Trigger Interview Questions

Triggers best practices :

1) One Trigger Per Object
A single Apex Trigger is all you need for one particular object. If you develop multiple Triggers for a single object, you have no way of controlling the order of execution if those Triggers can run in the same contexts.

2) Logic-less Triggers
If you write methods in your Triggers, those can't be exposed for test purposes. You also can't expose logic to be re-used anywhere else in your org.

3) Context-Specific Handler Methods
Create context-specific handler methods in Trigger handlers.

4) Bulkify your Code
Bulkifying Apex code refers to the concept of making sure the code properly handles more than one record at a time.

5) Avoid SOQL Queries or DML statements inside FOR Loops
An individual Apex request gets a maximum of 100 SOQL queries before exceeding that governor limit. So if this trigger is invoked by a batch of more than 100 Account records, the governor limit will throw a runtime exception.

6) Using Collections, Streamlining Queries, and Efficient For Loops
It is important to use Apex Collections to efficiently query data and store the data in memory. A combination of using collections and streamlining SOQL queries can substantially help writing efficient Apex code and avoid governor limits.

7) Querying Large Data Sets

The total number of records that can be returned by SOQL queries in a request is 50,000. If returning a large set of queries causes you to exceed your heap limit, then a SOQL query for loop must be used instead. It can process multiple batches of records through the use of internal calls to query and query more.

8) Use @future Appropriately
It is critical to write your Apex code to efficiently handle bulk or many records at a time. This is also true for asynchronous Apex methods (those annotated with the @future keyword). The differences between synchronous and asynchronous Apex can be found.

9) Avoid Hardcoding IDs
When deploying Apex code between sandbox and production environments, or installing Force.com AppExchange packages, it is essential to avoid hardcoding IDs in the Apex code. By doing so, if the record IDs change between environments, the logic can dynamically identify the proper data to operate against and not fail.

Few more Best Practices for Triggers

1. There should only be one trigger for each object.
2. Avoid complex logic in triggers. To simplify testing reuses, triggers should delegate to Apex classes that contain the actual execution logic. See Mike Leach's excellent trigger template for more infoBulkyfy any "helper" classes and/or methodsTriggersrs should be "bulky" and be able to process up to 200 records for each call.
3. Execute DML statements using collections instead of individual records per DML statement.
4. Use Collections in SOQL "WHERE" clauses to retrieve all records back in a single query
5. Use a consistent naming convention including the object name (e.g. AccountTrigger)

==========

Trigger Link : Important

https://developer.salesforce.com/forums/?id=906F000000093cKIAQ

Link : Trigger best Practices

https://www.simplysfdc.com/2016/12/salesforce-trigger-best-practices.html

https://niksdeveloper.com/salesforce/apex-trigger-best-practices-all-in-one/

https://developer.salesforce.com/forums/?id=906F0000000DBI8IAG

https://www.sfdcpoint.com/salesforce/apex-trigger-in-salesforce/

Trigger.old ==> update and delete trigger

Trigger.oldMap ==> update and delete trigger

Trigger.old - It returns a list of sobject records.

Trigger.oldMap - It returns a map of Id vs Sobject record.

---

trigger.new ==> insert and update triggers --> before triggers.
trigger.NewMap ==> before update, after insert, and after update triggers.

==========

1. Write An trigger : accounts have multiple opportunities. if we update the amount field in the opportunity then we need to update total_ammount field on the account object.
write an apex trigger ? for updating account child contact when update the account
2. Can we call future from trigger
3. We have a trigger and in the trigger we update some fields using the future method. It's throwing an error that we cannot call future from future. why? and how do we fix this issue?

==========

What is Trigger Syntax?

```
trigger TriggerName on ObjectName (trigger_events) {
   code_block
}
```

Trigger events in salesforce?

before insert
before update
before delete
after insert
after update
after delete
after undelete

Define Recursive Trigger and how to avoid it?

There is a possibility that the result of the trigger can end up calling the same trigger again and can run
in a loop, this is known as a recursive trigger. To avoid this scenario we should create a static variable
and check the value of this variable before we execute anything in the trigger

Types of Apex triggers

Before triggers are used to update or validate values of a record before they are saved to the database.

After triggers are used to access field values of the records that are stored in the database and use these values to make changes in other records.

What is a Recursive Trigger?

A recursive trigger is one that performs an action, such as an update or insert, which invokes itself owing to, say something like an update it performs.

What are context variables in triggers?

Variable	Usages
isExecuting	Returns true if the current context for the Apex code is a trigger, not a Visualforce page, a Web service, or an executeanonymous() API call.
isInsert	Returns true if this trigger was fired due to an insert operation.
isUpdate	Returns true if this trigger was fired due to an update operation.
isDelete	Returns true if this trigger was fired due to a delete operation.
isBefore	Returns true if this trigger was fired before any record was saved.
isAfter	Returns true if this trigger was fired after all records were saved.
isUndelete	If a record is recovered from the recycle bin it returns trigger true.
new	Returns a list of the new versions of the sObject records. This sObject list is only available in insert, update, and undelete triggers, and the records can only be modified before triggers.
newMap	A map of IDs to the new versions of the sObject records. This map is only available in before update, after insert, after update, and after undelete triggers.
old	Returns a list of the old versions of the sObject records. This sObject list is only available in update and deletes triggers.
oldMap	A map of IDs to the old versions of the sObject records. This map is only available in update and deletes triggers.
size	The total number of records in a trigger invocation.

Triggers	Workflow
• Trigger is a piece of code that executes before or after a record is inserted or updated.	• Workflow is automated process that fired an action based on Evaluation criteria and rule criteria.
• Coding is required.	• Coding is not required.
• We can access the trigger across the object and related to that objects.	• We can access a workflow across the object.
• More than 15 DML operations can be used in a single trigger.	• We cannot perform DML operation on workflow. Only insert and update.
• We can use 20 SOQL's from data base in one trigger.	• We cannot query from database.

Difference between Validation rules and Triggers?

Validation Rules	Triggers
Validation rules are used to confirm that the data entered into a record meet various data quality / business rules before letting the user save it.	Triggers can be used for various different things and can be executed at different times - e.g. when initiating a new record, before saving it, when a record is deleted.

Validation rules are very easy to create and virtually anyone can learn how to create these.	Triggers are more complex and generally require some development skills.

Important Difference:

Inser

• Partial insert is not supported.

• Roll back is not supported.

• If we use the DML statement (insert), then in bulk operation if error occurs, the execution will stop and Apex code throws an error which can be handled in try catch block.

Database.insert

• Database methods are static methods available in Database class.

• Partial insert is supported.

• Roll back is supported.

• If DML database methods (Database.insert) used, then if error occurs the remaining records will be inserted / updated means partial DML operation will be done.

Example: If you are inserting 10 records.

5 having all the required fields value and remaining 5 records missing required fields value.

In DML statement (insert) all the 10 records will be failed, because if one record is incorrect or error means all other remaining records will not be inserted. It will throw error.

In Database.insert 5 records will be inserted, remaining 5 records will be failed.(i.e. Partial DML Operation).

i) **What are the the context variable available with before insert event?**

Only Trigger.new is available.

ii) **What are the the context variable available with after insert event?**

Trigger.new and Trigger.newMap.

iii) **What are the the context variable available with before update event?**

Trigger.new, Trigger.old, Trigger.newmap and Trigger.oldmap

iv) **What are the the context variable available with after update event?**

Trigger.new, Trigger.old, Trigger.newmap and Trigger.oldmap

v) **What are the the context variable available with before delete event?**

Trigger.old and Trigger.oldMap.

vi) **What are the the context variable available with after delete event?**

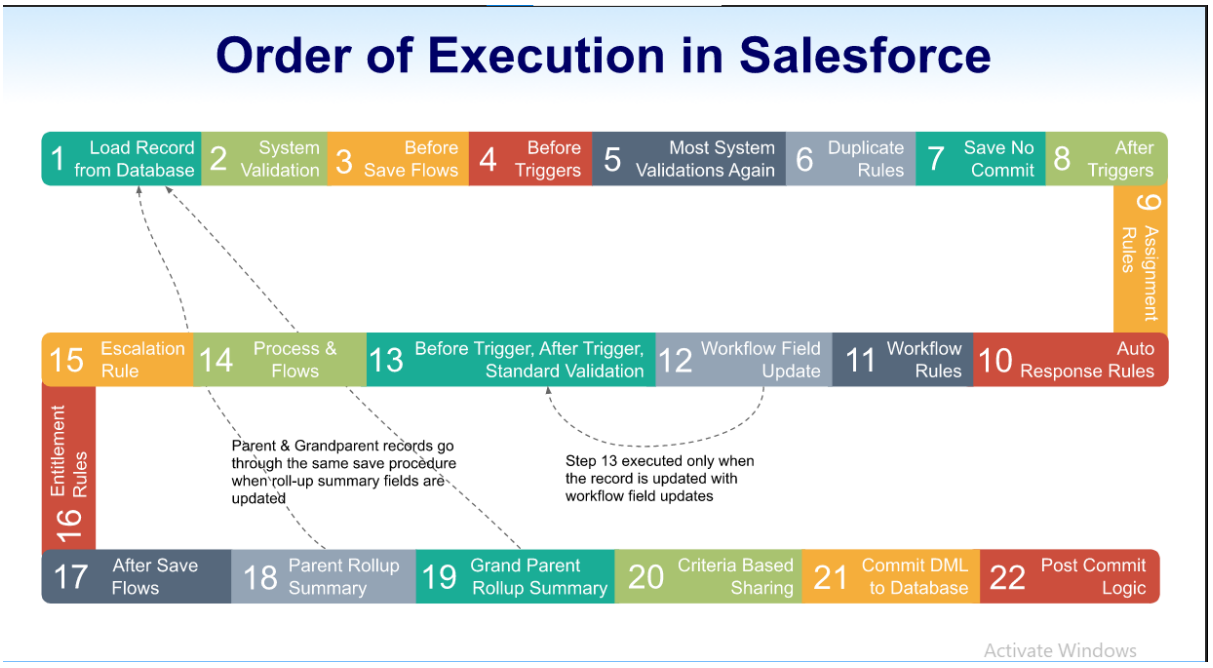
Trigger.old and Trigger.oldMap.

vii) **What are the the context variable available with after undelete event?**

Trigger.new and Trigger.newMap.

```
trigger objectTrigger on ObjectName (after delete, after insert, after undelete, after update,
before delete, before insert, before update) {

    objectTriggerHandler handler = new objectHandler();

    /* Before Insert */
    if(Trigger.isInsert && Trigger.isBefore){
        handler.OnBeforeInsert(Trigger.new);
    }
    /* After Insert */
    else if(Trigger.isInsert && Trigger.isAfter){
        handler.OnAfterInsert(Trigger.new);
    }
    /* Before Update */
    else if(Trigger.isUpdate && Trigger.isBefore){
        handler.OnBeforeUpdate(Trigger.old, Trigger.new, Trigger.newMap);
    }
    /* After Update */
```

```
    else if(Trigger.isUpdate && Trigger.isAfter){
        handler.OnAfterUpdate(Trigger.old, Trigger.new, Trigger.newMap);
    }
    /* Before Delete */
    else if(Trigger.isDelete && Trigger.isBefore){
        handler.OnBeforeDelete(Trigger.old, Trigger.oldMap);
    }
    /* After Delete */
    else if(Trigger.isDelete && Trigger.isAfter){
        handler.OnAfterDelete(Trigger.old, Trigger.oldMap);
    }

    /* After Undelete */
    else if(Trigger.isUnDelete){
        handler.OnUndelete(Trigger.new);
    }

}
```



1. System Validation Rules
2. Apex Before Triggers
3. Custom Validation Rules
4. Duplicate Rules
5. Apex After Triggers
6. Assignment Rules

7. Auto-Response Rules
8. Workflow Rules
9. Processes
10. Escalation Rules
11. Roll-Up Summary Fields

View State Error

View State: As the name suggests, it is an error of View State. This is important in the light that salesforce provides only a standard size of 135kb for any individual page. ... **If the size of a particular page exceeds 135kb**, the page will throw a view state error