To delete data that was created 30 days ago from multiple Salesforce objects and schedule this operation, we can use Apex Scheduler and Batch Apex. The idea is to query records that are older than 30 days from the objects you are interested in (like `Account`, `Contact`, `Opportunity`), and then delete them using a scheduled job.

**Steps:**

1. Create a Batch Apex Class to handle the deletion of records from multiple objects.

2. Create a Schedulable Apex Class to schedule the batch job.

3. Schedule the Job to run at a regular interval (daily, weekly, etc.).

---

**1. Batch Apex Class to Delete Data Older Than 30 Days**

Here’s how you can create the Batch Apex class that deletes records older than 30 days from multiple objects:

global class DeleteOldRecordsBatch implements Database.Batchable<SObject> {

global Database.QueryLocator start(Database.BatchableContext bc) {

// Get the date 30 days before today

Date thirtyDaysAgo = Date.today().addDays(-30);

// Query records older than 30 days from multiple objects

String accountQuery = 'SELECT Id FROM Account WHERE CreatedDate <= :thirtyDaysAgo';

String contactQuery = 'SELECT Id FROM Contact WHERE CreatedDate <= :thirtyDaysAgo';

String opportunityQuery = 'SELECT Id FROM Opportunity WHERE CreatedDate <= :thirtyDaysAgo';

// Combine the queries using UNION ALL (if applicable)

// You can combine the queries if you use a common field format or return a single SObject

return Database.getQueryLocator(accountQuery + ' UNION ALL ' + contactQuery + ' UNION ALL ' + opportunityQuery);

}

global void execute(Database.BatchableContext bc, List<SObject> scope) {

try {

// Delete records in the current batch

delete scope;

System.debug('Successfully deleted old records.');

} catch (DmlException e) {

// Handle any DML exceptions that occur during the delete operation

System.debug('Error occurred while deleting records: ' + e.getMessage());

}

}

global void finish(Database.BatchableContext bc) {

System.debug('Batch job finished.');

}

}

**Explanation:**

- `start` method: The query retrieves records from `Account`, `Contact`, and `Opportunity` objects where the `CreatedDate` is more than 30 days old (`<= :thirtyDaysAgo`). You can adjust the objects and fields as needed.

- `execute` method: This method deletes the records in bulk (per batch) using the `delete` DML statement.

- `finish` method: This runs when the batch job is finished. You can include additional logic here if necessary (like sending a notification).

---

**2. Schedulable Apex Class to Schedule the Job**

Next, we’ll create a Schedulable Apex class to schedule the batch job:

global class DeleteOldRecordsScheduler implements Schedulable {

global void execute(SchedulableContext sc) {

// Schedule the batch job to delete old records

Database.executeBatch(new DeleteOldRecordsBatch(), 200);

}

}

**Explanation:**

- The `execute` method schedules the `DeleteOldRecordsBatch` job with a batch size of 200 records (this can be adjusted based on your data volume).

---

**3. Scheduling the Job Using System.schedule()**

You can schedule the job to run at a regular interval, such as daily or weekly, using the `System.schedule` method. Here’s an example of scheduling the job to run daily at 2:00 AM:

// Schedule the job to run daily at 2:00 AM

String cronExpression = '0 0 2 \* \* ?'; // This is a CRON expression for 2:00 AM every day

DeleteOldRecordsScheduler scheduler = new DeleteOldRecordsScheduler();

System.schedule('DeleteOldRecordsJob', cronExpression, scheduler);

**Explanation of the Cron Expression:**

- `0 0 2 \* \* ?`: This is a CRON expression that schedules the job to run every day at 2:00 AM. You can adjust it based on your scheduling needs:

- `0`: seconds (start of the minute)

- `0`: minutes (start of the hour)

- `2`: hour (2 AM)

- `\*`: any day of the month

- `\*`: any month

- `?`: no specific day of the week

---

**Complete Code Implementation:**

**1. Batch Class (`DeleteOldRecordsBatch`)**

global class DeleteOldRecordsBatch implements Database.Batchable<SObject> {

global Database.QueryLocator start(Database.BatchableContext bc) {

Date thirtyDaysAgo = Date.today().addDays(-30);

// Query records older than 30 days

String accountQuery = 'SELECT Id FROM Account WHERE CreatedDate <= :thirtyDaysAgo';

String contactQuery = 'SELECT Id FROM Contact WHERE CreatedDate <= :thirtyDaysAgo';

String opportunityQuery = 'SELECT Id FROM Opportunity WHERE CreatedDate <= :thirtyDaysAgo';

return Database.getQueryLocator(accountQuery + ' UNION ALL ' + contactQuery + ' UNION ALL ' + opportunityQuery);

}

global void execute(Database.BatchableContext bc, List<SObject> scope) {

try {

delete scope;

System.debug('Successfully deleted old records.');

} catch (DmlException e) {

System.debug('Error occurred while deleting records: ' + e.getMessage());

}

}

global void finish(Database.BatchableContext bc) {

System.debug('Batch job finished.');

}

}

**2. Schedulable Class (`DeleteOldRecordsScheduler`)**

global class DeleteOldRecordsScheduler implements Schedulable {

global void execute(SchedulableContext sc) {

// Execute the batch job with a batch size of 200

Database.executeBatch(new DeleteOldRecordsBatch(), 200);

}

}

**3. Scheduling the Job**

// Schedule the batch job to run daily at 2:00 AM

String cronExpression = '0 0 2 \* \* ?'; // CRON expression for 2:00 AM daily

DeleteOldRecordsScheduler scheduler = new DeleteOldRecordsScheduler();

System.schedule('DeleteOldRecordsJob', cronExpression, scheduler);

---

**Notes**:

* **Governor Limits**: The use of Batch Apex helps handle large datasets by breaking them into manageable chunks (batches). Each chunk processes up to 200 records, keeping you within governor limits.
* **CRON Expressions**: You can adjust the CRON expression to run the job at different intervals, such as weekly or monthly.
* **Testing:** Before deploying to production, thoroughly test in a sandbox environment to ensure the job works as expected.

This approach efficiently deletes records older than 30 days from multiple Salesforce objects, ensuring regular clean-up through a scheduled batch process.