Stored Procedures in IBM BPM

Stored Procedure : Stored procedures are stored inside the database. They define a programming interface for the database rather than allowing the client application to interact with database objects directly**.** When a stored procedure is called at the first time, SQL Server creates an execution plan and stores it in the plan cache. In the subsequent executions of the stored procedure, SQL Server reuses the plan so that the stored procedure can execute very fast with reliable performance.

Creating Stored Procedures :

Right click on stored procedure tab ->Select new procedure -> specify the parameters->click ok

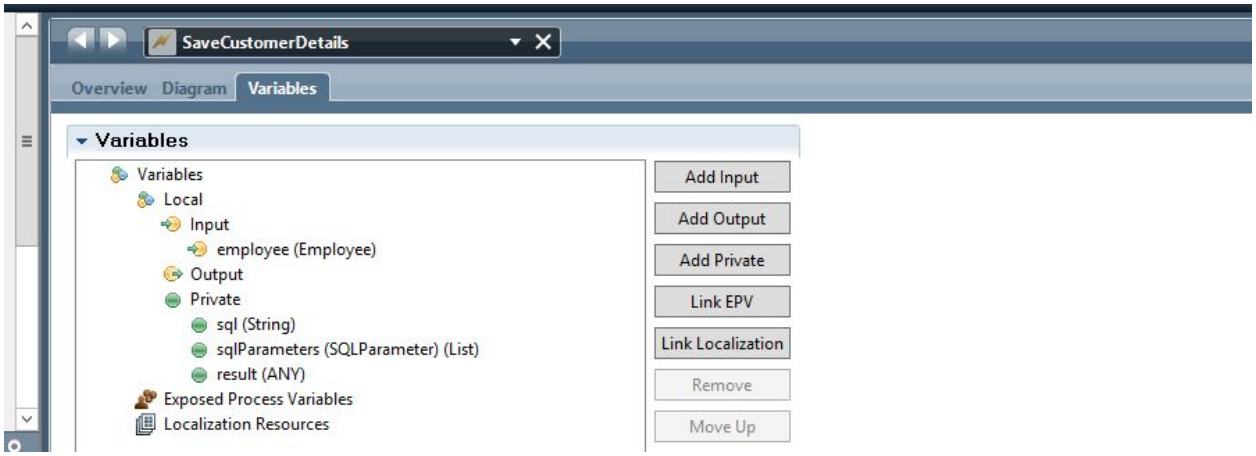
ex: Creating a stored procedure for insert operation

```
create or replace PROCEDURE SAVEEMPLOYEE
(
  EMPLOYEEID IN VARCHAR2
, EMPLOYEENAME IN VARCHAR2
, STATE1 IN VARCHAR2
, TOWN1 IN VARCHAR2
, DISTRICT1 IN VARCHAR2
, CREATEDDATE1 IN DATE
, LASTMODIFIEDDATE1 IN DATE
) AS
BEGIN

   insert into EMPLOYEE values(EMPLOYEEID,EMPLOYEENAME,TOWN1,DISTRICT1,STATE1,CREATEDDATE1,LASTMODIFIEDDATE1);

END SAVEEMPLOYEE;
```

After creating Procedure we need to compile it, At the time of compilation sql server create execution plan for procedure. After compilation we can execute procedure(using command **exec <procedure name >** from command prompt).

Calling a Stored procedure in BPM :

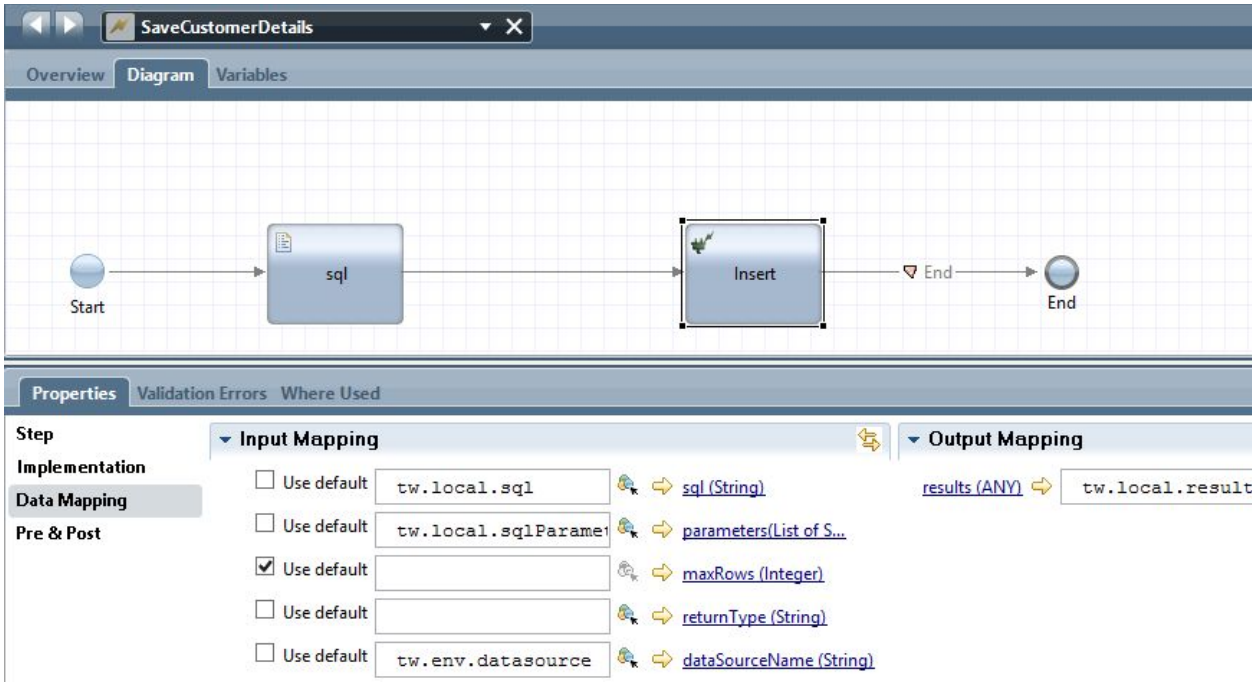
1. We need to create general system service with nested service component. For nested service open implementation tab and select Call Stored procedure service and pass required values to it.



2.Pass the parameters as below to the stored procedure.

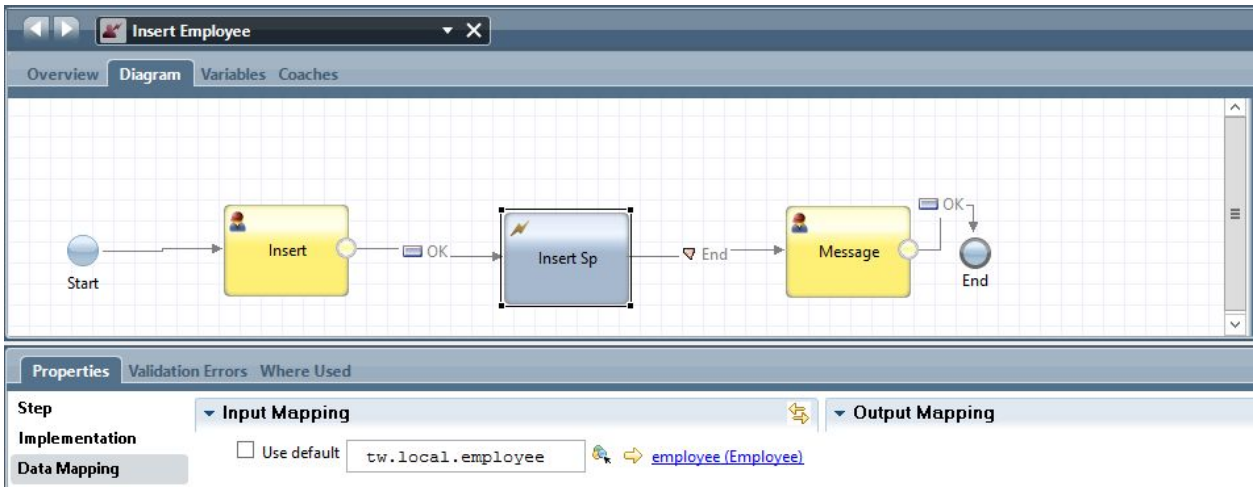
```
tw.local.sql="call SAVEEMPLOYEE(?,?,?,?,?,?,?)";
tw.local.sqlParameters= new tw.object.listOf.SQLParameter();

function setParams(count,type,mode,value){
tw.local.sqlParameters[count]=new tw.object.SQLParameter();
tw.local.sqlParameters[count].type=type;
tw.local.sqlParameters[count].mode=mode;
tw.local.sqlParameters[count].value=value;
}

setParams(0,"VARCHAR","IN",tw.local.employee.empId);
setParams(1,"VARCHAR","IN",tw.local.employee.empName);
setParams(2,"VARCHAR","IN",tw.local.employee.empAddress.state);
setParams(3,"VARCHAR","IN",tw.local.employee.empAddress.district);
setParams(4,"VARCHAR","IN",tw.local.employee.empAddress.street);
setParams(5,"DATE","IN",tw.local.employee.createdDate);
setParams(6,"DATE","IN",tw.local.employee.lastModifiedDate);
```

3.Open data mapping section for nested service and provide required values.

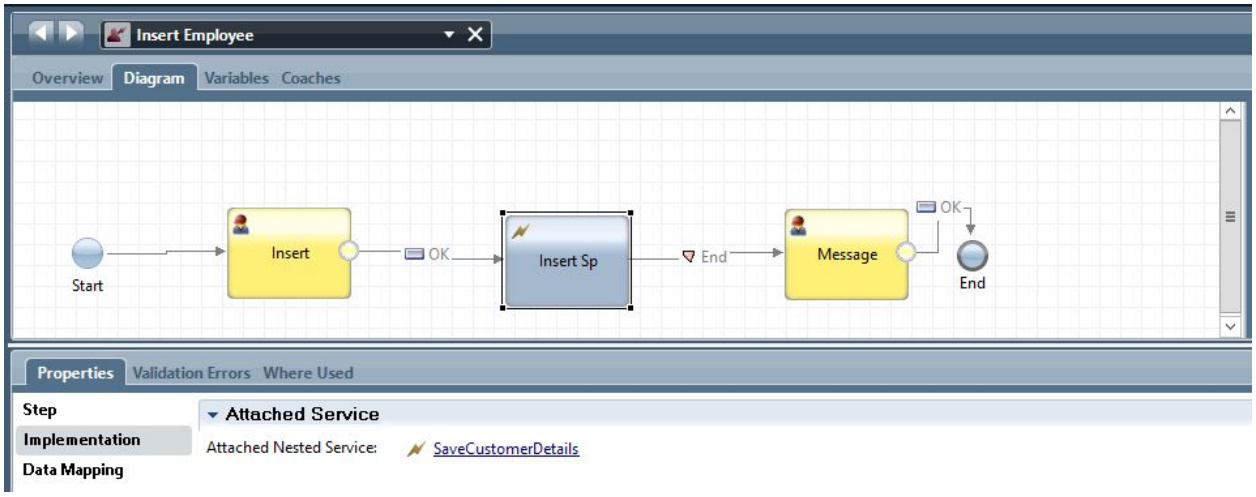


Next we will create a screen from where we will insert the details from the UI screen .

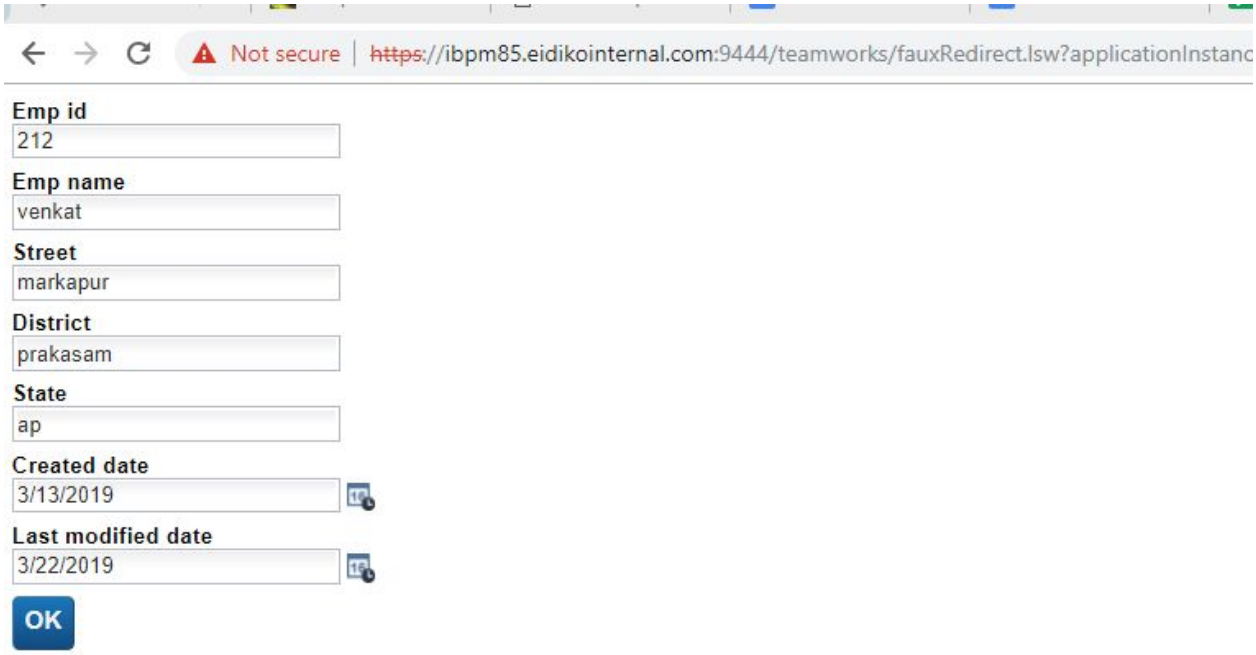
1.Create a Heritage Human service and drag a coach component and create employee as input variable.implement user screen by dragging employee variable into coach.

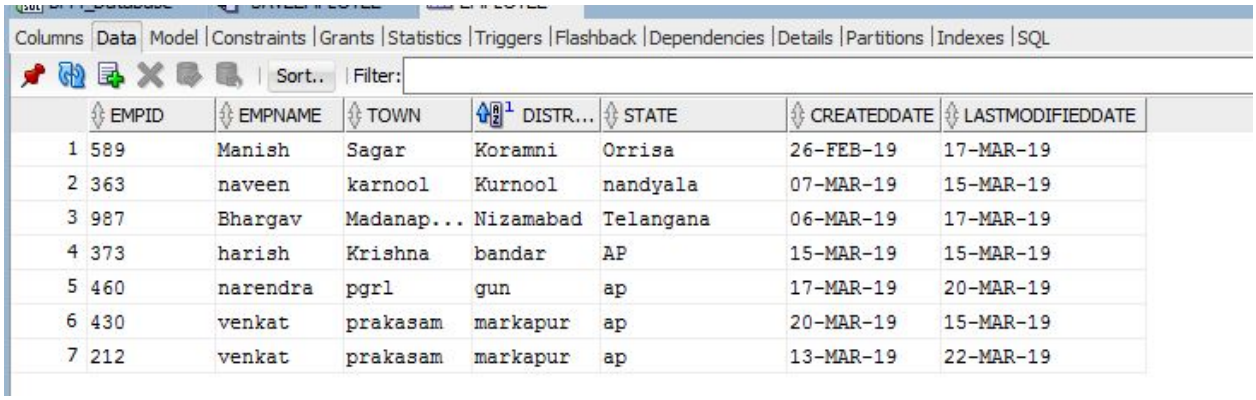


2.Use nested service , attach stored procedure general system service to it.



3.Now Run the Heritage HS(Insert Stored Procedure HS) and insert the details and check the table in database .





Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	EMPID	EMPNAME	TOWN	DISTR...	STATE	CREATEDDATE	LASTMODIFIEDDATE
1	589	Manish	Sagar	Koramni	Orrisa	26-FEB-19	17-MAR-19
2	363	naveen	karnool	Kurnool	nandyala	07-MAR-19	15-MAR-19
3	987	Bhargav	Madanap...	Nizamabad	Telangana	06-MAR-19	17-MAR-19
4	373	harish	Krishna	bandar	AP	15-MAR-19	15-MAR-19
5	460	narendra	pgrl	gun	ap	17-MAR-19	20-MAR-19
6	430	venkat	prakasam	markapur	ap	20-MAR-19	15-MAR-19
7	212	venkat	prakasam	markapur	ap	13-MAR-19	22-MAR-19

Similarly we can update the records by using update stored procedure syntax .The syntax is given below

```
create or replace PROCEDURE UPADATEEMPLOYEE (
  EMPLOYEEID IN VARCHAR2
, EMPLOYEENAME IN VARCHAR2
, STATE1 IN VARCHAR2
, TOWN1 IN VARCHAR2
, DISTRICT1 IN VARCHAR2
, CREATEDDATE1 IN DATE
, LASTMODIFIEDDATE1 IN DATE
) AS
BEGIN

  UPDATE  EMPLOYEE SET
EMPNAME=EMPLOYEENAME,TOWN=TOWN,DISTRICT=DISTRICT1,STATE=STATE1,CREA
TEDDATE=CREATEDDATE1,LASTMODIFIEDDATE=LASTMODIFIEDDATE1 WHERE
EMPID=EMPLOYEEID;


END UPADATEEMPLOYEE;
```

Similarly we can delete the records by using update stored procedure syntax .The syntax is given below .

```
create or replace PROCEDURE DELETEEMPLOYEE ( EMPLOYEEID IN VARCHAR2 ) AS
BEGIN
  Delete FROM EMPLOYEE WHERE EMPID=EMPLOYEEID;
END DELETEEMPLOYEE;
```

Similarly we can select the records by using update stored procedure syntax .The syntax is given below .

```
create or replace PROCEDURE <procedureName>
(
results OUT sys_refcursor
) AS
BEGIN
open results for SELECT * FROM EMPLOYEE;
END;
```

For nested BO we need to map data using below code.

```
tw.local.empDataOut=new tw.object.listOf.Employee();
for(var i=0;i<tw.local.results[0].rows.listLength;i++)
{
   tw.local.empDataOut[i]=new tw.object.Employee();
   tw.local.empDataOut[i].empId=tw.local.results[0].rows[i].indexedMap['EMPID'];
   tw.local.empDataOut[i].empName=tw.local.results[0].rows[i].indexedMap['EMPNAME'];

tw.local.empDataOut[i].lastModifiedDate=tw.local.results[0].rows[i].indexedMap['LASTMODIFIE
DDATE'];

tw.local.empDataOut[i].createdDate=tw.local.results[0].rows[i].indexedMap['CREATEDDATE'];
   tw.local.empDataOut[i].empAddress=new tw.object.Address();
   tw.local.empDataOut[i].empAddress.street=tw.local.results[0].rows[i].indexedMap['TOWN'];
   tw.local.empDataOut[i].empAddress.district=tw.local.results[0].rows[i].indexedMap['STATE'];
   tw.local.empDataOut[i].empAddress.state=tw.local.results[0].rows[i].indexedMap['DISTRICT'];

}
```