The Amazing and Elegant PL/SQL Function Result Cache

Steven Feuerstein
Oracle Developer Advocate for PL/SQL
Oracle Corporation

Email: steven.feuerstein@oracle.com
Twitter: @sfonplsql
Blog: stevenfeuersteinonplsql.blogspot.com
YouTube: Practically Perfect PL/SQL

ORACLE

Just in case I live in the future, even for a moment....

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

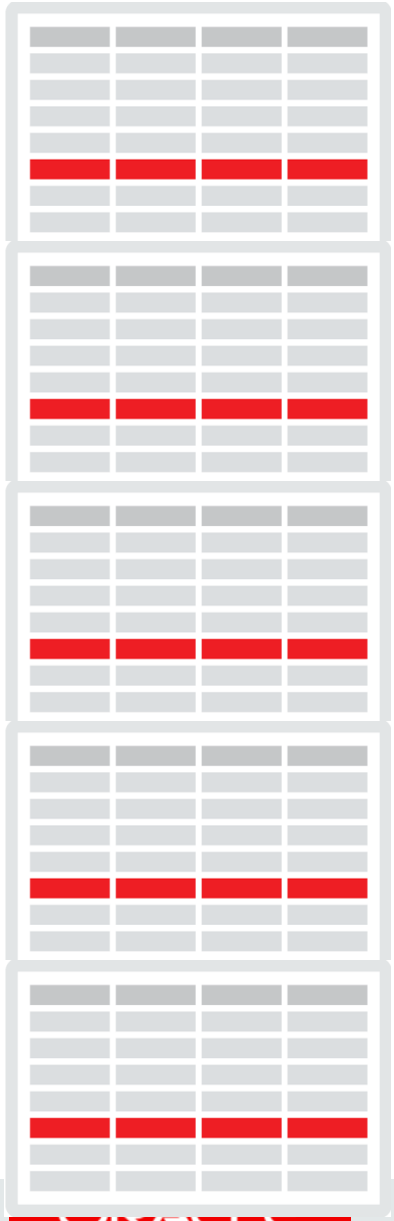
Most referenced code is available in my demo.zip file from the PL/SQL Learning Library: oracle.com/oll/plsql or direct download from **http://v.gd/05JIWC**

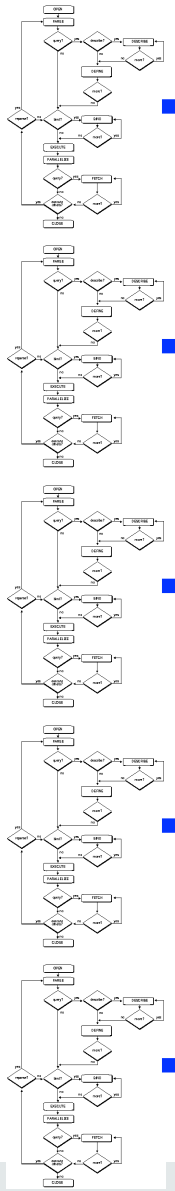
ORACLE

Resources for PL/SQL developers

- oracle.com/plsql – official home of PL/SQL
- oracle.com/oll – Oracle Learning Library
  - Download demo.zip file with all scripts **http://v.gd/05JIWC**
- plsqlchallenge.oracle.com - weekly PL/SQL quizzes, and more
- asktom.oracle.com – 'nuff said
- livesql.oracle.com – script repository and 12/7 12c database
- oracle-developer.net - great content from Adrian Billington
- oracle-base.com - great content from Tim Hall

What's wrong with this picture?





```
SELECT * FROM materialized_view
WHERE pky = 12345;
```

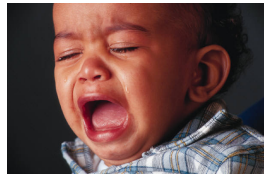
```
SELECT * FROM materialized_view
WHERE pky = 12345;
```

```
SELECT * FROM materialized_view
WHERE pky = 12345;
```

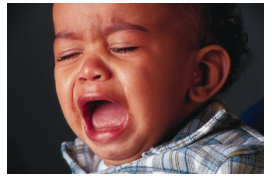
```
SELECT * FROM materialized_view
WHERE pky = 12345;
```

```
SELECT * FROM materialized_view
WHERE pky = 12345;
```

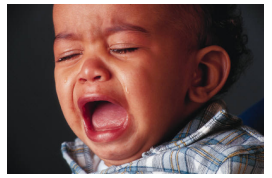
What's Taking So Long?



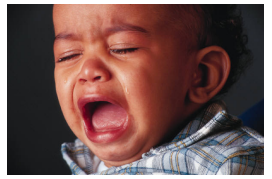
What's Taking So Long?



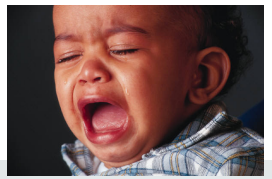
What's Taking So Long?



What's Taking So Long?



What's Taking So Long?



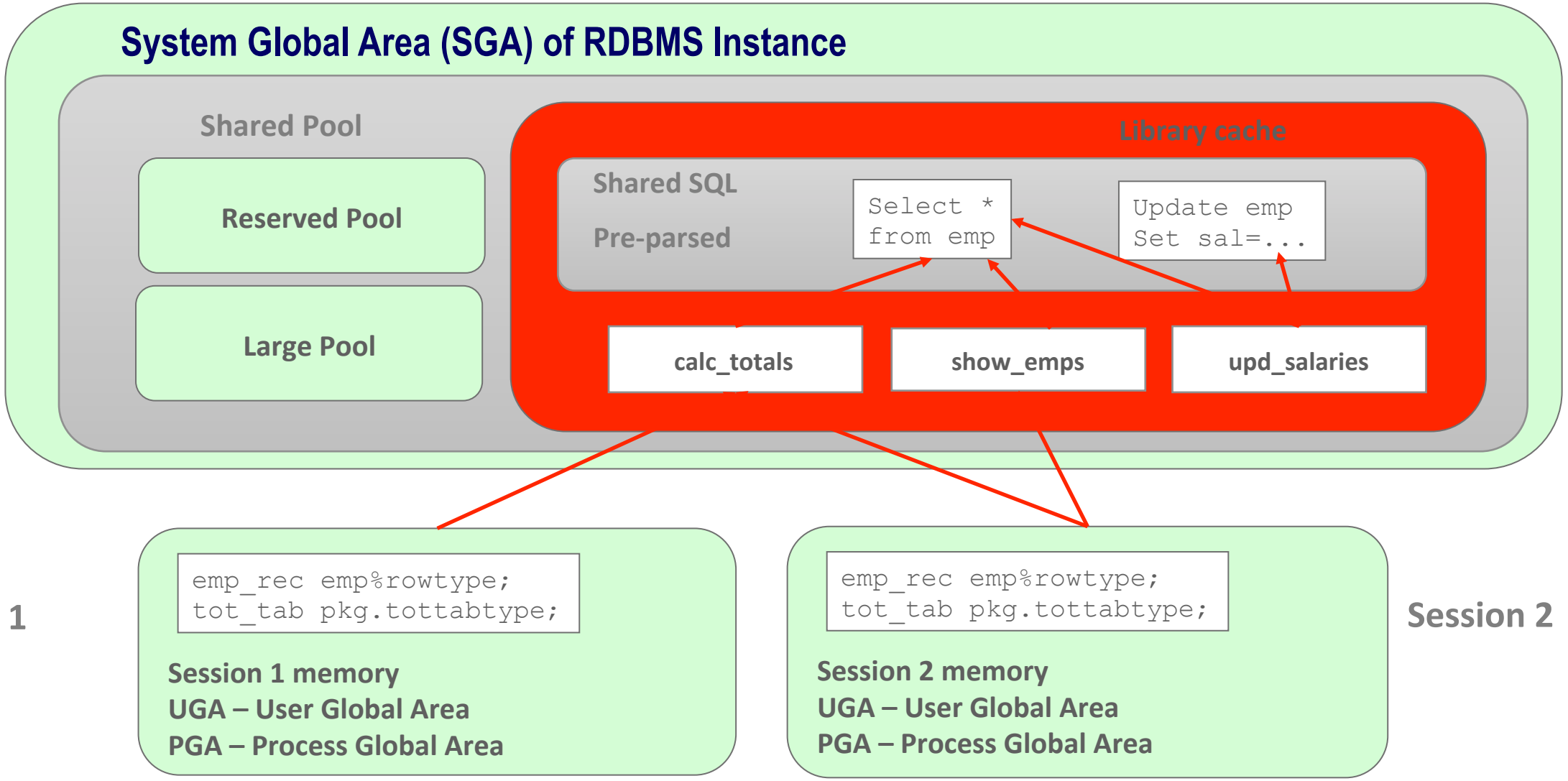
Data Caching Options in PL/SQL

- Caching is a time-honored technique for improving performance.
- Store data that doesn't change for some period of time in a location that can be accessed more quickly than the *source*.
- The SGA is an enormous, complex cache for the entire database instance.
- But there are other caches we can leverage in our PL/SQL code.
  - Deterministic functions
  - PGA caching using package-level variables
  - The function result cache: the most powerful and widely applicable technique

All referenced code available in my demo.zip file from the PL/SQL Learning Library: oracle.com/oll/plsql.

ORACLE

PL/SQL Runtime Memory Architecture



How PL/SQL uses the SGA, PGA and UGA

```
PACKAGE Pkg is                                    /* 11g feature! */
  Nonstatic_Constant CONSTANT PLS_INTEGER := My_Sequence.Nextval;
  Static_Constant    CONSTANT PLS_INTEGER := 42;
END Pkg;
```

- The **SGA** contains information that can be shared across sessions connected to the instance.
  - In PL/SQL, this is limited to package static constants.
- The **User Global Area** contains session-specific data that persists across server call boundaries
  - Package-level data
- The **Process Global Area** contains session-specific data that is released when the current server call terminates: "local" data.

Data Caching Options

- Functions declared as DETERMINISTIC
- PGA caching
  - Used most effectively with collections
  - Accessing PGA memory generally more efficient than SGA, especially if executing SQL.
- Oracle Database 11g Function Result Cache
  - The best caching technique and the most important new feature in 11g for PL/SQL developers.

DETERMINSTIC Functions

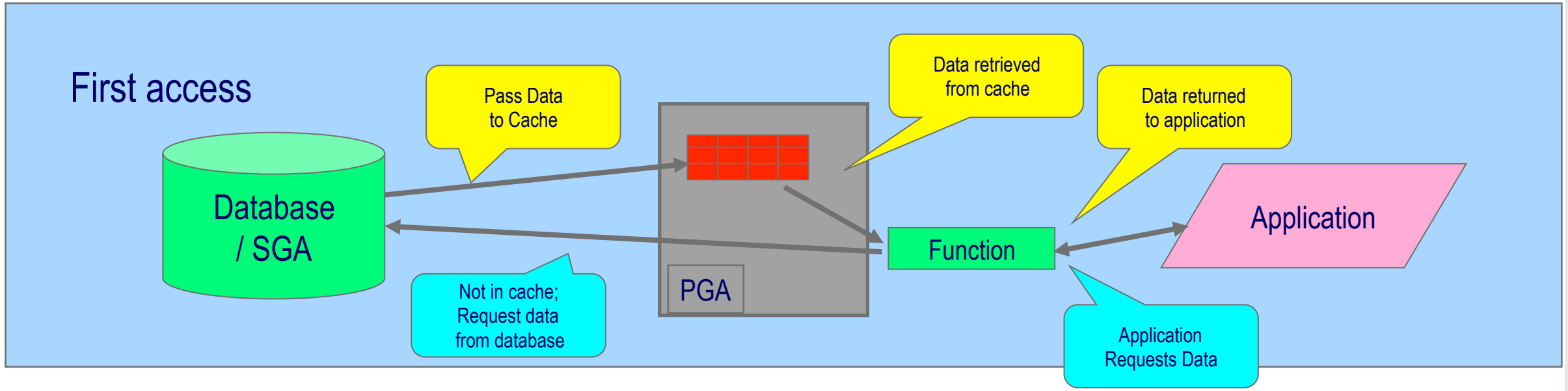
- A function is deterministic if the value it returns is *determined* completely by its inputs (IN arguments).
  - In other words, no *side effects*.
- Add the DETERMINISTIC keyword to your function to enable optimizations:
  - Function-based indexes
  - Cache results *in scope of a query*
- Don't lie!
  - Oracle will not (at compile time) reject your use of the keyword, even if it isn't true.

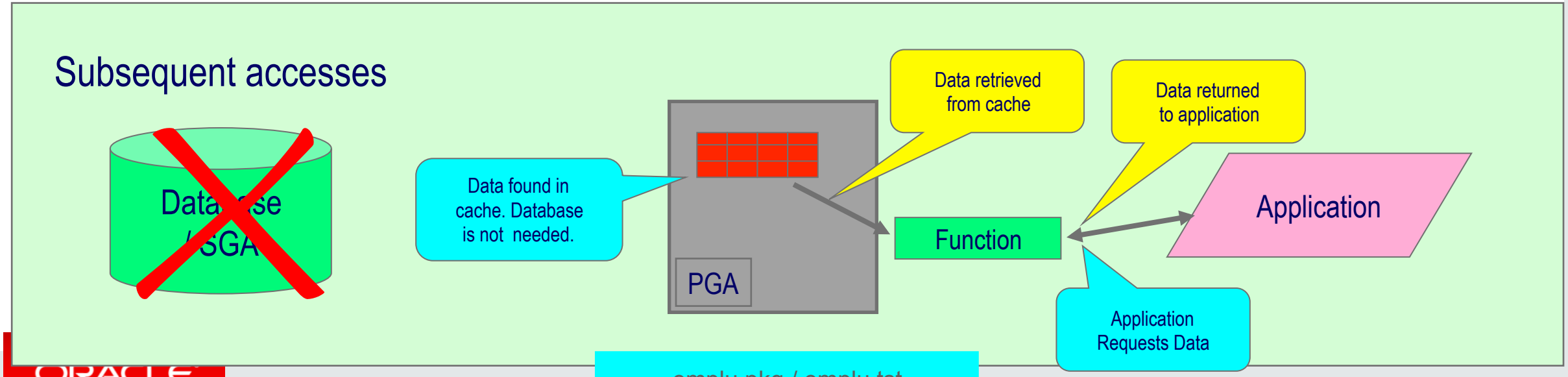
deterministic.sql
deterministic_in_plsql.sql

PGA-Based Caching

- When you declare variables at the package level, their state persists in your session.
  - A PGA-based cache, specific to each session.
- And if you declare a collection at the package level, you can cache multiple rows of data.
  - Very useful for static datasets like materialized views.
- Not a reliable technique for Web-based (usually stateless) applications

Avoiding Unnecessary SGA Lookups





emplu.pkg / emplu.tst

PGA Caching: Things to keep in mind

- Must use package-level data so that it persists.
  - Memory is consumed by the PGA and so is multiplied for all users of the application.
  - Not a reliable technique for stateless application (Internet)
- Very difficult to share cache across sessions in the same instance.
  - One possibility involves DBMS_PIPE.
- Very difficult to update the cache once the data source is changed.
  - Especially by/from, *other* sessions. Possible to use DBMS_ALERT.
- Useful under specific scenarios....
  - Small, static dataset or a single or small number of batch processes

syscache.pkg

The Function Result Cache

- Introduced in Oracle Database 11g, the Function Result Cache is definitely the caching mechanism of choice for PL/SQL developers.
- This cache is stored in the SGA ; shared across sessions; purged of dirty data automatically
- You can and should use it to retrieve data from any table that is *queried more frequently than updated*.
  - Static datasets like materialized views
  - Same rows fetched multiple times (parameter values serve as unique index into cache)
- Enterprise Edition only.

How the Function Result Cache Works

- Add the RESULT_CACHE clause to your function's header.
- When a call is made to function, Oracle compares IN argument values to the cache.
- If no match, the function is executed and the inputs and return data are cached.
- If a match is found, the function is not executed; cached data is returned.
- If changes to a "relies on" table are committed, the cache is marked invalid and will be re-built.

ORACLE

11g_frc_demo.sql

Minimal Impact on Code with Result Cache

```
CREATE OR REPLACE PACKAGE emplu11g
IS
   FUNCTION onerow (employee_id_in IN employees.employee_id%TYPE)
      RETURN employees%ROWTYPE
      RESULT_CACHE;
END emplu11g;

CREATE OR REPLACE PACKAGE BODY emplu11g
IS
   FUNCTION onerow (employee_id_in IN employees.employee_id%TYPE)
      RETURN employees%ROWTYPE
      RESULT_CACHE RELIES_ON (employees)
   IS
      ....
   END onerow;
END emplu11g;
```

- Add RESULT_CACHE keyword to header of function in both specification and body.
- RELIES_ON clause is deprecated in 11.2. Oracle will automatically determine all tables on which the function relies. RELIES_ON is then *ignored*.

Performance Impact of Result Cache

- The result cache is stored in the SGA.
- So we should expect it be slower than a PGA-based cache.
- But accessing result cache data does not require going through the SQL engine.
- So it should be much faster than executing a query.
  - Even if the statement is parsed and the data blocks are already in the SGA.
- Let's find out!

11g_emplu*.*

Result Cache – Things to Keep in Mind - 1

- If you have uncommitted changes in your session, dependent caches are ignored.
  - The cache will *not* override your own changed data.
- Caching is not performed for complex types: records with CLOBs, collections, etc.
  - But Oracle is optimistic!
- The cache is *not* related to SQL statements in your function.
  - It only keeps track of the input values and the RETURN clause data.

11g_frc_demo.sql

Result Cache – Things to Keep in Mind - 2

- You cannot use the result cache with invoker rights program units until 12.1.
  - Bypass execution of function body, Oracle cannot resolve references to objects - the *whole point* of IR.
- Functions with session-specific dependencies must be "result-cached" with great care.
  - Virtual private database configurations
  - References to SYSDATE, reliance on NLS_DATE_FORMAT, time zone changes
  - Application contexts (calls to SYS_CONTEXT)
- Solution: move all dependencies into parameter list.

11g_frc_vpd.sql
11g_frc_vpd2.sql

Managing the Result Cache

- Oracle offers a number of ways to manage the result cache and tune it to your specific application needs:
- RESULT_CACHE_MAX_SIZE initialization parameter
  - If the cache is too small, then the LRU algorithm negates the point of the cache.
- DBMS_RESULT_CACHE management package
- v$RESULT_CACHE_* performance views

show_frc_dependencies.sp

Fine Grained Dependencies in 11.2

- Oracle keeps track of table dependencies on a per-result level.
  - Each result cached could have a different set of dependencies.
- A change to a table could invalidate just a subset of the results in the cache.
  - It's *not* all or nothing - when your function's different logic paths could "hit" different tables.

11g_frc_dependencies.sql
11g_frc_dependencies2.sql

Make It Easy on Yourself

- I hope you will agree that the result cache is a great feature.
- But how easy will it be for you to apply it?
- If you write/duplicate queries throughout your code, upgrading will be expensive and slow.
- If you hide your queries behind functions, you have a single point of definition, and you can upgrade "instantly."

11g_frc_encapsulation.sql

Conclusions - Caching

- Oracle offers several different ways you can build upon its own caching.
- DETERMINISTIC for functions in SQL
  - Mostly useful for organizing code more effectively
- PGA caching
  - Very fast, but of limited use
- Function result cache
  - Simplest and most widely applicable technique
  - Get ready for it *now* by hiding queries inside functions.

Hardware and Software
Engineered to Work Together


ORACLE®