Oracle® Tutor™

Section 5 性能管理（Performance Management）

答不完题

1.Administer Resource Manager

考察 io 方面

2.Use Result Cache

获取时提高性能，考试一个难点，很可能做不出来

Practice 17-1: Using Result Cache

In this practice, you explore the various possibilities to cache query results in the SGA. Perform the following steps to understand the use of Query Result Cache.

1) Change to the `$HOME/solutions/qrc` directory and execute the `result_cache_setup.sh` script.

```
$ cd /home/oracle/solutions/qrc
$
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/acfsmounts/acfs_db1 is
/u01/app/oracle
$ ./result_cache_setup.sh

SQL*Plus: Release 11.2.0.1.0 Production on Wed Oct 7 11:16:29
2009

Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
With the Partitioning, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
SQL> SQL> SQL> SQL> drop user qrc cascade
          *
ERROR at line 1:
ORA-01918: user 'QRC' does not exist

SQL> SQL>   2    3
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL> drop table cachejfv purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> SQL>
Table created.

SQL> SQL>
1 row created.
```

Practice 17-1: Using Result Cache (continued)

```
SQL>
1 row created.

SQL>
2 rows created.

SQL>
4 rows created.

…

SQL>
524288 rows created.

SQL> SQL>
1 row created.

SQL> SQL>
Commit complete.

SQL> SQL>
System altered.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise
Edition Release 11.2.0.1.0 - Production
With the Partitioning, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
$
```

2) In your terminal window log in to SQL*Plus as the `QRC` user. From now on, do not disconnect from this session. Determine the current content of the query cache using the following statement:

```
select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;
```

You can use the `check_result_cache.sql` script. What do you observe? Right now, the query cache should be empty.

```
$ sqlplus qrc/qrc

SQL*Plus: Release 11.2.0.1.0 Production on Wed Oct 7 11:21:43
2009

Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
With the Partitioning, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options

SQL> @check result cache
```

```
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

no rows selected

SQL>
```

3) Set timing on and execute the following query. You can use the `query1.sql` script. Note the time that it takes to execute.

```
select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,
cachejfv c5
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';
```

```
SQL> set timing on
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:01.64
SQL>
```

4) Determine the execution plan of the previous query by using the `explain_query1.sql` script. What do you observe? Because of the `result_cache` hint, the result of the query is computed using the result cache.

```
SQL> @explain_query1
SQL> set echo on
SQL>
SQL> explain plan for select /*+ result_cache q_name(Q1) */
count(*) from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5 where c1.c='b' and c2.c='b' and c3.c='b' and
c4.c='b' and c5.c='b';

Explained.

Elapsed: 00:00:00.20
SQL>
SQL> set linesize 180
SQL>
SQL> @/u01/app/oracle/acfsmounts/acfs_db1/rdbms/admin/utlxpls
SQL> Rem
SQL> Rem $Header: utlxpls.sql 26-feb-2002.19:49:37 bdagevil
```

Practice 17-1: Using Result Cache (continued)

```
..
SQL> Rem
SQL> select plan_table_output from
table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
Plan hash value: 2522916280

--------------------------------------------------------------------
| Id  | Operation                  | Name
| Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------------------------
|   0 | SELECT STATEMENT           |
|     1 |  1260 |  2295G  (1)|999:59:59 |
|   1 |  RESULT CACHE              | c334km80wg4dq0418y5p1frmnz
|       |       |            |          |
|   2 |   SORT AGGREGATE           |
|     1 |  1260 |            |          |
|   3 |    MERGE JOIN CARTESIAN    |
|    92G|   105T|  2295G  (1)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN   |
|   591M|   555G|    14G  (1)|999:59:59 |
|   5 |      MERGE JOIN CARTESIAN  |
|  3794K|  2735M|    94M  (1)|314:36:35 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
----------------------------------------------------------
|   6 |       MERGE JOIN CARTESIAN|
| 24326 |    11M|   605K  (1)| 02:01:03 |
|*  7 |        TABLE ACCESS FULL  | CACHEJFV
|   156 | 39312 |  3856   (1)| 00:00:47 |
|   8 |         BUFFER SORT        |
|   156 | 39312 |   601K  (1)| 02:00:17 |
|*  9 |          TABLE ACCESS FULL | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  10 |        BUFFER SORT         |
|   156 | 39312 |    94M  (1)|314:35:49 |
|* 11 |         TABLE ACCESS FULL  | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  12 |       BUFFER SORT          |
|   156 | 39312 |    14G  (1)|999:59:59 |
|* 13 |        TABLE ACCESS FULL   | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  14 |      BUFFER SORT           |
|   156 | 39312 |  2295G  (1)|999:59:59 |
|* 15 |       TABLE ACCESS FULL    | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
----------------------------------------------------------
```

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

   7 - filter("C1"."C"='b')
   9 - filter("C2"."C"='b')
  11 - filter("C3"."C"='b')
  13 - filter("C4"."C"='b')
  15 - filter("C5"."C"='b')

Result Cache Information (identified by operation id):

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
-----------------------------------------------------------
   1 - column-count=1; dependencies=(QRC.CACHEJFV);
attributes=(single-row); parameters=(nls); name="select /*+
result_cache q_name(Q1) */ count(*) from cachejfv c1,cachejfv
c2,cac
hejfv c3,cachejfv c4,cachejfv c5 where c1.c='b' a"


Note
-----
   - dynamic sampling used for this statement (level=2)

40 rows selected.

Elapsed: 00:00:00.63
SQL>
```

5) Determine the current content of the query cache by using the `check_result_cache.sql` script. What do you observe? You can now see the result of your query cached.

```
SQL> @check_result_cache
SQL> set echo on
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0
```

```
Result     Published select /*+ result_cache q_name(Q1) */
count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5


Elapsed: 00:00:00.00
SQL>
```

6) Flush the buffer cache of your instance and rerun the query executed at step 3. What do you observe? The execution time for the query is now almost instantaneous.

```
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.01
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';
  COUNT(*)
----------
         1

Elapsed: 00:00:00.00
SQL>
```

7) Insert a new row into the `CACHEJFV` table using the following statement: `insert into cachejfv values('c');`. What do you observe? The corresponding result cache entry is automatically invalidated.

```
SQL> insert into cachejfv values('c');

1 row created.

Elapsed: 00:00:00.00
SQL> commit;

Commit complete.

Elapsed: 00:00:00.01
SQL> @check_result_cache
SQL> set echo on
SQL>
```

```
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */
count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5


Elapsed: 00:00:00.00
SQL>
```

8) Execute your first query again and check the result cache. What do you observe? Again, it takes some time to execute the query. Looking at the result cache shows that a new entry has been added for the new result.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:01.36
SQL>
SQL> @check_result_cache
SQL> set echo on
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
```

```
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */
count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5

Result     Published select /*+ result_cache q_name(Q1) */
count(*)
0   1

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5


Elapsed: 00:00:00.01
SQL>
```

9) Generate a detailed result cache memory report.

```
SQL> set serveroutput on
SQL> EXEC DBMS_RESULT_CACHE.MEMORY_REPORT(detailed=>true);
R e s u l t   C a c h e   M e m o r y   R e p o r t
[Parameters]
Block Size          = 1K bytes
Maximum Cache Size  = 2080K bytes (2080 blocks)
Maximum Result Size = 104K bytes (104 blocks)
[Memory]
Total Memory = 107836 bytes [0.041% of the Shared Pool]
... Fixed Memory = 9440 bytes [0.004% of the Shared Pool]
....... Memory Mgr = 124 bytes
....... Bloom Fltr = 2K bytes
....... Cache Mgr  = 4416 bytes
```

```
....... State Objs = 2852 bytes
... Dynamic Memory = 98396 bytes [0.038% of the Shared Pool]
....... Overhead = 65628 bytes
.......... Hash Table    = 32K bytes (4K buckets)
.......... Chunk Ptrs    = 12K bytes (3K slots)
.......... Chunk Maps    = 12K bytes
.......... Miscellaneous = 8284 bytes
....... Cache Memory = 32K bytes (32 blocks)
.......... Unused Memory = 29 blocks
.......... Used Memory = 3 blocks
.............. Dependencies = 1 blocks (1 count)
.............. Results = 2 blocks
................. SQL     = 1 blocks (1 count)
................. Invalid = 1 blocks (1 count)

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.33
SQL>
```

10) Execute your query again. What do you observe? The query again uses the result that was previously cached.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.01
SQL>
```

11) Make sure that you bypass the result cache before executing the next step.

```
SQL> exec DBMS_RESULT_CACHE.BYPASS(bypass_mode=>true);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00
SQL>
```

12) Execute your query again. What do you observe? The query is again taking longer to execute because it no longer uses the result cache.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
```

```
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:01.36
SQL>
```

13) Make sure that you no longer bypass the result cache and verify that your query is using it again.

```
SQL> exec DBMS_RESULT_CACHE.BYPASS(bypass_mode=>false);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.01
SQL>
```

14) Execute the following query by using the `query2.sql` script:

```
select count(*) from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';
```

What do you observe? Although the query is the same as the one used in step 3, it is not recognized as cached because it does not contain the hint. So, its execution time is long again.

```
SQL> @query2
SQL> select count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b';

  COUNT(*)
----------
         1
```

```
Elapsed: 00:00:01.38
SQL>
```

15) How would you force the previous query to use the cached result without using hints? Do it by using the `force_query2.sql` script and then verify that you successfully used the cached result. Finally, undo your change.

```
SQL> @force_query2
SQL> set echo on
SQL>
SQL> show parameter result_cache_mode

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
result_cache_mode                    string      MANUAL
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------- ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */
count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5

Result     Published select /*+ result_cache q_name(Q1) */
count(*)
0   1

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------- ---------- ----------
ROW_SIZE_AVG
------------
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
```

```
                    where c1.c='b' a
          5


Elapsed: 00:00:00.00
SQL>
SQL> alter session set result_cache_mode=force;

Session altered.

Elapsed: 00:00:00.01
SQL>
SQL> explain plan for select count(*) from cachejfv
c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5 where
c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

Explained.

Elapsed: 00:00:00.03
SQL>
SQL> @/u01/app/oracle/acfsmounts/acfs_db1/rdbms/admin/utlxpls
..
SQL> Rem
SQL> select plan_table_output from
table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
Plan hash value: 2522916280

--------------------------------------------------------------------
| Id  | Operation                  | Name
| Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------------------------
|   0 | SELECT STATEMENT           |
|     1 |  1260 |  2295G  (1)|999:59:59 |
|   1 |  RESULT CACHE              | c334km80wg4dq0418y5p1frmnz
|       |       |            |          |
|   2 |   SORT AGGREGATE           |
|     1 |  1260 |            |          |
|   3 |    MERGE JOIN CARTESIAN    |
|    92G|   105T|  2295G  (1)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN   |
|   591M|   555G|    14G  (1)|999:59:59 |
|   5 |      MERGE JOIN CARTESIAN  |
|  3794K|  2735M|    94M  (1)|314:36:35 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
|   6 |       MERGE JOIN CARTESIAN|
| 24326 |    11M|   605K  (1)| 02:01:03 |
```

```
|*  7 |         TABLE ACCESS FULL  | CACHEJFV
|   156 | 39312 |  3856   (1)| 00:00:47 |
|   8 |         BUFFER SORT        |
|   156 | 39312 |   601K  (1)| 02:00:17 |
|*  9 |          TABLE ACCESS FULL | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  10 |        BUFFER SORT         |
|   156 | 39312 |    94M  (1)|314:35:49 |
|* 11 |         TABLE ACCESS FULL  | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  12 |       BUFFER SORT          |
|   156 | 39312 |    14G  (1)|999:59:59 |
|* 13 |        TABLE ACCESS FULL   | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
|  14 |      BUFFER SORT           |
|   156 | 39312 |  2295G  (1)|999:59:59 |
|* 15 |       TABLE ACCESS FULL    | CACHEJFV
|   156 | 39312 |  3855   (1)| 00:00:47 |
----------------------------------------------------------------

PLAN_TABLE_OUTPUT
----------------------------------------------------------------

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

   7 - filter("C1"."C"='b')
   9 - filter("C2"."C"='b')
  11 - filter("C3"."C"='b')
  13 - filter("C4"."C"='b')
  15 - filter("C5"."C"='b')

Result Cache Information (identified by operation id):

PLAN_TABLE_OUTPUT
----------------------------------------------------------------
```

```
   1 - column-count=1; dependencies=(QRC.CACHEJFV);
attributes=(single-row); parameters=(nls); name="select /*+
result_cache q_name(Q1) */ count(*) from cachejfv c1,cachejfv
c2,cac
hejfv c3,cachejfv c4,cachejfv c5 where c1.c='b' a"


Note
-----
   - dynamic sampling used for this statement (level=2)

40 rows selected.
```

Practice 17-1: Using Result Cache (continued)

```
Elapsed: 00:00:00.48
SQL>
SQL> select count(*) from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5 where c1.c='b' and c2.c='b' and
c3.c='b' and c4.c='b' and c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.02
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */
count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5

Result     Published select /*+ result_cache q_name(Q1) */
count(*)
0   1

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5
                     where c1.c='b' a
           5


Elapsed: 00:00:00.01
```

```
SQL>
SQL> alter session set result_cache_mode=manual;

Session altered.

Elapsed: 00:00:00.00
SQL>
```

16) Clear the result cache. Query `V$RESULT_CACHE_OBJECTS` to verify the clear operation.

```
SQL> exec dbms_result_cache.flush;

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

no rows selected

Elapsed: 00:00:00.00
SQL>
```

17) Create a PL/SQL function that uses the result cache by running the `cre_func.sql` script.

```
SQL> @cre_func
SQL> create or replace function CACHEJFV_COUNT(v varchar2)
  2  return number
  3  result_cache relies_on (cachejfv)
  4  is
  5   cnt number;
  6  begin
  7   select count(*) into cnt
  8   from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
  9   where c1.c=v and c2.c=v and c3.c=v and c4.c=v and
c5.c=v;
 10
 11   return cnt;
 12  end;
 13  /

Function created.

Elapsed: 00:00:01.26
SQL>
```

18) Determine what is in the result cache by querying `V$RESULT_CACHE_OBJECTS`.

```
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;
```

```
no rows selected

Elapsed: 00:00:00.00
SQL>
```

19) Call the new function with 'b' as its argument. What do you observe? It takes again a long time to execute because the result is not cached yet. After executing the function, the function's result for argument 'b' is cached.

```
SQL> select cachejfv_count('b') from dual;

CACHEJFV_COUNT('B')
-------------------
                  1

Elapsed: 00:00:02.84
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
----------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Dependency Published QRC.CACHEJFV_COUNT
76961   0
           0

Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4


Elapsed: 00:00:00.00
SQL>
```

20) Call the new function with 'b' as its argument again. What do you observe? This time the function executes almost instantaneously.

```
SQL> select cachejfv_count('b') from dual;

CACHEJFV_COUNT('B')
-------------------
                  1
```

```
Elapsed: 00:00:00.00
SQL>
```

21) Call the new function with 'c' as its argument again. What do you observe? Again it takes a long time to execute the function because of the new value for the argument. After execution, the second result is cached.

```
SQL> select cachejfv_count('c') from dual;

CACHEJFV_COUNT('C')
-------------------
                  1

Elapsed: 00:00:01.11
SQL> select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;

TYPE        STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
76960   0
           0

Dependency Published QRC.CACHEJFV_COUNT
76961   0
           0

Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4


TYPE        STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ----------------------------------------
------------------------------------------------------------
------------------------ ---------- ----------
ROW_SIZE_AVG
------------
Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4

Elapsed: 00:00:00.01
```

Practice 17-1: Using Result Cache (continued)

```
SQL> select cachejfv_count('c') from dual;

CACHEJFV_COUNT('C')
-------------------
                  1

Elapsed: 00:00:00.00
SQL>
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition
Release 11.2.0.1.0 - Production
With the Partitioning, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
$
```

NAME TYPE VALUE ---------- ---------- ------------------------------ client_result_cache_lag big integer 3000 client_result_cache_size big integer 0 result_cache_max_result integer 5 result_cache_max_size big integer 10M result_cache_mode string MANUAL result_cache_remote_expiration integer 0	RESULT_CACHE_MOD #MANUAL 默认不使用，只有加了才使用 #FORCE 默认使用，只有加了才不使用 RESULT_CACHE_MAX_SIZE #设置为 32K 的倍数，如果为 0 则 Result Cache 功能失效 需要和上面的 MOD 配合 RESULT_CACHE_MAX_RESULT #0-100 指定最大内存使用比例

set autotrace on;
select * from COUNTRIES;
	alter system
	set result_cache_max_size =10M;
	alter system set
	result_cache_max_result=80;
	alter system set
	result_cache_mod=FORCE;
	select * from COUNTRIES;

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3996818343

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                 |    25 |   350 |     9   (0)| 00:00:01 |
|   1 |  INDEX FAST FULL SCAN| COUNTRY_C_ID_PK |    25 |   350 |     9   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          6  consistent gets
          0  physical reads
          0  redo size
       1167  bytes sent via SQL*Net to client
        430  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         25  rows processed
```

使用了 result cache 后：

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3996818343

-----------------------------------------------------------------------------------------------------
| Id  | Operation            | Name                        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                             |    25 |   350 |     9   (0)| 00:00:01 |
|   1 |  RESULT CACHE        | 21d03qy1vvhmvgjjbvrramruw6 |       |       |            |          |
|   2 |   INDEX FAST FULL SCAN| COUNTRY_C_ID_PK            |    25 |   350 |     9   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Result Cache Information (identified by operation id):
------------------------------------------------------

   1 - column-count=3; dependencies=(HR.COUNTRIES); name="select * from COUNTRIES"


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          0  consistent gets
          0  physical reads
          0  redo size
       1167  bytes sent via SQL*Net to client
        430  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         25  rows processed
```

3.Use multi column statistics

可能不会考

4.Gather statistics on a specific table without invalidating cursors

SELECT * FROM TABLE(dbms_xplan.display_cursor('cpm9ss48qd32f', 0));

将游标 Keep 到共享池中

SQL> select * from tp3;

SQL> select address,hash_value from v$sql where sql_text like 'select * from tp3';

ADDRESS HASH_VALUE

-------- ----------

2D20D408 1850964723

SQL> exec dbms_shared_pool.keep('2D20D408,1850964723','C');

PL/SQL 过程已成功完成。

注意，'C'不可以省略。

```
DBMS_STATS.GATHER_TABLE_STATS(ownname => USER, tabname => 'CASE_HDR',
method_opt => 'for all indexed columns',
cascade => true,
stattab => 'CASE_HDR_STAT_TAB',
statid => 1);



BEGIN
  DBMS_STATS.DELETE_TABLE_STATS('OE','ORDERS');
  DBMS_STATS.LOCK_TABLE_STATS('OE','ORDERS');
END;
/
```

4):

DBMS_STATS.GATHER_DICTIONARY_STATS

DBMS_STATS.GATHER_SYSTEM_STATS

-- ----------

stats export and import:

```
DBMS_STATS.CREATE_STATS_TABLE (ownname => USER, stattab => 'MY_STAT_TABLE',
tblspace => 'USERS');
exec
dbms_stats.export_database_stats('MY_STAT_TABLE','my_good_stats_001','SYSTEM');
execute dbms_stats.delete_table_stats('SCOTT','EMP')
execute dbms_stats.delete_table_stats('SCOTT','DEPT')
exec
dbms_stats.import_schema_stats('SCOTT','MY_STAT_TABLE','my_good_stats_001','syste
```

m');
exec
dbms_stats.export_database_stats('MY_STAT_TABLE','my_good_stats_002','SYSTEM');
DBMS_STATS.EXPORT_TABLE_STATS(ownname => USER, tabname => 'CASE_HDR',
stattab => 'CASE_HDR_STAT_TAB', statid => 1, cascade => true);
DBMS_STATS.IMPORT_TABLE_STATS(ownname => USER, tabname => 'CASE_HDR',
stattab => 'CASE_HDR_STAT_TAB', statid => 1, cascade => true, no_invalidate =>
true);

5.Use partitioned indexes

CREATE INDEX "HR"."P1_LOCAL" ON "HR"."P1" (a) TABLESPACE "P1" **LOCAL** ; -- 本地 CREATE INDEX "HR"."P1_GLOBAL" ON "HR"."P1" (a) TABLESPACE "P1" ; 全局 create index p2_i1 on p2(b) **global partition by range** (b)– 全局分区索引 (partition part_1 values less than ('30') tablespace users, partition part_2 values less than (maxvalue) tablespace users);
prefixed index 前缀索引，分区关键字在前需要索引的列在后，非前缀相反 create index p2_i on p2(a,b) local ;#本地前缀索引
CREATE INDEX HR.P1_com ON HR.P1 (a,b) compress ; -- 压缩只能在多索引列情况下使用
alter index XX partition P2001 rebuild ;

IOT
创建次要 B*Tree 索引
ALTER INDEX ... UPDATE BLOCK REFERENCES
更新索引，提高猜测命中率。
创建次要 Bitmap 索引
ALTER TABLE ... MAPPING TABLE | NOMAPPING

6.Administer and tune schema object to support various access methods

7.Interpret execution plan

8.Use SQL tuning tools and features

9.Use SQL Tuning Advisor

```
DECLARE
l_sql_tune_task_id VARCHAR2(100);
BEGIN
l_sql_tune_task_id := DBMS_SQLTUNE.create_tuning_task (
sql_id => '2uvvchaax0wzx',
scope => DBMS_SQLTUNE.scope_comprehensive,
time_limit => 600,
task_name => 'tuning_task_1',
description => 'Tuning task for statement 2uvvchaax0wzx');
DBMS_OUTPUT.put_line('l_sql_tune_task_id: ' || l_sql_tune_task_id);
END;
/

EXEC DBMS_SQLTUNE.execute_tuning_task(task_name => 'tuning_task_1');

-- EXEC DBMS_SQLTUNE.reset_tuning_task (task_name => 'tuning_task_1');

SELECT task_name, status FROM dba_advisor_log WHERE owner = 'LINK';

SET LONG 10000;
SET PAGESIZE 1000
SET LINESIZE 200
SELECT DBMS_SQLTUNE.report_tuning_task('tuning_task_1') AS recommendations FROM
dual;
SET PAGESIZE 24

execute dbms_sqltune.accept_sql_profile(task_name => 'tuning_task_1', replace =>
TRUE);


--> SQL profile "SYS_SQLPROF_0144aaae2c088000" used for this statement

-- ---------------------------------------------------------
SELECT decode(r.type,
'STATISTICS', 1,
'SQL PROFILE', 2,
'INDEX', 3, 4) as type,
count(*)
FROM dba_advisor_findings f,
dba_advisor_recommendations r
WHERE f.task_id = r.task_id AND r.task_id = :tid AND
f.finding_id = r.finding_id AND
f.object_id = :oid
GROUP BY r.type
```

```
ORDER BY type
/

-- ----------------------------------------------------------

begin
dbms_sqltune.accept_sql_profile (
task_name => 'FOLIO_COUNT',
name => 'FOLIO_COUNT_PROFILE'
description => 'Folio Count Profile',
category => 'FOLIO_COUNT');
end;


DBMS_SQLTUNE.drop_tuning_task (task_name => 'tuning_task_1');
== =========
```

10.Use SQL Access Advisor

```
--对语句进行分析，给出建立索引 MVIEW IOT 的建议。  Dbms_advisor
begin
dbms_advisor.QUICK_TUNE
( ADVISOR_NAME=>dbms_advisor.sqlaccess_advisor, TASK_NAME=>'mytask3',
ATTR1=>'select * from hr.emp where manager_id=100');
end;
/
```

```
Select DBMS_ADVISOR.GET_TASK_SCRIPT('MYTASK3') from dual;
```

11.Use SQL Performance Analyzer

sql_trace 做来源

SQL Performance Analyzer SPA 常用脚本汇总

SQL 性能分析器 SQL Performance Analyzer SPA

Oracle Database 11g 引入了 SQL 性能分析器；使用该工具可以准确地评估更改对组成工作量的 SQL 语句的影响。SQL 性能分析器可帮助预测潜在的更改对 SQL 查询工作量的性能影响。这种功能可向 DBA 提供有关 SQL 语句性能的详细信息，例如，执行前后的统计信息，提高或降低性能的语句。这样一来，您就可以执行诸如以下操作的操作：在测试环境中进行更改，以确定数据库 升级是否会改进工作量性能。

1. 11g 的新增功能

2. 目标用户：DBA、QA、应用程序开发人员
3. 帮助预测系统更改对 SQL 工作量响应时间的影响
4. 建立不同版本的 SQL 工作量性能（即 SQL 执行计划和执行统计信息）
5. 以串行方式执行 SQL（不考虑并发性）
6. 分析性能差异
7. 提供对单个 SQL 的细粒度性能分析
8. 与 SQL 优化指导集成在一起以优化回归

SQL 性能分析器：使用情形

SQL 性能分析器可用于预测和防止会影响 SQL 执行计划结构的任何数据库环境更改所带来的潜在性能问题。这些更改可以包括（但不限于）以下任何一种更改：

1. 数据库升级
2. 实施优化建议
3. 更改方案
4. 收集统计信息
5. 更改数据库参数
6. 更改操作系统和硬件

DBA 甚至可以使用 SQL 性能分析器为最复杂的环境预测先期更改导致的 SQL 性能更改。例如，随着应用程序在开发周期中的变化，数据库应用程序开发人员可以测试对方案、 数据库对象和重写应用程序的更改，以减轻任何潜在的性能影响。
使用 SQL 性能分析器还可以比较 SQL 性能统计信息。

SQL 性能分析器：概要

1. 收集 SQL：在这个阶段中，将收集用于表示生产系统中的 SQL 工作量的 SQL 语句集。可以使用 SQL 优化集或自动工作量资料档案库 (AWR) 来捕获要传送的信息。因为 AWR 本质上是捕获高负载的 SQL，所以应考虑修改默认的 AWR 快照设置和捕获的顶级 SQL，以确保 AWR 捕获最大数量的 SQL 语句。这可以确保捕获更加完整的 SQL 工作量。

2. 传送：在这个阶段中，应将得到的工作量结果传送到测试系统。从生产系统导出 STS，然后将 STS 导入到测试系统。

3. 计算“之前版本”性能：在进行任何更改之前，执行 SQL 语句，收集评估将来的更改对工作量性能的可能影响所需的基线信息。在此阶段收集的信息给出了系统工作量当前状态的一个快照。性能数据包括：

-执行计划（如由解释计划生成的计划）
-执行统计信息（如由占用时间、缓冲获取次数、磁盘读取次数和已处理的行数组成的信息）

4. 进行更改：获得了之前版本数据后，可以实施计划的更改，然后开始查看对性能的影响。

5. 计算“之后版本”性能：在数据库环境中进行了更改之后才执行此步骤。SQL 工作量的每个语句都在虚拟执行（仅收集统计信息）模式下运行，收集与步骤 3 所捕获的信息相同的信息。

6. 比较和分析 SQL 性能：在获得了两个版本的 SQL 工作量性能数据后，可以通过比较之后版本与之前版本的数据来进行性能分析。比较的根据是执行统计信息，如所用时间、CPU

时间和缓冲区获取次数等。

7. 优化回归的 SQL：在此阶段中，已经准确地确认了哪些 SQL 语句在进行数据库更改时可能导致性能问题。在此阶段中可以使用任何一种数据库工具来优化系统。例如，可以对确认的语句使用 SQL 优化指导或访问指导，然后实施相应的建议。也可以使用在步骤 3 中捕获的计划植入 SQL 计划管理 (SPM) 以确保计划保持不变。在实施了任何优化操作后，应重复该过程来创建新的之后版本，然后分析性能差异以确保新的性能是可接受的。
默认情况下 SPA 若涉及到 DML 语句则只有查询部分 Query 会被执行，但是貌似是从 11.2 开始可以执行完全的 DML 了，需要加入参数 EXECUTE_FULLDML，但是该参数目前有一些 BUG:

Bug 10428438 : WITH EXECUTE_FULLDML ROWS IS ALWAYS SET TO 0 11.2.0.1

Bug 14635522 : SPA SHOULD CAPTURE AND REPLAY TRANSACTIONS 11.2.0.3

By default, only the query portion of DMLs is executed. Using APIs, you can execute the full DML by using the EXECUTE_FULLDML task parameter.EXECUTE_FULLDML when set to TRUE executes DML statement fully, including acquiring row locks and modifying rows; When EXECUTE_FULLDML is set to FALSE (the default value is false) to execute only the query part of the DML without modifying data. When TRUE, SQL Performance Analyzer will issue a rollback following DML execution to prevent persistent changes from being made by the DML. So SPA does not make make any change to the data in the tables.

执行方法如下：

```
execute DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name   => 'TASK_21137', -
                                               parameter   => 'EXECUTE_FULLDML', -
                                               value       => 'TRUE');
```

从 cursor cache 中收集 tuning set, 持续 12 分钟，间隔 5 秒钟

```
begin
DBMS_SQLTUNE.CREATE_SQLSET (sqlset_name => 'MAC_SPA');
dbms_sqltune.capture_cursor_cache_sqlset(
sqlset_name => 'MAC_SPA' ,
time_limit => 12*60,
repeat_interval => 5);
```

```
end ;
/

basic_filter=> q'# module like 'DWH_TEST%' and sql_text not like '%applicat%' and
parsing_schema_name in ('APPS') #'

basic_filter   => 'sql_text LIKE ''%my_objects%'' and parsing_schema_name =
''SPA_TEST_USER''',
```

==>过滤条件使用

从当前 cursor cache 中匹配条件 获得 SQLset ROW

```
SELECT sql_id, sql_text
FROM table(DBMS_SQLTUNE.SELECT_CURSOR_CACHE('buffer_gets > 500'))
ORDER BY sql_id;

SELECT *
FROM table(DBMS_SQLTUNE.SELECT_CURSOR_CACHE('sql_id = ''4rm4183czbs7j'''));

 DECLARE
  cur sys_refcursor;
BEGIN
  OPEN cur FOR
    SELECT value(P)
    FROM table(DBMS_SQLTUNE.SELECT_CURSOR_CACHE) P;

  -- Process each statement (or pass cursor to load_sqlset).

  CLOSE cur;
END;
/

 -- create the tuning set
EXEC DBMS_SQLTUNE.CREATE_SQLSET('MAC_SPA');
-- populate the tuning set from the cursor cache
DECLARE
 cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
 OPEN cur FOR
   SELECT VALUE(P)
```

```
    FROM table(
      DBMS_SQLTUNE.SELECT_CURSOR_CACHE(
        'parsing_schema_name <> ''SYS'' AND elapsed_time > 5000000',
         NULL, NULL, NULL, NULL, 1, NULL,
        'ALL')) P;

DBMS_SQLTUNE.LOAD_SQLSET(sqlset_name => 'MAC_SPA',
                         populate_cursor => cur);

END;
/
```

从 AWR 快照中加载 SQLset ROW 到 SQL TUNING SET

```
DECLARE
  cur sys_refcursor;
BEGIN
  OPEN cur FOR
    SELECT VALUE (P)
    FROM table(dbms_sqltune.select_workload_repository(4146,4161)) P;

  -- Process each statement (or pass cursor to load_sqlset)
  DBMS_SQLTUNE.LOAD_SQLSET(sqlset_name => 'MAC_SPA',
                         populate_cursor => cur);
  CLOSE cur;
END;
/
```

将 SQL TUNING SET Pack 到表中：

```
set echo on
select name,statement_count from dba_sqlset;

drop table maclean.pack_sqlset purge;
```

```
exec DBMS_SQLTUNE.CREATE_STGTAB_SQLSET('PACK_SQLSET','MACLEAN');

exec
DBMS_SQLTUNE.PACK_STGTAB_SQLSET('MAC_SPA','SYS','PACK_SQLSET','MACLEAN');

SQL> desc maclean.pack_sqlset;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(30)
 OWNER                                              VARCHAR2(30)
 DESCRIPTION                                        VARCHAR2(256)
 SQL_ID                                             VARCHAR2(13)
 FORCE_MATCHING_SIGNATURE                           NUMBER
 SQL_TEXT                                           CLOB
 PARSING_SCHEMA_NAME                                VARCHAR2(30)
 BIND_DATA                                          RAW(2000)
 BIND_LIST                                          SQL_BIND_SET
 MODULE                                             VARCHAR2(48)
 ACTION                                             VARCHAR2(32)
 ELAPSED_TIME                                       NUMBER
 CPU_TIME                                           NUMBER
 BUFFER_GETS                                        NUMBER
 DISK_READS                                         NUMBER
 DIRECT_WRITES                                      NUMBER
 ROWS_PROCESSED                                     NUMBER
 FETCHES                                            NUMBER
 EXECUTIONS                                         NUMBER
 END_OF_FETCH_COUNT                                 NUMBER
 OPTIMIZER_COST                                     NUMBER
 OPTIMIZER_ENV                                      RAW(1000)
 PRIORITY                                           NUMBER
 COMMAND_TYPE                                       NUMBER
 FIRST_LOAD_TIME                                    VARCHAR2(19)
 STAT_PERIOD                                        NUMBER
 ACTIVE_STAT_PERIOD                                 NUMBER
 OTHER                                              CLOB
 PLAN_HASH_VALUE                                    NUMBER
 PLAN                                               SQL_PLAN_TABLE_TYPE
 SPARE1                                             NUMBER
 SPARE2                                             NUMBER
 SPARE3                                             BLOB
 SPARE4                                             CLOB
```

将测试对应 schema 的数据和 上述 PACK TABLE 导出导入到 目标测试库中：

```
set echo on
exec
DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET('MAC_SPA','SYS',TRUE,'PACK_SQLSET','MACL
EAN');
alter system flush buffer_cache;
alter system flush shared_pool;
```

创建 SPA 任务 并运行;

```
var sts_task varchar2(64);
exec :sts_task:= dbms_sqlpa.create_analysis_task(task_name =>
'10g_11g_spa',description => 'experiment for 10gR2 to 11gR2 upgrade',sqlset_name=>
'MAC_SPA');

PL/SQL procedure successfully completed.

var exe_task varchar2(64);
exec
:exe_task:=dbms_sqlpa.execute_analysis_task(task_name=>'10g_11g_spa',execution_na
me=>'10g_trail',execution_type=>'CONVERT SQLSET',execution_desc=>'10g sql trail');

var exe_task varchar2(64);
exec
:exe_task:=dbms_sqlpa.execute_analysis_task(task_name=>'10g_11g_spa',execution_na
me=>'11g_trail',execution_type=>'TEST EXECUTE',execution_desc=>'11g sql trail');
```

执行任务比较

比较 CPU_TIME

```
EXEC dbms_sqlpa.execute_analysis_task( -
  task_name => '10g_11g_spa', -
  execution_name => 'compare_10g_112_cpu', -
  execution_type => 'COMPARE PERFORMANCE', -
  execution_params =>
dbms_advisor.arglist('COMPARISON_METRIC','CPU_TIME','EXECUTION_NAME1','10g_trail','
EXECUTION_NAME2','11g_trail'), -
  execution_desc => 'Compare 10g SQL Trace Performance to 11g Test-Execute for
CPU_TIME')
  /
```

比较 BUFFER_GETS

```
EXEC dbms_sqlpa.execute_analysis_task( -
  task_name => '10g_11g_spa', -
  execution_name => 'compare_10g_112_buffergets', -
  execution_type => 'COMPARE PERFORMANCE', -
  execution_params =>
dbms_advisor.arglist('COMPARISON_METRIC','BUFFER_GETS','EXECUTION_NAME1','10g_tr
ail','EXECUTION_NAME2','11g_trail'), -
  execution_desc => 'Compare 10g SQL Trace Performance to 11g Test-Execute for
BUFFER_GETS')
  /
```

比较实际执行时长

```
begin
DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
task_name => 'SPA_TEST',
execution_type => 'COMPARE PERFORMANCE',
execution_name => 'Compare_elapsed_time',
execution_params => dbms_advisor.arglist('execution_name1', '10g_trail',
'execution_name2', '11g_trail', 'comparison_metric', 'elapsed_time') );
end;
/
```

比较物理读

```
begin
DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
task_name => '10g_11g_spa',
execution_type => 'COMPARE PERFORMANCE',
```

```
execution_name => 'Compare_physical_reads0',
execution_params => dbms_advisor.arglist('execution_name1', '10g_trail',
'execution_name2', '11g_trail', 'comparison_metric', 'disk_reads') );
end;
/
```

Set the comparison_metric parameter to specify an expression of execution statistics to use in the performance impact analysis. Possible values include the following metrics or any combination of them: elapsed_time (default), cpu_time, buffer_gets, disk_reads, direct_writes, and optimizer_cost.

获得 SPA 报告:

```
set long 100000 longchunksize 100000 linesize 200 head off feedback off echo off
spool spa_report_elapsed_time.html
SELECT dbms_sqlpa.report_analysis_task('SPA_TEST', 'HTML', 'ALL','ALL',
execution_name=>'Compare_elapsed_time') FROM dual;
spool off
```

产生 buffergets 比较 report

```
set heading off long 100000000 longchunksize 10000 echo off;
set linesize 1000 trimspool on;
spool buffergets_summary.html
select xmltype(dbms_sqlpa.report_analysis_task('10g_11g_spa',
                                               'html',
                                               'typical',
                                               'all',
                                               null,
                                               100,
                                               'compare_10g_112_buffergets')).getclobval(0,0)
from dual;
spool off
```

产生 errors 比较 report

```
spool errors_summary.html
select xmltype(dbms_sqlpa.report_analysis_task('10g_11g_spa',
```

```
                                    'html',
                                    'errors',
                                    'summary',
                                    null,
                                    100,
                                    '11g_trail')).getclobval(0,0)
from dual;
spool off

产生 unsupport 比较 report
spool unsuppor_all.html
select xmltype(dbms_sqlpa.report_analysis_task('10g_11g_spa',
                                    'html',
                                    'unsupported',
                                    'all',
                                    null,
                                    100,
                                    '11g_trail')).getclobval(0,0)
from dual;
spool off
```

execution_type
Type of the action to perform by the function. If NULL it will default to the value of the DEFAULT_EXECUTION_TYPE parameter. Possible values are:
[TEST] EXECUTE – test-execute every SQL statement and collect its execution plans and execution statistics. The resulting plans and statistics will be stored in the advisor framework. This is default.
EXPLAIN PLAN – generate explain plan for every statement in the SQL workload. This is similar to the EXPLAIN PLAN command. The resulting plans will be stored in the advisor framework in association with the task.
COMPARE [PERFORMANCE] – analyze and compare two versions of SQL performance data. The performance data is generated by test-executing or generating explain plan of the SQL statements. Use this option when two executions of type EXPLAIN_PLAN or TEST_EXECUTE already exist in the task
CONVERT SQLSET – used to read the statistics captured in a SQL Tuning Set and model them as a task execution. This can be used when you wish to avoid executing the SQL statements because valid data for the experiment already exists in the SQL Tuning Set.

For 9i Upgrade to 10g

exec dbms_stats.gather_system_stats(gathering_mode=>'NOWORKLOAD');

alter system set "_optim_peek_user_binds"=false; ==> 禁用 BIND PEEK 特性，该特性在 10g 中有

exec DBMS_STATS.SET_PARAM('method_opt','FOR ALL COLUMNS SIZE 1');
commit;

9i

?/rdbms/admin/dbmssupp

exec dbms_support.start_trace(binds=>TRUE, waits=> FALSE);

exec dbms_support.stop_trace;

exec dbms_support.start_trace_in_session(sid=>sid,serial=>ser, binds=>TRUE, waits=>FALSE);

select sid,serial# from v$SESSION WHERE ... ;

exec dbms_support.stop_trace_in_session(sid=>SID,serial=>ser);

```
create table mapping_table tablespace USERS as
select object_id id, owner, substr(object_name, 1, 30) name
  from dba_objects
 where object_type not in ('CONSUMER GROUP',
                           'EVALUATION CONTEXT',
                           'FUNCTION',
                           'INDEXTYPE',
                           'JAVA CLASS',
                           'JAVA DATA',
                           'JAVA RESOURCE',
                           'LIBRARY',
                           'LOB',
                           'OPERATOR',
                           'PACKAGE',
```

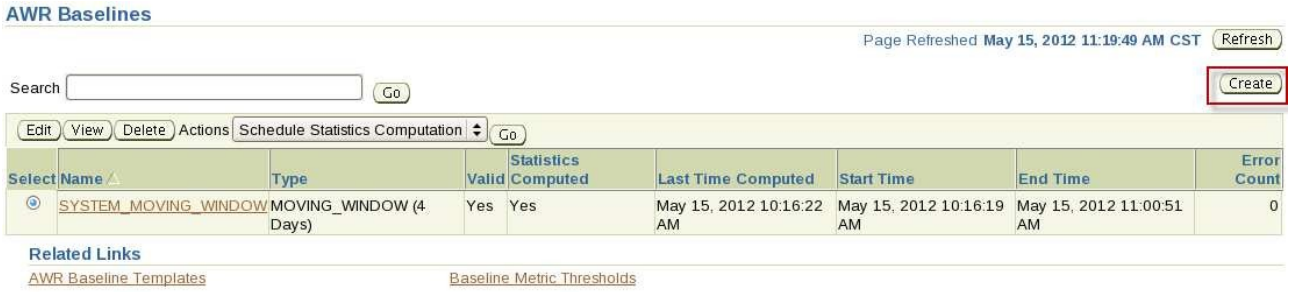
```
                        'PACKAGE BODY',
                        'PROCEDURE',
                        'QUEUE',
                        'RESOURCE PLAN',
                        'SYNONYM',
                        'TRIGGER',
                        'TYPE',
                        'TYPE BODY')
union all
select user_id id, username owner, null name from dba_users;

declare
  mycur dbms_sqltune.sqlset_cursor;
begin
  dbms_sqltune.create_sqlset('9i_prod_wkld');
    open mycur for
      select value(p)
      from table(dbms_sqltune.select_sql_trace(
                directory=>'SPADIR',
                file_name=>'%trc',
                mapping_table_name => 'MAPPING_TABLE',
                select_mode => dbms_sqltune.single_execution)) p;
  dbms_sqltune.load_sqlset(
    sqlset_name => '9i_prod_wkld',
    populate_cursor => mycur,
    commit_rows => 1000);

  close mycur;
end;
/

create user spadba identified by oracle;
grant dba to spadba;
grant all on dbms_sqlpa to spadba;

create public database link to10g connect to spadba identified by oracle using 'STRINGS';

var sts_task varchar2(64);
exec :sts_task:= dbms_sqlpa.create_analysis_task(task_name =>
'9i_11g_spa1',description => 'experiment for 9i to 11gR2 upgrade',sqlset_name=>
'9i_prod_wkld');

var exe_task varchar2(64);
exec
```

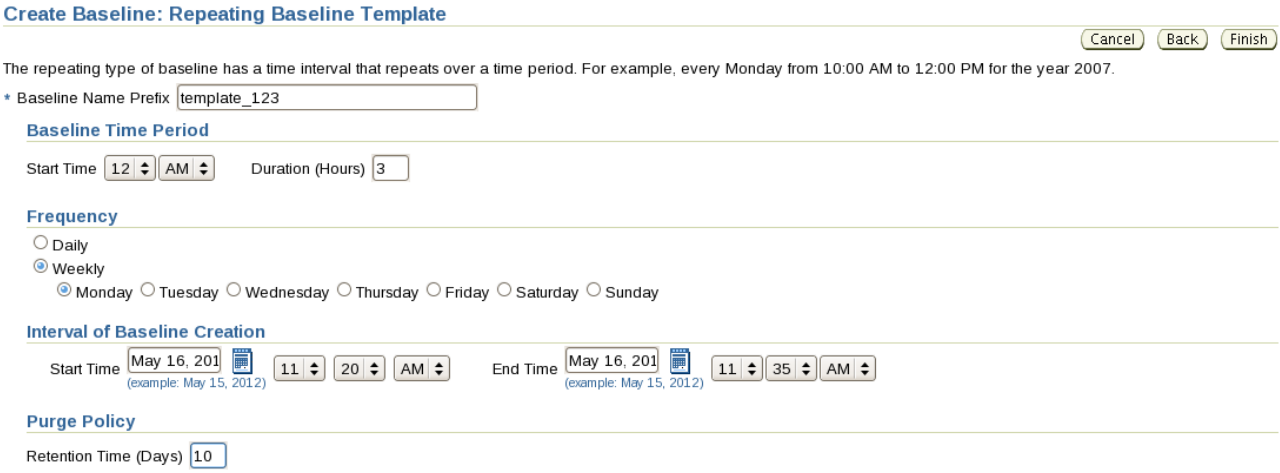
:exe_task:=dbms_sqlpa.execute_analysis_task(task_name=>'9i_11g_spa1',execution_name=>'9i_trail1',execution_type=>'CONVERT SQLSET',execution_desc=>'9i sql trail generated from sts');

dbms_sqlpa.execute_analysis_task(task_name=>'9i_11g_spa1',execution_name=>'10g_trail1',execution_type=>'TEST EXECUTE',execution_desc=>'10g trail test',-
execution_params=>dbms_advisor.arglist('DATABASE_LINK','DBLINKNAME'));

select sofar,totalwork from V$ADVISOR_PROGRESS where task_id=<TID>;

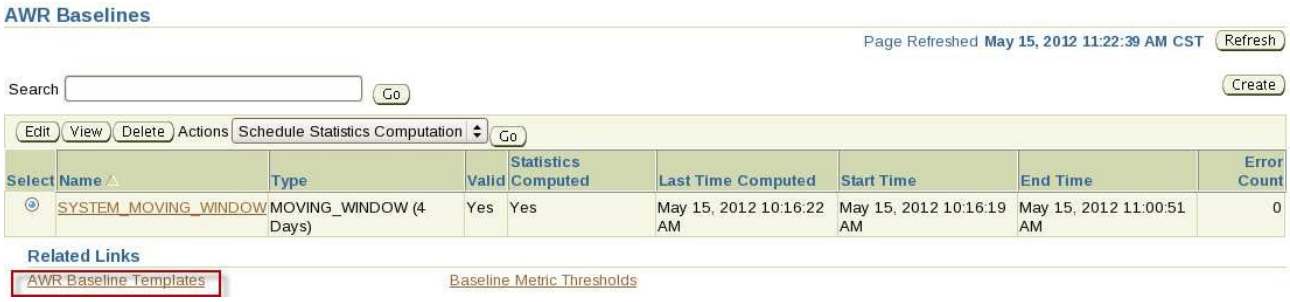
12.Configure baseline templates

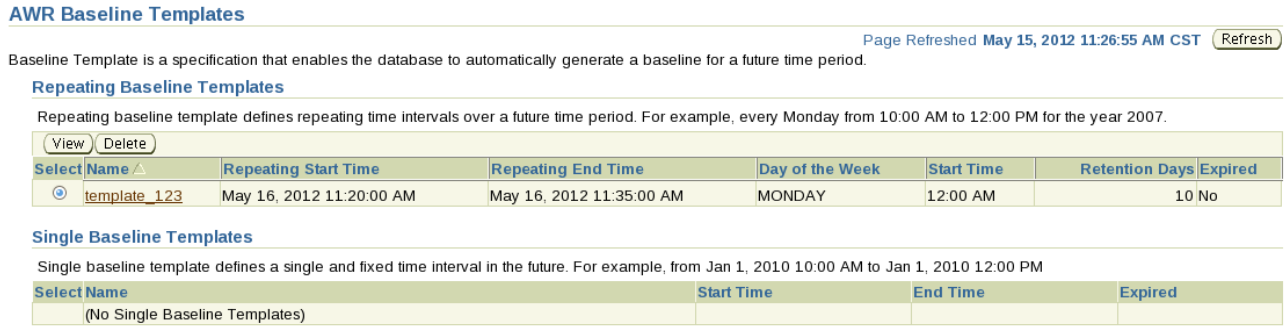




创建一个自动重复的 baseline 模板 select dbid from v$database; select template_name from DBA_HIST_BASELINE_TEMPLATE	创建单次 baseline 模板脚本 DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE(start_time IN DATE, end_time IN DATE, baseline_name IN VARCHAR2, template_name IN VARCHAR2, expiration IN NUMBER, dbid IN NUMBER DEFAULT NULL); 创建自动重复 baseline 模板脚本 DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE(day_of_weekIN VARCHAR2, hour_in_day IN NUMBER, duration IN NUMBER, start_time IN DATE, end_time IN DATE, baseline_name_prefix IN VARCHAR2, template_name IN VARCHAR2, expiration IN NUMBER, dbid IN NUMBER DEFAULT NULL); 删除模板 DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE_TEMPLATE(template_name IN VARCHAR2, dbid IN NUMBER DEFAULT NULL);

```
-- 查看一下
```





8 Resolving Performance Degradation Over Time

Performance degradation of the database over time happens when your database was performing optimally in the past, such as 6 months ago, but has gradually degraded to a point where it becomes noticeable to the users. The Automatic Workload Repository (AWR) Compare Periods report enables you to compare database performance between two periods of time.

While an AWR report shows AWR data between two snapshots (or two points in time), the AWR Compare Periods report shows the difference between two periods (or two AWR reports, which equates to four snapshots). Using the AWR Compare Periods report helps you to identify detailed performance attributes and configuration settings that differ between two time periods. The two time periods selected for the AWR Compare Periods report can be of different durations. The report normalizes the statistics by the amount of time spent on the database for each time period and presents statistical data ordered by the largest difference between the periods.

For example, a batch workload that historically completed in the maintenance window between 10:00 p.m. and midnight is currently showing poor performance and completing at 2 a.m. You can generate an AWR Compare Periods report from 10:00 p.m. to midnight on a day when performance was good and from 10:00 a.m. to 2 a.m. on a day when performance was poor. The comparison of these reports should identify configuration settings, workload profile, and statistics that were different in these two time periods. Based on the differences identified, you can more easily diagnose the cause of the performance degradation.

This chapter contains the following sections:

- Managing Baselines
- Running the AWR Compare Periods Reports

- Using the AWR Compare Periods Reports

See Also:

- "Gathering Database Statistics Using the Automatic Workload Repository"

Managing Baselines

Baselines are an effective way to diagnose performance problems. AWR supports the capture of baseline data by enabling you to specify and preserve a pair or a range of snapshots as a baseline. The snapshots contained in a baseline are excluded from the automatic AWR purging process and are retained indefinitely.

A moving window baseline corresponds to all AWR data that exists within the AWR retention period. Oracle Database automatically maintains a system-defined moving window baseline. The default size of the window is the current AWR retention period, which by default is 8 days.

This section contains the following topics:

- Creating a Baseline
- Deleting a Baseline
- Computing Threshold Statistics for Baselines
- Setting Metric Thresholds for Baselines

Creating a Baseline

Before creating a baseline, carefully consider the time period you choose as a baseline because it should represent the database operating at an optimal level. In the future, you can compare these baselines with other baselines or snapshots captured during periods of poor performance to analyze performance degradation over time.

You can create the following types of baseline:

- Creating a Single Baseline
- Creating a Repeating Baseline

Creating a Single Baseline

A single baseline is captured at a single, fixed time interval. For example, a single baseline may be captured on March 5, 2007 from 5:00 p.m. to 8:00 p.m.

You can choose a start time and an end time that are in the future to create a baseline that captures future database activity. If both the start time and the end time are in the future, a baseline template with the same name as the baseline will also be created. A baseline template is a specification that enables Oracle Database to automatically generate a baseline for a future time period.

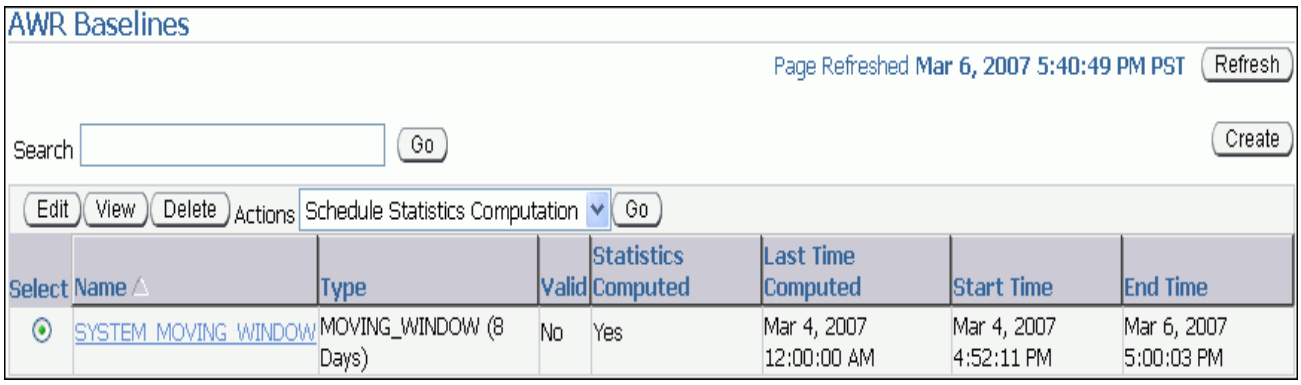
To create a single baseline:

1. From the Database Home page, click Server.

   The Server subpage appears.

2. Under Statistics Management, click AWR Baselines.

   The AWR Baselines page appears with a list of existing baselines displayed.

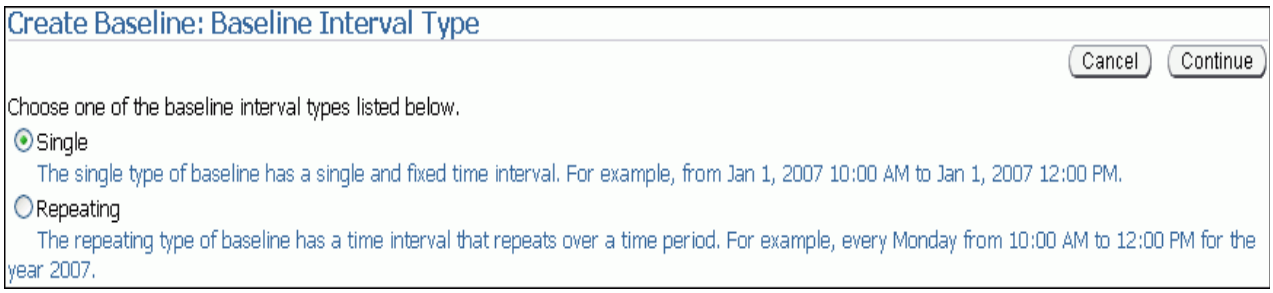


Description of the illustration awr_baselines.gif

3. Click Create.

   The Create Baseline: Baseline Interval Type page appears.

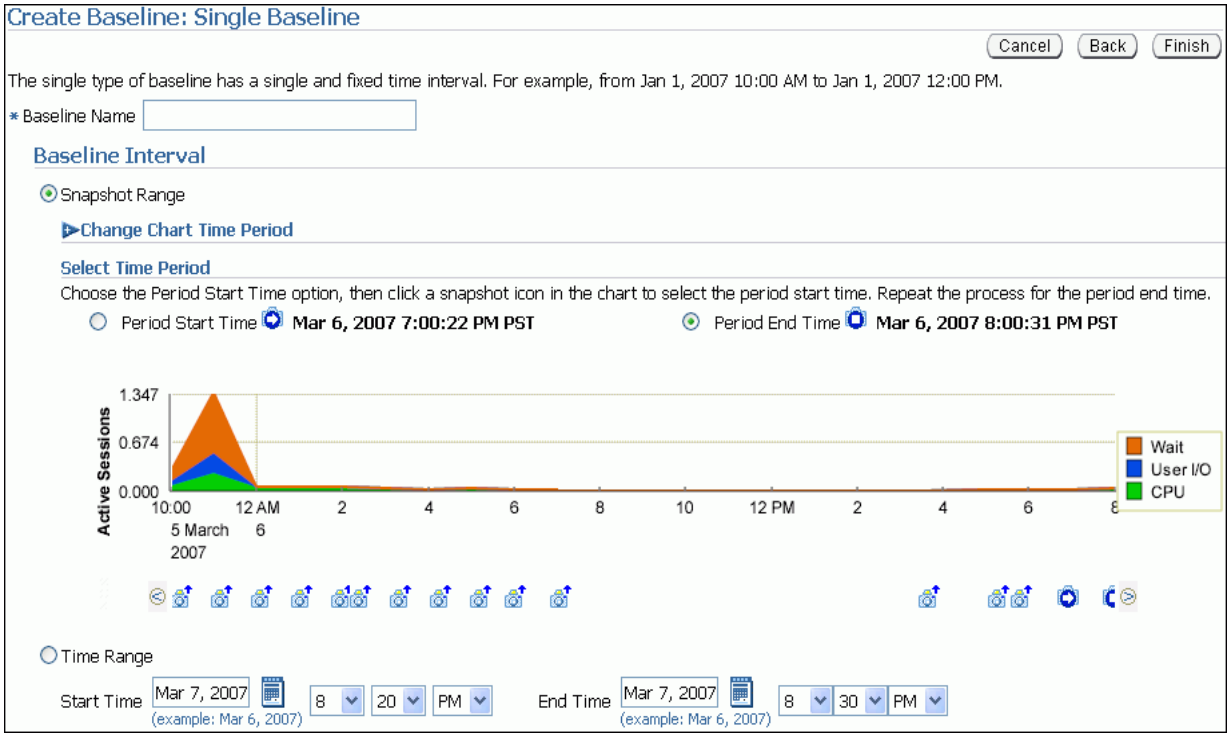
4. Select Single.



Description of the illustration baseline_interval_single.gif

5. Click Continue.

   The Create Baseline: Single Baseline page appears.

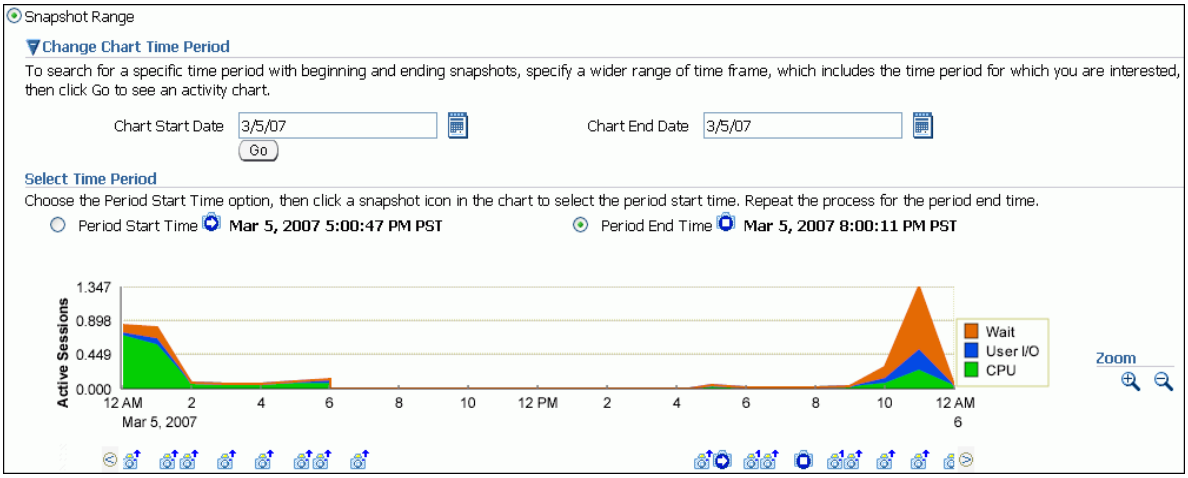


Description of the illustration create_baseline_single.gif

6. In the Baseline Name field, enter a name for the baseline.
7. Under Baseline Interval, select whether to use a snapshot range or a time range for the baseline. Do one of the following:

- To use a snapshot range, select Snapshot Range. Complete the following steps:
  - Optionally, to view older snapshots that are not displayed below the Active Sessions chart, expand Change Chart Time Period. Enter the desired start date in the Chart Start Date field and the desired end date in the Chart End Date field, and click Go.
  - Under Select Time Period, select a start time for the baseline by selecting Period Start Time and the snapshot icon below the Active Sessions chart that corresponds to the desired start time.
  - Select an end time for the baseline by selecting Period End Time and the snapshot icon below the Active Sessions chart that corresponds to the desired end time.

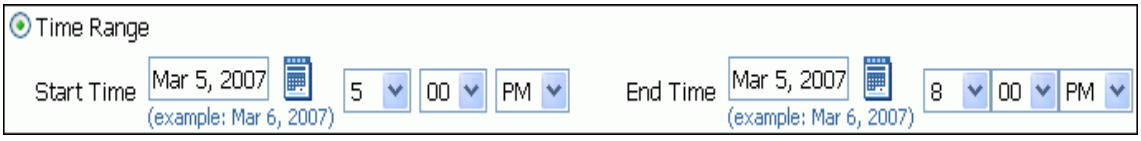
  In this example, a snapshot range on March 5, 2007 from 5:00 p.m. to 8:00 p.m. is selected.



Description of the illustration baseline_snapshot_range.gif

- To use a time range, select Time Range. Complete the following steps:
  - In the Start Time fields, select a start time for the baseline.
  - In the End Time fields, select an end time for the baseline.

  In this example, a time range from 5:00 p.m. to 8:00 p.m. on March 5, 2007 is selected.



Description of the illustration create_baseline_time_range.gif

8. Click Finish.

   The AWR Baselines page reappears with the newly created baseline displayed.

Creating a Repeating Baseline

A repeating baseline is a baseline that repeats during a time interval over a specific period. For example, a repeating baseline may repeat every Monday from 5:00 p.m. to 8:00 p.m. for the year 2007.

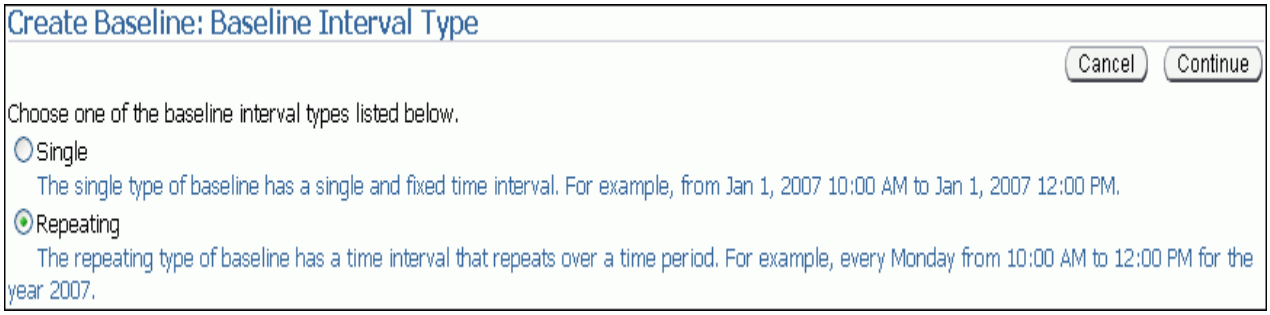
To create a repeating baseline:

1. From the Database Home page, click Server.

   The Server subpage appears.
2. Under Statistics Management, click AWR Baselines.

   The AWR Baselines page appears with a list of existing baselines displayed.
3. Click Create.

   The Create Baseline: Baseline Interval Type page appears.



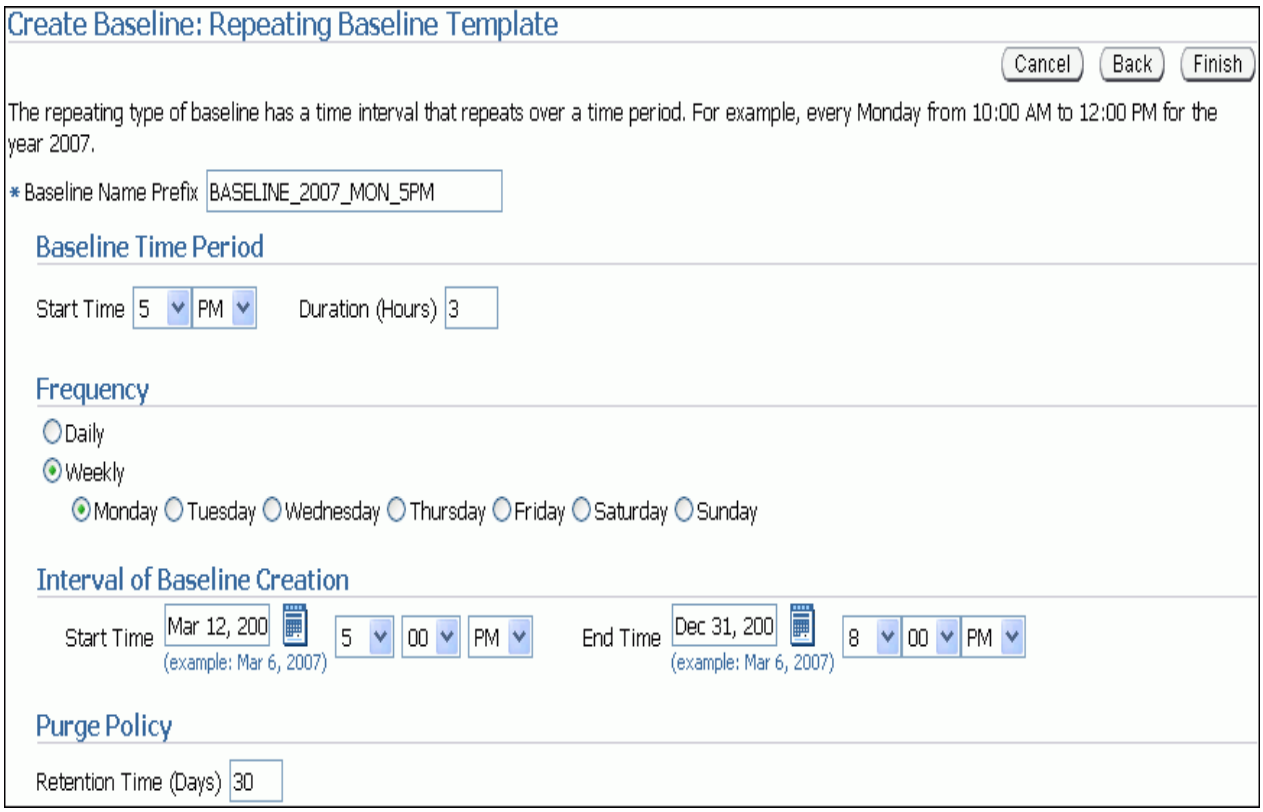
Description of the illustration baseline_interval_repeating.gif

4. Select Repeating and then click Continue.

   The Create Baseline: Repeating Baseline Template page appears.
5. In the Baseline Name Prefix field, enter a name prefix for the baseline.
6. Under Baseline Time Period, specify the time of the day that you want the baseline to begin collecting AWR data and the duration of the baseline collection.
7. Under Frequency, do one of the following:
   - Select Daily if you want the baseline to repeat on a daily basis.
   - Select Weekly if you want the baseline to repeat on a weekly basis and select the day of the week on which the baseline will repeat.
8. Under Interval of Baseline Creation, complete the following steps:
   - In the Start Time fields, select a date and time in the future when the data collection should begin.
   - In the End Time fields, select a date and time in the future when the data collection should end.
9. Under Purge Policy, enter the number of days to retain baselines that have been captured.
10. Click Finish.

    A baseline template with the same name as the baseline name prefix will also be created. A baseline template is a specification that enables Oracle Database to automatically generate a baseline for a future time period.

    In this example, a repeating baseline that repeats weekly on Mondays from 5:00 p.m. to 8:00 p.m. for the year 2007 will be created.



Description of the illustration create_baseline_repeating.gif

Deleting a Baseline

To conserve storage space, you may want to periodically delete unused baselines stored in the database.

To delete a baseline:

1. From the Database Home page, click Server.

   The Server subpage appears.

2. Under Statistics Management, click AWR Baselines.

   The AWR Baselines page appears with a list of existing baselines displayed.

3. Select a baseline and click Delete.

   The Confirmation page appears.

4. Select whether to purge the underlying data associated with the baseline.

   The underlying data includes the individual snapshots preserved in the baseline and any statistics that are computed for the baseline. Do one of the following:

   - To delete the underlying data, select Purge the underlying data associated with the baseline.
   - To preserve the underlying data, select Do not purge the underlying data associated with the baseline.

5. Click Yes.

   The AWR Baselines page reappears. A message informs you that the baseline was deleted successfully.

Computing Threshold Statistics for Baselines

Computing threshold statistics for baselines enables you to graphically display the computed statistics in the charts on the Performance page.

To compute threshold statistics for baselines:

1. From the Database Home page, click Server.

   The Server subpage appears.

2. Under Statistics Management, click AWR Baselines.

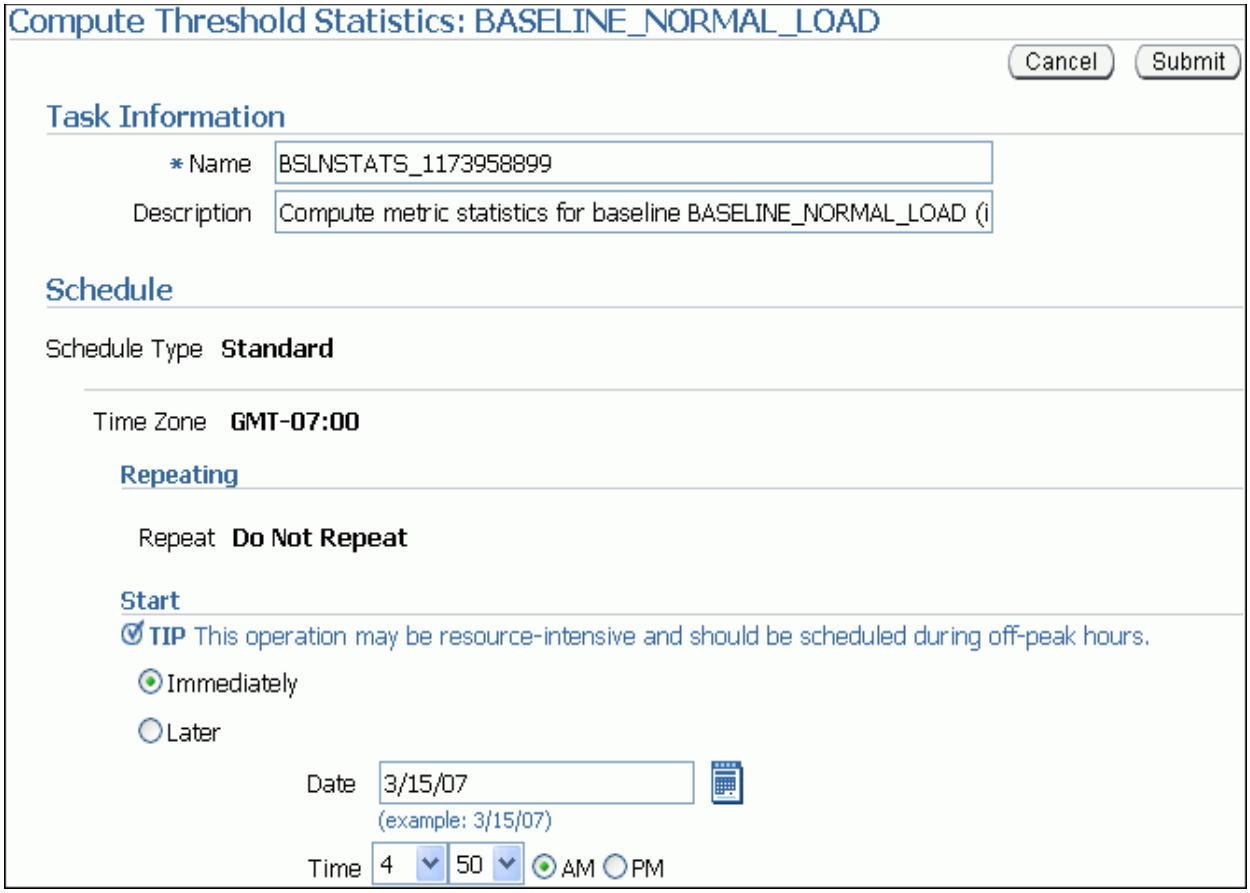
   The AWR Baselines page appears with a list of existing baselines displayed.

3. Select the baseline for which you want to compute statistics.

   Select a baseline that does not already have computed statistics. These baselines are identified by No in the Statistics Computed column.

4. From the Actions list, select Schedule Statistics Computation, and then click Go.

   The Compute Threshold Statistics page appears.



Description of the illustration compute_threshold_stats.gif

5. In the Name field, enter a name for the task.

   Alternatively, you can choose to use the system-generated name.

6. In the Description field, enter a description for the task.

   Alternatively, you can choose to use the system-generated description.

7. Under Start, do one of the following:

   - Select Immediately to run the task immediately after it has been submitted.

- Select Later to run the task at a later time as specified using the Date and Time fields.

8. Click Submit.

   The AWR Baselines page appears. A message informs you that statistics computation has been scheduled for the selected baseline.

See Also:

- "Customizing the Database Performance Page" for information about displaying computed statistics on the Performance page
- Oracle Database 2 Day DBA for information about thresholds and how to manage them

Setting Metric Thresholds for Baselines

As explained in "Setting Metric Thresholds for Performance Alerts", a metric is the rate of change in a cumulative statistic. Alerts notify you when particular metric thresholds are crossed. When the metric thresholds are crossed, the system is in an undesirable state. You can edit the threshold settings for baseline metrics.

You can create the following types of baseline:

- Setting Metric Thresholds for the Default Moving Baseline
- Setting Metric Thresholds for Selected Baselines

Setting Metric Thresholds for the Default Moving Baseline

This section describes the easiest technique for setting the metric thresholds for the default moving baseline. You can choose a group of basic metric threshold settings based on common database workload profiles: OLTP, data warehousing, and OLTP with nighttime batch jobs. After choosing a workload profile, you can expand or change the threshold values as needed.

To set metric thresholds for the default moving baseline:

1. On the Database Home page, under Related Links, click Baseline Metric Thresholds.

   The Threshold Configuration tab of the Baseline Metric Thresholds page appears.

2. Click Quick Configuration.

   The Quick Configuration: Baseline Metric Thresholds page appears.

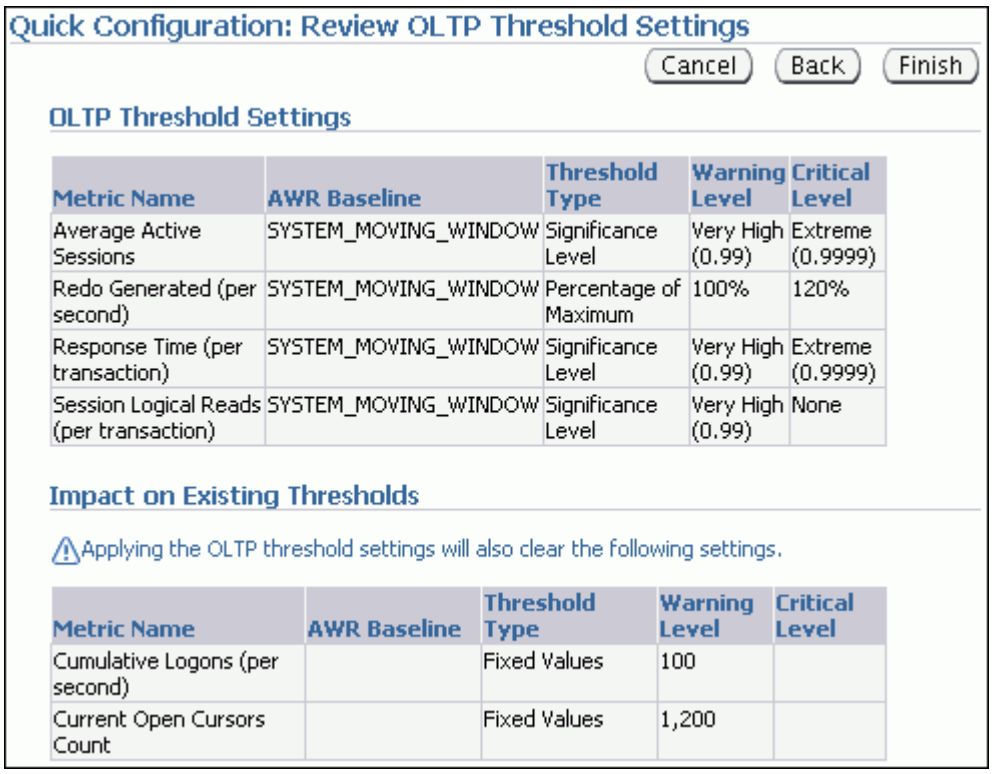
3. In Workload Profile, select one of the following options, depending on how you are using the database:

   - Primarily OLTP (pure transaction processing 24 hours a day)
   - Primarily Data Warehousing (query and load intensive)
   - Alternating (OLTP during the daytime and batch during the nighttime)

   In this example, select Primarily OLTP.

4. Click Continue.

   The Quick Configuration: Review OLTP Threshold Settings page appears.



Description of the illustration threshold_quick.gif

5. Review the metric threshold settings and click Finish.

   You are returned to the Baseline Metric Thresholds page, with the Threshold Configuration tab selected. The metric threshold settings are displayed.

Setting Metric Thresholds for Selected Baselines

This section explains how to select a baseline and edit its thresholds. You can configure the type of threshold, for example, whether it is based on significance levels, percentage of maximum values, or fixed values. You can also configure the threshold levels that determine when the system generates critical alerts and warnings.

You can edit thresholds for the default moving baseline or a baseline that you created in the AWR Baselines page. You can select a baseline in the Edit Thresholds page after you have scheduled statistics computation from the AWR Baselines page and the statistics have finished computing on the static baseline.

To set a metric threshold for the default moving baseline:

1. On the Database Home page, under Related Links, click Baseline Metric Thresholds.

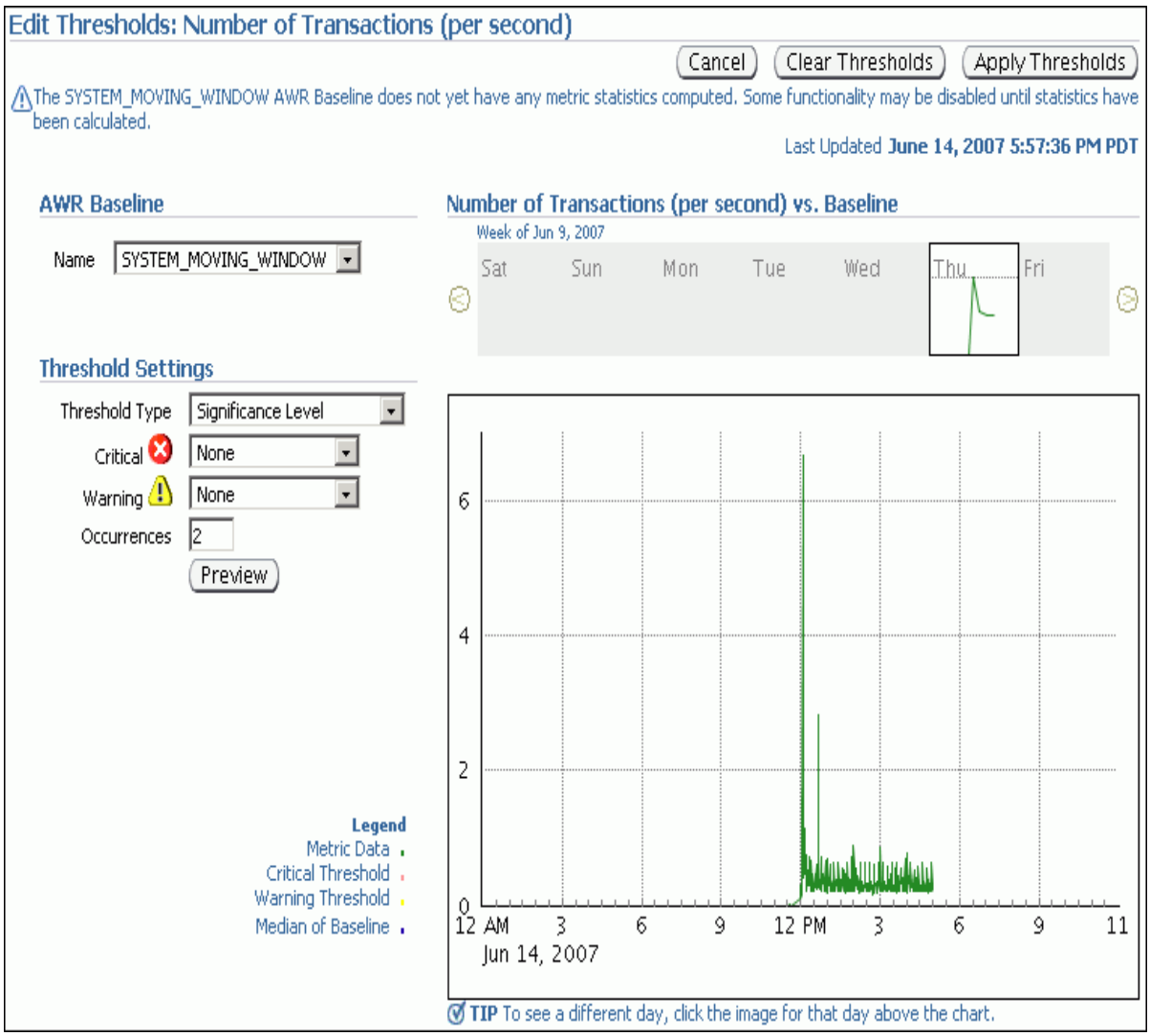
   The Threshold Configuration tab of the Baseline Metric Thresholds page appears.

2. In the View list, select Basic Metrics.

   The Baseline Metric Thresholds page appears.

3. In the Category/Name column, click the link for the metric whose threshold you want to set or change. For example, click Number of Transactions (per second).

   The Edit Thresholds: Number of Transactions (per second) appears.



Description of the illustration edit_thresholds.gif

The charts on this page provide thumbnail and detailed views of metric activity for a 24-hour period. In the top thumbnail chart, click a day to view the value of the metric plotted against a 24-hour period.

4. Under AWR Baseline, in the Name list, select either the default SYSTEM_MOVING_WINDOW or the name of a baseline created in the AWR Baselines page.

   A baseline appears in the AWR Baseline list after you have scheduled statistics computation from the AWR Baselines page and the statistics have finished computing on the static baseline.

   In this example, select AWR_BASELINES_2007.

   The page refreshes to show the charts for the baseline that you selected.

5. In the Threshold Settings section, complete the following steps:
   1. In Threshold Type, leave Significance Level selected.
   2. In Critical, select Extreme.
   3. In Warning, select Very High.
   4. In Occurrences, leave the current value.
6. Click Apply Thresholds.

You are returned to the Baseline Metric Thresholds page. This page shows the altered metric threshold settings.

Running the AWR Compare Periods Reports

This section describes how to run the Automatic Workload Repository (AWR) Compare Periods reports using Oracle Enterprise Manager.

You can use AWR Compare Periods reports to compare the database performance between two time periods by:

- Comparing a Baseline to Another Baseline or Pair of Snapshots
- Comparing Two Pairs of Snapshots

Comparing a Baseline to Another Baseline or Pair of Snapshots

When performance degradation happens to a database over time, you should run the AWR Compare Periods report to compare the degraded performance, captured as a new baseline or a pair of snapshots, to an existing baseline. You will need a baseline that represents the system operating at an optimal level. If an existing baseline is not available, then you can compare database performance between two periods of time by using two arbitrary pairs of snapshots, as described in "Comparing Two Pairs of Snapshots".

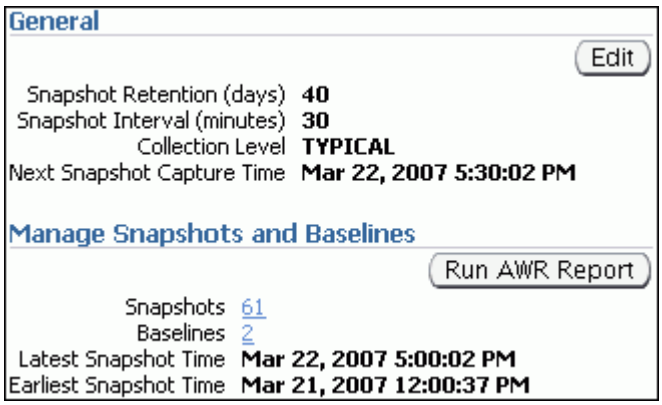
To compare a baseline to another baseline:

1. From the Database Home page, click Server.

   The Server subpage appears.

2. Under Statistics Management, click Automatic Workload Repository.

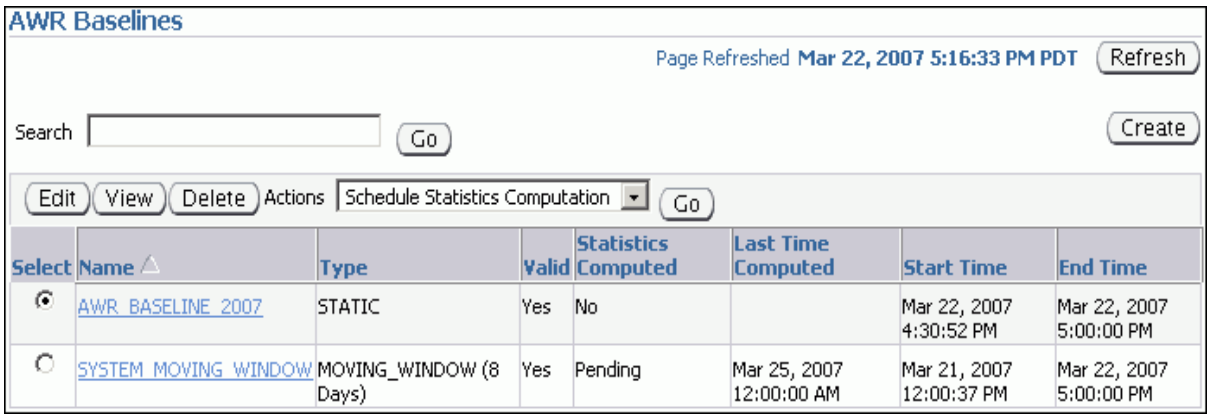
   The Automatic Workload Repository page appears.



Description of the illustration awr1.gif

3. Under Manage Snapshots and Baselines, click the link next to Baselines.

   The AWR Baselines page appears.



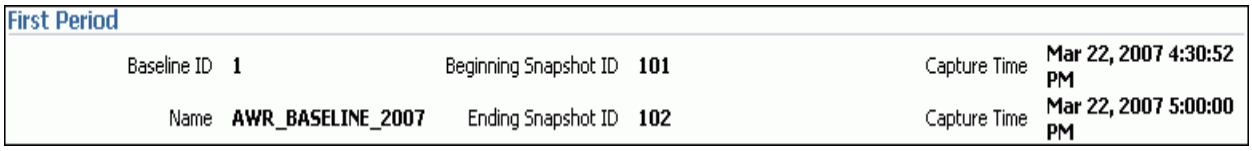
Description of the illustration awr_baselines_comp.gif

4. Complete the following steps:
    1. Select the baseline to use for the report.

       At least one existing baseline must be available.
    2. From the Actions list, select Compare Periods and click Go.

    The Compare Periods: Second Period Start page appears. Under First Period, the selected baseline is displayed.

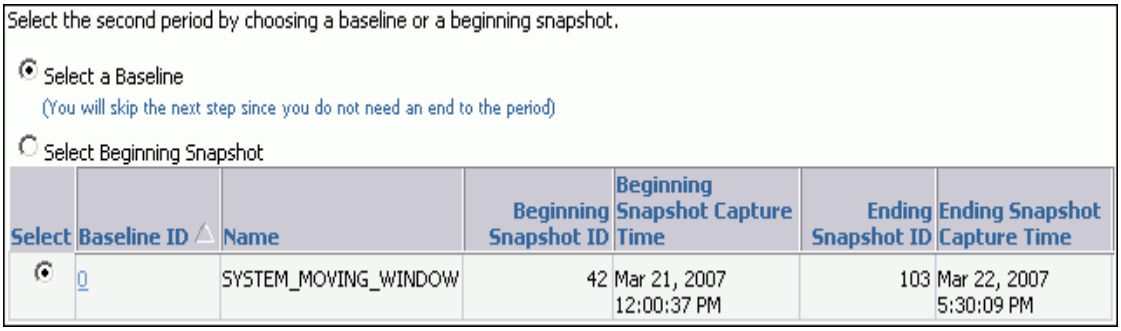
    In this example, the baseline named `AWR_BASELINE_2007` is selected.



Description of the illustration baseline_first_period.gif

5. Compare the baseline selected in the first period to another baseline or a pair of snapshots. Do one of the following:
    1. To compare to another baseline, select Select a Baseline and the baseline you want to use in the second period, and then click Next.

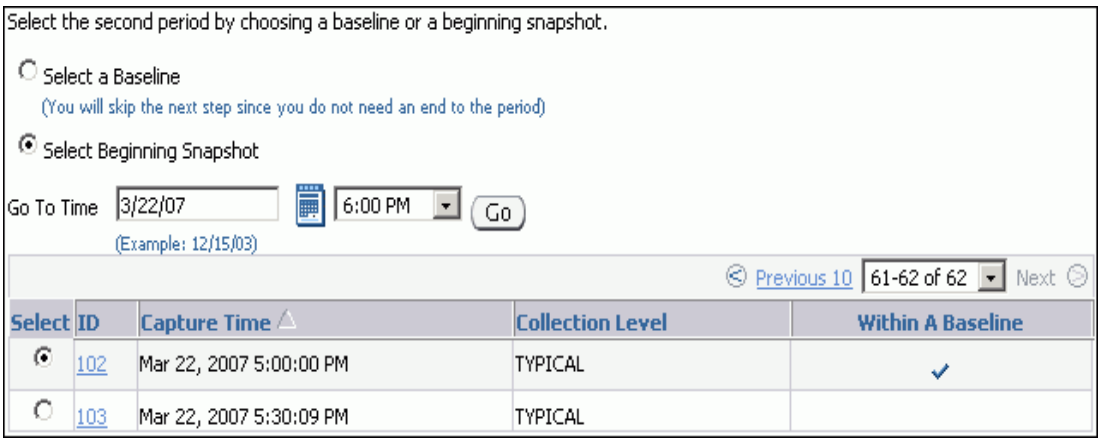
       In this example, the baseline named `SYSTEM_MOVING_WINDOW` is selected.



Description of the illustration awr_baseline2.gif

       The Compare Periods: Review page appears. Go to Step 7.
    2. To compare to a pair of snapshots, select Select Beginning Snapshot and the beginning snapshot to use in the second period, and then click Next.

       In this example, snapshot 102, taken on March 22, 2007 at 5:00 p.m., is selected.

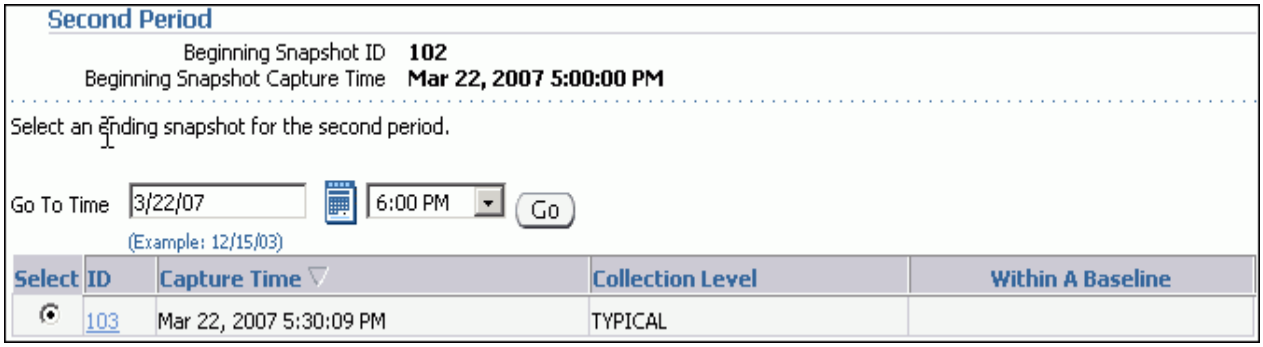


Description of the illustration awr_snapshot2_begin2.gif

The Compare Periods: Second Period End appears. Proceed to the next step.

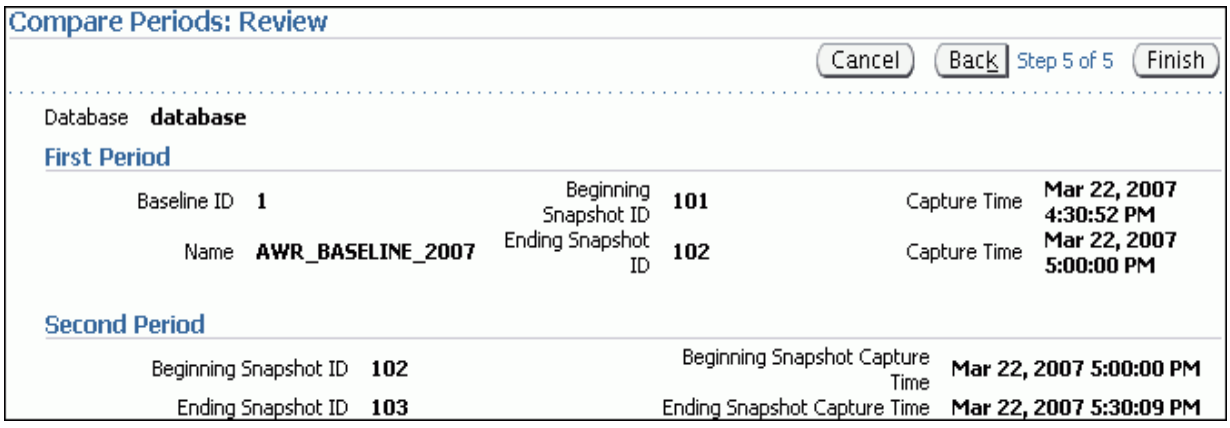
6. Select the ending snapshot for the snapshot period that will be included in the report and click Next.

   In this example, snapshot 103, taken on March 22, 2007 at 5:30 p.m., is selected.



Description of the illustration awr_snapshot2_end2.gif

The Compare Periods: Review page appears.



Description of the illustration awr2_review.gif

7. Review the periods to be included in the report and click Finish.

   The Compare Periods: Results page appears.

   Data from the selected periods appears under the General subpage. You can view data per second or per transaction by selecting an option from the View Data list.

Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			5,887.68	5,963.94	3.37	3.30
db block changes			19,944.00	26,344.00	11.42	14.56
execute count			20,285.00	20,770.00	11.61	11.48
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			18,051.00	18,514.00	10.33	10.23
parse count (total)			5,918.00	5,881.00	3.39	3.25
parse time cpu (seconds)			0.23	0.32	0.00	0.00
parse time elapsed (seconds)			0.26	1.13	0.00	0.00
physical reads			371.00	372.00	0.21	0.21
physical writes			1,202.00	923.00	0.69	0.51
redo size (KB)			3,680.53	4,473.99	2.11	2.47

Description of the illustration awr2_results_general.gif

In this example, parse time in the second period is much higher than the first.

8. Click Report to view the report.

   The Processing: View Report page appears while the report is being generated. After it completes, the report will appear. To change periods, click Change Periods. To save the report as an HTML file, click Save to File.

See Also:

- "Creating a Baseline"
- "Using the AWR Compare Periods Reports"

Comparing Two Pairs of Snapshots

If an existing baseline is not available, then you can compare the database performance by using two arbitrary pairs of snapshots, one pair taken when the database is performing optimally, and another pair when the database is performing poorly. At least four existing snapshots must be available.

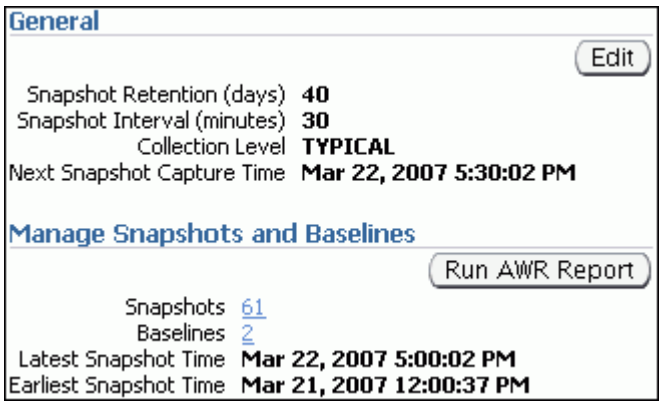
To compare performance using two pairs of snapshots:

1. From the Database Home page, click Server.

   The Server subpage appears.

2. Under Statistics Management, click Automatic Workload Repository.

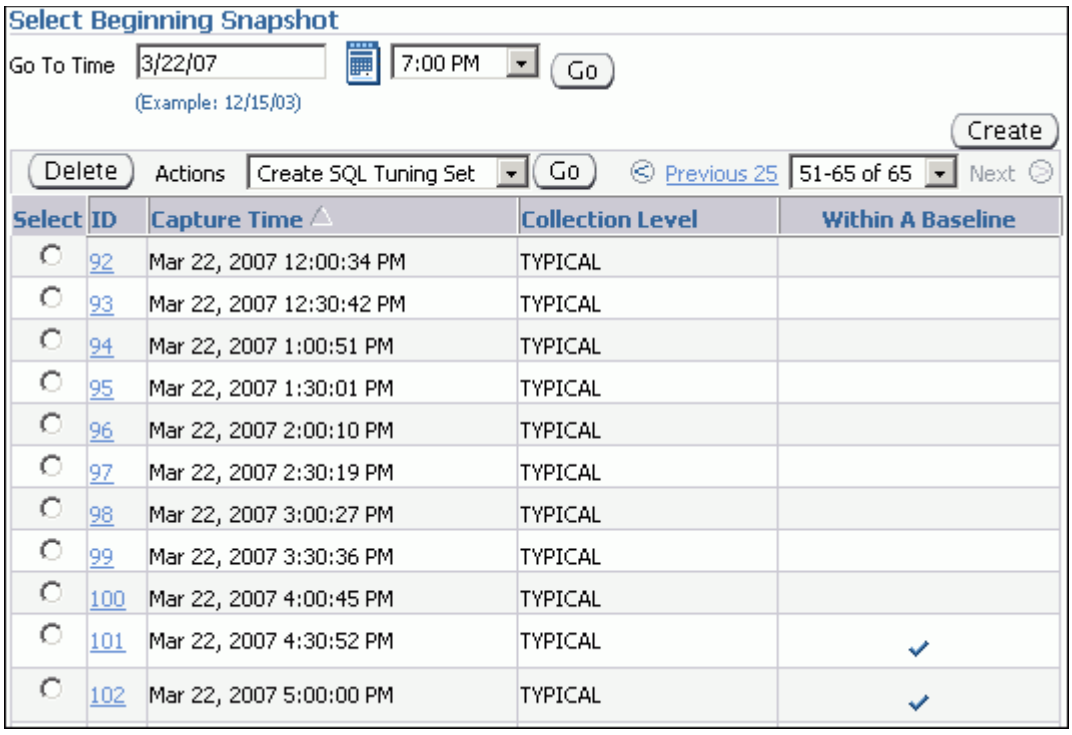
   The Automatic Workload Repository page appears.



Description of the illustration awr1.gif

3. Under Manage Snapshots and Baselines, click the link next to Snapshots.

   The Snapshots page appears.



Description of the illustration awr_snapshots.gif

4. From the Go To Time list, select the time for the starting snapshot and click Go.

   This action filters the snapshots and displays only the snapshot taken at the start of the comparison period. The time in this example is 5:00 p.m. on March 21, 2007.

Description of the illustration awr_gototime_1.gif

5. Under Select Beginning Snapshot, select the starting point for the first snapshot period to be included in the report.

   In this example, snapshot 53, taken on Mar 21, 2007 5:00 p.m., is selected.

Description of the illustration awr_snapshot1_begin.gif

6. From the Actions list, select Compare Periods and click Go.

   The Compare Periods: First Period End page appears.

7. Select the ending point for the first snapshot period to be included in the report and click Next.

   In this example, snapshot 55, taken on Mar 21, 2007 6:00 p.m., is selected.

◉	55	Mar 21, 2007 6:00:26 PM	TYPICAL
○	56	Mar 21, 2007 6:30:34 PM	TYPICAL
○	57	Mar 21, 2007 7:00:42 PM	TYPICAL

Description of the illustration awr_snapshot1_end.gif

   The Compare Periods: Second Period Start page appears.

8. Select the starting point for the second snapshot period to be included in the report and click Next.

   In this example, snapshot 104, taken on March 22, 2007 at 6:00 p.m., is selected.

◉	104	Mar 22, 2007 6:00:18 PM	TYPICAL
○	105	Mar 22, 2007 6:30:25 PM	TYPICAL
○	106	Mar 22, 2007 7:00:33 PM	TYPICAL

Description of the illustration awr_snapshot2_begin.gif

   The Compare Periods: Second Period End page appears.

9. Select the end point for the second period that will be included in the report and click Next.

   In this example, snapshot 106, taken on March 22, 2007 at 7:00 p.m., is selected.

◉	106	Mar 22, 2007 7:00:33 PM	TYPICAL
○	107	Mar 22, 2007 7:30:41 PM	TYPICAL
○	108	Mar 22, 2007 8:00:50 PM	TYPICAL

Description of the illustration awr_snapshot2_end.gif

   The Compare Periods: Review page appears.

First Period

Beginning Snapshot ID	**53**	Beginning Snapshot Capture Time	**Mar 21, 2007 5:00:09 PM**
Ending Snapshot ID	**55**	Ending Snapshot Capture Time	**Mar 21, 2007 6:00:26 PM**

Second Period

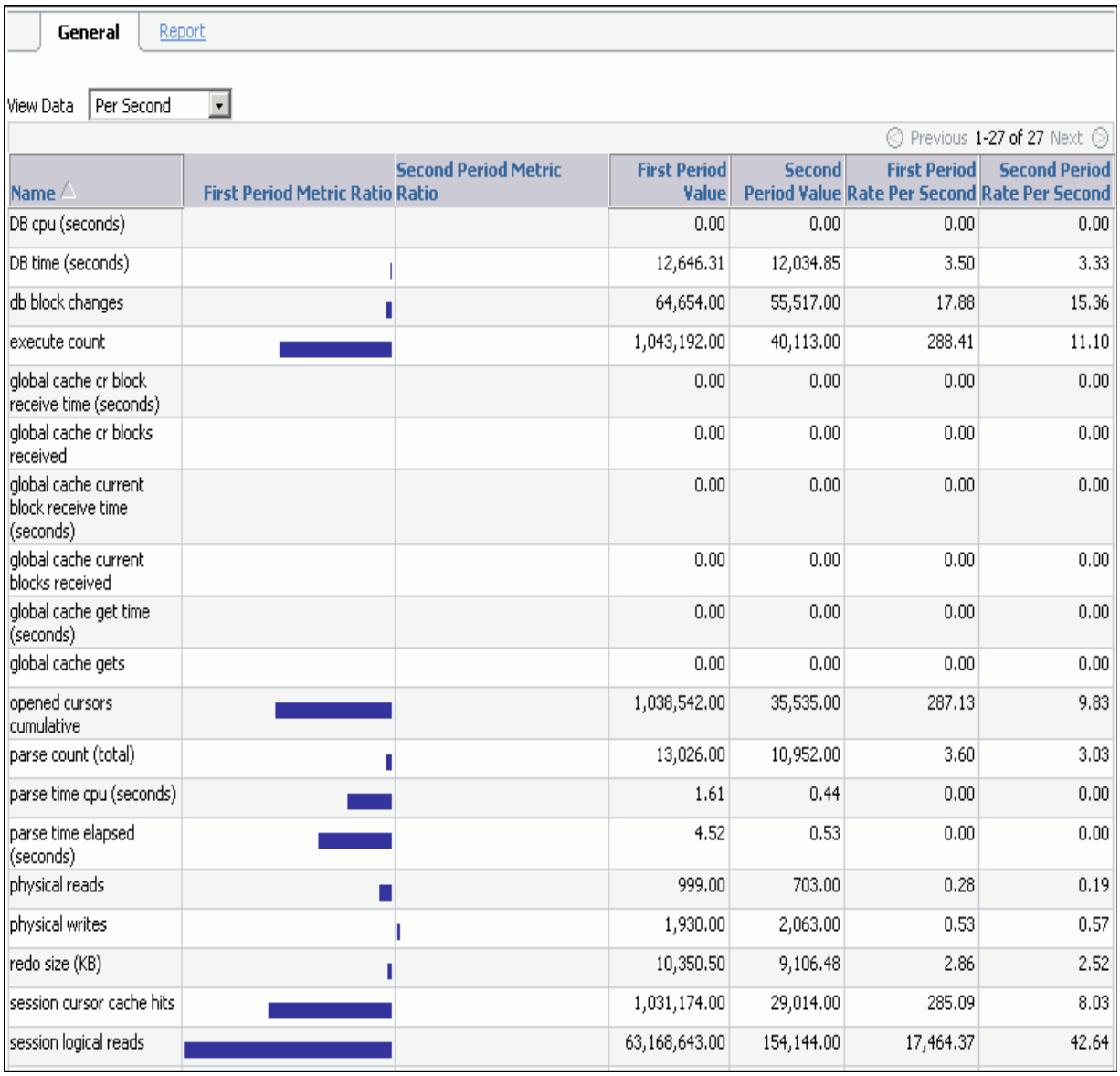
Beginning Snapshot ID	**104**	Beginning Snapshot Capture Time	**Mar 22, 2007 6:00:18 PM**
Ending Snapshot ID	**106**	Ending Snapshot Capture Time	**Mar 22, 2007 7:00:33 PM**

Description of the illustration awr_review.gif

10. Review the selected periods that will be included in the report and click Finish.

    The Compare Periods: Results page appears.

    Data from the selected periods appears under the General subpage. You can view data per second or per transaction by selecting an option from the View Data list.



Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			12,646.31	12,034.85	3.50	3.33
db block changes			64,654.00	55,517.00	17.88	15.36
execute count			1,043,192.00	40,113.00	288.41	11.10
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			1,038,542.00	35,535.00	287.13	9.83
parse count (total)			13,026.00	10,952.00	3.60	3.03
parse time cpu (seconds)			1.61	0.44	0.00	0.00
parse time elapsed (seconds)			4.52	0.53	0.00	0.00
physical reads			999.00	703.00	0.28	0.19
physical writes			1,930.00	2,063.00	0.53	0.57
redo size (KB)			10,350.50	9,106.48	2.86	2.52
session cursor cache hits			1,031,174.00	29,014.00	285.09	8.03
session logical reads			63,168,643.00	154,144.00	17,464.37	42.64

Description of the illustration awr_results_general.gif

In this example, the first period shows significantly more activity, especially in session reads, than the second period.

11. To view the report, click the Report tab.

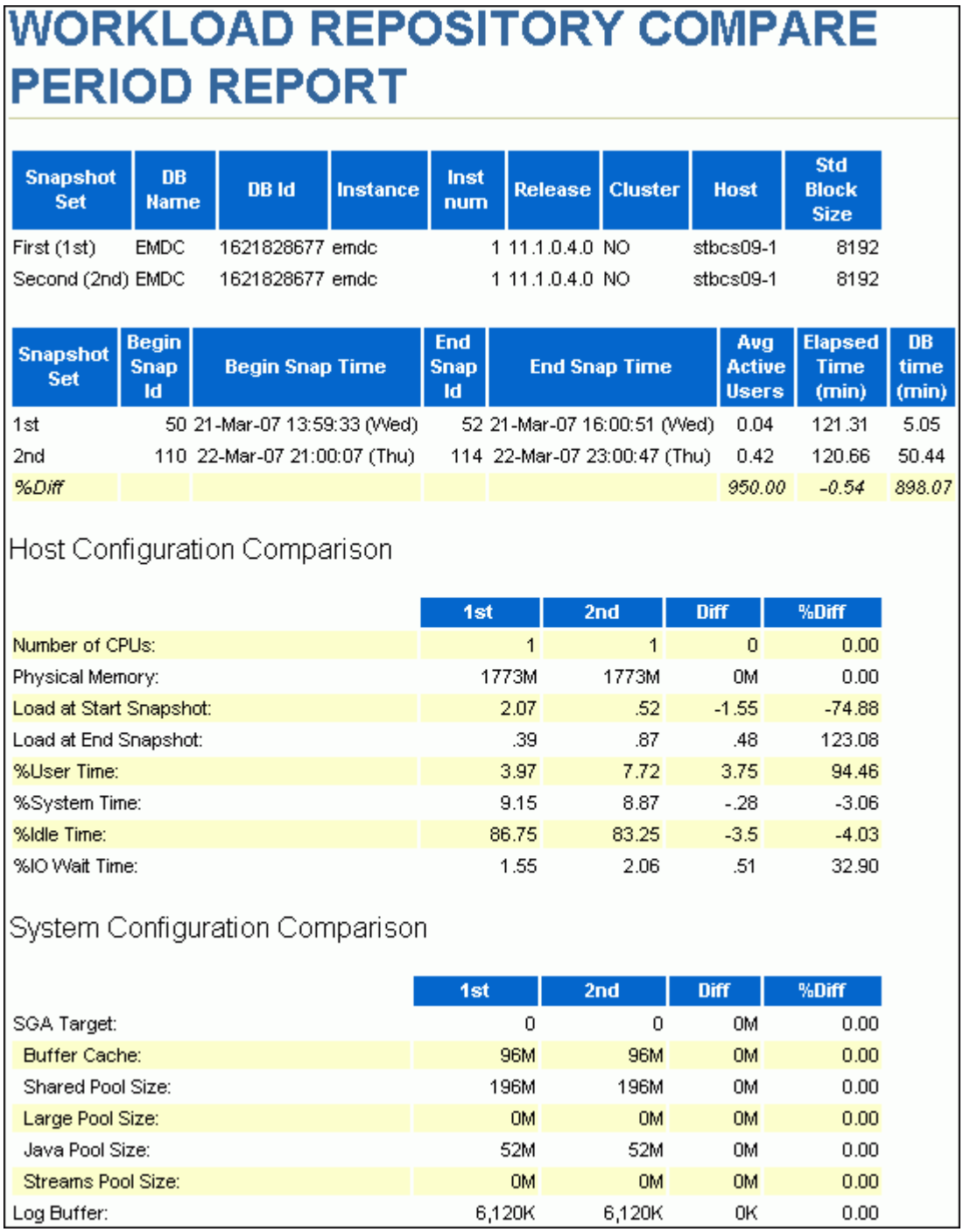
    The Processing: View Report page appears while the report is being generated. After it completes, the report will appear. To change periods, click Change Periods. To save the report as an HTML file, click Save to File.

Using the AWR Compare Periods Reports

After an AWR Compare Periods report is generated for the time periods you want to compare, you can use it to perform an analysis of performance degradation with Oracle Database that may have happened over time. To learn how to generate AWR Compare Periods reports, see "Running the AWR Compare Periods Reports".

Figure 8-1 shows an example of an AWR Compare Periods report.

Figure 8-1 AWR Compare Periods Report



WORKLOAD REPOSITORY COMPARE PERIOD REPORT

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host	Std Block Size
First (1st)	EMDC	1621828677	emdc	1	11.1.0.4.0	NO	stbcs09-1	8192
Second (2nd)	EMDC	1621828677	emdc	1	11.1.0.4.0	NO	stbcs09-1	8192

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	50	21-Mar-07 13:59:33 (Wed)	52	21-Mar-07 16:00:51 (Wed)	0.04	121.31	5.05
2nd	110	22-Mar-07 21:00:07 (Thu)	114	22-Mar-07 23:00:47 (Thu)	0.42	120.66	50.44
%Diff					*950.00*	*-0.54*	*898.07*

Host Configuration Comparison

	1st	2nd	Diff	%Diff
Number of CPUs:	1	1	0	0.00
Physical Memory:	1773M	1773M	0M	0.00
Load at Start Snapshot:	2.07	.52	-1.55	-74.88
Load at End Snapshot:	.39	.87	.48	123.08
%User Time:	3.97	7.72	3.75	94.46
%System Time:	9.15	8.87	-.28	-3.06
%Idle Time:	86.75	83.25	-3.5	-4.03
%IO Wait Time:	1.55	2.06	.51	32.90

System Configuration Comparison

	1st	2nd	Diff	%Diff
SGA Target:	0	0	0M	0.00
Buffer Cache:	96M	96M	0M	0.00
Shared Pool Size:	196M	196M	0M	0.00
Large Pool Size:	0M	0M	0M	0.00
Java Pool Size:	52M	52M	0M	0.00
Streams Pool Size:	0M	0M	0M	0.00
Log Buffer:	6,120K	6,120K	0K	0.00

Description of "Figure 8-1 AWR Compare Periods Report"

The AWR Compare Periods report is divided into the following sections:

- Summary of the AWR Compare Periods Report
- Details of the AWR Compare Periods Report
- Supplemental Information in the AWR Compare Periods Report

Summary of the AWR Compare Periods Report

The report summary is at the beginning of the AWR Compare Periods report, and summarizes information about the snapshot sets and loads used in the report. The report summary contains the following sections:

- Snapshot Sets

- Host Configuration Comparison
- System Configuration Comparison
- Load Profile
- Top Timed Events

Snapshot Sets

The Snapshot Sets section displays information about the snapshot sets used for this report, such as instance, host, and snapshot information.

In the example shown in Figure 8-1, the first snapshot period corresponds to the time when performance was stable on March 21, 2007 from 1:59 p.m. to 4:00 p.m. The second snapshot period corresponds to the time when performance degradation occurred on March 22, 2007 from 9:00 p.m. to 11:00 p.m.

Host Configuration Comparison

The Host Configuration Comparison section compares the host configurations used in the two snapshot sets. For example, the report compares physical memory and number of CPUs. Any differences in the configurations are quantified as percentages in the %Diff column.

System Configuration Comparison

The System Configuration Comparison section compares the database configurations used in the two snapshot sets. For example, the report compares the SGA and log buffer size. Any differences in the configurations are quantified as percentages in the %Diff column.

Load Profile

The Load Profile section compares the loads used in the two snapshot sets. Any differences in the loads are quantified as percentages in the %Diff column.

Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	0.01	0.25	*2,400.00*	0.04	0.70	*1,650.00*
CPU time:	0.01	0.04	*300.00*	0.02	0.12	*500.00*
Redo size:	2,676.70	8,280.03	*209.34*	7,945.55	23,502.95	*195.80*
Logical reads:	43.25	754.87	*1,645.36*	128.39	2,142.71	*1,568.91*
Block changes:	16.59	202.59	*1,121.16*	49.24	575.05	*1,067.85*
Physical reads:	0.10	8.19	*8,090.00*	0.29	23.23	*7,910.34*
Physical writes:	0.61	1.76	*188.52*	1.81	5.01	*176.80*
User calls:	3.37	3.17	*-5.93*	9.99	9.00	*-9.91*
Parses:	2.97	6.99	*135.35*	8.83	19.84	*124.69*
Hard parses:	0.01	0.38	*3,700.00*	0.02	1.07	*5,250.00*
Sorts:	2.18	3.88	*77.98*	6.48	11.03	*70.22*
Logons:	0.03	0.03	*0.00*	0.09	0.09	*0.00*
Executes:	11.04	23.38	*111.78*	32.78	66.38	*102.50*
Transactions:	0.34	0.35	*2.94*			

	1st	2nd	Diff
% Blocks changed per Read:	38.35	26.84	*-11.51*
Recursive Call %:	88.68	97.50	*8.82*
Rollback per transaction %:	20.14	0.63	*-19.51*
Rows per Sort:	3.18	101.17	*97.98*
Avg DB time per Call (sec):	0.00	0.08	*0.07*

Description of the illustration awr_load_profile.gif

Top Timed Events

The Top 5 Timed Events section displays the five timed events or operations that consumed the highest percentage of total DB time in each of the snapshot sets.

1st						2nd					
Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time	Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time
CPU time			111.72		36.84	CPU time			912.19		30.14
db file sequential read	User I/O	7,132	99.95	14.01	32.96	resmgr:cpu quantum	Scheduler	1,622	333.99	205.91	11.04
control file parallel write	System I/O	2,635	36.66	13.91	12.09	library cache lock	Concurrency	3	265.85	88,617.53	8.78
log file parallel write	System I/O	2,652	30.51	11.50	10.06	db file sequential read	User I/O	17,442	247.98	14.22	8.19
log file sync	Commit	622	19.33	31.07	6.37	db file scattered read	User I/O	3,229	84.87	26.28	2.80
-db file scattered read	User I/O	932	17.09	18.33	5.63	-log file parallel write	System I/O	3,373	63.08	18.70	2.08
-						-control file parallel write	System I/O	2,774	27.83	10.03	0.92
-						-log file sync	Commit	607	25.54	42.08	0.84

Description of the illustration awr_top5_timed_events.gif

In this example, CPU time is over eight times higher in the second period than in the first. The number of waits for the `db file sequential read` event in the second period is over double the number in the first.

Details of the AWR Compare Periods Report

The details section follows the summary of the AWR Compare Periods report, and provides statistics about the snapshot sets and loads used in the report. For example, the section includes statistics for database time, wait events, SQL execution time, and instance activity.

Supplemental Information in the AWR Compare Periods Report

The supplemental information is at the end of the AWR Compare Periods report, and provides additional information about initialization parameters and SQL statements. The init.ora Parameters section lists all the initialization parameter values for the first snapshot set. The Complete List of SQL Text section lists each statement by SQL ID and shows the text of the SQL statement.

2. for awr

```
execute dbms_workload_repository.create_snapshot;

select snap_interval, retention
from dba_hist_wr_control;

-- retention: 2 days
-- interval: 20 minutes
begin
dbms_workload_repository.modify_snapshot_settings (
interval => 20,
retention => 2*24*60
```

```
);
end;

rem -- change ADDM sensitiveness back to normal
exec dbms_advisor.set_default_task_parameter ( 'ADDM', 'DB_ACTIVITY_MIN', 300);

more sensitive:
exec dbms_advisor.set_default_task_parameter ( 'ADDM', 'DB_ACTIVITY_MIN', 30);

SELECT ADVISOR_NAME, PARAMETER_VALUE
FROM DBA_ADVISOR_DEF_PARAMETERS

-- ----------------------------------------------
```

Oracle assumes the value of the parameter (not intialization parameter) DBIO_EXPECTED is 10 milliseconds.

```
SELECT PARAMETER_VALUE
FROM DBA_ADVISOR_DEF_PARAMETERS
WHERE ADVISOR_NAME='ADDM'
AND PARAMETER_NAME='DBIO_EXPECTED'
```

If your hardware is significantly different, you can set the parameter value one time for all subsequent ADDM executions:

```
DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER('ADDM','DBIO_EXPECTED', 8000);

dbms_advisor.set_default_task_parameter('ADDM','DB_ACTIVITY_MIN',300);



BEGIN
-- Create an ADDM task.
DBMS_ADVISOR.create_task (
advisor_name => 'ADDM',
task_name => '970_1032_AWR_SNAPSHOT',
task_desc => 'Advisor for snapshots 970 to 1032.');

-- Set the start and end snapshots.
DBMS_ADVISOR.set_task_parameter (
task_name => '970_1032_AWR_SNAPSHOT',
parameter => 'START_SNAPSHOT',
value => 970);
```

```
DBMS_ADVISOR.set_task_parameter (
task_name => '970_1032_AWR_SNAPSHOT',
parameter => 'END_SNAPSHOT',
value => 1032);

-- Execute the task.
DBMS_ADVISOR.execute_task(task_name => '970_1032_AWR_SNAPSHOT');
END;
/

-- Display the report.
SET LONG 100000
SET PAGESIZE 50000
SELECT DBMS_ADVISOR.get_task_report('970_1032_AWR_SNAPSHOT') AS report
FROM dual;
SET PAGESIZE 24
```

13.Use SQL Plan Management feature

PL/SQL Packages and Types Reference=>132 DBMS_SPM

围绕 spa 策略

用来管理执行计划，设置 optimizer_capture_sql_plan_baselines=true 之后可以自动对进行计划进行捕捉。可以把执行计划保存下来或者传输给别的数据库使用，或者把别的数据库的执行计划拿来用，DBMS_SPM 包具体功能列表如下。

Summary of DBMS_SPM Subprograms

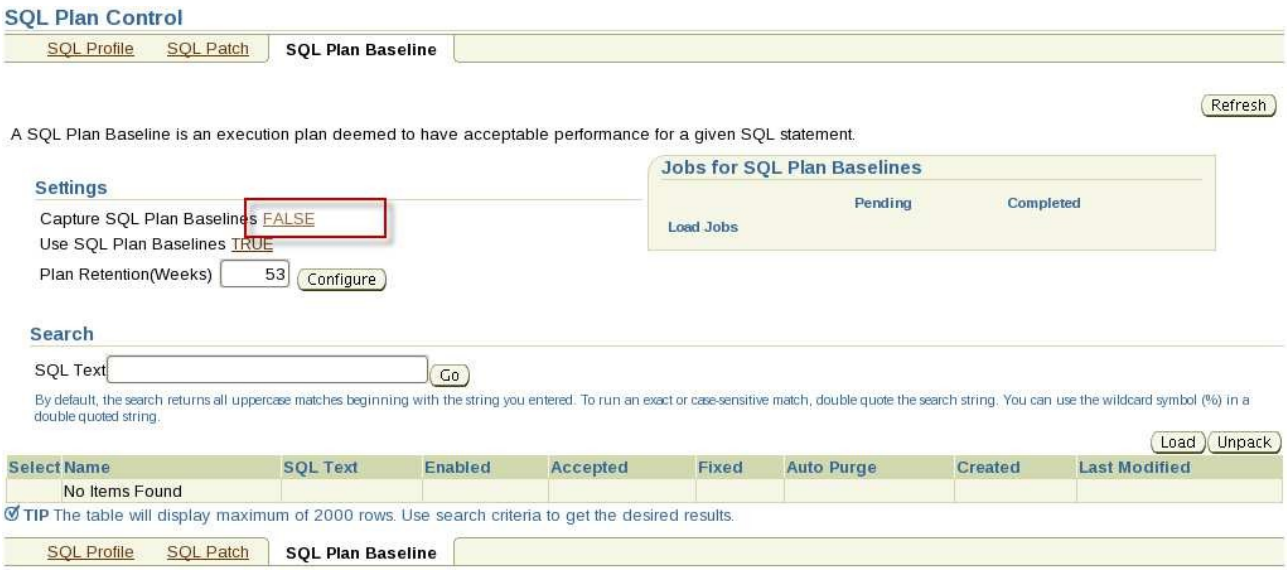
This table list the package subprograms in alphabetical order.

Table 132-2 DBMS_SPM Package Subprograms

Subprogram	Description
ALTER_SQL_PLAN_BASELINE Function	Changes an attribute of a single plan or all plans associated with a SQL statement using the attribute name/value format
CONFIGURE Procedure	Sets configuration options for SQL management base, in parameter/value format
CREATE_STGTAB_BASELINE Procedure	Creates a staging table that used for transporting SQL plan baselines from one system to another
DROP_SQL_PLAN_BASELINE Function	drops a single plan, or all plans associated with a SQL statement
EVOLVE_SQL_PLAN_BASELINE Function	Evolves SQL plan baselines associated with one or more SQL statements
LOAD_PLANS_FROM_CURSOR_CACHE Functions	Loads one or more plans present in the cursor cache for a SQL statement
LOAD_PLANS_FROM_SQLSET Function	Loads plans stored in a SQL tuning set (STS) into SQL plan baselines
MIGRATE_STORED_OUTLINE Functions	Migrates existing stored outlines to SQL plan baselines
PACK_STGTAB_BASELINE Function	Packs (exports) SQL plan baselines from SQL management base into a staging table
UNPACK_STGTAB_BASELINE Function	Unpacks (imports) SQL plan baselines from a staging table into SQL management base

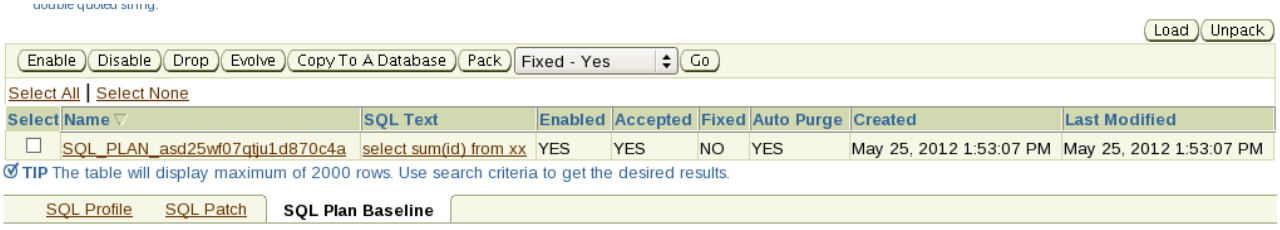
界面如下

要先启用 baseline 功能，**未启用的时候 Capture 那里为 False**

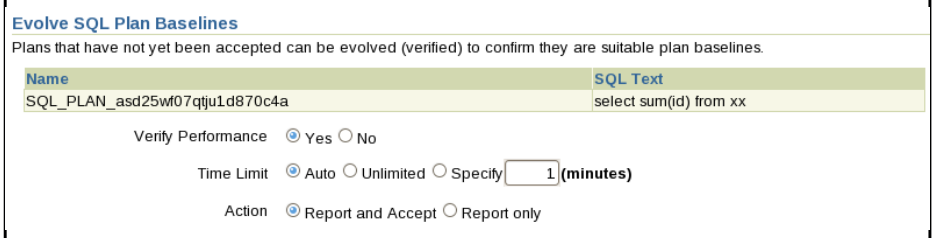


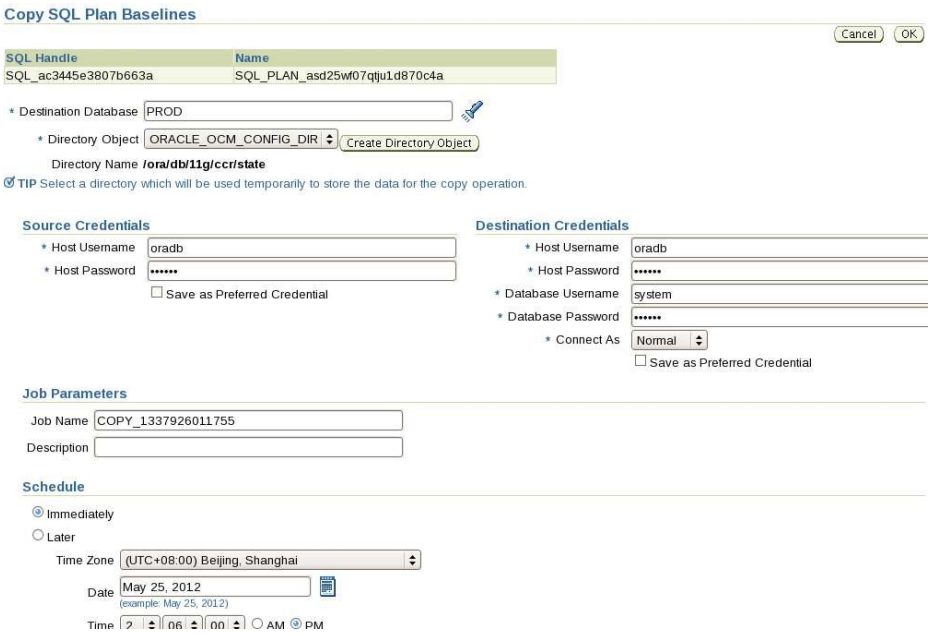
alter system set optimizer_capture_sql_plan_baselines=true scope=both;

对于一个已有的执行计划可以做以下操作



Enable	启用该执行计划 DBMS_SPM.ALTER_SQL_PLAN_BASELINE

Disable	禁用该执行计划 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
Drop	删除该执行计划　DBMS_SPM.DROP_SQL_PLAN_BASELINE
Evolve	进化！ 是否允许执行计划进化为效率更高的执行计划 DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE 
Copy To A database	把执行计划复制到其他的数据库
PACK	打包保存起来 DBMS_SPM.PACK_STGTAB_BASELINE
Fixed-Yes	固定 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
Fixed-No	不固定 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
AutoPurge - Yes	自动删除 是 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
AutoPurge - No	自动删除 是 DBMS_SPM.ALTER_SQL_PLAN_BASELINE



14.Implement instance caging

Oracle® Tutor™

升级考试 Section4 性能调优（Performance Management）

1.Set up and configure Resource Manager to control active sessions, number of I/Os, execution time..etc

2.Use Result Cache

3.Use multi column statistics

4.Gather statistics on a specific table without invalidating cursors

5.Use partitioned indexes

6.Use SQL Tuning Advisor

7.Use SQL Access Advisor

8.Configure baseline templates

9.Use SQL Plan Management feature

10.Replay a captured workload

replay 包括会话访问和并发，压力

1. Capture the workload on a database. (Task 1)
2. Optionally export the AWR data. (Task 1)
3. Restore the replay database on a test system to match the capture database at the start of the workload capture.
4. Make changes (such as performing an upgrade) to the test system as required.
5. Copy the generated workload files to the test system.
6. Preprocess the captured workload on the test system. (Task 2)
7. Configure the test system for the replay.
8. Replay the workload on the restored database. (Task 3)

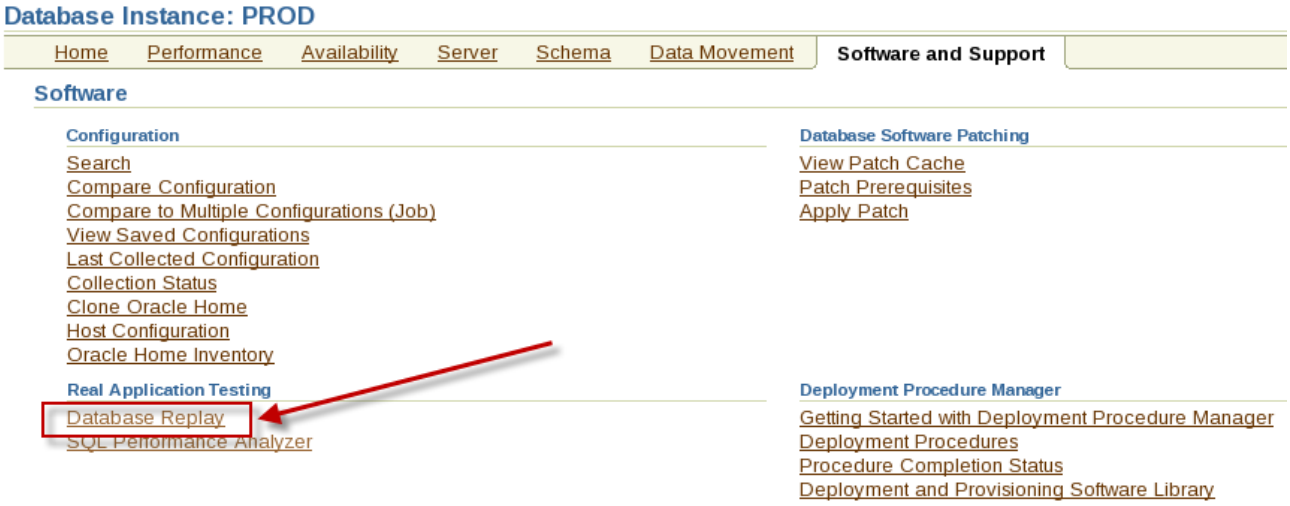
Capture the workload on a database

- 创建基础构架

create user tworkload identified by tt account unlock default tablespace users;

grant dba to tworkload;

conn tworkload/tt



Optionally export the AWR data

Replay captured workload

Database Replay

Database Replay allows workloads to be captured from production systems and re-executed with high fidelity on test copies of production databases. This enables detailed analysis of how the proposed changes may affect production systems; for instance, patching or upgrading database software.

Page Refreshed **May 24, 2012 9:34:49 AM CST** Refresh

Task List

Expand All | Collapse All

Task Name	Description	Go to Task
Capture Production Workload	Initiate or schedule a workload capture, export AWR data after capture, and copy captured files to the workload staging area.	
Capture Workload	Capture a workload from the production environment. This can be scheduled to accommodate a database restart if desired.	
Export AWR Data	Export AWR data to provide a better performance comparison between captured and replayed workloads.	
Copy to Workload Staging Area	Copy captured files away from production to the workload staging area for later preprocessing. For a cluster database, captured files from different database instances can be consolidated in the workload staging area.	
Prepare Test Database	Set up a test database from production, upgrade or otherwise modify the test database, and isolate the test database prior to replay.	
Prepare for Replay	Prepare the workload capture files for replay (preprocess), copy the preprocessed workload files to the workload staging area, deploy the Replay Clients, and copy the preprocessed workload files to the Replay Client hosts.	
Replay Workload on Test Database	Set up the workload replay on the test database, copy the replay results to the workload staging area, and analyze the results.	

Plan Environment — Options — Parameters — Schedule — Review

Capture Workload: Plan Environment

Database **PROD**
Logged In As **system**

Cancel Step 1 of 5 Next

The following prerequisites should be met to avoid potential problems before proceeding to capture the workload.

Prerequisite	Acknowledge
Make sure there is enough disk space to hold the captured workload. Consider doing a short duration workload capture and using it for estimating the disk space requirement of a full workload capture.	☑
Make sure you can restore the replay database to match the capture database at the start of the workload capture. A successful workload replay depends on application transactions accessing application data identical to that on a capture system. Common ways to restore application data state include point-in-time recovery, flashback, and import/export.	☑

Cancel Step 1 of 5 Next

Plan Environment — Options — Parameters — Schedule — Review

Capture Workload: Options

Database **PROD**
Logged In As **system**

Cancel Back Step 2 of 5 Next

捕捉过程是否有数据库重启情况

Database Restart Options

A database restart prior to a workload capture is recommended to ensure a complete and accurate capture. Not restarting could capture in-flight transactions, which may adversely affect the replay of subsequent captured transactions.

- ◉ Do not restart the database prior to the capture.
- ○ Restart the database prior to the capture.

记录筛选模式
exclusion 排除下面列表指定的信息
inclusion 只记录下面列表指定的信息

Workload Filters

Workload filters can customize the workload to be captured. By default, most external client requests made to the database are captured. Refer to the Oracle Real Ap Testing User's Guide for more information.

Filter Mode Exclusion

指定筛选选项
可以是 User Program Module Action Service SessionID

Excluded Sessions

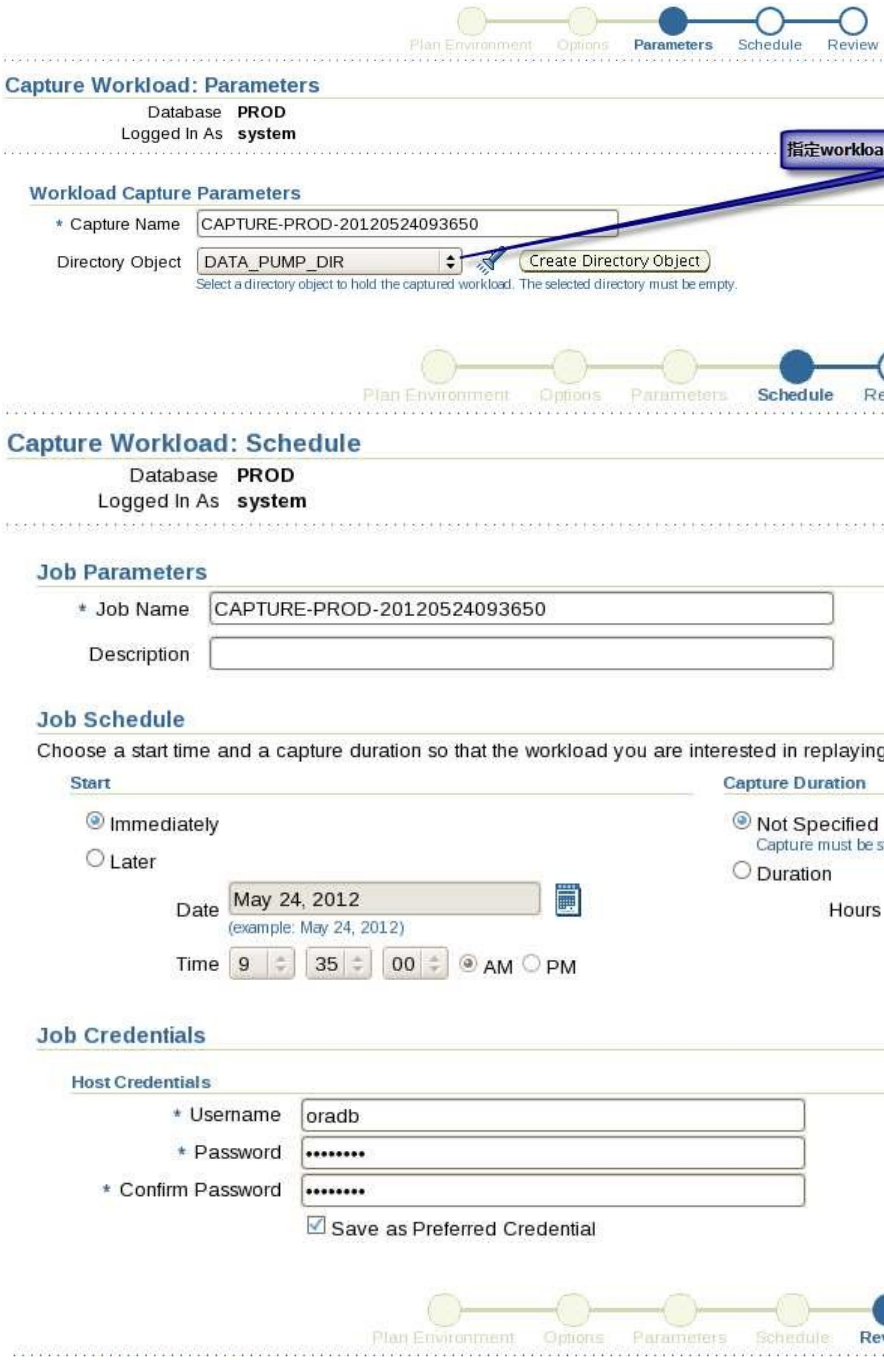
All sessions will be captured except for those listed below.

Filter Name	Type	Session Attribute	Va
Oracle Management Service (DEFAULT)	Excluded	Program	OMS
Oracle Management Agent (DEFAULT)	Excluded	Program	emagent%

Add Another Row

TIP You may use % for wildcard in a filter value.

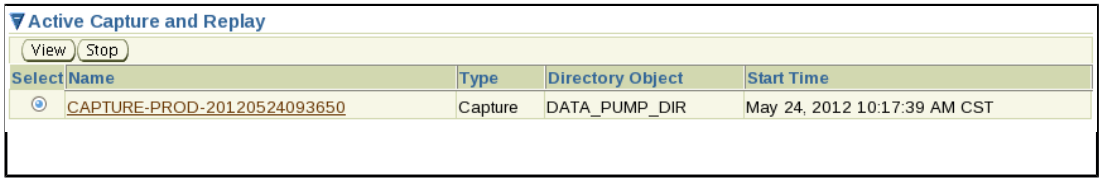
Cancel Back Step 2 of 5 Next


Plan Environment
Options
Parameters
Schedule
Review
Capture Workload: Parameters
Database PROD
Logged In As system
Cancel
Back
Step 3 of 5
Next
指定workload 保存名字和路径
Workload Capture Parameters
* Capture Name
CAPTURE-PROD-20120524093650
Directory Object
DATA_PUMP_DIR
Create Directory Object
Select a directory object to hold the captured workload. The selected directory must be empty.
Plan Environment
Options
Parameters
Schedule
Review
Capture Workload: Schedule
Database PROD
Logged In As system
Cancel
Back
Step 4 of 5
Next
Job Parameters
* Job Name
CAPTURE-PROD-20120524093650
Description
Job Schedule
Choose a start time and a capture duration so that the workload you are interested in replaying at a later time can be captured.
Start
Immediately
Later
Date
May 24, 2012
(example: May 24, 2012)
Time
9
35
00
AM
PM
Capture Duration
Not Specified
Capture must be stopped manually if an end is not specified
Duration
Hours
0
Minutes
0
Job Credentials
Host Credentials
* Username
oradb
* Password
* Confirm Password
Save as Preferred Credential
Plan Environment
Options
Parameters
Schedule
Review
Capture Workload: Review
Database PROD
Logged In As system
Cancel
Back
Step 5 of 5
Submit
Review the following settings for capturing the workload.
Job Name CAPTURE-PROD-20120524093650
Capture Name CAPTURE-PROD-20120524093650
Directory Object DATA_PUMP_DIR
Directory Path /ora/db/admin/PROD/dpdump/
Start Time Immediately
Capture Duration Not Specified
Restart Database No
Workload Filters: Excluded Sessions
Filter Name
Type
Session Attribute
Value
Oracle Management Service (DEFAULT)
Excluded
Program
OMS
Oracle Management Agent (DEFAULT)
Excluded
Program
emagent%
Cancel
Back
Step 5 of 5
Submit

注意！如果提交后报错，直接去 dump 目录删除所有的文件。

```
ORA-15505: cannot start workload capture because instance 1 encountered
errors
while accessing directory "/ora/db/admin/PROD/dpdump/"
ORA-06512: at "SYS.DBMS_WORKLOAD_CAPTURE", line 883
ORA-06512: at line 1
```

提交成功后在 Active Capture and Replay 会看到，此时在执行一些负载脚本，都会被记录下来



手工提交过程

```
--create capture filter
BEGIN
DBMS_WORKLOAD_CAPTURE.ADD_FILTER (fname => 'filter_test',fattribute
=> 'USER',fvalue => 'SYSTEM');
END;
/
--start capture
BEGIN
dbms_workload_capture.start_capture(UNISTR('CAPTURE-PROD-
20120524093650'), UNISTR('DATA_PUMP_DIR'), NULL, 'INCLUDE', FALSE);
END;
/
```

跑个压力脚本瞧瞧

```
truncate table xx;
begin
  for ctr in 1..1000000 loop
     insert into xx values
(ctr,lpad(round(dbms_random.value(1,9999999999999999)),20,0),sysd
ate);
commit;
  end loop;
commit;
end;
/
update xx set text=text*10;
commit;
```

跑完压力脚本后，去停掉 capture,点 STOP 就行

Active Capture and Replay

View | Stop

Select	Name	Type	Directory Object	Start Time
◉	CAPTURE-PROD-20120524093650	Capture	DATA_PUMP_DIR	May 24, 2012 10:17:39 AM CST

-- stop capture

exec dbms_workload_capture.FINISH_CAPTURE();

看看记录的小结,已经成功的记录了刚刚压力脚本的信息

View Workload Capture: CAPTURE-PROD-A

Export AWR Data | OK

The OK button will return to the Database Replay home page.

Status **Completed**

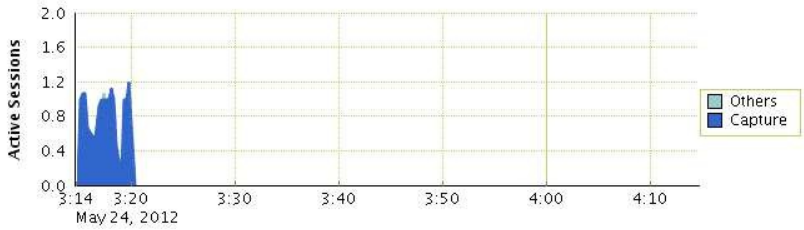
Summary

Name	CAPTURE-PROD-A	Captured Data Size (MB)	58.12
Directory Object	DATA_PUMP_DIR	Duration (hh:mm:ss)	00:05:56
Database Name	PROD	Start Time	May 24, 2012 3:14:38 PM CST
Capture Database Version	11.2.0.3.0	End Time	May 24, 2012 3:20:34 PM CST
Cluster Database	No	Start SCN	267909
DBID	195950047	End SCN	1488124
Capture Error Code	None	AWR Data Exported	No
Capture Error Message	None		

Workload Profile | Workload Filters

View Workload Capture Report

Average Active Sessions



Comparison

	Total	Capture	Percentage of Total
Database Time (hh:mm:ss)	00:05:17	00:05:12	98.42
Average Active Sessions	0.89	0.88	98.42
User Calls	1,626	572	35.18
Transactions	1,000,008	1,000,002	100.00
Session Logins	12	6	50.00
Application Errors	N/A	0	N/A

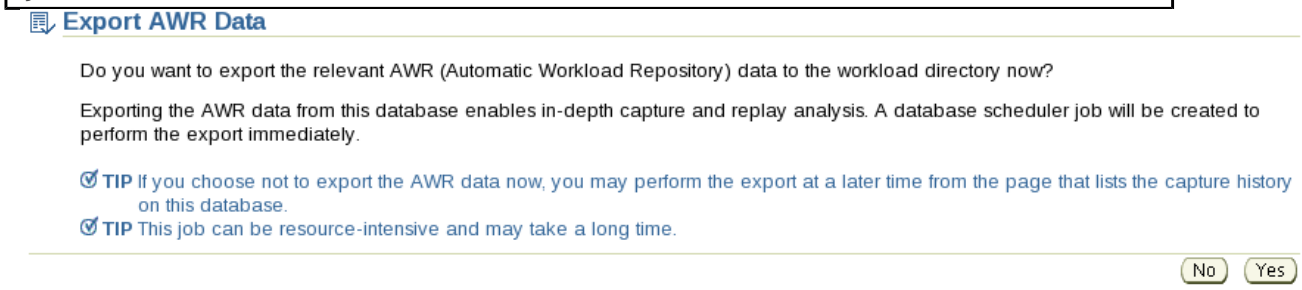
Workload Profile | Workload Filters

#保存的文件

```
[root@edu1 dpdump]# pwd
/ora/db/admin/PROD/dpdump
[root@edu1 dpdump]# du -m
1       ./capfiles/inst1/af
1       ./capfiles/inst1/ad
1       ./capfiles/inst1/ab
1       ./capfiles/inst1/ah
1       ./capfiles/inst1/ai
59      ./capfiles/inst1/aa
1       ./capfiles/inst1/ae
1       ./capfiles/inst1/ag
1       ./capfiles/inst1/aj
1       ./capfiles/inst1/ac
59      ./capfiles/inst1
59      ./capfiles
10      ./cap
68      .
```

Optionally export the AWR data

导出 AWR 数据作为参考值,其实就是做个快照，并且把这个快照保存起来，方便比对.



Replay captured workload

#文件复制，可以用 GC 的 Copy to Workload Staging Area 或者直接拿命令行复制，注意！整个 dmp 目录都要复制过去。

#我们以 /ora/db/admin/PROD/dpdump => /ora/db/admin/PROD/repdir 为例

>> 准备工作 replay 文件准备

Database Instance: PROD > Logged in As SYSTEM

Database Replay

Database Replay allows workloads to be captured from production systems and re-executed with high fidelity on test copies of production databases. This enables detailed analysis of how the proposed changes may affect production systems; for instance, patching or upgrading database software.

Page Refreshed **May 24, 2012 3:42:53 PM CST** Refresh

Task List

Expand All | Collapse All

Task Name	Description	Go to Task
Capture Production Workload	Initiate or schedule a workload capture, export AWR data after capture, and copy captured files to the workload staging area.	
Prepare Test Database	Set up a test database from production, upgrade or otherwise modify the test database, and isolate the test database prior to replay.	
Prepare for Replay	Prepare the workload capture files for replay (preprocess), copy the preprocessed workload files to the workload staging area, deploy the Replay Clients, and copy the preprocessed workload files to the Replay Client hosts.	
Preprocess Workload	Preprocessing prepares a captured workload for replay. You must do this once for every captured workload. Preprocessing is best performed in the test database. The captured workload must be accessible from the test database.	
Copy to Workload Staging Area	Copy preprocessed workload files to the workload staging area. The preprocessed workload files must be accessible by the database server and the Replay Clients during replay.	
Deploy Replay Clients	Deploy the Replay Client to one or more host machines. Replay Clients are used to replay the preprocessed workload.	
Copy Workload to Replay Client Hosts	Copy the preprocessed workload to one or more Replay Client host machines. Each Replay Client must be able to access the preprocessed workload during replay.	
Replay Workload on Test Database	Set up the workload replay on the test database, copy the replay results to the workload staging area, and analyze the results.	

Active Capture and Replay

Select	Name	Type	Directory Object	Start Time
	No items found			

Workload Capture History

选择目录准备方式，我们用一个手工复制过来的目录

Locate Workload — Copy Workload — Select Directory — Schedule — Review

Preprocess Captured Workload: Locate Workload

Database **PROD**
Version **11.2.0.3.0**
Logged In As **system**

Cancel Step 1 of 5 Next

The captured workload directory must be accessible from this database.

- ○ Copy the workload directory to this host from another host.
- ◉ Use an existing workload directory on this host.

指定目录位置后会自动把选定目录中的捕捉信息显示出来

定义个任务名，根据目录中的信息生成 replay 基础信息

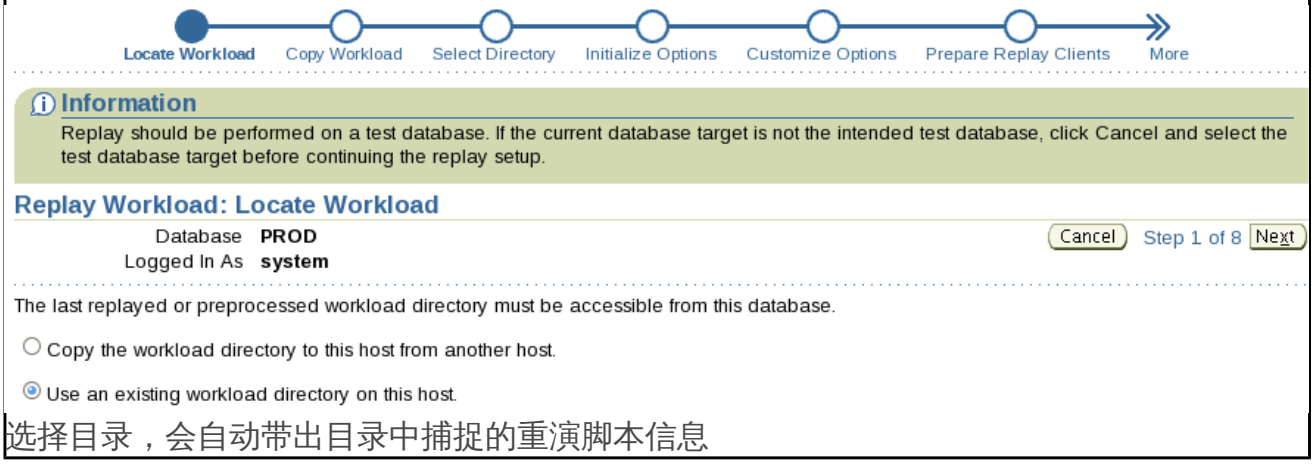
做成功后会在目录中生成 /ora/db/admin/PROD/repdir/pp11.2.0.3.0 目录，并且在下面会有很多 replay 包的信息

```
[root@edu1 pp11.2.0.3.0]# pwd
/ora/db/admin/PROD/repdir/pp11.2.0.3.0
[root@edu1 pp11.2.0.3.0]# ls -l
total 96
-rw-r----- 1 oradb oinstall  3508 May 24 15:56 wcr_calibrate.xml
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_commits.extb
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_conn_data.extb
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_data.extb
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_dep_graph.extb
-rw-r----- 1 oradb oinstall   603 May 24 15:56 wcr_login.pp
-rw-r----- 1 oradb oinstall    35 May 24 15:56 wcr_process.wmd
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_references.extb
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_scn_order.extb
-rw-r----- 1 oradb oinstall 12288 May 24 15:56 wcr_seq_data.extb
```

>> *Replay Workload 重演*

Replay Workload on Test Database	Set up the workload replay on the test database, copy the replay results to the workload staging area, and analyze the results.
Replay Workload	Replay the preprocessed workload on a test copy of the production database.
Copy to Workload Staging Area	Copy replay results to the workload staging area for comparision analysis with future replays.
Analyze Results	Analyze the effects of changes on workload performance.

使用已有的目录



选择目录，会自动带出目录中捕捉的重演脚本信息

Replay Workload: Select Directory

Cancel | Back | Step 3 of 8 | Next

Database **PROD**
Capture Name **CAPTURE-PROD-A**
Logged In As **system**

Select a directory object that contains the last replayed workload or a preprocessed workload.

Directory Object: repdir | Create Directory Object

Capture Summary

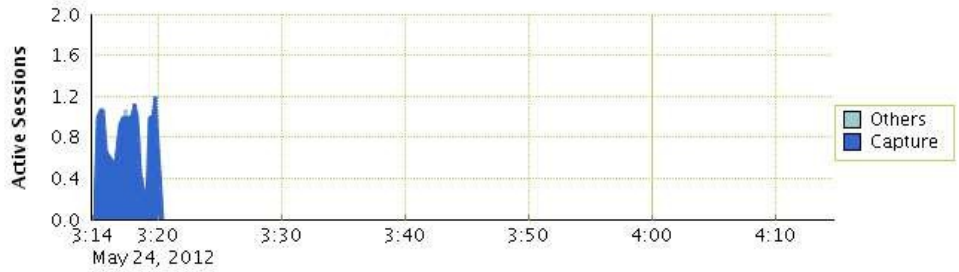
Name	**CAPTURE-PROD-A**	Captured Data Size (MB)	**58.12**
Status	**Completed**	Duration (hh:mm:ss)	**00:05:56**
Directory Object	**repdir**	Start Time	**May 24, 2012 3:14:38 PM CST**
Database Name	**PROD**	End Time	**May 24, 2012 3:20:34 PM CST**
Capture Database Version	**11.2.0.3.0**	Start SCN	**267909**
Cluster Database	**No**	End SCN	**1488124**
DBID	**195950047**	AWR Data Exported	**Yes**
Capture Error Code	**0**	Preprocessed Database Version	**11.2.0.3.0**
Capture Error Message	**None**		

Capture Details

Workload Profile | Workload Filters

View Workload Capture Report

Average Active Sessions



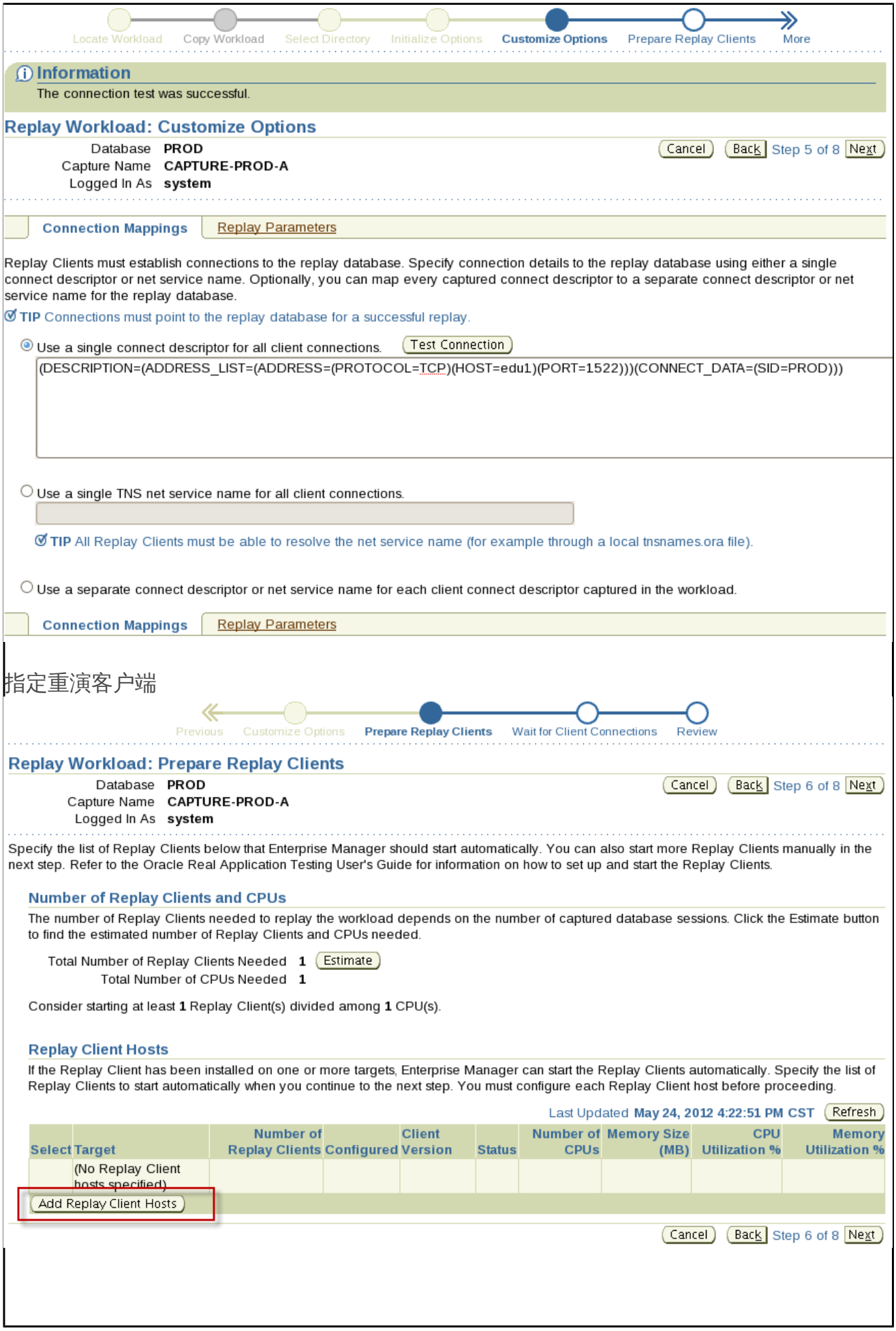
Comparison

	Total	Capture	Percentage of Total
Database Time (hh:mm:ss)	00:05:17	00:05:12	98.42
Average Active Sessions	0.89	0.88	98.42
User Calls	1,626	572	35.18
Transactions	1,000,008	1,000,002	100.00
Session Logins	12	6	50.00
Application Errors	N/A	0	N/A

惯例，要给个名字

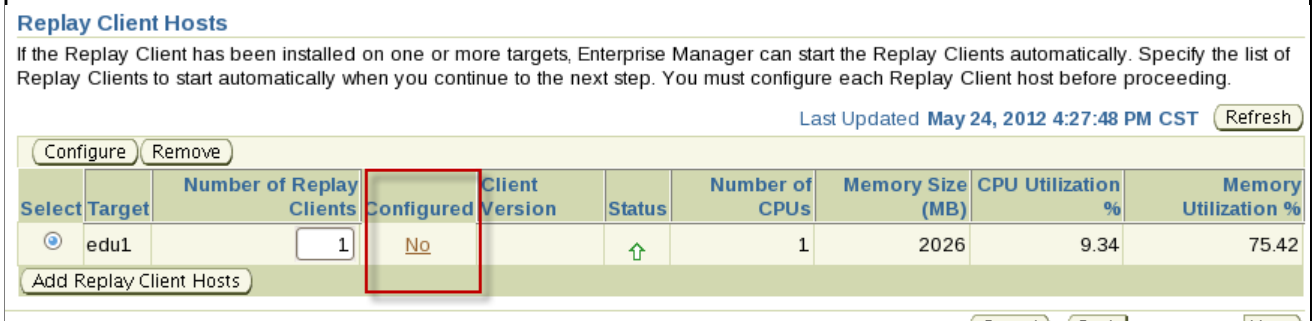

Locate Workload
Copy Workload
Select Directory
Initialize Options
Customize Options
Prepare Replay Clients
More
Replay Workload: Initialize Options
Database PROD
Capture Name CAPTURE-PROD-A
Logged In As system
Cancel
Back
Step 4 of 8
Next
* Replay Name
REPLAY-PROD-A
Identify Source
Choose the initial replay options.
Use the default replay options
Use replay options from a previous replay
Replay Name

指定重演连接方式


Locate Workload
Copy Workload
Select Directory
Initialize Options
Customize Options
Prepare Replay Clients
More
Information
The connection test was successful.
Replay Workload: Customize Options
Database PROD
Capture Name CAPTURE-PROD-A
Logged In As system
Cancel
Back
Step 5 of 8
Next
Connection Mappings
Replay Parameters
Replay Clients must establish connections to the replay database. Specify connection details to the replay database using either a single connect descriptor or net service name. Optionally, you can map every captured connect descriptor to a separate connect descriptor or net service name for the replay database.
TIP Connections must point to the replay database for a successful replay.
Use a single connect descriptor for all client connections.
Test Connection
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=edu1)(PORT=1522)))(CONNECT_DATA=(SID=PROD)))
Use a single TNS net service name for all client connections.
TIP All Replay Clients must be able to resolve the net service name (for example through a local tnsnames.ora file).
Use a separate connect descriptor or net service name for each client connect descriptor captured in the workload.
Connection Mappings
Replay Parameters

指定重演客户端

Previous
Customize Options
Prepare Replay Clients
Wait for Client Connections
Review
Replay Workload: Prepare Replay Clients
Database PROD
Capture Name CAPTURE-PROD-A
Logged In As system
Cancel
Back
Step 6 of 8
Next
Specify the list of Replay Clients below that Enterprise Manager should start automatically. You can also start more Replay Clients manually in the next step. Refer to the Oracle Real Application Testing User's Guide for information on how to set up and start the Replay Clients.
Number of Replay Clients and CPUs
The number of Replay Clients needed to replay the workload depends on the number of captured database sessions. Click the Estimate button to find the estimated number of Replay Clients and CPUs needed.
Total Number of Replay Clients Needed 1
Estimate
Total Number of CPUs Needed 1
Consider starting at least 1 Replay Client(s) divided among 1 CPU(s).
Replay Client Hosts
If the Replay Client has been installed on one or more targets, Enterprise Manager can start the Replay Clients automatically. Specify the list of Replay Clients to start automatically when you continue to the next step. You must configure each Replay Client host before proceeding.
Last Updated May 24, 2012 4:22:51 PM CST
Refresh
Select
Target
Number of Replay Clients
Configured
Client Version
Status
Number of CPUs
Memory Size (MB)
CPU Utilization %
Memory Utilization %
(No Replay Client hosts specified)
Add Replay Client Hosts
Cancel
Back
Step 6 of 8
Next



指定好重演 client 之后，配置他

配置好之后 apply 再 close 掉，配置成功的状态

Replay Client Hosts

If the Replay Client has been installed on one or more targets, Enterprise Manager can start the Replay Clients automatically. Specify the list of Replay Clients to start automatically when you continue to the next step. You must configure each Replay Client host before proceeding.

Last Updated **May 24, 2012 4:33:57 PM CST** Refresh

Configure Remove

Select	Target	Number of Replay Clients	Configured	Client Version	Status	Number of CPUs	Memory Size (MB)	CPU Utilization %	Memory Utilization %
◉	edu1	1	Yes		⇧	1	2026	7.49	75.22

Add Replay Client Hosts

Cancel Back Step 6 of 8 Next

下一步之后会进入 重演等待状态，另外开窗口手工启动重演命令 Number client 会变成 1 然后点 NEXT 开始重演

Replay Workload: Wait for Client Connections

Database **PROD**
Capture Name **CAPTURE-PROD-A**
Logged In As **system**

Cancel Back Step 7 of 8 Next

At this point all the clients that have been asked to start have been started. If you want to start more manually, do so now. Then proceed to the next step.

Client Connections

Host	Expected Number of Client Connections	Actual Number of Client Connections	Error Output
edu1	1	1	

Cancel Back Step 7 of 8 Next

```
wrc system/oracle mode=replay replaydir=/ora/db/admin/PROD/repdir
```

直接提交就可以了

Previous Customize Options Prepare Replay Clients Wait for Client Connections **Review**

Replay Workload: Review

Logged In As **system**

Cancel Back Step 8 of 8 Submit

Information

Time for resetting the clock: May 24, 2012 3:14:38 PM CST.

It is recommended that the system time on the database host platform be changed to a value that is close to the capture start time. This must be done just before replay is started. Not doing so might present an invalid data set to the replayed time-sensitive workload, thus causing data divergence. Examples include statements that use the SYSDATE and SYSTIMESTAMP functions. Resetting the time will also minimize job scheduling inconsistencies between replay and capture.

Workload CAPTURE-PROD-A will be replayed on database 'PROD'.

Database **PROD**
Capture Name **CAPTURE-PROD-A**
Replay Name **REPLAY-PROD-A**
Directory Object **repdir**
Connected Replay Clients **1**

Client Connections

Host	Expected Number of Client Connections	Actual Number of Client Connections	Error Output
edu1	1	1	

Cancel Back Step 8 of 8 Submit

看看执行状态 可以看到 capture 了多少时间，现在重演了多少时间，展开 Divergence 还可以看到重演是否有错误等信息

跑完之后点 ViewWorkload Replay Report 看看折腾半天的结果

View Workload Replay: REPLAY-PROD-A

Page Refreshed **May 24, 2012 4:46:34 PM CST** Refresh OK

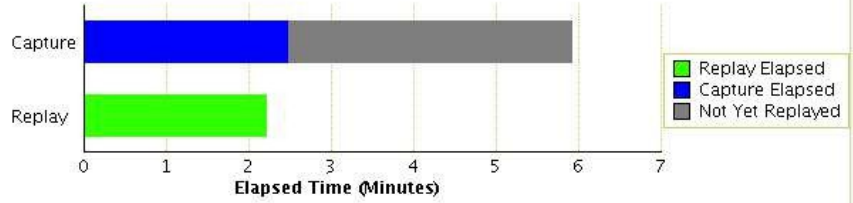
Status **In Progress** Stop Replay

Summary

Overview | Reports

Network Time (hh:mm:ss) **00:00:01**
Think Time (hh:mm:ss) **00:01:20**
Clients **1**
Clients Finished **0**

Elapsed Time Comparison



Assessing the Replay

The Elapsed Time Comparison chart shows how much time the replayed workload has taken to accomplish the same amount of work as captured.

When the Replay bar is shorter than the Capture bar, the replay environment is processing the workload faster than the capture environment.

The divergence table gives information about both the data and error discrepancies between the replay and capture environments, which can be used as a measure of the replay quality.

Detailed Comparison

	Capture	Replay	Percentage of Capture
Duration (hh:mm:ss)	00:05:56	00:02:13	37.36
Database Time (hh:mm:ss)	00:05:12	00:00:04	1.28
Average Active Sessions	0.88	0.03	3.43
User Calls	572	240	41.96

Divergence

Workload Profile

Top Events

(-) Hide

No data exists for this section of the report.

Top Service/Module/Action

(-) Hide

No data exists for this section of the report.

Top SQL with Top Events

(-) Hide

No data exists for this section of the report.

Top Sessions with Top Events

(-) Hide

No data exists for this section of the report.

Replay Divergence

Session Failures

By Application

(-) Hide

No data exists for this section of the report.

Error Divergence

By Application

(-) Hide

No data exists for this section of the report.

By SQL

(-) Hide

No data exists for this section of the report.

By Session

(-) Hide

No data exists for this section of the report.

DML Data Divergence

By Application

(-) Hide

No data exists for this section of the report.

By SQL

(-) Hide

No data exists for this section of the report.

By Divergence magnitude

(-) Hide

No data exists for this section of the report.

SELECT Data Divergence

By Application

(-) Hide

Service Name	Module Name	Action Name	Avg Rows Affected	Avg Absolute Rows Affected	Number of Distinct Sessions	Count	First Occurrence	Last Occurrence
SYS$USERS	PL/SQL Developer	SQL Window - select * from xx	-100	100	1	1	2012-05-24T16:45:42.671280+08:00	2012-05-24T16:45:42.671280+08:00

By Divergence magnitude

(-) Hide

Max divergence magnitude	Divergence distribution (%)	Count

DB Replay Report for REPLAY-PROD-A

DB Name	DB Id	Release	RAC	Replay Name	Replay Status
PROD	195950047	11.2.0.3.0	NO	REPLAY-PROD-A	COMPLETED

Replay Information

Information	Replay	Capture
Name	REPLAY-PROD-A	CAPTURE-PROD-A
Status	COMPLETED	COMPLETED
Database Name	PROD	PROD
Database Version	11.2.0.3.0	11.2.0.3.0
Start Time	24-05-12 16:44:20	24-05-12 15:14:38
End Time	24-05-12 16:50:06	24-05-12 15:20:34
Duration	5 minutes 46 seconds	5 minutes 56 seconds
Directory Object	repdir	repdir
Directory Path	/ora/db/admin/PROD/repdir	/ora/db/admin/PROD/repdir

Replay Options

Option Name	Value
Synchronization	SCN
Connect Time	100%
Think Time	100%
Think Time Auto Correct	TRUE
Number of WRC Clients	1 (1 Completed, 0 Running)

Replay Statistics

Statistic	Replay	Capture
DB Time	5.187 seconds	312.267 seconds
Average Active Sessions	.01	.88
User calls	572	572
Network Time	1.353 seconds	N/A
Think Time	553.346 seconds	N/A

Replay Divergence Summary

Divergence Type	Count	% Total
Session Failures During Replay	0	0.00
Errors No Longer Seen During Replay	0	0.00
New Errors Seen During Replay	0	0.00
Errors Mutated During Replay	0	0.00
DMLs with Different Number of Rows Modified	0	0.00
SELECTs with Different Number of Rows Fetched	1	0.17