Shareable Cursors

ACEs@Home – Episode 2

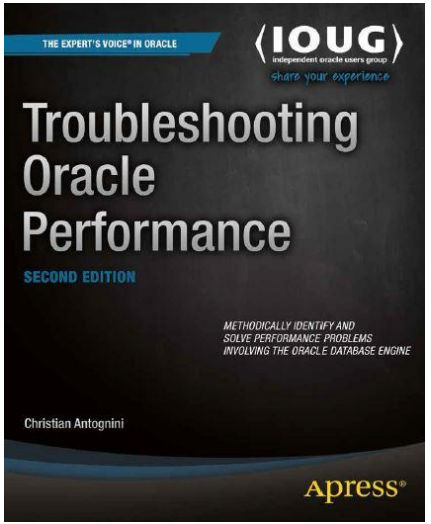
Christian Antognini

ChrisAntognini

antognini.ch

@ChrisAntognini

- Senior principal consultant, trainer and partner at Trivadis
  - christian.antognini@trivadis.com
- Focus: get the most out of database engines
  - Logical and physical database design
  - Query optimizer
  - Application performance management
- Author of *Troubleshooting Oracle Performance* (Apress 2008/14)
- OakTable Network, Oracle ACE Director



Contents

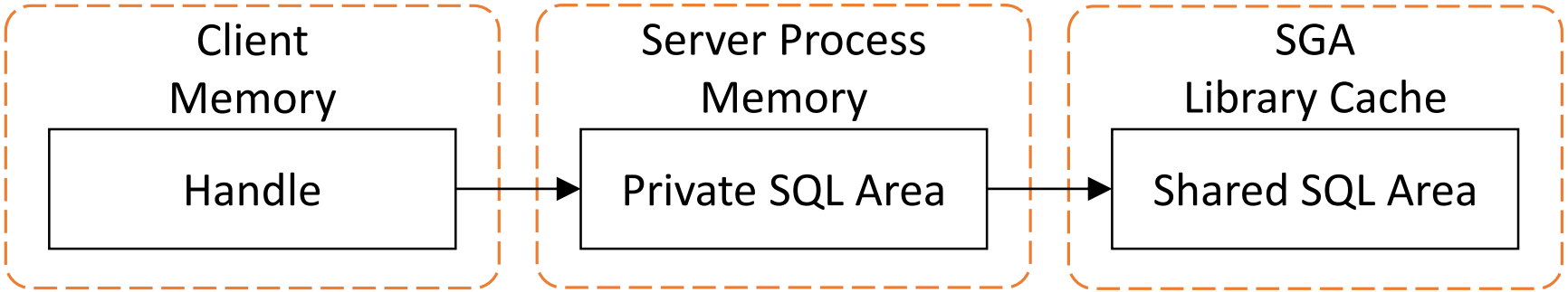
trivadis

- What Is a Cursor?
- Private and Shared SQL Areas
- Parent and Child Cursors
- Bind Variable Graduation
- Bind Variable Peeking
- Adaptive Cursor Sharing

What Is a Cursor?

trivadis

- A cursor, and its associated structures, is created when an application issues a parse call
- But, what is a cursor?
  - It is a handle to a *private SQL area* with an associated *shared SQL area*



Private and Shared SQL Areas

trivadis

- What does a private SQL area contain?
  - Data such as bind variable values and query execution state information
  - It is session specific and, therefore, it is stored in the UGA
- What does a shared SQL area contain?
  - The parse tree and the execution plan of the associated SQL statement
  - It is stored in the library cache
- In practice, the terms *cursor* and *private/shared* SQL area are used interchangeably
- The shared SQL areas contain the so-called *parent* and *child* cursors

Parent Cursors

trivadis

- The key information stored in a parent cursor is the text of the SQL statement on which it is based
- Several SQL statements share the same parent cursor if their text is identical
  - An exception is when CURSOR_SHARING != EXACT
- V$SQLAREA returns one row for each parent cursor
  - The SQL_ID column identifies a parent cursor

Child Cursors

trivadis

- The key information stored in a child cursor is the execution plan and the execution environment related to it
- Several SQL statements are able to share the same child cursor only if
  - they share the same parent cursor, and
  - their execution environments are compatible
- V$SQL returns one row for each child cursor
  - In general, the SQL_ID and CHILD_NUMBER columns identify a child cursor
- V$SQL_SHARED_CURSOR shows why a child cursor was created

10187168 – Enhancement to Obsolete Parent Cursors if VERSION_COUNT Exceeds a Threshold

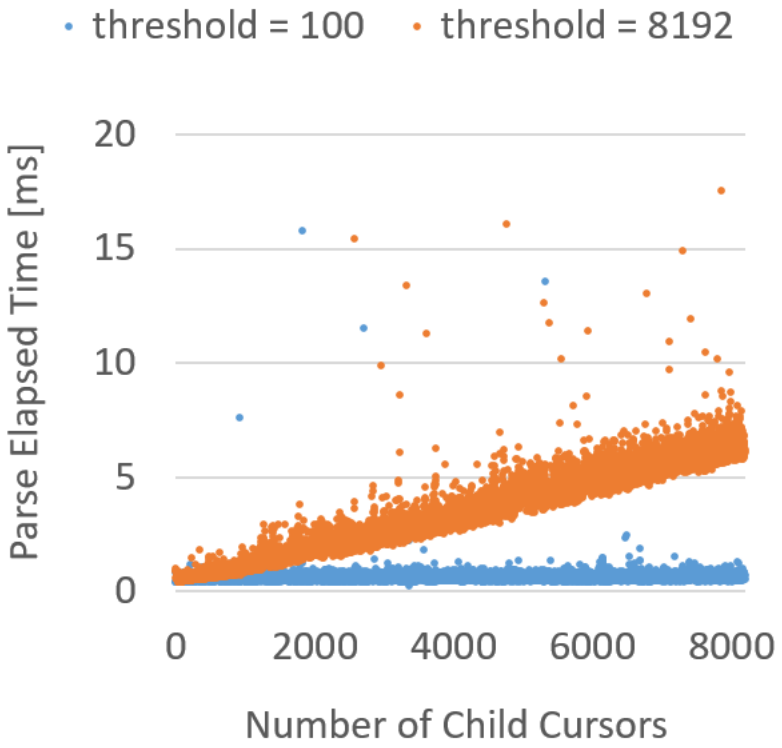
trivadis

- Child cursors associated to a parent cursor with a number of child cursors that exceeds _CURSOR_OBSOLETE_THRESHOLD are obsoleted
  - Default value: 11.2.0.3 = 100, 11.2.0.4 → 12.1.0.2 = 1024, 12.2.0.1+ = 8192
- To identify child cursors the ADDRESS column might be necessary
- DBMS_XPLAN cannot display obsoleted cursors

Too Many Child Cursors

trivadis

- Applications that “generate” many child cursors per parent cursor can experience performance issues after upgrading to 12.2+
  - E.g. waits related to mutexes
- 2431353.1 suggests to decrease _CURSOR_OBSOLETE_THRESHOLD



Does the Parse Time Increase Linearly with the Number Of Child Cursors?

Bind Variables

trivadis

- Bind variables impact applications in two ways:
  - They make programming either easier or more difficult
  - They introduce, from a performance point-of-view, both an advantage and a disadvantage
- Advantage
  - Sharing of cursors in the library cache (avoid hard parses)
- Disadvantage
  - Crucial information is hidden from the query optimizer

Bind Variable Graduation

trivadis

- For sharing a child cursor, the type and the size of bind variables matter
- To relax the limitation related to the size, *bind variable graduation* is used to part bind variables into four groups:
  - Up to 32 bytes
  - 33 – 128 bytes
  - 129 – 2,000 bytes
  - More than 2,000 bytes
- V$SQL_BIND_METADATA shows the maximum length of the group, not of the bind variable itself

Bind Variable Peeking

trivadis

- To address the disadvantage related to bind variables, the SQL engine uses *bind variable peeking*
- The SQL engine is able to peek at the value of bind variables
- The problem with this approach is that the generated execution plan depends on the values provided by the first execution
- Limitation
  - SQL statements executed through DBMS_SQL cannot take advantage of bind variable peeking (bug 13386678)

Adaptive Cursor Sharing

trivadis

- To address the problem introduced by bind variable peeking, the SQL engine uses *adaptive cursor sharing* (a.k.a. *bind-aware cursor sharing*)
- Its purpose is to automatically recognize when the reutilization of an already available child cursor leads to inefficient executions
- When an inefficiency is detected, bind awareness is enabled
  - The child cursor experiencing suboptimal performance is invalidated
  - A new, bind-aware child cursor is created
- To force bind awareness, use the BIND_AWARE hint or set at the session or system level _OPTIMIZER_ADAPTIVE_CURSOR_SHARING to FALSE

Adaptive Cursor Sharing – Limitations

trivadis

- It is not designed to work with many bind variables (1983132.1)
- It can only take place during parse calls
- If an application keeps a cursor open, the SQL engine won't be able to generate a new child cursor
  - Also applied to SQL statements executed by the PL/SQL engine!
- SQL statements executed through DBMS_SQL cannot take advantage of adaptive cursor sharing

Summary

trivadis

- To share a parent cursor, the text of the SQL statement must match the text of the cursor
- To share a child cursor, many criteria have to be fulfilled
- Bind variables are *not* always good
- Keeping cursors open is *not* always good
- For SQL statements that are parsed more than one time, keep in mind that potentially every parse call could lead to a different execution plan

Q&A

Making a **WORLD** possible in which **intelligent IT** facilitates **LIFE** and **WORK** as a **matter** of **course.**