Locking Salesforce records for concurrent users to view or to perform any other actions

Use Case

In SFDC instance, if record(s) are getting created those are assigned to the queue. List of records are getting displayed in Visualforce page. Now users can update or remove those records accessing from a Visualforce page. Requirement is, if one user from the queue is working on the list of records, no other users can perform any DML options.

Solutions Approach 1 (Use of Approval Process)

1. Create an approval process on that object and when records are getting created it will get assigned to queue.
2. After querying the list of records, initiate approval process for those records.

```
Map<Id,Account> accountMap = new Map<Id,Account>([SELECT Id, Name,.....
FROM Account WHERE Name Like 'Acme%']);

for(Id accountObjId:accountMap.KeySet())
{
  // Create an approval request for the Account
  Approval.ProcessSubmitRequest req1 = new Approval.ProcessSubmitRequest();
  req1.setComments('Currently working on this Account');
  req1.setObjectId(accountObjId);

  // Submit on behalf of a specific submitter
  req1.setSubmitterId(UserInfo.getUserId());

  // Submit the record to specific process and skip the criteria evaluation
  req1.setProcessDefinitionNameOrId('Update_Account_Process');
  req1.setSkipEntryCriteria(true);

  // Submit the approval request for the account
  Approval.ProcessResult result = Approval.process(req1);

  // Verify the result
  System.debug(result.isSuccess());
}

```

3. Just during save approve those items as follows:

```
private List<Id> workItemlst = new List<Id>();
public void retrieveWorkItemId(Set targetObjectIds)
{
  for(ProcessInstanceWorkitem workItem :[Select p.Id FROM ProcessInstanceWorkitem p
  WHERE p.ProcessInstance.TargetObjectId IN:targetObjectIds]) {
  workItemlst.add(workItem.Id);
 }
}

public PageReference save()
{
  retrieveWorkItemId(accountMap.accountMap.KeySet());
  approveRecords();
  return (new ApexPages.StandardController (new Account(Id=strId))).view();
}

public void approveRecords()
{
  // Approve the submitted request
  for(Id objId:workItemlst)
  {
   // Instantiate the new ProcessWorkitemRequest object and populate it
   Approval.ProcessWorkitemRequest req2 =
   new Approval.ProcessWorkitemRequest();
   req2.setComments('Approving request.');
   req2.setAction('Approve');
   req2.setWorkitemId(objId);

   // Submit the request for approval
   Approval.ProcessResult result2 = Approval.process(req2);

   // Verify the results
   System.debug(result2.getInstanceStatus());
  }
}
```

4. To restrict concurrent user access use this logic to check if records are already submitted by some other user.

```
//first check if records have been initiated for approval
List<ProcessInstance> lstProcessInstance = [SELECT Id, SubmittedById
 FROM ProcessInstance
 WHERE TargetObjectId IN:accountMap.KeySet()
 AND Status = 'Pending'];

if(lstProcessInstance.size()>0){
  //if the logged on user previously submitted record for approval and which are still pending
  Id usrId = lstProcessInstance.get(0).SubmittedById;
 if(usrId == userid) {
  //retrieve ProcessInstanceWorkitem of the Account which have been already submitted.
  retrieveWorkItemId(accountMap.KeySet());
 return;
 }
 else {
  //show the error message that some other user currently working.
  isErrorFromApproval = true;
  List<User> lstUser = [SELECT FirstName, LastName FROM User WHERE Id =:usrId];
  String userName = lstUser.get(0).FirstName + ' ' + lstUser.get(0).LastName;
  throw new CustomException (userName + ' is currently working on Accounts');
 }
}
```

Advantages:

Above approach works well for locking and unlocking a record through approval process.

Disadvantages:

Salesforce will send approval request mail for each records which user is currently working.

Workaround to restricts emails can be done upon selecting 'Receive Approval Request Emails' attribute of User record to 'Never'. But this way user will not receive any emails from all the approval processes configured in the system. Moreover, to send email from approval processes, workflow email updates to be configured.

Solutions Approach 2 (Without using Approval Process)

Leveraging Set Approval Process Locks and Unlocks with Apex Code which is available from Winter'16 release.

1. After querying the records, lock those records using this method.

```
public void lockRecords()
{
 String errorString='';
 try{

  List<Account> lstRecord = accountMap.values();
  // Lock list of Accounts
  Approval.LockResult[] lrList = Approval.lock(lstRecord, false);

  // Iterate through each returned result
  for(Approval.LockResult lr : lrList) {
   if (lr.isSuccess()) {
   // Operation was successful, so get the ID of the record that was processed
   System.debug('Successfully locked Service Assets with ID: ' + lr.getId());
   }
   else {
   // Operation failed, so get all errors
    for(Database.Error err : lr.getErrors()) {
     System.debug('The following error has occurred.');
     System.debug(err.getStatusCode() + ': ' + err.getMessage());
     System.debug('Service Assets fields that affected this error: ' + err.getFields());
     errorString = 'Service Assets fields that affected this error: ' + err.getFields();
    }
   }
  }
  if(errorString.length()>0) throw new CustomException(errorString);
 }catch(Exception ex)
 {
  ApexPages.Message msg = new ApexPages.Message(ApexPages.Severity.Error, ex.getMessage());
  ApexPages.addMessage(msg);
 }
}
```

2. Before saving unlock the record as follows:

```apex
public void lockRecords()
{
 String errorString='';
 try{

  List<Account> lstRecord = accountMap.values();
  // unlock list of Accounts
  Approval.UnlockResult[] lrList = Approval.unlock(lstRecord);

  // Iterate through each returned result
  for(Approval.UnlockResult lr : lrList) {
   if (lr.isSuccess()) {
   // Operation was successful, so get the ID of the record that was processed
   System.debug('Successfully unlocked Service Assets with ID: ' + lr.getId());
   }
   else {
   // Operation failed, so get all errors
    for(Database.Error err : lr.getErrors()) {
     System.debug('The following error has occurred.');
     System.debug(err.getStatusCode() + ': ' + err.getMessage());
     System.debug('Service Assets fields that affected this error: ' + err.getFields());
     errorString = 'Service Assets fields that affected this error: ' + err.getFields();
    }
   }
  }
  if(errorString.length()>0) throw new CustomException(errorString);
 }catch(Exception ex)
 {
  ApexPages.Message msg = new ApexPages.Message(ApexPages.Severity.Error, ex.getMessage());
  ApexPages.addMessage(msg);
 }
}
```

3. Restricting other users accessing those locked records with the use of Approval.isLocked() method.

```apex
public boolean isRecordLocked(List<Id> accountIds)
{
 Map<Id,Boolean> mapLockedId = Approval.isLocked(accountIds);

```

```
  //here for example if any of the record is locked, it is returning true
  for(boolean bl:mapLockedId.values())
  {
   if(bl == true)
   {
   return true;
   }
  }
 return false;
}
```

Advantages:

1. Without using approval process, records can be locked or unlocked.