end up with multiple calls. This is where overloading static variables can come in.

The Problem

Lets say you have a trigger on a Contact that needs information from our mostly static MyObject__c and then the Contact trigger then updates a Case. The Case trigger also need information from the MyObject__c. Normally this would require two SOQL queries even if it was in a utility class. We can use some of the built-in functionality in Apex to overload a static variable.

The Class

```
public with sharing class MyObjectUtils {
     public static List<MyObject__c> myObjectList {
          get {
               if (myObjectList == null) {
                    myObjectList = [
                         select Name
                         from MyObject__c
                    ];
               }

               return myObjectList;
          }
          set;
     }

     public static Map<String, MyObject__c> myObjectMap {
          get {
               if (myObjectMap == null) {
                    myObjectMap = new Map<String, MyObject__c>();
                    for (MyObject__c obj: myObjectList) {
                         myObjectMap.put(obj.Name, obj);
                    }
               }

               return myObjectMap;
          }
          set;
     }
}
```

The Implementation

In the trigger where we want to use the MyObject instance we can do the following

```
MyObject__c obj = MyObjectUtils.myObjectMap.get('Foo');
```

Now, the next time the map or list are used in the same execution we will have it "cached" and will not have to make an additional query.