Module 6

Optmizing PL/SQL Code

Created by Teerachai Laothong

Agenda

- Using Bind Variable in Stored Procedure
- Array processing and bulk-collect
- Using NOCOPY?
- IF Statement
- Function Result Cache
- Using Virtual Columns

Using Bind Variable in Stored Procedure

```
CREATE FUNCTION CONDITIONAL_COLUMN_LEN_BIND(
 TABLE_NAME IN VARCHAR2, COLUMN_NAME IN VARCHAR2,
 COND_FIELD IN VARCHAR2, COND_VALUE IN VARCHAR2) RETURN NUMBER
IS
  L_RESULT NUMBER := 0;
  L_STMT VARCHAR2(2000);
BEGIN
  L_STMT := 'SELECT MAX(LENGTH(' || COLUMN_NAME ||
   ')) FROM ' || TABLE_NAME || ' WHERE ' || COND_FIELD ||
   ' = :COND_VALUE';
  EXECUTE IMMEDIATE L_STMT INTO L_RESULT USING COND_VALUE;
  RETURN L_RESULT;
END;
/
```

Array processing and bulk-collect

- Using FOR loop
- Using bulk-collect
- Using bulk-collect memory-friendly

Using FOR loop

```
SET TIMING ON
BEGIN
  FOR aRow IN (SELECT CUST_ID, CUST_FIRST_NAME FROM CUSTOMERS)
  LOOP
    INSERT INTO sh.MY_CUSTOMERS (CUST_ID, CUST_FIRST_NAME)
      VALUES (aRow.CUST_ID, aRow.CUST_FIRST_NAME);
  END LOOP;
END;
/
SET TIMING OFF
```

Using bulk-collect

```
SET TIMING ON
DECLARE
  TAB_ID T_ID;
  TAB_NAME T_NAME;
BEGIN
  SELECT CUST_ID, CUST_FIRST_NAME
  BULK COLLECT INTO TAB_ID, TAB_NAME
  FROM CUSTOMERS;

  FORALL J IN TAB_ID.FIRST..TAB_ID.LAST
    INSERT INTO sh.MY_CUSTOMERS (CUST_ID, CUST_FIRST_NAME)
      VALUES (TAB_ID(J), TAB_NAME(J));
END;
/
SET TIMING OFF
```

Using bulk-collect memory-friendly

```
SET TIMING ON
DECLARE
  TAB_ID T_ID;
  TAB_NAME T_NAME;
  CURSOR MY_CURSOR IS SELECT CUST_ID, CUST_FIRST_NAME FROM CUSTOMERS;
BEGIN
  OPEN MY_CURSOR;
  LOOP
    FETCH MY_CURSOR BULK COLLECT INTO TAB_ID, TAB_NAME LIMIT 200;
    EXIT WHEN TAB_ID.COUNT = 0;

    FORALL J IN TAB_ID.FIRST..TAB_ID.LAST
      INSERT INTO sh.MY_CUSTOMERS (CUST_ID, CUST_FIRST_NAME)
        VALUES (TAB_ID(J), TAB_NAME(J));
  END LOOP;
  CLOSE MY_CURSOR;
END;
/
```

Using NOCOPY?

Passing an array

```
CREATE OR REPLACE FUNCTION MY_VALUE(ATABLE IN OUT TAB_NUMBERS,
 AIND IN NUMBER) RETURN NUMBER
IS
  L_VALUE NUMBER := 0;
BEGIN
  L_VALUE := ATABLE(AIND);
  RETURN L_VALUE;
END;
/
```

Using NOCOPY

```
CREATE OR REPLACE FUNCTION MY_VALUE_NOCOPY(
 ATABLE IN OUT NOCOPY TAB_NUMBERS,
 AIND IN NUMBER) RETURN NUMBER
IS
  L_VALUE NUMBER := 0;
BEGIN
  L_VALUE := ATABLE(AIND);
  RETURN L_VALUE;
END;
/
```

Conclusion

Even the example used in the Oracle documentation, about the use of NOCOPY, shows the same timing with or without the use of NOCOPY.

IF Statement

- Evaluate a compound `IF` statement of more than one condition, may affect performance.

The Original Code

```
SET TIMING ON
DECLARE
  TAB_QTY DBMS_SQL.NUMBER_TABLE;
  TAB_TIME DBMS_SQL.DATE_TABLE;
  CNT NUMBER := 0;
BEGIN
  SELECT AMOUNT_SOLD, TIME_ID
    BULK COLLECT INTO TAB_QTY, TAB_TIME FROM SALES;
  FOR J IN TAB_QTY.FIRST..TAB_QTY.LAST LOOP
    IF TAB_QTY(J) > 1 AND TAB_TIME(J) < '27-JUN-98' THEN
      CNT := CNT + 1;
    END IF;
  END LOOP;
END;
/
SET TIMING OFF
```

Reordering conditions

```
SET TIMING ON
DECLARE
  TAB_QTY DBMS_SQL.NUMBER_TABLE;
  TAB_TIME DBMS_SQL.DATE_TABLE;
  CNT NUMBER := 0;
BEGIN
  SELECT AMOUNT_SOLD, TIME_ID
   BULK COLLECT INTO TAB_QTY, TAB_TIME FROM SALES;
  FOR J IN TAB_QTY.FIRST..TAB_QTY.LAST LOOP
    IF TAB_TIME(J) < '27-JUN-98' AND TAB_QTY(J) > 1 THEN
      CNT := CNT + 1;
    END IF;
  END LOOP;
END;
/
SET TIMING OFF
```

Conclusion

- This behavior is called `short-circuit IF`, because the execution flow takes the shortest route to the destination.
- A similar behavior also occurs in logical ORed conditions, but in this case, the short circuit shows when the first condition is true and hence the predicate.

Function Result Cache

- Using the result cache can lead to a huge performance gain when we have a deterministic function—a function which always returns the same result for the same parameters—often invoked with the same parameters.

Without Result Cache

```
CREATE OR REPLACE FUNCTION C_N_K (N IN NUMBER, K IN NUMBER)
  RETURN NUMBER
IS
  N_FAT NUMBER := 1;
  K_FAT NUMBER := 1;
  N_K_FAT NUMBER := 1;
BEGIN
  FOR J IN 1..N LOOP
    N_FAT := N_FAT * J;
  END LOOP;
  FOR J IN 1..K LOOP
    K_FAT := K_FAT * J;
  END LOOP;
  FOR J IN 1..(N - K) LOOP
    N_K_FAT := N_K_FAT * J;
  END LOOP;
  RETURN (N_FAT / (N_K_FAT * K_FAT));
END;
```

With Result Cache

```
CREATE OR REPLACE FUNCTION C_N_K_CACHE (N IN NUMBER,
  K IN NUMBER) RETURN NUMBER RESULT_CACHE
IS
  N_FAT NUMBER := 1;
  K_FAT NUMBER := 1;
  N_K_FAT NUMBER := 1;
BEGIN
  FOR J IN 1..N LOOP
    N_FAT := N_FAT * J;
  END LOOP;
  FOR J IN 1..K LOOP
    K_FAT := K_FAT * J;
  END LOOP;
  FOR J IN 1..(N - K) LOOP
    N_K_FAT := N_K_FAT * J;
  END LOOP;
  RETURN (N_FAT / (N_K_FAT * K_FAT));
END;
```

Test those functions

```
CREATE OR REPLACE PROCEDURE STRESS(ANUM NUMBER)
IS
  AVAL NUMBER;
BEGIN
  FOR J IN 1..ANUM LOOP
    AVAL := C_N_K (50,10);
  END LOOP;
END;
/
```

```
CREATE OR REPLACE PROCEDURE STRESS_CACHE(ANUM NUMBER)
IS
  AVAL NUMBER;
BEGIN
  FOR J IN 1..ANUM LOOP
    AVAL := C_N_K_CACHE (50,10);
  END LOOP;
END;
/
```

Using Result Cache with tables

- The result cache can also be used for functions with a result based on the contents of infrequently updated tables.
- In this case, when we define a function we add the `RELIES ON` clause.

```
CREATE OR REPLACE FUNCTION FOO (APARAMETER NUMBER, …)
RETURN NUMBER RESULT_CACHE
RELIES ON (EMPLOYEES)
IS…
```

Using Virtual Columns

Virtual columns, is a new feature in Oracle Database 11g, to avoid the use of DML triggers, resulting in a performance gain in our applications.

An ordinary table

```
CREATE TABLE sh.LOANS (
  LOAN_ID INT NOT NULL,
  PAYMENT NUMBER,
  NUMBER_PAYMENTS NUMBER,
  GROSS_CAPITAL NUMBER);
```

Using triggers

```
CREATE OR REPLACE TRIGGER TR_LOANS_INS
  BEFORE UPDATE OR INSERT ON sh.LOANS
  FOR EACH ROW
BEGIN
  :new.GROSS_CAPITAL := :new.PAYMENT * :new.NUMBER_PAYMENTS;
END;
/
```

A table with a virtual column

```
CREATE TABLE sh.LOANS_VC (
  LOAN_ID INT NOT NULL,
  PAYMENT NUMBER,
  NUMBER_PAYMENTS NUMBER,
  GROSS_CAPITAL AS (PAYMENT * NUMBER_PAYMENTS));
```

To test the performance

```sql
SET TIMING ON
INSERT INTO sh.LOANS_VC (LOAN_ID, PAYMENT, NUMBER_PAYMENTS)
  SELECT
    ROWNUM, AMOUNT_SOLD, QUANTITY_SOLD
  FROM SALES;
SELECT COUNT(*)
  FROM sh.LOANS_VC
  WHERE GROSS_CAPITAL < 10000;
SET TIMING OFF
```

THE END

- Source code & documentation
- Back to Course Outline