Count large number of records (more than 50,000)

Chun Wu August 28, 2014 6 Comments

In a Salesforce org that has more than 50,000 records of an object, the following simple count() query will still hit the Salesforce governor limit: “System.LimitException: Too many query rows: 50001”

```
System.debug('total: ' + [select count() from Contact]);
```

Even though it seems that the count function does not need to traverse the whole Contact table, it still does, for specific reasons like checking sharing settings on records so that different users may get different number of records. So is there a way of retrieving the total number of records that are more than 50,000. There are a few, each of which has their cons and pros. Surprisingly, for any of these methods, such simple task would need more coding than we expected.

Method 1: Use Visualforce page with readOnly attribute set to true

Controller class:

```
public class StatsController {
    public Integer numberOfContacts {
        get {
            if (numberOfContacts == null) {
                numberOfContacts = [select count() from Contact];
            }
            return numberOfContacts;
        }
        private set;
    }
}
```

Visualforce page:

```
<apex:page controller="StatsController" readOnly="true">
    <p>Number of Contacts: {!numberOfContacts}</p>
</apex:page>
```

Then you will be able to see the result when you access this page in the browser. Note that the readOnly attribute has to be set to “true”, otherwise you will still get a Visualforce error complaining “Too many query rows: 50001 “. However, you won’t be able to create a simple controller and a page in Production orgs.

Method 2: Batchable class

```
public class ContactBatchable implements Database.Batchable<sObject>, Database.Stateful {
    Integer total = 0;

    public Database.QueryLocator start(Database.BatchableContext BC){
        return Database.getQueryLocator('select Id from Contact');
    }

    public void execute(
            Database.BatchableContext BC,
            List<sObject> scope){
        total += scope.size();
    }

    public void finish(Database.BatchableContext BC){
        System.debug('total: ' + total);
    }
}
```

And then execute the following statement in developer console.

```
Database.executeBatch(new ContactBatchable(), 2000);
```

This is going to be an asynchronous apex job so it can take time. When it is finished, you will be able to see a log with category “Batch Apex” at the top. The debug statement prints info into that log. You will also see a few logs with category “SerialBatchApexRangeChunkHandler” which is the log for each batch.

In the ContactBatchable the start method has to use the query locator, so that it can retrieve the records up to 50 million. (after 50 million? Ask Salesforce support). If you use an iterable in the batch class, the governor limit for the total number of records retrieved by SOQL queries (50,000 for the moment) is still enforced.

If the start method of the batch class returns a QueryLocator, the optional scope parameter of Database.executeBatch can have a maximum value of 2,000. If set to a higher value, Salesforce chunks the records returned by the QueryLocator into smaller batches of up to 2,000 records.

Still, this method cannot be applied to production orgs as you won’t be able to create an Apex class in Production org.

Method 3: Make a http request using the REST API

By far, this seems to be the simplest way of achieving this and it can be used in production orgs. In developer console, run the following statements:

```
HttpRequest req = new HttpRequest();
req.setEndpoint('https://'+URL.getSalesforceBaseUrl().getHost()+'/services/data/v20.0/query/?q=SELECT+Id+from+Contact');
req.setMethod('GET');

string autho = 'Bearer '+ userInfo.getsessionId();
req.setHeader('Authorization', autho);

Http http = new Http();
HTTPResponse res = http.send(req);
string response = res.getBody();
string total = response.substring(response.indexOf('totalSize":') + 11, response.indexOf(','));
system.debug('Total: '+ total);
```

You will need to add a remote site setting with URL set to your production org’s URL (say **https://na11.salesforce.com**) in the setup.