Inserting/updating parent and child records in a single transcation using external ID field

Naval Sharma November 20, 2017 2 Comments

If you are wondering, how to use external Id field to insert/update parent-child records without writing unnecessary lines of code then probably this post is worth reading. If you have been working on complex data integration then you might see this useful. When you have plenty of data which needs to be inserted/updated in the system, in such a case writing code for DML operations is very crucial as there may be chances to run into governor limits. If you have an external ID field available on the objects then careful use of DML statements (Inert, Update, Upsert etc) can save you from all the pain of dealing with the issues.

While working with external IDs it is always a best practice to make the field unique identifier. Now, I will explain the business logic so it will help in understanding the code snippet. Company xyz sells family insurance plans and it is looking for a way to sync its data from their external system (ERP) so they went with an integration.

Since it's an external system it has a column named as ID (unique identifier) in the table which we will map in Salesforce to the external ID field. So an Account is a person who is purchasing an insurance policy and members (contact records) are associated with the account record. We need to maintain the same relationship in Salesforce and where external Id field comes in the role. So, whenever sync job runs in the Salesforce it will update existing records or created new ones if they don't exist here.

```
public static void syncInsuranceBuyers( List<BuyerWrapper> lstBuyers ) {

    List<Account> lstBuyers = new List<Account>();
    List<Contact> lstMembers = new List<Contact>();

    // Iterate over all the buyers to create a list for upserting the records
    for( BuyerWrapper bw : lstBuyers ) {

        // Create a new instance of Account sObject and add it into the list
        lstBuyers.add(
            new Account(
                Name = bw.name,
                ERP_Id__c = bw.IdFromExternalSystem
                // You can add addional fields mapping here
            )
        );

        // Iterate over the related members
        for( MemberWrapper mw : bw.members ) {
            // Create a new instance of Contact sObject and add it into the list
            lstMembers.add(
                new Contact(
                    FirstName = mw.firstName,
                    LastName = mw.lastName,
                    ERP_Id__c = mw.IdFromExternalSystem,
                    Account = new Account( ERP_Id__c = bw.IdFromExternalSystem )
                    // You can add addional fields mapping here
                )
            );
        }
    }

    // Will have to make 2 different DML calls as we need to specify the ExternID field
    // which will be used to identify the existing records and will update them

    Database.UpsertResult[] resultsAcc = Database.upsert( lstBuyers, Account.Fields.ERP_Id__c );
    Database.UpsertResult[] resultsCon = Database.upsert( lstMembers, Contact.Fields.ERP_Id__c );

    /****
    * If it was just an insert opertaion then it could be achieved in a single DML
    *
        List<sObject> lstAllRecords = new List<sObject>();
        lstAllRecords.addAll( lstBuyers );
        lstAllRecords.addAll( lstMembers );
        Database.SaveResult[] results = Database.insert( lstAllRecords );

    * Since, upsert DML operation requires an external ID field defination as an extra parameter and we have 2 different fields so above can't be
    * achieved in a single DML call.
    ***/
}
```