Apex

(Records in the Database)

Exercise Guide


salesforce

Table of Contents

Exercise 4-1: Exploring Query Relationships Using Force.com IDE Tools

Goals:

1. Describe how queries from child to parent are different than queries from parent to child.
2. Use the schema browser to construct and execute queries.
3. Use the Execute Anonymous tool to run code against your server.

Scenario:

Universal Containers will be requiring developers to write code that uses SOQL. The company wants developers to understand the tools available in the Force.com IDE for executing and testing SOQL, as well as the language constructors, such as `for` loops, which support working with SOQL.

Tasks:

1. Use the schema browser to construct a query from the child `Job_Application__c` that pulls fields from the parent `Position__c`.
2. Create a code block to perform the query that was built in the schema browser on the `Job_Application__c` sObject and to cycle through the results.
3. Use the schema browser to construct a query from the parent `Position__c` that pulls fields from the child `Job_Application__c`.
4. Create a code block to perform the query that was built in the schema browser on the `Position__c` sObject and to cycle through the results.

Time:

30 minutes

Instructions:

1. Use the schema browser to construct a query from the child `Job_Application__c` that pulls fields from the parent `Position__c`.
   A. In Eclipse, open your project in the `Package Explorer` pane, then double-click **salesforce.schema**, found at the bottom of the project, to open the schema browser.
   B. In the Schema pane, open **Job_Application__c | Fields**, then select the `Name__c` and `Name` checkboxes.
   C. In the Query Results pane, click **Run Me**.
   D. Within **Job Application__c | Fields**, open **Position__c | Type Data – reference | Reference To | Position__c | Fields**, then select the **Title** checkbox. Notice that

`Position__c` is automatically added to the query as well as the **Name** field. You can remove `Position__c` from the query in the Query Results pane if you wish.

Note: Name is a required field for all sObjects. However, when designing the sObject, you can specify a different label for this field. `Title` is the label for **Name** field of the `Position__c` sObject, which is why `Name` is added to the query when you select the **Title** checkbox.

E. In the Query Results pane, click **Run Me**.

F. Click on a returned result to open up a dialog box showing the field retrieved from the parent sObject, and then close the dialog box.

2. Create a code block to perform the query that was built in the schema browser on the `Job_Application__c` sObject and to cycle through the results.

A. Copy the SOQL query from the Query Results pane.

B. In the **Source to Execute** text area of the Execute Anonymous tab:

i. Remove any code that is currently in place.

ii. Declare a `List` object called `jobApps` whose elements are of type `Job_Application__c`, and give it the initial value of your query.

iii. Create a `for` loop that cycles through the `List` object, outputting all the fields you have retrieved using the `System.debug()` method.

C. Click **Execute Anonymous** and examine the results of your code.

D. Modify your code so that you use the query directly in the `for` loop, rather than creating a `List` object to iterate through.

E. Click **Execute Anonymous** to test your revision.

Note: If you want to keep your code, you should copy it to a separate document now.

3. Use the schema browser to construct a query from the parent `Position__c` that pulls fields from the child `Job_Application__c`.

A. In the Schema pane of the schema browser, deselect `Job_Application__c` and close up that part of the tree by clicking on the arrow next to the name.

B. Open **Position__c | Fields**, then select the **Name** checkbox.

C. Open **Position__c | Child Relationships | Job_Application__c | Fields**, then select the **Name** checkbox within that list.

D. In the Query Results pane, click **Run Me**.

E. Double-click on a returned result to open up a dialog box showing the fields retrieved from the child sObjects for that result, then close the dialog box.

4. Create a code block to perform the query that was built in the schema browser on the `Position__c` sObject and to cycle through the results.

A. Copy the SOQL query from the Query Results pane.

B. In the **Source to Execute** text area of the Execute Anonymous tab:

i. Remove any code that is currently in place.

ii. Create a `for` loop for iterating through the query result returned.

iii. Within the `for` loop, create a second for loop for iterating through the returned field `Job_Applications__r`, which is a list of `Job_Application__c` sObjects.

iv. Within the inner `for` loop, output the fields you have retrieved from both the `Position__c` sObject and the associated `Job_Applications__c` sObject.

C. Click **Execute Anonymous** to test your code.

Note: If you want to keep your code, you should copy it to a separate document now.

Review

1. If you performed an `update` statement in the Execute Anonymous pane, would the data in the associated org be modified?

2. Why is it necessary to have a nested `for` loop to go through results when a query from a parent sObject includes fields from the child sObject, but not when the query is from the child sObject and includes parent fields?

Exercise 4-2: Creating an Automated Chatter Subscription (Part 1)

Goal:

Write a class to support the automated creation of a Chatter subscription to follow an object.

Scenario:

Hiring managers at Universal Containers want to follow positions for which they are responsible via Chatter. They would like this to happen automatically.

Tasks:

1. Create a new Apex class named `SubscriptionsClass` with a method that subscribes hiring managers to their positions.
2. Enable feed tracking on the `Position__c` sObject.
3. Test `SubscriptionsClass` using Execute Anonymous.

Time:

20 minutes

Instructions:

1. Create a new Apex class named `SubscriptionsClass` with a method that subscribes hiring managers to their positions.
   A. In the Force.com IDE, right-click the project folder and select **New | Apex Class**.
      i. **Name:** `SubscriptionsClass`
      ii. Click **Finish.**
   B. Replace the contents of `SubscriptionsClass` with the contents of the lab file `4-2.SubscriptionsClass.txt` from the `Exercises` folder.
   C. In your browser go to http://developer.force.com, and choose **Technical Library | Cheat Sheets.**
   D. Under **Collaboration: Chatter Cheat sheets**, download the Chatter Cheat Sheet.
   E. Review the Chatter Data Model, particularly `EntitySubscription` and its related data.
   F. (Optional) Search http://developer.force.com for Chatter Code Recipes to see if you can find a recipe that will help you to complete the `TODOs`.
   G. In Eclipse, complete the sections marked `TODO` in `SubscriptionsClass`.
   H. Save your changes.

2. Enable feed tracking on the `Position__c` sObject.
   A. In Salesforce, navigate to **Setup | Build | Customize | Chatter | Feed Tracking | Position.**
      i. Select `Enable Feed Tracking`, `Status`, and `Sub-Status` checkboxes.
      ii. Click **Save.**
3. Test `SubscriptionsClass` using Execute Anonymous.
   A. In the Force.com IDE, in the **Source to Execute** text area of the Execute Anonymous tab:
      i. Remove any code that is currently in place.
      ii. Declare a `List` object called `positions` whose elements are of type `Position__c`, and give it the initial value of `SELECT id, Hiring_Manager__c FROM Position__c WHERE name='SW Engineer'`.
      iii. Write a `System.debug()` statement that outputs the number of elements in the `positions` list.
      iv. Write a System.debug() statement that outputs the number of `EntitySubscription` objects associated with the position you retrieved earlier by using the query `SELECT count() FROM EntitySubscription WHERE parentID in :positions`.
      v. Invoke the `HiringManagerSubscribeNewPosition` method of `SubscriptionsClass` passing it the `positions` list.
      vi. Write a `System.debug()` statement that outputs the number of `EntitySubscription` objects associated with the position you retrieved earlier by using the query `SELECT count() FROM EntitySubscription WHERE parentID in :positions`.
      vii. Click **Execute Anonymous** to execute your code.

      Note: If you want to keep your code, you should copy it to a separate document now.

Review

1. Under what circumstances should the `HiringManagerSubscribeNewPosition` method be called?

2. What Chatter related classes would you use to create a Chatter post if you wanted to inform someone that a new job application is available for a position, but not necessarily have them follow that object?