Big Object - Bring Big Data In Salesforce

by : arpit vijayvergiya 41

Hi guys today I am going to discuss about **Big Object** in salesforce. Big Object allows you to store tons of data into salesforce platform. It provide consistent performance whether there are 1 million records or 1 Billion records . But we can store data in custom object also then **why we use Big Object** .

Big object is worth to use if you want to store historical data.

Eg. All the Case record which are closed and were created years ago.
Transaction history of a user.

There are 2 type of Big Object In salesforce.

1. Standard Big Object : These Object are defined by Salesforce itself.
2. Custom Big Object :These Object are defined by user in its Salesforce org.

Custom Object VS Big Object :

Sr no	Custom Object	Big Object
1.	Suffix is __c	Suffix is __b
2.	Can be created from UI.	Can't be created from UI. We must need to use metadata api to create Big Object and add field to Custom Big Object
3.	Support standard ui element such as list view , details page.	Do not support standard ui element such as list view , page layout.
4.	Standard Reports and Dashboard are available.	Standard Report and Dashboard are not available.
5.	Limit for maximum no of Custom Object can be created in org. Depend on the license type. In developer Org it is 400.	Limit for maximum no of Big Object can be created in org. Depend on the license type. In developer Org it is 100.
6.	Triggers, flow and process builder are available.	To support the scale of data in Big Object Triggers, flow and process builder are not available.
7.	Support lot of data types.	Support DateTime, Lookup, Number, Text, and LongText Area only
8.	Custom fields can be deleted.	Custom fields can not be deleted once created.
9.	Records can be updated and deleted.	Records Can't be deleted once inserted.

Define a Big Object : Define a custom big object through the Metadata API by creating XML files that contain its definition, fields, and index . Files we need are specified below.

1. **Object File** :- Create a .object file. The name of file should be similar to API of object. Eg. myobject__b.object

2. **permissionSet/Profile file** :- This file is not required. We create permissionSet or profile file to give access permission for each field.

3. **package.xml** :- Create a file for the metadata package to specify the contents.

Let's see an example. We are creating a Big Object Employee (You can also download source file.)
So 1st we will make an Employee__b.object file . It will look like this.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>

    <fields>
        <fullName>First_Name__c</fullName>
        <label>First Name </label>
       <length>50</length>
       <type>Text</type>
       <unique>false</unique>
       <required>true</required>
    </fields>
```

```
    <fields>
        <fullName>Salary__c</fullName>
        <label>Salary</label>
        <type>Number</type>
        <scale>2</scale>
        <precision>16</precision>
        <required>false</required>
        <unique>false</unique>
    </fields>

    <fields>
        <fullName>Dob__c</fullName>
        <label>Date Of Birth</label>
        <type>DateTime</type>
        <required>false</required>
        <unique>false</unique>
    </fields>

    <fields>
        <fullName>Contact__c</fullName>
        <label>Contact</label>
        <referenceTo>Contact</referenceTo>
        <relationshipName>Employee_Con</relationshipName>
        <type>Lookup</type>
    </fields>
    <indexes>
        <fullName>EmployeeIndex</fullName>
        <label> Employee index</label>
        <fields>
            <name>First_Name__c</name>
            <sortDirection>DESC</sortDirection>
        </fields>
    </indexes>

    <label>Employee</label>
    <pluralLabel>Employee</pluralLabel>
</CustomObject>
```

Now we will make perssionSet File. It's extension will be .permissionSet

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionSet xmlns="http://soap.sforce.com/2006/04/metadata">

    <fieldPermissions>
        <editable>true</editable>
        <field>Employee__b.firstName__c</field>
        <readable>true</readable>
    </fieldPermissions>

    <fieldPermissions>
        <editable>true</editable>
        <field>Employee__b.email__c</field>
        <readable>true</readable>
    </fieldPermissions>

    <fieldPermissions>
        <editable>true</editable>
        <field>Employee__b.salary__c</field>
        <readable>true</readable>
    </fieldPermissions>

    <fieldPermissions>
        <editable>true</editable>
        <field>Employee__b.dob__c</field>
        <readable>true</readable>
    </fieldPermissions>

</PermissionSet>
```

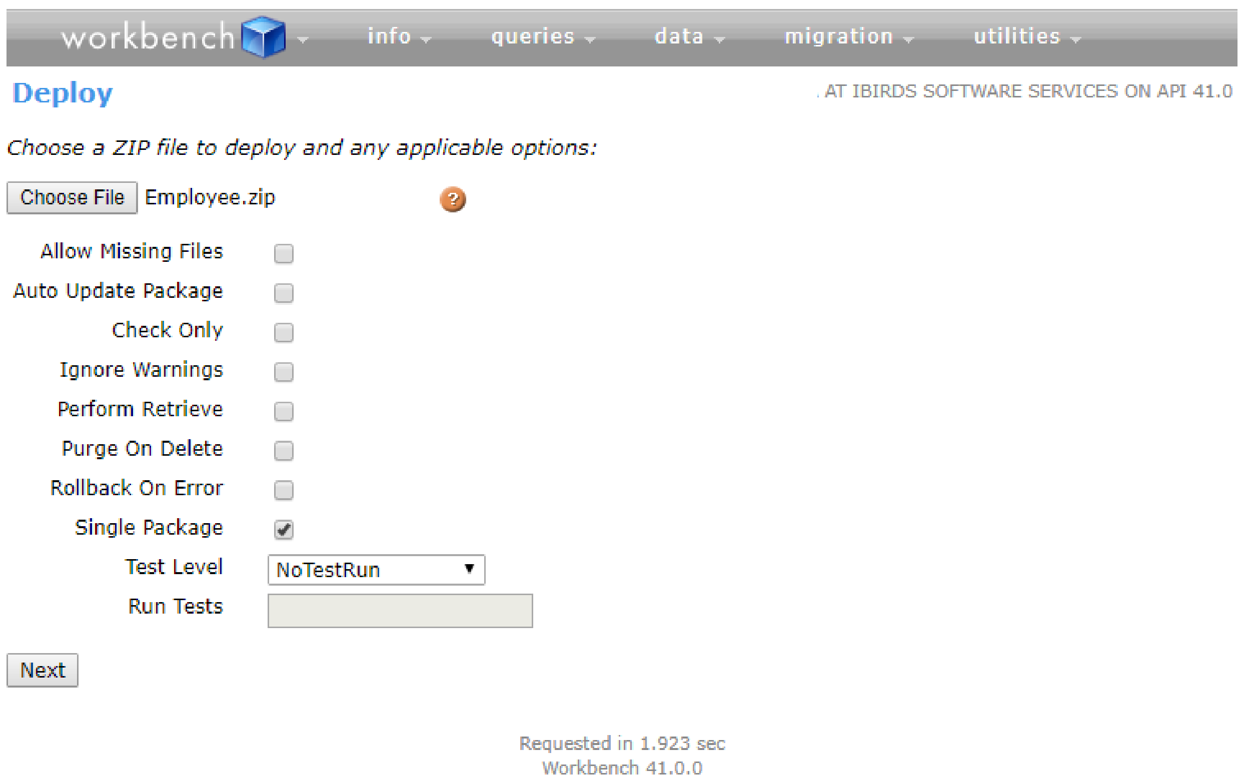
In last we need package file. (package.xml)

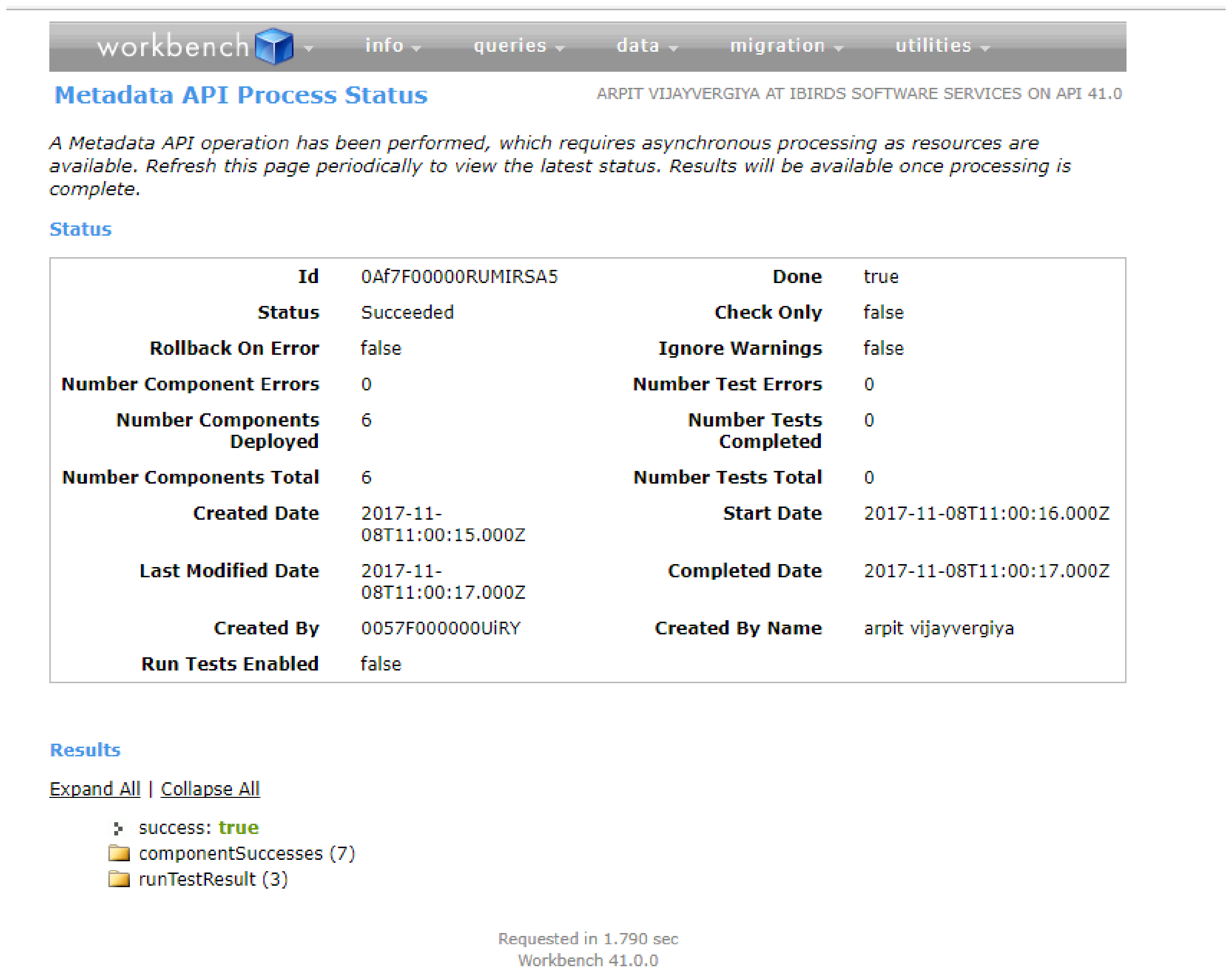
```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>

    <types>
```

```
        <members>*</members>
        <name>PermissionSet</name>
    </types>
    <version>41.0</version>
</Package>
```

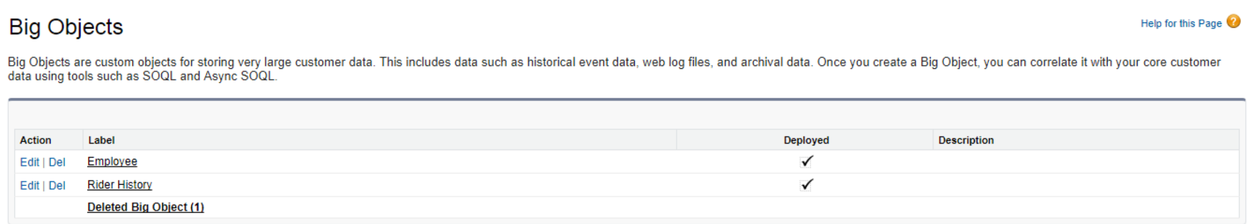
Deploy Custom Big Object : We are using workbench to deploy. To create zip (package) file for deploying custom Big Object make sure that the object file should be in objects folder, permissionSet file should be in permissionSets folder and the package.xml file should be in root directory (must not inside any folder).

See the image

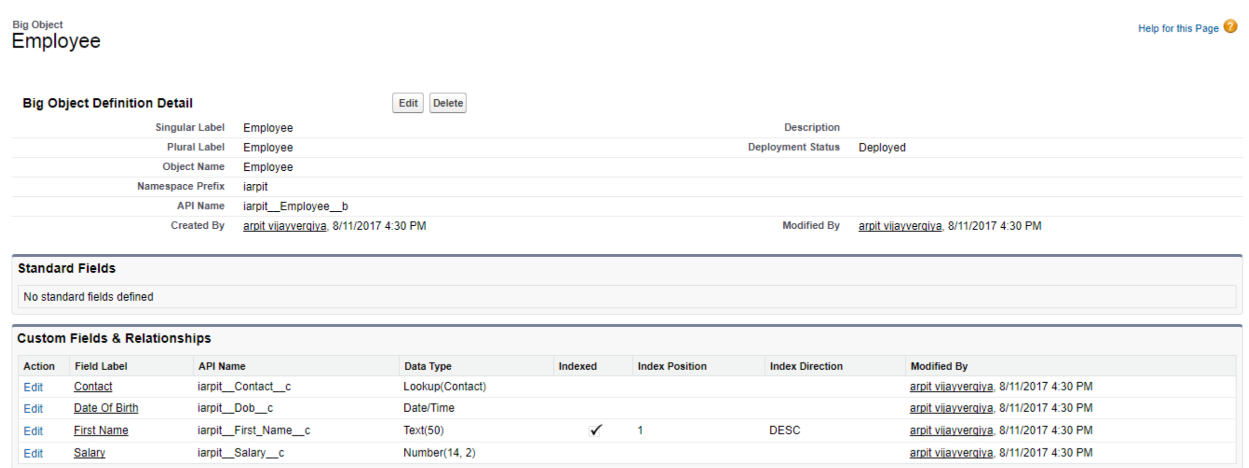




Once you have successfully deployed Big object you can view them under Big Object in setup ui .



When we see the details of Employee Big Object we see that there is no button is available for delete and create new custom field . What if we want to add some new fields and want to delete any of existing.
Let me clear this that **you can't delete a custom field created on Big Object.** To add new custom field we will go through metadata deployment.



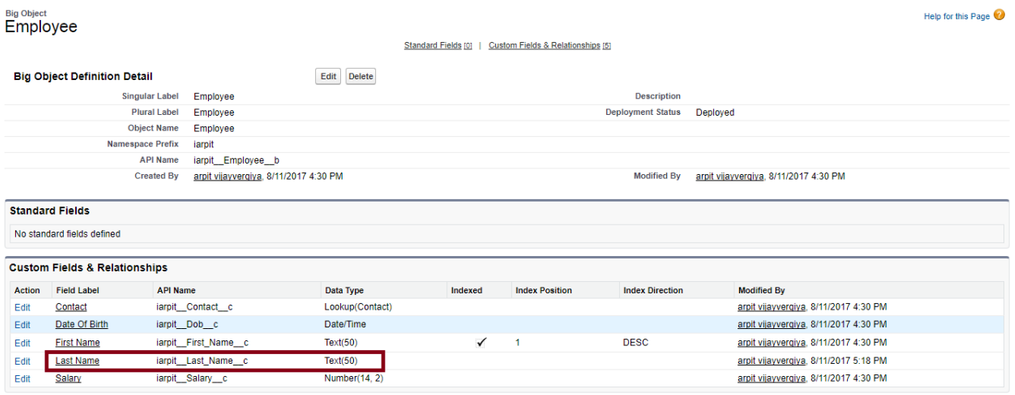
Add new Custom Field to Custom Big Object :- *Earlier we created a custom Big Object . Now we want to add a new custom field in it.*

We need to create an .object file with the same name of previous one (Employee__b.object). It will contain the following code

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>

    <fields>
```

```
    <fullName>Last_Name__c</fullName>
    <label>Last Name </label>
    <length>50</length>
    <type>Text</type>
    <unique>false</unique>
  </fields>

  <label>Employee</label>
  <pluralLabel>Employee</pluralLabel>
</CustomObject>
```

And now we will deploy it using workbench. For deploying we will follow the previous example structure . This time we are not using permissionSet file . package.xml will be same as previous. After deployment we can see there is new Custom field Last Name on Employee__b Big Object.



Insert Data into Big Object :
Using CSV :- To insert data into Big Data we can upload csv file using data loader.

Using Apex Code :- Using **Database.insertImmediate()** we can insert data into Big Object.
Here is the snippet using it you can insert data into our Employee Object.

```
List<Employee__b> listOfEmp  = new List<Employee__b>();
Employee__b objEmp = new Employee__b();
objEmp.first_Name__c = 'Naveen';
objEmp.Salary__c = 5000;
objEmp.Last_Name__c = 'Soni';
listOfEmp.add(objEmp);
Database.insertImmediate(listOfEmp);
```

Note : **Big objects don't support transactions including both big objects, standard object and custom objects**
Then How can we Insert records in Big Object. Solution is "Async SOQL", we will go through it in next steps.
If you are not able to insert record in then check for field permission for your profile. Even if you are a system administrator .

Update Data into Big Object :

Code snippet :

```
List<iarpit__Employee__b> listOfEmp = new List<iarpit__Employee__b>();
for(iarpit__Employee__b objEmp : [SELECT  iarpit__First_Name__c, iarpit__Last_Name__c FROM iarp
    iarpit__Employee__b objEmp1 = new iarpit__Employee__b();
    objEmp1.iarpit__last_name__c = 'rohit';
    objEmp1.iarpit__first_name__c = objEmp.iarpit__first_name__c;
    listOfEmp.add(objEmp1);
}
System.debug(listOfEmp);
if(listOfEmp.size() > 0){
    Database.insertImmediate(listOfEmp);
}
```

Want to view records of Big Object :-

As I already mention that there is no standard ui is available for Big Object. So if we want to see data then we can create our custom visualforce page and view data on it.

Here I created a vf page which display first name and last name of employee.

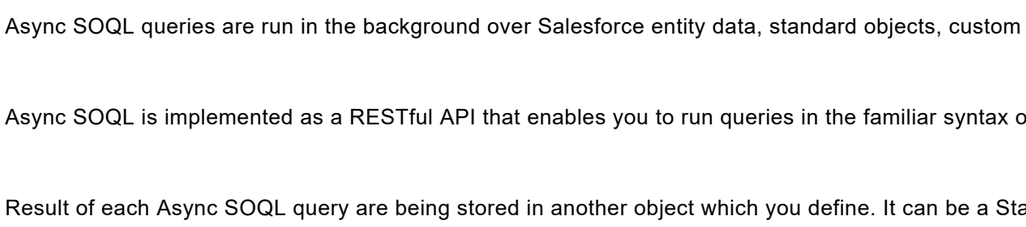
Class

```
public class BigObjExample{
    public List<Employee__b> listOfEmp {get;set;}
    public BigObjExample(){
        listOfEmp = [SELECT  iarpit__First_Name__c, iarpit__Last_Name__c FROM iarpit__Employee__b ];
        if(listOfEmp == null){
            listOfEmp = new List<Employee__b>();
        }
    }
}
```

Page:

```
<apex:page controller="BigObjExample" title="Big Object Example">
    <apex:sectionHeader title="Employee" subtitle="All Employee"/>
    <apex:pageBlock>
        <apex:pageBlockSection columns="1">
            <apex:pageBlockTable value="{!listOfEmp}" var="emp">
                <apex:column value="{!emp.first_name__c}"/>
                <apex:column value="{!emp.last_name__c}"/>
            </apex:pageBlockTable>
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:page>
```

Screenshot ;



SOQL In Big Object :-

There are some limitation for soql query on Big Object. We can use all the fields in ‘WHERE’ , all operators are not supported.

Only the indexed fields can be used in WHERE condition in soql.

You can use =, <, >, <=, or >=, or IN on the last field in your query. Any prior fields in your query can only use the = operator.
The !=, LIKE, NOT IN, EXCLUDES, and INCLUDES operators are not valid in any query

Async SOQL :

When we want to query large amount of data we go for Async SOQL .

Async SOQL queries are run in the background over Salesforce entity data, standard objects, custom objects, and big objects.

Async SOQL is implemented as a RESTful API that enables you to run queries in the familiar syntax of SOQL.

Result of each Async SOQL query are being stored in another object which you define. It can be a Standard , Custom or Big Object.

Note :Async SOQL for standard and custom objects are not generally available.

URL on which we send request for Async SOQL

We set query in the body of HttpRequest.

Example of Query :-

```
{
 "query": "SELECT iarpit__First_Name__c FROM iarpit__Employee__b",
 "operation": "insert",
 "targetObject": "iarpit__BigEmp__c",
```

```
 "targetFieldMap": {"iarpit__First_Name__c":"name"}
}
```

In this I have created a Custom Object BigEmp__c

Snippet of class using we can send request for Async SOQL :-

```
/*
Name     :     BigObjectToCustom
Date     :     07/11/2017
Author   :     Arpit Vijayvergiya
Description : this is being used for transfering data from Big object to Custom obje
*/
public class BigObjectToCustom{
    public BigObjectToCustom(){
        String query = '{"query": "SELECT iarpit__First_Name__c FROM iarpit__Employe
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setMethod('POST');
        request.setEndPoint('https://arpitvijayvergiya-dev-ed.my.salesforce.com/serv
        request.setHeader('Content-Type', 'application/json');
        request.setBody(query);
        request.setHeader('Authorization', 'Bearer ' + UserInfo.getSessionID());
        HttpResponse response = http.send(request);
        System.debug(response.getBody());
    }

}
```