

# Migration of VISTA From Oracle 10.2.0.3 on SUNOS To Oracle 10.2.0.3 RAC on Linux

## Table of Contents:

Migration of VISTA From Oracle 10.2.0.3 on SUNOS To Oracle 10.2.0.3 RAC on Linux ...	1
Process Overview.....	2
db_link.sql.....	4
cms_file_content_at_source.sql.....	5
cms_file_content_updates.sql.....	6
cms_chunk_up.sql.....	7
cms_file_content_whole.sql.....	8
do_whole_copy.sql.....	9
time_per_1000.sql.....	10
time_per_1000.lst (Sample Output).....	11
blob_sizes.sql.....	12
cms_update.sql.....	13



## Process Overview

Due to the size of the LOB data in the source database we were not able to use Data Pump to successfully "move" the data from the source database to the target. To mitigate this problem we formulated the following plan:

1. About 10-15 days prior to cutover from the source database to the target database we started to move the LOB data using the procedure outlined below. The process took 4 days to move 1.2T (3.5 million rows) of data across the network.
2. Once the initial load was complete, we created all indexes and primary key constraints on the LOB data tables.
3. Between the time the initial LOB copy completed and the cutover date, we moved "new" LOB data from the source to the target on a daily basis.
4. At cutover, we moved the last remaining LOB data from the source to the target and exported all non LOB data from our source database and imported it into the target database.

The initial LOB Data Load From Source to Target

READ.ME

Step	File	Task
1.	db_link.sql	Create the database link between the new production (the target) system and the old production (the source) system. This was created on the target.
2.	cms_file_content_at_source.sql	Creates a synonym for the source CMS_FILE_CONTENT table on the target.
3.	cms_file_content_updates.sql	This is the driver table for the updates. Creates the table CMS_FILE_CONTENT_UPDATES locally. This table stores all records processed, making the script re-enterable. It also allowed me to track the time for The updates so that we could calculate "Percent Complete".
4.	cms_file_content_whole.sql	Creates the stored procedure CMS_FILE_CONTENT_UPDATE.
5.	cms_chunk_up.sql	Creates batches of LOB data. Populates the cms_file_content_updates table. This allowed me to control the size of the commit-batch. It also allowed me to monitor progress and predict completion times.

- |    |                   |   |
|----|-------------------|---|
| 6. | do_whole_copy.sql | Executes the copy of LOB data from source to target.                                      |
| 7. | time_per_1000.sql | Monitored the copy of data from source to target.   |
| 8. | blob_sizes.sql    | Running this while the copy is happening allowed better monitoring of the copy procedure. |

A note on blob\_sizes.sql:

We ran the initial load twice; once as a proof of concept/test and once for the production load. I did NOT empty out the cms\_file\_content\_updates table between the loads since the "blob\_sizes" had already been run. I simply updated the "updated\_at" column to null. This gave me a great way to accurately watch the progress of the load without negatively impacting the process by running blob\_sizes.sql again.

Once the initial load was completed, I ran the following script daily to keep cms\_file\_content up-to-date with our source.

READ.ME

Step	File	Task
-----		
1.	cms_update.sql	Keeps the source and target table synchronized.

## ***db\_link.sql***

```
CREATE DATABASE LINK source
  CONNECT TO source-un identified BY source-pw
  USING 'tnsconnect-for-source';
```

## ***cms\_file\_content\_at\_source.sql***

```
CREATE SYNONYM cms_file_content_at_source  
    FOR cms_file_content@source;
```

## ***cms\_file\_content\_updates.sql***

```
CREATE TABLE cms_file_content_updates (  
    begin_id          NUMBER(20)  
    ,end_id           NUMBER  
    ,updated_at       DATE  
    ,total_blob_size  NUMBER  
);
```

## ***cms\_chunk\_up.sql***

```
DECLARE
    CURSOR cms_file_content_cur IS
        SELECT id
        FROM cms_file_content_at_vistadb
        ORDER BY 1;

        --number of rows to commit at a time
        batch_commit NUMBER := 999;
        --number of rows inserted but not committed
        number_in_batch NUMBER := 0;

        number_of_recs NUMBER := 0;

        begin_number NUMBER :=0;
        end_number NUMBER :=0;

BEGIN
    FOR cms_file_content_rec IN cms_file_content_cur LOOP
        number_in_batch := number_in_batch + 1;
        IF number_in_batch > batch_commit THEN
            INSERT INTO cms_file_content_updates
                VALUES(begin_number, cms_file_content_rec.id, null, null);
            number_in_batch := 0;
        ELSIF number_in_batch = 1 THEN
            begin_number := cms_file_content_rec.id;
        END IF;

    END LOOP;

END;
/
```

## ***cms\_file\_content\_whole.sql***

```
CREATE OR REPLACE PROCEDURE cms_file_content_whole IS
  CURSOR batch_cur IS
    SELECT  begin_id
           ,end_id
           ,id
    FROM    cms_file_content_updates
           ,cms_file_content_at_source
    WHERE   id BETWEEN begin_id AND end_id
    AND     updated_at IS NULL
    ORDER  BY begin_id;

  previous_begin_id      NUMBER          := 0;
  previous_end_id        NUMBER          := 0;

  first_rec              BOOLEAN         := TRUE;

BEGIN

  FOR batch_rec IN batch_cur LOOP
    IF first_rec THEN
      previous_begin_id := batch_rec.begin_id;
      previous_end_id   := batch_rec.end_id;
      first_rec := FALSE;
    ELSIF batch_rec.begin_id != previous_begin_id THEN
      UPDATE CMS_FILE_CONTENT_UPDATES
         SET updated_at = SYSDATE
         WHERE begin_id = previous_begin_id
         AND   end_id   = previous_end_id;
      COMMIT;
      previous_begin_id := batch_rec.begin_id;
      previous_end_id   := batch_rec.end_id;
    END IF;

    INSERT INTO cms_file_content
      (SELECT * FROM cms_file_content_at_vistadb where id = batch_rec.id);

  END LOOP;

END;
/
```

## ***do\_whole\_copy.sql***

```
PROMPT Setting Serveroutput On - unlimited  
set serveroutput on size unlimited
```

```
PROMPT Executing cms_file_content copy...  
exec cms_file_content_whole
```

```
exit
```

## ***time\_per\_1000.sql***

```
set pagesi 1000
set linesi 110

column begin_id          format 9999999999999999 heading "Beginning|ID"
column mins_per_1000     format 999.99 heading "Minutes to|Transfer|1000 Recs"
column running_avg       format 999.99 heading "Running|Average"
column total_blob_size   format 999,999,999 heading "Total Blob|Size (Mb)"
column begin_time        heading "Begin|Time"
column end_time          heading "End|Time"
column mb_per_min        heading "Mbytes|per Min" format 999,999.99
column avg_mb_per_min    heading "AVG Mbytes|per Min" format 9,999.99

break on report
compute sum of total_blob_size on report

spool time_per_1000

with time_per_1000 as
(
    select  begin_id
           ,to_char(updated_at,'Mon-dd hh:mi:ss') end_time
           ,total_blob_size/1048576 total_blob_size
           ,to_char(lag(updated_at, 1) over (order by begin_id),'hh:mi:ss') begin_time
           ,(updated_at - lag(updated_at, 1) over (order by begin_id))*24*60 mins_per_1000
    from    cms_file_content_updates
    where   updated_at is not null
)
select  begin_id
       ,end_time
       ,begin_time
       ,total_blob_size
       ,mins_per_1000
       ,avg(mins_per_1000) over (order by begin_id) running_avg
       ,total_blob_size/mins_per_1000 mb_per_min
       ,avg(total_blob_size/mins_per_1000) over (order by begin_id) avg_mb_per_min
from time_per_1000
order by begin_id
/

spool off
```

**time\_per\_1000.lst (Sample Output)**

Beginning ID	End Time	Begin Time	Total Blob Size (Mb)	Minutes to Transfer 1000 Recs	Running Average	Mbytes per Min	AVG Mbytes per Min
823964565031	Aug-10 07:00:21	06:59:27	145	.90	1.89	160.85	195.18
824193248051	Aug-10 07:01:32	07:00:21	208	1.18	1.89	175.97	194.54
824671749051	Aug-10 07:02:47	07:01:32	257	1.25	1.89	205.52	194.89
824842262061	Aug-10 07:03:59	07:02:47	252	1.20	1.89	210.13	195.37
825409739011	Aug-10 07:05:25	07:03:59	209	1.43	1.89	145.62	193.86
825858971011	Aug-10 07:06:43	07:05:25	193	1.30	1.89	148.58	192.53

## ***blob\_sizes.sql***

```
DECLARE
  CURSOR UPDATES_DRIVER_CUR IS
    SELECT  begin_id
           ,end_id
           ,updated_at
    FROM    cms_file_content_updates
    WHERE   updated_at is not null
    AND     total_blob_size is null
    ORDER BY begin_id;

BEGIN
  FOR updates_driver_rec in updates_driver_cur LOOP
    UPDATE cms_file_content_updates
      set total_blob_size =
        (SELECT sum(dbms_lob.getlength(content))
         FROM   cms_file_content
         WHERE  id BETWEEN updates_driver_rec.begin_id
                       AND updates_driver_rec.end_id)
      WHERE begin_id = updates_driver_rec.begin_id;
    COMMIT;
  END LOOP;
END;
/

exit
```

## ***cms\_update.sql***

```
INSERT INTO cms_file_content
  (SELECT *
   FROM cms_file_content_at_vistadb t_vistadb
   WHERE not exists
     (SELECT 1
      FROM cms_file_content t_wcdbp
      WHERE t_vistadb.id = t_wcdbp.id)
  )
/

EXIT
```