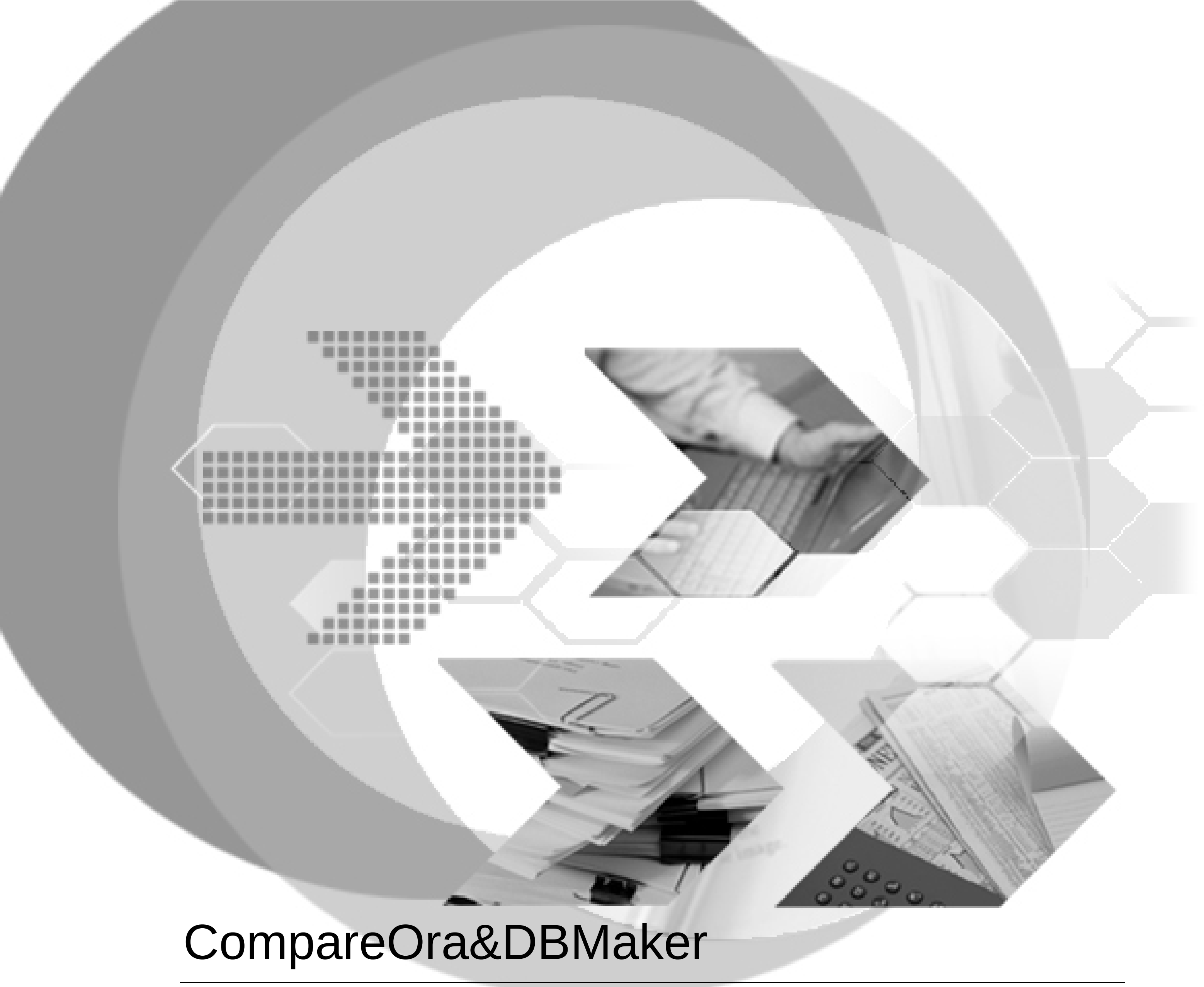


CompareOra&DBMaker

Reference document for PL_SQL SP

Compare to DBMaker 5.4 SQL SP

Version: DBMaker5.4 & Oracle 11g

Document No: **54/DBM54-T07222016-01-CSOD**
Author: **DBMaster Support Team & Production Team, Research & Development Division, SYSCOM Computer Engineering CO.**
Print Date: **July 22, 2016**

凌羣電腦
THE SYSCOM GROUP

Table of Content

1. Introductions 4

1.1 Purpose 4
1.2 Other Sources of Information 4
1.3 Technical Support 4

2. Stored Procedure Syntax 6

2.1 Syntax 6
2.1.1 CREATE SQL STORED PROCEDURE FROM FILE 7
2.1.2 CREATE SQL STORED PROCEDURE IN SCRIPT 7
2.2 Explanation of Create Syntax 8
2.3 Executing SQL Stored Procedure 8
2.4 Drop a Stored Procedure 9
2.5 Recompile a Stored Procedure 9
2.6 Getting Information on Procedures 9
2.7 Exception Function 9

3. The Possible Problems when change Database from Oracle to DBMaker 11

3.1 Supported Data Language 11
3.2 Case Sensitive 11
3.3 Reserved Word 11
3.4 Data Type 12
3.5 Automatic growth data type 15
3.6 Comment 16
3.7 The syntax of run SQL script 16
3.8 Variable Type 16
3.9 Define Variable 16
3.10 Assign value to variable 17
3.11 Parameter Type 17
3.12 ALTER PROCEDURE 17

3.13 Transaction 17
3.14 Cursor 18
3.14.1 Cursor type 18
3.14.2 Get Value 18
3.15 SELECT INTO STATEMENT 18
3.16 IF Judgment 20
3.16.1 IF-THEN-ENDIF statement 20
3.16.2 IF-THEN-ELSE-ENDIF statement 20
3.16.3 IF-THEN-ELSEIF-ENDIF statement 20
3.16.4 Case 20
3.17 WHILE LOOP 21
3.18 Loop 21
3.18.1 LOOP-EXIT-END LOOP 21
3.18.2 LOOP-EXIT WHEN-END LOOP 21
3.18.3 WHILE-LOOP-END LOOP 22
3.18.4 FOR-IN-LOOP-END 22
3.19 FOR IN in cursor 22
3.20 Cursor 23
3.20.1 The cursor with parameter 23
3.20.2 The cursor with for loop and where current of 24
3.20.3 The cursor with exception handle 24

4. The example for Create Stored Procedure......... 26

4.1 IF 26
4.1.1 IF ...THEN...END IF 26
4.1.2 IF...THEN...ELSE...END IF 27
4.1.3 IF...THEN...ELSIF...END IF 28
4.1.4 Nested IF 29
4.2 CASE 31
4.2.1 CASE Statement 31
4.2.2 Searching CASE 32
4.3 WHILE 34
4.4 FOR 35
4.5 About Cursor 37
4.5.1 Explicit cursor 37
4.5.2 The cursor with parameter 38
4.5.3 Cursor for Loop and where current of 40
4.6 Exception 40
4.6.1 Handle the named exception 40
4.6.2 The predefined Exception in oracle 42

4.6.3 USER DEFINED EXCEPTION 45
4.7 Record Type 46
4.7.1 RECORD 46
4.7.2 PL/SQL RECORD TYPE TABLE 47
4.7.3 ASSIGN VALUE FOR ROW OF THE TABLE IN LOOP 48
4.7.4 PL/SQL VARIABLE ARRAY 49
4.8 Integrated type stored procedure............................50
4.8.1 DISPLAY THE COLUMN OF TABLE（DYNAMIC SQL WITH INPUT PARAMETER IN） 50
4.8.2 THE PROCEDURE WITHOUT PARAMETER....................................52
4.8.3 THE STORED PROCEDURE WITH IN AND OUT PARAMETER.......54
4.8.4 SYS_REFCURSOR AND PROCEDURE CALL PROCEDURE 55
4.8.5 THE SP AND ITS PARAMETER TYPE IS IN OUT 56

5. Some features need to be cared in stored procedure.. 59
5.1 Data table alias 59
5.2 When select one of the column in stored procedure59
5.3 When use select into syntax in stored procedure...59
5.4 The relationship about alias and column name in stored procedure 60

6. Oracle & DBMaker SP compare conclusion....... 61
6.1 Basic comparison for SP.......................................61

1. Introductions

1.1 Purpose

This document mainly used to introduce the SP of PL/SQL, and all of the examples focusing on if can migrate PL/SQL SP to DBMaker as the main line to write. About the syntax and functions what are supported in DBMaker but Oracle, we not mentioned in this document, such as, in DBMaker, DDL statement can be included in SQL SP, repeat can in LOOP, but not do any test in Oracle for such unsupported aspects.

1.2 Other Sources of Information

DBMaster standard version provides many other user's guides and reference manuals in addition to this reference. For more information on a particular subject, consult one of these books:

- For more information on the designing, administering, and maintaining a DBMaster database, refer to the *"*Database Administrator's Guide"*.*
- For more information on the SQL language implemented by DBMaster, refer to the "SQL Command and Function Reference".
- For more information on the ESQL/C language implemented by DBMaster, refer to the "ESQL/C Programmer's Guide".
- For more information on error and warning messages, refer to the "Error and Message Reference".
- For more information on using stored procedure, refer to the "Stored Procedure user's guide".

1.3 Technical Support

CASEMaker provides thirty days of complimentary email and phone support during the evaluation period. When software is registered an additional thirty days of support will be included. Thus extend the total support period for software to sixty days. However, CASEMaker will continue to provide email support for any bugs reported after the complimentary support or registered support has expired (free of charges).

Additional support is available beyond the sixty days for most products and may be purchased for twenty percent of the retail price of the product. Please contact sales@casemaker.com for more details and prices.

CASEMaker support contact information for your area (by snail mail, phone, or email) can be located at: http://www.casemaker.com/support. It is recommended that the current database of FAQ's be searched before contacting CASEMaker support staff.

Please have the following information available when phoning support for a troubleshooting enquiry or include the information with a snail mail or email enquiry:

- Product name and version number
- Registration number
- Registered customer name and address
- Supplier/distributor where product was purchased
- Platform and computer system configuration
- Specific action(s) performed before error(s) occurred
- Error message and number, if any
- Any additional information deemed pertinent

2. Stored Procedure Syntax

2.1 Syntax

Oracle

```
CREATE OR REPLACE PROCEDURE Procedure Name
(
   Parameter1 IN NUMBER,
   Parameter2 IN NUMBER
) IS
Variable1 INTEGER :=0;
Variable2 DATE;
BEGIN
DML
END Procedure Name
```

```
CREATE [OR REPALCE] PROCEDURE < Procedure Name >
[(Parameter1 [IN | OUT | IN OUT] datatype [, Parameter2 [IN | OUT | IN OUT] datatype]…)]
 IS|AS [Declare statement]
 BEGIN
 Execute statements
 [EXCEPTION Exception Handlers]
  END;
```

OR REPLACE means the procedure is to replace an existing procedure.

Procedure-name is the name of the procedure.

Parameter is the name of a parameter that is passed to the procedure.

IN, OUT, IN OUT is the mode of the parameter

IN：An IN parameter must be set to a value when the procedure is run. The value of an IN parameter cannot be changed in the procedure body. It’s the default mode.

OUT：The parameter means that the procedure passes a value back to the caller through this argument， the value is write-only and cannot allocate the default value. If the procedure execute successfully, it will be assigned.

IN OUT：Specify a value, and the procedure replaces it with a value. The value will be read then written.

Type is the parameter type.

Procedure_body contains the actual code for the procedure.

DBMaker

```
CREATE OR REPLACE PROCEDURE project_name.module_name.sp_name [arg_list]
LANGUAGE SQL
BEGIN
 [Declare list]
[statement list]
END;
```

OR REPLACE is used to re-create the procedure if it already exists, that is to say, you can use this clause to change the definition of an existing procedure.

NOTE *Not support CREATE OR REPLACE PROCEDURE syntax while setting AUTOCOMMIT OFF.*

2.1.1 CREATE SQL STORED PROCEDURE FROM FILE

SQL stored procedure can be created using an external file (*.sp). Use the dmSQL command line tool to create SQL stored procedures by referencing external files as shown in the following example.

a) Save SQLSP script to a file.

```
CREATE PROCEDURE project_name.module_name.sp_name [arg_list]
LANGUAGE SQL
BEGIN
[declare_list]
[statement_list]
END;
```

b) To create a procedure using the following syntax.

```
dmSQL> Create procedure from path\*.sp
```

Or

```
dmSQL> Create or replace procedure from path\*.sp
```

2.1.2 CREATE SQL STORED PROCEDURE IN SCRIPT

From DBMaker 5.3, users can create SQL stored procedures not only from files, but also in dmSQL.

The CREATE PROCEDURE syntax as follows:

```
dmSQL> set block delimiter @@;
dmSQL> @@
2> create procedure sp_in_script2
3> language sql
4> begin
5> insert into t1 values(1);
6> end;
7> @@
dmSQL> set block delimiter;
```

NOTE *SQLSP contains more than one SQL statements, and each statement is end of ';'. So dmSQL must support block delimiter. Block delimiter can be a serial of a-z, A-Z, @, %, ^, and contains two characters at least and seven characters at most. In block delimiter, ';' doesn't denote end of the input. Users must set block delimiter before write SQL stored procedure in dmSQL, otherwise, it will return error.*

2.2 Explanation of Create Syntax

Oracle

First line：CREATE OR REPLACE PROCEDURE is a SQL statement which used to inform Oracle to create a stored procedure named sp_name, if it had existed in database, replace it. Parameter list including the calling procedure is passed to the procedure of parameter name and this procedure return to calling procedure information.

Second line：IS keyword shows the PL/SQL block will follow behind. Local variable and variable are declared after IS. If there's not any local variable need to declare, nothing needed between IS and BEGIN.

Third line：BEGIN keyword shows the beginning of PL/SQL block. Execute statements write behind of BEGIN, front of EXCEPTION or END. There must have one executable statement at least in the process. Keyword EXCEPTION and Exception Handlers are optional.

The fourth line：Execute statements. There must have one executable statement at least in PL/SQL.

The fifth line：END keyword shows PL/SQL block end.

DBMaker

LANGUAGE SQL：DBMaker SQL SP msut declare with this keyword, because DBMaker supports stored procedures not only SQL SP but also E/SQL SP, JAVA SP.

Block header：It's used to describe the name and parameter of SQL SP.

Block end：It show block end.

Arg_list：It's a parameter list, input parameters and return parameters in declare, and if the parameter is NULL, There will be no return value.

Declare_list：It's a variable declare list.

Statement_list：It's a process control statement, which consists of a variety of different control statements and SQL statements, to perform the operation.

2.3 Executing SQL Stored Procedure

Oracle

```
Execute sp_name
```

Call method: You can call procedure with PL/SQL. The method is specify the Procedure Name, if have parameters that still need to specify the parameter list in parentheses, call statements were separated by ";".

Name Method: Specify the name and value of parameters when call.

<Procedure Name>（variable1＝value1[，variable2＝value2[，…]]）

Location Method: Not need to specify parameter name but need to the parameter location and procedure parameter keep consistent.

<Procedure Name>（parameter1[，parameter2[，…]]）or

e.g.：

```
DECLARE
          v_no varchar2 (6);
```

```
      BEGIN
           v_no:='1001';
           proc_custom_back (v_no);
      END;
     or
     EXEC  proc_wzbm_back ('1001');
```

DBMaker

```
Call sp_name
```

2.4 Drop a Stored Procedure

Oracle

```
Drop procedure sp_name
```

DBMaker

```
Drop procedure sp_name
```

2.5 Recompile a Stored Procedure

Oracle

```
ALTER PROCEDURE procedure name COMPILE;
```

DBMaker

```
Not support recompile, that need to drop then recreate.
```

2.6 Getting Information on Procedures

Oracle

```
SELECT object_name, aggregate, parallel
FROM user_procedures
WHERE object_name = 'spname';
```

DBMaker

```
dmSQL> SELECT * FROM SYSPROCINFO;
dmSQL> SELECT * FROM SYSPROCPARAM;
```

2.7 Exception Function

Oracle

SQLCODE: The function use to return an error code, it's a minus. We can assign it to a NUMBER variable.

SQLERRM: The function use to return error information with related error code. The max length of error information is 512 bytes. We can assign it to a VARCHAR2 variable.

DBMaker

SQLCODE: Error code

SQLSTATE: State

3. The Possible Problems when change Database from Oracle to DBMaker

Since the logic doesn't change, the principle is not to modify the application code, just modify the created tables and initialized SQL.

We will list our encountered problems and corresponding solution as below.

3.1 Supported Data Language

Oracle

PL/SQL unsupport Data Definition Language DDL statement, for example, CREATE.ALTER and DROP. Data Control Language Data Control Language DCL statement GRANT and REVOKE are invalid in PL/SQL.

DBMaker

DBMaker SQL SP support Data Definition Language DDL statement, CREATE ALTER,DROP.

3.2 Case Sensitive

Oracle

Generally, it's case-insensitive in oracle.

DBMaker

In DBMaker, the SQL SP needs to be capitalized.

NOTE *So that need to ensure the SP of Oracle needs to create with capital letters.*

3.3 Reserved Word

Oracle

In Oracle, the reserved word can create as table name or filed name and does not affect normal use.

DBMaker

In DBMaker, the reserved word can't create as table name or filed name, if attempting to add an object that uses a reserved word as an identifier will return an error by default.

The solve methods as below:

1) Add “” for reserved word in SQL statement.

2) Set **DB_ResWd** to 0, that will allows objects to be imported that contain reserved words and not return an error.

3.4 Data Type

Oracle

Basic Data Type

Composite Data Type: %type

Record Type Variable: %rowtype

Syntax format:

```
TYPE recordtypename IS RECORD
(Var_name1 datatype
……
) ;
Recordname recordtypename；
```

PL/SQL Table:

One Dimension Data Type

Multi-Dimension Data Type

Same with record declare, PL/SQL table type declared in two steps.

1) Declare a PL/SQL table type with TYPE statement, the structure can use any scalar data types.

2) Based on the declared type in previous step, declare a practical table.

Syntax format

```
TYPE tabletypename IS TABLE OF
Datatype
INDEX BY BINARY_INTEGER;
```

PL/SQL Variable array

Same with table declare, PL/SQL VARRAY declared in two steps.

1) Declare a PL/SQL VARRAY type with TYPE statement. TYPE declaration include a value which used to set the Max limit for VARRAY, the Min limit is always 1.

2) Based on the declared type in previous step, declare a VARRAY.

Syntax:

```
DECLARE
TYPE varraytypename IS VARRAY(size) OF ElementType；
Varrayname varraytypename；
```

The conclusion of data type in Oracle11g which can be used in PL/SQL

- Character datatypes

aTypeSyntx	Oracle 11g	Explanation
char(size)	Maximum size of 2000 bytes.	Where ***size*** is the number of characters to store. Fixed-length strings. Space padded.
nchar(size)	Maximum size of 2000 bytes.	Where ***size*** is the number of characters to store. Fixed-length NLS string Space padded.
nvarchar2(size)	Maximum size of 4000 bytes.	Where ***size*** is the number of characters to store. Variable-length NLS string.
varchar2(size)	Maximum size of 4000 bytes.	Where ***size*** is the number of characters to store. Variable-length string.
long	Maximum size of 2GB.	Variable-length strings. (backward compatible)
raw	Maximum size of 2000 bytes.	Variable-length binary strings
long raw	Maximum size of 2GB.	Variable-length binary strings. (backward compatible)

- Numeric Datatypes

DataTypeSyntax	Oracle 11g	Explanation
number(p,s)	Precision can range from 1 to 38. Scale can range from -84 to 127.	Where ***p*** is the precision and ***s*** is the scale. For example, number (7, 2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
numeric(p,s)	Precision can range from 1 to 38.	Where ***p*** is the precision and ***s*** is the scale. For example, numeric (7, 2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
float		
dec(p,s)	Precision can range from 1 to 38.	Where ***p*** is the precision and ***s*** is the scale. For example, dec (3, 1) is a number that has 2 digits before the decimal and 1 digit after the decimal.
decimal(p,s)	Precision can range from 1 to 38.	Where ***p*** is the precision and ***s*** is the scale. For example, decimal (3, 1) is a number that has 2 digits before the decimal and 1 digit after the decimal.
integer		
int		
smallint		
real		

double precision		

- Date/Time Datatypes

Data Type Syntax	Oracle 11g	Explanation
date	A date between Jan 1, 4712 BC and Dec 31, 9999 AD.	
timestamp (*fractional seconds precision*)	***fractional seconds precision*** must be a number between 0 and 9. (default is 6)	Includes year, month, day, hour, minute, and seconds. For example: timestamp(6)
timestamp (*fractional seconds precision*) with time zone	***fractional seconds precision*** must be a number between 0 and 9. (default is 6)	Includes year, month, day, hour, minute, and seconds; with a time zone displacement value. For example: timestamp(5) with time zone
timestamp (*fractional seconds precision*) with local time zone	***fractional seconds precision*** must be a number between 0 and 9. (default is 6)	Includes year, month, day, hour, minute, and seconds; with a time zone expressed as the session time zone. For example: timestamp(4) with local time zone
interval year (*year precision*) to month	***year precision*** is the number of digits in the year. (default is 2)	Time period stored in years and months. For example: interval year(4) to month
interval day (*day precision*) to second (*fractional seconds precision*)	***day precision*** must be a number between 0 and 9. (default is 2) ***fractional seconds precision*** must be a number between 0 and 9. (default is 6)	Time period stored in days, hours, minutes, and seconds. For example: interval day(2) to second(6)

- Large Object (LOB) Datatypes

DataTypeSyntax	Oracle 11g	Explanation
bfile	Maximum file size of 2^{64}-1 bytes.	File locators that point to a binary file on the server file system (outside the database).
blob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage).	Stores unstructured binary large objects.
clob	Store up to (4 gigabytes -1) * (the value of the	Stores single-byte and multi-byte

	CHUNK parameter of LOB storage) of character data.	character data.
nclob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage) of character text data.	Stores unicode data.

- Rowid Datatypes

DataTypeSyntax	**Oracle 11g**	**Explanation**
rowid	The format of the rowid is: BBBBBBB.RRRR.FFFFF Where BBBBBBB is the block in the database file; RRRR is the row in the block; FFFFF is the database file.	Fixed-length binary data. Every record in the database has a physical address or **rowid**.
urowid(size)		Universal rowid. Where ***size*** is optional.

DBMaker

Support the following 14 data types:

SMALLINT, INTEGER, BIGINT, FLOAT, DOUBLE, DECIMAL, DATE, TIME, TIMESTAMP, BINARY, CHAR, VARCHAR, NCHAR, NVARCHAR.

NOTE *replace*

e.g.:

```
varchar2<-→varchar
Number←→numeric number (1), numeric (1)
```

3.5 Automatic Growth Data Type

Oracle

Oracle doesn't have auto growth data type, that needs to create an automatic growth sequence, and give the next value to this filed when insert record.

CREATE SEQUENCE sequence_name (It's better table name + sequence flag) INCREMENT BY 1 START WITH 1

MAXVALUE 99999 CYCLE NOCACHE;

The Max value decided by the field length, if define automatic growth sequence NUMBER (6), the max value is 999999

INSERT statement insert value should be sequence_name.NEXTVAL

DBMaker

Serial

3.6 Comment

Oracle

-- /**/

DBMaker

--/**/

3.7 The Syntax of Run SQL Script

Oracle

```
@path\file_name.sql
```

DBMaker

```
Run "path\file_name.sql";
```

NOTE *In Oracle, the space wasn't allowed in filename of path（sqlplus can't properly recognize ）and the script is saved using suffix '.sql'. In DBMaker, the filename of path can include space（e.g：sql sp）， and the script suffix not any limit, but you'd better chose representative suffix, such as'.sp' and so on.*

3.8 Variable Type

Oracle

In oracle: Local variable and binding variable（Host Variable）

Declare Local variable: variable_name type

Declare Host variable: VARIBLE variable_name type and need to add ':' in front of variable_name before using e.g.: ':variable_name';

DBMaker

Don't distinguish.

3.9 Define Variable

Oracle

Variable Data type

In Oracle, you also can use %type to anchor a variable data type.

e.g.:

```
Declare
var1 number（3）；
var2 var1%type；
```

DBMaker

Declare variable data type

3.10 Assign Value to Variable

Oracle

```
Name char (10)
Age int
Name: = 'SNOW'
Age: = 23
```

DBMaker

```
Declare name char (10)
Set name = 'snow'
```

3.11 Parameter Type

Oracle

Stored procedure is an PL/SQL program block, to receive 0 or more parameters as INPUT or OUTPUT、or both take it as input and output(INOUT).

DBMaker

In DBMaker, there're 2 type parameters: To receive 0 or more parameters as IN/INPUT, or OUT/OUTPUT

3.12 ALTER PROCEDURE

Oracle

You can modify the created stored procedure with ALTER PROCEDURE command.

```
alter procedure procedure-name compile
```

DBMaker

It couldn't modify, you need to drop it, then recreate.

3.13 Transaction

Oracle

It can include transaction, commit, rollback

```
Create or replace procedure emptyparampro
Is
  Errorcode varchar2 (50);
  Error's varchar2 (200);
Begin
  Insert into budgetsection (year, bid) values (2005,'401');
  Commit;-- used to commit trasaction
  Exception--used to throw exception
  When others then
    Errorcode: = sqlcode;
    Errorstr: = sqlerrm (sqlcode);
```

```
End;
DBMaker：
```

3.14 Cursor

Oracle

3.14.1 CURSOR TYPE

PL/SQL have 2 type cursor：Static cursors and Dynamic cursors

Static cursors：The content of this cursor will be known when compile. Such Cursor object which includes one SQL always based on one SQL statement.

Static cursors also have 2 types: **Implicit cursors** and **explicit cursor**

Implicit：We don't declare such cursor, for each DML which was declared, managed and closed by PL/SQL.

Explicit cursor：When one SQL statement of PL/SQL return multi records from the base table, we declare an explicit cursor. The rows of explicit cursor which queried by such SQL statement form the activity set, when open cursor, it will point to the first line of the activity set, it just can queried and operated one record of the activity set for each time. The pointer will move to the net line when get one row, and the cursor will return it points to.

Dynamic cursors：Such type cursor using the cursor variable which value can be changed, this variable can be used to different SQL statement in different time.

3.14.2 GET VALUE

Oracle

In oracle, the cursor doesn't has such keywords **NEXT | PRIOR | FIRST | LAST.**

DBMaker

In DBMaker, the cursor has these keywords.

3.15 SELECT INTO STATEMENT

Oracle

Put the select result to the variable, and you also can put multi-columns save to multi-variables, and must have one record else the exception will be throwed.(if not record will throw NO_DATA_FOUND).

For example:

```
BEGIN
   select col1,col2 into variable1,variable2 FROM table-name where xxx;
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
          xxxx;
END;
```

DBMaker

But such type SP couldn’t execute in DBMaker, ERROR INFO

ERROR (6589): [DBMaker] preprocessor translating error: error message in: TEST1SYSADM.msg

```
CREATE PROCEDURE TEST1 (OUT VAL1 INT, OUT VAL2 INT)
LANGUAGE SQL
BEGIN

    DECLARE TMP_VAL1 INT;
    DECLARE TMP_VAL2 INT;
     WHILE (SQLSTATE = '00000')
        DO
    SELECT C1, C2 FROM TB2 INTO TMP_VAL1, TMP_VAL2 WHERE C1 = 1;
      END WHILE;

    SET VAL1 = TMP_VAL1;
    SET VAL2 = TMP_VAL2;
END;
```

Modify it to executable sp

```
CREATE PROCEDURE TEST1 (OUT VAL1 INT, OUT VAL2 INT)
LANGUAGE SQL
BEGIN

    DECLARE TMP_VAL1 INT;
    DECLARE TMP_VAL2 INT;
     #WHILE (SQLSTATE = '00000')
        #DO
    #SELECT C1, C2 FROM TB2 INTO TMP_VAL1,TMP_VAL2 WHERE C1 = 1;
//In DBMamker SQLSP that don’t allow assign value to variable by select keyword
    #SET TMP_VAL1 = SELECT C1 FROM TB2 WHERE C1 = 1;
    #SET TMP_VAL2 = SELECT C2 FROM TB2 WHERE C1 = 1;

    #SET TMP_VAL1 = 1;
    #SET TMP_VAL2 = 1000;

    DECLARE CUR CURSOR FOR SELECT C1,C2 FROM TB2 WHERE C1 = 1;
    OPEN CUR;
    FETCH CUR INTO TMP_VAL1, TMP_VAL2;
    CLOSE CUR;
//In DBMaker,if using the SELECT to assign value to variable that must using it in cursor

      #ENDS WHILE;

    SET VAL1 = TMP_VAL1;
    SET VAL2 = TMP_VAL2;
END;
```

3.16 IF Judgment

Oracle

```
IF V_TEST=1 THEN
    BEGIN
     Do something
   END;
END IF;
```

3.16.1 IF-THEN-ENDIF STATEMENT

```
IF condition THEN
   Statement Group；
END IF；
```

3.16.2 IF-THEN-ELSE-ENDIF STATEMENT

```
IF condition THEN
   Statement group1；
ELSE
   Statement group2；
END IF；
```

3.16.3 IF-THEN-ELSEIF-ENDIF STATEMENT

Oracle

```
IF condition1 THEN
   Statement group1；
ELSIF condition2 THEN//it different from ELSEIF of DBMaker
    Statement group2；
[ELSIF condition3 THEN
   statement3；…]
END IF；
```

DBMaker

```
IF <condition_value> THEN <sp_statement_main>
[ELSEIF <condition_value> THEN <sp_statement_main>]
[ELSE <sp_statement_main>]
END IF
```

3.16.4 CASE

```
CASE [variable_name]
When value1\condition1 then action_statement1；
When value2\condition2 then action_statement1；
……
Else action_statement；
End case；
```

Oracle

CASE statement: The statement with value become a case statement, case statement use the variable name as selector.

Search CASE statement: The statement with condition become a search CASE statement, search CASE statement cannot use variable name as selector.

DBMaker

Same with Oracle

3.17 WHILE LOOP

Oracle

```
WHILE V_TEST=1 LOOP
    BEGIN
      XXXX
    END;
 END LOOP;
```

DBMaker

```
WHILE <condition_name>
 DO
<sp_statement_main>
END WHILE
```

3.18 Loop

Oracle

WHILE, LOOP, FOR

DBMker

FOR, LOOP, WHILE,REPEAT；

3.18.1 LOOP-EXIT-END LOOP

Oracle

```
LOOP
Statement group1；
IF condition THEN
EXIT；
END IF；
Statement group2；
END LOOP；
```

3.18.2 LOOP-EXIT WHEN-END LOOP

Oracle

```
LOOP
  Statement group1；
  EXIT WHEN condition ;
   Statement;
END LOOP;
```

3.18.3 WHILE-LOOP-END LOOP

Oracle

```
WHILE condition LOOP
     Statement group;
END LOOP;
```

3.18.4 FOR-IN-LOOP-END

```
FOR loop variable IN[REVERSE] lower_bound…upper_bound LOOP
                  Statement group;
                END LOOP;
```

3.19 FOR IN in cursor

Oracle

Syntax:

```
FOR variable-name IN cursor-name LOOP
           Statement group;
         END LOOP;
```

NOTE *The FOR loop of cursor can automatically declare a row which can accept the cursor , open cursor and get data from cursor, then close cursor until get the last row data from cursor.*

```
CREATE OR REPLACE PROCEDURE TEST (name in number)
   IS
CURSOR cur IS select * FROM xxx; define a cursor and put the query result to it, the
cursor same as a pointer.
   BEGIN
      FOR cur_result in cur LOOP
         BEGIN
         V_SUM :=cur_result.colum-name;
         END;
      END LOOP;
END;
```

For example:

```
      CREATE OR REPLACE PROCEDURE TEST (name in number)
      Is
      V_SUM varchar2 (200);
      Cursor cur is select * from LAD_USER;
      BEGIN
         For V_RESULT in cur LOOP
             BEGIN
                   V_SUM:= V_RESULT.User_Name;
             END;
         End LOOP;
         DBMS_OUTPUT.put_line(V_SUM); --output command
      END TEST;
```

DBMaker

```
FOR [<var_name> AS] cursor_name [CURSOR FOR <select_statement>;]
DO <sp_statement_main>;
END FOR
```

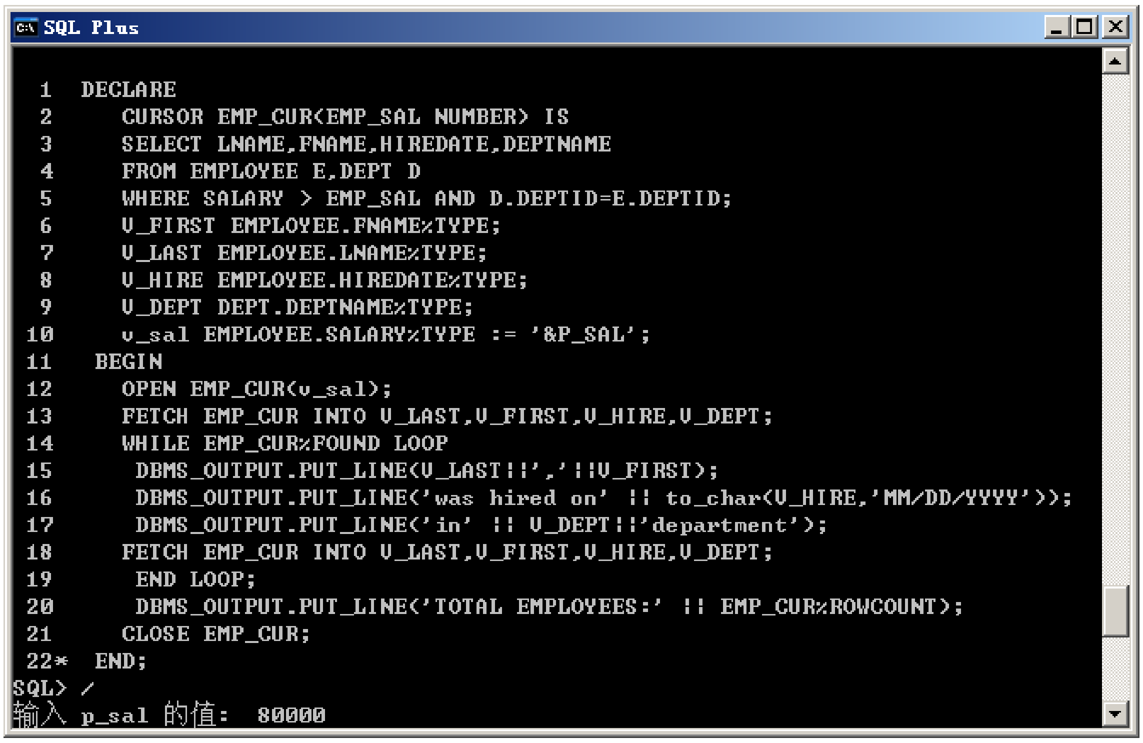
3.20 Cursor

3.20.1 THE CURSOR WITH PARAMETER

Oracle

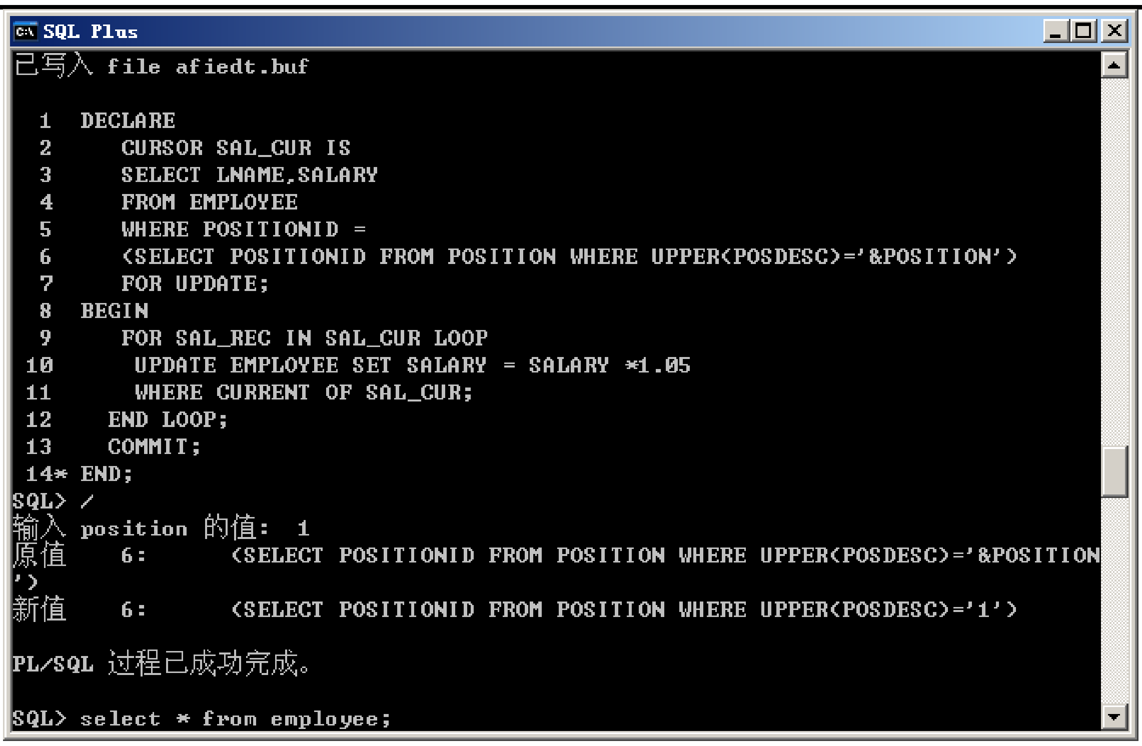
```
CURSOR C_USER (C_ID NUMBER) IS select NAME FROM USER where TYPEID=C_ID;
   OPEN C_USER(variable-value);
LOOP
  FETCH C_USER INTO V_NAME;
  EXIT FETCH C_USER%NOTFOUND;
      Do something
END LOOP;
CLOSE C_USER;
```

For example：



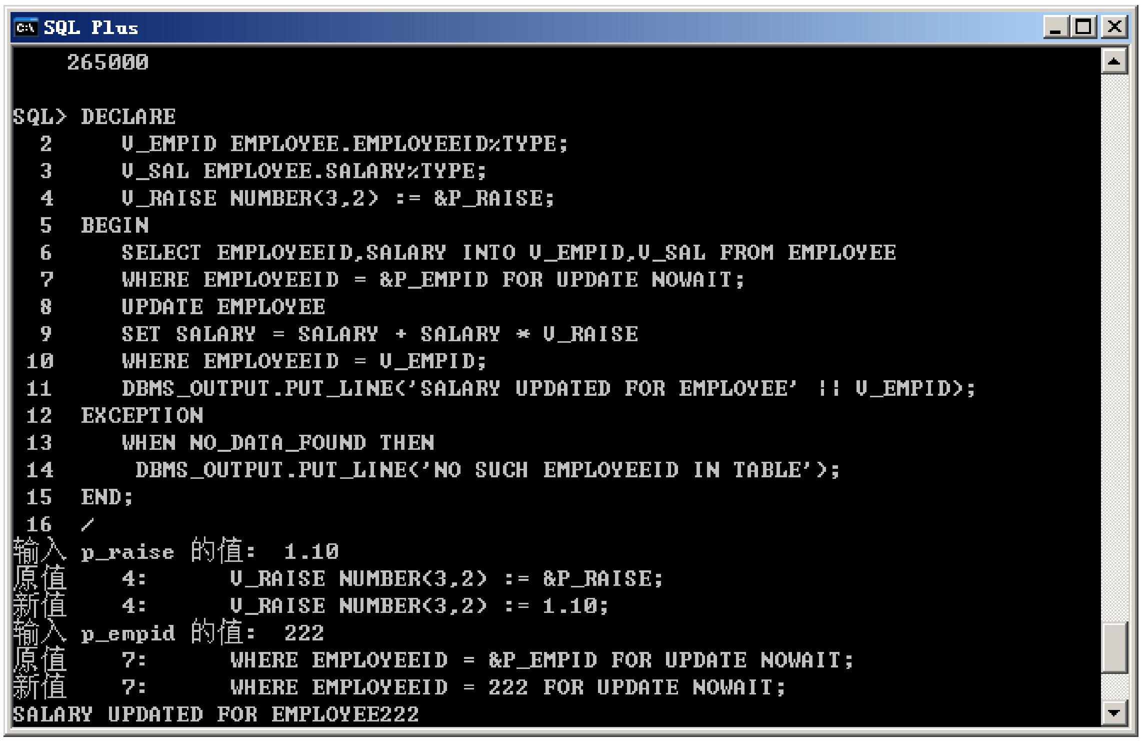
```
  1  DECLARE
  2     CURSOR EMP_CUR(EMP_SAL NUMBER) IS
  3     SELECT LNAME,FNAME,HIREDATE,DEPTNAME
  4     FROM EMPLOYEE E,DEPT D
  5     WHERE SALARY > EMP_SAL AND D.DEPTID=E.DEPTID;
  6     V_FIRST EMPLOYEE.FNAME%TYPE;
  7     V_LAST EMPLOYEE.LNAME%TYPE;
  8     V_HIRE EMPLOYEE.HIREDATE%TYPE;
  9     V_DEPT DEPT.DEPTNAME%TYPE;
 10     v_sal EMPLOYEE.SALARY%TYPE := '&P_SAL';
 11   BEGIN
 12     OPEN EMP_CUR(v_sal);
 13     FETCH EMP_CUR INTO V_LAST,V_FIRST,V_HIRE,V_DEPT;
 14     WHILE EMP_CUR%FOUND LOOP
 15      DBMS_OUTPUT.PUT_LINE(V_LAST||','||V_FIRST);
 16      DBMS_OUTPUT.PUT_LINE('was hired on' || to_char(V_HIRE,'MM/DD/YYYY'));
 17      DBMS_OUTPUT.PUT_LINE('in' || V_DEPT||'department');
 18     FETCH EMP_CUR INTO V_LAST,V_FIRST,V_HIRE,V_DEPT;
 19      END LOOP;
 20      DBMS_OUTPUT.PUT_LINE('TOTAL EMPLOYEES:' || EMP_CUR%ROWCOUNT);
 21     CLOSE EMP_CUR;
 22*  END;
SQL> /
输入 p_sal 的值:  80000
```

3.20.2 THE CURSOR WITH FOR LOOP AND WHERE CURRENT OF



```
已写入 file afiedt.buf

  1  DECLARE
  2     CURSOR SAL_CUR IS
  3     SELECT LNAME,SALARY
  4     FROM EMPLOYEE
  5     WHERE POSITIONID =
  6     (SELECT POSITIONID FROM POSITION WHERE UPPER(POSDESC)='&POSITION')
  7     FOR UPDATE;
  8  BEGIN
  9     FOR SAL_REC IN SAL_CUR LOOP
 10      UPDATE EMPLOYEE SET SALARY = SALARY *1.05
 11      WHERE CURRENT OF SAL_CUR;
 12    END LOOP;
 13    COMMIT;
 14* END;
SQL> /
输入 position 的值:  1
原值    6:      (SELECT POSITIONID FROM POSITION WHERE UPPER(POSDESC)='&POSITION
')
新值    6:      (SELECT POSITIONID FROM POSITION WHERE UPPER(POSDESC)='1')

PL/SQL 过程已成功完成。

SQL> select * from employee;
```

3.20.3 THE CURSOR WITH EXCEPTION HANDLE



```
SQL Plus
    265000

SQL> DECLARE
  2     V_EMPID EMPLOYEE.EMPLOYEEID%TYPE;
  3     V_SAL EMPLOYEE.SALARY%TYPE;
  4     V_RAISE NUMBER(3,2) := &P_RAISE;
  5  BEGIN
  6     SELECT EMPLOYEEID,SALARY INTO V_EMPID,V_SAL FROM EMPLOYEE
  7     WHERE EMPLOYEEID = &P_EMPID FOR UPDATE NOWAIT;
  8     UPDATE EMPLOYEE
  9     SET SALARY = SALARY + SALARY * V_RAISE
 10     WHERE EMPLOYEEID = V_EMPID;
 11     DBMS_OUTPUT.PUT_LINE('SALARY UPDATED FOR EMPLOYEE' || V_EMPID);
 12  EXCEPTION
 13     WHEN NO_DATA_FOUND THEN
 14      DBMS_OUTPUT.PUT_LINE('NO SUCH EMPLOYEEID IN TABLE');
 15  END;
 16  /
输入 p_raise 的值:  1.10
原值    4:      V_RAISE NUMBER(3,2) := &P_RAISE;
新值    4:      V_RAISE NUMBER(3,2) := 1.10;
输入 p_empid 的值:  222
原值    7:      WHERE EMPLOYEEID = &P_EMPID FOR UPDATE NOWAIT;
新值    7:      WHERE EMPLOYEEID = 222 FOR UPDATE NOWAIT;
SALARY UPDATED FOR EMPLOYEE222
```

DBMaker

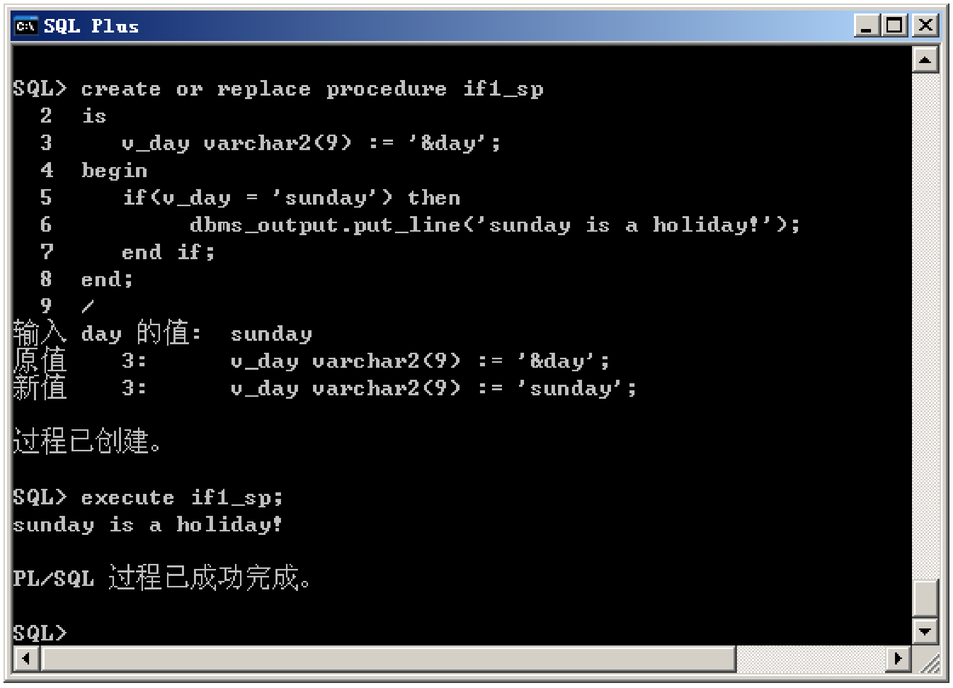
DBMaker doesn’ t have anonymous blocks

4. The example for Create Stored Procedure

4.1 IF

4.1.1 IF …THEN…END IF

Oracle



```
SQL> create or replace procedure if1_sp
  2  is
  3     v_day varchar2(9) := '&day';
  4  begin
  5     if(v_day = 'sunday') then
  6          dbms_output.put_line('sunday is a holiday!');
  7     end if;
  8  end;
  9  /
输入 day 的值:  sunday
原值    3:      v_day varchar2(9) := '&day';
新值    3:      v_day varchar2(9) := 'sunday';

过程已创建。

SQL> execute if1_sp;
sunday is a holiday!

PL/SQL 过程已成功完成。

SQL>
```

DBMaker

Sp code:

```
CREATE PROCEDURE if1_sp (IN CD VARCHAR (9), OUT VAL1 VARCHAR (30))
LANGUAGE SQL
BEGIN
    IF (CD = 'sunday') then
       SET VAL1 = 'sunday is a holiday!';
    END IF;
END;
```

Create and execute result:

```
DmSQL> CREATE PROCEDURE FROM 'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\IF1.sp';
dmSQL> CALL IF1_SP ('sunday',?);
```

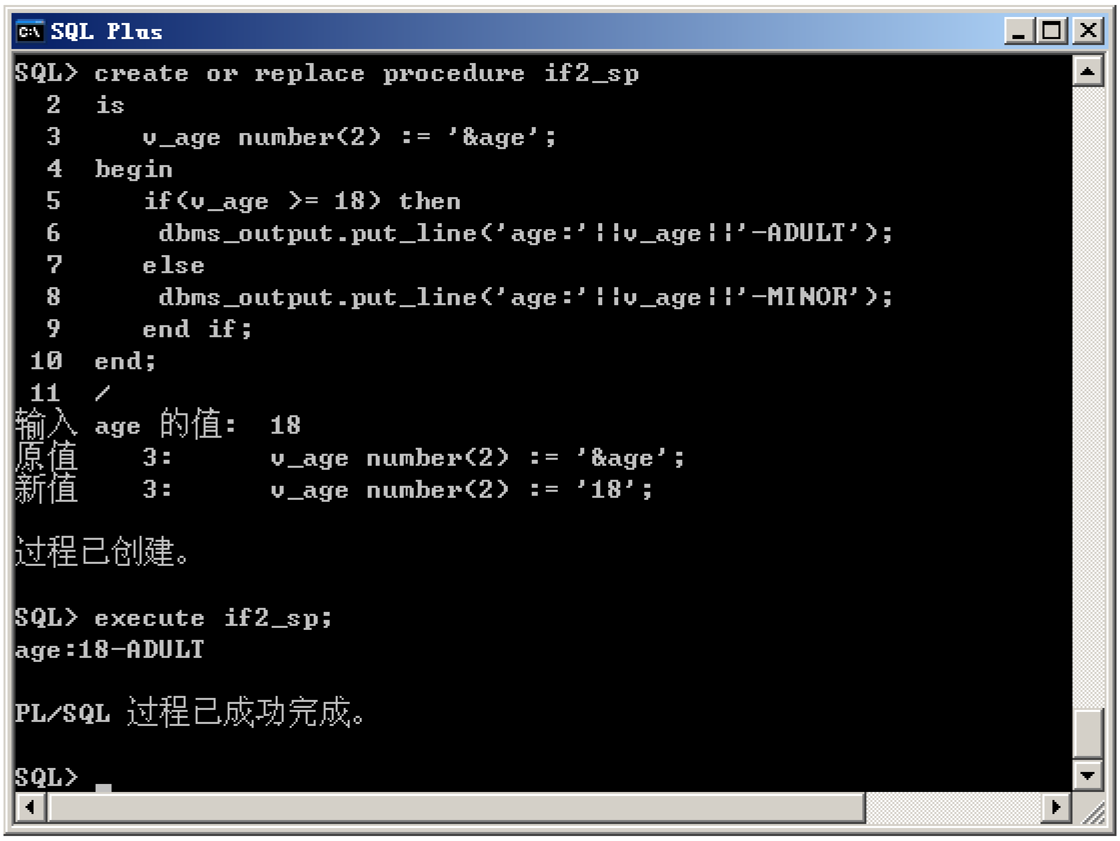
```
VAL1                                    : Sunday is a holiday!
```

Compare:
1) Oracle can use “&var_name” to replace a dynamic input parameter.

In DBMaker, SQLSP use an input parameter to equivalent.
2) Oracle has DBMS_OUTPU print statement, DBMaker use output parameter to equivalent.

4.1.2 IF…THEN…ELSE…END IF

Oracle



```
SQL> create or replace procedure if2_sp
  2  is
  3     v_age number(2) := '&age';
  4  begin
  5     if(v_age >= 18) then
  6      dbms_output.put_line('age:'||v_age||'-ADULT');
  7     else
  8      dbms_output.put_line('age:'||v_age||'-MINOR');
  9     end if;
 10  end;
 11  /
输入 age 的值:  18
原值    3:      v_age number(2) := '&age';
新值    3:      v_age number(2) := '18';

过程已创建。

SQL> execute if2_sp;
age:18-ADULT

PL/SQL 过程已成功完成。

SQL> _
```

DBMaker

Sp code

```
CREATE PROCEDURE IF2_SP (IN AGE INT, OUT STR CHAR (128))
LANGUAGE SQL
BEGIN
    DECLARE V_AGE INT;
    DECLARE V_STR CHAR (128);
    SET V_AGE = AGE;
    IF (V_AGE > 18) THEN
           SET V_STR = 'AGE HAD BEEN -ADULT';
    ELSE
           SET V_STR = 'AGE HAD BEEN -MINOR';
    END IF;
    SET STR = V_STR;
END;

创建 sp 及执行结果：
```

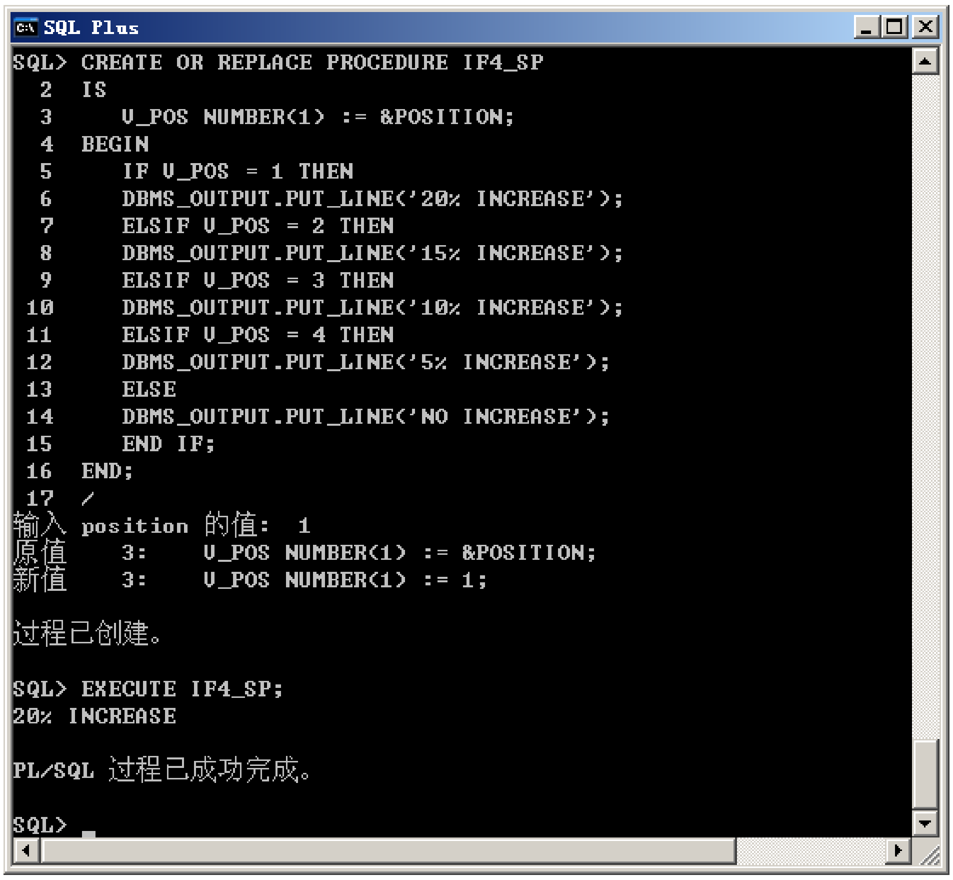
```
dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\if2.sp';

dmSQL> CALL IF2_SP (25,?);
STR: AGE HAD BEEN -ADULT

dmSQL> CALL IF2_SP (15,?);
STR: AGE HAD BEEN –MINOR
```

Compare:

Same with 3.2.1

4.1.3 IF…THEN…ELSIF…END IF

Oracle



```
SQL> CREATE OR REPLACE PROCEDURE IF4_SP
  2  IS
  3     V_POS NUMBER(1) := &POSITION;
  4  BEGIN
  5     IF V_POS = 1 THEN
  6     DBMS_OUTPUT.PUT_LINE('20% INCREASE');
  7     ELSIF V_POS = 2 THEN
  8     DBMS_OUTPUT.PUT_LINE('15% INCREASE');
  9     ELSIF V_POS = 3 THEN
 10     DBMS_OUTPUT.PUT_LINE('10% INCREASE');
 11     ELSIF V_POS = 4 THEN
 12     DBMS_OUTPUT.PUT_LINE('5% INCREASE');
 13     ELSE
 14     DBMS_OUTPUT.PUT_LINE('NO INCREASE');
 15     END IF;
 16  END;
 17  /
输入 position 的值:  1
原值    3:     V_POS NUMBER(1) := &POSITION;
新值    3:     V_POS NUMBER(1) := 1;

过程已创建。

SQL> EXECUTE IF4_SP;
20% INCREASE

PL/SQL 过程已成功完成。

SQL> _
```

DBMaker

Sp code

```
CREATE PROCEDURE IF4_SP (IN POSITION DECIMAL (1, 0), OUT SITUATION VARCHAR (20))
LANGUAGE SQL
BEGIN
     DECLARE V_POS DECIMAL (1, 0);
     SET V_POS = POSITION;
     IF V_POS = 1 THEN
     SET SITUATION = '20% INCREASE';
     ELSEIF V_POS = 2 THEN
```

```
    SET SITUATION = '15% INCREASE';
    ELSEIF V_POS = 3 THEN
    SET SITUATION = '10% INCREASE';
    ELSEIF V_POS = 4 THEN
    SET SITUATION = '5% INCREASE';
    ELSE
    SET SITUATION = 'NO INCREASE';
    END IF;
END;
```

Create and execute result:

```
dmSQL> create procedure from
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\if4.sp';

dmSQL> CALL IF4_SP (1,?);
SITUATION                      : 20% INCREASE
```

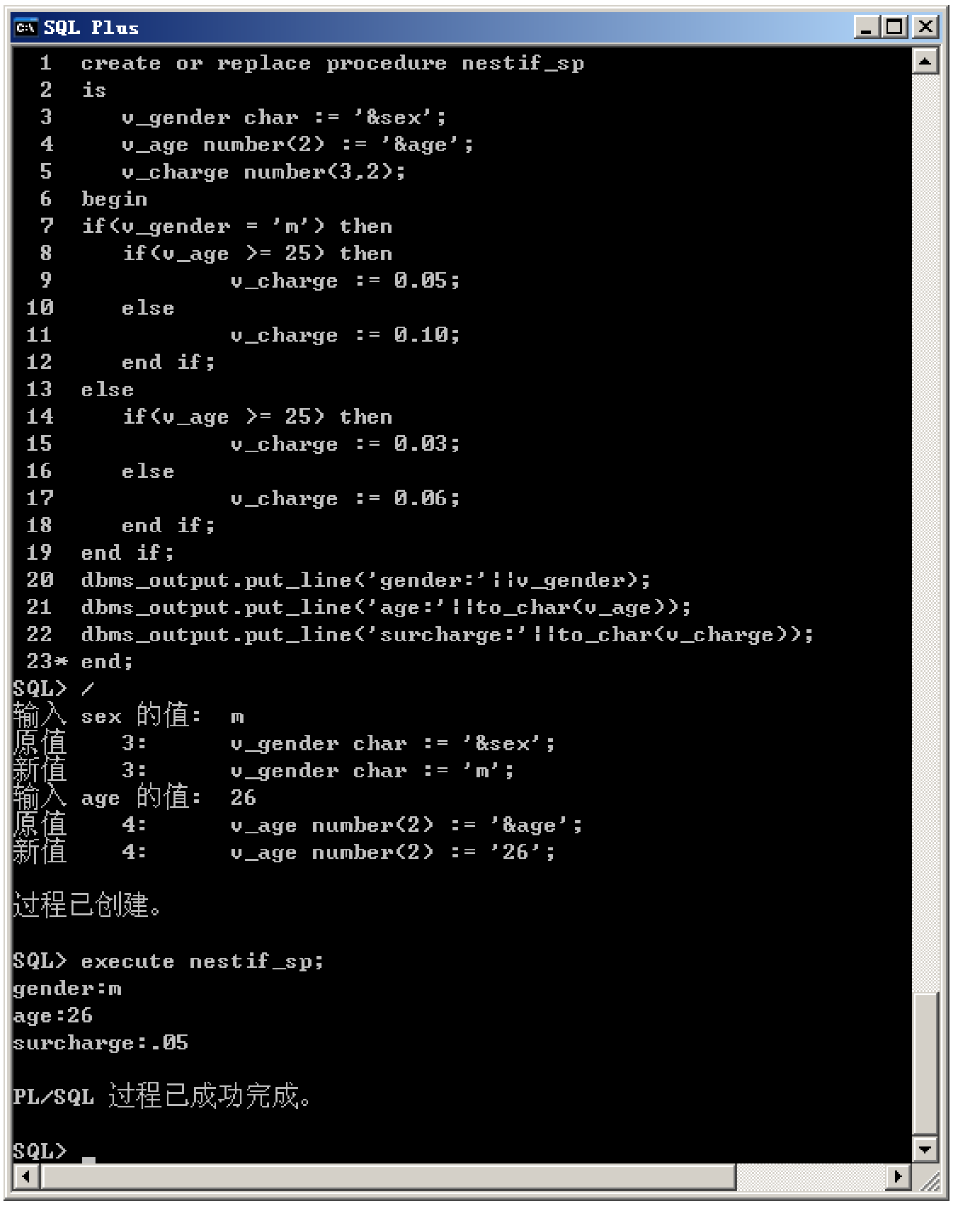
Compare:

Besides 3.1.1, the biggest difference is below:

1) Oracle use ELSIF and DBMaker use ELSEIF.
2) DBMaer use OUT parameter to display execute result because of there’s not output statement in it, so when DBMaker SQL SP call a stored procedure with CALL statement, the output parameter should use “?” to instead and can’t omit.

4.1.4 NESTED IF

Oracle



```
SQL Plus
create or replace procedure nestif_sp
is
   v_gender char := '&sex';
   v_age number(2) := '&age';
   v_charge number(3,2);
begin
if(v_gender = 'm') then
   if(v_age >= 25) then
           v_charge := 0.05;
   else
           v_charge := 0.10;
   end if;
else
   if(v_age >= 25) then
           v_charge := 0.03;
   else
           v_charge := 0.06;
   end if;
end if;
dbms_output.put_line('gender:'||v_gender);
dbms_output.put_line('age:'||to_char(v_age));
dbms_output.put_line('surcharge:'||to_char(v_charge));
end;
SQL> /
输入 sex 的值:  m
原值    3:      v_gender char := '&sex';
新值    3:      v_gender char := 'm';
输入 age 的值:  26
原值    4:      v_age number(2) := '&age';
新值    4:      v_age number(2) := '26';

过程已创建。

SQL> execute nestif_sp;
gender:m
age:26
surcharge:.05

PL/SQL 过程已成功完成。

SQL> _
```

DBMaker

Sp code:

```
CREATE PROCEDURE NESTIF_SP (IN gender CHAR (1), IN age INT, OUT charge DECIMAL (3, 2))
LANGUAGE SQL
BEGIN
    DECLARE V_GENDER CHAR (1);
    DECLARE V_AGE INT;
    DECLARE V_CHARGE DECIMAL (3, 2);
    SET V_GENDER = gender;
    SET V_AGE = age;
    IF V_GENDER = 'M' THEN
           IF (V_AGE >= 25) THEN
                  SET V_CHARGE= 0.05;
           ELSE
                  SET V_CHARGE= 0.10;
           END IF;
    ELSE
```

```
            IF (V_AGE >= 25) THEN
                   SET V_CHARGE= 0.03;
            ELSE
                    SET V_CHARGE= 0.06;
            END IF;
    END IF;
    SET charge = V_CHARGE;

END;
```

Create and execute result:

```
dmSQL> create procedure from
'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\nestedIF.sp';

dmSQL> call nestif_sp ('M',26,?);
CHARGE                          : 0.05
```

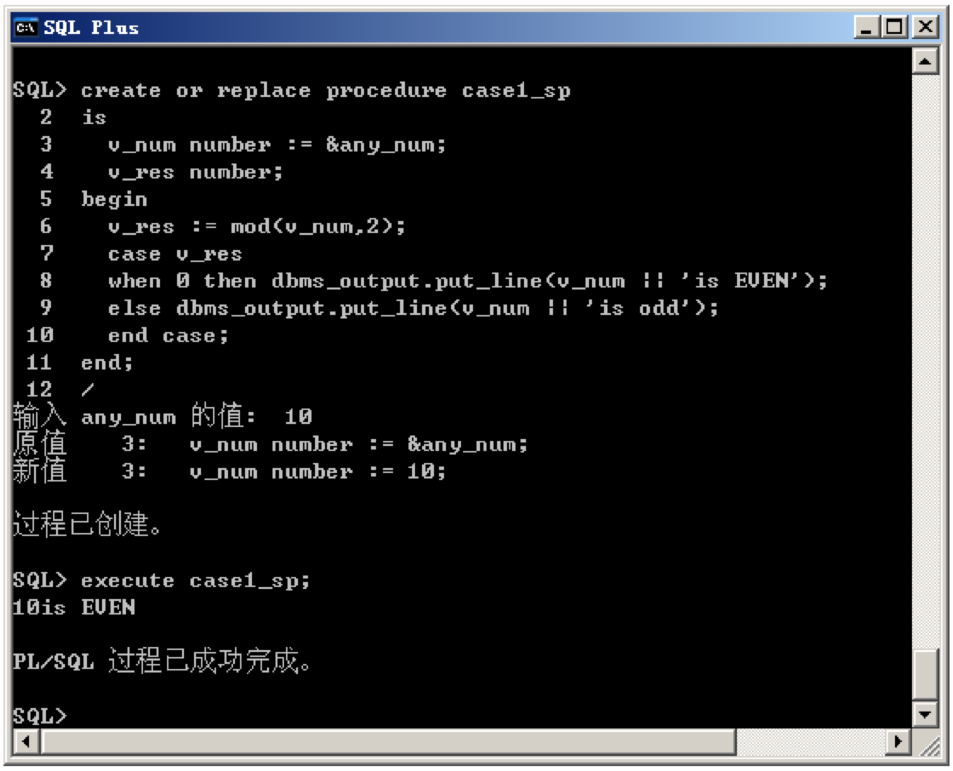
Compare:

1) DBMaker also support IF Nested Loop.
2) For such data types like CHAR, the default size = 1, but if in DBMaker SQL SP, you should specify the value for SIZE, else the error will happen.
3) Assignment symbol, in Oracle, use” : =” to assign value for variable, but in DBMaker, use”=” and add SET keyword in front of the variable.

4.2 CASE

4.2.1 CASE STATEMENT

Oracle



```
SQL> create or replace procedure case1_sp
  2  is
  3    v_num number := &any_num;
  4    v_res number;
  5  begin
  6    v_res := mod(v_num,2);
  7    case v_res
  8    when 0 then dbms_output.put_line(v_num || 'is EVEN');
  9    else dbms_output.put_line(v_num || 'is odd');
 10    end case;
 11  end;
 12  /
输入 any_num 的值:  10
原值    3:   v_num number := &any_num;
新值    3:   v_num number := 10;

过程已创建。

SQL> execute case1_sp;
10is EVEN

PL/SQL 过程已成功完成。

SQL>
```

DBMaker

SP CODE:

```
CREATE PROCEDURE CASE1_SP (IN V_NUM INT, OUT THISNUM VARCHAR (50))
LANGUAGE SQL
BEGIN
    DECLARE V_RES INT;
    SET V_RES = MOD (V_NUM, 2);
    CASE V_RES
    WHEN 0 THEN
            SET THISNUM = 'IS EVEN';
    ELSE
            SET THISNUM = 'IS ODD';
    END CASE;
END;
```

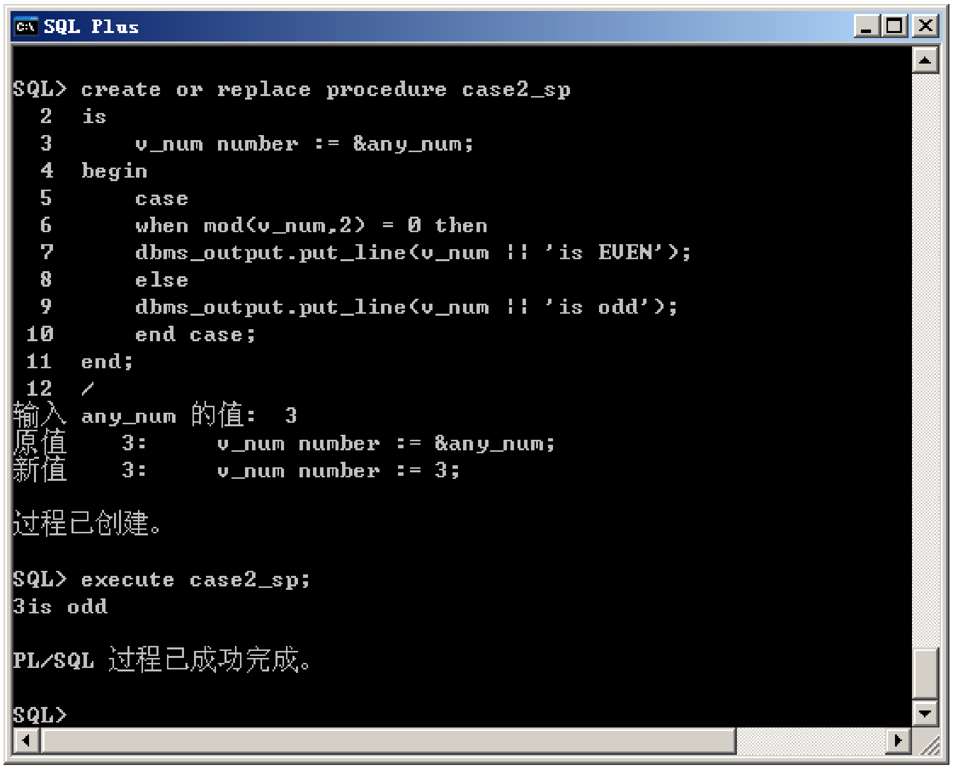
Create and execute result:

```
dmSQL> create procedure from
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\case1.sp';

dmSQL> CALL CASE1_SP (3,?);
THISNUM                         : IS ODD

dmSQL> CALL CASE1_SP (4,?);
THISNUM                         : IS EVEN
```

Compare: ditto

4.2.2 SEARCHING CASE

Oracle



```
SQL> create or replace procedure case2_sp
  2  is
  3      v_num number := &any_num;
  4  begin
  5      case
  6      when mod(v_num,2) = 0 then
  7      dbms_output.put_line(v_num || 'is EVEN');
  8      else
  9      dbms_output.put_line(v_num || 'is odd');
 10      end case;
 11  end;
 12  /
输入 any_num 的值:  3
原值    3:      v_num number := &any_num;
新值    3:      v_num number := 3;

过程已创建。

SQL> execute case2_sp;
3is odd

PL/SQL 过程已成功完成。

SQL>
```

DBMaker

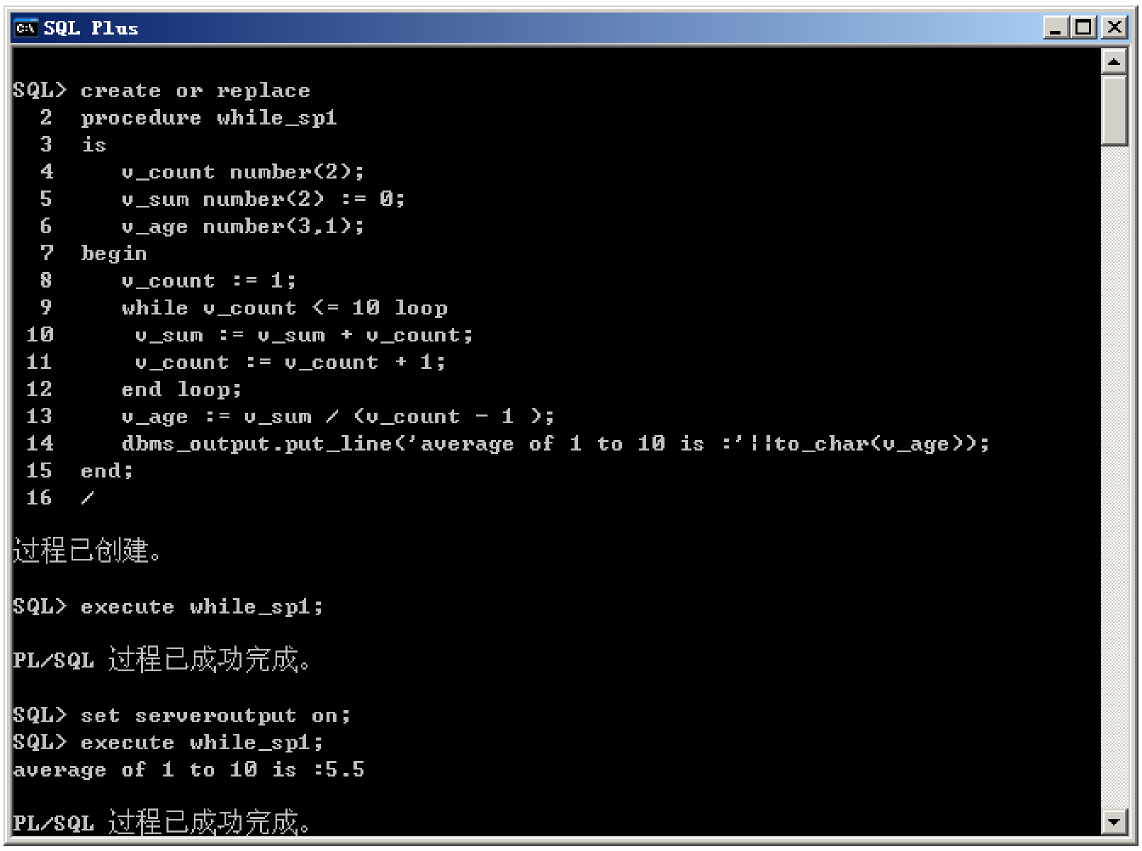
SP CODE:

```
CREATE PROCEDURE CASE2_SP (IN V_NUM INT, OUT THISNUM VARCHAR (50))
LANGUAGE SQL
BEGIN
    DECLARE VAR1 INT;
    SET VAR1 = MOD (V_NUM, 2);
    CASE
    WHEN VAR1 = 0 THEN
            SET THISNUM = 'IS EVEN';
    ELSE
            SET THISNUM = 'IS ODD';
    END CASE;
END;
```

Create and execute result:

```
dmSQL> create procedure from
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\case2.sp';

dmSQL> CALL CASE2_SP (3,?);
THISNUM                          : IS ODD

dmSQL> CALL CASE2_SP (4,?);
THISNUM                          : IS EVEN
```

4.3 WHILE



SQL Plus

```
SQL> create or replace
     procedure while_sp1
     is
        v_count number(2);
        v_sum number(2) := 0;
        v_age number(3,1);
     begin
        v_count := 1;
        while v_count <= 10 loop
         v_sum := v_sum + v_count;
         v_count := v_count + 1;
        end loop;
        v_age := v_sum / (v_count - 1 );
        dbms_output.put_line('average of 1 to 10 is :'||to_char(v_age));
     end;
     /

过程已创建。

SQL> execute while_sp1;

PL/SQL 过程已成功完成。

SQL> set serveroutput on;
SQL> execute while_sp1;
average of 1 to 10 is :5.5

PL/SQL 过程已成功完成。
```

DBMaker

Sp code:

```
CREATE PROCEDURE WHILE_SP1 (OUT average FLOAT)
LANGUAGE SQL
BEGIN
    DECLARE V_COUNT INT;
    DECLARE V_SUM INT DEFAULT 0;
    DECLARE V_AGE FLOAT;
    SET V_COUNT = 1;
    WHILE (V_COUNT <= 10) DO
           SET V_SUM = V_SUM + V_COUNT;
           SET V_COUNT = V_COUNT + 1;
    END WHILE;
    SET V_AGE = V_SUM/ (V_COUNT - 1);
    SET average = V_AGE;
END;
```

Create and execute result:

```
dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\while.sp';

dmSQL> CALL WHILE_SP1 (?);
AVERAGE                          :  5.50000000000000e+000
```

Compare: Others same with above

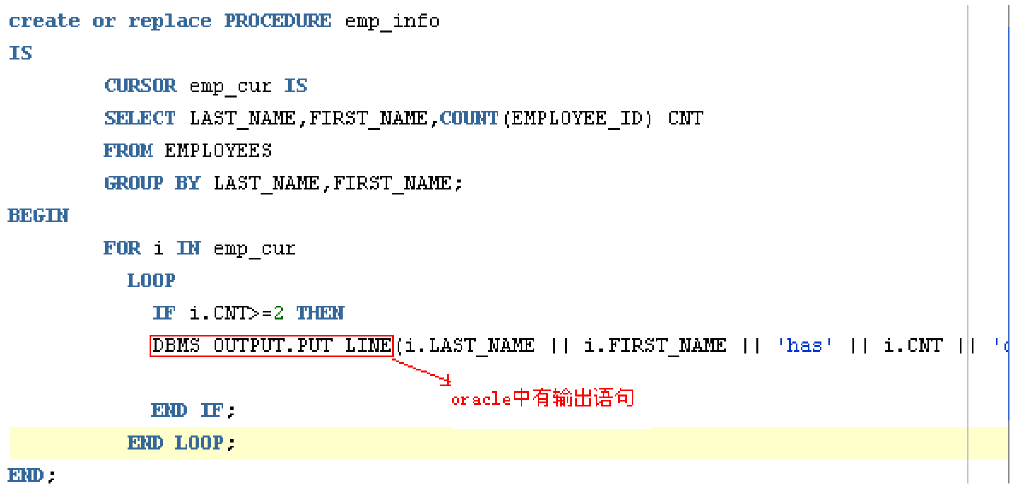
The syntax difference of WHILE:

- Oracle: WHILE……LOOP
- DBMaker: WHILE……DO

4.4 FOR

Output all of the records which accorded with the condition.

Oracle



```
create or replace PROCEDURE emp_info
IS
        CURSOR emp_cur IS
        SELECT LAST_NAME,FIRST_NAME,COUNT(EMPLOYEE_ID) CNT
        FROM EMPLOYEES
        GROUP BY LAST_NAME,FIRST_NAME;
BEGIN
        FOR i IN emp_cur
          LOOP
            IF i.CNT>=2 THEN
            DBMS_OUTPUT.PUT_LINE(i.LAST_NAME || i.FIRST_NAME || 'has' || i.CNT || '

            END IF;
          END LOOP;
END;
```

DBMaker

If support above function in DBMaker that need to using different syntax, using for statement can’t reach to the purpose, below syntax just can output the last group qualified records value, because DBMaker doesn’t support output statement like in oracle.

```
CREATE PROCEDURE emp_info (OUTPUT VAL1 VARCHAR (10), OUTPUT VAL2 VARCHAR (10))
LANGUAGE SQL
BEGIN
    DECLARE TMP_VAL1 VARCHAR (11);
    DECLARE TMP_VAL2 VARCHAR (11);
    DECLARE emp_cur CURSOR FOR
    SELECT FIRST_NAME, LAST_NAME, COUNT (EMPLOYEE_ID) CNS FROM EMPLOYEES GROUP BY
FIRST_NAME, LAST_NAME;
        SET NUM = 0;
    FOR emp_cur
        DO
            IF emp_cur.CNS =2 THEN
                    IF NUM = 0 THEN
                            SET TMP _VAL1 = emp_cur.FIRST_NAME;
                            SET TMP _VAL2 = emp_cur.LAST_NAME;
                            SET VAL1 = emp_cur.FIRST_NAME;
                            SET VAL2 = emp_cur.LAST_NAME;
                    END IF;
    END FOR;
END;
```

e.g1: To solve the problem that just can output the last group value, we can continue to use the loop syntax by multi-define output variables, but we need to know how many value need to output at first and the output format isn't we wanted.

```
CREATE PROCEDURE emp_info(OUTPUT VAL1 VARCHAR(10),OUTPUT VAL2 VARCHAR(10) ,OUTPUT VAL3
VARCHAR(10),OUTPUT VAL4 VARCHAR(10),OUTPUT VAL5 VARCHAR(10),OUTPUT VAL6 VARCHAR(10)  )
LANGUAGE SQL
BEGIN
    DECLARE TMP_VAL1 VARCHAR (11);
    DECLARE TMP_VAL2 VARCHAR (11);
    DECLARE NUM INT;
    DECLARE emp_cur CURSOR FOR
    SELECT FIRST_NAME, LAST_NAME, COUNT (EMPLOYEE_ID) CNS FROM EMPLOYEES GROUP BY
FIRST_NAME, LAST_NAME;
        SET NUM = 0;
    FOR emp_cur
        DO
            IF emp_cur.CNS = 1 THEN
                    IF NUM = 0 THEN
                            SET VAL1 = emp_cur.FIRST_NAME;
                            SET VAL2 = emp_cur.LAST_NAME;
                            SET NUM = 2;
                    ELSEIF NUM = 2 THEN
                            SET VAL3 = emp_cur.FIRST_NAME;
                            SET VAL4 = emp_cur.LAST_NAME;
                            SET NUM = 4;
                    ELSEIF NUM = 4 THEN
                            SET VAL5 = emp_cur.FIRST_NAME;
                            SET VAL6 = emp_cur.LAST_NAME;
                            SET NUM = 6;
                    END IF;
            END IF;
    END FOR;
END;
```

e.g2: so we can use the specific "with return" statement in our SP to output all of the qualified records, and condition statement control by defines cursor.

```
CREATE PROCEDURE emp_TEST
RETURNS VARCHAR (10) V1, VARCHAR (10) V2
LANGUAGE SQL

BEGIN
    DECLARE TMP_VAL1 VARCHAR (11);
    DECLARE TMP_VAL2 VARCHAR (11);
    DECLARE emp_cur CURSOR WITH RETURN FOR
        SELECT FIRST_NAME, LAST_NAME, COUNT (EMPLOYEE_ID) CNS FROM EMPLOYEES GROUP BY
FIRST_NAME, LAST_NAME HAVING COUNT(EMPLOYEE_ID) = 1;
    OPEN emp_cur;
END;
```

4.5 About Cursor

4.5.1 Explicit Cursor

Oracle

Create procedure:

```
create or replace procedure cur_sp
is
    cursor emp_cur is
    select last_name, first_name, salary
    from employees where salary is not null;
    v_first employees.first_name%type;
    v_last employees.last_name%type;
    v_sal employees.salary%type;
    v_sum number (10):= 0;
begin
    open emp_cur;
    fetch emp_cur into v_last, v_first, v_sal;
    while emp_cur%found
    loop
      v_sum:= v_sum + v_sal;
      dbms_output.put_line (v_last||','||v_first);
      DBMS_OUTPUT.PUT_LINE ('makes'|| TO_CHAR (V_SUM,'$999,999'));
      fetch emp_cur into v_last, v_first, v_sal;
    end loop;
end;
```

Execute and the result:

```
SQL> @F:\snow\SQLSP\PL_ORACLE\sp\script\cur.sp;
 21 /

过程已创建。

SQL> execute cur_sp;
Smith, James
makes $800,000
Johnson, Ron
makes#########
Hobbs, Fred
makes#########
Jones, Susan
makes#########
Jones, Susan
makes#########
love, snow
makes#########

PL/SQL。
```

DBMaker

SP CODE:

```
CREATE PROCEDURE CUR_SP
RETURNS VARCHAR (30) EMP_NAME, DECIMAL (6, 0) RETURN_TOTAL
LANGUAGE SQL
BEGIN
    DECLARE v_last, v_first VARCHAR (15);
    DECLARE v_salary DECIMAL (6, 0);
    DECLARE v_commission DECIMAL (5, 0);
    DECLARE emp_cur CURSOR WITH RETURN FOR
    SELECT LNAME||','||FNAME as 'EMP_NAME', (SALARY+COMMISSION)
    FROM EMPLOYEE WHERE SALARY IS NOT NULL;
    OPEN emp_cur;
END;
```

Compare:

DBMaker doesn’ t support DBMS_OUTPUT output statement, so we need to use SQL statement to output the records which meet to the conditions.

Because the output format is different, we couldn’t control the output format is we wanted, so only list column-name /data format like below:

Create and execute result:

```
dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\cur.sp';

dmSQL> CALL CUR_SP;

           EMP_NAME                RETURN_TOTAL
============================== ====================
Smith, john                        300000
Houston, Larry                     160000
Roberts, Sandi                     NULL
Dev, Derek                         100000
Stanley, john                      80000
Chen, sunny                        NULL
6 rows selected
```

4.5.2 THE CURSOR WITH PARAMETER

Oracle

```
CREATE OR REPLACE PROCEDURE PARA_CUR
IS
    CURSOR EMP_CUR (EMP_SAL NUMBER) IS
    SELECT LNAME, FNAME, HIREDATE, DEPTNAME
    FROM EMPLOYEE E, DEPT D
    WHERE SALARY > EMP_SAL AND D.DEPTID=E.DEPTID;
    V_FIRST EMPLOYEE.FNAME%TYPE;
    V_LAST EMPLOYEE.LNAME%TYPE;
```

```
     V_HIRE EMPLOYEE.HIREDATE%TYPE;
     V_DEPT DEPT.DEPTNAME%TYPE;
     v_sal EMPLOYEE.SALARY%TYPE: = '&P_SAL';
   BEGIN
     OPEN EMP_CUR (v_sal);
     FETCH EMP_CUR INTO V_LAST, V_FIRST, V_HIRE, AND V_DEPT;
     WHILE EMP_CUR%FOUND LOOP
      DBMS_OUTPUT.PUT_LINE (V_LAST||','||V_FIRST);
      DBMS_OUTPUT.PUT_LINE ('was hired on' || to_char (V_HIRE,'MM/DD/YYYY'));
      DBMS_OUTPUT.PUT_LINE ('in' || V_DEPT||'department');
     FETCH EMP_CUR INTO V_LAST, V_FIRST, V_HIRE, V_DEPT;
       END LOOP;
      DBMS_OUTPUT.PUT_LINE ('TOTAL EMPLOYEES:' || EMP_CUR%ROWCOUNT);
     CLOSE EMP_CUR;
END;
```

Execute result:

```
SQL> @F:\snow\SQLSP\PL_ORACLE\sp\script\parameter_cur.sp;
 24 /
輸入 p_sal 的值:  8000
原值   11:      v_sal EMPLOYEE.SALARY%TYPE: = '&P_SAL';
新值   11:      v_sal EMPLOYEE.SALARY%TYPE: = '8000';

过程已创建。
SQL> EXECUTE PARA_CUR;
Dev, Derek
was hired on03/15/0095
inInfoSysdepartment
Garner, Stanley
was hired on02/29/0096
inSalesdepartment
Smith, John
was hired on04/15/0060
inFinancedepartment
Roberts, Sandi
was hired on12/02/0091
inFinancedepartment
McCall, Alex
was hired on05/10/0097
inInfoSysdepartment
Shaw, Jinku
was hired on01/03/0010
inSalesdepartment
Chen, Sunny
was hired on08/15/0099
inFinancedepartment
TOTAL EMPLOYEES: 7
PL/SQL。
```

DBMaker

Sp code:

```
CREATE PROCEDURE PARA_CUR (IN sal FLOAT)
RETURNS VARCHAR (15) L_NAME, VARCHAR (15) F_NAME, DATE MYHIREDATE, VARCHAR (12) MYDEPTNAME
LANGUAGE SQL
BEGIN
     DECLARE str VARCHAR (1000);
     DECLARE cur CURSOR WITH RETURN FOR stmt;
     SET str =
         'SELECT LNAME, FNAME, HIREDATE, DEPTNAME
      FROM EMPLOYEE E, DEPT D
      WHERE SALARY > ' sal—Dynamic input parameter value
      'AND D.DEPTID=E.DEPTID';

     PREPARE stmt FROM str;
     OPEN cur USING sal;
END;
```

NOTE *Since DBMaker doesn' t handle well for dynamic input parameter on SQLSP at current version, so can't execute success temporary.*

4.5.3 CURSOR FOR LOOP AND WHERE CURRENT OF

Oracle

```
CREATE OR REPLACE PROCEDURE FOR_CURRENT_SP
IS
     CURSOR SAL_CUR IS
     SELECT LNAME, SALARY
     FROM EMPLOYEE
     WHERE POSITIONID =
     (SELECT POSITIONID FROM POSITION WHERE UPPER (POSDESC) ='&POSITION')
     FOR UPDATE;
BEGIN
     FOR SAL_REC IN SAL_CUR LOOP
      UPDATE EMPLOYEE SET SALARY = SALARY *1.10
      WHERE CURRENT OF SAL_CUR;
     END LOOP;
   COMMIT;
 END;
```

DBMaker

```
DBMaker doesn't support such clause.
```

4.6 Exception

4.6.1 HANDLE THE NAMED EXCEPTION

Oracle

```
CREATE OR REPLACE PROCEDURE NAMED_EXCEPTION
IS
    V_FIRST EMPLOYEE.FNAME%TYPE;
    V_LAST EMPLOYEE.LNAME%TYPE;
    D_ID NUMBER (2):= &DEPARTMENT_ID;
BEGIN
    SELECT LNAME, FNAME
    INTO V_LAST, V_FIRST
    FROM EMPLOYEE
    WHERE DEPTID = D_ID;
    DBMS_OUTPUT.PUT_LINE (' ');
    DBMS_OUTPUT.PUT_LINE (V_LAST||','||V_FIRST);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE ('NO SUCH DEPARTMENT WITH EMPLOYEES ');
    WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE ('MORE THAN ONE EMPLOYEE IN DEPT:'||D_ID);
END;
```

Execute process and Display the result:

```
輸入 department_id 的值:  20
原値 5: D_ID NUMBER (2):= &DEPARTMENT_ID;
新値 5: D_ID NUMBER (2):= 20;

过程已创建。

SQL> EXECUTE NAMED_EXCEPTION
MORE THAN ONE EMPLOYEE IN DEPT:20

PL/SQL 过程已成功完成。

SQL> /
輸入 department_id 的值:  4
原値 5: D_ID NUMBER (2):= &DEPARTMENT_ID;
新値 5: D_ID NUMBER (2):= 4;

过程已创建。

SQL> EXECUTE NAMED_EXCEPTION
Houston, Larry

PL/SQL 过程已成功完成。

SQL> /
輸入 department_id 的值:  50
原値 5: D_ID NUMBER (2):= &DEPARTMENT_ID;
新値 5: D_ID NUMBER (2):= 50;
```

```
过程已创建。

SQL> EXECUTE NAMED_EXCEPTION
NO SUCH DEPARTMENT WITH EMPLOYEES

PL/SQL 过程已成功完成。
```

DBMaker

Sp code:

```
CREATE PROCEDURE EXP1 (IN D_ID DECIMAL (2, 0))
RETURNS VARCHAR (15) OUTPUT_LNAME, VARCHAR (15) OUTPUT_FNAME, VARCHAR (100) ret_code
LANGUAGE SQL
BEGIN
    DECLARE ERR1 VARCHAR (100);
    DECLARE str CHAR (128);
    DECLARE ex_cur1 CURSOR WITH RETURN FOR stmt;
    SET str = 'SELECT LNAME, FNAME FROM EMPLOYEE WHERE DEPTID = ?';
    PREPARE stmt FROM str;
    OPEN ex_cur1 USING D_ID;

    IF 'NOT FOUND' THEN
            SET ERR1 = 'NO SUCH DEPARTMENT WITH EMPLOYEES';
    END IF;
END;
```

Conclusion:

```
Error info：
ERROR (6590): [DBMaker] source compiling error: error message in: EXP1SYSADM.msg
```

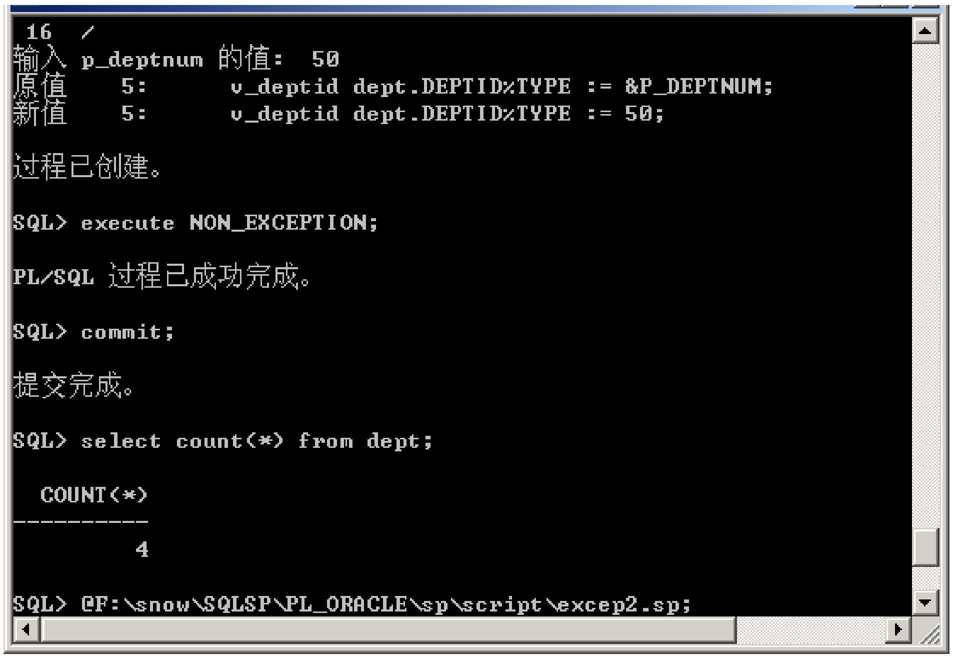
4.6.2 The Predefined Exception in Oracle

Oracle

Create:

```
CREATE OR REPLACE PROCEDURE NON_EXCEPTION
IS
    emp_remain EXCEPTION;
    PRAGMA EXCEPTION_INIT (emp_remain,-2292);
    v_deptid dept.DEPTID%TYPE: = &P_DEPTNUM;
BEGIN
    DELETE FROM DEPT
    WHERE DEPTID = v_deptid;
    COMMIT;
EXCEPTION
    WHEN emp_remain THEN
    DBMS_OUTPUT.PUT_LINE ('DEPARTMENT: '||TO_CHAR (v_deptid));
    DBMS_OUTPUT.PUT_LINE (‘can not be removed. ');
    DBMS_OUTPUT.PUT_LINE ('Employee in department ');
END;
```

Execute result:



```
 16  /
输入 p_deptnum 的值:  50
原值    5:       v_deptid dept.DEPTID%TYPE := &P_DEPTNUM;
新值    5:       v_deptid dept.DEPTID%TYPE := 50;

过程已创建。

SQL> execute NON_EXCEPTION;

PL/SQL 过程已成功完成。

SQL> commit;

提交完成。

SQL> select count(*) from dept;

  COUNT(*)
----------
         4

SQL> @F:\snow\SQLSP\PL_ORACLE\sp\script\excep2.sp;
```

DBMaker

Sp code:

```
CREATE PROCEDURE condition_sp (num int, OUT val INT)
LANGUAGE SQL
BEGIN
         DECLARE con CONDITION FOR SQLSTATE '23000';
         DECLARE CONTINUE HANDLER FOR con SET val = 1;
#       DECLARE EXIT HANDLE FOR con SET val = 1;

         SET val = 0;
         insert into t1 values (num);
         SET val = 2;
END;
```

Create:

```
dmSQL> create procedure from
'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\exp_continue.sp';

dmSQL> create procedure from
'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp1\exp_exit.sp';
```

Execute script;

```
Create table t1 (c1 int);
Create unique index idx on t1 (c1);
Create procedure ‘condition_sp.sp’;
Call condition_sp (1,?);
Call condition_sp (1, ?);
```

Execute result:

```
dmSQL> call condition_sp(1,?);
VAL: 2
```

```
dmSQL> call condition_sp (1,?);
VAL: 2

dmSQL> call condition_sp1 (1,?);
ERROR (401): [DBMaker] unique key violation: INDEX ID 0.340.9
```

Result analysis:

At first, we create sp condition_sp and condition_sp1 by exit|continue then CALL procedure, we will find the condition_sp1which declared EXIT handle will throw exception and exit when input repeated parameters, but condition_sp which defined CONTIONUE handle ,although it has unique constrains but can continue to execute sp and input VAL last result.

SQLCODE and SQLERRM

Oracle

```
CREATE OR REPLACE PROCEDURE CODE_ERRM
IS
    V_FIRST EMPLOYEE.FNAME%TYPE;
    V_LAST EMPLOYEE.LNAME%TYPE;
    D_ID NUMBER (2):=&DEPARTMENT_ID;
    V_CODE NUMBER;
    V_MSG VARCHAR2 (255);
BEGIN
    SELECT LNAME, FNAME
    INTO V_LAST, V_FIRST
    FROM EMPLOYEE
    WHERE DEPTID = D_ID;
    DBMS_OUTPUT.PUT_LINE (' ');
    DBMS_OUTPUT.PUT_LINE (V_LAST||','||V_FIRST);
EXCEPTION
    WHEN OTHERS THEN
    V_CODE:= SQLCODE;
    V_MSG:= SQLERRM;
    DBMS_OUTPUT.PUT_LINE ('ERROR CODE: '||SQLCODE);
    DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;
```

Execute Result:

```
department_id 的值:  10
    5:      D_ID NUMBER (2):=&DEPARTMENT_ID;
    5:      D_ID NUMBER (2):=10;

过程已创建。

SQL> EXECUTE CODE_ERRM;
ERROR CODE: -1422
ORA-01422: 实际返回的行数超出请求的行数
PL/SQL。
```

DBMaker

Sp code:

```
CREATE PROCEDURE EXP3 (OUT SUM INT, OUT STA_EXCUTE VARCHAR (10), OUT CODE_EXCUTE INT)
LANGUAGE SQL
BEGIN
    DECLARE P_SUM INTEGER;
    DECLARE P_SAL INTEGER;
    DECLARE CUR CURSOR FOR SELECT c1 FROM t1;
    SET P_SUM = 0;
    OPEN CUR;
    FETCH FROM CUR INTO P_SAL;
            WHILE (P_SAL! = NULL) DO
                    SET P_SUM = P_SUM + P_SAL;
                    FETCH NEXT FROM CUR INTO P_SAL;
            END WHILE;
    CLOSE CUR;
    SET SUM = P_SUM;
    SET STA_EXCUTE = SQLSTATE;
    SET CODE_EXCUTE = SQLCODE;
END;
```

Create and execute result:

```
dmSQL> CREATE PROCEDURE FROM
'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\exp3.sp';
ERROR (6589): [DBMaker] preprocessor translating error: error message in : EXP3SYSADM.msg

dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\exp3.sp';

dmSQL> CALL EXP3 (?,?,?);
SUM                                :          15
STA_EXCUTE                         : 00000
CODE_EXCUTE                        :         100
```

4.6.3 USER DEFINED EXCEPTION

Oracle

```
CREATE OR REPLACE PROCEDURE USR_EXCEPTION
IS
    invalid_commission EXCEPTION;
    no_commission EXCEPTION;
    v_comm employee.commission%type;
BEGIN
    SELECT COMMISSION
    INTO v_comm
    FROM EMPLOYEE
    WHERE EMPLOYEEID = &emp_id;
    IF v_comm < 0 THEN
```

```
      RAISE invalid_commission;
     ELSIF v_comm IS NULL THEN
      RAISE no_commission;
     ELSE
      DBMS_OUTPUT.PUT_LINE (TO_CHAR (v_comm));
     END IF;
EXCEPTION
     WHEN invalid_commission THEN
     DBMS_OUTPUT.PUT_LINE ('Commission is negative);
     WHEN no_commission THEN
     DBMS_OUTPUT.PUT_LINE ('No commission value');
     WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE ('No such ID');
END;
```

DBMaker

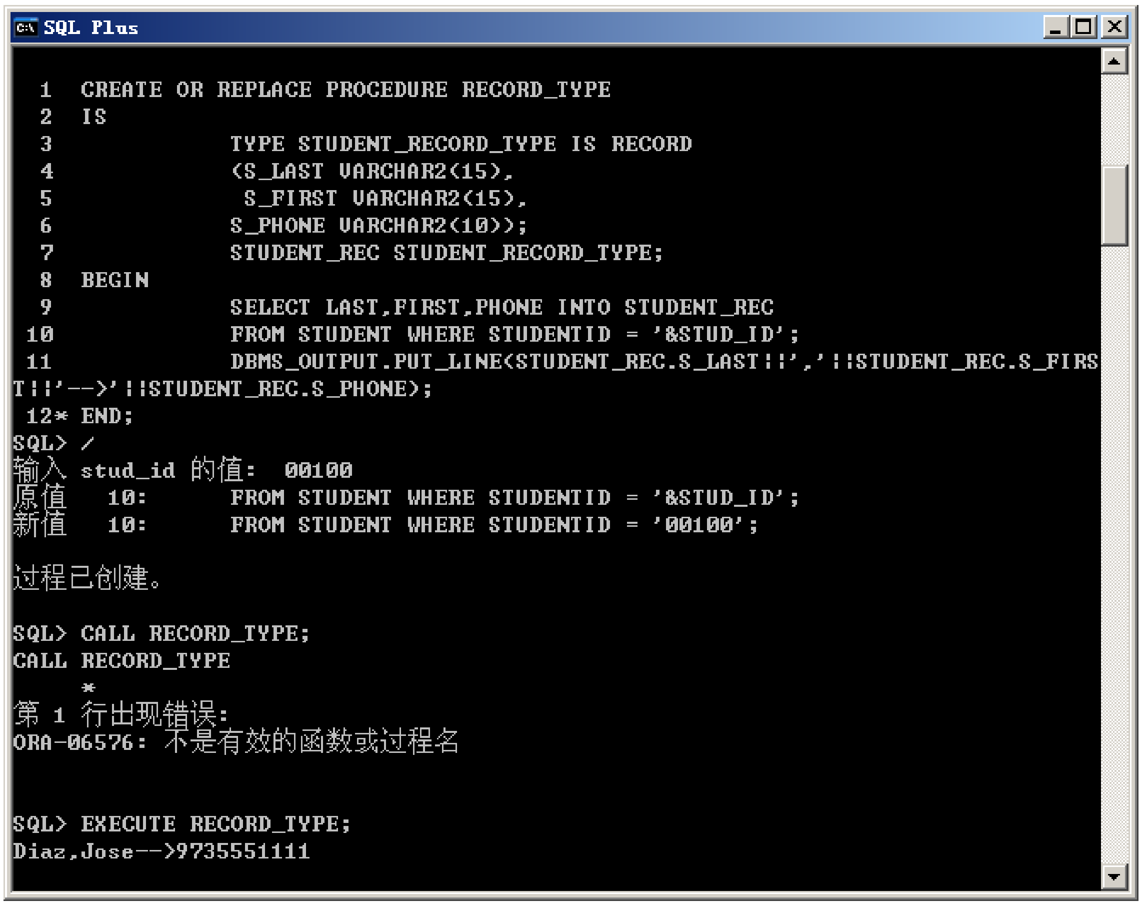
```
Unsupport.
```

4.7 Record Type

4.7.1 RECORD

Oracle

Declare record type variable, and dynamically execute SQL statement, according to different input value to query the corresponding qualified records.



```
  1  CREATE OR REPLACE PROCEDURE RECORD_TYPE
  2  IS
  3             TYPE STUDENT_RECORD_TYPE IS RECORD
  4             (S_LAST VARCHAR2(15),
  5              S_FIRST VARCHAR2(15),
  6             S_PHONE VARCHAR2(10));
  7             STUDENT_REC STUDENT_RECORD_TYPE;
  8  BEGIN
  9             SELECT LAST,FIRST,PHONE INTO STUDENT_REC
 10             FROM STUDENT WHERE STUDENTID = '&STUD_ID';
 11             DBMS_OUTPUT.PUT_LINE(STUDENT_REC.S_LAST||','||STUDENT_REC.S_FIRS
T||'-->'||STUDENT_REC.S_PHONE);
 12* END;
SQL> /
输入 stud_id 的值:  00100
原值   10:          FROM STUDENT WHERE STUDENTID = '&STUD_ID';
新值   10:          FROM STUDENT WHERE STUDENTID = '00100';

过程已创建。

SQL> CALL RECORD_TYPE;
CALL RECORD_TYPE
     *
第 1 行出现错误:
ORA-06576: 不是有效的函数或过程名


SQL> EXECUTE RECORD_TYPE;
Diaz,Jose-->9735551111
```

DBMaker

Sp code:

```
CREATE PROCEDURE dym7 (IN cd DECIMAL)
RETURNS VARCHAR (15) V1, VARCHAR (15) V2, DECIMAL (2, 0) V3
LANGUAGE SQL
BEGIN
    DECLARE str char (128);
    DECLARE cur CURSOR WITH RETURN FOR stmt;

    SET str= 'select LNAME, FNAME, DEPTID from EMPLOYEE where EMPLOYEEID=?';

    PREPARE stmt FROM str;
    OPEN cur USING cd;
END;
```

Create and execute result:

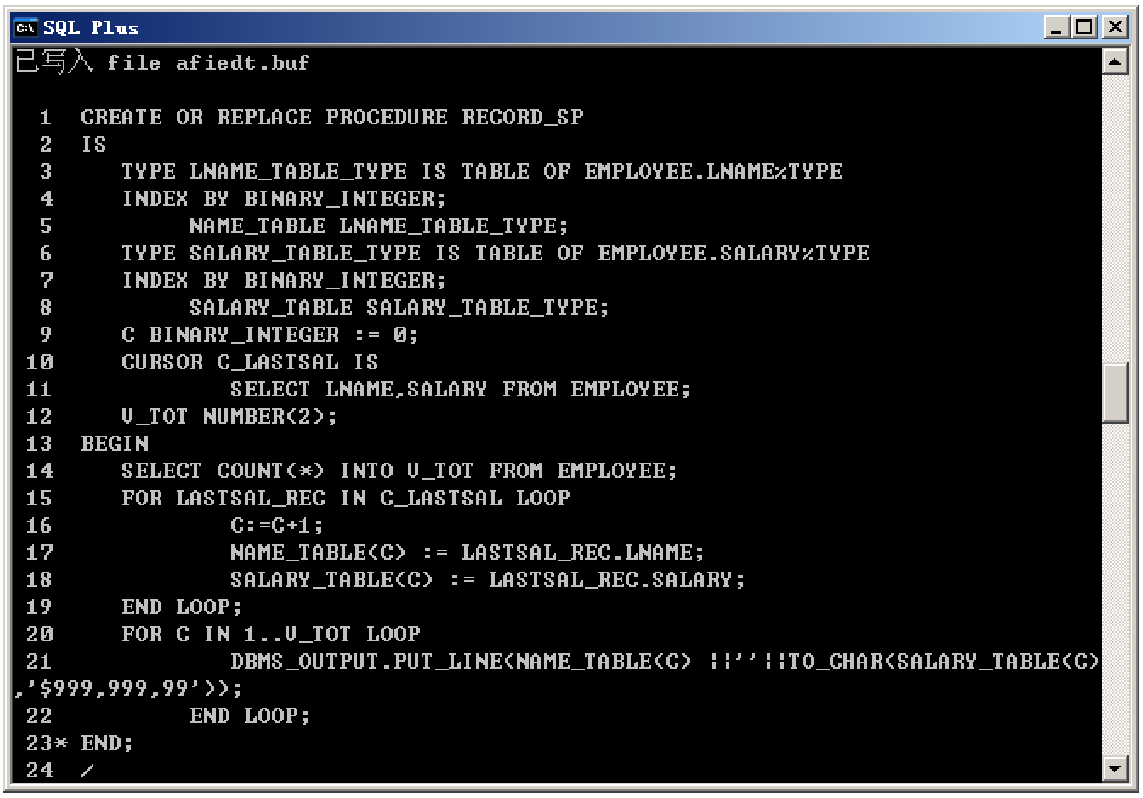
```
dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\dsp7.sp';

dmSQL> CALL DYM7 (111);

      V1              V2                  V3
=============== =============== =====================
    Smith           john                  10
1 rows selected
```

Compare:

1) Because DBMaker doesn’t support record type declare format, use ordinary type to declare.
2) We use dynamic cursor to print corresponding person information according input different ID.

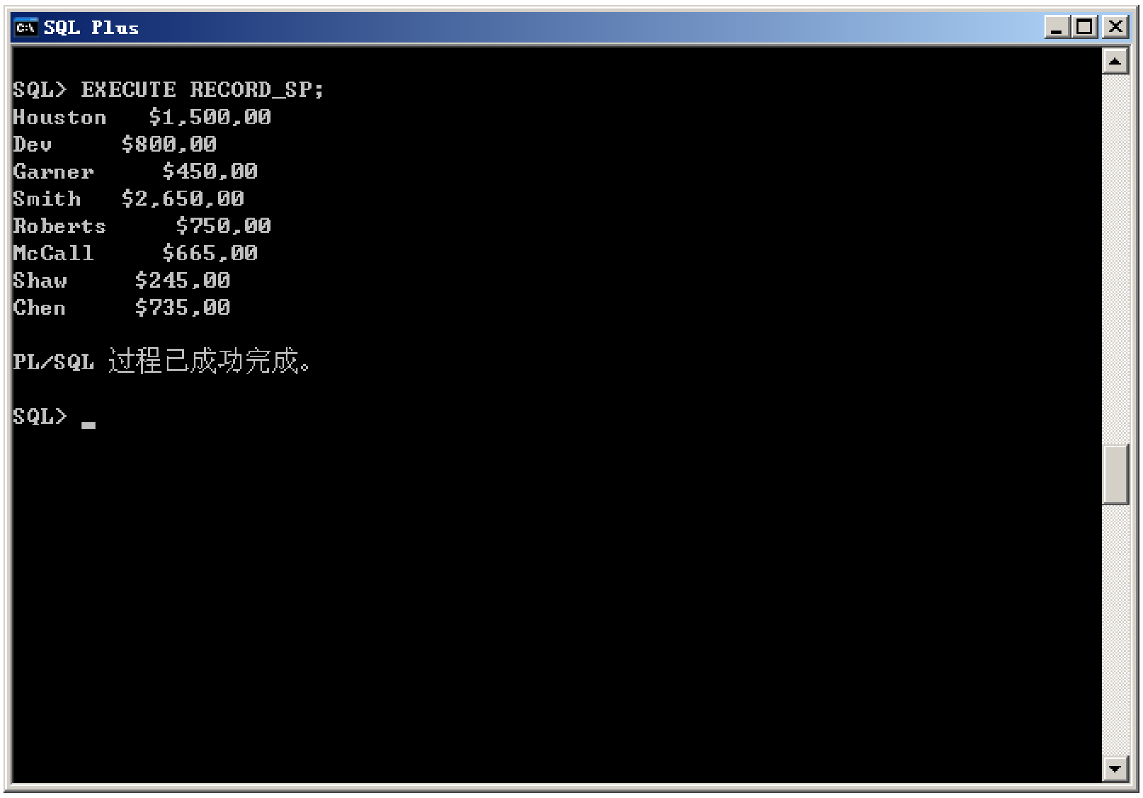
4.7.2 PL/SQL Record Type Table

Oracle



```
已写入 file afiedt.buf

CREATE OR REPLACE PROCEDURE RECORD_SP
IS
    TYPE LNAME_TABLE_TYPE IS TABLE OF EMPLOYEE.LNAME%TYPE
    INDEX BY BINARY_INTEGER;
        NAME_TABLE LNAME_TABLE_TYPE;
    TYPE SALARY_TABLE_TYPE IS TABLE OF EMPLOYEE.SALARY%TYPE
    INDEX BY BINARY_INTEGER;
        SALARY_TABLE SALARY_TABLE_TYPE;
    C BINARY_INTEGER := 0;
    CURSOR C_LASTSAL IS
            SELECT LNAME,SALARY FROM EMPLOYEE;
    V_TOT NUMBER(2);
BEGIN
    SELECT COUNT(*) INTO V_TOT FROM EMPLOYEE;
    FOR LASTSAL_REC IN C_LASTSAL LOOP
            C:=C+1;
            NAME_TABLE(C) := LASTSAL_REC.LNAME;
            SALARY_TABLE(C) := LASTSAL_REC.SALARY;
    END LOOP;
    FOR C IN 1..V_TOT LOOP
            DBMS_OUTPUT.PUT_LINE(NAME_TABLE(C) ||''||TO_CHAR(SALARY_TABLE(C)
,'$999,999.99'));
        END LOOP;
END;
/
```

Execute result:



```
SQL> EXECUTE RECORD_SP;
Houston   $1,500.00
Dev     $800.00
Garner      $450.00
Smith   $2,650.00
Roberts      $750.00
McCall      $665.00
Shaw     $245.00
Chen     $735.00

PL/SQL 过程已成功完成。

SQL> _
```

DBMaker doesn't support table type.

4.7.3 ASSIGN VALUE FOR ROW OF THE TABLE IN LOOP

Use Sunday_sp to print all of Sunday in 2004.

Oracle

```
Create or replace
```

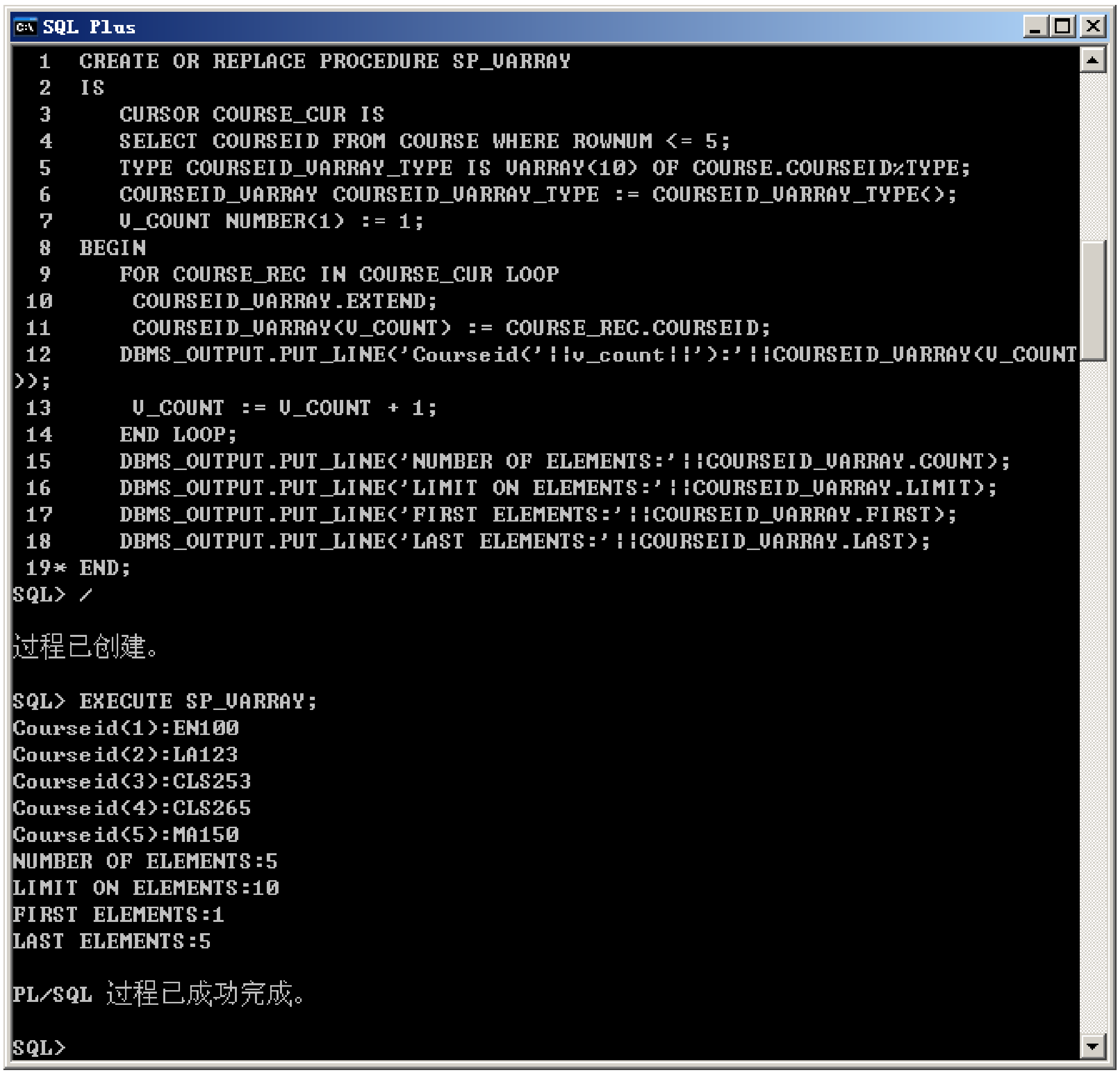
```
PROCEDURE SUNDAY_SP
IS
    TYPE DATE_TABLE_TYPE IS TABLE OF DATE
    INDEX BY BINARY_INTEGER;
    SUNDAY_TABLE DATE_TABLE_TYPE;

    V_DAY BINARY_INTEGER:= 1;
    V_DATE DATE;
    V_COUNT NUMBER (3):= 1;
BEGIN
    V_DATE :='01-1月- 04';
    WHILE V_COUNT <=365 LOOP
    IF UPPER(TO_CHAR(V_DATE,'DAY')) LIKE '%星期日%' THEN
            SUNDAY_TABLE (V_DAY):= V_DATE;
            DBMS_OUTPUT.PUT_LINE (TO_CHAR (SUNDAY_TABLE (V_DAY),'MONTH DD, YYYY'));
            V_DAY:= V_DAY + 1;
    END IF;
    V_COUNT:= V_COUNT + 1;
    V_DATE:= V_DATE + 1;
    END LOOP;
END;
```

DBMaker

```
DBMaker doesn’t support table type.
```

4.7.4 PL/SQL VARIABLE ARRAY

Oracle：



```
SQL Plus
  1  CREATE OR REPLACE PROCEDURE SP_VARRAY
  2  IS
  3     CURSOR COURSE_CUR IS
  4     SELECT COURSEID FROM COURSE WHERE ROWNUM <= 5;
  5     TYPE COURSEID_VARRAY_TYPE IS VARRAY(10) OF COURSE.COURSEID%TYPE;
  6     COURSEID_VARRAY COURSEID_VARRAY_TYPE := COURSEID_VARRAY_TYPE();
  7     V_COUNT NUMBER(1) := 1;
  8  BEGIN
  9     FOR COURSE_REC IN COURSE_CUR LOOP
 10      COURSEID_VARRAY.EXTEND;
 11      COURSEID_VARRAY(V_COUNT) := COURSE_REC.COURSEID;
 12     DBMS_OUTPUT.PUT_LINE('Courseid('||v_count||'):'||COURSEID_VARRAY(V_COUNT
));
 13      V_COUNT := V_COUNT + 1;
 14     END LOOP;
 15     DBMS_OUTPUT.PUT_LINE('NUMBER OF ELEMENTS:'||COURSEID_VARRAY.COUNT);
 16     DBMS_OUTPUT.PUT_LINE('LIMIT ON ELEMENTS:'||COURSEID_VARRAY.LIMIT);
 17     DBMS_OUTPUT.PUT_LINE('FIRST ELEMENTS:'||COURSEID_VARRAY.FIRST);
 18     DBMS_OUTPUT.PUT_LINE('LAST ELEMENTS:'||COURSEID_VARRAY.LAST);
 19* END;
SQL> /

过程已创建。

SQL> EXECUTE SP_VARRAY;
Courseid(1):EN100
Courseid(2):LA123
Courseid(3):CLS253
Courseid(4):CLS265
Courseid(5):MA150
NUMBER OF ELEMENTS:5
LIMIT ON ELEMENTS:10
FIRST ELEMENTS:1
LAST ELEMENTS:5

PL/SQL 过程已成功完成。

SQL>
```

DBMaker

The array type wasn't supported in SQL SP of DBMaker.

4.8 Integrated Type Stored Procedure

4.8.1 DISPLAY THE COLUMN OF TABLE（DYNAMIC SQL WITH INPUT PARAMETER IN）

Oracle

```
create or replace
procedure tab_col (p_tablename in varchar2, p_colname in varchar2)
    is
    tab_flag number default 0;
    col_flag number default 0;
    no_table exception;--- User-defined Exceptions,DBMaker doesn't support
    no_col exception;
    v_cur sys_refcursor;
    v_tablename varchar2 (20);
    v_colname varchar2 (20);
```

```
   v_sql varchar2 (1000);
   v_value varchar2 (200);
   begin
   v_tablename:=p_tablename;
   v_colname:=p_colname;
  select count (*) into tab_flag from user_tables where table_name=v_tablename
;
   if tab_flag=0 then
   raise no_table;
   end if;
   Select count (*) into col_flag from user_tab_cols where table_name=v_tablename and
column_name=v_colname;
   if col_flag=0 then
   raise no_col;
   end if;
   v_sql:='select '||v_colname||' from '||v_tablename;
   open v_cur for v_sql;
   loop
   fetch v_cur into v_value;
   exit when v_cur%notfound;
   dbms_output.put_line (v_value);
   end loop;
   exception
   when no_table then
   dbms_output.put_line ('the table does not exist');
   when no_col then
   dbms_output.put_line ('the col does not exist');
   when others then
   dbms_output.put_line ('some other errors occur');
   end;
```

Execute result:

```
DECLARE
  P_TABLENAME VARCHAR2 (200);
  P_COLNAME VARCHAR2 (200);
BEGIN
  P_TABLENAME:= 'DEPT';
  P_COLNAME := 'DEPTID';// input parameter（table name,and the column name in table）
  TAB_COL (
    P_TABLENAME => P_TABLENAME,
    P_COLNAME => P_COLNAME
  );
END;
Connecting to the database orcl.
10
20
30
40
Process exited.
```

```
Disconnecting from the database orcl.
```

DBMaker

Sp code:

```
CREATE PROCEDURE SYSADM.tab_col (IN p_tablename VARCHAR (20), IN p_colname VARCHAR (20))
RETURNS DECIMAL (2, 0) deptID_out
LANGUAGE SQL
BEGIN
     DECLARE v_tab VARCHAR (20);
     DECLARE v_col VARCHAR (20);
     DECLARE v_sql VARCHAR (2000);
     DECLARE CONTINUE HANDLER FOR NOT FOUND;

     DECLARE cur CURSOR
     WITH RETURN FOR stmt;
     SELECT v_col FROM v_tab;
     SET v_tab = p_tablename;
     SET v_col = p_colname;
     SET v_sql = 'SELECT' || v_col || 'FROM' || v_tab;
     PREPARE stmt FROM v_sql;
     OPEN cur;
     DEALLOCATE PREPARE stmt;
END;
```

Create dynamic SQL sp:

```
dmSQL> CREATE PROCEDURE FROM
'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\dsp5.sp';
```

Execute result:

```
dmSQL> call tab_col ('DEPT','DEPTID');


     DEPTID_OUT
=====================
ERROR (6002): [DBMaker] syntax error near or At: FROM?”
```

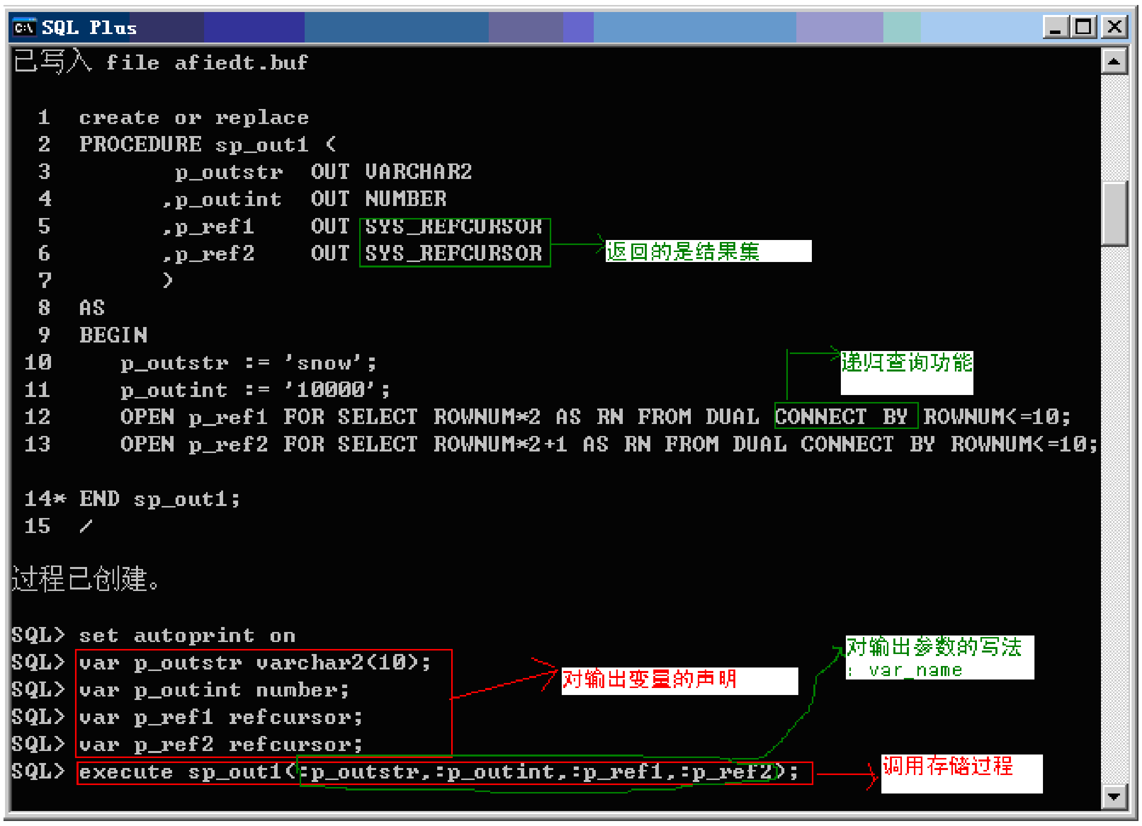
Compare:

```
In DBMaker,the declaration and open syntax of dynamic cursor is different from ordinary
cursor:
declare：DECLARE <dynamic cursor name> [WITH RETURN] FOR <prepare statement name>
open：OPEN <dynamic cursor name> [<input using clause>]
```

4.8.2 THE PROCEDURE WITHOUT PARAMETER

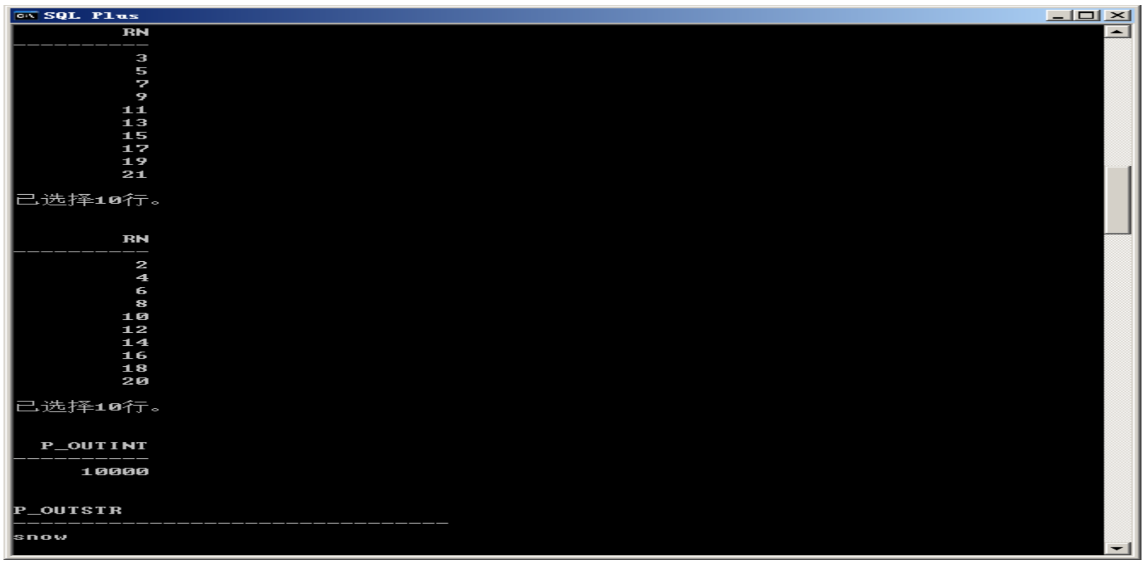
Oracle

Create and execute procedure:



```
已写入 file afiedt.buf

  1  create or replace
  2  PROCEDURE sp_out1 (
  3          p_outstr  OUT VARCHAR2
  4         ,p_outint  OUT NUMBER
  5         ,p_ref1    OUT SYS_REFCURSOR
  6         ,p_ref2    OUT SYS_REFCURSOR
  7         )
  8  AS
  9  BEGIN
 10     p_outstr := 'snow';
 11     p_outint := '10000';
 12     OPEN p_ref1 FOR SELECT ROWNUM*2 AS RN FROM DUAL CONNECT BY ROWNUM<=10;
 13     OPEN p_ref2 FOR SELECT ROWNUM*2+1 AS RN FROM DUAL CONNECT BY ROWNUM<=10;

 14* END sp_out1;
 15  /

过程已创建。

SQL> set autoprint on
SQL> var p_outstr varchar2(10);
SQL> var p_outint number;
SQL> var p_ref1 refcursor;
SQL> var p_ref2 refcursor;
SQL> execute sp_out1(:p_outstr,:p_outint,:p_ref1,:p_ref2);
```

Result:



```
        RN
----------
         3
         5
         7
         9
        11
        13
        15
        17
        19
        21

已选择10行。

        RN
----------
         2
         4
         6
         8
        10
        12
        14
        16
        18
        20

已选择10行。

  P_OUTINT
----------
     10000

P_OUTSTR
--------------------------------
snow
```

DBMaker

Sp code:

```
CREATE PROCEDURE dsp4 (OUT p_outstr VARCHAR (10), OUT p_outint DECIMAL (6, 0))
LANGUAGE SQL
BEGIN
    SET p_outstr = 'snow';
    SET p_outint = 10000;
END;
```

Create and execute result:

```
dmSQL> CREATE PROCEDURE from'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\sdp4.sp';
dmSQL> call dsp4 (?,?);--In DBMaker SQL SP, only need to use placeholder " ?" to replace
the output parameter in CALL statement, don't need to rewrite the parameter again.
P_OUTSTR                          : snow
P_OUTINT                          : 10000
```

Compare:

1) About parameters writing in a different order

2) The declare condition is different for parameters, In DBMaker that must need to specify the type size, but in Oracle only need to specify the type, then it will be in the parameter list of header. If you don't specify the size the type which can specify size, error will be returned, for example:

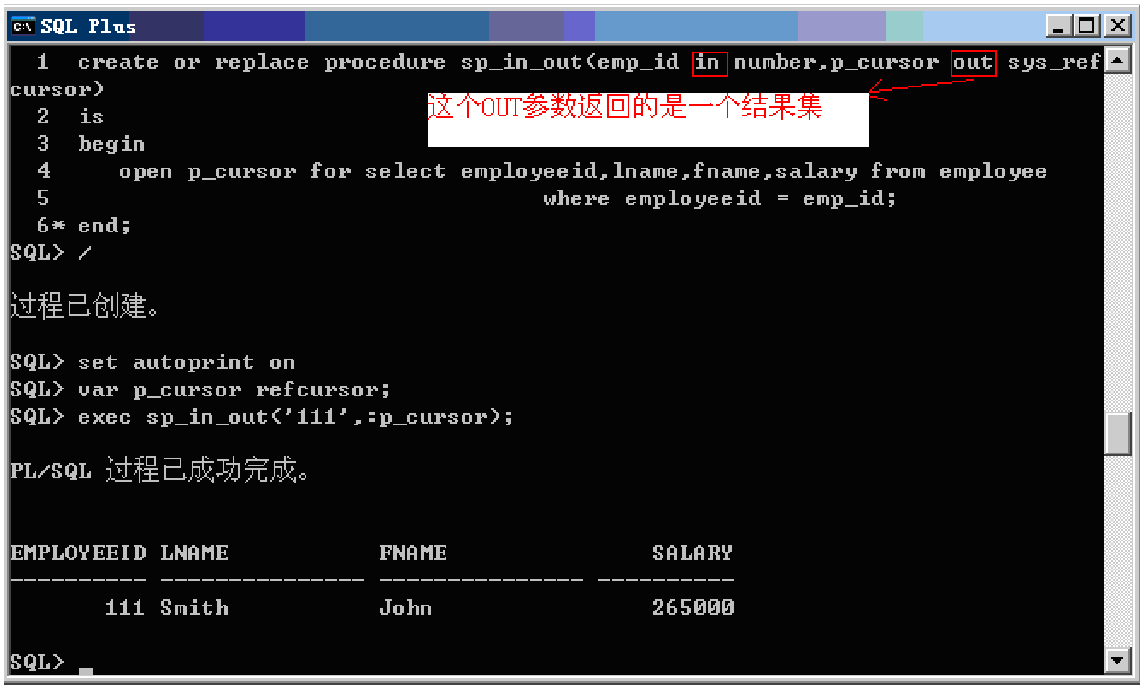
```
CREATE PROCEDURE dsp4 (OUT p_outstr VARCHAR, OUT p_outint DECIMAL)—only specify type
LANGUAGE SQL
BEGIN
    SET p_outstr = 'snow';
    SET p_outint = 10000;
END;
```

The error happen when create sp error info as below:

```
dmSQL> create procedure from 'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\sdp4.sp';
ERROR (6589): [DBMaker] preprocessor translating error: error message in: DSP4SYSADM.msg
```

4.8.3 THE STORED PROCEDURE WITH IN AND OUT PARAMETER

Oracle



```
create or replace procedure sp_in_out(emp_id in number,p_cursor out sys_ref
cursor)
is
begin
    open p_cursor for select employeeid,lname,fname,salary from employee
                                where employeeid = emp_id;
end;
SQL> /

过程已创建。

SQL> set autoprint on
SQL> var p_cursor refcursor;
SQL> exec sp_in_out('111',:p_cursor);

PL/SQL 过程已成功完成。


EMPLOYEEID LNAME           FNAME           SALARY
---------- --------------- --------------- ----------
       111 Smith           John            265000

SQL>
```

DBMaker

SQL SP code:

```
CREATE PROCEDURE dsp3 (IN emp_id DECIMAL (3, 0))
RETURNS DECIMAL (3, 0) VAL1, VARCHAR (15) VAL2, VARCHAR (15) VAL3, DECIMAL (6, 0) VAL4
LANGUAGE SQL
```

```
BEGIN
    DECLARE cur CURSOR WITH RETURN FOR
    SELECT EMPLOYEEID, LNAME, FNAME, SALARY
    FROM EMPLOYEE
    WHERE EMPLOYEEID = emp_id;
    OPEN cur;
END;
```

Create and execute result:

```
DmSQL> create procedure from 'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\dsp3.sp';
DmSQL> CALL DSP3 (111);

       VAL1              VAL2           VAL3            VAL4
==================== =============== =============== ================
       111               smith          john          265000

1 rows selected
```

DBMaker&Oracle SP compare:

In Oracle, the output variables need to be declared and call sp with: var_name format, but in DBMaker that doesn't need to list variable when return result set by using WITH RETURN.

4.8.4 SYS_REFCURSOR AND PROCEDURE CALL PROCEDURE

Oracle

Create stored procedure proc_ref_cursor, pass query result set v_rc to sp: proc_ref_cursor when call_proc_ref_cursor, then get the value from the result set by loop.

```
create or replace procedure proc_ref_cursor (rc in sys_refcursor)
as
  v_a number;
  v_b varchar2 (10);

begin
  loop
    fetch rc into v_a, v_b;
    exit when rc%notfound;
    dbms_output.put_line (v_a || ' ' || v_b);
  end loop;
end;
```

```
create or replace procedure call_proc_ref_cursor
as
  v_rc sys_refcursor;
begin
  open v_rc for
    select * from table_ref_cursor;
  proc_ref_cursor (v_rc);
  close v_rc;
end;
```

Execute result:

```
Connecting to the database orcl.
1, one
2 ,two
3, three
4, four
Process exited.
Disconnecting from the database orcl.
```

DBMaker

DBMaker doesn't support result set type cursor sys_refcursor, and not has the function that return the result set with WITH RETURN，so we create below 2 sp，call dsp2 by call_dsp2 to return record set.

```
CREATE PROCEDURE dsp2
RETURNS DECIMAL VAL1, VARCHAR (10) VAL2
LANGUAGE SQL
BEGIN
    DECLARE cur CURSOR WITH RETURN FOR
    SELECT * FROM DEPT;
    OPEN cur;
END;
```

```
CREATE PROCEDURE call_dsp2
RETURNS DECIMAL VAL1, VARCHAR (10) VAL2
LANGUAGE SQL
BEGIN
    DECLARE cur CURSOR WITH RETURN FOR
    CALL dsp2; --call dsp2
    OPEN cur;
END;
```

Create and execute result:

```
dmSQL> create procedure from 'F:\snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\CALL_dsp2.sp';

dmSQL> call call_dsp2;

        VAL1            VAL2
===================== ==========
                   10 finance
                   20 Infosys
                   30 sales
                   40 marketing
4 rows selected
```

4.8.5 THE SP AND ITS PARAMETER TYPE IS IN OUT

Oracle

Create sp:

```
create or replace procedure sp_3paras (
   v_id in dept.deptid%type,
   v_name out dept.deptname%type,
   v_loc in out dept.location%type)
is
   loc1   dept.location%type;
   dname1 dept.deptname%type;
begin
   select location into loc1
   from dept
   where deptid=v_id;
   select deptname into dname1
   from dept
   where deptid=v_id;
   v_loc:='地址:'||loc1;
   v_name:='姓名:'||dname1;
end;
```

Execute and result call sp sp_3paras by writing an anonymous block:

```
declare
     myid dept.deptid%type;
     mydname dept.deptname%type;
     myloc dept.location%type;
begin
     myid: =10;
     myloc: = 'Woodbridge';
     sp_3paras (myid, mydname, myloc);
     dbms_output.put_line (myid);
     dbms_output.put_line (mydname);
     dbms_output.put_line (myloc);
end;
```

Result:

```
V_NAME = 姓名:Finance
V_LOC = 地址:Charlotte
```

DBMaker

DBMaker doesn’t support in out type parameter and output statement, if use to select that need to implement with cursor.

SQL SP dsp1code:

```
CREATE PROCEDURE dsp1 (IN v_id DECIMAL (2, 0), OUT v_name VARCHAR (12), OUT v_loc VARCHAR
(15))
LANGUAGE SQL
BEGIN
     DECLARE cur CURSOR FOR
     SELECT DEPTNAME, LOCATION FROM DEPT
     WHERE DEPTID = v_id;
     OPEN cur;
     FETCH cur INTO v_name, v_loc;
```

```
    CLOSE cur;
END;
```

Create sp:

```
Create procedure from 'F: \snow\SQLSP\PL_ORACLE\sp\DBMaker_sp\dsp1.sp';
```

Call and return result:

```
dmSQL> call dsp1 (10,?,?);
V_NAME                          : finance
V_LOC                           : charlotte
```

Oracle&DBMaker sp compare:

1) DBMaker SQLSP needs to write sp code in all capitals for current version.
2) The declare order is different for IN/OUT paremater"

   Oracle: var_name in|out|in out type

   DBMaker: in |out var_name type
3) DBMaker sp must use LANGUAGE SQL keyword, Oracle sp need to use keyword IS|AS between header and begin.
4) Some data type such as varchar2 and number etc which supported by ORACLE but DBMaker, so we need to find the replaceable and identifiable data type for DBMaker to replace.
5) In Oracle that doesn't need Declare when declare but var_name type directly, but in DBMaker that need to Declare and the syntax is Declare var_name type.
6) In DBMaker, The parameter name and column name cannot repeat, else the error will be happen.

5. Some features need to be cared in stored procedure

5.1 Data Table Alias

Oracle

In Oracle, you can add as for data table alias that may cause conflict with keyword AS of stored procedure.

DBMaker

Not any limit on as keyword in this part.

5.2 When Select One of the Column in Stored Procedure

Oracle

If select one of the column, that must followed by into, if select total record, can use cursor.

DBMaker

Select must use in cursor.

5.3 When Use SELECT INTO Syntax in Stored Procedure

Oracle

You must ensure database has this record before use select...into... syntax, else "no data found" exception will return.
Before use this syntax, you'd better check the record exist in database or not with select count (*) from …, if exist, then can use syntax select...Into...

DBMaker

Select must use in cursor.

5.4 The Relationship about Alias and Column Name in Stored Procedure

Oracle

In Oracle stored procedure, the alias and filed name cannot be same, otherwise although it can be complied successfully but the error will be returned in run phase.

DBMaker

DBMaker support alias and column name are same in stored procedure.

e.g.:

```
CREATE PROCEDURE alia
RETURNS VARCHAR (15) val1, VARCHAR (15) val2, DECIMAL (2, 0) val3
LANGUAGE SQL
BEGIN
    DECLARE alias_cur CURSOR WITH RETURN FOR
    SELECT FNAME, LNAME, DEPTNAME FROM EMPLOYEE fname, DEPT deptid
// alias and column name are same and sp run normally.
    WHERE
    fname.DEPTID=deptid.DEPTID;
    OPEN alias_cur;
END;
```

Execute result:

```
dmSQL> CALL alia;

     VAL1             VAL2              VAL3
=============== =============== =====================
john                  smith                 finan
larry                 houston               marke
sandi                 roberts               finan
derek                 dev                   infos
john                  stanley               sales
sunny                 chen                  finan

6 rows selected
```

6. Oracle & DBMaker SP compare conclusion

6.1 Basic Comparison for SP

1) DBMaker SQLSP needs to write the sp code with capital letters in current version.
2) The INPUT/OUTPUT parameters declare order is different:

   Oracle：var_name in|out|in out type

   DBMaker: in |out var_name type
3) About the parameters declaration of Header, not only the order is different, but also type format. In Oracle, only need to specify the type but SIZE, but in DBMaker that also need to specify the size such like TYPE (SIZE).
4) DBMaker sp must use LANGUAGE SQL keyword, Oracle sp need to use keyword IS|AS between header and begin.
5) The declare location is different for parameter, In Oracle that doesn't need Declare keyword, and the parameter declaration writing after IS|AS, and before begin, but in DBMaker that need to write the parameter declaration between BEGIN……END.
6) The data types varchar2 and number which supported by ORACLE but DBMaker, So we need to find the replaceable and identifiable data type for DBMaker to replace.
7) In Oracle that doesn't need Declare when declare but var_name type directly, but in DBMaker that need to Declare and the syntax is Declare var_name type.
8) About assignment symbol, in Oracle use "：=" assignment value to parameter, but in DBMaker use "=" and also need to use SET keyword before parameter.
9) The parameter name and column name cannot be same in DBMaker, else error will happen.
10) Create syntax is different, in Oracle, create directly or with @path\sp_file.sql, in DBMaker, create with syntax CREATE PROCEDURE FROM path\sp_file.sql.
11) Execute procedure is different, in Oracle, you can use EXECUTE (EXEC) sp_name to call, or write an anonymous block to call procedure. In DBMaker , call procedure only by syntax CALL sp_name（[parameter]）；.
12) About IF, Oracle :IF ……ELSIF, DBMaker :IF……ELSEIF.