Salesforce Apex DML Statements – Data Manipulation Language

- Different Salesforce Apex DML Statements
  - Insert DML Statement.
  - Update DML Statement
  - Upsert DML Statement
  - Delete DML Statement
  - Undelete DML Statement
  - Merge DML Statement
- Different ways to perform DML statements

Salesforce Apex DML Statements

Salesforce Apex DML (Data Manipulation Language) statements are used to Insert, Update, Merge, Delete and restore data in Salesforce. We can perform DML operations using the Apex DML statements or the methods of the **Database** class. We can perform 6 DML operations they are

1. Insert.
2. Update.
3. Delete.
4. UpSert.
5. Merge and
6. Restore.

Insert DML Statement

The insert DML operation adds one or more sObjects, such as individual accounts or contacts, to your organization's data. insert is analogous to the INSERT statement in SQL.

Syntax

```
insert sObject
```

```
insert sObject

insert sObject[]
```

Example

```
Account newAcct =
new Account(name =
```

```
Account newAcct = new Account(name = 'Acme');
try {
   insert newAcct;
} catch (DmlException e) {
// Process exception here
}
```

Update DML Statement

The update DML operation modifies one or more existing sObject records, such as individual accounts or contacts, invoice statements, in your organization's data. Update is analogous to the UPDATE statement in SQL.

Syntax

```
update sObject
```

```
update sObject

update sObject[]
```

Example

```
Account a = new
Account(Name='Acme2'
```

```
Account a = new Account(Name='Acme2');
insert(a);

Account myAcct = [SELECT Id, Name, BillingCity FROM Account WHERE Id = :a.Id];
myAcct.BillingCity = 'San Francisco';

try {
   update myAcct;
} catch (DmlException e) {
   // Process exception here
}
```

Upsert DML Statement

The upsert DML operation creates new records and updates sObject records within a single statement, using a specified field to determine the presence of existing objects, or the ID field if no field is specified.

Syntax

```
upsert sObject??
[opt_field]
```

```
upsert sObject?? [opt_field]
upsert sObject[]?? [opt_field]
```

Example

```
List<Account> acctList
= new List<Account>
```

```
List<Account> acctList = new List<Account>();
// Fill the accounts list with some accounts

try {
    upsert acctList;
} catch (DmlException e) {

}
```

Delete DML Statement

The delete DML operation deletes one or more existing sObject records, such as individual accounts or contacts, from your organization's data. Delete is analogous to the delete() statement in the SOAP API.

Syntax

```
delete sObject
delete sObject[]
```

```
delete sObject
delete sObject[]
```

Example

```
Account[]
deomedAccts =
```

```
Account[] doomedAccts = [SELECT Id, Name FROM Account
                         WHERE Name = 'DotCom'];
try {
    delete doomedAccts;
} catch (DmlException e) {
    // Process exception here
}
```

Undelete DML Statement

The undelete DML operation restores one or more existing sObject records, such as individual accounts or contacts, from your organization's Recycle Bin. undelete is analogous to the UNDELETE statement in SQL.

Syntax

```
undelete sObject | ID
undelete sObject[] | ID[]
```

Example

```
Account[] savedAccts = [SELECT Id, Name FROM Account WHERE Name = 'Universal Containers' ALL ROWS];
try {
    undelete savedAccts;
} catch (DmlException e) {
    // Process exception here
}
```

Merge DML Statement

The merge statement merges up to three records of the same sObject type into one of the records, deleting the others, and re-parenting any related records.This DML operation does not have a matching Database system method.

Syntax

```
merge sObject sObject
```

```
merge sObject sObject

merge sObject sObject[]

merge sObject ID

merge sObject ID[]
```

Example

```
List<Account> ls =
```

```
List<Account> ls = new List<Account>{new Account(name='Acme Inc.'),new Account(name='Acme')};
insert ls;
Account masterAcct = [SELECT Id, Name FROM Account WHERE Name = 'Acme Inc.' LIMIT 1];
Account mergeAcct = [SELECT Id, Name FROM Account WHERE Name = 'Acme' LIMIT 1];
try {
    merge masterAcct mergeAcct;
} catch (DmlException e) {
    // Process exception here
}
```

Different ways to perform DML operations.

1. By using DML Statements
2. By using Database Class.

By using DML Statements.

```
List <Account> accList
```

```
List <Account> accList = new List<Account>();
accList.add (new Account(Name='Account1'));
accList.add(new Account(Name='Account2'));
insert accList;
```

By using Database Class.

```
List <Account> accList
```

```
List <Account> accList = new List<Account>();
accList.add (new Account(Name='Account1'));
accList.add(new Account(Name='Account2'));
Database.SaveResult[] Sr = Database.insert(acctList,false);
```

As shown above, there is one difference between the two operations. In the database class method, we can specify whether to allow partial processing of the records if any errors are encountered by paring the boolean values as a parameters to Database.insert.

- If we give the parameter as 'True'. if any error occurs it doesn't allow the operation to continue.
- If we give the parameter as 'false', the remaining DML operations can still succeed. Where as insert in DML if any one of the record fails the total operation is discarded.

Salesforce Apex

- What is Salesforce Apex?

Apex Basics

- How to Enable Developing Mode in Salesforce?
- How to use Salesforce developer Console.
- Apex Data types.
- Apex - Variables
- Apex - Class
- Apex - Methods
- Apex - Constructors
- Apex - Scheduler
- What is Salesfore Batch Apex
- Batch Apex - Governor Limits
- Triggers in Salesforce
- Email Messages - Inbound, Outbound
- Apex - Interface
- Apex - DML statements