Easy Application Failover with DataGuard

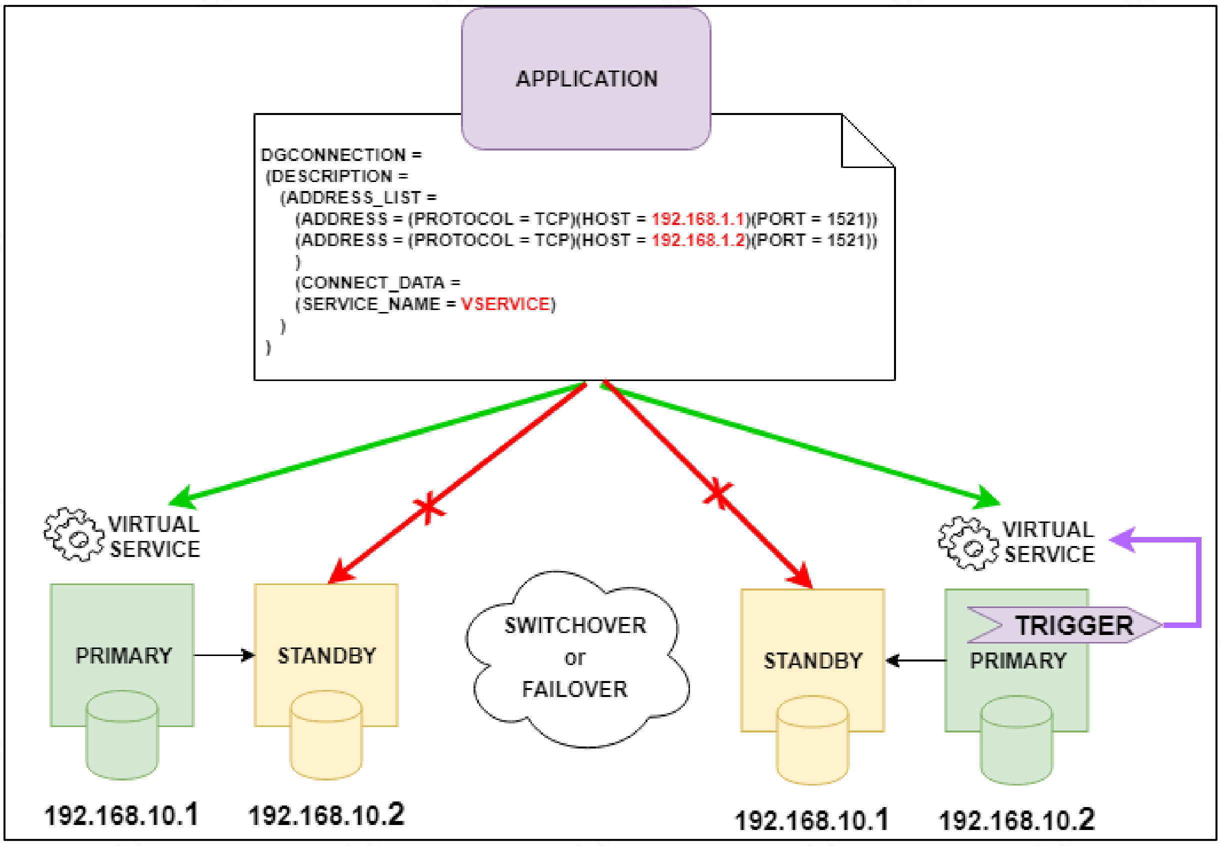
Application failover when database failover or switchover is facilitated by Data Guard. This is not the real seamless failover but works perfect for most of the scenarios...

1. PREFACE

This tutorial covers the situation where an application still has to be able to connect to the primary database after a failover or a switcover is initiated on a dataguard system. This is not the actual seamless application failover. It does not employ any clusterware components like FAN or TAF. That is why I call this "EASY" Application Failover. It may actually have a different name but I didn't come across with an Oracle document about it... so it is "Easy Applcation Failover with Datagaurd" for me.

2. HOW IT WORKS

The below diagram shows the architecture of the setup.



```
DGCONNECTION =
 (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.1)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.2)(PORT = 1521))
     )
     (CONNECT_DATA =
     (SERVICE_NAME = VSERVICE)
   )
 )
```

The logic behind the application continuity in the case of a swithover (or failover) depends on a virtual service that can be started or stopped with a trigger depending on the role of the database server. To be more precise, we should implement a role based service that will only be enabled when the database server is in primary role. In fact, from the version 11.2 on, this role based service is not handled by triggers but the clusterware as follows:

```
srvctl add service -db orcl -service vservice -role primary
```

However, it might be the case that we are working on a single instance database without a clusterware, so the trigger methodology can still be used.

As a result, the service name that the application should be using, would be active only on the primary side at a given time. Even the application may try to go to the standby (if it is the first connection defined), it will see that there is no such service and proceed to the next adress in the list.

3. IMPLEMENTATION

Create a virtual service on the primary database. (Of course, every definition/creation of an object in primary database will be replicated to the standby server too.)

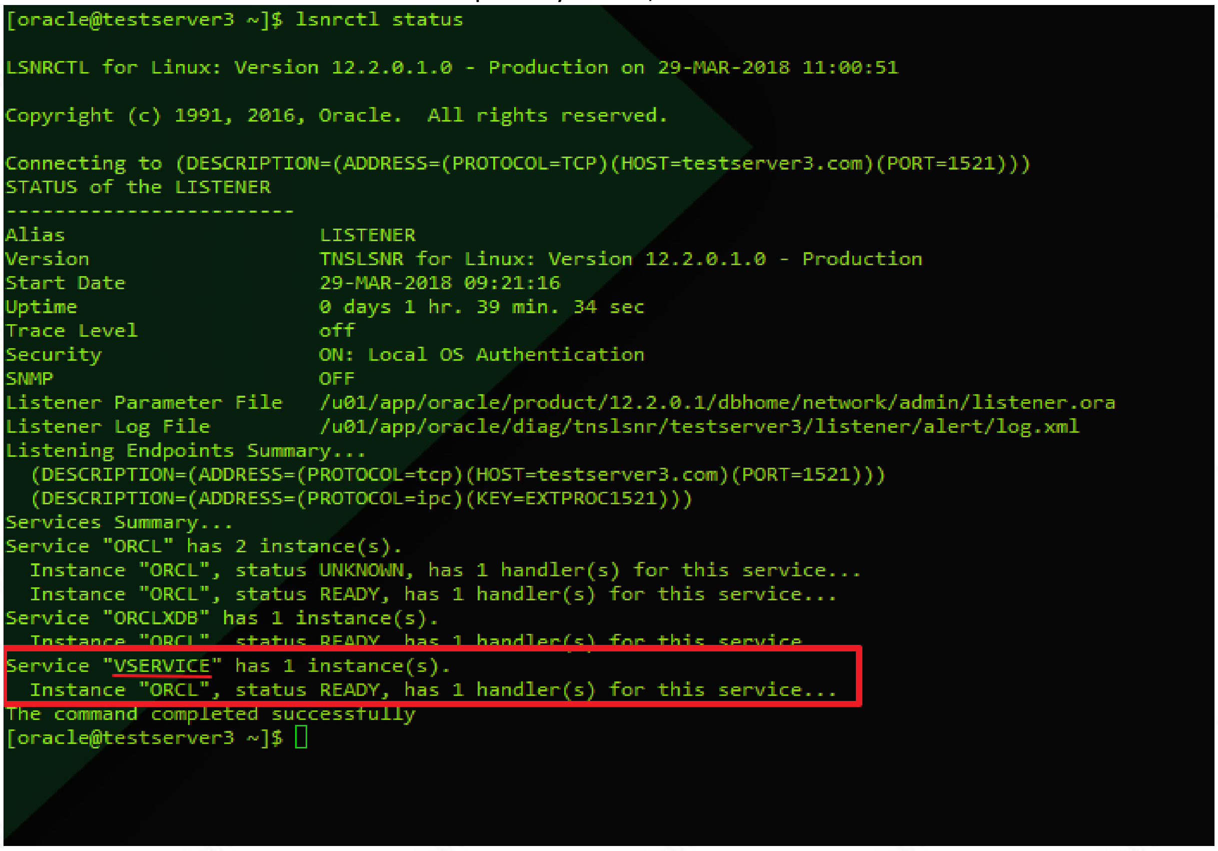
Create a virtual service

```
DECLARE
 PARAM_ARRAY DBMS_SERVICE.SVC_PARAMETER_ARRAY;
BEGIN
   PARAM_ARRAY('FAILOVER_TYPE') := 'SELECT';
   PARAM_ARRAY('REPLAY_INITIATION_TIMEOUT'):=100;
   PARAM_ARRAY('RETENTION_TIMEOUT') :=86400;
   PARAM_ARRAY('FAILOVER_DELAY') :=1;
   PARAM_ARRAY('FAILOVER_RETRIES') :=5;
   PARAM_ARRAY('COMMIT_OUTCOME') :='TRUE';
   PARAM_ARRAY('aq_ha_notifications') :='TRUE';
   DBMS_SERVICE.CREATE_SERVICE('VSERVICE','VSERVICE' , PARAM_ARRAY);
END;
/
```

Start the virtual service

```
BEGIN
    DBMS_SERVICE.START_SERVICE('VSERVICE');
END;
/
```

If we have a look at the services on the primary server, we should see this new virtual service:



```
[oracle@testserver3 ~]$ lsnrctl status

LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 29-MAR-2018 11:00:51

Copyright (c) 1991, 2016, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=testserver3.com)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date                29-MAR-2018 09:21:16
Uptime                    0 days 1 hr. 39 min. 34 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/product/12.2.0.1/dbhome/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/testserver3/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=testserver3.com)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "ORCL" has 2 instance(s).
  Instance "ORCL", status UNKNOWN, has 1 handler(s) for this service...
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCLXDB" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "VSERVICE" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
The command completed successfully
[oracle@testserver3 ~]$
```

Now we have to create the triggers that will trigger the activation of this virtual service in the case of a role transition. There should be 2 triggers one for detecting the change of the database role and the other for detecting the startup of the database. (Meybe merging them into one single trigger is a good idea but this is better for readibility)

Trigger1

```
CREATE TRIGGER START_SERVICE_ONROLECHG AFTER DB_ROLE_CHANGE ON DATABASE
DECLARE
  V_ROLE VARCHAR(30);
BEGIN
  SELECT DATABASE_ROLE INTO V_ROLE FROM V$DATABASE;

  IF V_ROLE = 'PRIMARY' THEN
    DBMS_SERVICE.START_SERVICE('VSERVICE');
  ELSE
    DBMS_SERVICE.STOP_SERVICE('VSERVICE');
  END IF;
END;
/
```

Trigger2

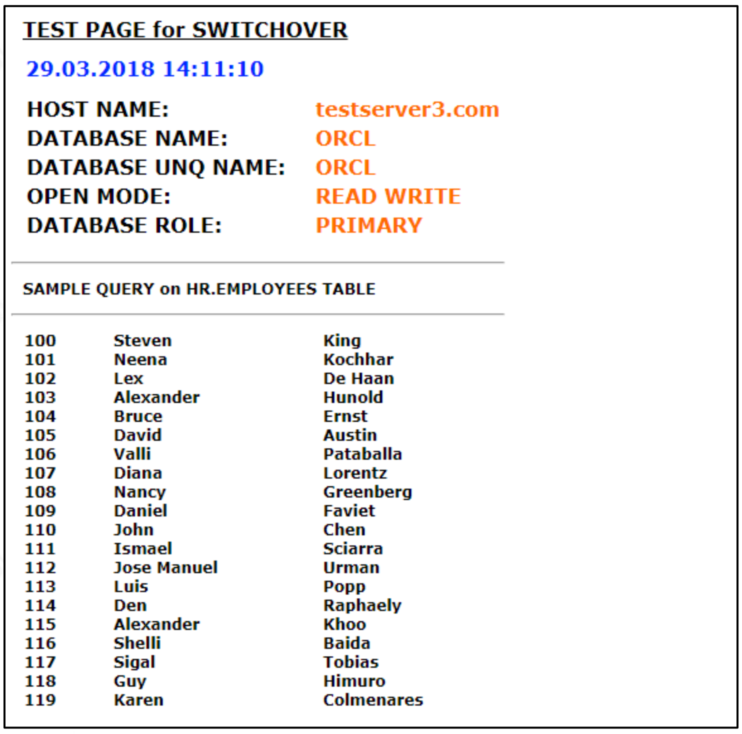
```
CREATE OR REPLACE TRIGGER START_SERVICE_ONSTARTUP AFTER STARTUP ON DATABASE
DECLARE
  V_ROLE VARCHAR(30);
BEGIN
  SELECT DATABASE_ROLE INTO V_ROLE FROM V$DATABASE;

  IF V_ROLE = 'PRIMARY' THEN
    DBMS_SERVICE.START_SERVICE('VSERVICE');
  ELSE
    DBMS_SERVICE.STOP_SERVICE('VSERVICE');
  END IF;
END;
/
```

Now, the application that is to connect to our database should use a connection string like:

```
DGCONNECTION =
 (DESCRIPTION =
    (ADDRESS_LIST =
       (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.136.111)(PORT = 1521))
       (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.136.112)(PORT = 1521))
       )
       (CONNECT_DATA =
       (SERVICE_NAME = VSERVICE)
    )
 )
```

Both database servers are included in the connection but one of them (primary) has the virtual service at a given time.

Actually, I also tested this configuration with a tiny .Net web application:



TEST PAGE for SWITCHOVER

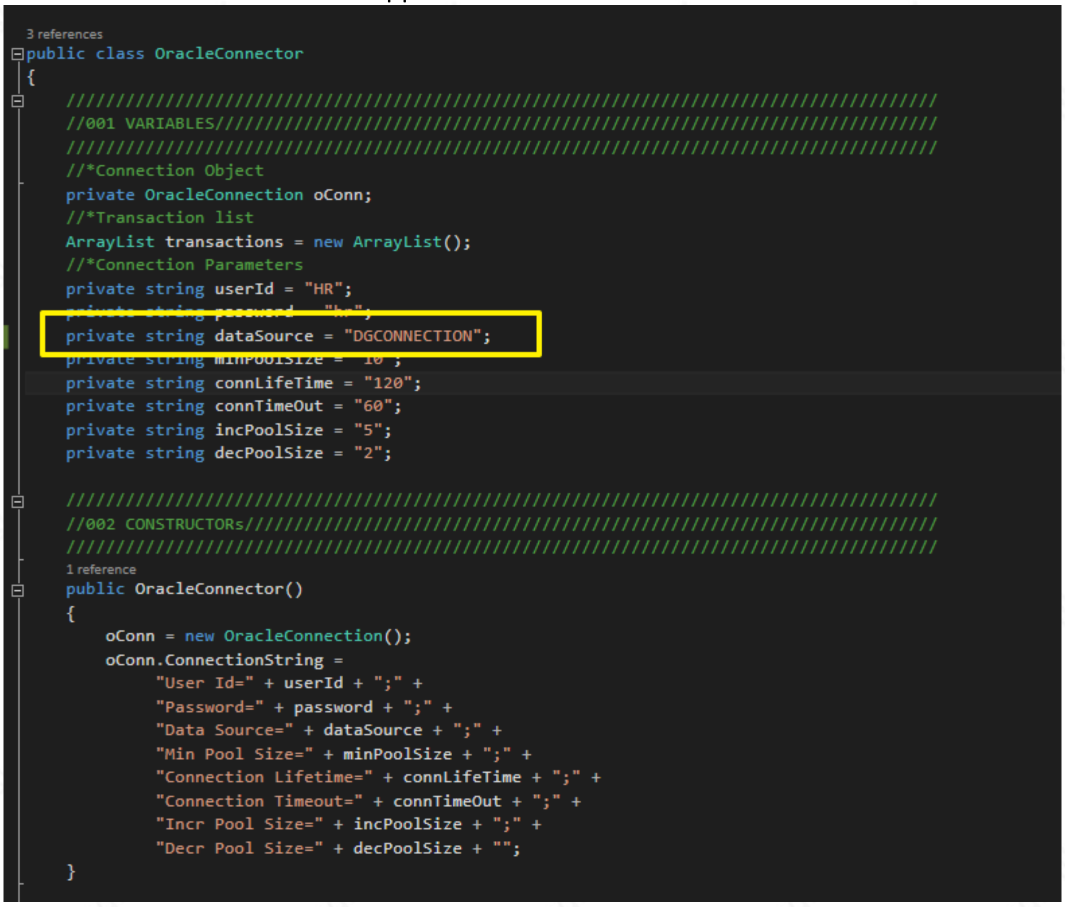
29.03.2018 14:11:10

HOST NAME:	testserver3.com
DATABASE NAME:	ORCL
DATABASE UNQ NAME:	ORCL
OPEN MODE:	READ WRITE
DATABASE ROLE:	PRIMARY

SAMPLE QUERY on HR.EMPLOYEES TABLE

100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Pataballa
107	Diana	Lorentz
108	Nancy	Greenberg
109	Daniel	Faviet
110	John	Chen
111	Ismael	Sciarra
112	Jose Manuel	Urman
113	Luis	Popp
114	Den	Raphaely
115	Alexander	Khoo
116	Shelli	Baida
117	Sigal	Tobias
118	Guy	Himuro
119	Karen	Colmenares

The connections I used for the application is as follows:

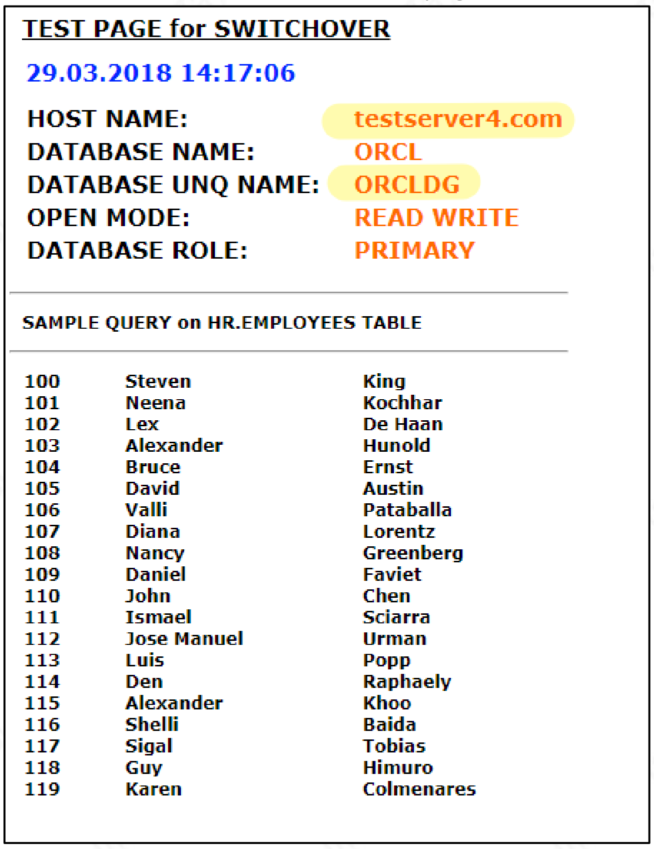


```
3 references
public class OracleConnector
{
    ///////////////////////////////////////////////////////////////////////////////////////////
    //001 VARIABLES///////////////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////////////////////////////
    //*Connection Object
    private OracleConnection oConn;
    //*Transaction list
    ArrayList transactions = new ArrayList();
    //*Connection Parameters
    private string userId = "HR";
    private string password = "hr";
    private string dataSource = "DGCONNECTION";
    private string minPoolSize = "10";
    private string connLifeTime = "120";
    private string connTimeOut = "60";
    private string incPoolSize = "5";
    private string decPoolSize = "2";

    ///////////////////////////////////////////////////////////////////////////////////////////
    //002 CONSTRUCTORs////////////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////////////////////////////
    1 reference
    public OracleConnector()
    {
        oConn = new OracleConnection();
        oConn.ConnectionString =
             "User Id=" + userId + ";" +
             "Password=" + password + ";" +
             "Data Source=" + dataSource + ";" +
             "Min Pool Size=" + minPoolSize + ";" +
             "Connection Lifetime=" + connLifeTime + ";" +
             "Connection Timeout=" + connTimeOut + ";" +
             "Incr Pool Size=" + incPoolSize + ";" +
             "Decr Pool Size=" + decPoolSize + "";
    }
```

And the TNS entry is as follows:

```
DGCONNECTION =
 (DESCRIPTION =
    (ADDRESS_LIST =
       (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.136.111)(PORT = 1521))
       (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.136.112)(PORT = 1521))
       )
       (CONNECT_DATA =
       (SERVICE_NAME = VSERVICE)
    )
 )
```

When I switchover the servers, the page shows (naturally):



TEST PAGE for SWITCHOVER

29.03.2018 14:17:06

HOST NAME:	testserver4.com
DATABASE NAME:	ORCL
DATABASE UNQ NAME:	ORCLDG
OPEN MODE:	READ WRITE
DATABASE ROLE:	PRIMARY

SAMPLE QUERY on HR.EMPLOYEES TABLE

100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Pataballa
107	Diana	Lorentz
108	Nancy	Greenberg
109	Daniel	Faviet
110	John	Chen
111	Ismael	Sciarra
112	Jose Manuel	Urman
113	Luis	Popp
114	Den	Raphaely
115	Alexander	Khoo
116	Shelli	Baida
117	Sigal	Tobias
118	Guy	Himuro
119	Karen	Colmenares

And the refreshes on the page has no delays...

Long story short, if you are planning to switchover on a planned basis, this mechanism can be used. But **it is not for a 7/24 running critical production system**. It does not provide a real Application Failover!