11g OCM Upgrade EXAM Topic

目录

DOC List

Administrator's Guide
Backup and Recovery Reference
Backup and Recovery User's Guide
Data Guard Concepts and Administration
Data Warehousing Guide
Error Messages
Net Services Administrator's Guide
Net Services Reference
Performance Tuning Guide
PL/SQL Packages and Types Reference
Reference
Security Guide
Streams Replication Administrator's Guide

1 Database, RMAN, EM and Network Configuration

[Y]Configure server-side network

Net Services Reference=> 7 Oracle Net Listener Parameters (listener.ora)

#listener.ora

```
PROD=(ADDRESS_LIST= (ADDRESS=(PROTOCOL=tcp)(HOST=edu1)(PORT=1521)))
SID_LIST_PROD=(SID_LIST= (SID_DESC=(SID_NAME=prod)(ORACLE_HOME=/ora/db11g) ))
```

[Y]Configure client-side network

Net Services Reference=> 6 Local Naming Parameters (tnsnames.ora)

#tnsnames.ora

```
prod=(DESCRIPTION= (ADDRESS_LIST=
      (load_blance=yes)(FAILOVER=on)
      (ADDRESS=(PROTOCOL=tcp)(HOST=edu1)(PORT=1522))
      (ADDRESS=(PROTOCOL=tcp)(HOST=edu1)(PORT=1523)))
   (CONNECT_DATA=(SERVICE_NAME=prod))
```

```
PROD=(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edu1)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=PROD)))
PRODSTD=(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edu1)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=PRODSTD)))
```

[Y]Create and Manage encrypted tablespaces

Administrator's Guide=> 13 Managing Tablespaces=> Creating Tablespaces

Net Services Reference=> 5 Parameters for the sqlnet.ora File

#创建 wallet

```
mkdir -p /ora/db11g/admin/PROD/wallet
```

vi sqlnet.ora # 最后一行添加

```
WALLET_LOCATION=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/ora/db11g/admin/PROD/wallet/)))
```

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "welcome1";
```

#创建加密表空间

```
CREATE TABLESPACE securespace DATAFILE '/u01/app/oracle/oradata/orcl/secure01.dbf' SIZE 10M
ENCRYPTION DEFAULT STORAGE(ENCRYPT);
```

#创建使用 3DES168 进行加密的表空间

```
CREATE TABLESPACE securespace DATAFILE '/u01/app/oracle/oradata/orcl/secure01.dbf' SIZE 10M
ENCRYPTION USING '3DES168' DEFAULT STORAGE(ENCRYPT);
```

#Check

```
SELECT t.name, e.encryptionalg algorithm    FROM    v$tablespace t, v$encrypted_tablespaces e WHERE t.ts# = e.ts#;
NAME                                ALGORITHM
SECURESPACE                         3DES128
```

[Y]Create and Manage a tablespace that uses NFS mounted file system file

服务器端 修改文件 然后 service nfsd restart

vi /etc/exports

```
/ins *(rw)
```

客户端

mount nfs:/ins /ins

或者 修改 /etc/fstab

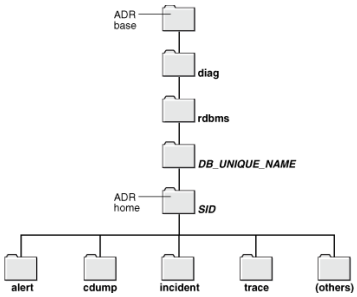
```
nfs:/ins /ins    nfs    rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0 0 0
```

[Y]Set up ADR file based repository

Administrator's Guide=> 9 Managing Diagnostic Data

log 及 trace 的统一管理模式 #Config ADR base

alter system set diagnostic_dest='/u01/oracle/log'




```
SELECT NAME, VALUE FROM V$DIAG_INFO;

NAME                 VALUE
-------------------- ----------------------------------------------------
Diag Enabled         TRUE
ADR Base             /u01/oracle/log
ADR Home             /u01/oracle/log/diag/rdbms/dbn/osi
Diag Trace           /u01/oracle/log/diag/rdbms/dbn/osi/trace
Diag Alert           /u01/oracle/log/diag/rdbms/dbn/osi/alert
Diag Incident        /u01/oracle/log/diag/rdbms/dbn/osi/incident
Diag Cdump           /u01/oracle/log/diag/rdbms/dbn/osi/cdump
Health Monitor       /u01/oracle/log/diag/rdbms/dbn/osi/hm
Default Trace File   /u01/oracle/log/diag/rdbms/dbn/osi/trace/osi_ora_10533.
Active Problem Count 0
Active Incident Count 0
```

[Y]Perform cold database backup

冷备，直接关了数据库 CP

[Y]Manage user accounts and use case sensitive passwords

Security Guide=> 3 Configuring Authentication

敏感密码，主要是密码安全性方面的设置。

#安装密码验证 function

conn / as sysdba

@?/rdbms/admin/utlpwdmg.sql

--配置给指定 profile

ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION verify_function_11g;

--撤销 profile 中的密码验证

ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION NULL;

alter user test profile default;

[Y]Use OPatch to install a patch

#用 opatch 修补或升级数据库部分模块

$ORACLE_HOME/OPatch/opatch apply

$ORACLE_HOME/OPatch/opatch lsinventory

cat /etc/oraInst.loc

```
inventory_loc=/ora/oraInventory
inst_group=oinstall
```

cat /ora/oraInventory/ContentsXML/inventory.xml

```
<?xml version="1.0" standalone="yes" ?>
<!-- Copyright (c) 2009 Oracle Corporation. All rights Reserved -->
<!-- Do not modify the contents of this file by hand. -->
<INVENTORY>
<COMPOSITEHOME_LIST>
</COMPOSITEHOME_LIST>
<VERSION_INFO>
   <SAVED_WITH>10.2.0.5.0</SAVED_WITH>
   <MINIMUM_VER>2.1.0.6.0</MINIMUM_VER>
</VERSION_INFO>
<HOME_LIST>
<HOME NAME="OraDb11g_home1" LOC="/ora/db/11g" TYPE="O" IDX="1"/>
<HOME NAME="db10g" LOC="/ora/GC/db10g" TYPE="O" IDX="2"/>
<HOME NAME="oms10g" LOC="/ora/GC/oms10g" TYPE="O" IDX="3"/>
<HOME NAME="agent10g" LOC="/ora/GC/agent10g" TYPE="O" IDX="4"/>
</HOME_LIST>
</INVENTORY>
```

[Y]Install and configure EM Agent

web 界面下载文件 http://edu2:4889/agent_download/ 或者手工安装，并且指定端口及密码等信息

进入 10.2.0.1.1/linux 找到 agentDownload.linux 右键 save link as 到本地盘

```
export PATH=$PATH:$ORACLE_HOME/jdk/bin
agentDownload.linux -b /ora
#最终安装路径为 /ora/agent10g
```

代理的管理#重点看的是上传时间，如果为空则证明代理不通，检查网络

```
cd /ora/agent10g
./emctl status agent
./emctl upload
./emctl config addTarget -home/oracle/newTarget.xml
```

```
agentca -d #重新配置
agentca -f -c "node1 node2"
```

[Y]Create Base Recovery Catalog

Backup and Recovery User's Guide=> 12 Managing a Recovery Catalog

```
#创建表空间
CREATE TABLESPACE RCAT DATAFILE '/ora/DB/oradata/EMREP/rcat.dbf' SIZE 100M AUTOEXTEND ON;
#创建用户
CREATE USER RCADMIN IDENTIFIED BY rcadmin DEFAULT TABLESPACE RCAT ACCOUNT UNLOCK
GRANT CONNECT TO RCADMIN;
alter user rc_admin quota unlimited on rcat;
grant recovery_catalog_owner to rc_admin;
#创建 catalog
rman target sys/oracle@prod catalog rcadmin/rcadmin@emrep  #连接目标库与编目库
create catalog;  #创建编目
register database; #注册
report schema #  验证
```

[Y]Configure RMAN

Backup and Recovery User's Guide=> 5 Configuring the RMAN Environment

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
#保留策略 备份保留几份 # report obsolete #report need backup #delete obsolete  #记录及是被备份文件
CONFIGURE RETENTION POLICY TO recovery window of 3 days;  #可以恢复到 3 天内任何一个时间点
```

```
CONFIGURE BACKUP OPTIMIZATION OFF; # default#设定备份优化 完整备份后不发变化的文件下次不进行备份，
强制备份忽略掉此参数 backup database force
```

```
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
# CONFIGURE DEFAULT DEVICE TYPE TO SBT; 设置备份设备
```

```
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
#知否自动备份控制文件和 init 文件 建议打开，当控制文件发生变化时，自动进行备份
```

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
#自动备份的备份文件保存的位置，默认闪回区，或者 dbs 下面
```

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2 BACKUP TYPE TO compress BACKUPSET;
#备份并发度 #备份集 #copy 完全和原文件相同
```

```
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default #备份 copy 备份几份
```

```
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default #备份  copy 归档 备份几份
```

```
CONFIGURE MAXSETSIZE TO UNLIMITED; # default #单个备份集大小限制
```

```
#加密备份 Backup and Recovery Advanced User's Guide
CONFIGURE ENCRYPTION FOR DATABASE ON;  -- 开启加密备份功能
set encryption on identified by '111111' only;  --设置加密密码
```

backup tablespace tbs1; --备份

set decryption identified by '111111'; --设置解密密码

restore tablespace tbs1 -- 恢复加密备份

CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default

CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default

CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/ora/DB/dbs/snapcf_prod.f'; # default

#控制文件的快照 RAC 环境中快照需要指定在共享存储中

CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET PARALLELISM 1; --压缩备份

backup as compressed backupset tablespace system;

CONFIGURE CHANNEL DEVICE TYPE DISK format '/ora/backup/backup_%U' --指定备份到某个目录

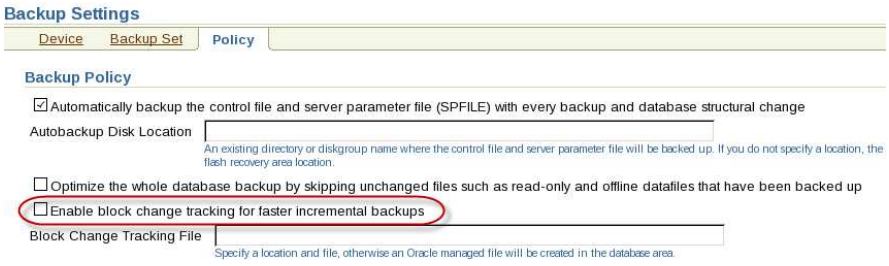
backup database format '/ora/backup/backup_%U'

enable fast incremental backup

alter database enable block change tracking using file '/ora/oradata/PROD.block.trace';

alter database disable block change tracking;

SELECT * FROM v$block_change_tracking;



[Y]Perform multisection backup of a datafile

Backup and Recovery User's Guide=> 9 Backing Up the Database: Advanced Topics 把一个文件被分成多个备份文件

backup section size 200M datafile 1 format '/ora/backup/%U_.bk';

[Y]Create an Archival Backup

Backup and Recovery User's Guide=> 8 Backing Up the Database

backup archivelog all delete input;

```
BACKUP ARCHIVELOG FROM  SEQUENCE 121 UNTIL SEQUENCE 125;
```

2 Data Guard

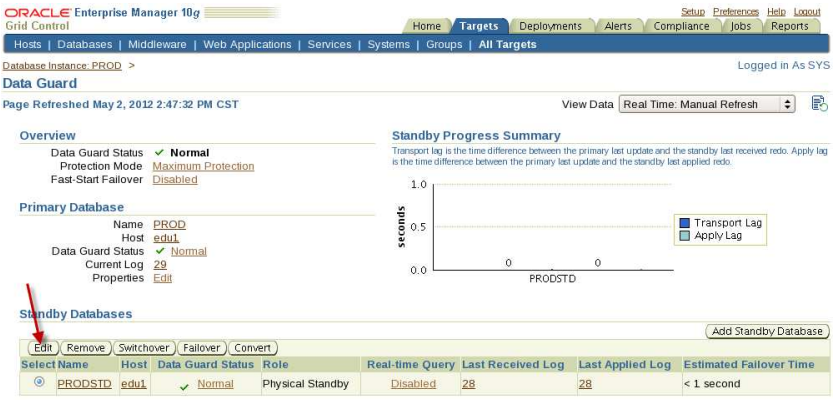
[Y]Create Physical Standby Database with real-time apply.

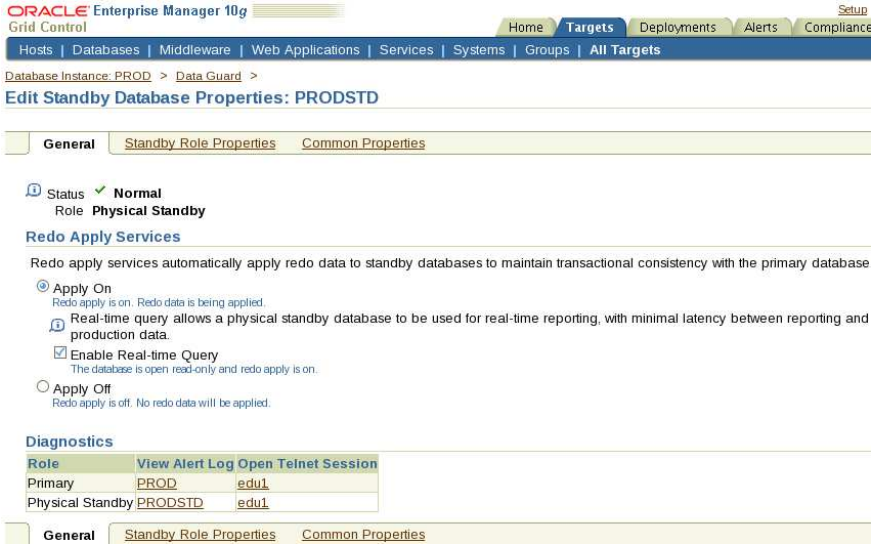
Data Guard Concepts and Administration=>3 Creating a Physical Standby Database

Data Guard Concepts and Administration=>7 Apply Services

GC 建好 DataGuard 之后在 DataGuard 属性页面里面进行修改成 real-time apply

做好之后 show parameter log 查看 DG 日志传输为 **LGWR SYNC AFFIRM**





[Y]Configure the data guard environment to reduce overheads of fast incremental backups on the primary database

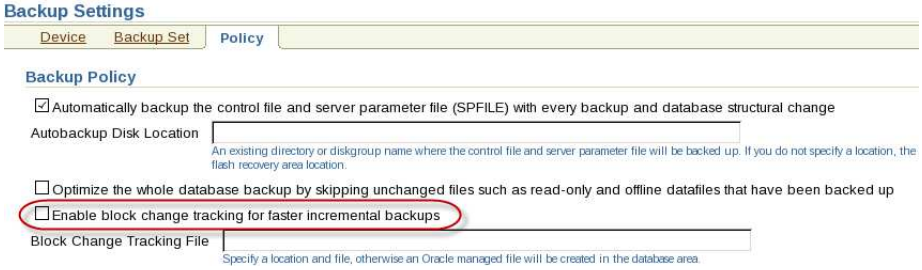
使用快速增量备份的备份集创建 data guard?

enable fast incremental backup

alter database enable block change tracking using file '/ora/oradata/PROD.block.trace';

alter database disable block change tracking;

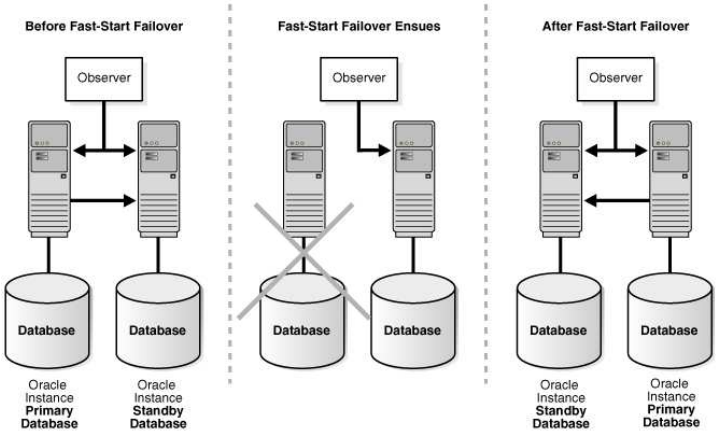
SELECT * FROM v$block_change_tracking;



[Y]Configure the Observer

Data Guard Broker=> 5 Switchover and Failover Operations => 5.5 Fast-Start Failover

Figure 5-1 Relationship of Primary and Standby Databases and the Observer



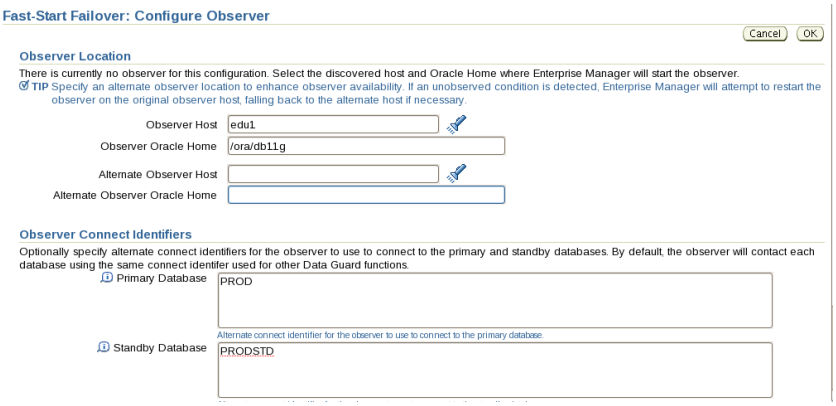
指定 OBserver 进行变更记录以达到快速切换的目的.

手工开启 OBserver 进程#我们在主库上面做测试，如果 GC 起不来 OB 的时候再做.

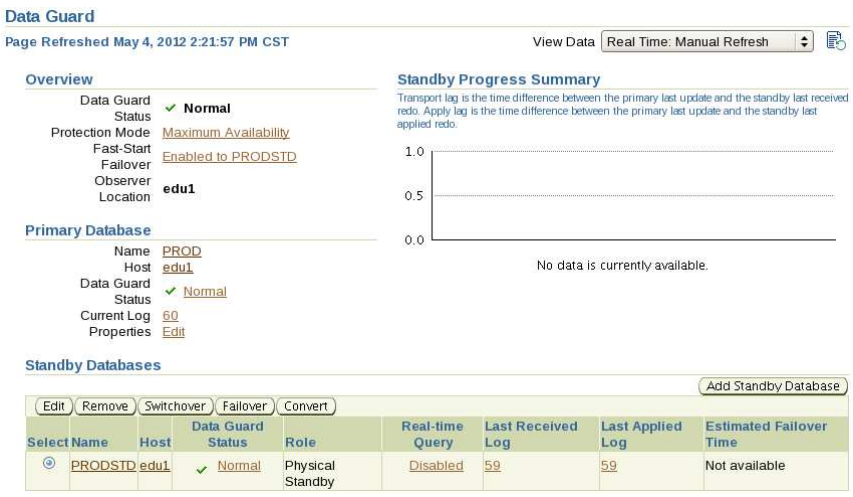
```
$>dgmgrl
DGMGRL>connect sys/oracle@prod
DGMGRL>start observer file='/ora/observer.dat'
#启动后会一直挂在那里进行监听
```

配置 DG #注意：不能在 snapshot standby 上做

在界面里面点 **Fast-Start Failover** 进入配置界面



做好的效果

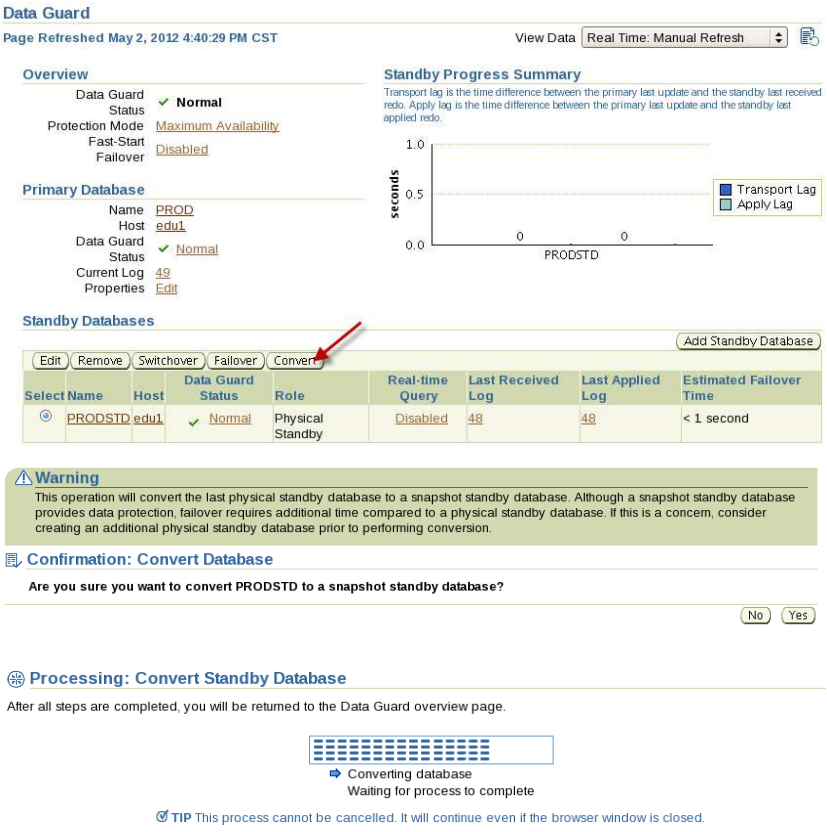


[Y]Switchover and switch back

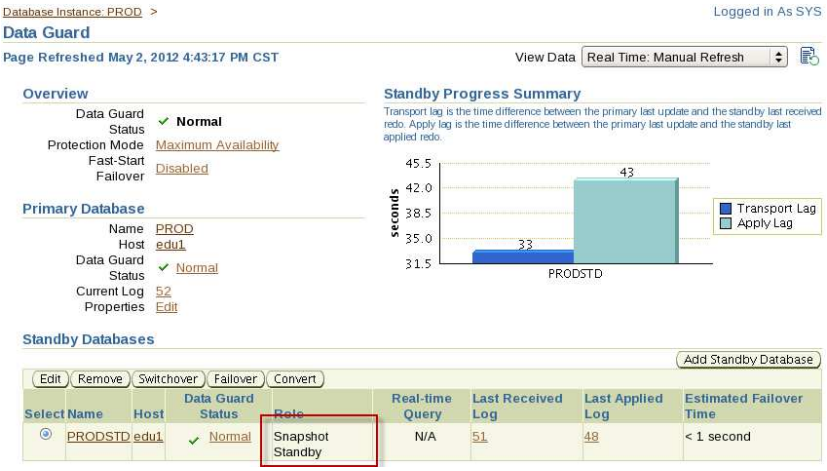
GC 中操作，切换回来之前先在当前主库 switch 几下 logfile 以避免 GC 报错

[Y]Convert the standby to a snapshot standby

Data Guard Broker=> 6 Scenarios Using the DGMGRL Command-Line Interface



中间可能报错，重新刷新一下就好了，最终应该是这个样子的



[Y]Configure archivelog deletion policy for the dataguard configuration

#PROD RMAN **CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;** **CONFIGURE DB_UNIQUE_NAME PRODSTD CONNECT IDENTIFIER 'PRODSTD';**
#PRODSTD RMAN **CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 1 TIMES TO DEVICE TYPE DISK;**

#PROD CONFIGURE DB_UNIQUE_NAME 'PRODSTD' CONNECT IDENTIFIER 'PRODSTD'; CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;	# PRODSTD CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 1 TIMES TO DISK;

3 Data and Data Warehouse Management

[Y]Troubleshoot fast materialized views to fast refresh and query rewrite

Data Warehousing Guide=> 8 Basic Materialized Views

Data Warehousing Guide=> 9 Advanced Materialized Views

-- MGV 多表合并组合查询物化视图 -- MAV 单表查询物化视图 -- 权限

grant execute on dbms_mview to hr;

grant create materialized view to hr;

-- create mview log

create materialized view log on t with rowid ,sequence(id,name) including new values ;

-- 语句改写

select id,name from t group by id,name;

等效于

select distinct id,name from t;

-- 快刷的

create materialized view mv1 refresh fast on commit enable query rewrite as select id,name from t group by id,name;

-- 快刷+自动刷新 # 每分钟一次

create materialized view mv1 refresh fast **start with sysdate next sysdate+1/24/60** as select id,name from t group by id,name;

--手工快刷

exec dbms_mview.refresh('mv1','fast');

[Y]Add a tablespace by using Transportable Tablespace Feature of Data Pump Import (cross platform transportable tablespace)

Administrator's Guide=> 13 Managing Tablespaces => Transporting Tablespaces Between Databases

同构/异构迁移 可 GC

先要导入一个 dmp 文件到 PROD 需要 **fromuser=ssh touser=oltp_user** 然后传输到 EVEN 的 EMREP 上并导入表空间。源库，在 GC 中操作，表空间定义做成一个 DMP 文件，并且将数据文件复制到同一目录，目标库，在 GC 中进行导入操作，指定 DMP 文件和数据文件，选择导入后文件的位置后进行导入

[Y]Configure a schema to support a star transformation query

Note: Oracle does not recommend setting CURSOR_SHARING to FORCE in a DSS environment or if you are using complex queries. Also, star transformation is not supported with CURSOR_SHARING set to either SIMILAR or FORCE. For more information, see the "Enabling Query Optimizer Features" on page 11-5.

alter system set cursor_sharing=exact;

alter system set star_transformation_enabled=TRUE;

TRUE：Oracle 优化器自动识别语句中的事实表和约束维度表并进行星型转换。这一切优化尝试都在 CBO 的范畴内，优化器需要确定转换后的执行计划成本要低于不转换的执行计划；同时优化器还会尝试利用物化的临时表，如果那样真的好的话。

FALSE：优化器不会考虑星型转换。

TEMP_DISABLE:当一个维度表超过 100 个块时， 如果简单地设置 star_transformation_enabled 为 TRUE 来启用星型变换，那么会话会创建一个内存中的全局临时表(global temporary table)来保存已过滤的维度数据，这在过去会造成很多问题；这里说的 100 个块其实是隐式参数 _temp_tran_block_threshold(number of blocks for a dimension before we temp transform)的默认值，此外隐式参数 _temp_tran_cache(determines if temp table is created with cache option， 默认为 TRUE)决定了这类临时表是否被缓存住；为了避免创建全局临时表可能带来的问题，就可以用到 TEMP_DISABLE 这个禁用临时 表的选项，让优化器不再考虑使用物化的临时表。

示例：我们以 ORACLE 默认 SAMPLE SH 为例

SELECT c.cust_city, t.calendar_quarter_desc, SUM(s.amount_sold) sales_amount

```
  FROM sh.sales s, sh.times t, sh.customers c, sh.channels ch
 WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id AND s.channel_id = ch.channel_id
   AND c.cust_state_province = 'FL' AND ch.channel_desc = 'Direct Sales'
   AND t.calendar_quarter_desc IN ('2000-01', '2000-02','1999-12')
 GROUP BY c.cust_city, t.calendar_quarter_desc;
```

未开启 star_transformation_enabled 的执行计划

```
-------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name       | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |            |   607 | 46132 |   968   (3)| 00:00:12 |       |       |
|   1 |  HASH GROUP BY                     |            |   607 | 46132 |   968   (3)| 00:00:12 |       |       |
|*  2 |   HASH JOIN                        |            |  2337 |   173K|   967   (3)| 00:00:12 |       |       |
|   3 |    PART JOIN FILTER CREATE         | :BF0000    |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|*  4 |     TABLE ACCESS FULL              | TIMES      |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|*  5 |    HASH JOIN                       |            | 12456 |   729K|   948   (3)| 00:00:12 |       |       |
|   6 |     MERGE JOIN CARTESIAN           |            |   383 | 14937 |   409   (1)| 00:00:05 |       |       |
|*  7 |      TABLE ACCESS FULL             | CHANNELS   |     1 |    13 |     3   (0)| 00:00:01 |       |       |
|   8 |      BUFFER SORT                   |            |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|*  9 |       TABLE ACCESS FULL            | CUSTOMERS  |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|  10 |     PARTITION RANGE JOIN-FILTER    |            |   918K|    18M|   533   (3)| 00:00:07 |:BF0000|:BF0000|
|  11 |      TABLE ACCESS FULL             | SALES      |   918K|    18M|   533   (3)| 00:00:07 |:BF0000|:BF0000|
-------------------------------------------------------------------------------------------------------------------------

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

   2 - access("S"."TIME_ID"="T"."TIME_ID")
   4 - filter("T"."CALENDAR_QUARTER_DESC"='1999-12' OR "T"."CALENDAR_QUARTER_DESC"='2000-01' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-02')
   5 - access("S"."CUST_ID"="C"."CUST_ID" AND "S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   7 - filter("CH"."CHANNEL_DESC"='Direct Sales')
   9 - filter("C"."CUST_STATE_PROVINCE"='FL')
```

开启 star_transformation_enabled 后的执行计划

```
-------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                    | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                         |   238 | 13566 |   551   (1)| 00:00:07 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION         |                         |       |       |            |          |       |       |
|   2 |   LOAD AS SELECT                   | SYS_TEMP_0FD9D6617_D3180|       |       |            |          |       |       |
|*  3 |    TABLE ACCESS FULL               | CUSTOMERS               |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|   4 |   HASH GROUP BY                    |                         |   238 | 13566 |   145   (3)| 00:00:02 |       |       |
|*  5 |    HASH JOIN                       |                         |   238 | 13566 |   144   (2)| 00:00:02 |       |       |
|*  6 |     HASH JOIN                      |                         |   238 |  9996 |   142   (2)| 00:00:02 |       |       |
|*  7 |      TABLE ACCESS FULL             | TIMES                   |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|   8 |      VIEW                          | VW_ST_A3F94988          |   238 |  6188 |   123   (1)| 00:00:02 |       |       |
|   9 |       NESTED LOOPS                |                         |   238 | 13566 |   100   (1)| 00:00:02 |       |       |
|  10 |        PARTITION RANGE SUBQUERY    |                         |   237 |  6660 |    56   (2)| 00:00:01 |KEY(SQ)|KEY(SQ)|
|  11 |         BITMAP CONVERSION TO ROWIDS|                         |   237 |  6660 |    56   (2)| 00:00:01 |       |       |
|  12 |          BITMAP AND                |                         |       |       |            |          |       |       |
|  13 |           BITMAP MERGE             |                         |       |       |            |          |       |       |
|  14 |            BITMAP KEY ITERATION    |                         |       |       |            |          |       |       |
|  15 |             BUFFER SORT            |                         |       |       |            |          |       |       |
|* 16 |              TABLE ACCESS FULL     | CHANNELS                |     1 |    13 |     3   (0)| 00:00:01 |       |       |
|* 17 |            BITMAP INDEX RANGE SCAN | SALES_CHANNEL_BIX       |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  18 |           BITMAP MERGE             |                         |       |       |            |          |       |       |
|  19 |            BITMAP KEY ITERATION    |                         |       |       |            |          |       |       |
|  20 |             BUFFER SORT            |                         |       |       |            |          |       |       |
|* 21 |              TABLE ACCESS FULL     | TIMES                   |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|* 22 |            BITMAP INDEX RANGE SCAN | SALES_TIME_BIX          |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  23 |           BITMAP MERGE             |                         |       |       |            |          |       |       |
|  24 |            BITMAP KEY ITERATION    |                         |       |       |            |          |       |       |
|  25 |             BUFFER SORT            |                         |       |       |            |          |       |       |
|  26 |              TABLE ACCESS FULL     | SYS_TEMP_0FD9D6617_D3180|   383 |  1915 |     2   (0)| 00:00:01 |       |       |
|* 27 |            BITMAP INDEX RANGE SCAN | SALES_CUST_BIX          |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  28 |       TABLE ACCESS BY USER ROWID   | SALES                   |     1 |    29 |    67   (0)| 00:00:01 | ROWID | ROWID |
|  29 |     TABLE ACCESS FULL              | SYS_TEMP_0FD9D6617_D3180|   383 |  5745 |     2   (0)| 00:00:01 |       |       |
-------------------------------------------------------------------------------------------------------------------------

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

   3 - filter("C"."CUST_STATE_PROVINCE"='FL')
   5 - access("ITEM_1"="CO")
   6 - access("ITEM_2"="T"."TIME_ID")
   7 - filter("T"."CALENDAR_QUARTER_DESC"='1999-12' OR "T"."CALENDAR_QUARTER_DESC"='2000-01' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-02')
  16 - filter("CH"."CHANNEL_DESC"='Direct Sales')
  17 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  21 - filter("T"."CALENDAR_QUARTER_DESC"='1999-12' OR "T"."CALENDAR_QUARTER_DESC"='2000-01' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-02')
  22 - access("S"."TIME_ID"="T"."TIME_ID")
  27 - access("S"."CUST_ID"="CO")

Note
-----
   - star transformation used for this statement
```

我们可以看到执行计划尾部的 Note 中已经明确提示了使用 star transformation used for this statment

[Y]Configure and use parallel execution for queries

优先顺序 Hint > session > object

-- hint

```
select /*+ parallel(4)*/ * from hr.employees;

select /*+ parallel(hr.employees,4)*/ * from hr.employees;
```

-- objects

```
alter table employees parallel 4;

alter index emp_id parallel 4;
```

-- session

```
alter session enable parallel query;

alter session enable parallel dml;
```

alter session enable parallel ddl;

SELECT pdml_status,PQ_STATUS FROM v$session where sid=170;

并行查询会占用更多的内存排序区，并行成本=成本*并行数*2,例如开 1G 的排序区，20 个并行进程，那么在极限情况下内存占用应为 20*1G*2=40G 内存小于此量会占用 pagefile 性能急剧下降

alter system set parallel_max_servers=50 -- 建议至少 2 倍于表指定并行的数量

redo 并行 x$ksppi 中查询隐含参数

[Y]Use and access SecureFile LOBS

SecureFiles and Large Objects Developer's Guide=> 4 Using Oracle SecureFiles LOBs

Net Services Reference=> 5 Parameters for the sqlnet.ora File

Administrator's Guide=> 13 Managing Tablespaces => Creating Tablespaces=> Encrypted Tablespaces

SQL Language Reference=> 12 SQL Statements: ALTER TABLE to ALTER TABLESPACE

SQL Language Reference=> 16 SQL Statements: CREATE SYNONYM to CREATE TRIGGER

启用

mkdir -p /ora/db11g/admin/PROD/wallet

vi sqlnet.ora # 最后一行添加

```
WALLET_LOCATION=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/ora/db11g/admin/PROD/wallet/)))
```

SQL>ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "welcome1";

#报错 ORA-28368: cannot auto-create wallet 的话 是目录或者 sqlnet.ora 的问题

#SQL> ALTER SYSTEM SET db_securefile = 'ALWAYS'; -- 修改模式

ALWAYS：尝试将所有 LOB 创建为 SecureFile LOB，但是仅可将自动段空间管理(ASSM) 表空间外的任何 LOB 创建为 BasicFile LOB	FORCE：强制将所有 LOB 创建为 SecureFile LOB	PERMITTED：允许创建 SecureFiles（默认值）	NEVER：禁止创建 SecureFiles	IGNORE：禁止创建 SecureFiles，并忽略使用 SecureFiles 选项强制创建 BasicFiles 而导致的任何错误

securefile LOB.txt

-- 创建 securefile

CREATE TABLE t1 (a CLOB ENCRYPT **IDENTIFIED BY foo**) LOB(a) STORE AS SECUREFILE (CACHE); --指定密码的 securefile

CREATE TABLE lob_1(id number, doc CLOB **ENCRYPT USING 'AES128'**) LOB(doc) STORE AS **SECUREFILE (DEDUPLICATE LOB)**;

CREATE TABLE lob_2 (id number, doc CLOB) LOB(doc) STORE AS **SECUREFILE(COMPRESS HIGH KEEP_DUPLICATES)** ;

-- 由于创建好的 securefile 未创建相应的 LOB 段，无法看到已经启用 securefile 我们插入几条数据后才能看到加密段

insert into lob_1 values(1,'asdfgjkwqetui2rtyu');

insert into lob_2 values(1,'asdfgjkwqetui2rtyu');

commit;

--此时可以看到数据已经有 securefile 加密的 LOB 段了

SELECT segment_name, segment_type, segment_subtype FROM user_segments where segment_type like 'LOB%';

```
SEGMENT_NAME                         SEGMENT_TYPE       SEGMENT_SU
------------------------------------ ------------------ ----------
SYS_IL0000107267C00002$$             LOBINDEX           ASSM
SYS_IL0000107270C00002$$             LOBINDEX           ASSM
SYS_LOB0000107267C00002$$            LOBSEGMENT         SECUREFILE
SYS_LOB0000107270C00002$$            LOBSEGMENT         SECUREFILE
```

注意：索引不被加密

delete lob_1;	truncate table lob_1;	drop table lob_1 purge;
delete lob_2;	truncate table lob_2;	drop table lob_2 purge;
commit;-- 删除数据后段仍然存在	--清空表后加密段还健在	-- 这样就删没了

ALTER TABLE t1 MODIFY LOB(a) (**KEEP_DUPLICATES**); #禁用取消重复

ALTER TABLE t1 MODIFY **PARTITION p1 LOB(a) (DEDUPLICATE LOB)**; #启用分区取消重复

ALTER TABLE t1 MODIFY LOB(a) (**NOCOMPRESS**); #禁用压缩

ALTER TABLE t1 MODIFY **PARTITION p1 LOB(a) (COMPRESS HIGH)**; #对单一分区中的 SecureFiles 启用压缩

ALTER TABLE t1 MODIFY **(a CLOB ENCRYPT USING '3DES168')**; #使用 3DES168 启用加密

ALTER TABLE t1 MODIFY **PARTITION p1(LOB(a) (ENCRYPT)**; #对分区启用加密

ALTER TABLE t1 MODIFY **(a CLOB ENCRYPT IDENTIFIED BY ghYtp)**; #使用口令启用加密并构建加密密钥

SecureFiles 迁移：示例

```
CREATE TABLE cust(c_id NUMBER PRIMARY KEY,c_zip NUMBER,c_name VARCHAR(30) DEFAULT NULL,c_lob CLOB);
INSERT INTO cust VALUES(1, 94065, 'hhh', 'ttt');
commit;
```

```
CREATE TABLE cust_int(c_id NUMBER NOT NULL,c_zip NUMBER,c_name VARCHAR(30) DEFAULT NULL,c_lob CLOB)
LOB(c_lob) STORE AS SECUREFILE (NOCACHE FILESYSTEM_LIKE_LOGGING);
```

```
-- 注意修改文档中 owner 的部分
DECLARE
col_mapping VARCHAR2(1000);
BEGIN
-- map all the columns in the interim table to the original table
col_mapping :='c_id c_id , '||'c_zip c_zip , '||'c_name c_name, '||'c_lob c_lob';
DBMS_REDEFINITION.START_REDEF_TABLE('SH', 'cust', 'cust_int', col_mapping);
END;
/
DECLARE
error_count pls_integer := 0;
BEGIN
DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS('SH', 'cust', 'cust_int', 1, TRUE,TRUE,TRUE,FALSE, error_count);
DBMS_OUTPUT.PUT_LINE('errors := ' || TO_CHAR(error_count));
END;
/
EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE('sh', 'cust', 'cust_int');
```

```
select table_name , column_name ,segment_name,SECUREFILE    from user_lobs;
```

[Y]Create partitioned tables (includes reference and interval partitioning)

SQL Language Reference=> 16 SQL Statements: CREATE SYNONYM to CREATE TRIGGER

 # interval partitioning 按照指定规则自动创建分区,初始分区为创建时指定的分区,当新数据插入时自动创建新分区

```
CREATE TABLE interval_demo (    customer_id number, name varchar2(20))
PARTITION BY RANGE (customer_id) INTERVAL (10)    (PARTITION p1 VALUES LESS THAN (10) tablespace users);
```

检查命令

```
col high_value format a20
col segment_name format a20
col partition_name format a20
SELECT partition_name, high_value FROM user_tab_partitions    WHERE table_name like    '%DEMO%';
select    SEGMENT_NAME,PARTITION_NAME from user_segments where segment_name like '%DEMO%';
```

刚创建好的状态

```
PARTITION_NAME       HIGH_VALUE
-------------------- --------------------
P1                   10

SEGMENT_NAME         PARTITION_NAME
-------------------- --------------------
INTERVAL_DEMO        P1
```

--插入 50 条数据

```
begin
for a in 1..50 loop
insert into interval_demo
values(a,'NAME');
end loop;
commit;
end;
/
```

```
PARTITION_NAME       HIGH_VALUE
-------------------- --------------------
P1                   10
SYS_P21              20
SYS_P22              30

SYS_P23              40
SYS_P24              50
SYS_P25              60

SEGMENT_NAME         PARTITION_NAME
-------------------- --------------------
INTERVAL_DEMO        P1
INTERVAL_DEMO        SYS_P21
INTERVAL_DEMO        SYS_P22
INTERVAL_DEMO        SYS_P23
INTERVAL_DEMO        SYS_P24
INTERVAL_DEMO        SYS_P25
```

#Reference Partitioning 外键自动关联主键分区

```
--主分区表 必须有主键，且主键关键字 not null 并且不支持 interval 形式分区表
CREATE TABLE ref_m_demo(customer_id number not null,name varchar2(20),
CONSTRAINT ref_m_demo_pk PRIMARY KEY(customer_id))
PARTITION BY RANGE (customer_id) (PARTITION p1 VALUES LESS THAN (10),PARTITION p2 VALUES LESS THAN (20));
```

```
-- 外键同步分区表,同样，外键关联字段必须 not null
```

```
CREATE TABLE ref_s_demo (emp_id number not null, city varchar2(20),
CONSTRAINT      ref_s_demo_fk FOREIGN KEY(emp_id) REFERENCES ref_m_demo(customer_id))
PARTITION BY REFERENCE (ref_s_demo_fk);
```

```
col owner format a5
col segment_name format a20
col partition_name format a20
select OWNER, SEGMENT_NAME,PARTITION_NAME
from dba_segments where    SEGMENT_NAME like 'REF%';
```

```
OWNER SEGMENT_NAME         PARTITION_NAME
----- -------------------- --------------
SH    REF_M_DEMO           P1
SH    REF_M_DEMO           P2
SH    REF_M_DEMO_PK
SH    REF_S_DEMO           P1
SH    REF_S_DEMO           P2
```

分区类型

Range　数值范围

Hash　关键词值进行分区，分的更均匀，适合等值查询

List　等于指定值，精确匹配分区

range-hash 组合分区，大 range 小 hash

Range-list 组合分区，大 range 小 list 子分区定义模板，如果在本分区之内特意指定了子分区信息则不使用模板

alter table p1 rename partition part_1 to partx_1;　--分区改名字

alter table p1 truncate partition partx_1 **update index**;　--清除分区数据自动更新索引

alter table p1 drop partition part_3; -- 删除分区时自动清理掉本地索引　删除分区

alter table p1 add partition part_4 values less than ('40') tablespace users; --增加 range 分区

ALTER TABLE "HR"."P1" MOVE PARTITION "PART_MAX" TABLESPACE "P3" ; --移动

alter table p1 split partition part_4 at ('35');　---range partition　切割分区

alter table p3 split partition part_1 values('good');　---list partition 切割分区

alter table p1 merge partitions part_3,part_4;　---range ,list 合并分区

alter table p2 coalesce partition; --收缩 hash 分区，每次减少一个合并分区

alter table p3 modify partition part_4 drop values('zzz');　--修改 list 分区分区关键字

alter table p3 modify partition part_4 add values('zzz'); --修改 list 分区分区关键字

分区表的分区与普通表进行交换

```
alter table p1 exchange partition part_1 with table test [without validation]; 不校验数据是否符合分区规范
alter table p1 exchange partition part_1 with table test [with validation];--default 校验数据
alter table p1 exchange partition part_1 with table test [including indexes];
alter table p1 exchange partition part_1 with table test [excluding indexes];--default
```

导出一个分区

```
exp hr/hr tables=p1:part_1 file=p1_part_1.dmp
```

移动分区，包括 LOB 的

```
alter table image_table move tablespace users lob (image_DATA) store as (tablespace users);
```

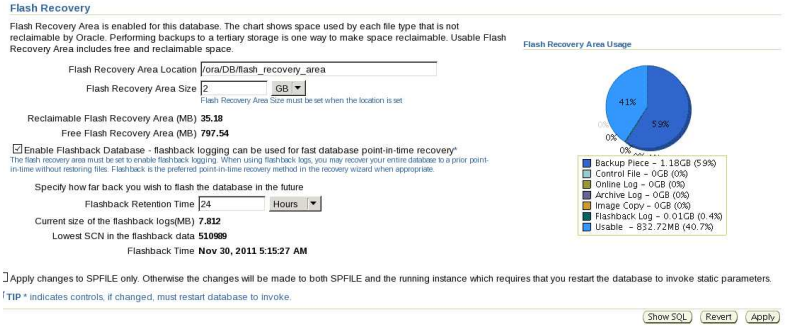
[Y]Configure Flashback Data Archive

配置

```
alter system set DB_RECOVERY_FILE_DEST='/ora/archive/EDU' scope=spfile;
alter system set DB_RECOVERY_FILE_DEST_SIZE =2G    scope=spfile;
alter system set    db_flashback_retention_target=4320 scope=spfile; -- 保存 3 天
```

```
shutdown immediate -- 启用
startup mount
alter database flashback on;
alter database open;
```

```
select flashback_on from v$database
```



查看 FLA 指定的目录，一般看看用了多少，以及剩余空间

闪回数据库到指定时间

start mount

RMAN> flashback database to time=to_date('20111110 14:35','yyyymmdd hh24:mi')

闪回到指定 SCN

SQL> flashback database to scn=200127;

-- select CURRENT_SCN from v$database;

根据 LOG 号 闪回

-- select * from v$log;

RMAN> flashback database to sequence=52 thread=1;

TIMESTAMP 闪回

SQL> flashback database to timestamp (sysdate-1/24)；-- 1 小时前

回收站有几个删除的表，有一个带指定的列，闪回来就行

-- query from recyclebin (DROP TYPE)

select OWNER ,ORIGINAL_NAME, OBJECT_NAME from dba_recyclebin;

show recyclebin;

DESC "BIN$tKhVpDPhdSvgQKjACxwGcw==$0"

-- flash back drop table and rename to new tablename

flashback table "BIN$tKhVpDPhdSvgQKjACxwGcw==$0" to before drop rename to emp_bdrop;

-- flash backup transcation

create table t_ft(id number,dt date);

```
begin
for a in 1..1000 loop
insert into t_ft values(a,sysdate);
dbms_lock.sleep(0.01);
commit;
end loop;
end;
/
```

-- 查看每个动作的回退命令

```
SELECT operation, undo_sql, table_name,
a.start_scn,a.start_timestamp,a.commit_scn,a.commit_timestamp
FROM flashback_transaction_query a
where table_name='T_FT';
```

-- 闪回表

```
FLASHBACK TABLE t_ft TO TIMESTAMP to_timestamp('10:42','hh24:mi');
FLASHBACK TABLE t_ft TO SCN 257322;
```

根据历史版本创建一个视图供查询

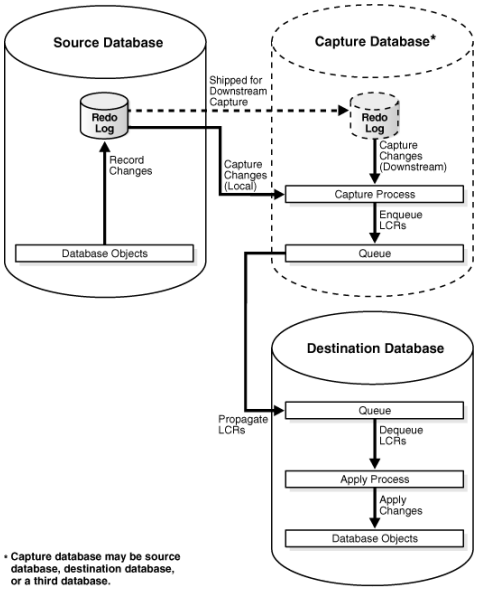
```
create view hist_salary as
select * from hr.t_ft as of scn 257464
```

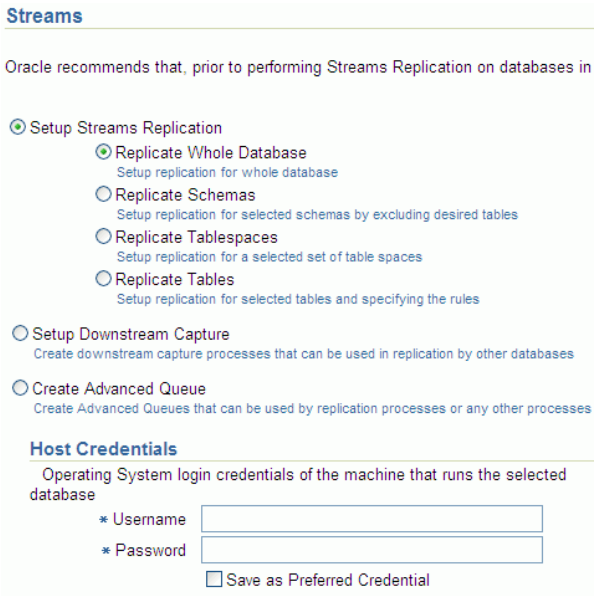
[Y]Use Oracle Streams to capture and propagate changes in a table

Streams Advanced Queuing User's Guide

Streams Replication Administrator's Guide=> 1 Preparing for Oracle Streams Replication

=> 2 Simple Oracle Streams Replication Configuration



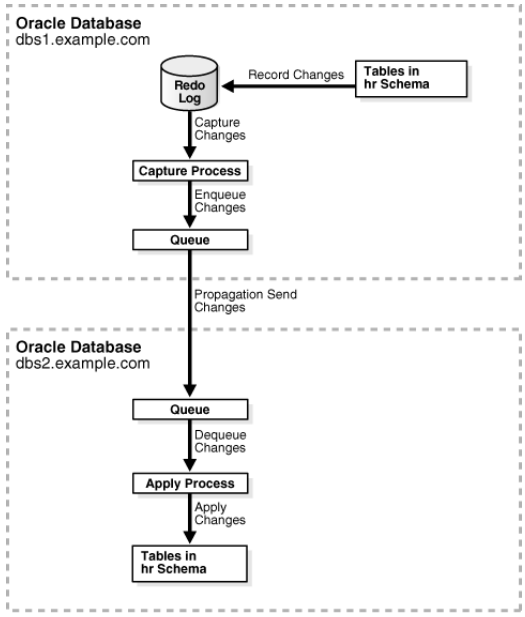


Stream 捕捉 LOG 中的变化进行记录并保存在高级队列中进行发布，目标库去队列中取得相应数据并且到本地库进行重做。可以实现整个数据库以及指定用户、指定表空间、指定表的流复制。

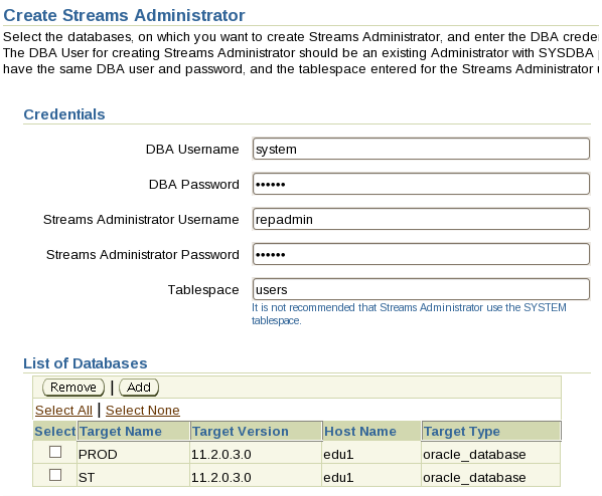
需要配置三种角色 1、数据捕捉者 2、数据分发者 3、订阅者。

考纲说了 Use Oracle Streams to capture and propagate changes in a table 只考虑表 stream 就行了

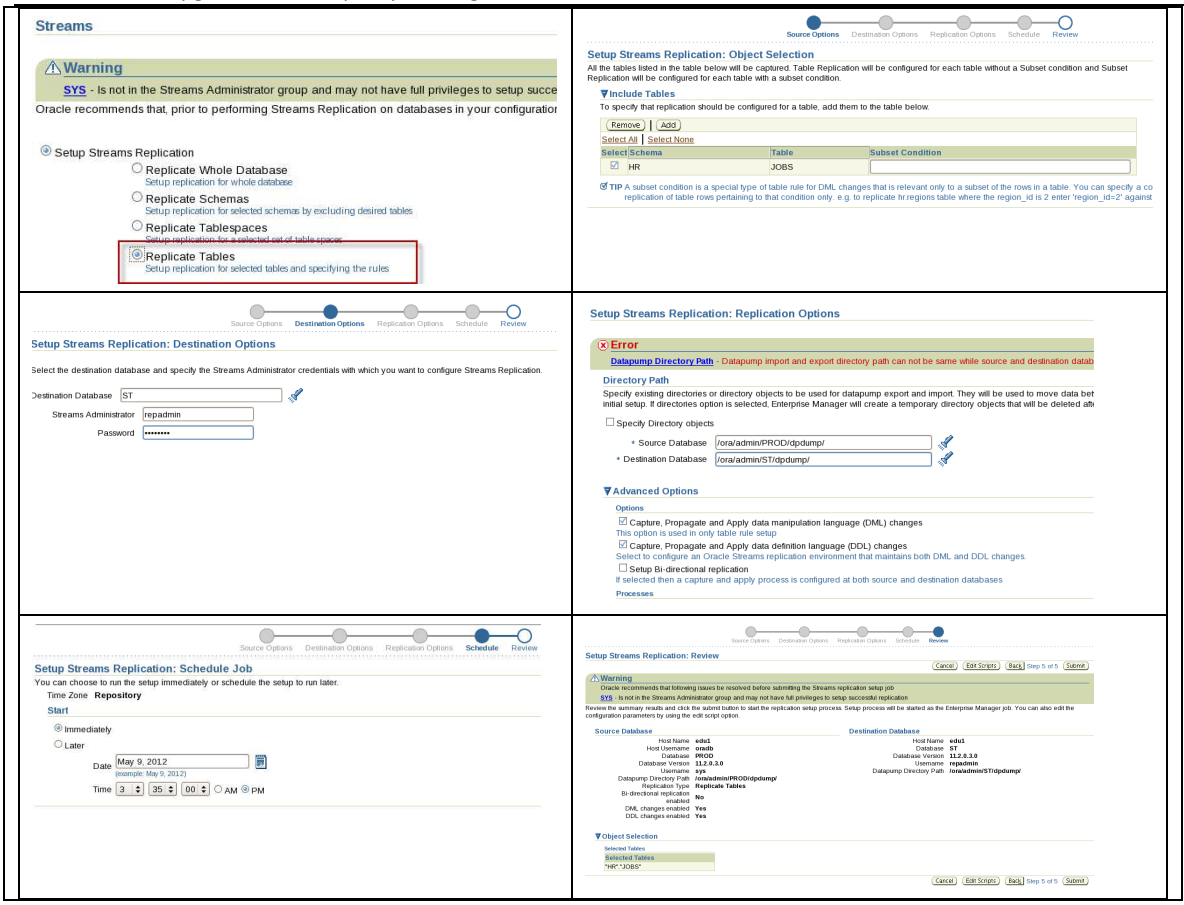
建议 GC，手工是在太麻烦了



1、Create Streams Administrator



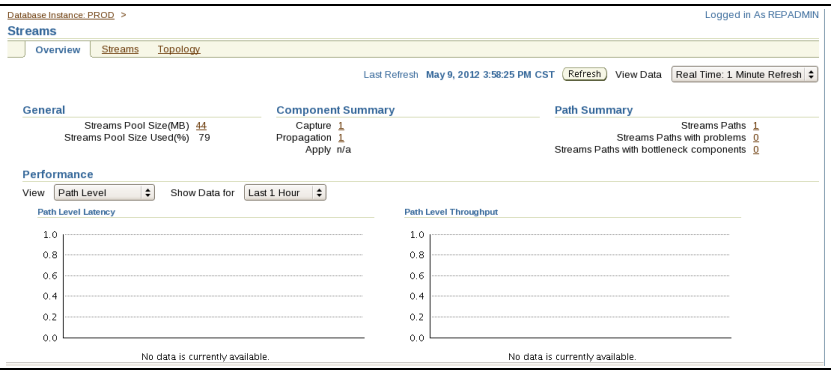
2、Connfig Replicate tables **#注意，这里需要使用 repadmin 登陆才可以要不然做不成功**



-- 配置脚本

config stream.txt

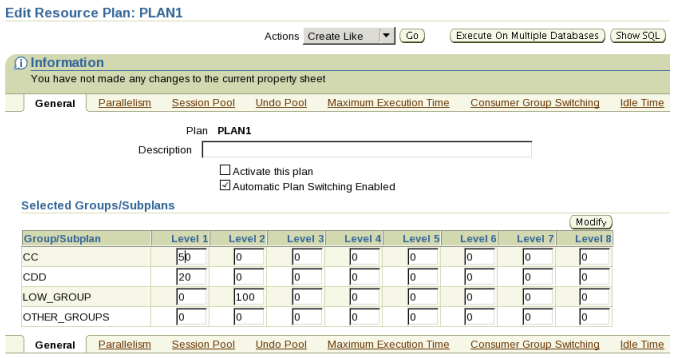
--右侧为配置好的效果



4 Performance Management

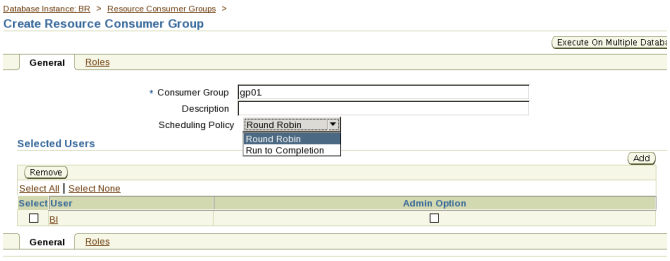
[Y]Set up and configure Resource Manager to control active sessions, number of I/Os, execution time..etc

创建资源计划

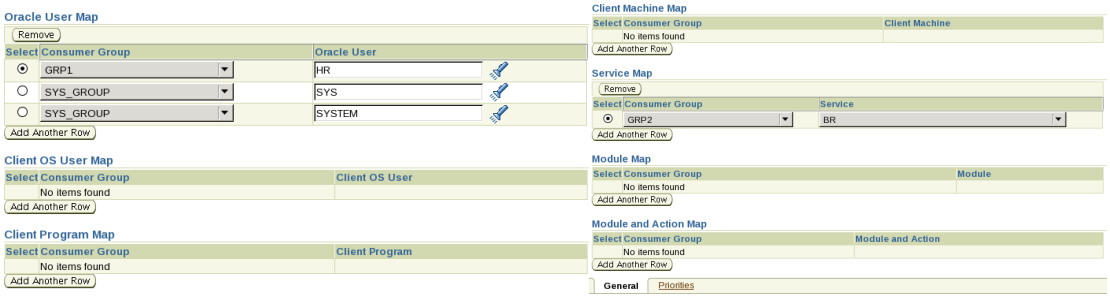


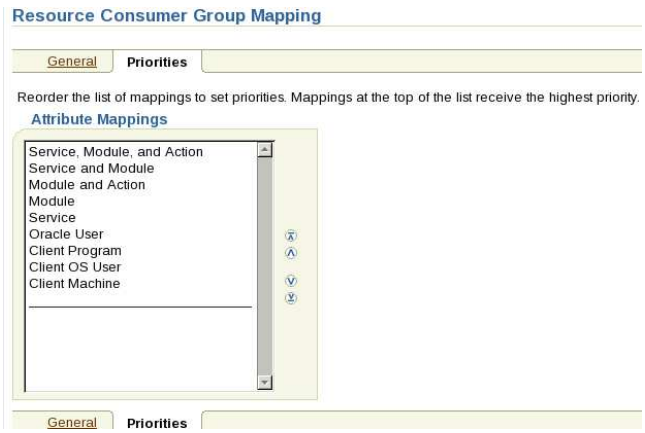
可以分别指定 CPU 资源分配比例、并行度、会话数、undo 容量分配、最大持续执行时间、超时切换、空闲时间

创建资源计划组 policy round robin 随机分配 Run to completion 抢占



分配资源计划给指定的用户或者资源





指定当一个用户同时隶属于不同资源组时先分配顺序

[Y]Use Result Cache

NAME	TYPE	VALUE
client_result_cache_lag	big integer	3000

RESULT_CACHE_MOD

#MANUAL 默认不使用，只有加了/*+result_cache*/才使用

#FORCE 默认使用，只有加了/*+no_result_cache*/才不使用

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_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;

```
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
```

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 |  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
```

[Y]Use multi column statistics

dbms_stat 包的使用，要会分析表 分析索引 分析一个 schema，分析数据库

Dbms_stats	Dba_tables	Dba_tab_statistics	Dba_indexes

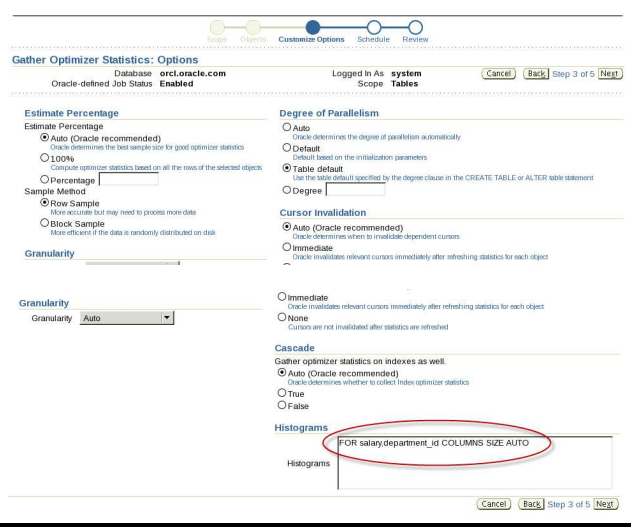
exec dbms_stats.gather_table_stats('HR','EMPLOYEES'); --分析表

exec dbms_stats.gather_index_stats('HR','EMPLOYEES_ID'); --分析索引

exec dbms_stats.gather_schema_stats('HR'); --分析 schema

生成指定列的直方图，适用于数据分布不均匀的情况，**COLUMNS SIZE 数量为桶的数量**

```
begin
dbms_stats.gather_table_stats(
ownname=> 'HR',
tabname=> 'EMPLOYEES' ,
method_opt=> 'FOR salary,department_id
COLUMNS SIZE 255');
end;
/
```



[Y]Gather statistics on a specific table without invalidating cursors

exec dbms_stats.gather_table_stats('HR','EMPLOYEES'); --分析表

exec DBMS_STATS.LOCK_TABLE_STATS ('HR','EMPLOYEES'); 锁定统计数据

[Y]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 ;

[Y]Use SQL Tuning Advisor

对给出的语句进行分析，语句语法级别的建议。 **dbms_sqltune**

[Y]Use SQL Access Advisor

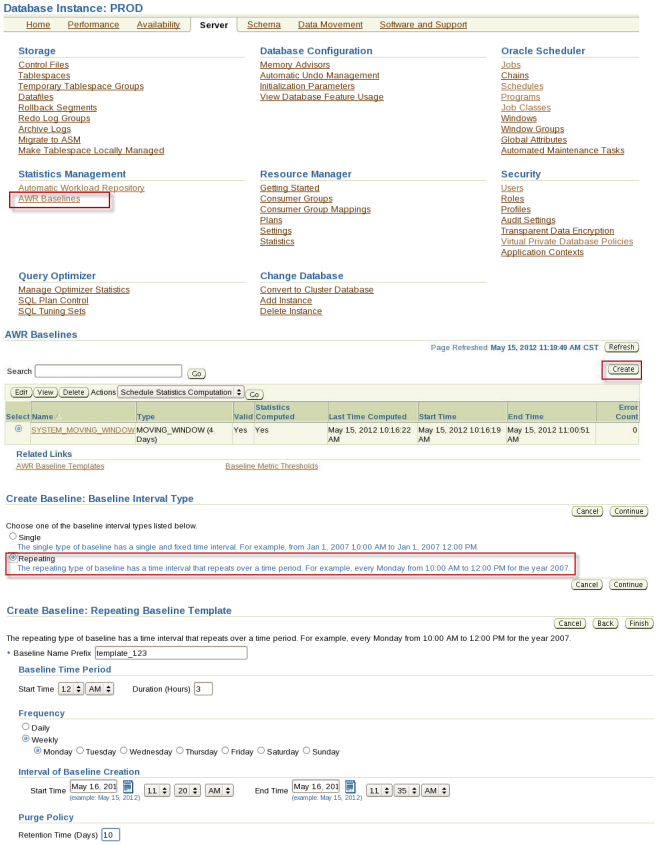
--对语句进行分析，给出建立索引 MVIEW IOT 的建议。 Dbms_advisor

```
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;

[Y]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_TE
MPLATE(start_time IN DATE,
   end_time IN DATE,
   baseline_name IN VARCHAR2,
   template_name IN VARCHAR2,
   expiration IN NUMBER,
   dbid IN NUMBER DEFAULT NULL);
```

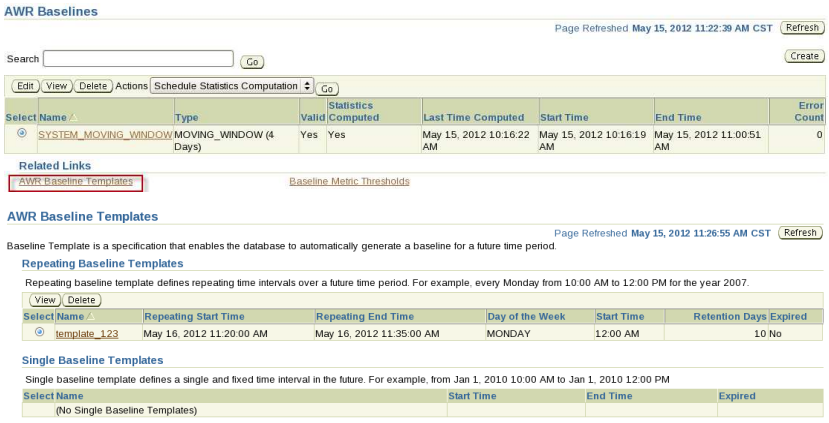
创建自动重复 baseline 模板脚本

```
DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TE
MPLATE(day_of_week IN 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_TEMP
LATE(template_name IN VARCHAR2,
   dbid IN NUMBER DEFAULT NULL);
```

-- 查看一下



[Y]Use SQL Plan Management feature

PL/SQL Packages and Types Reference=>132 DBMS_SPM

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

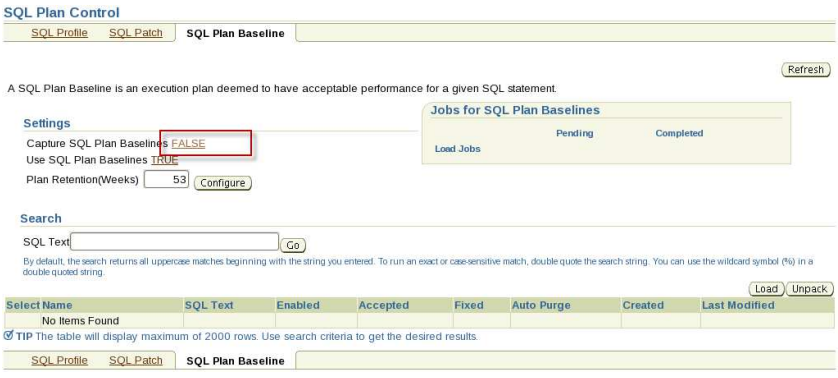
This table list the package subprograms in alphabetical order.

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

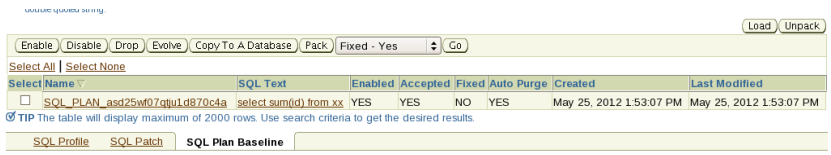
界面如下

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

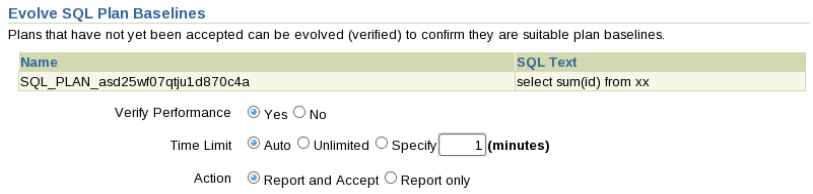
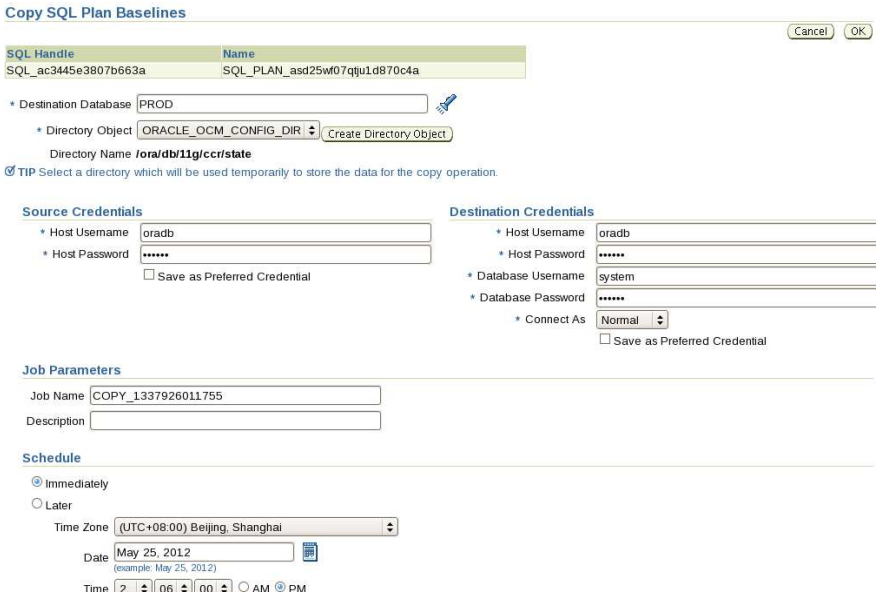


alter system set optimizer_capture_sql_plan_baselines=true scope=both;

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

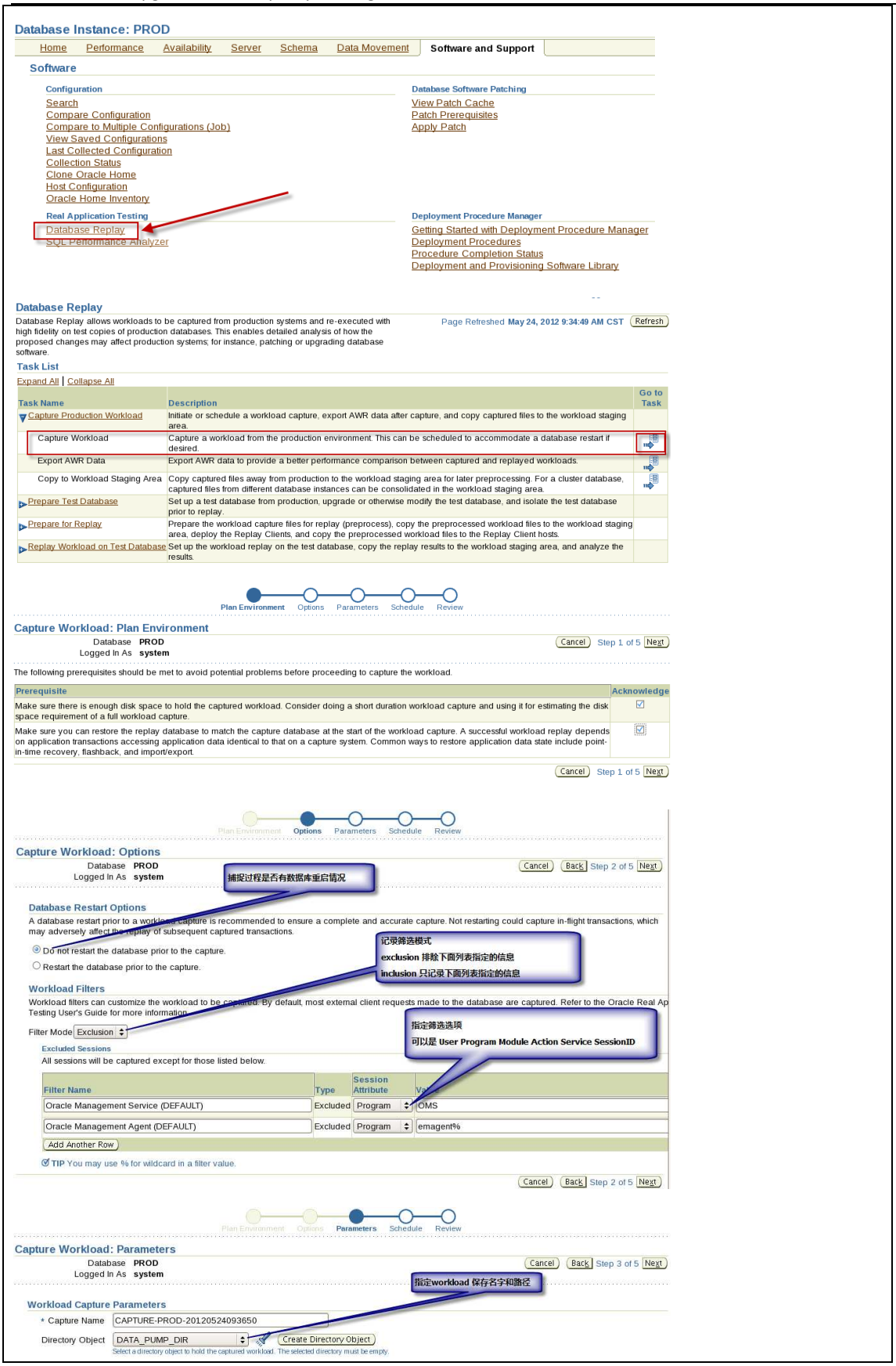
Disable	禁用该执行计划 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
Drop	删除该执行计划 DBMS_SPM.DROP_SQL_PLAN_BASELINE
Evolve	进化！ 是否允许执行计划进化为效率更高的执行计划 DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE 
Copy To A database	把执行计划复制到其他的数据库 
PACK	打包保存起来 DBMS_SPM.PACK_STGTAB_BASELINE
Fixed-Yes	固定 DBMS_SPM.ALTER_SQL_PLAN_BASELINE
Fixed-No	不固定 DBMS_SPM.ALTER_SQL_PLAN_BASELINE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

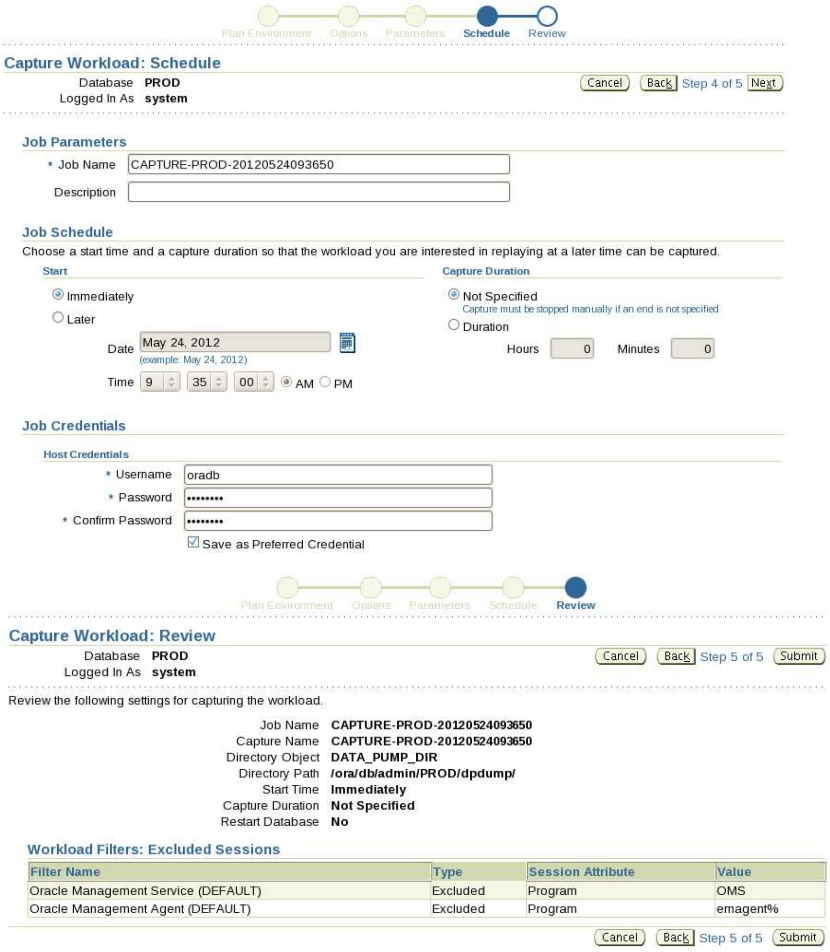
[Y]Replay a captured workload

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
create table xx(id number,text varchar2(20),dd date);
```

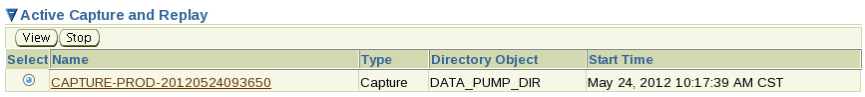

Database Instance: PROD
Home Performance Availability Server Schema Data Movement Software and Support
Software
Configuration
Search
Compare Configuration
Compare to Multiple Configurations (Job)
View Saved Configurations
Last Collected Configuration
Collection Status
Clone Oracle Home
Host Configuration
Oracle Home Inventory
Database Software Patching
View Patch Cache
Patch Prerequisites
Apply Patch
Real Application Testing
Database Replay
SQL Performance Analyzer
Deployment Procedure Manager
Getting Started with Deployment Procedure Manager
Deployment Procedures
Procedure Completion Status
Deployment and Provisioning Software Library
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 Application 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
Value
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.



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(1,999999999999999999)),20,0),sysdate);
```

```
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

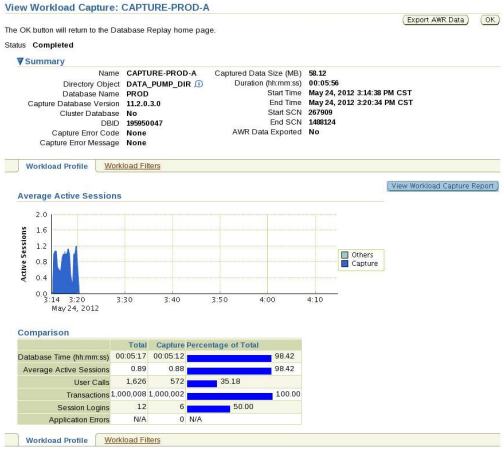
Active Capture and Replay

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

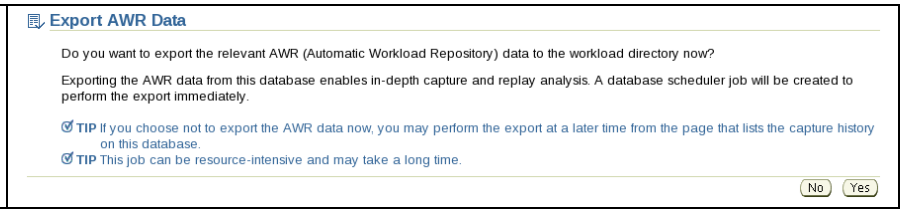


#保存的文件

```
[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.

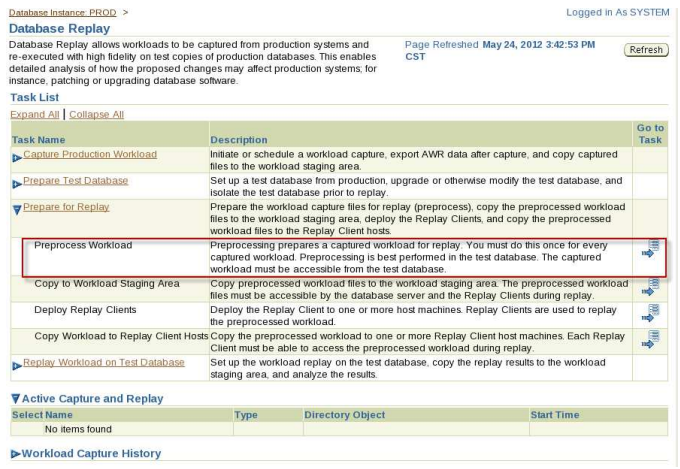


>Replay captured workload

#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



选择目录准备方式，我们用一个手工复制过来的目录

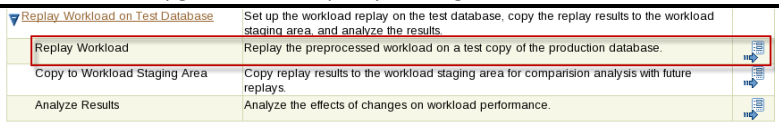
指定目录位置后会自动把选定目录中的捕捉信息显示出来

定义个任务名，根据目录中的信息生成 replay 基础信息

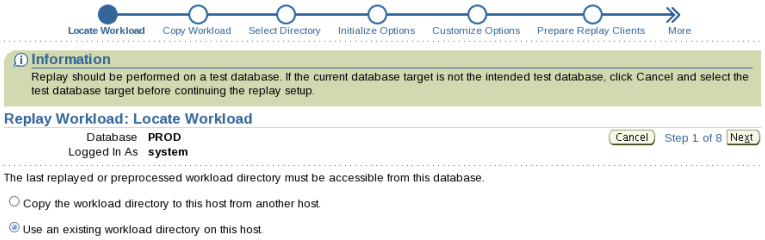
做成功后会在目录中生成 /ora/db/admin/PROD/repdir/pp11.2.0.3.0 目录，并且在下面会有很多 replay 包的信息

```
[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
```

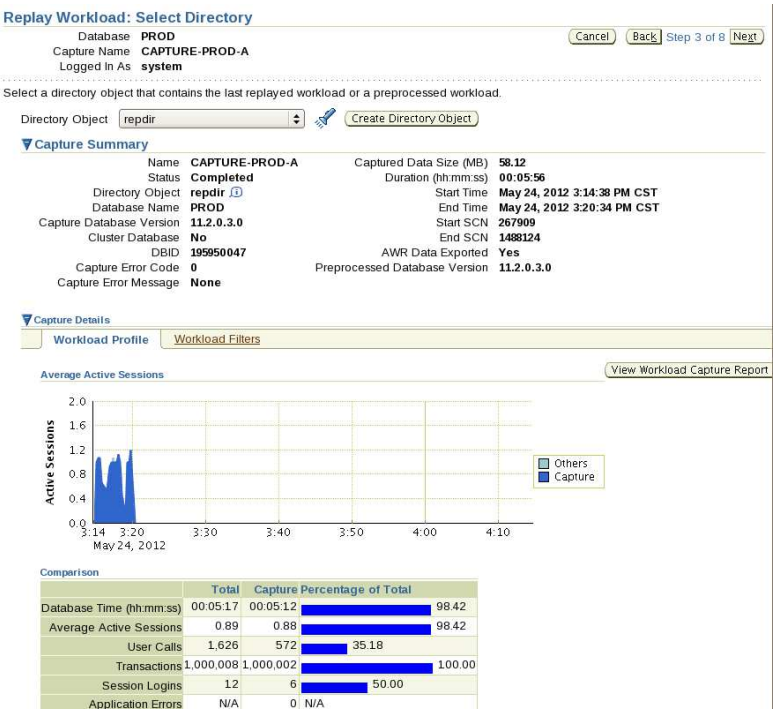
>> Replay Workload　重演



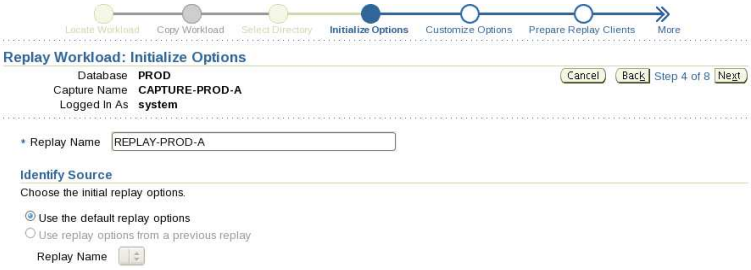
使用已有的目录



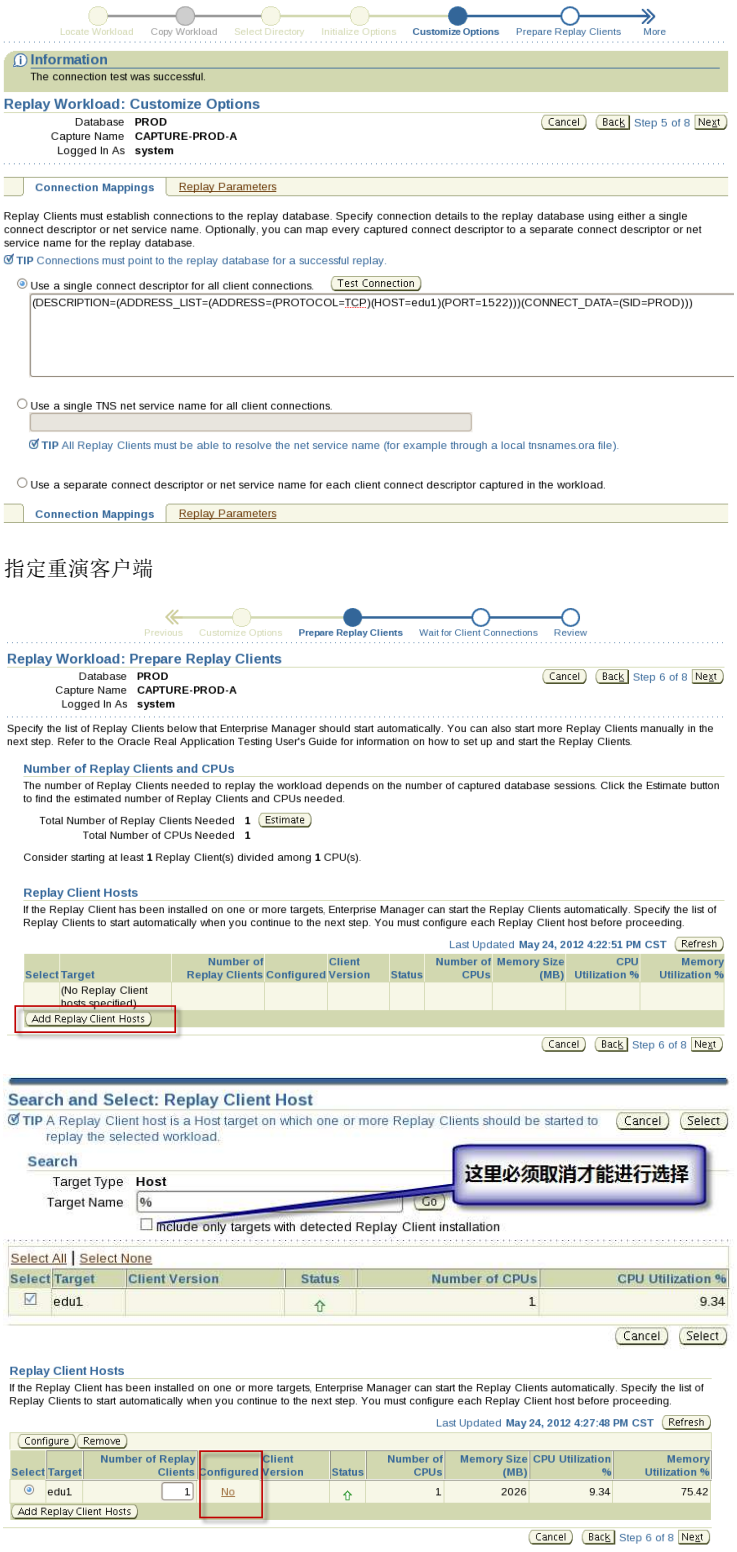
选择目录，会自动带出目录中捕捉的重演脚本信息



惯例，要给个名字

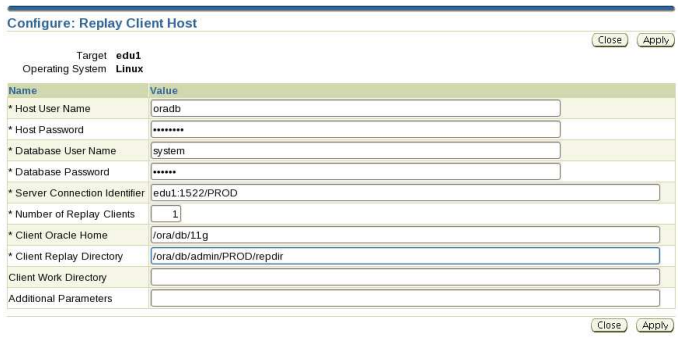


指定重演连接方式

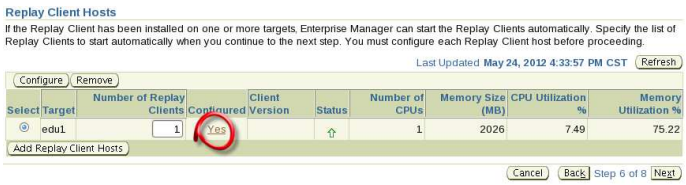


指定重演客户端

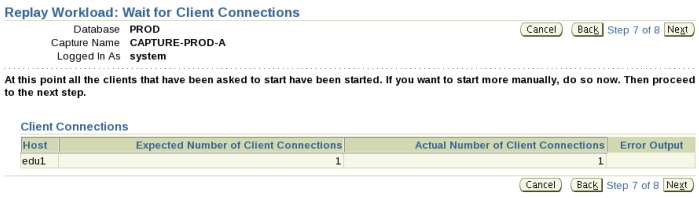
指定好重演 client 之后，配置他



配置好之后 apply 再 close 掉，配置成功的状态

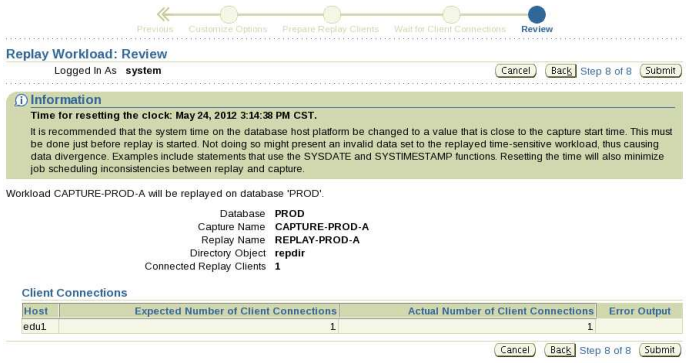


下一步之后会进入 重演等待状态，另外开窗口手工启动重演命令 Number client 会变成 1 然后点 NEXT 开始重演



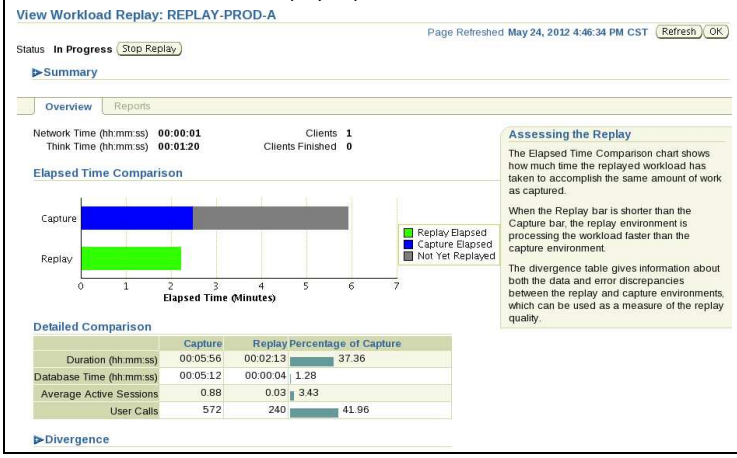

```
wrc system/oracle mode=replay replaydir=/ora/db/admin/PROD/repdir
```

直接提交就可以了



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



DB Replay Report for REPLAY-PROD-A

DB Name	DB Id	Release	RAC	Replay Name	Replay Status
PROD	1959500947	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

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