Oracle10G
New Features in PL/SQL

Oracle Corporation

Marc Sewtz , Michael Hichwa

Thu., December 11th 2003

ORACLE

Agenda

- PL/SQL Performance Improvements
- PL/SQL New Features
- PL/SQL Programming Tips and Best Practices
- Using PL/SQL

Improved Performance

- Oracle Database 10g brings a new PL/SQL compiler and a newly tuned PL/SQL execution environment.
- PL/SQL programs will execute twice as fast as before.

Performance results come from three distinct sources:

- Tests done at Oracle HQ using a test suite, which is downloadable for customers to use from the OTN site (http://otn.oracle.com/tech/pl_sql/htdocs/New_In_10gR1.htm).
- Customer tests conducted during the 10g Beta Program by the Oracle Partner IFS.
- Tests using the benchmark suite developed and owned by the Applications Development Team at Oracle HQ.

Improved Performance

PL/SQL Compiler font-end

- Compiler font-end analyzes the source code for a single PL/SQL compilation unit, checking it for syntactical and semantic correctness.
- Output of the front-end is either an internal representation, which exactly captures the source code's semantics, or an error report.

Improved Performance

PL/SQL Compiler back-end

- Compiler back-end generates an executable representation of the program in the machine code of the target machine.
- Before 9iR1, the output was always code for the PL/SQL Virtual Machine.
- 9iR1 introduced the option (via native compilation) to output code for the hardware of the underlying computer.

Improved Performance

PL/SQL Compiler back-end

- The back-end can generate its output code in two different representations:
  - In the interpreted mode, it simply stores the machine code in system-managed structures in the SYS schema.
  - In the native mode, it translates the machine code into C source code with the same semantics.

Improved Performance

What changes were made in 10g?

- Front-end support for new language features:
  - The binary_float and binary_double data types (IEEE data types)
  - The regexp_like, regexp_instr, regexp_substr and regexp_replace built-ins to support regular expression manipulation with standard POSIX syntax
  - Multiset operations on nested table instances supporting operations like equals, union, intersect, except, member
  - User-defined quote character
  - INDICES OF and VALUES OF syntax for FORALL

Improved Performance

What changes were made in 10g?

- Brand new machine code generator using state-of-the-art optimizing technology
- The new code generator has existed side-by-side with the old one in the ORACLE executable for quite some time and a switch has allowed choosing between the one or the other.

Improved Performance

What changes were made in 10g?

- Substantial PVM upgrade
  - Obsolete instructions have been removed, and new ones have been added.
  - The system for consuming the instructions has been streamlined.
  - The C routines that implement the instructions have been tuned.

Improved Performance

What changes were made in 10g?

- Changes in the regime for native compilation
  - Pre 10g, the DLL generated by the back-end was stored as a file on an operating system directory.
  - In 10g, the DLL is stored in the database and is cached as a file on an operating system directory only on demand.
  - The configuration steps that the DBA follows to set up for native PL/SQL compilation have been radically simplified.
  - The dependency on the platform's make utility has been removed.
  - The subsystem that derives the C code from the machine code has been reworked to generate more efficient C.

Improved Performance

Performance results

- Baseline_80 shows the subset of programs that run in 80 and later under the conditions 80, 8i, 9iR2 native and 10g native, optimize level 2. (Full results for all the available compilation conditions are included with the download kit.) Here the improvement factor is calculated with respect to 80.
- Baseline_9iR2 shows the larger subset of programs that run in 9iR2 and later under the conditions 9iR2 interpreted, 9iR2 native and 10g native, optimize level 2. Here the improvement factor is calculated with respect to 9iR2 interpreted.
- Baseline_10g shows all programs under the conditions 10g interpreted, optimize level 1, 10g interpreted, optimize level 2, 10g native, optimize level 1 and 10g native, optimize level 2. Here the improvement factor is calculated with respect to 10g interpreted level.

Improved Performance

Performance results

baseline_80	80	8i	9iR2 Nat	10g Nat L2
Min	1.00	0.94	1.54	1.64
1st quartile	1.00	1.08	1.85	2.94
Median	1.00	1.13	2.27	3.94
3rd quartile	1.00	1.40	2.63	5.36
Max	1.00	2.07	3.33	10.57

Improved Performance

Performance results

baseline_9iR2	9iR2 Int	9iR2 Nat	10g Nat L2
Min	1.00	1.12	1.19
1st quartile	1.00	1.36	2.13
Median	1.00	1.50	2.64
3rd quartile	1.00	1.61	3.66
Max	1.00	2.88	7.57

Improved Performance

Performance results

baseline_10g	10g I L1	10g I L2	10g N L1	10g N L2
Min	1.00	1.00	1.05	1.05
1st quartile	1.00	1.03	1.17	1.26
Median	1.00	1.09	1.35	1.41
3rd quartile	1.00	1.13	1.49	1.67
Max	1.00	1.38	2.32	2.42

Regular Expressions

- You can use UNIX-style regular expressions while performing queries and string manipulations.
- Use the REGEXP_LIKE operator in SQL queries.
- Use the REGEXP_INSTR, REGEXP_REPLACE, and REGEXP_SUBSTR functions anywhere you would use INSTR, REPLACE, and SUBSTR.

FORALL Support for Non-Consecutive Indexes

- FORALL lets you run multiple DML statements very efficiently
- It can only repeat a single DML statement, unlike a general-purpose FOR loop.
- Use the INDICES OF and VALUES OF clauses with the FORALL statement to iterate over non-consecutive index values.
- For example, you can delete elements from a collection, and still use that collection in a FORALL statement.

Using FORALL with Part of a Collection

- The bounds of the FORALL loop can apply to part of a collection, not necessarily all the elements:

```
DECLARE
  TYPE NumList IS VARRAY(10) OF NUMBER;
  depts NumList :=
    NumList(20,30,50,55,57,60,70,75,90,92);
BEGIN
  FORALL j IN 4..7  -- use only part of varray
  UPDATE emp SET sal = sal * 1.10
  WHERE deptno = depts(j);
END;
```

Quoting Mechanism for String Literals

- Instead of doubling each single quote inside a string literal, specify your own delimiter character for the literal, and then use single quotes inside the string:

```
string_var :=  'I''m a string, you''re a string.';

string_var := q'!I'm a string, you're a string.!';
```

Implicit Conversion Between CLOB and NCLOB

- Implicit conversion from CLOB to NCLOB or from NCLOB to CLOB.
- Because this can be an expensive operation, it might help maintainability to continue using the TO_CLOB and TO_NCLOB functions.

New IEEE Floating-Point Types

- New data types BINARY_FLOAT and BINARY_DOUBLE represent floating-point numbers in IEEE 754 format.
- Because many computer systems support IEEE 754 floating-point operations through native processor instructions, these types are efficient for intensive computations involving floating-point data.

Improved Overloading

- You can now overload subprograms that accept different kinds of numeric arguments, to write math libraries with specialized versions of each subprogram for different data types.

Nested Table Enhancements

- Nested tables defined in PL/SQL have many more operations than previously.
- Compare nested tables for equality.
- Test whether an element is a member of a nested table.
- Test whether one nested table is a subset of another.
- Perform set operations such as union and intersection.

PL/SQL Programming Tips and Best Practices

- Keep the code simple and readable
- Use cursor FOR loops
- Avoid “select into …” statements
- Use bulk binds and bulk collect into
- Use associative arrays
- Always use PL/SQL packages
- Use cursor variables
- Use inline views

Using PL/SQL

- Data Manipulation
- Dynamic Web Applications
- HTML DB