Maximize The Benefits Of Parallelism By Managing Locks

BLOG / SALESFORCE TECHNOLOGY

AMAR KULKARNI
NOVEMBER 22, 2017

Are you facing this issue? Are you processing records concurrently? Are you processing records in bulk? Do you have any of workflows, triggers, batches, roll-up summary operation on the object? Are you processing records having master-detail or lookup relationship? Ah Yes! You need a solution for such issues, let's understand first.

What is SaaS?

SaaS applications are web-based and run on defined provider's server. In which all users and applications share a single, common infrastructure and code base which is centralized. In order to protect shared resources, the locks are served.

How could this happen?

Let's take a simple example where there are 2 objects Account and Opportunity having a master-detail relationship. When we are loading records in bulk and batches are running concurrently say Opportunity and Account which are loading records concurrently where some accounts having a reference in the opportunity batch then it may cause row lock or unable to obtain exclusive access issues since they are locked out by detail object to ensure referential integrity.

How we can solve the issue?

Let's analyze each scenario of locking and understand how each can be avoided.

1. Master-Detail Relationship

As we saw in the above example, Salesforce locks the parent records to ensure referential integrity. In order to avoid such scenarios, we'll have to ensure that there are no parent records batches running concurrently instead batches can be run sequentially.

2. Lookup Relationship

Yes, this can happen for lookup relationship as well. Because when you are performing an operation on the base object's records salesforce locks the target lookup records to ensure referential integrity. For Lookups, it can be minimized by avoiding locks where we need to set "What to do if the lookup record is deleted?" field to "Clear the value of the field" under Lookup options. If you set this option, Salesforce will not lock the lookup records rather it only validates that the lookup values exist.

3. Roll-up Summary

As discussed earlier, Salesforce locks the parent records so it can update roll-up summary fields without any issues. If the child records that are a lookup to same parent record updated in concurrent batches that are running then it can cause the lock exception. In order to avoid lock exceptions, we may have to remove unnecessary roll-up fields or to run batches sequentially.

4. Triggers

There’s high possibility that trigger can cause lock issues. For example, if you are having one or more trigger which is performing an operation on other objects apart from the one you are inserting or updating it Salesforce locks those records. To avoid such scenario triggers can be turned off while loading records in bulk.

5. Workflow Rules

Salesforce locks the records for which workflow updates the field. When those records also getting updated by the other thread concurrently then it may cause lock exceptions. So to avoid this set the criteria of your workflow rules such a way they do not fire during a bulk load.

6. Platform Events

If you have already subscribed to platform events such as in processes, flows, Apex triggers or CometD clients which are active and having transactions on records other than the one you are performing DML operation can cause lock errors. Best practice is to have them deactivated if you are making transactions in bulk.

Key Takeaway:

- Bulk loads has to be Mutually exclusive to avoid concurrent processes which are having references in the other.
- Process records sequentially if possible to avoid such issues however it may impact on the performance.
- If you want to process records concurrently and there is still a possibility of lock exception, have a retry attempt logic built.