

OurCustomer Database Replication

Arup Nanda

Version: 1.2 Last Revised: 2/4/02

Contents

Typographical Conventions	1
Purpose	1
Intended Audience	1
Scope	1
Technical Notes	5
To Turn Replication ON or OFF at the Master Site for a Table	5
The Location of the Snapshot Logs	5
Changing the Refresh Interval.....	5
Stopping Replication Temporarily at Snapshot Site	5
Making A Snapshot Full Refresh	6
Small and Medium Tables	6
Large Tables	6
Altering Master Tables Used in Replication	10
Problem	10
Solution Description	10
Step by Step Actions	10

Typographical Conventions

Normal Text	Descriptions
<code>Monotype</code>	Used for exact phrase as in case of typing commands, etc. Enter them as is or better yet, copy and paste them in place of commands.
<i><Angle Bracketed Italicized Monotype></i>	A specific word to be replaced by an actual name in the command. For example, <i><tablename></i> should be replaced by the actual name of the name whenever it appears. Do not add the angle brackets.

Purpose

To describe the components of the replications setup at OurCustomer for the OurCustomer Database and help diagnose and resolve problems.

Intended Audience

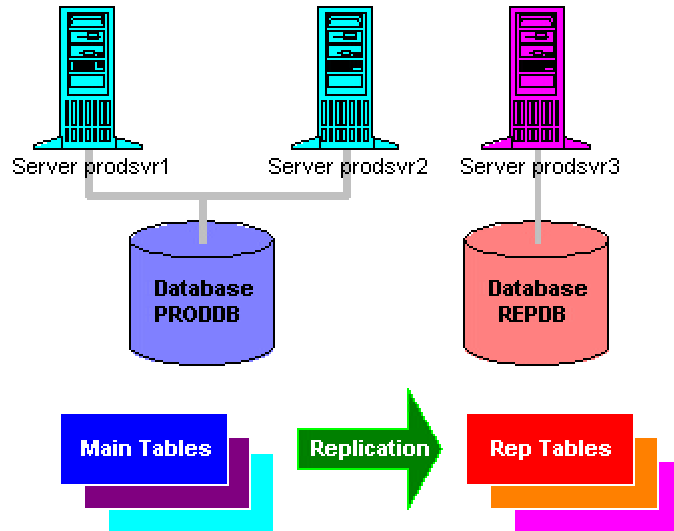
This document is intended for the Database Administrators of the Production Database. A prior knowledge of basic replication setup is assumed on the part of the reader.

Scope

This document describes the specific setup of the replication environment at OurCustomer. It does not describe replication setup in general nor is intended to be an oracle replication textbook.

Setup

The following describes the replication setup at OurCustomer.



The main database, a Two-Node Oracle Parallel Server resides on prodsvr1 and prodsvr2 machines under database name DBPROD. The replication database is on repsvr3 under name DBREP. The replication is based on the Single Master Read Only Snapshot model.

For the purpose of simplicity and ease of administration, two snapshot groups called MG_SMALL_TABLES and MED_TABLES have been formed. They include all the small and medium tables respectively. Each of the other 20 tables, which are big in size and number of rows, has been placed under a refresh group bearing the same name as the table. For example table MYTABLE belongs to the refresh group MYTABLE.

All snapshots are read only type. The large tables have been set to a refresh interval of 5 minutes, the medium tables 5 minutes and the small tables 5 minutes. They can be easily changed however.

The following Table describes the Refresh Groups.

Small Tables Refresh Group MG_SMALL_TABLES. They are controlled by a job which kicks off every 5 minutes. The row counts and sizes are as of 1/17/02.

Owner	Table Name	Size(MB)	Row Count
MSCH	<tablename hidden>	0.06	0
MSCH	<tablename hidden>	5	0
MSCH	<tablename hidden>	40	0
MSCH	<tablename hidden>	10	1

MSCH	<tablename hidden>	30	0
MSCH	<tablename hidden>	30	0
MSCH	<tablename hidden>	0.06	49
MSCH	<tablename hidden>	0.31	0
MSCH	<tablename hidden>	1	0
MSCH	<tablename hidden>	0.06	1
MSCH	<tablename hidden>	10	0
MSCH	<tablename hidden>	8	0
MSCH	<tablename hidden>	5	0
MSCH	<tablename hidden>	5	0
MSCH	<tablename hidden>	15	0
MSCH	<tablename hidden>	12	0
MSCH	<tablename hidden>	1	73
MSCH	<tablename hidden>	0.06	11
MSCH	<tablename hidden>	0.06	18
MSCH	<tablename hidden>	0.06	16
MSCH	<tablename hidden>	0.06	6
MSCH	<tablename hidden>	10	0
MSCH	<tablename hidden>	4	0
MSCH	<tablename hidden>	10	0
MSCH	<tablename hidden>	4	0
MSCH	<tablename hidden>	4	0
MSCH	<tablename hidden>	2	0
MSCH	<tablename hidden>	6	0
MSCH	<tablename hidden>	0.13	52
MSCH	<tablename hidden>	0.25	0
MSCH	<tablename hidden>	2	105
MSCH	<tablename hidden>	1	7
MSCH	<tablename hidden>	2	10
MSCH	<tablename hidden>	40	0
MSCH	<tablename hidden>	4	0
MSCH	<tablename hidden>	2	0
MSCH	<tablename hidden>	2	0

Medium Sized Tables Refresh Group MED_TABLES. They are controlled by a job which kicks off every 5 minutes. The row counts and sizes are as of 1/17/02.

Owner	Table Name	Size(MB)	Row Count
MSCH	<tablename hidden>	1	336
MSCH	<tablename hidden>	1	335
MSCH	<tablename hidden>	2	668
MSCH	<tablename hidden>	25	638
MSCH	<tablename hidden>	15	648
MSCH	<tablename hidden>	4	658

MSCH	<tablename hidden>	4	866
RSCH	<tablename hidden>	0.06	3
RSCH	<tablename hidden>	0.06	1
RSCH	<tablename hidden>	0.06	4
RSCH	<tablename hidden>	0.06	7
RSCH	<tablename hidden>	10	0

Large tables are not grouped into refresh groups. Each large table is a part of a refresh group that bears its name. So each refresh group contains only one table with the same name as its own. The tables and their statistics have been given below. The sizes and row counts as of 1/17/02. Please note that the job numbers can be different if a full refresh has been made. So please update the numbers in this doc.

Table/Group Name	Job#	Size(MB)	Row Count
<tablename hidden>	11	59.00	997,717
<tablename hidden>	12	28.00	997,717
<tablename hidden>	15	50.00	0
<tablename hidden>	22	112.00	2,363,903
<tablename hidden>	16	50.00	0
<tablename hidden>	13	80.00	997,718
<tablename hidden>	21	144.00	1,719,138
<tablename hidden>	40	368.00	10,184,177
<tablename hidden>	17	105.00	2,086
<tablename hidden>	38	664.00	20,959,246
<tablename hidden>	52	12,097.97	20,959,246
<tablename hidden>	50	6,338.00	49,643,583
<tablename hidden>	43	1,793.00	19,903,632
<tablename hidden>	44	3,329.00	38,106,982
<tablename hidden>	42	1,408.00	20,959,311
<tablename hidden>	47	4,929.00	56,928,877
<tablename hidden>	19	10.00	26,119
<tablename hidden>	20	136.00	1,462,748
<tablename hidden>	18	8.00	3,430
<tablename hidden>	49	8,897.97	49,643,799

Technical Notes

To Turn Replication ON or OFF at the Master Site for a Table

Login to prodsrv1 in as the schema owner (MSCH or RSCH) or a dba account like sys and issue the statement to turn off replication.

```
execute dbms_snapshot.begin_table_reorganization  
( '<owner>', '<tablename>' )
```

This destroys the replication triggers on that table and will stop writing to the snapshot log of that table. To turn it on, use the procedure, `dbms_snapshot.end_table_reorganization` with the same parameters. No action is needed at the client site.

To turn replication on for all the tables in the SMALL_TABLES group, use the script

`start_small_tab_repl.sql` in `repl` directory.

To stop replication, use the script

`stop_small_tab_repl.sql`. The scripts for the MED_TABLES group are

`start_med_tab_repl.sql` and

`stop_med_tab_repl.sql`, respectively.

The Location of the Snapshot Logs

The snapshot log of a table is a combination of two tables named, `MLOG$_<tablename>` and

`RUPD$_<tablename>` owned by the same owner which owns the table. In order to know how many records are in the log to be applied, you can issue a `select count(*)` from these tables.

Caution: When replication is turned off via the procedure described above, the changes to the master table(s) are not recorded to the snapshot log(s). So when replication is turned on again, the changes that occurred between the times it was turned off and on later *will not be transmitted* to the snapshot site, as they will not be present in the snapshot log(s). Therefore, when you turn off replication, *always do a full refresh* after turning it on.

Changing the Refresh Interval

To change the interval, login to oracle on repsvr3 as snapadmin/snapadmin and issue the statement

```
execute dbms_refresh.change  
( '<refresh_group_name>', interval=>'sysdate+<minutes>/(24*60)'
```

where `<refresh_group_name>` is the name of the refresh group in uppercase and `<minutes>` is the refresh interval in minutes .

Stopping Replication Temporarily at Snapshot Site

You may want to stop replication temporarily in order to do some administrative tasks on a table at the snapshot site. For instance you want to run an alter table validate structure, or move the table to a different tablespace, etc. To do that, on repsvr3, login to oracle as `snapadmin` and issue

```
select job from dba_refresh where rname = '<tablename>' where <tablename>  
is in uppercase.
```

```
execute dbms_job.broken(<jobno>,TRUE) where <jobno> is as obtained in the previous  
step.
```

Commit; This is important, don't forget to issue a commit.

After you are done, restart the replication by issuing

```
execute dbms_job.run(<jobno>)
```

Replication Problems and Resolutions

Making A Snapshot Full Refresh

Small and Medium Tables

The full snapshot refresh can be a simple task or a complicated one depending upon the size of the snapshot. We have tables (and their snapshots) in three different sizes – small, medium and big as described in the main replication setup chapter. For the small and medium ones, the procedure is fairly simple. Login to repsvr3 as MSCH or RSCH as appropriate and execute from sqlplus (make sure the replication is turned on for that table before executing the following)

```
execute dms_snapshot.REFRESH('<tablename>', 'C')
```



to refresh a single snapshot. To refresh multiple ones, use a comma separated list in the placeholder *<tablename>*. To refresh all the snapshots of the group MG_SMALL_TABLES, use the script *refresh_small_tabs.sql* in *repl* directory of *cosprd3*. The same for group MED_TABLES is *refresh_med_tabs.sql*. To turn replication on for the small tables, execute on *prodsvr1*, as user *sys*, execute the script *start_small_tab_repl* in *repl* directory. The script for medium tables is *start_med_tab_repl*.

Large Tables


The large tables cannot be refreshed by the above method. Please note that the above method can be used for large tables if all roll back segments are very large (approximately 6 GB in size for the largest tables) and large temporary tablespace are available on *prodsvr1* and *repsvr3*. Since this process is infeasible, an alternative approach has been designed. Find a quiet time to do this when less data updates may be happening.

In summary the steps are

1. Halt the refresh job.
2. Drop the snapshot and all associated objects at replication database.
3. Halt the replication activity on the master table.
4. Clean the snapshot log on the master table.
5. Start the replication activity on the master table.
6. Built an empty table at replication database same structure as the snapshot.
7. Export the master table and import into replication database.
8. Store the contents of the snapshot log at a separate place.
9. Build the snapshot and refresh groups at replication site.
10. Insert the saved rows of the snapshot log into it.
11. Run the job to sync up the refresh.

Here are the steps in more detail. The icons  and  indicate where the activity is to be done - *prodsvr1* and *repsvr3*, respectively

On *prodsvr1* (Source)

1.  Logon to oracle using sqlplus as MSCH and issue

```
execute dbms_snapshot.begin_table_reorganization ('MSCH',  
'<tablename>').
```

This will stop the replication triggers to write to the snapshot logs.

2. ✎ Issue `execute dbms_snapshot.purge_log ('MSCH.<tablename>', 999, 'DELETE');`

This will delete all the records from the snapshot log of that table.

3. ✎ Issue `truncate table MSCH.mlog$_<tablename>.`

This will truncate the snapshot log table to reset the high watermark that might have been reached when the snapshot log has grown much bigger than needed.

On repsvr3 (Destination)

4. ✎ Logon to oracle using sqlplus as `snapadmin/snapadmin.`

5. ✎ Find out the job number of the job that fires the refresh for that table. Issue

```
select job from dba_refresh where rname = '<tablename>;
```

6. ✎ Stop that job by issuing

```
execute dbms_job.broken(<jobno>, TRUE)
```

where `<jobno>` is the job number as obtained in the previous step.

7. ✎ Then issue `Commit;`

It is very important to issue commit here.

8. ✎ See if the job is already running. Issue

```
select sid from dba_jobs_running where job = <jobno>;
```

9. ✎ If you see a job running, then you need to kill it. Issue

```
select serial#, status from v$session where sid = <sid>;
```

where `<sid>` is the value obtained in the previous step.

10. ✎ Kill the session by issuing.

```
alter system kill session '<sid>, <serial#>;
```

11. ✎ Repeat all the previous steps to make sure that the job is not running. If the status from `v$session` shows `KILLED`, wait till it completely disappears from `v$session`.

12. ✎ Know which tablespace the table/snapshot is housed in. Issue

```
select tablespace_name from dba_segments where segment_name =  
'<tablename>;
```

13. ✎ Connect to oracle as `snapadmin/snapadmin`

14. ✎ Drop the refresh group by issuing

```
execute dbms_refresh.destroy('<tablename>')
```

15. ✎ Drop the snapshot group by issuing

```
execute dbms_repcat.drop_snapshot_repgroup ('<tablename>', true)
```

16. ✎ Finally drop the snapshot. It might have been dropped in the last stage, but issue this just to be sure. If it was dropped earlier, then the command will result in error. Ignore it. Connect as MSCH and issue.

```
drop snapshot <tablename>;
```

17. ✎ Next drop the underlying table. **Warning : Make sure you are on repsvr3 box.**

```
drop table <tablename>;
```

18. ✎ Now it's time to rebuild that snapshot. First we need to rebuild the table. Connecting as MSCH, issue the statement

```
create table <tablename>  
tablespace <tablespace_name>  
nologging  
as  
select *  
from MSCH.<tablename>@DBPROD  
where 1 = 2;
```

where <tablespace_name> is the name of the tablespace where it would be placed. You would already know that from earlier steps.

Data Transfer Part (Both prodsvr1 and repsvr3)

19. ✎ Logon to the unix box prodsvr1 as oracle.
20. ✎ Issue the following to create a unix pipe to transfer the data across. Before that remove it if it's present.

```
rm /tmp/exp_pipe  
mknod /tmp/exp_pipe p (note the p at the end)
```

21. ✎ Logon to oracle on prodsvr1 as MSCH and turn on replication for that table. Issue

```
execute dbms_snapshot.end_table_reorganization  
( 'MSCH', '<tablename>' ).
```

This will start the replication triggers to write to the snapshot logs.

22. ✎ Logon to unix box repsvr3 as oracle.
23. ✎ Issue to receive contents from the export pipe.

```
rm /tmp/imp_pipe  
mknod /tmp/imp_pipe p (note the p at the end)
```

24. ✎ On prodsvr1, issue `cd repl`.
25. ✎ There is a file `exp_<tablename>.par` which will be used for export parameter definition. To make sure that all the parameters are right, open it and verify. Important parameter is the first line `file=/tmp/exp_pipe`. That should be present.
26. ✎ Run the export by issuing

```
exp MSCH/iclaim parfile=exp_<tablename>.par &
```

Note the ampersand at the end. This will run it in the background.

27. ✎ Immediately after that issue

```
cat /tmp/exp_pipe | rsh repsvr3 dd of=/tmp/imp_pipe &
```


This will push the contents of the pipe to the destination box.

28. ✂ After 1 minute, not immediately, go to repsvr3 box and issue
- ```
cd repl
imp MSCH/iclaim parfile=imp_<tablename>.par &
```

29. ✂ Make sure the import is going smoothly. Logon to oracle as MSCH and issue

```
select count(*) from <tablename>;
```

You should see some non-zero number there.

30. ✂ I have also put together a utility script how much of the import is done. It's called `insrate.sql`. Edit the script to change the table name to `<tablename>` and the number of rows. You can run the script periodically to see how much is done, the rate of inserts, how much is left and the approximate finish time.

31. Depending upon the size of the table, the export import for a table may take up to three hours. The next step is primary key index building. All the following steps are on repsvr3, unless otherwise noted.

32. ✂ Logon to oracle as user MSCH.

33. ✂ Create the index to enforce the primary key constraint. In the directory `repl`, you will find the primary key index creation scripts named in the format `pk_<tablename>.sql`. Login as MSCH and run the appropriate index creation script.

34. ✂ The next step is creation of snapshot from the table just created. First we will have to store the contents of the snapshot log created so far because the registration of the snapshot at the replication site will destroy it. After the snapshot is created, we need to restore the contents back to the snapshot log. I have written a batch sql script to achieve all these. Logging in to oracle on repsvr3 as MSCH, run the following script

```
@cr_repl_prebuilt <tablename>
```

where `<tablename>` is in uppercase. This will create all necessary replication objects and will enable them for replication. It produces a spool file `cr_repl_prebuilt.log`. Check it for errors.

35. ✂ Get the job number the snapshot group is assigned to. Issue the following.

```
select job from dba_refresh where rname = '<tablename>';
```

36. ✂ Login to oracle as `snapadmin/snapadmin` and issue

```
execute dbms_job.run(<jobno>)
```

37. ✂ On prodsvr1 logon to oracle as MSCH and issue

```
select count(*) from mlog$_<tablename>;
```

This should show a very small number. Wait for five minutes and reissue the query. The number should change.

38. ✂ Finally, create the indexes on repsvr3. There are no separate scripts for the indexes. So run the script, `cr_MSCH_indexes.sql` located in the `repl` directory. This will result in some failures, as the indexes will exist before. Just ignore them.

Now replication is enabled on that table.

## Altering Master Tables Used in Replication

---

### **Problem**

When the master tables are altered in some way that affects the snapshot definition, for instance as in case of adding a column to the master table demands that the same column be added to the snapshot for the replication, Oracle replication poses a problem – it does not allow changing the snapshot’s query. The snapshot must be dropped and recreated with the new column in the query. However, large snapshots take an enormous amount of time to be recreated and require considerable rollback and temporary space, making the process infeasible. Here is an alternative approach, undocumented in Oracle, to make it work.

### **Solution Description**

Snapshots can be made in two ways

- ~~✍~~ Directly from the master table, where the snapshot is created as `select * from the master table` suffixed by the db link.
- ~~✍~~ A table can be first created in the replication environment, and the snapshot can be created on the table with the option “prebuilt table”. In case of large tables, a table can be pre-built using `exp/imp`, `create table as select` or the `copy` command in `sqlplus`.

When a snapshot is dropped, the storage is dropped as well; however, when the snapshot created using the pre-built table is dropped, the pre-built table is not dropped, just the snapshot is.

Since the snapshot used the pre-built table during creation, it must be using the same table for all the DML activity, since there is no need to create another segment for the snapshot. So when the snapshot is dropped, the table is left *as the same state* at the time the snapshot was dropped. At that time the table can be applied the same DDLs the master table was subjected to and then the snapshot can be recreated on the table using the pre-built option. Assuming no database change happens between the snapshot is dropped and recreated, oracle engine at the replication site is made to assume that the table was indeed as a same state as the last refresh and thus fast refresh can continue as usual.

This solution has been described in detail here.

### **Step by Step Actions**

---

| <b>Step</b> | <b>Master Site (prodsvr1)</b> | <b>Replication Site (repsvr3)</b>                                                                                                                                                                                        |
|-------------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1           |                               | Identify the job that refreshes the snapshot.<br>Logging in as <code>snapadmin/snapadmin</code> in <code>sqlplus</code> ,<br>issue<br><br><pre>Select job from user_refresh where<br/>rname = '&lt;tablename&gt;';</pre> |

---

| Step | Master Site (prodsvr1)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Replication Site (repsvr3)                                                                                                                                                                            |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <p>Shutdown that job by issuing</p> <pre>execute dbms_job.broken (&lt;jobno&gt;, TRUE)</pre> <p>This will hold the job. <b>Important</b> : Issue a <code>commit</code> after the execute command.</p> |
| 3    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <p>Drop the snapshot. Logging in as MSCH, issue the command</p> <pre>Drop snapshot &lt;tablename&gt;;</pre>                                                                                           |
| 4    | <p>Stop replication activity on the group. Logging in as MSCH issue:</p> <pre>execute dbms_repcat.suspend_master_activity ('&lt;tablename&gt;')</pre>                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                       |
| 5    | <p>Logging in as the user MSCH, issue</p> <pre>execute dbms_repcat.alter_master_reobject (sname =&gt; 'MSCH', oname =&gt; '&lt;tablename&gt;', type =&gt; 'TABLE', ddl_text =&gt; 'alter table MSCH.&lt;tablename&gt; ...');</pre> <p>The alter script is put here. <i>Do not use the end ; as in case of a sql script.</i> This command alters the master table, so there is no need to issue another alter from sqlplus. Make sure you prefixed the tablename with MSCH as the schema owner in the <code>ddl_text</code> argument.</p> |                                                                                                                                                                                                       |
| 6    | <p>Regenerate the replication support for the table. Logging in as user repadmin/repadmin, issue</p> <pre>execute dbms_repcat.generate_replication_support (sname=&gt;'MSCH', oname=&gt;'&lt;tablename&gt;', type=&gt;'TABLE', min_communication=&gt;TRUE )</pre>                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                       |
| 7    | <p>Start the replication activity. As repadmin user, issue</p> <pre>execute dbms_repcat.resume_master_activity ('&lt;tablename&gt;')</pre>                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                       |
| 8    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <p>Alter the replication table. Logging in as MSCH</p>                                                                                                                                                |

| Step | Master Site (prodsvr1) | Replication Site (repsvr3)                                                                                                                                                                                                                                                                                                                                                                                           |
|------|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9    |                        | <p>user, issue the alter table statement for that table name.</p> <p>Create the snapshot. Run the script cr_ref.sql. From sqlplus prompt type</p> <pre>@cr_repl_prebuilt &lt;tablename&gt;</pre> <p>where &lt;tablename&gt; is table name in uppercase. It will actually error out later stating that the snapshot existed earlier, but ignore it. Describe the table to make sure the new columns are in there.</p> |
| 10   |                        | <p>Check the snapshot. As snapadmin, issue in sqlplus</p> <pre>execute dbms_refresh.refresh('&lt;tablename&gt;')</pre>                                                                                                                                                                                                                                                                                               |
| 11   |                        | <p>Check the job. As snapadmin in sqlplus issue</p> <pre>execute dbms_job.run(&lt;jobno&gt;)</pre>                                                                                                                                                                                                                                                                                                                   |

At this point the snapshot at replication site behaves like a fast refreshable snapshot. The most important thing to make sure that no data changes occurs between the first and last steps.