**SOQL For Loops**

SOQL for loops iterate over all of the sObject records returned by a SOQL query.

The syntax of a SOQL for loop is either:

|  |  |  |
| --- | --- | --- |
| 1 | for (variable : [soql\_query]) { | |
| 2 | code\_block |

|  |  |
| --- | --- |
| 3 | } |

or

|  |  |  |
| --- | --- | --- |
| 1 | for (variable\_list : [soql\_query]) { | |
| 2 | code\_block |

|  |  |
| --- | --- |
| 3 | } |

Both *variable* and *variable\_list* must be of the same type as the sObjects that are returned by the *soql\_query*. As in standard SOQL queries, the [*soql\_query*] statement can refer to code expressions in their WHERE clauses using the : syntax. For example:

|  |  |
| --- | --- |
| 1 | String s = 'Acme'; |
| 2 | for (Account a : [SELECT Id, Name from Account | |

|  |  |  |
| --- | --- | --- |
| 3 | where Name LIKE :(s+'%')]) { | |
| 4 | // Your code |

|  |  |
| --- | --- |
| 5 | } |

The following example combines creating a list from a SOQL query, with the DML update method.

|  |  |
| --- | --- |
| 01 | // Create a list of account records from a SOQL query |
| 02 | List<Account> accs = [SELECT Id, Name FROM Account WHERE Name = 'Siebel']; | |

|  |  |
| --- | --- |
| 03 |  |
| 04 | // Loop through the list and update the Name field | |

|  |  |  |
| --- | --- | --- |
| 05 | for(Account a : accs){ | |
| 06 | a.Name = 'Oracle'; |

|  |  |  |
| --- | --- | --- |
| 07 | } | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | // Update the database | |
| 10 | update accs; |

SOQL For Loops Versus Standard SOQL Queries

SOQL for loops differ from standard SOQL statements because of the method they use to retrieve sObjects. While the standard queries discussed in [SOQL and SOSL Queries](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_SOQL.htm) can retrieve either the count of a query or a number of object records, SOQL for loops retrieve all sObjects, using efficient chunking with calls to the query and queryMore methods of the SOAP API. Developers should always use a SOQL for loop to process query results that return many records, to avoid the limit on [heap size](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm#total_heap_size_limit_desc).

Note that queries including an [aggregate function](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_SOQL_agg_fns.htm) don't support queryMore. A run-time exception occurs if you use a query containing an aggregate function that returns more than 2,000 rows in a for loop.

SOQL For Loop Formats

SOQL for loops can process records one at a time using a single sObject variable, or in batches of 200 sObjects at a time using an sObject list:

* The single sObject format executes the for loop's <code\_block> once per sObject record. Consequently, it is easy to understand and use, but is grossly inefficient if you want to use data manipulation language (DML) statements within the for loop body. Each DML statement ends up processing only one sObject at a time.
* The sObject list format executes the for loop's <code\_block> once per list of 200 sObjects. Consequently, it is a little more difficult to understand and use, but is the optimal choice if you need to use DML statements within the for loop body. Each DML statement can bulk process a list of sObjects at a time.

For example, the following code illustrates the difference between the two types of SOQL query for loops:

|  |  |  |  |
| --- | --- | --- | --- |
| 01 | // Create a savepoint because the data should not be committed to the database | | |
| 02 | | Savepoint sp = Database.setSavepoint(); |

|  |  |
| --- | --- |
| 03 |  |
| 04 | insert new Account[]{new Account(Name = 'yyy'), | |

|  |  |
| --- | --- |
| 05 | new Account(Name = 'yyy'), |
| 06 | new Account(Name = 'yyy')}; | |

|  |  |
| --- | --- |
| 07 |  |
| 08 | // The single sObject format executes the for loop once per returned record | |

|  |  |
| --- | --- |
| 09 | Integer i = 0; |
| 10 | for (Account tmp : [SELECT Id FROM Account WHERE Name = 'yyy']) { | |

|  |  |  |
| --- | --- | --- |
| 11 | i++; | |
| 12 | } |

|  |  |  |
| --- | --- | --- |
| 13 | System.assert(i == 3); // Since there were three accounts named 'yyy' in the | |
| 14 | // database, the loop executed three times |

|  |  |
| --- | --- |
| 15 |  |
| 16 | // The sObject list format executes the for loop once per returned batch | |

|  |  |  |
| --- | --- | --- |
| 17 | // of records | |
| 18 | i = 0; |

|  |  |
| --- | --- |
| 19 | Integer j; |
| 20 | for (Account[] tmp : [SELECT Id FROM Account WHERE Name = 'yyy']) { | |

|  |  |  |
| --- | --- | --- |
| 21 | j = tmp.size(); | |
| 22 | i++; |

|  |  |
| --- | --- |
| 23 | } |
| 24 | System.assert(j == 3); // The list should have contained the three accounts | |

|  |  |  |
| --- | --- | --- |
| 25 | | // named 'yyy' |
| 26 | System.assert(i == 1); // Since a single batch can hold up to 200 records and, | | |

|  |  |  |
| --- | --- | --- |
| 27 | // only three records should have been returned, the | |
| 28 | // loop should have executed only once |

|  |  |
| --- | --- |
| 29 |  |
| 30 | // Revert the database to the original state | |

|  |  |
| --- | --- |
| 31 | Database.rollback(sp); |

Note

* The break and continue keywords can be used in both types of inline query for loop formats. When using the sObject list format, continue skips to the next list of sObjects.
* DML statements can only process up to 10,000 records at a time, and sObject list for loops process records in batches of 200. Consequently, if you are inserting, updating, or deleting more than one record per returned record in an sObject list for loop, it is possible to encounter runtime limit errors. See [Execution Governors and Limits](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm).
* You might get a QueryException in a SOQL for loop with the message Aggregate query has too many rows for direct assignment, use FOR loop. This exception is sometimes thrown when accessing a large set of child records (200 or more) of a retrieved sObject inside the loop, or when getting the size of such a record set. For example, the query in the following SOQL for loop retrieves child contacts for a particular account. If this account contains more than 200 child contacts, the statements in the for loop cause an exception.

|  |  |  |
| --- | --- | --- |
| 1 | for (Account acct : [SELECT Id, Name, (SELECT Id, Name FROM Contacts) | |
| 2 | FROM Account WHERE Id IN ('<ID value>')]) { |

|  |  |  |
| --- | --- | --- |
| 3 | List<Contact> contactList = acct.Contacts; // Causes an error | |
| 4 | Integer count = acct.Contacts.size(); // Causes an error |

|  |  |
| --- | --- |
| 5 | } |

* To avoid getting this exception, use a for loop to iterate over the child records, as follows.
* [view source](https://developer.salesforce.com/docs/#viewSource)
* [print](https://developer.salesforce.com/docs/#printSource)[?](https://developer.salesforce.com/docs/#about)

|  |  |  |
| --- | --- | --- |
| 1 | for (Account acct : [SELECT Id, Name, (SELECT Id, Name FROM Contacts) | |
| 2 | FROM Account WHERE Id IN ('<ID value>')]) { |

|  |  |
| --- | --- |
| 3 | Integer count=0; |
| 4 | for (Contact c : acct.Contacts) { | |

|  |  |  |
| --- | --- | --- |
| 5 | count++; | |
| 6 | } |

|  |  |
| --- | --- |
| 7 | } |