Salesforce.com APEX Sharing and Record Owners

Recently I ran into an error in a Salesforce.com trigger where I was creating APEX shares based on fields on the record in an organization with a private security model. In this particular case, it was a custom object that had a Lookup to an Account, and a second Lookup to a User. The goal was not only to ensure that the User referenced on the record had access to it, but also to the associated Account.

Normally it is easy enough to create APEX shares using the following bit of code.

```
AccountShare accountShare = new AccountShare();
accountShare.AccountId = customObj.Account__c;
accountShare.UserOrGroupId = customObj.User__c;
accountShare.AccountAccessLevel = 'Edit';
accountShare.OpportunityAccessLevel = 'Edit';
insert accountShare;
```

For the more curious, here is a wiki post that goes into much greater detail implementing this logic in a real environment.

However, I ran into an error recently months after this code had been running.

> *INSUFFICIENT_ACCESS_ON_CROSS_REFERENCE_ENTITY, insufficient access rights on cross-reference id: []*

After doing a bit of research, I happened across this blog post which explains quite a few different reasons why this error might pop up. In my case, the issue was that the User I was trying to share access to the Account was actually the owner of the Account. It appears that even though APEX shares between the same record and user tend to overwrite themselves if you keep reassigning them, the interpretation of owner access is a separate sharing reason that conflicts with an APEX share.

To rectify this, I did a query for all the Accounts related to the custom objects referenced in the trigger's batch prior to iterating over those records. I put all those Accounts into a map keyed by the Account's Id, and reference that to look up the Account Owner and compare it to the User Lookup field prior to creating a share.

```
Set<Id> accountIds = new Set<Id>();
for (Custom__c customObj : Trigger.new) {
   if (customObj.Account__c != null) {
      accountIds.add(customObj.Account__c);
   }
}

Map<Id, Account> accountMap = new Map<Id, Account>();
List<Account> accounts = [SELECT Id, OwnerId FROM Account WHERE Id =: accountIds];
for (Account acct : accounts) {
   accountMap.put(acct.Id, acct);
}

...

if (customObj.Account__c != null && accountMap.get(customObj.Account__c).OwnerId != customObj.User__c) {
   // Create APEX Share
}
```

Of course, this opens up its own can of worms if you later change the owner of the Account and deprive this User access, but that can be handled by an additional trigger on the Account or a nightly scheduled job to verify all the APEX sharing is up to date.