Oracle 数据库性能优化与运维最佳实践

优化

- 优化方法论
- SQL优化
- 实例优化

运维

- 定制运维工具

优化方法论

系统运行缓慢可能的原因

- 数据库
- 应用
- 主机配置
- 网络环境
- 存储配置与状态
- 部署方式

谁进行优化

- 数据库
- 应用
- 主机配置
- 网络环境
- 存储配置与状态
- 部署方式

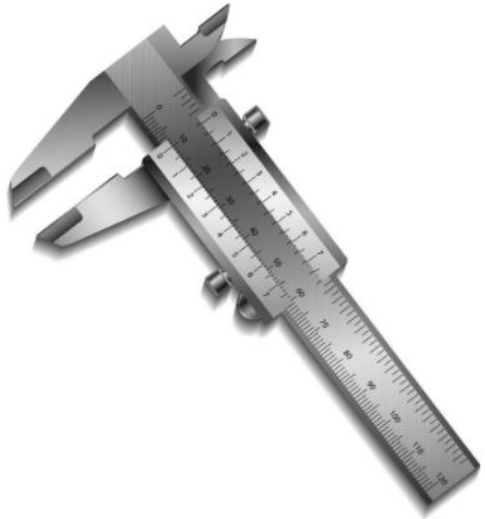
- 数据库管理员
- 应用架构师/开发人员
- 系统管理员
- 网络管理员
- 存储管理员
- 系统架构师

进行优化的范围

- 性能优化范围：
  - 应用程序：
    - SQL 语句性能
    - 更改管理

    DBA与应用相关人员共享
  - 实例优化：DBA
    - 内存
    - 数据库结构
    - 实例配置
  - 其他系统交互：
    - 存储I/O
    - 网络

    DBA与其他管理员共享



有效的优化目标

- 具备下列特征：
  - 具体化
  - 可量化
  - 可实现

优化的指导思想

- 自上而下优化以下内容：
  - 在优化应用程序代码之前先优化设计
  - 在优化实例之前先优化代码
- 对可以带来最大潜在好处的方面进行优化，并确定：
  - 最长的等待
  - 最慢的SQL
- 达到目标时停止优化。

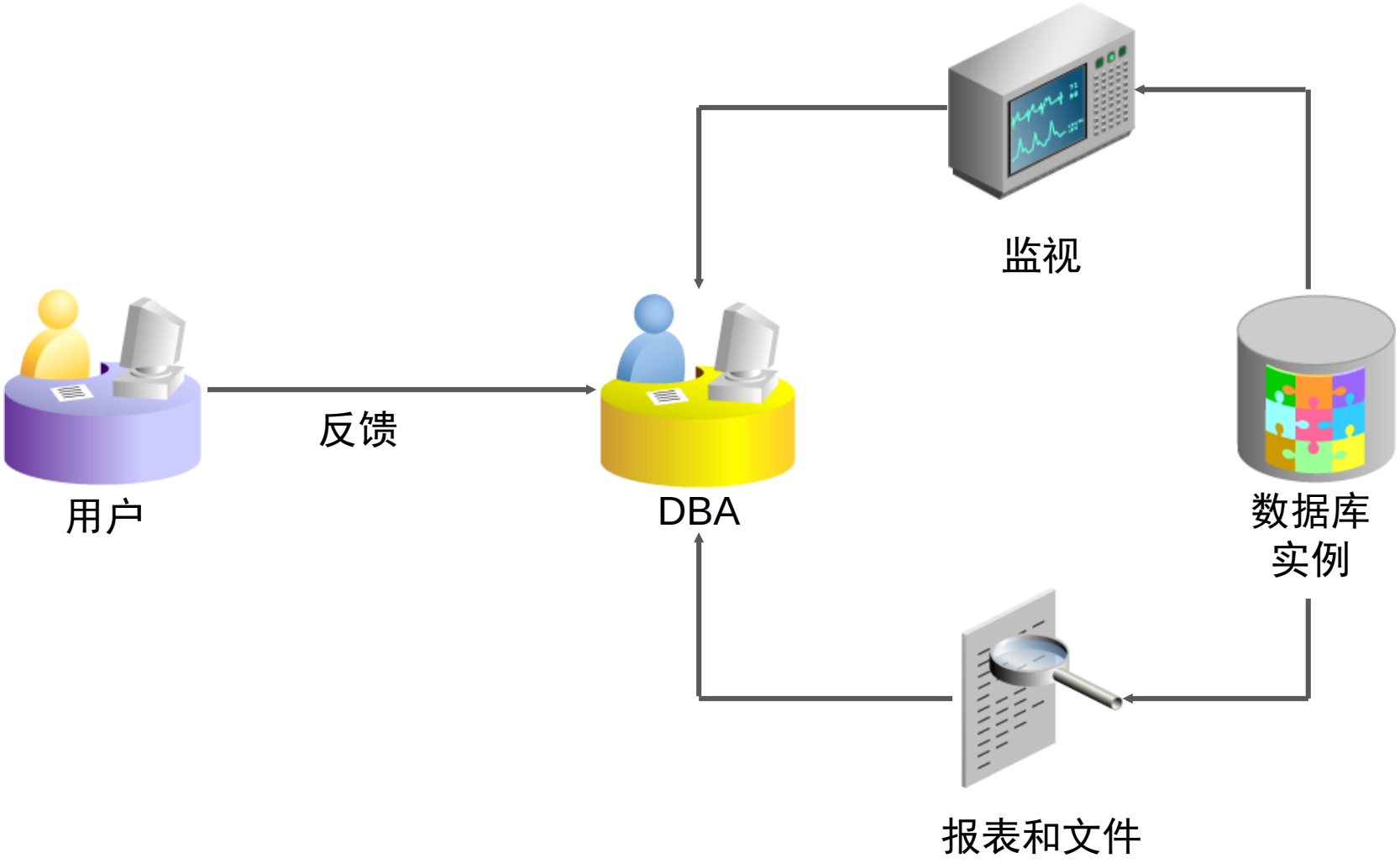
优化过程

- 优化的过程都是相同的：
  1. 定义问题并陈述目标。
  2. 收集当前性能统计信息。
  3. 考虑一些常见的性能错误。
  4. 制定试用解决方案。
  5. 实施并度量更改。
  6. 决定：“该解决方案是否达到目标？”
     - 否？转到步骤 3 并重复相关过程。
     - 是？完成。

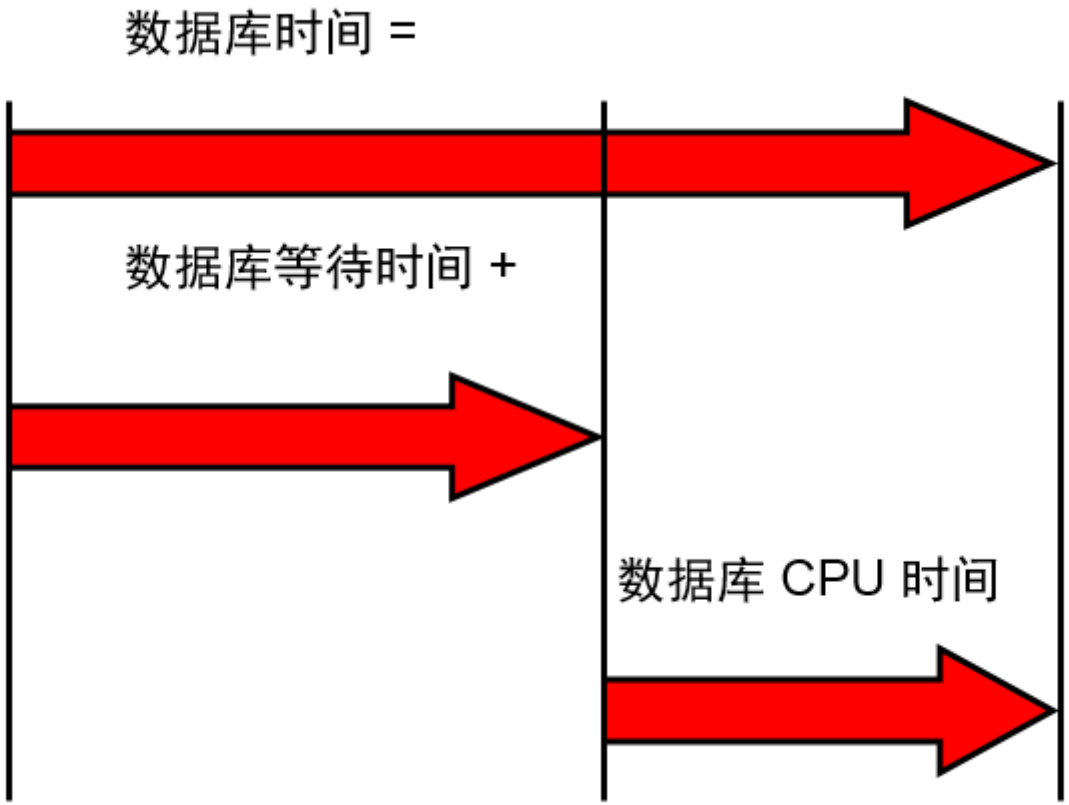
优化的实践方法

- 监视和诊断
  - 使用基于 AWR 的工具确定问题
- SQL 优化
- 实例优化

定义问题

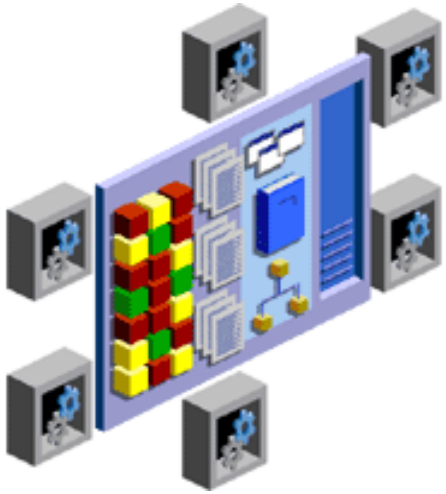


数据库时间



限制范围

- 问题出在何处？
  - 应用程序 (SQL)
  - 实例



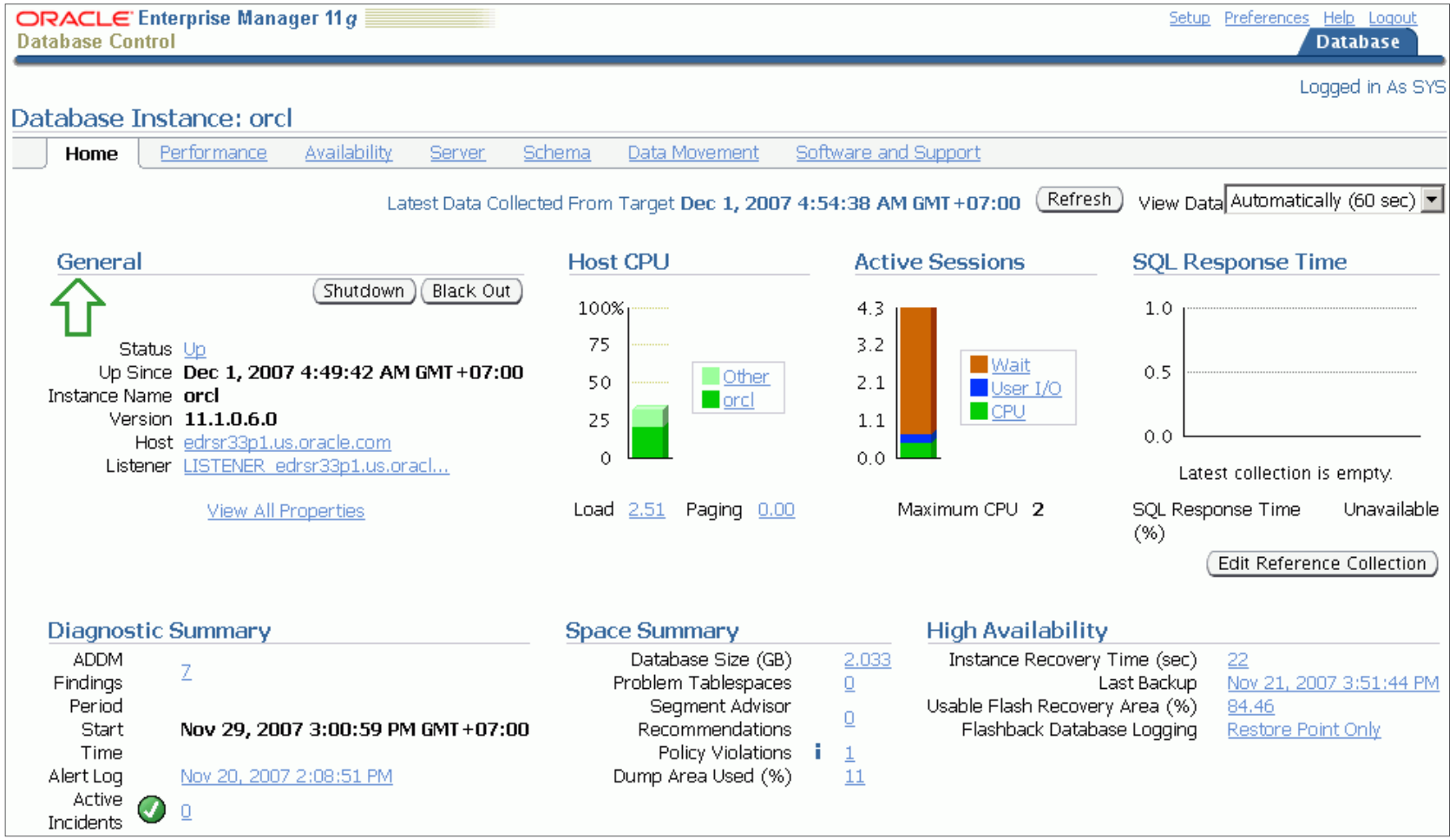
设置优先级

- 选择影响最大的问题：
  - 通过完成工作（CPU 时间或服务时间）与等待工作所用时间（等待时间）的对比来分析系统性能。
  - 确定占用时间最长的组件。
  - 细化以优化该组件。

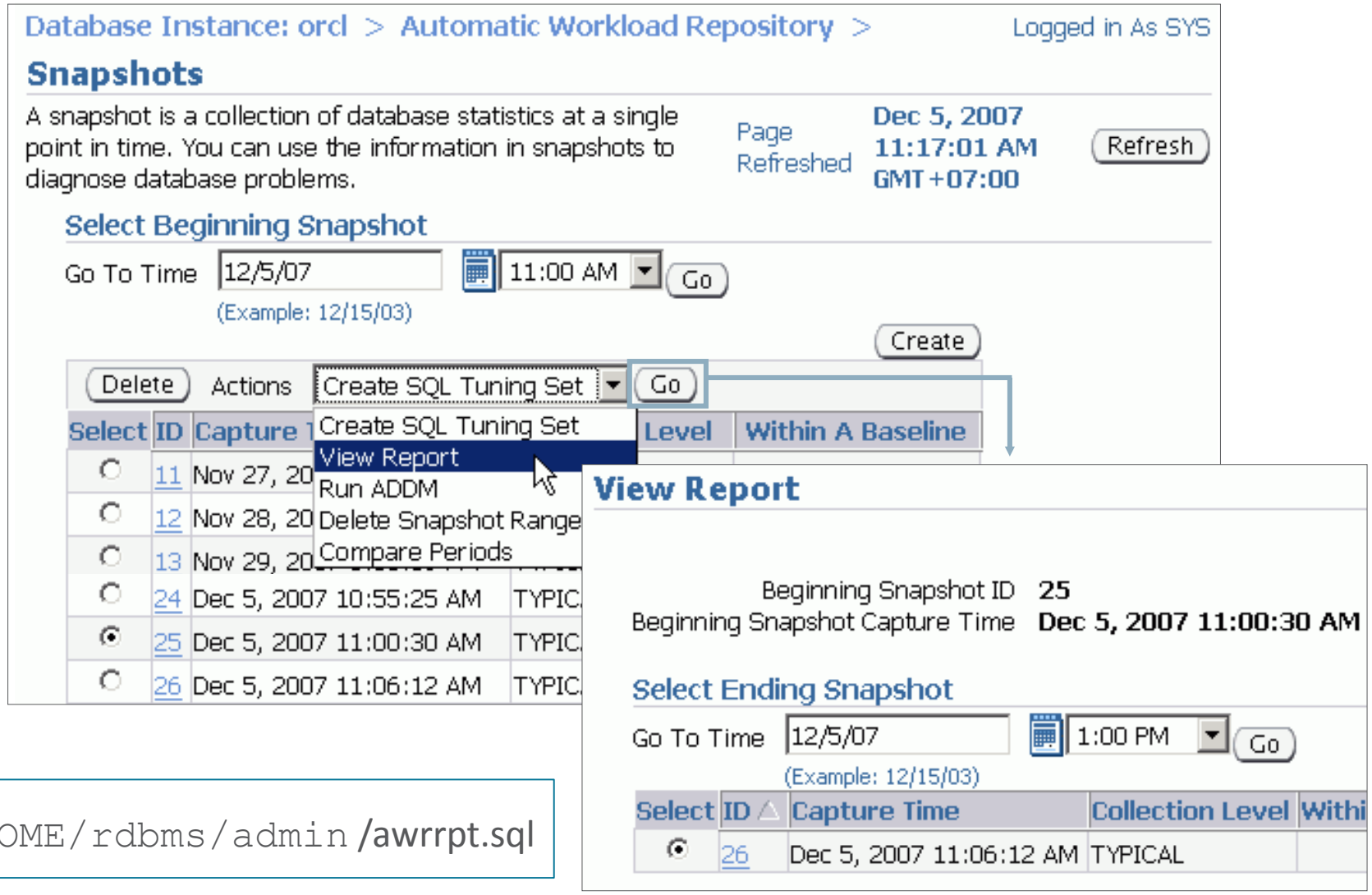
使用基于 AWR 的工具确定问题

访问数据库EM主页

Oracle Enterprise Manager 性能页



在 EM 中生成 AWR 报表



执行ORACLE_HOME/rdbms/admin/awrrpt.sql

生成 的AWR 报表

Database Instance: orcl > Automatic Workload Repository > Snapshots > Logged in As SYS

Snapshot Details

View ADDM Run

Details | Report

Save to File

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
ORCL	1166921678	orcl	1	05-Dec-07 10:12	11.1.0.6.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
edrsr33p1.us.oracle.com	Linux IA (32-bit)	2	2	1	1.97

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	25	05-Dec-07 11:00:30	26	.8
End Snap:	26	05-Dec-07 11:06:12	42	4.3
Elapsed:		5.69 (mins)		
DB Time:		29.13 (mins)		

阅读 AWR 报表

- 第一部分提供
  - 概览
  - 最重要的诊断

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
ORCL	1166921678	orcl	1	05-Dec-07 10:12	11.1.0.6.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
edrsr33p1.us.oracle.com	Linux IA (32-bit)	2	2	1	1.97

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	25	05-Dec-07 11:00:30	26	.8
End Snap:	26	05-Dec-07 11:06:12	42	4.3
Elapsed:		5.69 (mins)		
DB Time:		29.13 (mins)		

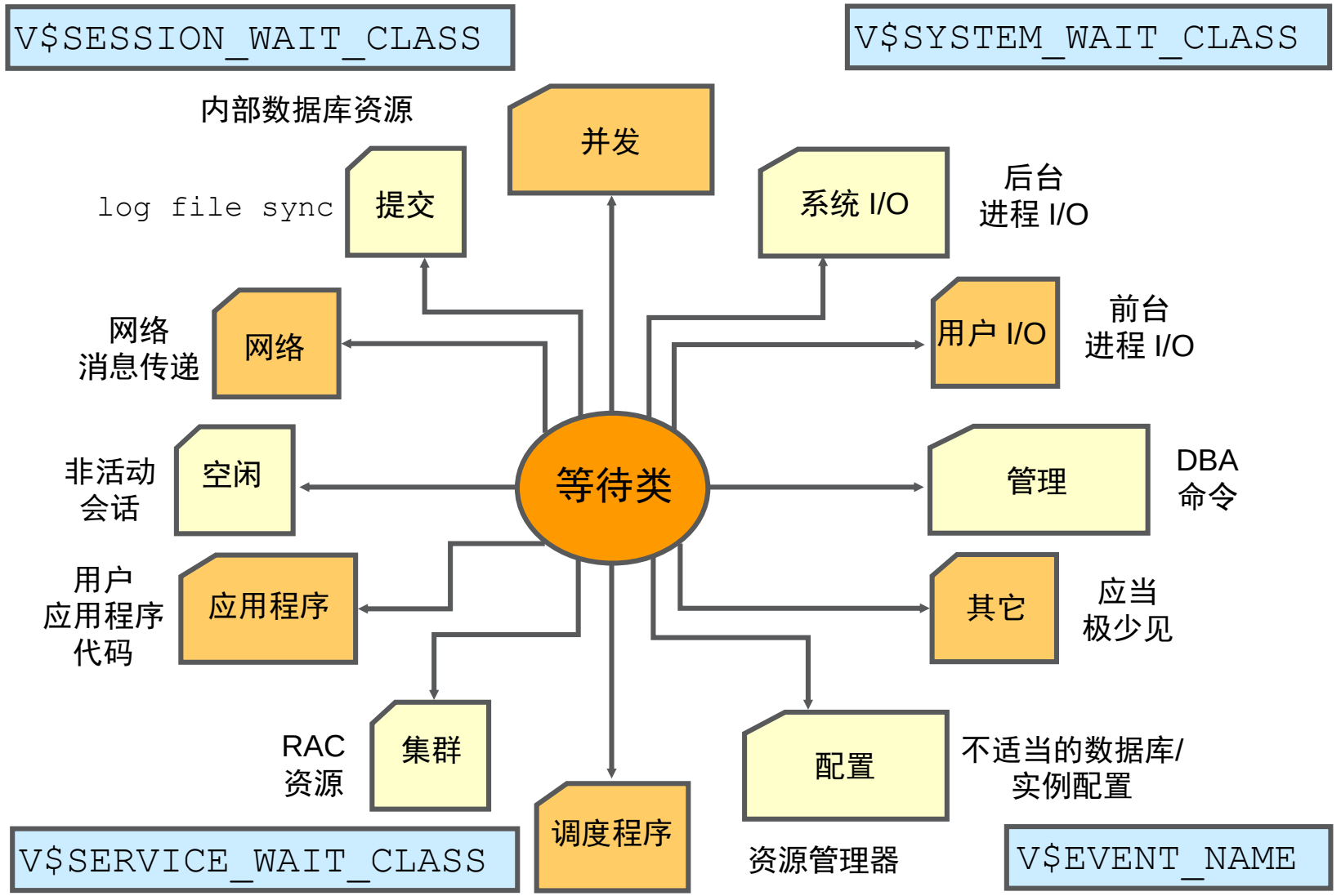
阅读 AWR 报表

– 等待事件提供由于各种原因不得不等待或必须等待的会话的有关信息。

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
free buffer waits	688,778	8,239	12	52.80	Configuration
buffer busy waits	67,632	3,817	56	24.46	Concurrency
enq: TX - index contention	18,705	1,475	79	9.45	Concurrency
log file sync	11,306	860	76	5.51	Commit
db file sequential read	180,952	552	3	3.53	User I/O

等待类



常见等待事件

等待事件	区域	可能的原因	检查
buffer busy waits	缓冲区高速缓存、DBWR	取决于缓冲区类型。PK 索引和序列	出现问题时的 V$SESSION_WAIT（块）
free buffer waits	缓冲区高速缓存、DBWR I/O	DBWR 速度慢	使用 OS 统计信息时的入时间缓冲区高速缓存统计信息
db file scattered read、db file sequential read	I/O、SQL 优化	SQL 优化不当，I/O 系统速度慢	V$SQLAREA 磁盘读取数。V$FILESTAT 读取时间
Enqueue waits (enq:)	锁	取决于入队类型	V$ENQUEUE_STAT
Library cache waits	闩锁	SQL 分析/共享	V$SQLAREA 分析调用，子游标
log buffer space	日志缓冲区 I/O	缓冲区小，I/O 速度慢	V$SYSSTAT 重做缓冲区分配重试次数，磁盘
Log file sync	过度提交、I/O	I/O 速度慢，未批量提交	V$SYSSTAT 视图中的提交数和回退数，磁盘数

AWR案例分析

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
log file sync	362,909	74,975	207	52.60	Commit
db file sequential read	6,237,273	32,588	5	22.86	User I/O
DB CPU		11,313		7.94	
read by other session	276,912	1,232	4	0.86	User I/O
buffer busy waits	503,152	787	2	0.55	Concurrency

AWR案例分析

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		474		44.85	
db file sequential read	2,003	5	3	0.48	User I/O
buffer busy waits	17	1	69	0.11	Concurrency
direct path read	105,669	0	0	0.02	User I/O
db file scattered read	67	0	2	0.01	User I/O

Host CPU (CPUs: 2 Cores: 2 Sockets: 1)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
1.94	2.10	64.0	1.8	0.3	34.2

Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
65.0	98.8	0.0

AWR案例分析

时间模型统计信息

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	1,050.06	99.34
DB CPU	474.04	44.85
parse time elapsed	10.82	1.02
hard parse elapsed time	10.70	1.01
PL/SQL execution elapsed time	1.78	0.17
hard parse (sharing criteria) elapsed time	0.12	0.01
PL/SQL compilation elapsed time	0.09	0.01
connection management call elapsed time	0.02	0.00
repeated bind elapsed time	0.01	0.00
sequence load elapsed time	0.01	0.00
hard parse (bind mismatch) elapsed time	0.00	0.00
DB time	1,057.02	

顶级SQL报表

按用时排序的 SQL

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
1,006	453	1,757	0.57	95.22	fu02q80b2kva1	DEMO	select time_id, QUANTITY_SOLD...
31	10	0		2.94	4km		CLARE pid NUMBER := 37; ...

fu02q80b2kva1

按 CPU 时间排序的 SQL

CPU Time (s)	Elapsed Time (s)	Executions	CPU per Exec (s)	% Total	% Total DB Time	SQL Id	SQL Module	SQL Text
453	1,006	1,757	0.26	95.58	95.22	fu02q80b2kva1	DEMO	select time_id, QUANTITY_SOLD...
11	21	0		2.29	1.99	710ydq8q1fj3r	SQL*Plus	DECLARE pid NUMBER := 39; ...

按执行数排序的 SQL

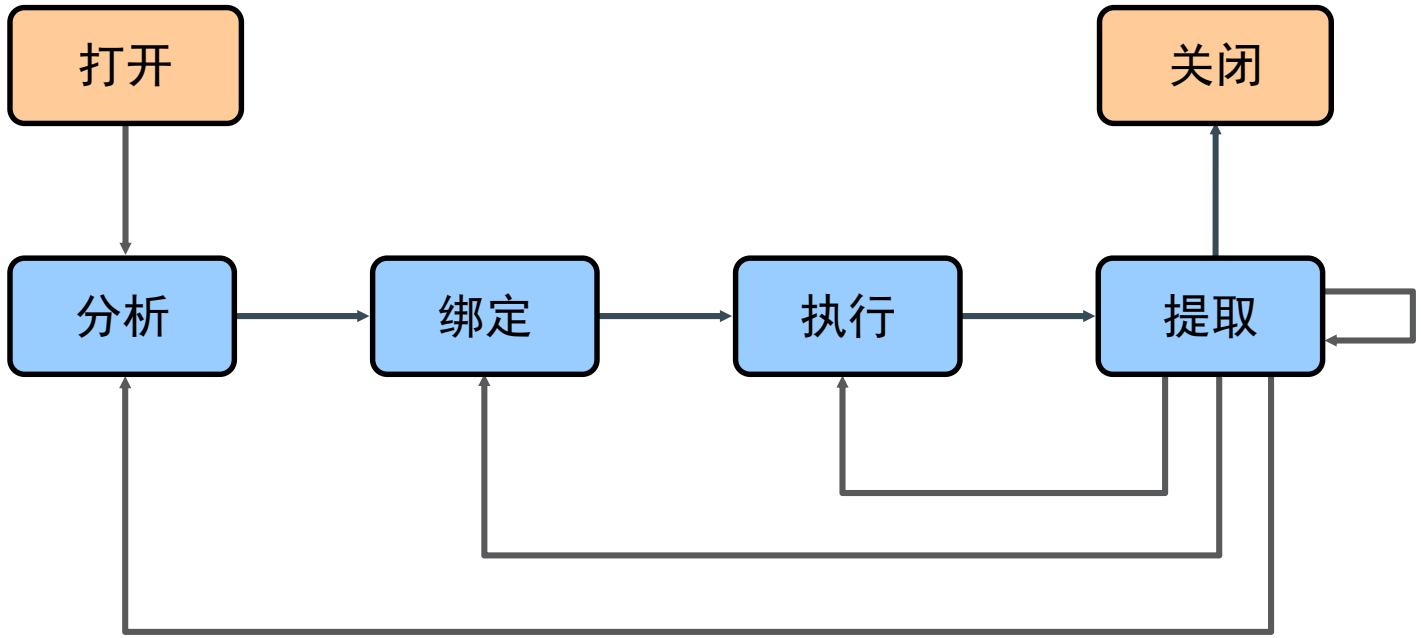
Executions	Rows Processed	Rows per Exec	CPU per Exec (s)	Elap per Exec (s)	SQL Id	SQL Module	SQL Text
1,757	5,271	3.00	0.26	0.57	fu02q80b2kva1	DEMO	select time_id, QUANTITY_SOLD...
415	284	0.68	0.00	0.00	96q93hntrzjtr		select /*+ rule */ bucket_cnt,...

按缓冲区获取数排序的 SQL

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
5,579,956	1,757	3,175.84	93.92	453.08	1006.49	fu02q80b2kva1	DEMO	select time_id, QUANTITY_SOLD...
134,514	0		2.26	10.84	21.04	710ydq8q1fj3r	SQL*Plus	DECLARE pid NUMBER := 39; ...

SQL优化

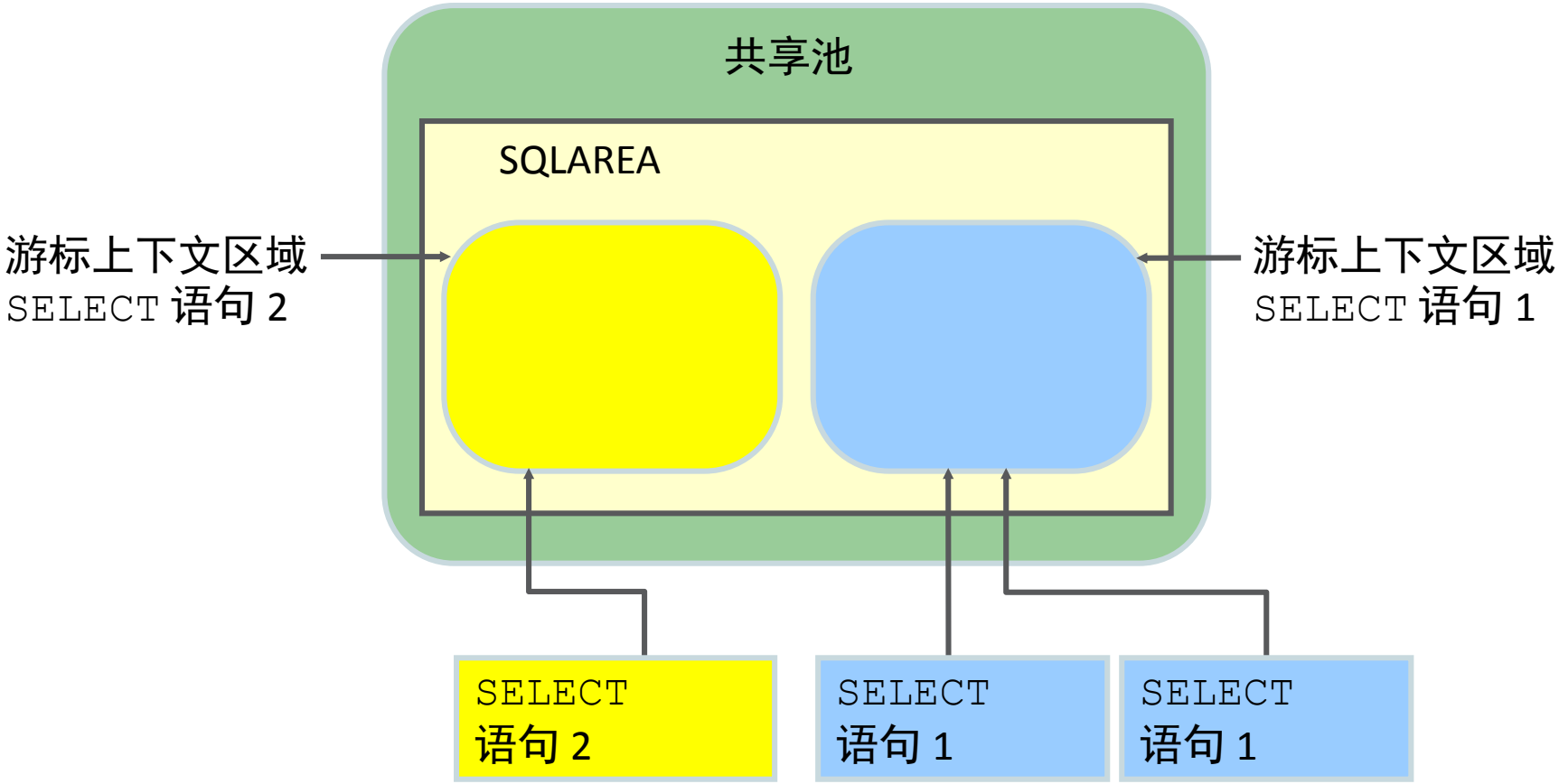
SQL 语句处理阶段



分析阶段

- 分析阶段：
  - 始终：
    - 检查语法
    - 检查语义和权限
  - 软分析：
    - 在共享池中搜索语句
  - 硬分析：
    - 合并视图定义和子查询
    - 确定执行计划

SQL存储



重复的 SQL

Duplicate SQL

Page Refreshed **Jul 6, 2007 4:49:59 AM GMT+07:00** Refresh

Applications can cause database to consume excessive CPU by parsing SQL statements that are similar and that can share the same SQL text. Such applications can also cause slow performance by creating contention in the database for Library Cache or Shared Pool.

CPU Consumption Since Instance Started

CPU Used as percentage of Total CPU (%) **2.19**

CPU Used for Parsing as percentage of CPU Used(%) **17.20**

Duplicate SQL Statements

This report identifies similar SQL statements that could be shared by a single SQL statement if the database application used bind variables to replace literals and SQL coding conventions to remove differences based only on character case or whitespace. You can re-write the SQL statements to gain the efficiency of using a single, shared statement.

Note: Only the first 2000 SQL statements that are executed only once are examined. The actual number of SQL statements that are duplicates can be more than 2000.

Expand All | Collapse All

Duplicates	Plan Hash Value	SQL Text
Duplicates		
5	1445457117	select * from hr.employees where employee_id = 148
	1445457117	select * from hr.employees where employee_id = 148
	1445457117	select * from hr.employees where employee_id = 145
	1445457117	select * from hr.employees where employee_id = 133
	1445457117	select * from hr.employees where employee_id = 100
	1445457117	select * from hr.employees where employee_id = 132

绑定变量候选项

硬解析的原因

- 内存不足，可以缓存的SQL有限
- 不一致的SQL太多，但执行计划一样
  - Select sal from emp where ename='Smith'
  - Select sal from emp where ename='Jack'

```
String v_id = 'Smith';
String v_sql = 'select sal from emp where ename = ? '; //嵌入绑定变量
stmt = con.prepareStatement( v_sql );
stmt.setString(1, v_id ); //为绑定变量赋值
stmt.executeQuery();
```

SQL 语句处理阶段：绑定

- 绑定阶段：
  - 检查语句的绑定变量
  - 为绑定变量分配或重新分配值
- 下列情况下，绑定变量会影响性能：
  - 通过使用共享游标来减少分析。

Cursor_sharing参数有3个值可以设置：

1)、EXACT：通常来说，exact值是Oracle推荐的，也是默认的，它要求SQL语句在完全相同时才会重用，否则会被重新执行硬解析操作。

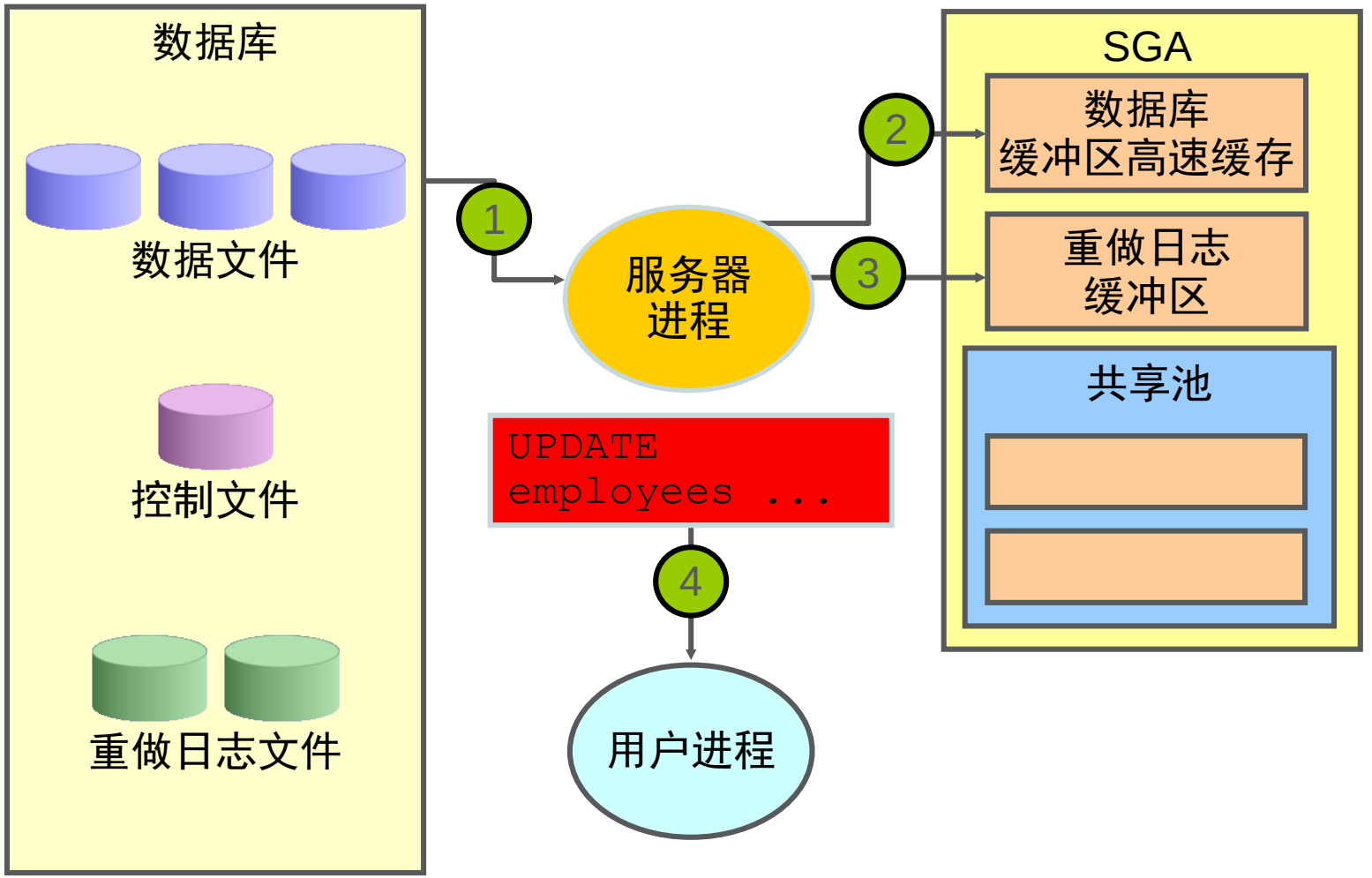
2)、SIMILAR：similar是在Oracle认为某条SQL语句的谓词条件可能会影响到它的执行计划时，才会被重新分析，否则将重用SQL。

3)、FORCE：force是在任何情况下，无条件重用SQL。

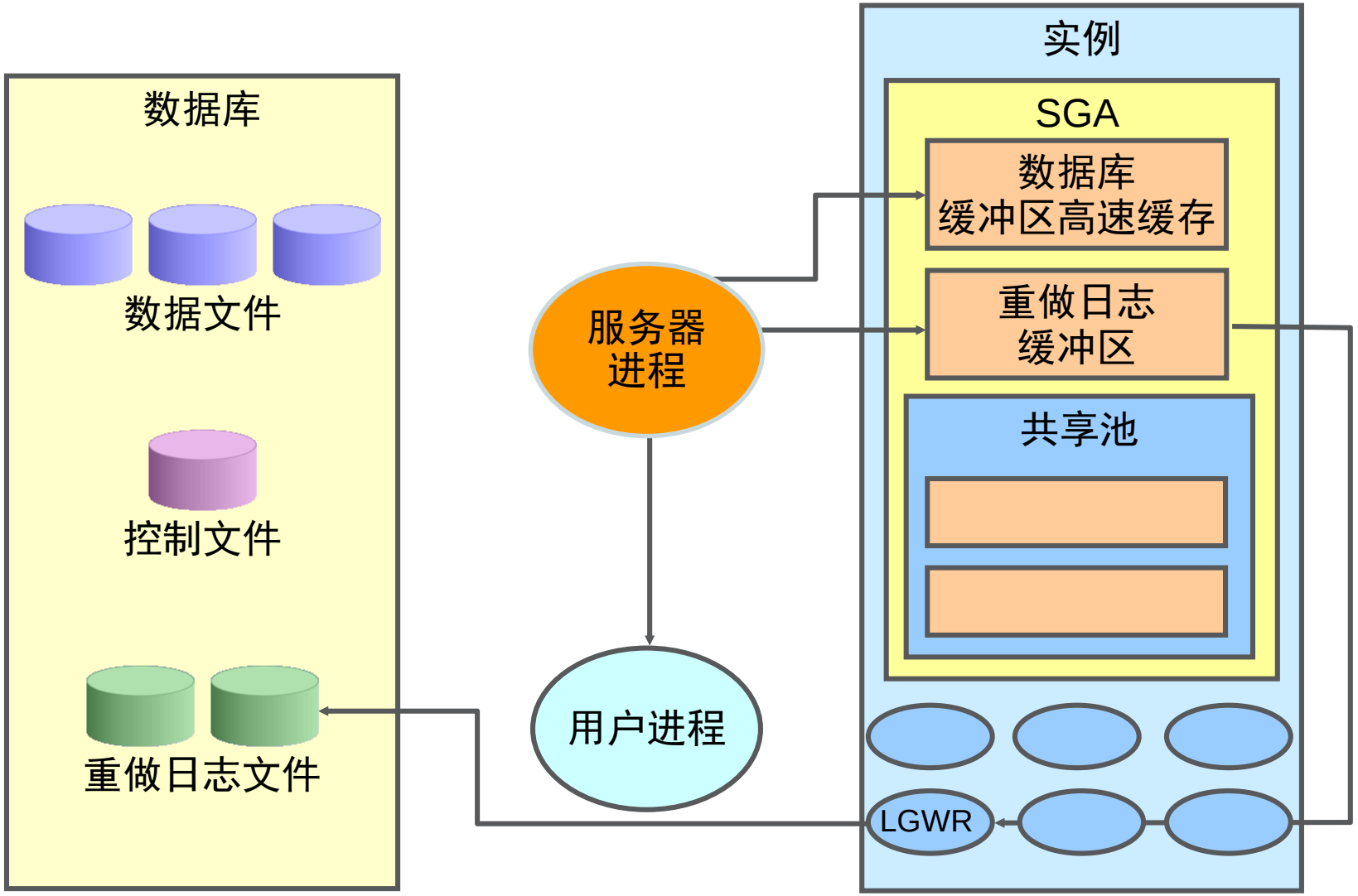
SQL 语句处理阶段：执行和提取

- 执行阶段：
  - 执行 SQL 语句
  - 针对数据操纵语言 (DML) 语句执行所需的 I/O 和排序
- 提取阶段：
  - 针对查询检索行
  - 视需要对查询进行排序
  - 使用数组提取机制

处理 DML 语句



提交处理



确定不良 SQL

- 不良 SQL 会在没有必要的情况下过多地使用资源。
- 不良 SQL 具有以下特征：
  - 运行时间长
  - 过多的 I/O（物理读写）
  - 过多的 CPU 时间
  - 过多的等待
- COST高

顶级 SQL报表

按 CPU 时间排序的 SQL

CPU Time (s)	Elapsed Time (s)	Executions	CPU per Exec (s)	% Total	% Total DB Time	SQL Id	SQL Module	SQL Text
48	50	1,440	0.03	13.89	11.93	6gvch1xu9ca3g		DECLARE job BINARY_INTEGER := …
19	35	69,080	0.00	5.61	8.34	6v7n0y2bq89n8	OEM.SystemPool	BEGIN EMDW_LOG.set_context(MGM…
18	18	482	0.04	5.16	4.28	bjsc9c4gg59jj	OEM.CacheModeWaitPool	BEGIN EMDW_LOG.set_context(MGM…

按获取数排序的 SQL

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
1,408,382	482	2,921.95	17.40	17.91	17.97	bjsc9c4gg59jj	OEM.CacheModeWaitPool	BEGIN EMDW_LOG.set_context(MGM…
1,128,856	1,440	783.93	13.94	48.18	50.11	6gvch1xu9ca3g		DECLARE job BINARY_INTEGER := …
643,279	69,080	9.31	7.95	19.48	35.02	6v7n0y2bq89n8	OEM.SystemPool	BEGIN EMDW_LOG.set_context(MGM…
484,685	965	502.26	5.99	4.88	4.90	fyk8b9986ntk7	OEM.CacheModeWaitPool	SELECT EXECUTION_ID, STATUS, …

顶级 SQL查询—v$session_longops

SID	Session标识
SERIAL#	Session串号
OPNAME	操作简要说明
TARGET	操作运行所在的对象
TARGET_DESC	目标对象说明
SOFAR	至今为止完成的工作量
TOTALWORK	总工作量
UNITS	工作量单位
START_TIME	操作开始时间
LAST_UPDATE_TIME	统计项最后更新时间
TIMESTAMP	操作的时间戳
TIME_REMAINING	预计完成操作的剩余时间(秒)
ELAPSED_SECONDS	从操作开始总花费时间(秒)
CONTEXT	前后关系
MESSAGE	统计项的完整描述
USERNAME	执行操作的用户ID
SQL_ADDRESS	关联v$sql
SQL_HASH_VALUE	关联v$sql
SQL_ID	关联v$sql

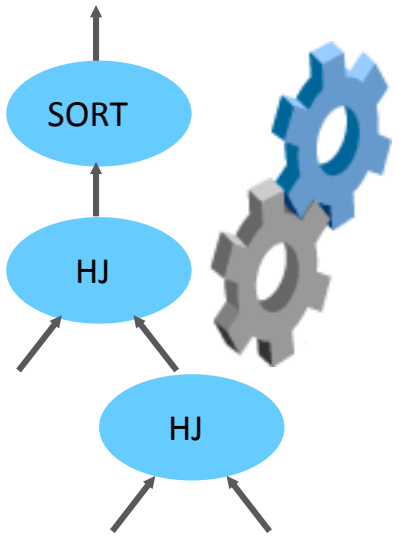
Oracle 优化程序的作用

- Oracle 查询优化程序确定最高效的执行计划，这是处理任何 SQL 语句过程中最重要的一个步骤。
- 优化程序将：
  - 评估表达式和条件
  - 使用对象和系统统计信息
  - 确定如何访问数据
  - 确定如何联接表
  - 确定哪条访问路径最有效

什么是执行计划

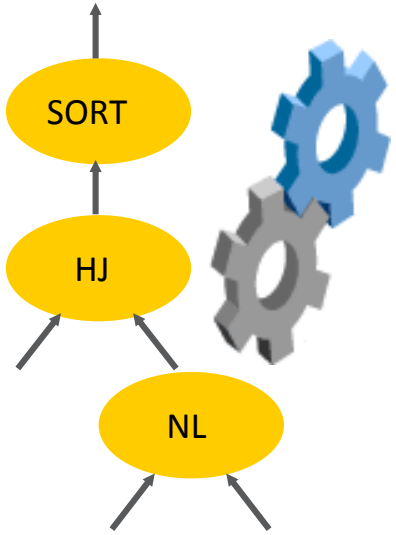
- 执行计划是当优化程序执行 SQL 语句并执行某一操作时所执行的一组步骤。

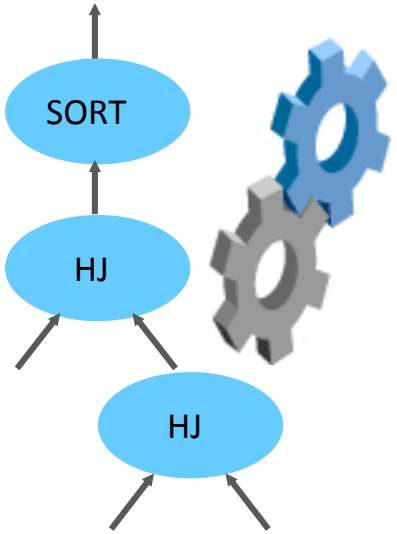
```
Plan hash value: 1343509718

-------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |   106 |  2862 |     6  (17|
|   1 |  MERGE JOIN                   |             |   106 |  2862 |     6  (17|
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |   432 |     2   (0|
|   3 |    INDEX FULL SCAN            | DEPT_ID_PK  |    27 |       |     1   (0|
|*  4 |   SORT JOIN                   |             |   107 |  1177 |     4  (25|
|   5 |    TABLE ACCESS FULL          | EMPLOYEES   |   107 |  1177 |     3   (0|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
       filter("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

18 rows selected.
```



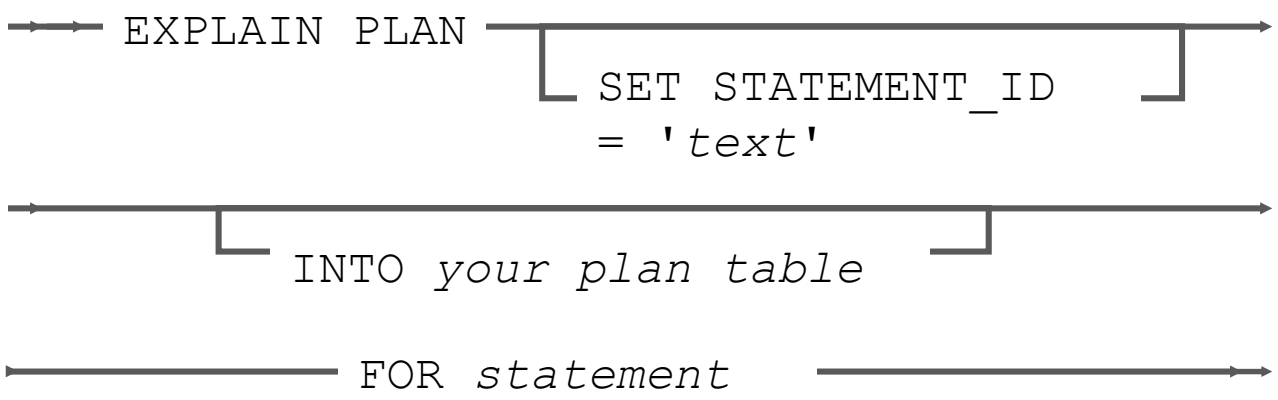
使用执行计划

- 确定当前的执行计划
- 确定索引的效果
- 确定访问路径
- 确认使用索引
- 确认可使用的执行计划





EXPLAIN PLAN 命令：示例




```
EXPLAIN PLAN
FOR
SELECT e.last_name, d.department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id;
Explained.
```

注：EXPLAIN PLAN 命令不实际执行语句。

EXPLAIN PLAN 命令：输出

```
SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
Plan hash value: 1343509718

-------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |   106 |  2862 |     6  (17|
|   1 |  MERGE JOIN                   |             |   106 |  2862 |     6  (17|
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |   432 |     2   (0|
|   3 |    INDEX FULL SCAN            | DEPT_ID_PK  |    27 |       |     1   (0|
|*  4 |   SORT JOIN                   |             |   107 |  1177 |     4  (25|
|   5 |    TABLE ACCESS FULL          | EMPLOYEES   |   107 |  1177 |     3   (0|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
       filter("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

18 rows selected.
```

EXPLAIN PLAN 命令：输出

```
SELECT PLAN_TABLE_OUTPUT FROM
TABLE(DBMS_XPLAN.DISPLAY_CURSOR('cfz0cdukrfdnu'));
```

```
SQL_ID  cfz0cdukrfdnu, child number 0
-------------------------------------
SELECT e.last_name, d.department_name
FROM hr.employees e, hr.departments d WHERE
e.department_id =d.department_id

Plan hash value: 1343509718
-------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |       |       |     6 (100|
|   1 |  MERGE JOIN                   |             |   106 |  2862 |     6  (17|
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |   432 |     2   (0|
|   3 |    INDEX FULL SCAN            | DEPT_ID_PK  |    27 |       |     1   (0|
|*  4 |   SORT JOIN                   |             |   107 |  1177 |     4  (25|
|   5 |    TABLE ACCESS FULL          | EMPLOYEES   |   107 |  1177 |     3   (0|
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
       filter("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

24 rows selected.
```

查询 AWR中SQL ID的执行计划

```
SELECT PLAN_TABLE_OUTPUT FROM TABLE
(DBMS_XPLAN.DISPLAY_AWR('454rug2yva18w'));
```

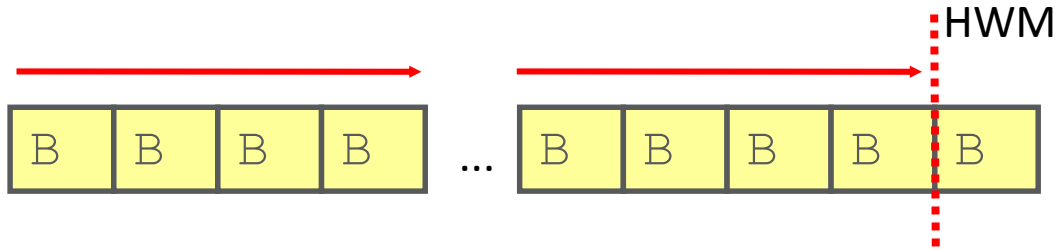
```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
SQL_ID 454rug2yva18w
--------------------
select /* example */ * from hr.employees natural join hr.departments

Plan hash value: 2052257371

-------------------------------------------------------------------------------------
| Id  | Operation           | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |             |       |       |     6 (100)|          |
|   1 |  HASH JOIN          |             |    11 |   968 |     6  (17)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPARTMENTS |    11 |   220 |     2   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMPLOYEES   |   107 |  7276 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

全表扫描

- 缺少索引
- 大量数据
- 小表

- 多块 I/O 调用

```
select * from emp where ename = 'KING';
-----------------------------------------------------------------------
| Id  | Operation             |  Name       | Rows  | Bytes | Cost  |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |     1 |    22 |     2 |
|*  1 |  TABLE ACCESS FULL    | EMP         |     1 |    22 |     2 |
-----------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("EMP"."ENAME"='KING')
```



影响优化程序的因素

- 索引
- 统计信息

影响优化程序的因素—索引

使用索引可提高 SQL 性能

- 对下列项使用索引：
  - 联接列
  - 频繁出现在 **WHERE** 子句中的列
- 对下列项使用组合索引：
  - 频繁一起使用在 WHERE 子句中的列

影响优化程序的因素—统计信息

优化程序统计信息：

- 是某一时间点的快照
- 每次重新启动实例后会变为永久信息
- 可自动收集

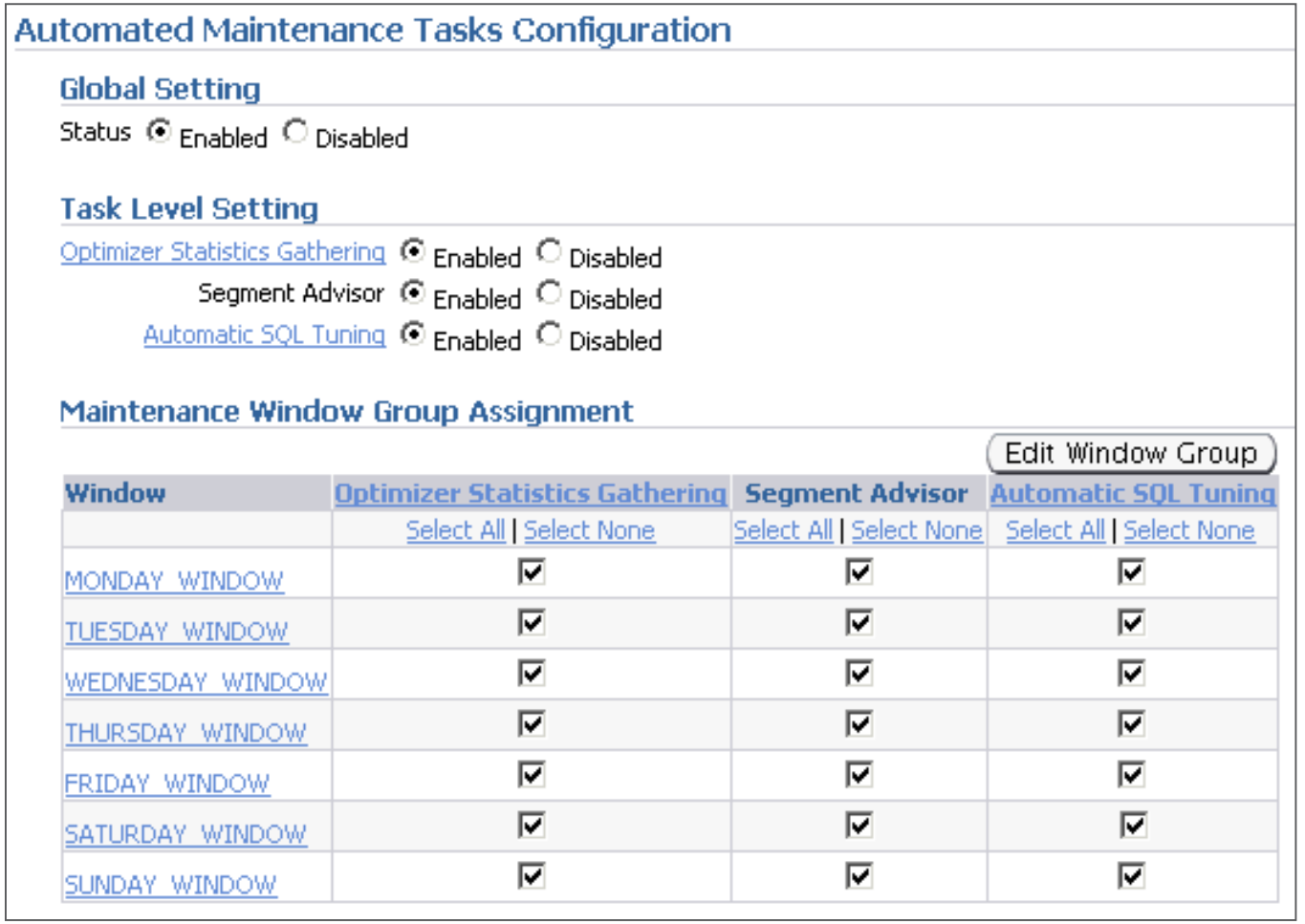
```
SQL> SELECT COUNT(*) FROM hr.employees;
  COUNT(*)
----------
       214
SQL> SELECT num_rows FROM dba_tables
  2 WHERE owner='HR' AND table_name = 'EMPLOYEES';
  NUM_ROWS
----------
       107
```

影响优化程序的因素—统计信息

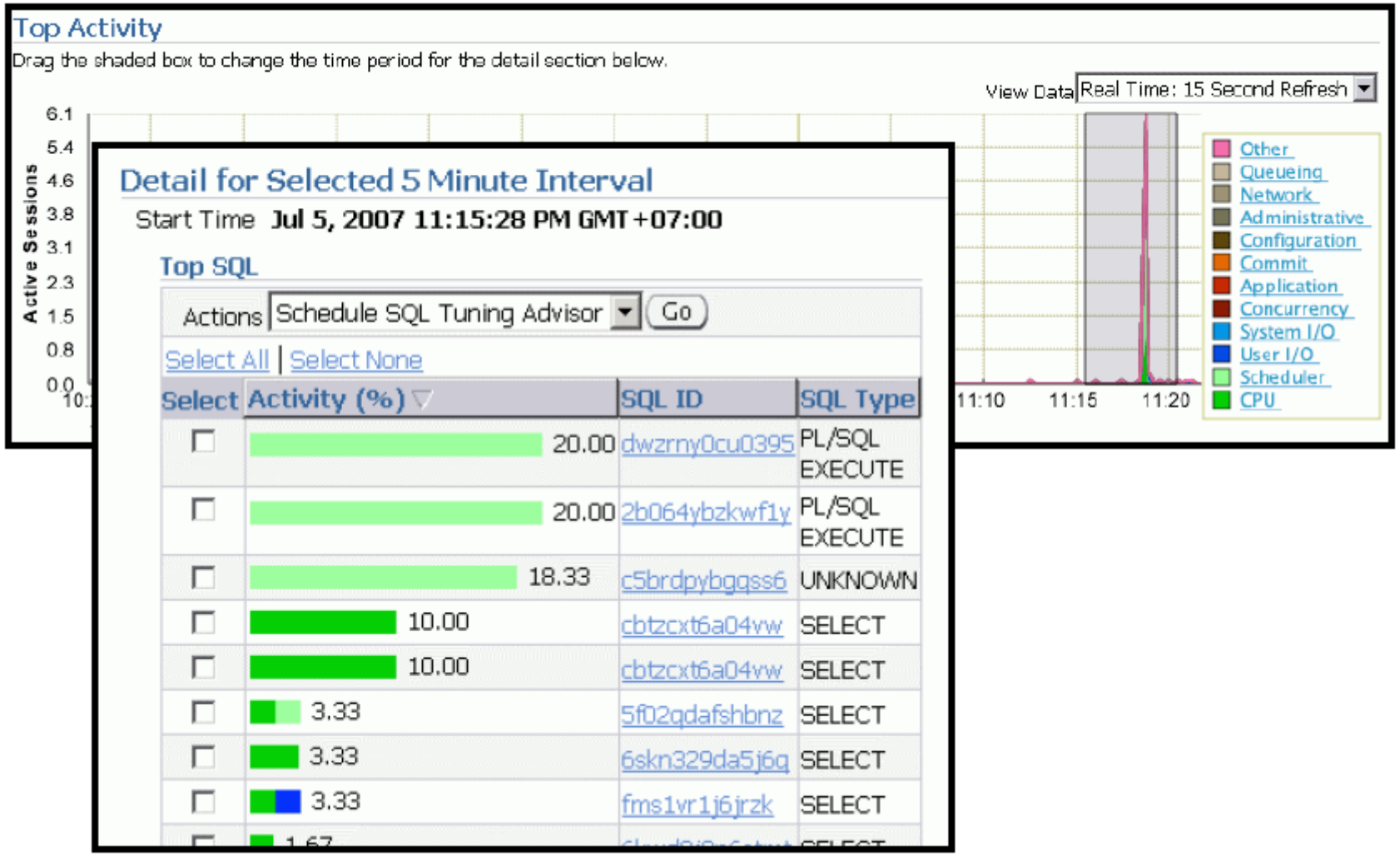
- 主要是描述数据库中表，索引的大小，规模，数据分布状况等的一类信息。
- 例如，表的行数，块数，平均每行的大小，索引的leaf blocks，索引字段的行数，不同值的大小等，都属于统计信息。
- 优化程序正是根据这些统计信息数据，计算出不同访问路径下，不同join 方式下，各种计划的成本，最后选择出成本最小的计划。
- 统计信息是存放在数据字典表中
- last_analyzed 字段表示上次统计信息搜集的时间，可以根据这个字段，快速的了解最近一次统计信息搜集的时间

优化自动维护任务

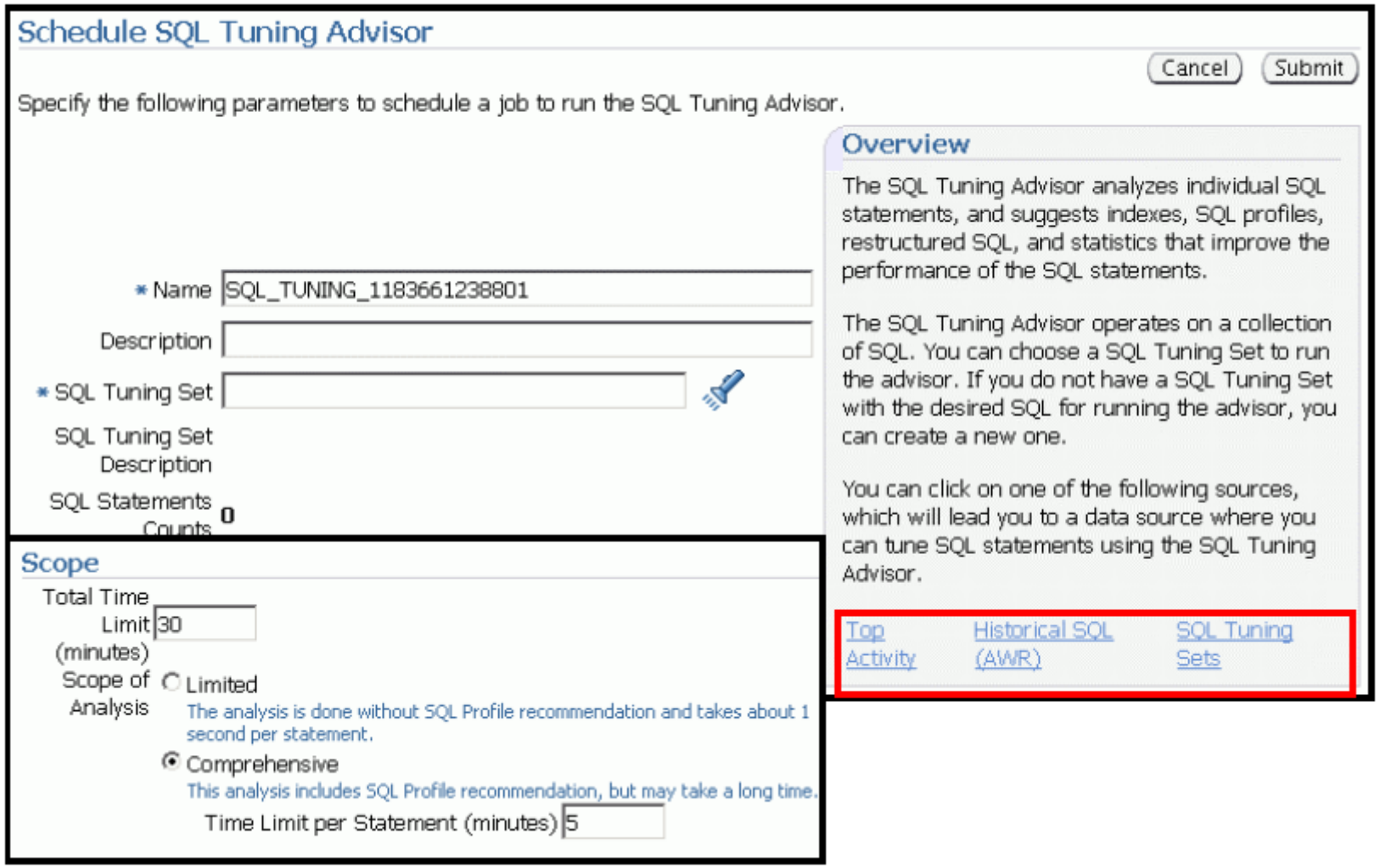
“Server > Automatic Maintenance Tasks > Configure
（服务器 > 自动维护任务 > 配置）”



Oracle SQL调优工具



Oracle SQL调优工具



Oracle SQL调优工具—构建优化任务

```
DECLARE
my_task_name varchar2(30);
my_sqltext clob;
BEGIN
my_sqltext:=q'[SQL正文]';
my_task_name:=DBMS_SQLTUNE.create_tuning_task(
sql_text => my_sqltext,
user_name => '用户名',
scope => 'COMPREHENSIVE',
time_limit => 600,
task_name => 'tuningA01');
END;
/
```

```
DECLARE
my_task_name varchar2(30);
BEGIN
my_task_name:=DBMS_SQLTUNE.create_tuning_task(
sql_id =>  ‘SQL ID',
scope => 'COMPREHENSIVE',
time_limit => 600,
task_name => 'tuningA02');
END;
/
```

Oracle SQL调优工具—执行优化任务

```
exec DBMS_SQLTUNE.execute_tuning_task('tuningA01');
```

Oracle SQL调优工具—获取优化结果

```
set long 999999
set LONGCHUNKSIZE 999999
set serveroutput on size 999999
set linesize 200
select dbms_sqltune.report_tuning_task(' tuningA01') from dual;
```

Oracle SQL调优工具—获取优化结果

```
DBMS_SQLTUNE.REPORT_TUNING_TASK('tuningA01')
-------------------------------------------
GENERAL INFORMATION SECTION
-------------------------------------------
Tuning Task Name   : tuningA01
Tuning Task Owner  : P
Workload Type      : Single SQL Statement
Scope              : COMPREHENSIVE
Time Limit(seconds): 900
Completion Status  : COMPLETED
Started at         : 11/11/2015 10:57:07
Completed at       : 11/11/2015 10:57:16


DBMS_SQLTUNE.REPORT_TUNING_TASK('tuningA01')
-----------------------------------------------
Schema Name: P
SQL ID     : 0qxhp13hqrsxk
SQL Text   : select count(*) as countX from (SELECT a.c_ci_sub_comp,
             a.c_ci_com_nme FROM Web_Clm_Ply_Coinsurance a where 1=1 )
             tmp_count_t

-----------------------------------------------
FINDINGS SECTION (2 findings)
-----------------------------------------------
```

Oracle SQL调优工具—获取优化结果

```
-------------------------------------------
FINDINGS SECTION (2 findings)
-------------------------------------------

DBMS_SQLTUNE.REPORT_TUNING_TASK('tuningA01')
-------------------------------------------
1- Statistics Finding
---------------------
  Optimizer statistics for index "P"."PK_CLM_PLY_COINSURANCE" are stale.

  Recommendation
  --------------
  - Consider collecting optimizer statistics for this index.
    execute dbms_stats.gather_index_stats(ownname => 'P', indname =>
            'PK_CLM_PLY_COINSURANCE', estimate_percent =>
            DBMS_STATS.AUTO_SAMPLE_SIZE);

DBMS_SQLTUNE.REPORT_TUNING_TASK('tuningA01')
-----------------------------------------------

  Rationale
  ---------
    The optimizer requires up-to-date statistics for the index in order to
    select a good execution plan.
```

Oracle SQL调优工具—获取优化结果

```
2- Statistics Finding
---------------------
  Optimizer statistics for table "P"."WEB_CLM_PLY_COINSURANCE" and its
  indices are stale.

  Recommendation

DBMS_SQLTUNE.REPORT_TUNING_TASK('tuningA01')
--------------------------------------------------
  - Consider collecting optimizer statistics for this table.
    execute dbms_stats.gather_table_stats(ownname => 'P', tabname =>'WEB_CLM_PLY_COINSURANCE', estimate_percent
=>DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO',cascade=>true,degree=>4);

  Rationale
  ---------
    The optimizer requires up-to-date statistics for the table in order to
    select a good execution plan.
```

Oracle SQL调优工具—示例

```
DBMS_SQLTUNE.REPORT_TUNING_TASK('TUNING111204')
-------------------------------------------------

GENERAL INFORMATION SECTION
-------------------------------------------------

Tuning Task Name   : tuning111204
Tuning Task Owner  : P
Workload Type      : Single SQL Statement
Scope              : COMPREHENSIVE
Time Limit(seconds): 900
Completion Status  : COMPLETED
Started at         : 11/12/2016 10:55:02
Completed at       : 11/12/2016 10:55:02


DBMS_SQLTUNE.REPORT_TUNING_TASK('TUNING111204')
-------------------------------------------------

Schema Name: P
SQL ID     : 98tz06au4x099
SQL Text   : select count(*) as countX from
             (select c_cde, c_cnm, 'codeKind'
             from web_bas_comm_code a where a.c_is_valid = '1' and  1=1 )
             tmp_count_t
```

Oracle SQL调优工具—示例

```
-------------------------------------------------------
FINDINGS SECTION (2 findings)
-------------------------------------------------------


DBMS_SQLTUNE.REPORT_TUNING_TASK('TUNING111204')
-------------------------------------------------------
1- Statistics Finding
---------------------
  Optimizer statistics for table "P"."WEB_BAS_COMM_CODE" and its indices
  are stale.

  Recommendation
  --------------
  - Consider collecting optimizer statistics for this table.
    execute dbms_stats.gather_table_stats(ownname => 'P', tabname =>
            'WEB_BAS_COMM_CODE', estimate_percent =>
            DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO');

  Rationale
  ---------
    The optimizer requires up-to-date statistics for the table in order to
    select a good execution plan.
```

Oracle SQL调优工具—示例

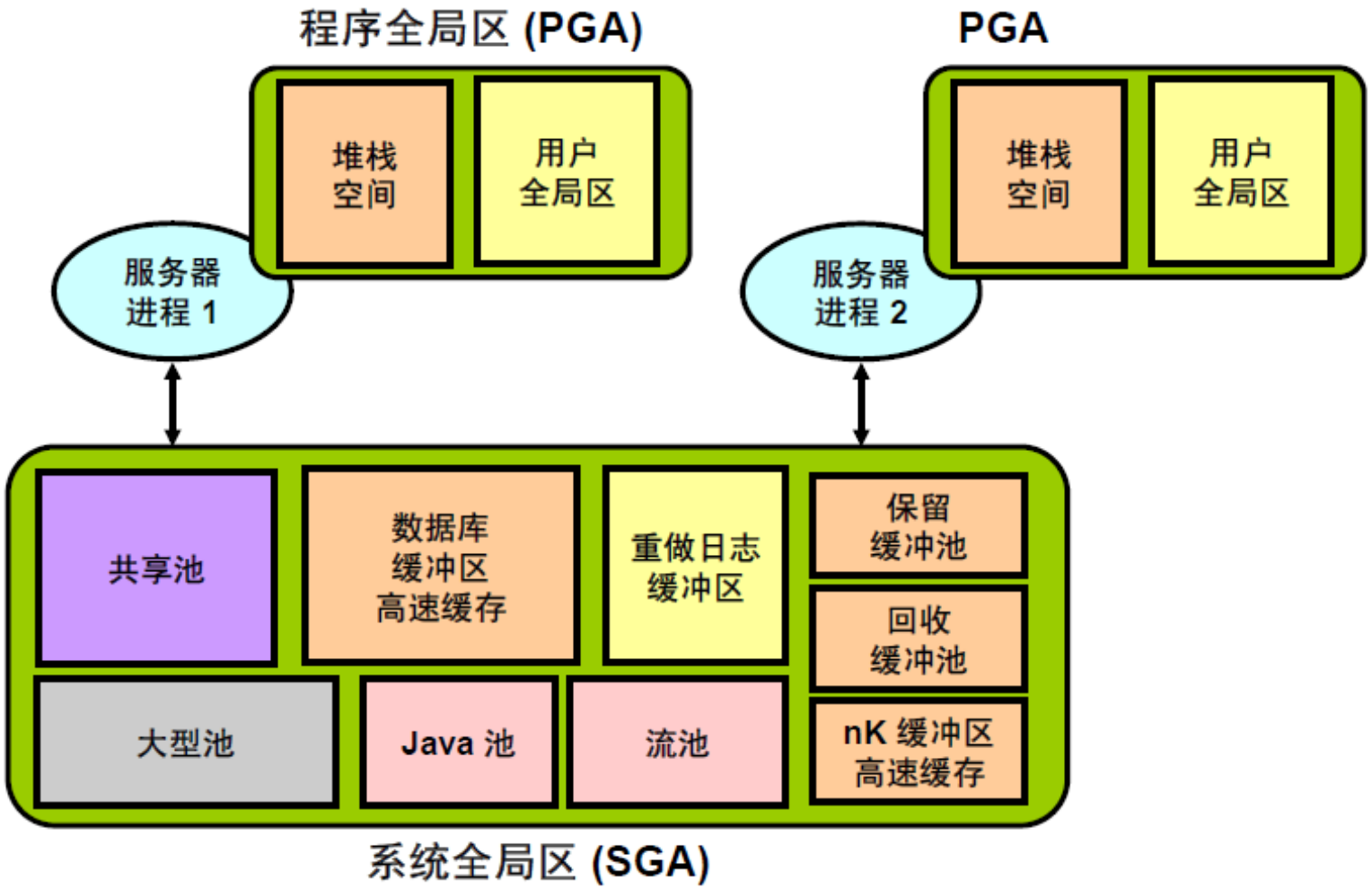
```
2- Index Finding (see explain plans section below)
--------------------------------------------------
  The execution plan of this statement can be improved by creating one or more
  indices.

DBMS_SQLTUNE.REPORT_TUNING_TASK('TUNING111204')
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  Recommendation (estimated benefit: 57.61%)
  ------------------------------------------
  - Consider running the Access Advisor to improve the physical schema design
    or creating the recommended index.
    create index P.IDX$$_1A1C70001 on P.WEB_BAS_COMM_CODE("C_IS_VALID");

  Rationale
  ---------
    Creating the recommended indices significantly improves the execution plan
    of this statement. However, it might be preferable to run "Access Advisor"
    using a representative SQL workload as opposed to a single statement. This
    will allow to get comprehensive index recommendations which takes into
    account index maintenance overhead and additional space consumption.
```

Oracle SQL调优工具—示例

```
-------------------------------------------------------------------------------
EXPLAIN PLANS SECTION
-------------------------------------------------------------------------------

1- Original
-----------

DBMS_SQLTUNE.REPORT_TUNING_TASK('TUNING111204')

Plan hash value: 3892849356

-----------------------------------------------------------------------------------------
| Id  | Operation          | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                    |     1 |     2 |    69   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE    |                    |     1 |     2 |            |          |
|*  2 |   TABLE ACCESS FULL| WEB_BAS_COMM_CODE  | 15146 | 30292 |    69   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------

2- Using New Indices
--------------------
Plan hash value: 2145693986

-----------------------------------------------------------------------------------------
| Id  | Operation          | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                    |     1 |     2 |    29   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |                    |     1 |     2 |            |          |
|*  2 |   INDEX RANGE SCAN | IDX$$_1A1C70001    | 15146 | 30292 |    29   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

实例优化

Oracle DB 内存结构



调整内存初始大小

- 最初估计的内存分配如下：
  - 保留可用内存的 20% 供其它应用程序使用。
  - 为 Oracle 实例分配 80% 的内存。

```
MEMORY_MAX_TARGET=(total_mem*80%)
```

  - 默认情况下，内存分配在开始时为 SGA 占 60%，PGA 占 40%。
  - 根据工作量调整内存分配。

设置 PGA_AGGREGATE_TARGET 的初始值

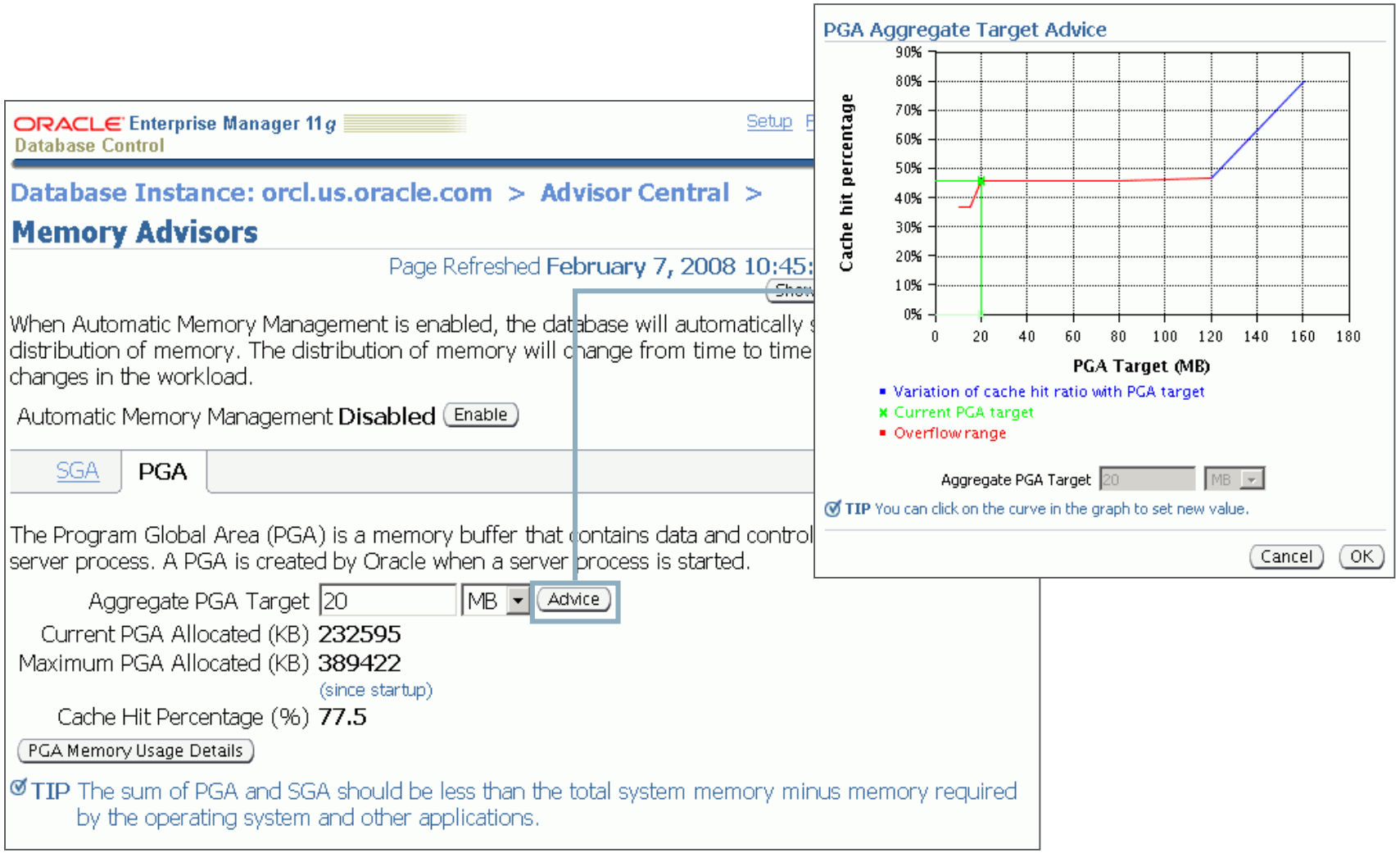
- 为其它应用程序保留 20% 的可用内存。
- 为 Oracle 实例保留 80% 的内存。
- 对于 OLTP：

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*20%
```

- 对于 DSS：

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*50%
```

自动 PGA 和 Oracle Enterprise Manager



查看AWR中的PGA建议

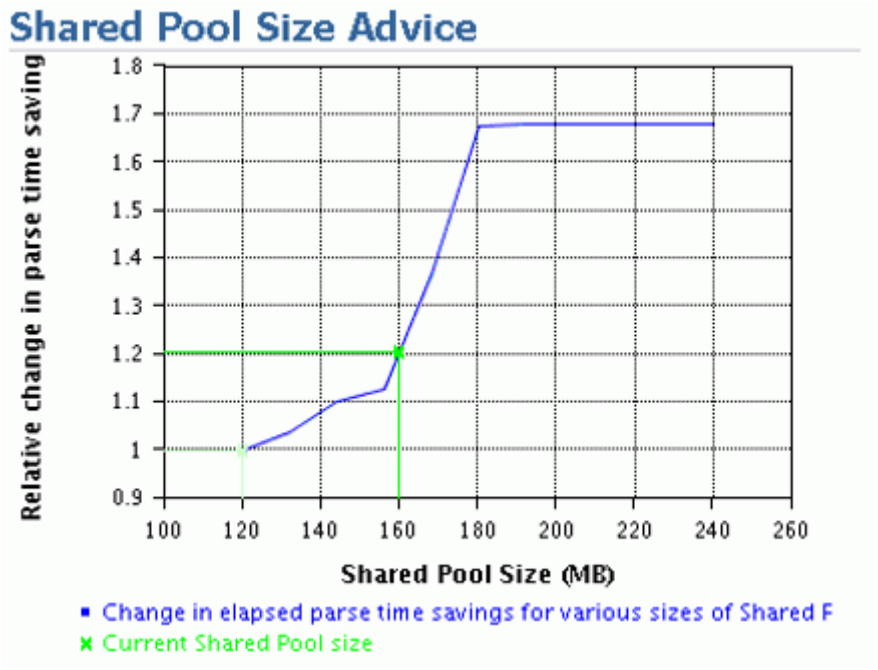
PGA Memory Advisory

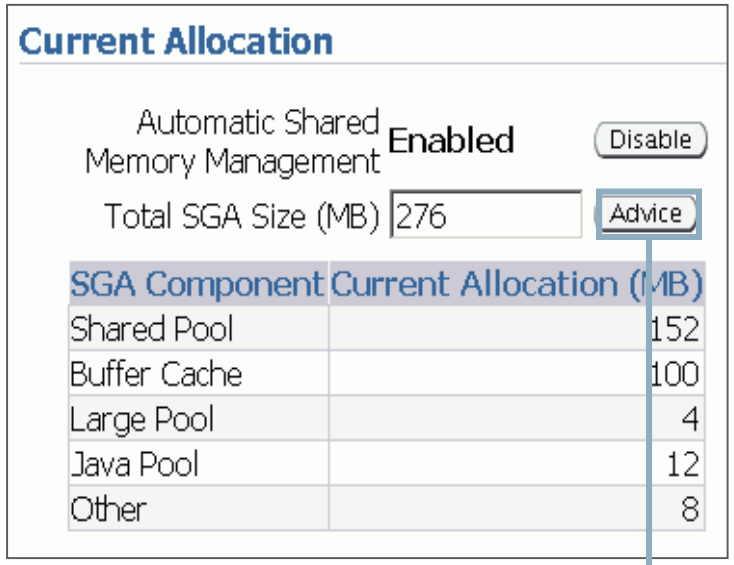
- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

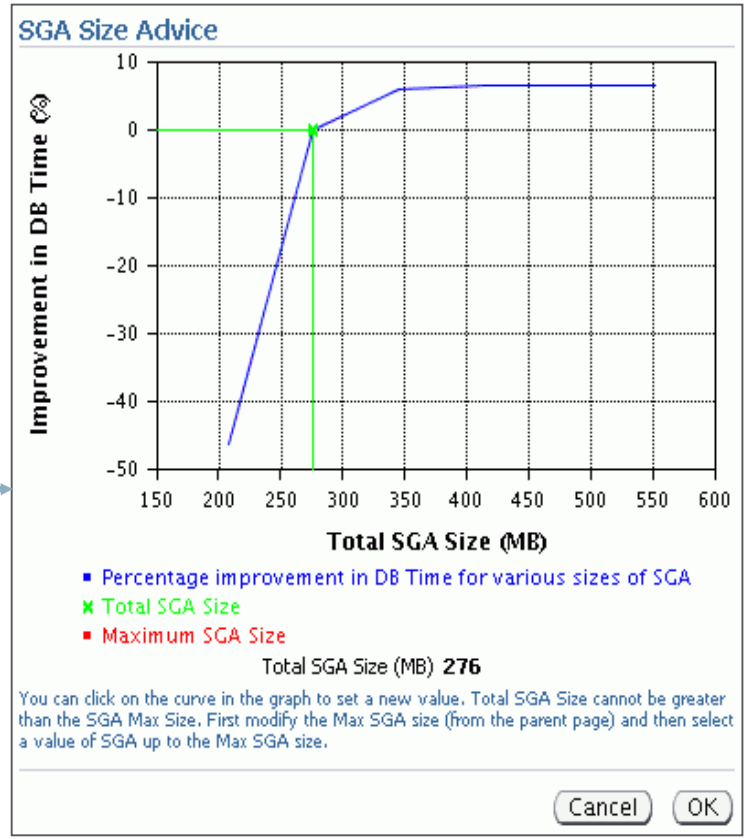
PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count	Estd Time
1,024	0.13	95,542,671.74	61,856.28	100.00	4,919	23,220,299,513
2,048	0.25	95,542,671.74	7,000.30	100.00	1	23,206,976,166
4,096	0.50	95,542,671.74	6,996.15	100.00	0	23,206,975,158
6,144	0.75	95,542,671.74	6,996.15	100.00	0	23,206,975,158
8,192	1.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490
9,830	1.20	95,542,671.74	3,979.55	100.00	0	23,206,242,490
11,469	1.40	95,542,671.74	3,979.55	100.00	0	23,206,242,490
13,107	1.60	95,542,671.74	3,979.55	100.00	0	23,206,242,490
14,746	1.80	95,542,671.74	3,979.55	100.00	0	23,206,242,490
16,384	2.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490
24,576	3.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490
32,768	4.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490
49,152	6.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490
65,536	8.00	95,542,671.74	3,979.55	100.00	0	23,206,242,490

ORACLE 甲骨文

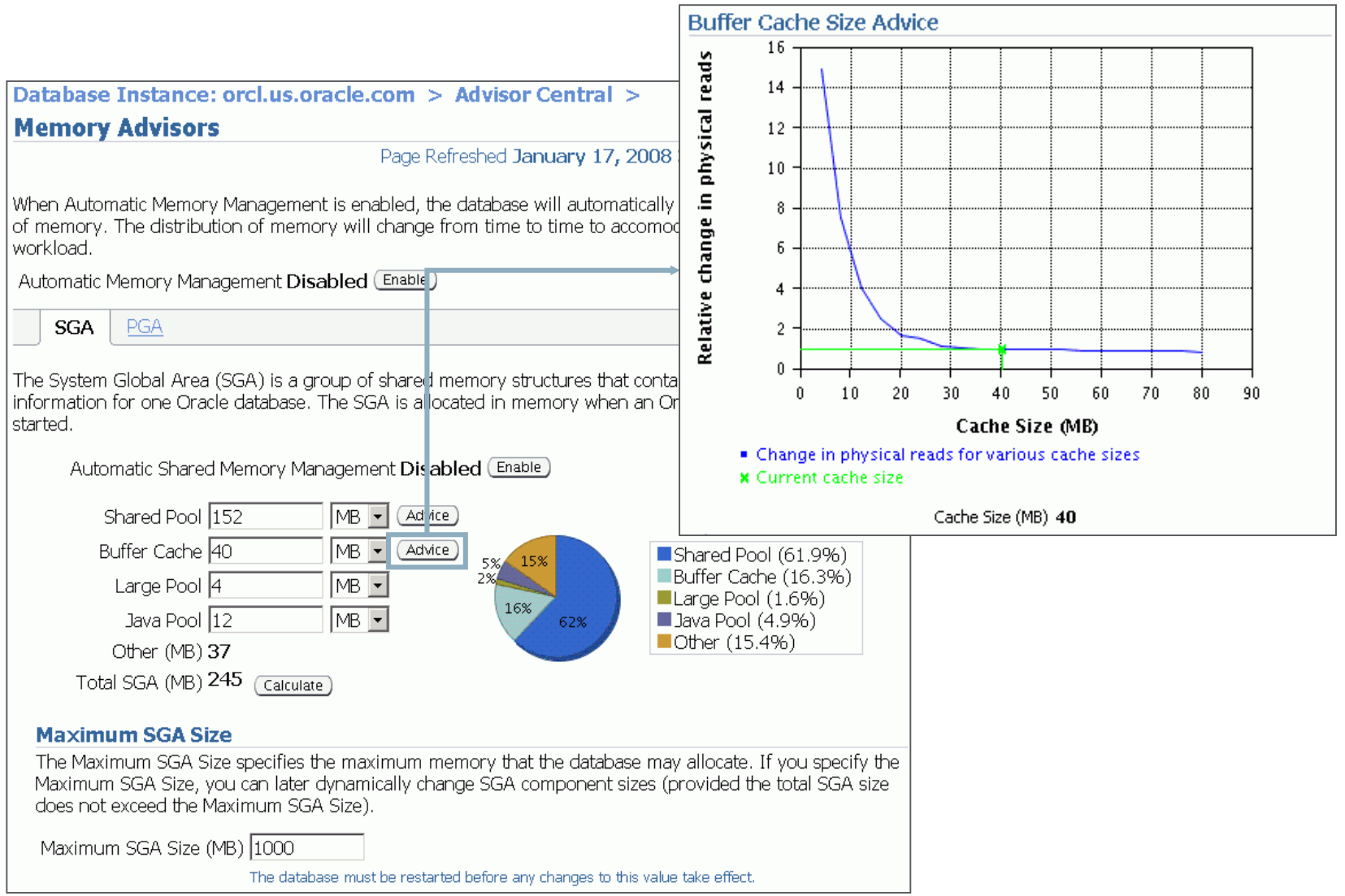
在 EM 中使用建议







在 EM 中使用缓冲区高速缓存建议



查看AWR中的Buffer Pool建议

Buffer Pool Advisory

- Only rows with estimated physical reads >0 are displayed
- ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers (thousands)	Est Phys Read Factor	Estimated Phys Reads (thousands)	Est Phys Read Time	Est %DBtime for Rds
D	4,992	0.10	615	4.82	52,883,553	1	221653093.00
D	9,984	0.20	1,231	3.40	37,334,637	1	154521700.00
D	14,976	0.29	1,846	2.67	29,320,128	1	119919572.00
D	19,968	0.39	2,462	2.19	24,040,849	1	97126641.00
D	24,960	0.49	3,077	1.83	20,102,834	1	80124513.00
D	29,952	0.59	3,693	1.52	16,623,146	1	65101193.00
D	34,944	0.68	4,308	1.34	14,665,028	1	56647153.00
D	39,936	0.78	4,923	1.21	13,218,378	1	50401333.00
D	44,928	0.88	5,539	1.10	12,070,619	1	45445957.00
D	49,920	0.98	6,154	1.02	11,149,043	1	41467113.00
D	51,072	1.00	6,296	1.00	10,965,565	1	40674964.00
D	54,912	1.08	6,770	0.95	10,380,895	1	38150689.00
D	59,904	1.17	7,385	0.89	9,733,485	1	35355534.00
D	64,896	1.27	8,000	0.84	9,177,050	1	32953167.00
D	69,888	1.37	8,616	0.79	8,690,740	1	30853554.00
D	74,880	1.47	9,231	0.75	8,254,084	1	28968321.00
D	79,872	1.56	9,847	0.72	7,861,367	1	27272788.00
D	84,864	1.66	10,462	0.68	7,503,284	1	25726793.00
D	89,856	1.76	11,078	0.65	7,168,242	1	24280269.00
D	94,848	1.86	11,693	0.62	6,852,572	1	22917385.00
D	99,840	1.95	12,308	0.60	6,532,467	1	21535356.00

查看AWR中的Shared Pool建议

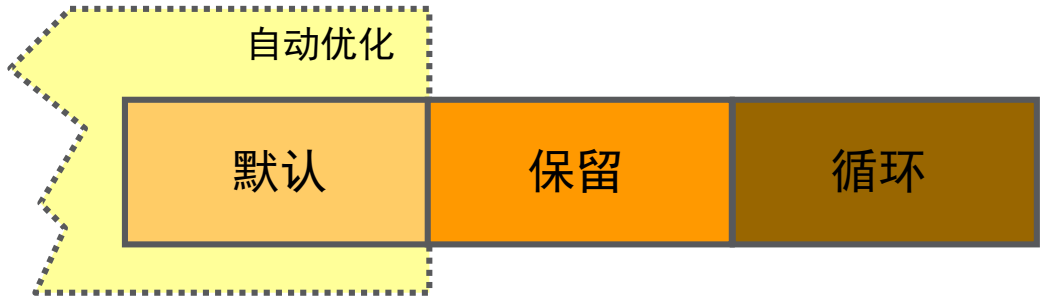
Shared Pool Advisory

- SP: Shared Pool Est LC: Estimated Library Cache Factr: Factor
- Note there is often a 1:Many correlation between a single logical object in the Library Cache, and the physical number of memory objects associated with it. Therefore comparing the number of Lib Cache objects (e.g
  invalid.

Shared Pool Size(M)	SP Size Factr	Est LC Size (M)	Est LC Mem Obj	Est LC Time Saved (s)	Est LC Time Saved Factr	Est LC Load Time (s)	Est LC Load Time Factr	Est LC Mem Obj Hits (K)
2,176	0.19	1,175	65,558	1,595,050,362	1.00	8,997,500	1.56	2,670,172
3,328	0.29	2,320	90,948	1,596,720,654	1.00	7,327,208	1.27	2,815,462
4,480	0.39	3,470	117,341	1,597,157,958	1.00	6,889,904	1.19	2,862,912
5,632	0.49	4,621	145,711	1,597,413,679	1.00	6,634,183	1.15	2,884,442
6,784	0.60	5,773	173,663	1,597,620,770	1.00	6,427,092	1.11	2,901,885
7,936	0.70	6,923	201,789	1,597,804,277	1.00	6,243,585	1.08	2,917,409
9,088	0.80	8,074	229,972	1,597,972,680	1.00	6,075,182	1.05	2,931,647
10,240	0.90	9,225	257,805	1,598,129,324	1.00	5,918,538	1.03	2,944,833
11,392	1.00	10,376	286,111	1,598,277,136	1.00	5,770,726	1.00	2,957,172
12,544	1.10	11,528	314,758	1,598,416,737	1.00	5,631,125	0.98	2,968,805
13,696	1.20	12,679	343,778	1,598,549,920	1.00	5,497,942	0.95	2,979,877
14,848	1.30	13,830	372,899	1,598,678,087	1.00	5,369,775	0.93	2,990,493
16,000	1.40	14,981	402,138	1,598,801,466	1.00	5,246,396	0.91	3,000,714
17,152	1.51	16,132	432,502	1,598,921,029	1.00	5,126,833	0.89	3,010,601
18,304	1.61	17,283	462,518	1,599,037,809	1.00	5,010,053	0.87	3,020,203
19,456	1.71	18,435	491,945	1,599,151,900	1.00	4,895,962	0.85	3,029,572
20,608	1.81	19,586	519,379	1,599,263,805	1.00	4,784,057	0.83	3,038,754
21,760	1.91	20,738	543,727	1,599,374,152	1.00	4,673,710	0.81	3,047,810
22,912	2.01	22,304	575,954	1,599,485,096	1.00	4,562,766	0.79	3,056,960

多个缓冲池

- 有三种缓冲池：
  - 默认：SYS 和非标记表或索引
  - 保留：热对象
  - 循环：很少访问
- 对使用已知访问路径的小而简单的方案非常有用



启用多个缓冲池

- 使用 BUFFER_POOL 子句。
- 此子句对表、聚簇和索引有效。
- 变更后，缓冲池可用于将来的读取。
- 对象可以具有多个缓冲池。

```
CREATE INDEX cust_idx …
  STORAGE (BUFFER_POOL KEEP …);

ALTER TABLE customer
  STORAGE (BUFFER_POOL RECYCLE);

ALTER INDEX cust_name_idx
  STORAGE (BUFFER_POOL KEEP);
```

计算多个池的命中率

```
SQL> SELECT name, 1 - (physical_reads /
  2  (db_block_gets + consistent_gets)) "HIT_RATIO"
  3  FROM V$BUFFER_POOL_STATISTICS
  4  WHERE db_block_gets + consistent_gets > 0;

NAME                 HIT_RATIO
---------------      ----------
KEEP                 .983520845
RECYCLE              .503866235
DEFAULT              .790350047
```

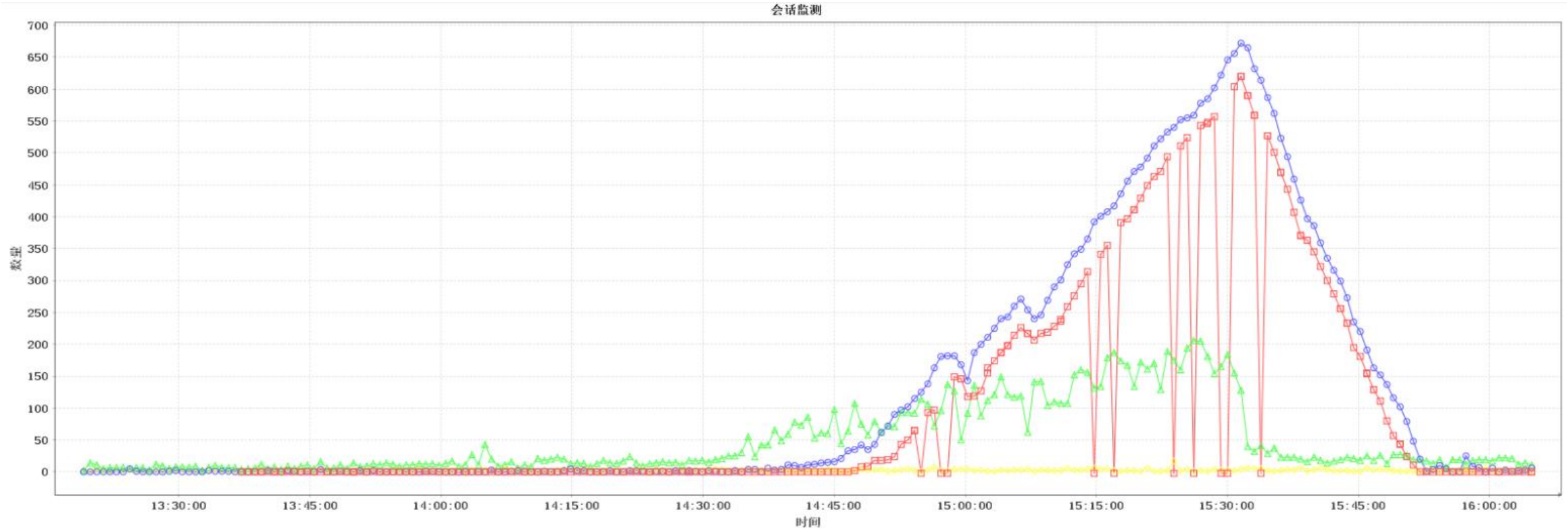
对表进行高速缓存

- 通过下列方式可以在全表扫描期间启用高速缓存：
  - 使用 CACHE 子句创建表
  - 使用 CACHE 子句变更表
  - 在查询中使用 CACHE 提示
- 准则：不要使缓冲区高速缓存过度拥挤。
- 使用保留池。

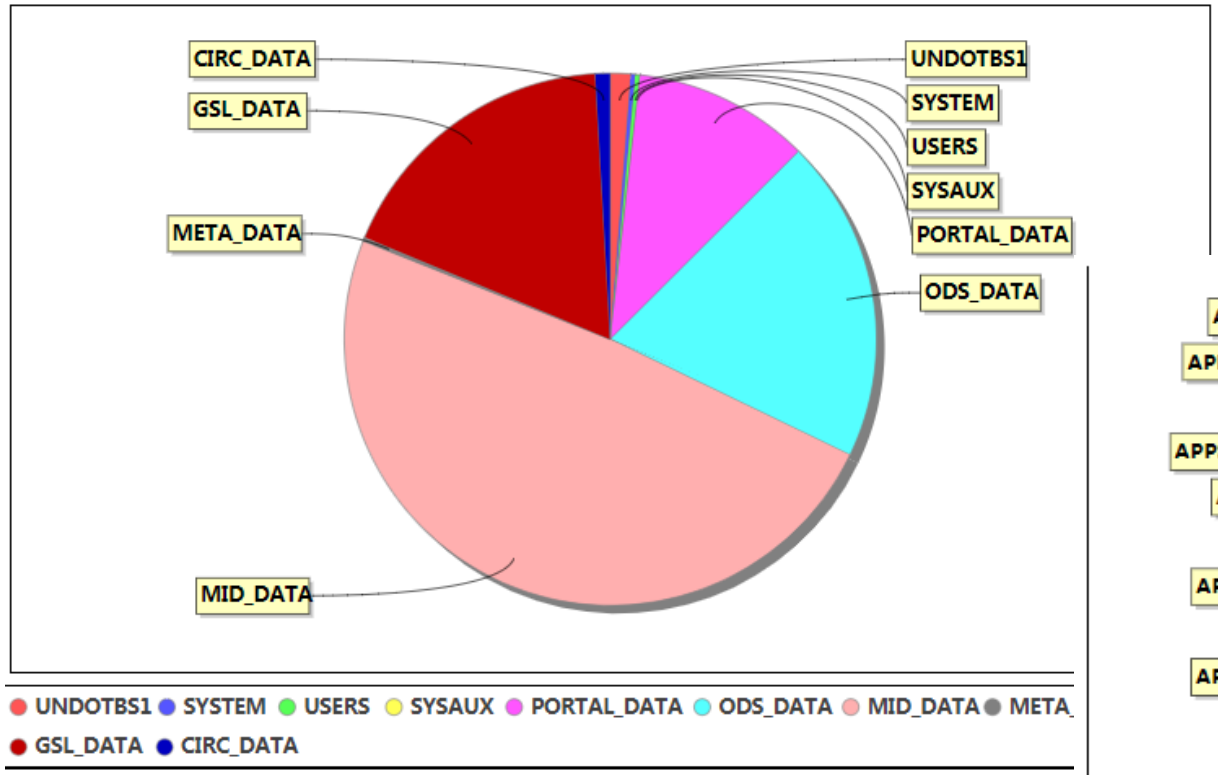
Oracle优化创新

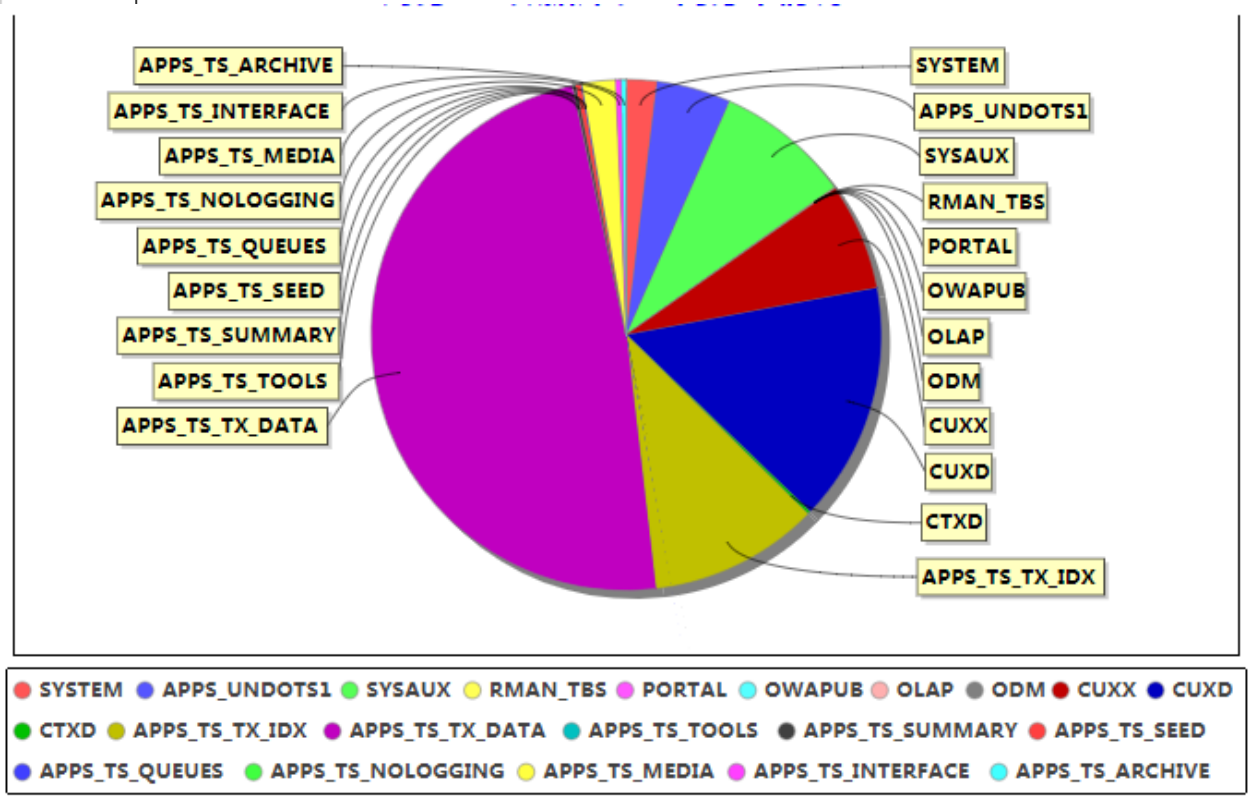
运维

定制运维工具—批量会话监控



定制运维工具—数据使用量分布





创新的云运维

ORACLE Enterprise Manager Cloud Control 12c


ORACLE®