Salesforce UNABLE_TO_LOCK_ROW: unable to obtain exclusive access to this record

Unable to lock row - Record currently unavailable errors

Description

When a record is being updated or created, we place a lock on that record to prevent another operation from updating the record at the same time and causing inconsistencies on the data.

These locks normally last for a few seconds and when the lock is released, other operations can do whatever processing they are supposed to do on the record in question. However, a given transaction can only wait a maximum of 10 seconds for a lock to be released, otherwise it will time out.

Resolution

What and when records get locked depends on the operation you are performing and the main record you are working on. The **Force.com Record Locking Cheatsheet** provides detailed information on this and it's highly recommended that you familiarize yourself with its contents.

Common Scenarios that prevent unlocking record

a. Email-To-Case

When an email is processed by email-to-case, triggers on the email message object or related objects (i.e the parent account) will attempt to lock those records for processing. If another process is holding a lock on these records and the processing of the email has to wait for more than 10 seconds, a timeout will occur and you will see this error.

b. Apex Triggers/API

Assume there is an After Insert Apex Trigger on Tasks, and it runs for about 14 seconds while doing some processing. This trigger will run when a task is created. When tasks are created and are related to an Account, we place a lock on the parent Account while the task is being created. This means that the account cannot be updated while the task creation is in progress.
Reduce using Locking Statements.

Scenario:

1. User A imports a task via the data loader and assigns it to an existing account record. When the task is inserted, the apex trigger is fired.
2. Just 2 seconds after User A starts the insert via the data loader, User B is manually editing the same account record the task is related to.
3. When User B clicks Save, internally we attempt to place a lock on the account, but the account is already locked so we cannot place another lock. The account had already been locked by the creation of the Task.

The 2nd transaction then waits for the lock to be removed. Because the task takes about 14 seconds to be created, the lock is held for 14 seconds. The second transaction (from User B) times out, as it can only wait for a maximum of 10 seconds.

In this case, user B would see an error on the screen similar to the ones mentioned above. **Apex Tests can also incur locks if run against production data**.

c. Bulk API

Inserting or updating records through the Bulk API can cause multiple updates on the same parent record at once, because the batches are processed in parallel. For example, if two batches are being processed at the same time and the two of them contain records pointing to the same parent record, one of the batches will attempt to place a lock in the parent record, which can lead to the other batch throwing a "unable to lock row" error as the batch was not able to get a lock within 10 seconds.

To prevent this, you can do either of the following:

- Reduce the batch size
- Process the records in Serial mode instead of parallel, that way on batch is processed at a time.
- Sort main records based on their parent record, to avoid having different child records (with the same parent) in different batches when using parallel mode.

For these type of scenarios, it's also highly recommended that you become familiar with the guidelines found on the article

d. Master-detail Relationship

If a record on the master side of a master-detail relationship has too many child records (thousands) you are likely to encounter these errors as well, as every time you edit the detail record, the master record is locked. The more detail records you have, the more likely that these will be edited by users, causing the parent record to be locked.

To prevent this issue you can move some child records to another parent, as to reduce the amount of child records attached to a single parent record.
Record Level Locking is a common scenario which can be reduced.

Troubleshooting:

1. You can enable Debug logs for the user who is facing the error , to find the offending trigger/flow/ValidationRules/ causing the issue.
2. Check for any dependent background jobs that are running on the same object. If there is any, try to pause the jobs and then perform the actions to reduce row locks.

Locking Statements

In Apex, you can use `FOR UPDATE` to lock sObject records while they're being updated in order to prevent race conditions and other thread safety problems.

While an sObject record is locked, no other client or user is allowed to make updates either through code or the Salesforce user interface. The client locking the records can perform logic on the records and make updates with the guarantee that the locked records won't be changed by another client during the lock period. The lock gets released when the transaction completes.

To lock a set of sObject records in Apex, embed the keywords `FOR UPDATE` after any inline SOQL statement. For example, the following statement, in addition to querying for two accounts, also locks the accounts that are returned:

```
1 Account [] accts = [SELECT Id FROM Account LIMIT 2 FOR UPDATE];
```

To start debugging the issue, create debug logs and also check Apex Jobs at the same time. Apex job will have asynchronous calls in your flow and must have some DML which is updating same record on which you are getting UNABLE_TO_LOCK_ROW error. You need to check both the updates, the update on which you are facing UNABLE_TO_LOCK_ROW error and the asynchronous method or class which is updating the same record.