

PLSQL Native Compilation

- PL/SQL Native Compilation
- Using Native Compilation
- Procedure to convert the entire database and recompile all PL/SQL modules into native mode
- Testing
- Conclusion

Applies to:

PL/SQL - Version: 11.1.X

For additional information please see Metalink Doc ID's: 762351.1 and 216205.1

PL/SQL Native Compilation

In 11g, the new spfile.ora parameter PLSQL_CODE_TYPE decides the compilation mode for PL/SQL blocks. Only this parameter needs to be set to compile PL/SQL into native code.

There are 2 options for PLSQL_CODE_TYPE:

- 1) Interpreted - Using this mode, the PL/SQL statements in a PL/SQL unit are compiled into an intermediate form, system code, which is stored in the database dictionary and interpreted at run time.
- 2) Native -Using this mode, the native code will **not** be interpreted at runtime. **Hence it executes faster**. The default value for PLSQL_CODE_TYPE is set to interpreted.

Native compilation can be used with Oracle supplied packages / procedures / triggers and custom code.

The DBA_PLSQL_OBJECT_SETTINGS view includes the current PLSQL_CODE_TYPE column for each PL/SQL object which states whether an object has been compiled natively or interpreted.

Oracle recommends having an optimization level (set by PLSQL_OPTIMIZE_LEVEL) of at least 2. If the optimisation level is less than 2, the compiler generates interpreted code, regardless of the PLSQL_CODE_TYPE. If you specify NATIVE, the compiler warns you that NATIVE was ignored.

Note that in 11g, the parameters PLSQL_NATIVE_LIBRARY_DIR and PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT have no effect and are not required, as natively compiled code is now stored in the database, and not in a file system.

Natively compiled subprograms and interpreted subprograms can call each other.

Using Native Compilation

You can perform PL/SQL native compilation by setting a session parameter before creating or recompiling stored code:

```
ALTER SESSION SET PLSQL_CODE_TYPE = NATIVE;
```

Or

```
ALTER PROCEDURE my_proc COMPILE PLSQL_CODE_TYPE = NATIVE
```

Procedure to convert the entire database and recompile all PL/SQL modules into native mode

- 1) Shut down database
- 2) Edit spfile.ora and set PLSQL_CODE_TYPE =native and plsqli_optimise_level=2 (at least)
- 3) connect sys/password as sysdba
startup upgrade

4) @\$ORACLE_HOME/rdbms/admin/dbmsupgnv.sql (which updates the execution mode of all PL/SQL modules to native) (You can use the TRUE command line parameter with the dbmsupgnv.sql script to exclude package specs from recompilation to NATIVE, saving time in the conversion process.)

5) shutdown immediate

startup

@\$ORACLE_HOME/rdbms/admin/utlrp.sql (to recompile all invalid objects)

Testing

I choose 2 PL/SQL procedures, one being a computation-intensive procedure, and the other which is a very simple procedure (see below).

Although Oracle states that compilation time could be slower and execution times might be faster for native compilation, my results indicated that there is a consistent improvement both compilation and execution.

	Interpreted code	Native code	Improvement
Procedure 1			
Compilation	0.76 secs	0.3 secs	39.5% faster
Execution	54.1 secs	48.88 secs	10.7% faster
Procedure 2			
Compilation	0.1 secs	0.03 secs	30% faster
Execution	0.02 secs	0.01 secs	50% faster

Converting the whole database to Native code:

Although it is very simple to convert a database, a restart is required and recompilation of invalid objects after running the convert script is required.

It is necessary to check how many invalid objects are before the change, as with the conversion all invalid objects would need to be recompiled.

In the database I used to test it, there were about 50000 invalid objects after native conversion.

To recompile the invalid objects took 5 hours.

Procedures used during testing:

Row#	OWNER	NAME	TYPE	PLSQL_OPTIMIZE_LEVEL	PLSQL_CODE_TYPE
1	MARGA	MY_TEST	PROCEDURE	2	NATIVE
2	MARGA	PERFECT_TRIANGLES	PROCEDURE	2	NATIVE

Procedure 1

http://www.oracle.com/technology/sample_code/tech/pl_sql/htdocs/x/Native_Compilation/Cr_Computational_Intensive_Algorithm.htm

```
set serveroutput on
```

```
drop procedure Perfect_Triangles;
```

```
create or replace procedure Perfect_Triangles ( p_max in integer ) is
  t1 integer; t2 integer;
  long integer; short integer; hyp number; ihyp integer;
  type side_r is record ( short integer, long integer );
  type sides_t is table of side_r index by binary_integer;
  unique_sides sides_t; n integer:=0 /* curr max elements in unique_sides */;
  dup_sides sides_t; m integer:=0 /* curr max elements in dup_sides */;
```

```
procedure Store_Dup_Sides ( p_long in integer, p_short in integer ) is
  mult integer:=2; long_mult integer:=p_long*2; short_mult integer:=p_short*2;
begin
  while ( long_mult < p_max ) or ( short_mult < p_max )
    loop
```

```

    n := n+1;
    dup_sides(n).long := long_mult; dup_sides(n).short := short_mult;
    mult := mult+1; long_mult := p_long*mult; short_mult := p_short*mult;
end loop;
end Store_Dup_Sides;

function Sides_Are_Unique ( p_long in integer, p_short in integer )
return boolean is
begin
for j in 1..n
loop
if ( p_long = dup_sides(j).long ) and ( p_short = dup_sides(j).short )then
return false;
end if;
end loop;
return true;
end Sides_Are_Unique;

begin /* Perfect_Triangles */
t1 := Dbms_UTILITY.Get_Time;
for long in 1..p_max
loop

for short in 1..long
loop
hyp := Sqrt ( long*long + short*short ); ihyp := Floor(hyp);
if hyp-ihyp < 0.01 then
if ( ihyp*ihyp = long*long + short*short )then
if Sides_Are_Unique ( long, short )then
m := m+1;
unique_sides(m).long := long; unique_sides(m).short := short;
Store_Dup_Sides ( long, short );
end if;
end if;
end if;
end loop;

end loop;
t2 := Dbms_UTILITY.Get_Time;
Dbms_Output.Put_Line (
chr(10) || To_Char( ((t2-t1)/100), '9999.9' ) || ' sec' );
end Perfect_Triangles;
/
Show Errors

```

Procedure 2

```

create or replace procedure my_test is
begin
dbms_output.put_line('hello world');
end;
/
show errors;

```

Conclusion

Oracle has certified the use of compiled (native) PL/SQL code in Oracle Application Release 11i environments.

While Oracle states that compilation times may be slower for native PL/SQL code, my initial tests have shown better compilation times than interpreted PL/SQL. The improvement in runtime performance ranges from 10 up to 50% in some cases.

More large scale benchmark testing is required in this area as well as testing large patches in 11i.