1 B

2 C

3 B

4 D

5 A

6 B

7 B

8 B

9 C

10 C

<https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_order_of_execution.htm>

**Best Practices for Designing Bulk Programs**

The following are the best practices for this design pattern:

* Minimize the number of data manipulation language (DML) operations by adding records to collections and performing DML operations against these collections.
* Minimize the number of SOQL statements by preprocessing records and generating sets, which can be placed in single SOQL statement used with the IN clause.

# Understanding Trigger.old and Trigger.new Salesforce Apex Triggers

Publish Date: Mar 21, 2025

Description

This article describes how Trigger.old vs. Trigger.new values are used before or after any changes are made to Salesforce records.  Trigger.old and Trigger.new are [**Context Variables**](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_context_variables.htm).

To access the records that caused the trigger to fire, we use context variables. For example, Trigger.new contains all the records that were inserted in insert or update triggers. Trigger.old provides the old version of sObjects before they were updated in update triggers, or a list of deleted sObjects in delete triggers.

Triggers can fire when one record is inserted, or when many records are inserted in bulk via the API or Apex. Therefore, context variables, such as Trigger.new, can contain only one record or multiple records. You can iterate over Trigger.new to get each individual sObject.

Resolution

## Use Case

* When a field value is updated to a certain value, we use trigger.old and trigger.new to compare the older and new version values of the field values on a record and perform the required business logic accordingly.
* trigger.old is available only on the update and delete events and represents the old version of the records before the trigger event.
* trigger.new is available only on insert, update, and undelete triggers and represents the new version of the records after the trigger event.

## ****Example Scenario****

You have a workflow on an Account object creation that updates a field such as the "Description" field.

You then have a trigger that fires after the workflow update and iterates over trigger.old, expecting to get the "Description" field that was modified by the workflow.

In this case, you will get a nullpointerException.

## ****Rationale****

The reason comes down to understanding the values held by these 2 data structures.

Trigger.old values will be the same before and after the CURRENT transaction (user action). The same applies to Trigger.new.

In other words - Trigger.old won't hold the newly updated field in the workflow because the example is an insert scenario.  However, if we proceed to manually edit the record, the trigger will fire again (and this is viewed as a new “update transaction”). Trigger.old will hold the field that was updated on the previous transaction by the workflow rule.

Going back to the example above:

The values in Trigger.old after the workflow update will NOT contain the “Description” field that was updated in the workflow.

The values in Trigger.new after the workflow update will contain any existing fields that were populated upon the object’s creation AND the “Description” workflow updated field.

You'll have to query the database after the workflow field update fires In order to obtain that same field.

For example:

List<Account> accts = [Select Id, FieldUpdatedByWorkflow\_\_c from Account where Id in : Trigger.old];

See the sample trigger below.

Note: Create a workflow field update on Account objects creation to see the results in the debug log.

trigger testTrigger on Account (before update, after update, before insert, after insert)

{

if (Trigger.isBefore) {

System.debug('\*\*\*\*\*\*\*\*Trigger values\*\*\*\*\*\*\*\*\*\*\*');

System.debug('\*\*\*SFDC: Trigger.old is: ' + Trigger.old);

System.debug('\*\*\*SFDC: Trigger.new is: ' + Trigger.new);

}

if (Trigger.isAfter) {

System.debug('\*\*\*\*\*\*\*\*Trigger values\*\*\*\*\*\*\*\*\*\*\*');

System.debug('\*\*\*SFDC: Trigger.old is: ' + Trigger.old);

System.debug('\*\*\*SFDC: Trigger.new is: ' + Trigger.new);

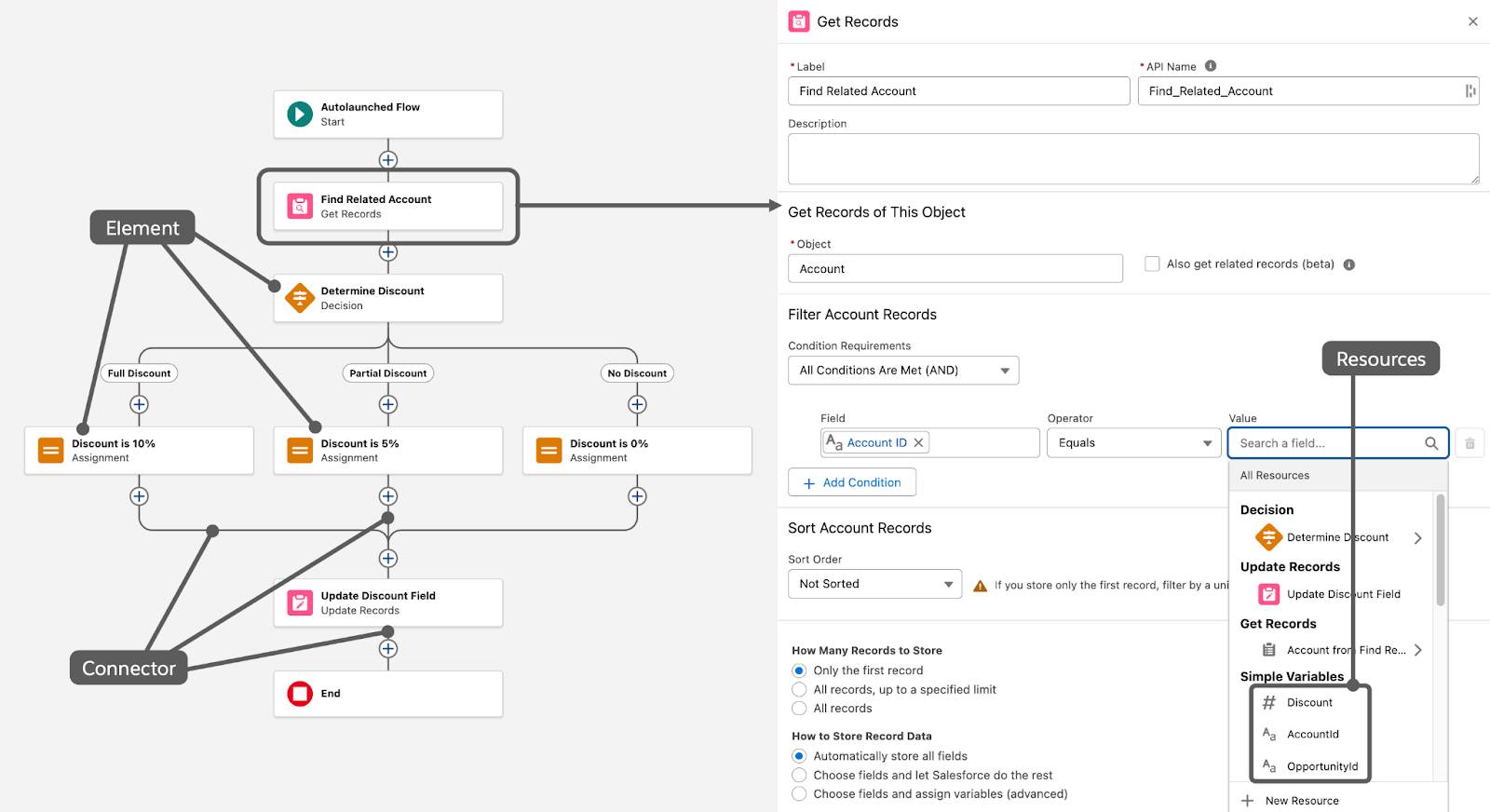
}

}

<https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_context_variables.htm>

<https://architect.salesforce.com/decision-guides/trigger-automation>

<https://trailhead.salesforce.com/content/learn/modules/search_solution_basics/search_solution_basics_choosing?trailmix_creator_id=strailhead&trailmix_slug=prepare-for-your-salesforce-platform-developer-i-credential>



Flow builders

**✅ Final Answer Key:**

1. B
2. C
3. C
4. B
5. A
6. B
7. B
8. D
9. A
10. B

* **Record-Triggered Flows:** Use Flow Builder to create automation that runs when a record is created, edited, or deleted.
* **Schedule-Triggered Flows:** Use Flow Builder to create automation that runs at a time and frequency you specify.
* **Platform Event-Triggered Flows:** Use Flow Builder to create automation that runs when a platform event message is received.
* **Data Cloud-Triggered Flow:** Use Flow Builder to create automation that runs when a change is made to data in Data Cloud.
* **Apex:** Use Apex code to write reusable blocks of automation. You can trigger this code in a variety of ways.

### **Scenario 1: Auto-Close Case with No Response**

If a customer does not respond within **7 days** after a support agent replies, the Case should automatically be closed.

Which automation should be used?

**Answer:** **Scheduled-Triggered Flow**

* Runs daily, checks for Cases with “Waiting for Customer” status older than 7 days, and closes them.

### **Scenario 2: Create Default Child Records**

Whenever a new **Account** is created, automatically create:

* A “Primary Contact” record
* A “Billing Contact” record

Which Flow solution should be used?

**Answer:** **Record-Triggered Flow (After Insert)**

* Flow triggers after Account is created.
* Uses **Create Records** element to add Contacts.

### **Scenario 3: Guided Case Intake**

Support agents need a **wizard** to capture customer issue details:

1. Capture customer information.
2. Ask issue-related questions.
3. Display a summary before creating the Case.

Which Flow should be used?

**Answer:** **Screen Flow**

* Multi-screen guidance is only possible with Screen Flows.

### **Scenario 4: Escalation to Manager**

If an **Opportunity Stage = Negotiation** and the **Discount > 30%**, then automatically create a Task for the Sales Manager to review.

Which Flow should be used?

**Answer:** **Record-Triggered Flow**

* Runs when Opportunity is updated.
* Decision element checks Discount %.
* Creates Task for Sales Manager.

### **Scenario 5: Auto-Update Parent Record**

When a **Contact** is updated, the related **Account’s Last Contacted Date** field should also update.

Which Flow solution should be used?

**Answer:** **Record-Triggered Flow (After Update on Contact)**

* Flow updates the parent Account record.

### **Scenario 6: Mass Update on Schedule**

Every Monday morning, update the **“Open Cases Count”** field on Account with the number of related Cases in Open status.

Which Flow should be used?

**Answer:** **Scheduled-Triggered Flow**

* Runs weekly.
* Uses **Get Records + Loop** to count Cases.
* Updates Account field.

### **Scenario 7: Error Handling in Flow**

A Flow creates an Opportunity for every new Lead converted. If the Opportunity creation fails (e.g., missing required field), it should log an error in a custom object called “Flow Error Logs.”

Which Flow design handles this?

**Answer:** **Fault Path in Flow**

* Fault paths catch the error and insert a record in the log object.

### **Scenario 8: Dynamic Assignment of Cases**

When a Case is created:

* If Type = “Technical,” assign to the Tech Support Queue.
* If Type = “Billing,” assign to Billing Queue.
* Else, assign to General Support Queue.

Which Flow feature is used?

**Answer:** **Decision Element + Update Records** in a **Record-Triggered Flow**.

### **Scenario 9: Prevent Invalid Input**

During a Screen Flow, users must enter a **Phone Number** in a specific format (XXX-XXX-XXXX).

Which Flow feature should be used?

**Answer:** **Screen Component with Validation Rule (Regex)**

* Ensures proper formatting before record creation.

### **Scenario 10: Reuse Flow Logic**

Multiple Flows need to call the same logic that calculates a Customer Loyalty Score. Instead of duplicating, how should it be designed?

Which Flow feature is best?

**Answer:** **Subflow**

* Reusable logic packaged in one flow, invoked by others.