# Hacking Oracle's Memory
# About Internals & Troubleshooting

Stefan Koehler

# About me

## Stefan Koehler

- Independent Oracle performance consultant and researcher
- 13+ years using Oracle RDBMS - Independent since 2011
- Oracle performance and internals geek
- Main interests: Cost based optimizer and Oracle RDBMS internals

## Services: “All about performance & troubleshooting”

- Oracle performance tuning (e.g. Application, CBO, Database, Design, SQL)
- Oracle core internals researching (e.g. DTrace, GDB, Perf, etc.)
- Troubleshooting nontrivial Oracle RDBMS issues (e.g. Heap dumps, System state dumps, etc.)
- Services are mainly based on short-term contracting

www.soocs.de contact@soocs.de @OracleSK

# Agenda

- X$ tables - A window into Oracle's memory structure

- SGA (System Global Area)
  - SGA memory structure overview
  - Granules
  - Buffer pool / DB_CACHE
  - Shared pool implementation and 12c enhancement

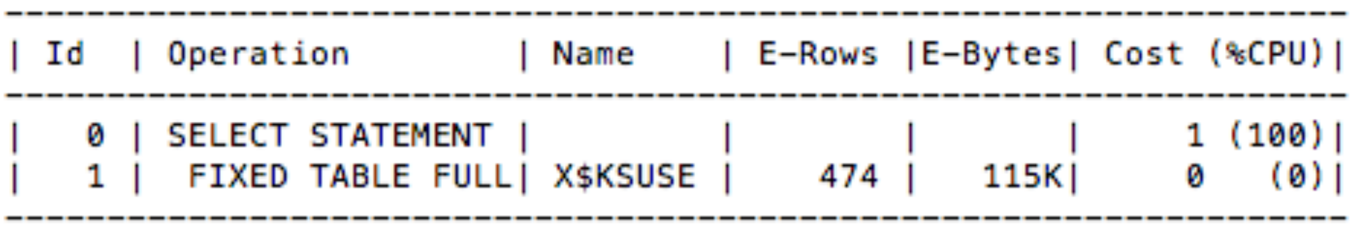
- PGA (Program Global Area)
  - Analyze PGA memory usage on Oracle / SQL level
  - Capture and source PGA memory allocations

**Disclaimer:** Almost everything is based on research and testing. Test it yourself – with your release and operating system – always! Do not trust anybody! ☺

# X$ tables - A window into Oracle’s memory structure

- Queries on X$ tables read from C memory structure via fixed table row-source function in execution plan, parse the data and display the results in tabular form

Example of X$KSUSE (V$SESSION)

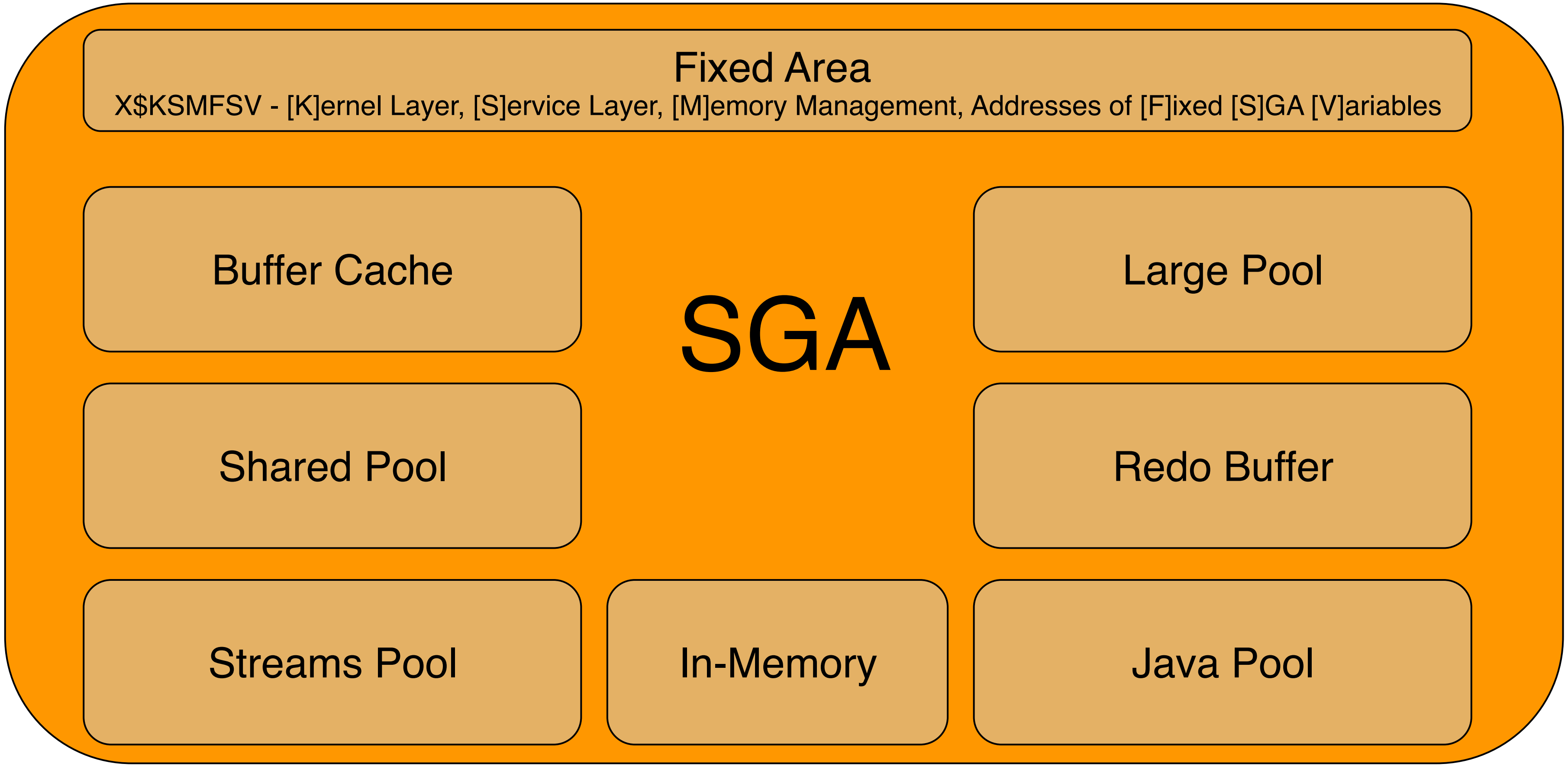


```
-----------------------------------------------------------------------
| Id  | Operation         | Name     | E-Rows |E-Bytes| Cost (%CPU)|
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT |          |        |       |     1 (100)|
|   1 |  FIXED TABLE FULL| X$KSUSE |    474 |   115K|     0   (0)|
-----------------------------------------------------------------------
```

- X$ tables (e.g. X$KCCLE - V$LOG) may rely on helper functions (e.g. reading from control file) which copy the needed data into memory first before the common X$ table processing kicks in
- X$ table name derivation - MOS ID #22241.1 (Google it)
- Be aware that running queries on X$ tables may result in heavy latch contention (e.g. X$KSMSP - shared pool latch)

# SGA memory structure overview

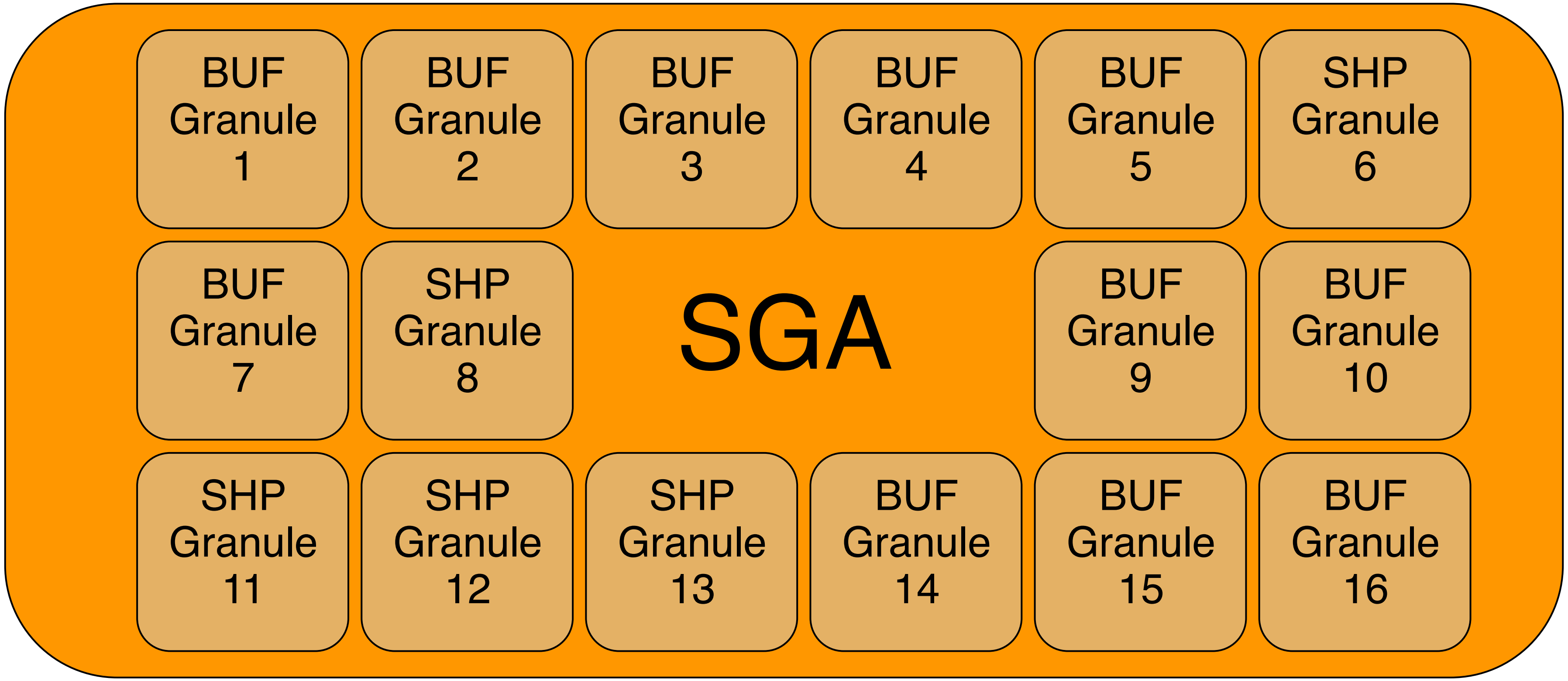
- SGA consists of different components


Fixed Area
X$KSMFSV - [K]ernel Layer, [S]ervice Layer, [M]emory Management, Addresses of [F]ixed [S]GA [V]ariables
Buffer Cache
SGA
Large Pool
Shared Pool
Redo Buffer
Streams Pool
In-Memory
Java Pool

- Implemented as System V shared memory segments on OS disregarding the combination of AMM and Linux

# SGA - Granules

- SGA has been re-engineered in Oracle 9i to relocate memory between areas (e.g. buffer cache & shared pool) in an easy way



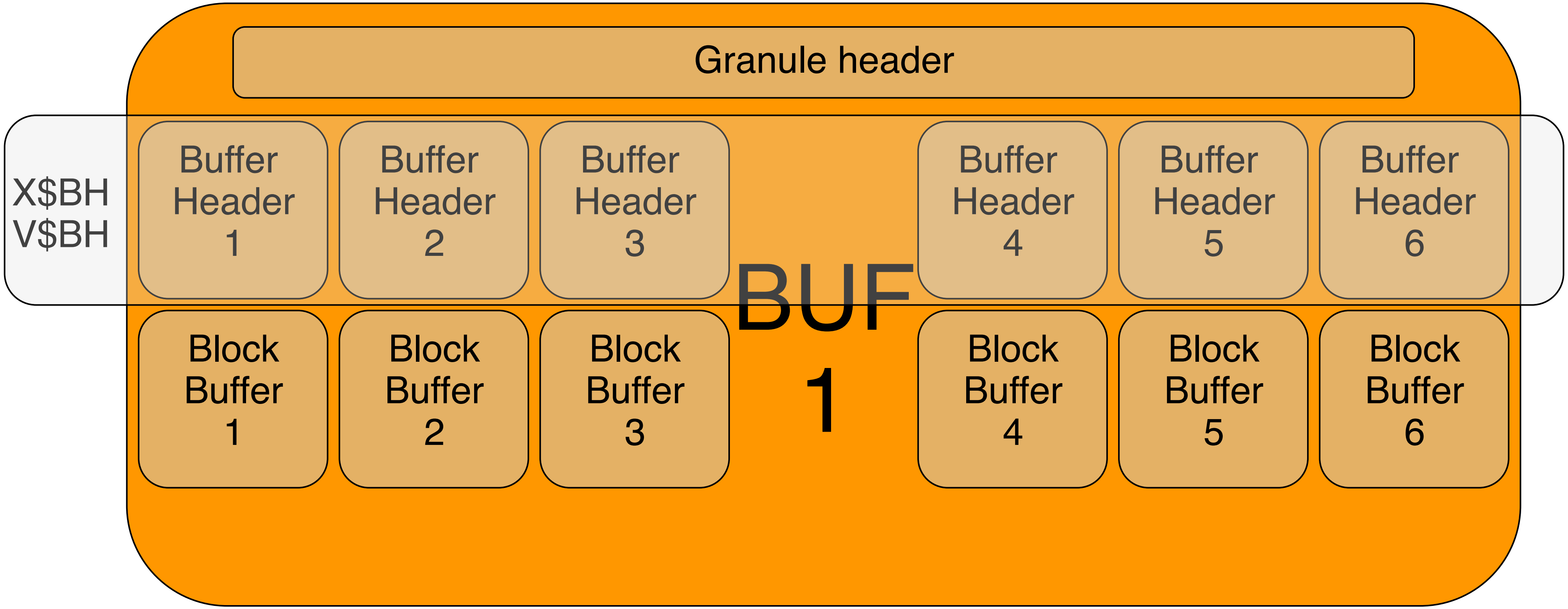
| | | | | | |
|---|---|---|---|---|---|
| BUF Granule 1 | BUF Granule 2 | BUF Granule 3 | BUF Granule 4 | BUF Granule 5 | SHP Granule 6 |
| BUF Granule 7 | SHP Granule 8 | SGA | | BUF Granule 9 | BUF Granule 10 |
| SHP Granule 11 | SHP Granule 12 | SHP Granule 13 | BUF Granule 14 | BUF Granule 15 | BUF Granule 16 |

- Granule size varies based on OS, Oracle version and SGA size (e.g. MOS ID #947152.1)

DEMO

# SGA - Buffer pool / DB_CACHE

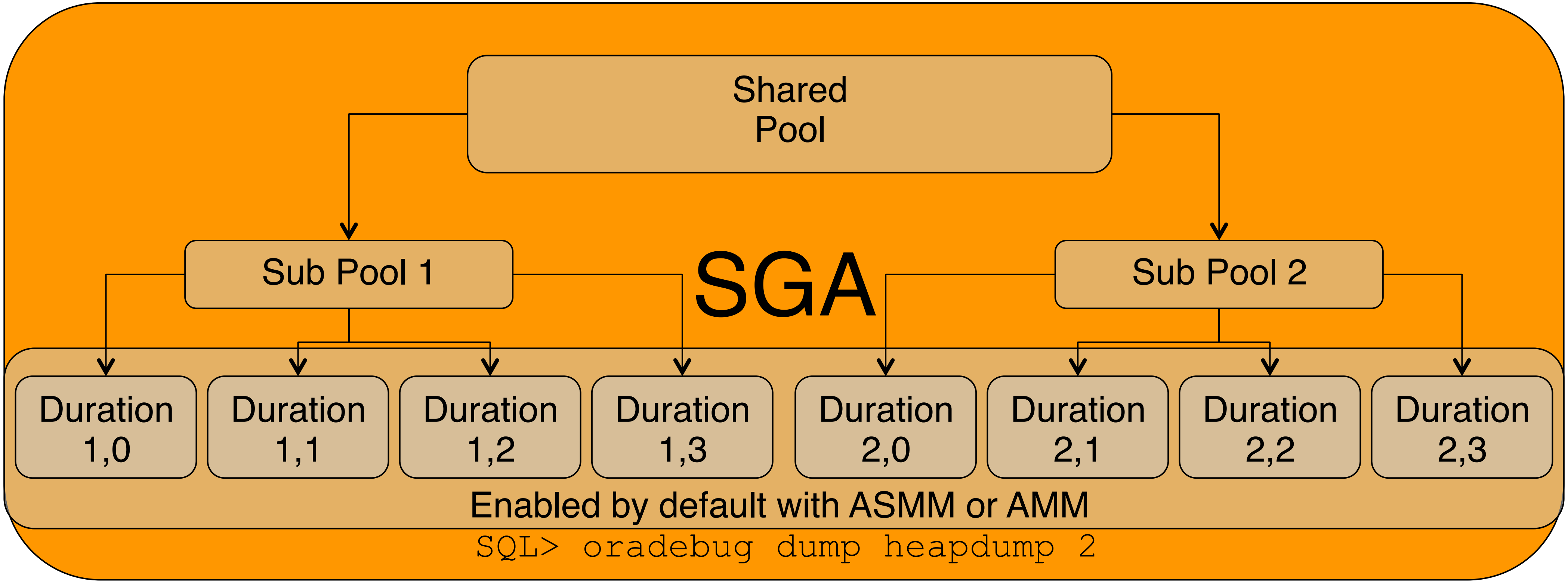
- The following is an illustration of one buffer pool granule



DEMO

# SGA - Shared pool implementation (1)

- Shared pool structure since Oracle 10g R2

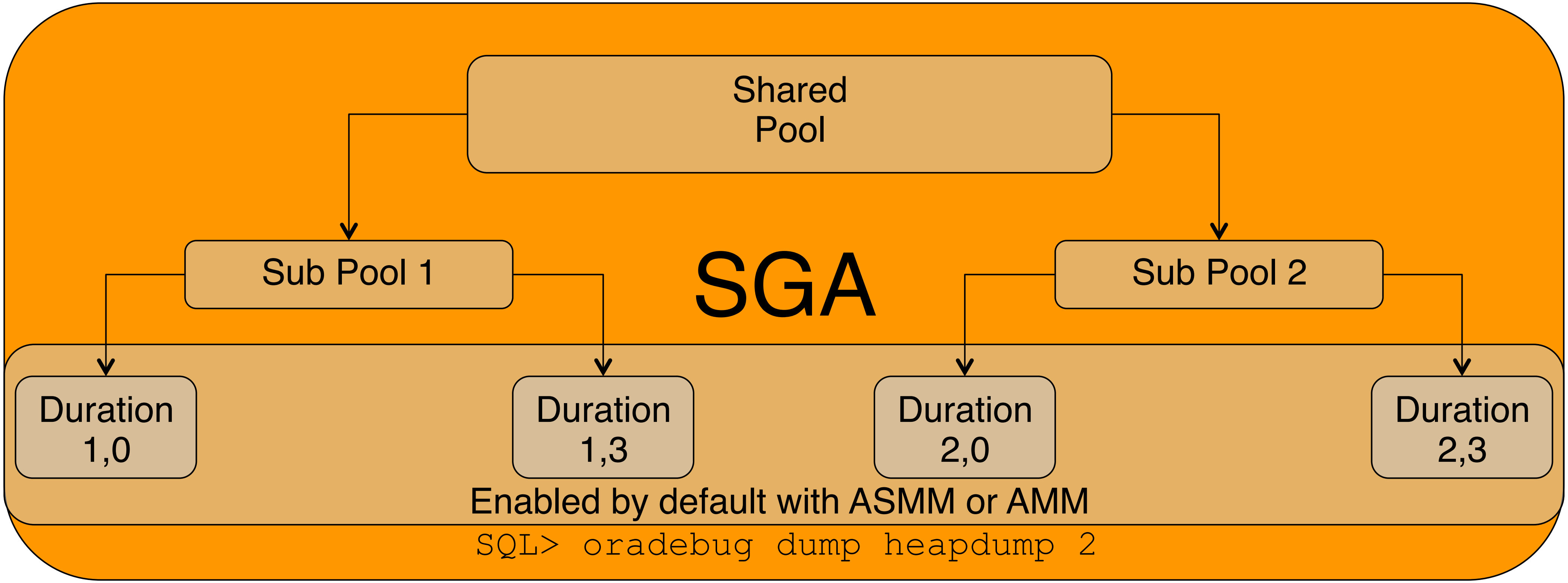


```
SQL> oradebug dump heapdump 2
```

- Shared pool is split into sub pools and durations due to latching scalability (one latch per sub-pool) and memory fragmentation
- Each duration allocates at least one granule

# SGA - Shared pool implementation (2)

- Shared pool structure since Oracle 12c



Enabled by default with ASMM or AMM

```
SQL> oradebug dump heapdump 2
```

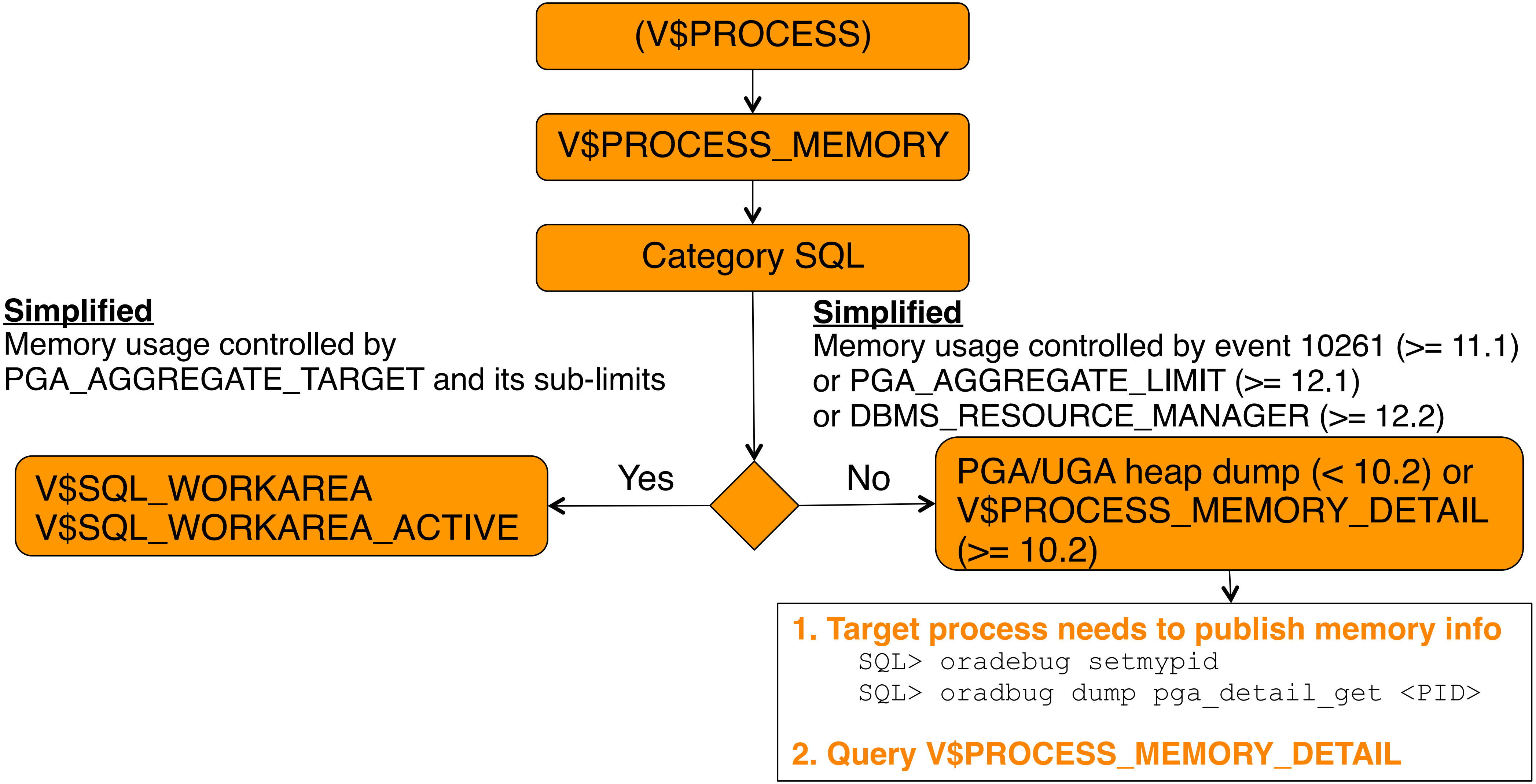
```
ORA-04031: unable to allocate 352 bytes of shared memory
("shared pool","unknown object","sga heap(2,0)","krsdicle")
```

- Group shared pool durations in 2 groups for better share-ability of memory and to avoid ORA-4031 (e.g. MOS ID #1675470.1)

# PGA - Analyze PGA memory usage on Oracle / SQL level

DEMO

- Approach for troubleshooting high PGA memory usage



(V$PROCESS)

V$PROCESS_MEMORY

Category SQL

**Simplified**
Memory usage controlled by PGA_AGGREGATE_TARGET and its sub-limits

**Simplified**
Memory usage controlled by event 10261 (>= 11.1)
or PGA_AGGREGATE_LIMIT (>= 12.1)
or DBMS_RESOURCE_MANAGER (>= 12.2)

Yes

V$SQL_WORKAREA
V$SQL_WORKAREA_ACTIVE

No

PGA/UGA heap dump (< 10.2) or V$PROCESS_MEMORY_DETAIL (>= 10.2)

**1. Target process needs to publish memory info**

```
    SQL> oradebug setmypid
    SQL> oradbug dump pga_detail_get <PID>
```

**2. Query V$PROCESS_MEMORY_DETAIL**

# PGA - Capture and source PGA memory allocations

- **Exemplary scenario:** *You run a PL/SQL application and notice a continuous increase in PGA memory. You check the view V$PROCESS_MEMORY_DETAIL and notice an increase in category "Other".*
- **Related questions:** *Which part of the possibly complex PL/SQL code is causing these memory allocations? Is this a memory leak in my custom code or a possible Oracle bug?*

Oracle is "just" a C program that allocates heap memory for PGA memory requests through specific C functions (kghal*).

```
SQL> oradebug doc component
…
Components in library GENERIC:
 --------------------------
   KGH                 KGH Memory Allocator (kgh)
…
```

# Questions and answers

Download links and further information about all mentioned tools and procedures can be found on website www.soocs.de/public/talk/

🏠 www.soocs.de ✉ contact@soocs.de @OracleSK