To Cache or not to Cache; and How?

Presented by: John Jay King

Download this paper from: http://www.kingtraining.com

King Training Resources

Session Objectives

- Understand Oracle's SQL & PL/SQL caching features
- Choose caching that is appropriate to the task at hand
- Evaluate the performance implications of caching choices

Who Am I?

- John King – Partner, King Training Resources
- Oracle Ace Director
- Member Oak Table Network
- Providing training to Oracle and IT community for over 25 years – http://www.kingtraining.com
- "Techie" who knows Oracle, ADF, SQL, Java, and PL/SQL pretty well (along with many other topics)
- Member of AZORA, ODTUG, IOUG, and RMOUG

Arizona, USA

Who Are You?

- Application Developer
- DBA
- Business Analyst
- Other?

Oracle Caching Capabilities

- Deterministic Functions
- SQL Scalar Subqueries
- SQL Statement Cache
- PL/SQL Function Cache
- PL/SQL defined in SQL WITH clause
- PL/SQL PRAGMA UDF

LOB Buffer Cache Options

- BasicFile - Direct Read/Write avoids cache but can cause other I/O to wait
- SecureFile – Data may be compressed and encrypted; special shared LOB pool usually offers better performance than BasicFile
- NOCACHE - LOBs don't use buffer cache
- CACHE – LOB data uses buffer cache; best for LOBs with lots of READ/WRITE
- CACHE READS – LOBs use buffer cache only for READs; best for simple READs

SQL and PL/SQL Caching

- Deterministic Functions
- Scalar subquery caching
- SQL Statement Result Cache
- PL/SQL Statement Result Cache
- SQL WITH
- PRAGMA UDF

Deterministic Functions

- Produce identical (cached) results for given parameters/arguments each time called
- Depend solely upon the parameters/ arguments passed to them
- Do not use or modify the database or package variables

Note: Oracle cannot verify that a function is truly deterministic, using non-deterministic functions inappropriately yields “unpredictable” results

Deterministic Syntax

- DETERMINISTIC keyword is specified after return value type in CREATE FUNCTION (standalone function, function in CREATE PACKAGE or CREATE TYPE)

```
CREATE FUNCTION TIMES_2 (INVAL NUMBER)
  RETURN NUMBER
  DETERMINISTIC
IS
BEGIN
  RETURN INVAL * 2;
END;
```

Deterministic Requirement

- DETERMINISTIC is required for:
  - Functions used in a function-based index
  - Functions used in a materialized view with FAST REFRESH or ENABLE QUERY REWRITE

Deterministic Use Case

- Oracle does not require explicit declaration of DETERMINISTIC, but it is a good for some use cases:
  - Where required in Function-based indexes or Materialized Views (see previous page)
  - Functions in WHERE, ORDER BY, or GROUP BY clauses selecting rows for the result set
  - Functions that MAP or ORDER methods of a SQL type

Scalar Subquery Caching

- Performance of functions called as part of SQL may be improved by using a subquery

```
SELECT EMPLOYEE_ID,LAST_NAME,SALARY
      ,AVG_SAL_JOB(JOB_ID) AVG_JOB_SAL
    FROM HR.EMPLOYEES
      ORDER BY AVG_JOB_SAL,SALARY,LAST_NAME;
Elapsed: 00:00:00.182

SELECT EMPLOYEE_ID,LAST_NAME,SALARY,
   (SELECT AVG_SAL_JOB(JOB_ID) FROM DUAL) AVG_JOB_SAL
    FROM HR.EMPLOYEES
      ORDER BY AVG_JOB_SAL,SALARY,LAST_NAME;
Elapsed: 00:00:00.017
```

SQL Results Caching

- Caching is nothing new to Oracle; what's new is the caching of results…
- Once materialized, query results are cached and re-presented when the same query is run more than once
  (similar to how Materialized Views work, but more-dynamic)
- Oracle 11g "result_cache" hint asks Oracle to cache query results

Result Cache – Test Query

```
select cust_last_name || ', ' || cust_first_name cust_name
      ,cust_city
      ,prod_id
      ,count(*) nbr_sales
 from sh.customers cust
    join sh.sales sales
      on cust.cust_id = sales.cust_id
 where country_id = 52789
   and prod_id in (120,126)
 group by cust_last_name,cust_first_name,cust_city,prod_id
 having count(*) > 10
 order by cust_name,nbr_sales;
```

- This query was run three times in succession with timing turned on; resulting timings were
  - Elapsed: 00:00:00.67
  - Elapsed: 00:00:00.46
  - Elapsed: 00:00:00.37

Using Result Cache

```
select /*+ result_cache */ cust_last_name || ', ' || cust_first_name
   cust_name
      ,cust_city
      ,prod_id
      ,count(*) nbr_sales
 from sh.customers cust
    join sh.sales sales
      on cust.cust_id = sales.cust_id
 where country_id = 52789
   and prod_id in (120,126)
 group by cust_last_name,cust_first_name,cust_city,prod_id
 having count(*) > 10
 order by cust_name,nbr_sales;
```

- This query was run three times in succession with timing turned on; resulting timings were
  - Elapsed: 00:00:00.23 (results not cached yet)
  - Elapsed: 00:00:00.01
  - Elapsed: 00:00:00.03

SQL Result Cache Use Cases

- The SQL Result Cache trades long term storage of results for I/O
- Probably best used when a query is rerun repeatedly with the same input parameters AND the result set is not large AND the data is mostly static
- Probably not a good idea if the results of the query are large unless the query is run infrequently but with the same inputs

Result Cache Parameters

- result_cache_mode - Manual/Force (default is 'manual' requiring explicit hint)
- result_cache_max_size – Size allocated from the shared pool but maintained separately (not flushed with shared pool)
- result_cache_max_result – Highest percentage of result cache that may be used by a single result set (def. 5%)
- result_cache_remote_expiration - Number of minutes result cache resultset based upon a remote object is considered valid

Catalog Support

- V_$RESULT_CACHE_DEPENDENCY
  Dependencies in result cache results
- V_$RESULT_CACHE_MEMORY
  Result cache memory block statistics
- V_$RESULT_CACHE_OBJECTS
  Object (& attributes) in result cache results
- V_$RESULT_CACHE_STATISTICS
  Result cache memory use statistics
- V$CLIENT_RESULT_CACHE_STATS
  Cache settings and memory statistics

PL/SQL Result Cache

- PL/SQL allows specification of a result_cache for function calls
- Add the new "result_cache" clause just before the "AS/IS" keyword in the definition (Oracle 11g R1 also used now-obsolete "relies_on" clause)
- The results of a call to the Function with a specific set of input parameters is stored (cached) for later re-use

PL/SQL Result Cache - Code

```
CREATE OR REPLACE FUNCTION RESULT_CACHE_ON
   (in_cust_id sh.customers.cust_id%type,  in_prod_id
   sh.sales.prod_id%type)
RETURN number
RESULT_CACHE -- RELIES_ON (SH.CUSTOMERS, SH.SALES)
authid definer
AS
 sales number(7,0);
BEGIN
select count(*) nbr_sales  into sales
 from sh.customers cust join sh.sales sales
      on cust.cust_id = sales.cust_id
 where cust.cust_id = in_cust_id
  and  prod_id = in_prod_id;
 return sales;
EXCEPTION
  when no_data_found then return 0;
END RESULT_CACHE_ON;
```

PL/SQL Function Cache Results

- Results of running the function cache three times; note the first execution builds the cached results

```
Elapsed: 00:00:00.136

Elapsed: 00:00:00.001

Elapsed: 00:00:00.002
```

PL/SQL Result Cache Uses

- PL/SQL Function Resuts Cache is best used:
  - When same function called repeatedly with the same input values
  - If function is called frequently with different values but the result is small
  - If function is called with a small set of different inputs and the result is large
  - If data returned is relatively static

PL/SQL in WITH

- Oracle 12c allows definition of PL/SQL Functions and Procedures using SQL's Common Table Expression (WITH)
  - Defining PL/SQL locally reduces SQL-PL/SQL context-switching costs
  - Local PL/SQL overrides stored PL/SQL with the same name
  - Local PL/SQL is not stored in the database
  - Local PL/SQL is part of the same source code as the SQL that uses it
  - PL/SQL Result Cache no use in Local PL/SQL

Example PL/SQL in WITH

```
with function times_42(inval number)
 return number
as
begin
  return inval * 42;
end;
select channel_id,count(*) nbr_rows,
       sum(quantity_sold) qtysold,
       sum(times_42(cust_id)) cust42
  from sh.sales
  group by channel_id
  order by channel_id
/
```

- Use /*+ WITH_PLSQL */ to place WITH in subquery

PL/SQL in WITH Uses

- The advantage to local PL/SQL defined in SQL WITH is the reduction in context switching as SQL switches back-and-forth between SQL and PL/SQL
- Best use for local PL/SQL is stand-alone functions like calculations or string manipulations – make sure benefits are worth the complexity if the function will be duplicated in multiple SQLs
- Advantage of local PL/SQL is forfeit if the function issues other PL/SQL calls

PL/SQL UDF

- Oracle 12c allows functions to be defined using “PRAGMA UDF” to specify that a function will be used in SELECTS (behaving similar to function in WITH)
- This optimizes code for use within a SELECT or other SQL
  (Probably not a good option for functions also used from PL/SQL !)
- Does not have code duplication drawbacks of WITH clause

Example PL/SQL UDF

```
create or replace function times_42(inval number)
 return number
as
  pragma udf;
begin
  return inval * 42;
end;
/
```

Comparison

	1st Run	2nd Run	3rd Run
Function in WITH	0.854	0.825	0.929
Compiled Function in database	2.018	1.945	1.928
Compiled UDF Function in database	0.667	0.602	0.664

- Clearly, there are savings to be had
  - If the PL/SQL in question calls other PL/SQL, then, WITH and UDF might not be the best choice
  - If a PL/SQL function will be called from PL/SQL, UDF may cause performance to be off since the optimization will be incorrect

King Training Resources

Making wise choices

- Make functions that qualify Deterministic
- Use scalar subquery when SQL function calls may be cached
- SQL results cache is best used for frequent execution of same query
- PL/SQL results cache is best used for repeated function execution with same inputs
- Use PRAGMA UDF and PL/SQL in WITH for self-contained PL/SQL primarily used in SQL (UDF probably more-maintainable)

King Training Resources

Wrapping it Up

- Oracle provides many CACHEing features designed to improve performance
- SQL Query and PL/SQL Function statement caches reduce the cost of repetitive executions (at the cost of memory)
- SQL using PL/SQL in WITH and PL/SQL UDF functions reduce the cost of repetitive executions of PL/SQL used in SQL statements

www.rmoug.org


COLLABORATE17
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY
Save the Date for
COLLABORATE 17
We'll see you again in Vegas
WELCOME
TO The Fabulous
LAS VEGAS
NEVADA
APRIL 2 - 6, 2017 | MANDALAY BAY RESORT & CASINO

Please Complete Session Evaluations

To Cache or not to Cache; and How?

To contact the author:

John King

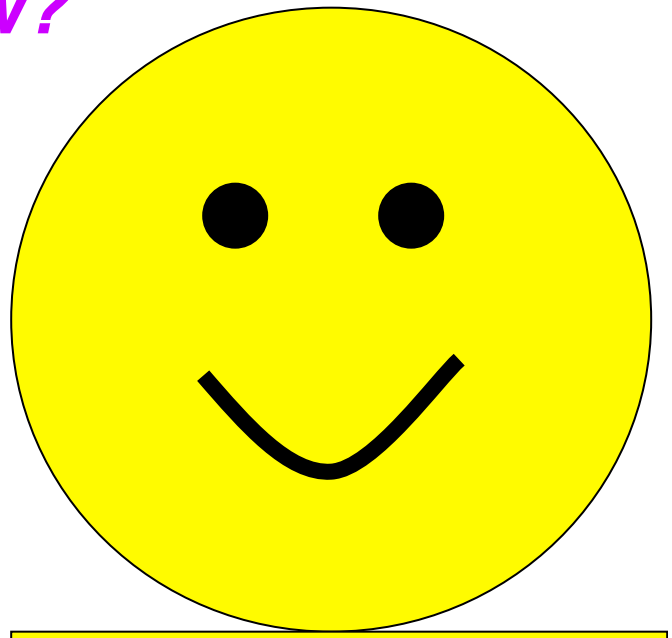
King Training Resources

P. O. Box 1780

Scottsdale, AZ 85252 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

Today's slides and examples are on the web:

http://www.kingtraining.com

- End