**Mark problem as resolved:**

resolvedProblem(ticketId in number, serviceCharge in number, serviceDate in date)

*Algorithm:* Procedure will mark the **issue as resolved** in the tables for the details corresponding to input parameters.

**Service charged** will be added to the account.

**Status** in the table is updated.

*Input:*1) ticketId

2) serviceCharge

3) serviceDate

*Output:* Message stating the issue is resolved and is updated in table.

/\* This procedure takes ticket id(ticket\_id),service charge(serviceCharge), and service date(servicedate) as an input \*/

Create or replace PROCEDURE resolved\_serviceProblem(ticket\_ID in number,serviceCharge in number,servicedate in timestamp) IS

a\_id int;

previous\_balance number;

final\_balance number;

Begin

select account.aid,account\_balance into a\_id,previous\_balance from service\_problem,account where ticket\_ID = tid and service\_problem.aid = account.aid;

-- This variable adds the service charge to the balance of the account

final\_balance := previous\_balance + serviceCharge;

/\* The problem will be updated to resolved,service charge will be added to the table and also the date the issue was resolved will be updated. \*/

UPDATE Service\_problem SET  status = 'Resolved' , service\_charge = serviceCharge, resolve\_date = servicedate WHERE tid = ticket\_id and status = 'in progress';

--The account balance will be updated to reflect the service charge

UPDATE account SET account\_balance = final\_balance WHERE a\_id = aid;

/\*A row will be inserted in the message table to notify that the problem has been fixed \*/

Insert Into Message Values (message\_id\_seq.nextval ,a\_id,'The ticket has been resolved',sysdate);

Exception

when no\_data\_found then Dbms\_output.put\_line('We can not find the ticket number.Please check the number and try again!');

when too\_many\_rows then dbms\_output.put\_line('too many rows');

End;

/\* example execution \*/

execute resolved\_serviceProblem(7,200, timestamp '2017-03-01 09:00:30.75');