Oracle Procedures

A procedure is a group of PL/SQL statements that can be called by name. The call specification (sometimes called call spec) specifies a java method or a third-generation language routine so that it can be called from SQL and PL/SQL.

Understanding Stored Procedure Syntax in Oracle

```
CREATE OR REPLACE PROCEDURE UPDATE_SAL
(
   P_EMP IN NUMBER,
   P_AMOUNT IN NUMBER
)
IS
BEGIN
   UPDATE Employee
   SET Salary=SALARY + P_AMOUNT
   WHERE Id=P_EMP_ID;
   COMMIT;
   EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLCODE);
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;



EXECUTE UPDATE_SAL(100,50);

BEGIN
UPDATE_SAL(100,50);
END;

//Using Anonymous Block in Oracle to Call Stored Procedures

BEGIN
UPDATE_SAL(&EMP_ID, &AMOUNT);
END;

select * from employee;




SELECT * FROM DBA_OBJECTS WHERE OBJECT_NAME='UPDATE_SAL';

SELECT * FROM USER_SOURCE WHERE NAME='UPDATE_SAL' ORDER BY LINE;
```

Parameter

`IN`

An `IN` parameter is read-only. You can reference an `IN` parameter inside a procedure, but you cannot change its value. Oracle uses `IN` as the default mode. It means that if you don’t specify the mode for a parameter explicitly, Oracle will use the `IN` mode.

`OUT`

An `OUT` parameter is writable. Typically, you set a returned value for the `OUT` parameter and return it to the calling program. Note that a procedure ignores the value that you supply for an `OUT` parameter.

`INOUT`

An `INOUT` parameter is both readable and writable. The procedure can read and modify it.

Note that `OR REPLACE` option allows you to overwrite the current procedure with the new code.

Create procedure

```
DROP Table Employee;

CREATE TABLE Employee (
  Id INT,
  Name VARCHAR(15),
  Salary NUMBER(8, 2)
);
```

```
create or replace procedure Insert_Employee
(Id INT,Name VARCHAR,Salary NUMBER)
is
begin
INSERT INTO Employee VALUES(Id, Name,Salary);
end;

begin
   Insert_Employee(1002, 'Smith', 45000);
end;

select * from Employee;

begin
   Insert_Employee(1003, 'STEVE', 45000);
```

```
end;

select * from Employee;

//-----------------------------------------------------

CREATE OR REPLACE PROCEDURE Update_Salary
(
   i INT,s NUMBER
)
IS
BEGIN
  UPDATE Employee
  SET salary=s
  WHERE id=i;
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


begin
   Update_Salary(1002,1300);
end;
select * from Employee;

//-----------------------------------------------

CREATE OR REPLACE PROCEDURE SHOW_EMP
(
    P_EMP_ID Employee.id%TYPE,
    P_F_NAME OUT Employee.name%TYPE,
    P_SAL OUT Employee.salary%TYPE
 )
IS
BEGIN
   SELECT name, salary
   INTO P_F_NAME, P_SAL
   FROM
   Employee
   WHERE id=P_EMP_ID;
   EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLCODE);
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;

//Calling

VARIABLE B_FIRST_NAME VARCHAR2(100)
VARIABLE B_SAL NUMBER
EXECUTE SHOW_EMP (1005,:B_FIRST_NAME, :B_SAL)
PRINT B_FIRST_NAME B_SAL;

//calling
DECLARE
```

```
V_FIRST_NAME Employee.Name%TYPE;
V_SAL Employee.Salary%TYPE;
BEGIN
  SHOW_EMP(1005,V_FIRST_NAME,V_SAL );
  DBMS_OUTPUT.PUT_LINE(V_FIRST_NAME);
  DBMS_OUTPUT.PUT_LINE(V_SAL);
END;
```

```
//-------------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE SHOW_EMP
(
    P_EMP_ID Employee.id%TYPE,
    P_F_NAME OUT Employee.name%TYPE,
    P_SAL OUT Employee.salary%TYPE
 )
IS
BEGIN
   SELECT name, salary
   INTO P_F_NAME, P_SAL
   FROM
   Employee
   WHERE id=P_EMP_ID;
   EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLCODE);
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

```
DECLARE
V_FIRST_NAME Employee.id%TYPE;
V_SAL Employee.salary%TYPE;
BEGIN
  SHOW_EMP(105,V_FIRST_NAME,V_SAL );
  DBMS_OUTPUT.PUT_LINE(V_FIRST_NAME);
  DBMS_OUTPUT.PUT_LINE(V_SAL);
END;
```

```
///-------------------------------------------------------
DROP TABLE PRODUCTS;
CREATE TABLE PRODUCTS
(
  PROD_ID NUMBER,
  PROD_NAME VARCHAR2(20),
  PROD_TYPE VARCHAR2(20),
  CONSTRAINT PRODUCTS_PK PRIMARY KEY (PROD_ID)
);
```

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='UNKNOWN',
   P_PROD_TYPE VARCHAR2 DEFAULT 'Unknown'
)
IS
BEGIN
  INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME ,
P_PROD_TYPE);
  DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||
'  INSERTED ' );
  COMMIT;

  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR IN INSERT '||P_PROD_ID||' '
||P_PROD_NAME);
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


BEGIN
ADD_PRODUCTS (10,'Bajaj');
ADD_PRODUCTS (10,'Laptop');
ADD_PRODUCTS (20,'Samsung');
END;

SELECT * FROM PRODUCTS;
```

```
create or replace procedure Insert_Employee
(Id INT,Name VARCHAR,Salary NUMBER)
is
begin
INSERT INTO Employee VALUES(Id, Name,Salary);
end;

begin
   Insert_Employee(1002, 'Smith', 45000);
end;

select * from Employee;

begin
   Insert_Employee(1003, 'STEVE', 45000);
end;

select * from Employee;

//-----------------------------------------------------

CREATE OR REPLACE PROCEDURE Update_Salary
(
   i INT,s NUMBER
```

```
)
IS
BEGIN
  UPDATE Employee
  SET salary=s
  WHERE id=i;
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


begin
   Update_Salary(1002,1300);
end;
select * from Employee;

//--------------------------------------------

CREATE OR REPLACE PROCEDURE SHOW_EMP
(
    P_EMP_ID Employee.id%TYPE,
    P_F_NAME OUT Employee.name%TYPE,
    P_SAL OUT Employee.salary%TYPE
 )
IS
BEGIN
   SELECT name, salary
   INTO P_F_NAME, P_SAL
   FROM
   Employee
   WHERE id=P_EMP_ID;
   EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLCODE);
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;

//-----------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE SHOW_EMP
(
    P_EMP_ID Employee.id%TYPE,
    P_F_NAME OUT Employee.name%TYPE,
    P_SAL OUT Employee.salary%TYPE
 )
IS
BEGIN
   SELECT name, salary
   INTO P_F_NAME, P_SAL
   FROM
   Employee
   WHERE id=P_EMP_ID;
   EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLCODE);
```

```
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;



DECLARE
V_FIRST_NAME Employee.id%TYPE;
V_SAL Employee.salary%TYPE;
BEGIN
  SHOW_EMP(105,V_FIRST_NAME,V_SAL );
  DBMS_OUTPUT.PUT_LINE(V_FIRST_NAME);
  DBMS_OUTPUT.PUT_LINE(V_SAL);
END;



///-----------------------------------------------------
DROP TABLE PRODUCTS;
CREATE TABLE PRODUCTS
(
  PROD_ID NUMBER,
  PROD_NAME VARCHAR2(20),
  PROD_TYPE VARCHAR2(20),
  CONSTRAINT PRODUCTS_PK PRIMARY KEY (PROD_ID)
);

CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='UNKNOWN',
   P_PROD_TYPE VARCHAR2 DEFAULT 'Unknown'
)
IS
BEGIN
  INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME ,P_PROD_TYPE);
  DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||'  INSERTED ' );
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR IN INSERT '||P_PROD_ID||' '||P_PROD_NAME);
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


BEGIN
ADD_PRODUCTS (10,'Bajaj');
ADD_PRODUCTS (10,'Laptop');
ADD_PRODUCTS (20,'Samsung');
END;

SELECT * FROM PRODUCTS;
```

Drop Procedore

```
DROP PROCEDURE Insert_Employee;
```

```
CREATE OR REPLACE PROCEDURE FORMATE_TEL
(
   P_TEL IN OUT VARCHAR2
)
IS
BEGIN
   P_TEL:=SUBSTR(P_TEL,1,3)||'('||SUBSTR(P_TEL,4,2)||')'||SUBSTR(P_TEL,7);
END;
```

```
VARIABLE B_TELEPHONE VARCHAR2(20);
EXECUTE :B_TELEPHONE:='1234567890';
EXECUTE FORMATE_TEL(:B_TELEPHONE);
PRINT B_TELEPHONE;
```

```
DECLARE
V_TEL VARCHAR2(100):='1234567890';
BEGIN
   FORMATE_TEL(V_TEL);
   DBMS_OUTPUT.PUT_LINE(V_TEL);
END;
```

Errors

```
CREATE OR REPLACE PROCEDURE UPDATE_SALARY
(
   P_EMP_ID IN NUMBER,
   P_AMOUNT IN NUMBER
)
IS
BEGIN
  UPDATE Employee
  SET Salary=Salary+P_AMOUNT
  WHERE Id=P_EMP_ID
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
```

```
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;

SELECT * FROM USER_ERRORS WHERE NAME = 'UPDATE_SALARY';
```

Comparison of Parameter Modes in Oracle

We have three parameter modes IN, OUT, IN/OUT parameter modes. We will be distinguishing each parameter mode from other modes.

IN	OUT	IN/OUT
Default Mode	Must be Specified	Must be Specified
Values are passed into a subprogram	Returned to the calling environment	Passed into subprogram returned to calling environment
The formal parameter acts as a constant	Uninitialized variable	Initialized variable
The actual parameter can be a literal expression, constant, or initialized variable	Must be a variable	Must be a variable
Can be assigned a default value	Cannot be assigned a default value	Cannot be assigned a default value

```
CREATE TABLE PRODUCTS
(
  PROD_ID NUMBER,
  PROD_NAME VARCHAR2(20),
  PROD_TYPE VARCHAR2(20),
  CONSTRAINT PRODUCTS_PK PRIMARY KEY (PROD_ID)
);
```

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2,
   P_PROD_TYPE VARCHAR2
)
IS
BEGIN
  INSERT INTO PRODUCTS VALUES (P_PROD_ID,P_PROD_NAME,P_PROD_TYPE);
  COMMIT;

  EXCEPTION
  WHEN OTHERS THEN
```

```
  DBMS_OUTPUT.PUT_LINE ('ERROR IN INSERT ');
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


EXECUTE ADD_PRODUCTS(1,'Bajaj','Bike');
SELECT * FROM PRODUCTS;
EXECUTE ADD_PRODUCTS(2, 'SAMSUNG');
EXECUTE ADD_PRODUCTS(1,'Bajaj','Bike');
```

Notations of Passing Parameters in Oracle Procedure

Now, we will discuss the available notations in oracle while passing parameters to stored procedures in oracle. When calling a subprogram, you can write the actual parameters using the following notations:

1. **Positional**
2. **Named**
3. **Mixed**

Named Notation for Passing Parameters in Oracle Stored Procedure:

This is very simple and, in this case, while calling the procedure we need to mention the parameter name. So, to execute the ADD_PRODUCTS procedure we need to call the procedure and pass the parameters as follows.

```
EXECUTE ADD_PRODUCTS (P_PROD_ID=>2, P_PROD_NAME=>'SAMSUNG', P_PROD_TYPE=>'MOBILE');
```

```
EXECUTE ADD_PRODUCTS (P_PROD_NAME=>'Pen', P_PROD_ID=>3, P_PROD_TYPE=>'Stationary');
```

Mixed Notation for Passing Parameters in Oracle Stored Procedure:

This is very simple. We can mix both positional and named notations and use them here. Let us try to understand Mixed Notation with an example. Please call the stored procedure as follows which uses Mixed Notation for Passing the Stored Procedure Parameters.

```
EXECUTE ADD_PRODUCTS (4, PROD_TYPE=>'Software', PROD_NAME=>'Windows 10');
```

Default Values for Parameters in Oracle Stored Procedure

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='UNKNOWN',
   P_PROD_TYPE VARCHAR2 DEFAULT 'UNKNOWN'
)
IS
BEGIN
  INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME, P_PROD_TYPE);
  DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||' INSERTED ' );
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR IN INSERT '||P_PROD_ID||' '||P_PROD_NAME);
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;
```

`EXECUTE ADD_PRODUCTS (10);`

Exception Handing in Oracle Stored Procedure

```
CREATE TABLE PRODUCTS
(
  PROD_ID NUMBER,
  PROD_NAME VARCHAR2(20),
  PROD_TYPE VARCHAR2(20),
  CONSTRAINT PRODUCTS_PK PRIMARY KEY (PROD_ID)
);

DELETE PRODUCTS;
SELECT * FROM PRODUCTS;
```

```
  CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='UNKNOWN',
   P_PROD_TYPE VARCHAR2 DEFAULT 'Unknown'
)
IS
BEGIN
  INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME ,P_PROD_TYPE);
  DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||'  INSERTED ' );
```

```
  COMMIT;
  EXCEPTION
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('ERROR IN INSERT '||P_PROD_ID||' '||P_PROD_NAME);
  DBMS_OUTPUT.PUT_LINE (SQLCODE);
  DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;


BEGIN
ADD_PRODUCTS (10,'Bajaj');
ADD_PRODUCTS (10,'Laptop');
ADD_PRODUCTS (20,'Samsung');
END;


SELECT * FROM PRODUCTS;


DELETE PRODUCTS;
SELECT * FROM PRODUCTS;


CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='Unknown',
   P_PROD_TYPE VARCHAR2 DEFAULT 'Unknown'
)
IS
BEGIN
   INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME, P_PROD_TYPE);
   DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||'  INSERTED ' );
   COMMIT;
END;


BEGIN
ADD_PRODUCTS (10,'Bajaj');
ADD_PRODUCTS (10,'Laptop');
ADD_PRODUCTS (20,'Samsung');
END;

SELECT * FROM PRODUCTS;

DELETE PRODUCTS;


CREATE OR REPLACE PROCEDURE ADD_PRODUCTS
(
   P_PROD_ID NUMBER,
   P_PROD_NAME VARCHAR2:='Unknown',
   P_PROD_TYPE VARCHAR2 DEFAULT 'Unknown'
)
IS
BEGIN
    INSERT INTO PRODUCTS VALUES (P_PROD_ID, P_PROD_NAME, P_PROD_TYPE);
    DBMS_OUTPUT.PUT_LINE (P_PROD_ID||' '||P_PROD_NAME||' INSERTED ' );
```

```
END;


BEGIN
ADD_PRODUCTS (10,'Bajaj');
ADD_PRODUCTS (10,'Laptop');
ADD_PRODUCTS (20,'Samsung');
COMMIT;
END;

SELECT * FROM PRODUCTS;
```

Boolean Parameters

```
CREATE OR REPLACE PROCEDURE P
(
   X BOOLEAN
)
IS
BEGIN
    IF X THEN
    DBMS_OUTPUT.PUT_LINE ('X is true');
    END IF;
END;


Declare
V Boolean;
V:=true;
P(V);
End;
```