Implementando Fast Connection Failover sobre Aplicaciones .NET

Por Francisco Riccio

Introducción

Fast Connection Failover (FCF) es un mecanismo que ofrece un failover de conexión de base de datos a nuestras aplicaciones, el cual se ejecutará automáticamente cuando ocurra una indisponibilidad en la instancia conectada; siendo una acción transparente para los usuarios finales.

Las librerías requeridas para el uso de FCF están disponibles desde la versión Oracle Client 10gR1, pero en el caso de ODP.NET están disponibles desde la versión 10gR2. Es importante resaltar que a partir de la versión Oracle Client 10gR2 se incluyó el feature de notificación de balanceo de carga sobre una base de datos Oracle RAC a través de notificaciones FAN.

FCF tiene las siguientes características:

- Ofrece rápida detección de la indisponibilidad de una instancia e inicia la reconexión en otra disponible. La detección de dicha indisponibilidad no se da por eventos de TCP Timeouts sino mediante eventos recibidos por el Oracle RAC, siendo más rápido y eficiente.
- Automáticamente realiza una limpieza de las conexiones conectadas sobre una instancia indisponible de base de datos antes de reingresarlas al pool.
- En un ambiente en Oracle RAC, ODP.NET estará informado en todo momento sobre la carga actual de cada una de las instancias de base de datos, permitiendo que al abrir nuevas conexiones se reubiquen sobre la instancia menos congestionada. Asimismo reconoce cuando nuevos nodos son agregados al clúster y los considerará para la colocación de nuevas conexiones que se soliciten.

FCF consigue una detección rápida de indisponibilidad y además enterarse de cada evento que ocurre en nuestra base de datos Oracle RAC mediante Fast Application Notification (FAN).

FAN es un mecanismo de notificación que Oracle RAC utiliza para notificar a otros procesos sobre su nivel de servicio, disponibilidad y configuración. Estas notificaciones FAN son las que nuestra aplicación podrá recibir para tomar alguna acción. La publicación de estos eventos FAN se realiza mediante Oracle Notification Service (ONS).

Se presenta la siguiente arquitectura:

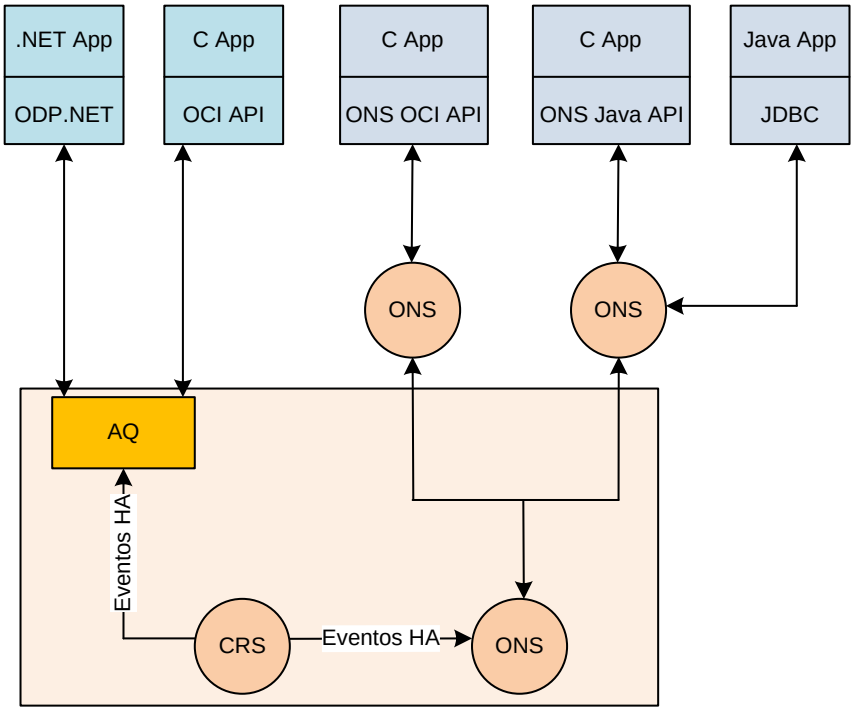


Figura 1

En la figura 1 podemos apreciar que dependiendo del tipo de aplicación que tengamos, los mensajes FAN serán recibidos ya sea por un componente ONS o mediante una cola de mensajes en Oracle Advanced Queuing (AQ).

Nuestra implementación que más adelante se documentará estará basado en un desarrollo sobre Microsoft .NET utilizando ODP.NET para el acceso de una base de datos Oracle RAC 12c compuesto de 2 nodos.

Nuestra base de datos estará corriendo sobre una plataforma Oracle Linux v5.10 x64 bits y nuestra aplicación será de tipo cliente/servidor corriendo sobre clientes Windows 7 x64 bits.

Adicionalmente contamos con un servidor DNS que resolverá las IPs de los hostnames solicitados, además por buena práctica, será responsable de devolver cualquiera de las 3 IP's asociadas al IP SCAN de nuestra configuración de Oracle RAC 12c.

La aplicación utilizará Oracle Data Access Components (ODAC) versión 12c de x64 bits como driver para el acceso a la base de datos.

La arquitectura de nuestra solución será la siguiente:

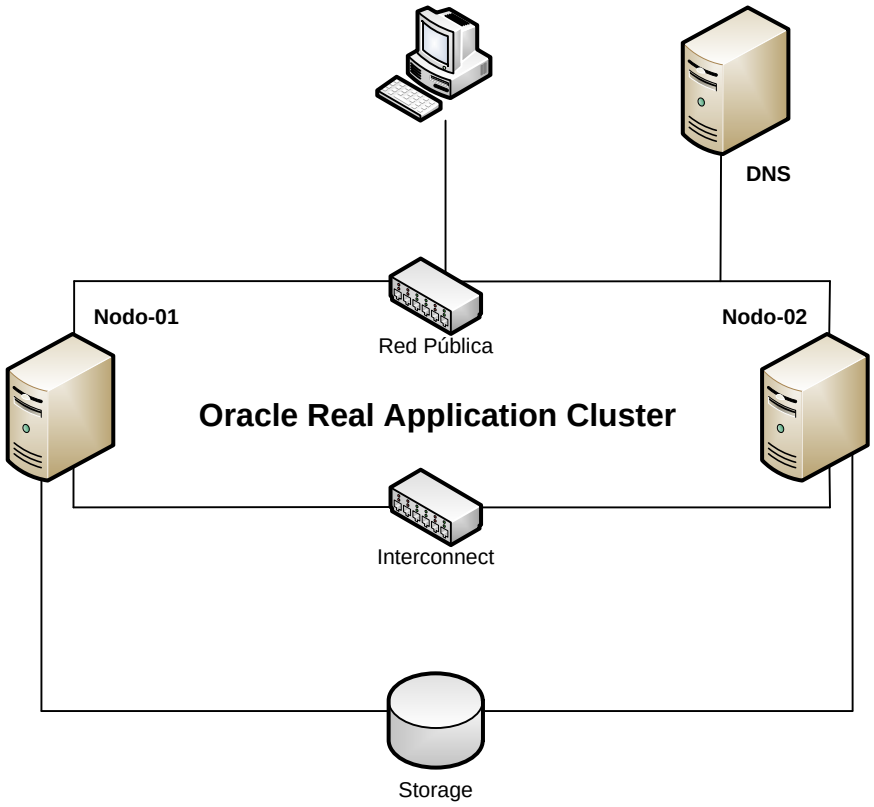


Figura 2

Nota 1: Como consideración en nuestro diseño, no deberíamos implementar FCF y TAF juntos como mecanismo de failover de conexión para nuestras aplicaciones. Recordemos que TAF es útil cuando son aplicaciones que ejecutan muchas operaciones de lectura.

Nota 2: A partir de la versión Oracle Database 12c, FCF ha sido extendido hacia Oracle Dataguard y Oracle GoldenGate para ODP.NET 12c a través de Global Data Services.

Nota 3: Cuando un servicio de base de datos se detiene de manera manual las conexiones de base de datos que se encuentran en estado de ocio son cerradas y las que aún están en ejecución terminan su operación con normalidad. Este feature llamado (Planned Outage) solo está disponible para ODP.NET Unmanaged Driver.

Nota 4: ODP.NET no soporta la distribución de conexiones de base de datos cuando un nuevo nodo inicia, sin embargo si soporta failover de conexión de base de datos y re-conectarlo al nuevo nodo si lo ve conveniente.

Implementación

Requisitos

Paso 1.- Parámetro AQ_TM_PROCESSES > 0

El parámetro aq_tm_process establece el número de procesos internos utilizados para la gestión de las colas en Oracle AQ.

Debemos recordar que Oracle AQ será el repositorio que tendrá los mensajes FAN que deseamos recoger para ser notificados.

Paso 2.- Habilitamos Advanced Queuing a nuestro servicio de base de datos (Opción –q true).

srvctl modify service –d PRD –s PROD –q true

Procedemos a su validación:

```
[oracle@srv2-121 ~]$ srvctl config service -d PRD -s PROD
Service name: PROD
Service is enabled
Server pool: PRD_PROD
Cardinality: 2
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: true
Global: false
Commit Outcome: false
Failover type:
Failover method:
TAF failover retries:
TAF failover delay:
Connection Load Balancing Goal: SHORT
Runtime Load Balancing Goal: SERVICE_TIME
TAF policy specification: NONE
Edition:
Pluggable database name:
Maximum lag time: ANY
SQL Translation Profile:
Retention: 86400 seconds
Replay Initiation Time: 300 seconds
Session State Consistency:
Preferred instances: PRD1,PRD2
Available instances:
```

Nota: Para la versión Oracle Database 10gR2 podemos modificar el servicio con el siguiente script:

SQL> execute dbms_service.modify_service(service_name=>'PROD',**aq_ha_notifications=>true**);

Paso 3.- Permisos sobre la cola de Oracle AQ sobre eventos FAN.

SQL> execute DBMS_AQADM.GRANT_QUEUE_PRIVILEGE('DEQUEUE','SYS.SYS$SERVICE_METRICS', '<NOMBRE_USUARIO_BD> ');

Aquí debemos utilizar el usuario de base de datos con el que nuestra aplicación se conectará.

En el ejercicio a desarrollar, el usuario de base de datos que será utilizado por la aplicación para conectarse será: FRICCIO.

Paso 4.- SQLNET.ORA

El archivo SQLNET.ORA ubicado en la estación cliente debemos agregarle las siguientes entradas:

SQLNET.OUTBOUND_CONNECT_TIMEOUT = 5 (Tiempo en segundos que tiene un cliente para establecer una conexión hacia una instancia de base de datos).

SQLNET.TCP.CONNECT_TIMEOUT=3 (Tiempo en segundos para establecer una conexión TCP hacia la base de datos, este valor debe ser menor al parámetro SQLNET.OUTBOUND_CONNECT_TIMEOUT).

El archivo SQLNET.ORA ubicado en las instancias de base de datos debemos agregarle la siguiente entrada como recomendación:

SQLNET.EXPIRE_TIME=10 (Tiempo en segundos que se realizará un ping a las conexiones de base de datos con la finalidad de identificar conexiones huérfanas).

Aplicación .NET

La cadena de conexión de base de datos debe contar con una correcta configuración, para nuestro caso hemos registrado la configuración en el archivo app.config.

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>

  <appSettings>

    <add key="CONEXION" value="Data Source=(DESCRIPTION = (ADDRESS_LIST =
(FAILOVER=YES)(LOAD_BALANCE=YES)(ADDRESS = (PROTOCOL = TCP)(HOST = srvscan-
121.riccio.com)(PORT = 1521)))(CONNECT_DATA=(SERVICE_NAME = PROD)(SERVER =
DEDICATED)));User Id=friccio;Password=oracle;Pooling=true;Min Pool Size=1;Max Pool Size=1;
```

```
Validate Connection=true;HA Events=true;Load Balancing=true"/>
 </appSettings>
</configuration>
```

Analizaremos las características incluidas en la cadena de conexión.

- Pooling, construirá un pool de conexión si su valor es true.
- Min Pool Size & Max Pool Size, definen la mínima y máxima cantidad de conexiones de base de datos que tendrá el pool creado.
- Validate Connection, valida si una conexión es válida cuando regresa al pool.
- HA Events, permite recibir eventos en nuestra aplicación al recibir ODP.NET un evento FAN.
- Load Balancing, habilita al pool de conexiones balancear las nuevas conexiones que ingresen acorde a cómo está la carga distribuida actualmente entre las instancias de base de datos basándose en los algoritmos definidos por el Load Balance Advisor (LBA) y Connection Load Balancing (CLB). Permitiendo que las aplicaciones reciban conexiones de una instancia que ofrece el mejor desempeño.
- El usuario de base de datos debe ser el mismo que le hemos entregado permisos en la cola de mensajes de AQ descrito en la sección de Requisitos.

Para implementar FCF en nuestra aplicación es de carácter obligatorio contar con las opciones: Pooling, HA Events y muy recomendable Load Balacing.

Se recomienda la revisión del siguiente URL para obtener mayor información acerca de todas las opciones disponibles en la cadena de conexión:

http://docs.oracle.com/cd/E51173_01/win.122/e17732/featConnecting.htm#ODPNT165

Nota: ODP.NET 12c únicamente está certificado para sistemas operativos Windows Server 2012/2008R2/2008 y Windows 7. Para mayor información o actualización revisar en **My Oracle Support (MOS) Nota: 726240.1 (Oracle Data Provider for .NET (ODP) Supported Configurations).**

Una vez definida la cadena de conexión de base de datos debemos programar el evento HAEventHandler(OracleHAEventArgs eventArgs), el cual se disparará ni bien se reciba un evento FAN del Oracle RAC.

```
private OracleConnection getConexion()
{
    string conexion = System.Configuration.ConfigurationManager.AppSettings["CONEXION"].ToString();
    if (_conexion == null)
    {
        _conexion = new OracleConnection(conexion);
        _conexion.Open();
        reloj.Enabled = true;
        lblestado.Text = "Estado: Normal";
        OracleConnection.HAEvent += new OracleHAEventHandler(HAEventHandler);
    }
    return _conexion;
}

public void HAEventHandler(OracleHAEventArgs eventArgs)
{
    if ((eventArgs.Status == OracleHAEventStatus.Down) && (eventArgs.Reason == "FAILURE")
        && (eventArgs.ServiceName.ToLower().ToString() == "prod"))
    {
        if (eventArgs.InstanceName.ToLower() == _conexion.InstanceName.ToLower())
        {
            _conexion.Close();
            _conexion.Open();
        }
    }
}
```

Podemos apreciar que parámetro eventArgs nos devuelve la información recibida del evento FAN. Para nuestro caso preguntamos si ha habido alguna caída en el servicio PROD, si es así, solicitaremos abrir de nuevo la conexión de base de datos y este será conectado automáticamente a una de las instancias sobrevivientes que publican el mismo servicio con el mejor desempeño.

Nuestra aplicación a implementar tendrá como objetivo mostrar la instancia donde se ha establecido la conexión de base de datos y la hora actual en todo momento.

Se presenta todo el código utilizado:

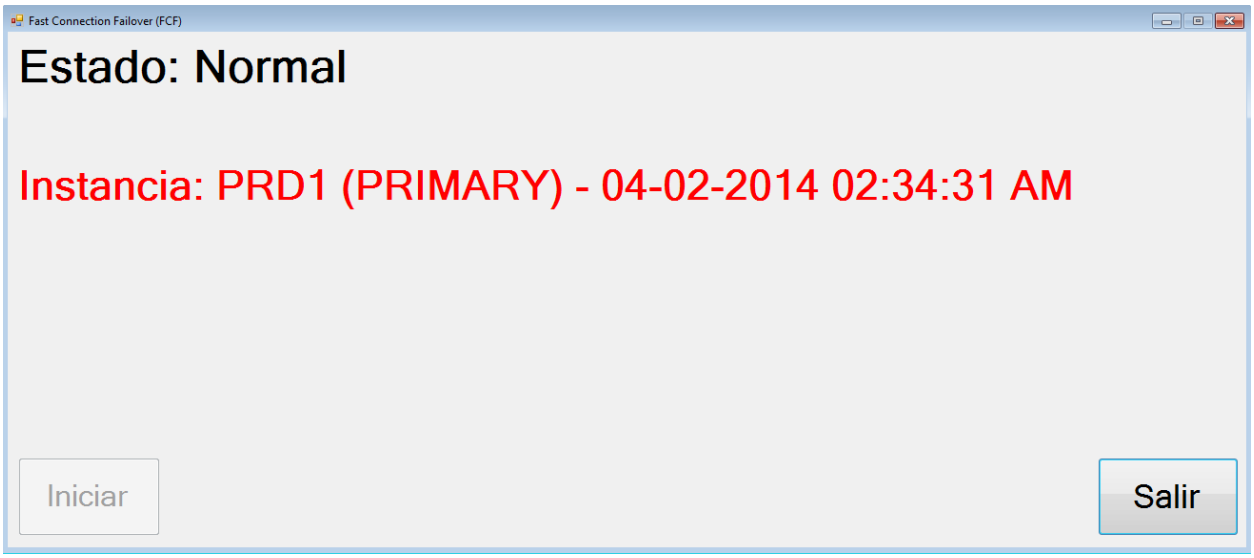
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Oracle.DataAccess.Client;

namespace AppFCF
{
    public partial class FrmFCF : Form
    {

        private OracleConnection _conexion = null;

        public FrmFCF()
        {
            InitializeComponent();
        }
```

```
        private OracleConnection getConexion()
        {
            string conexion = System.Configuration.ConfigurationManager.AppSettings["CONEXION"].ToString();
            if (_conexion == null)
            {
                _conexion = new OracleConnection(conexion);
                _conexion.Open();
                reloj.Enabled = true;
                lblestado.Text = "Estado: Normal";
                OracleConnection.HAEvent += new OracleHAEventHandler(HAEventHandler);
            }
            return _conexion;
        }

        public void HAEventHandler(OracleHAEventArgs eventArgs)
        {
            if ((eventArgs.Status == OracleHAEventStatus.Down) && (eventArgs.Reason == "FAILURE")
                && (eventArgs.ServiceName.ToLower().ToString() == "prod"))
            {
                if (eventArgs.InstanceName.ToLower() == _conexion.InstanceName.ToLower())
                {
                    _conexion.Close();
                    _conexion.Open();
                }
            }
        }

        private void btniniciar_Click(object sender, EventArgs e)
        {
            getConexion();
            btniniciar.Enabled = false;
        }

        private void FrmTAF_Load(object sender, EventArgs e)
        {
            lblinstancia.Text = "";
            lblestado.Text = "";
        }

        private void btnsalir_Click(object sender, EventArgs e)
        {
            reloj.Enabled = false;
            if (_conexion != null)
            {
                _conexion.Close();
            }
            _conexion = null;
            Application.Exit();
        }


        private void reloj_Tick(object sender, EventArgs e)
        {
            try
            {
                if (_conexion.State == ConnectionState.Open)
                {

                    OracleCommand cm = _conexion.CreateCommand();
                    cm.CommandText = "select instance_name||' ('||(select database_role from v$database)||')
                     - '||to_char(sysdate,'DD-MM-YYYY HH24:MI:SS AM') as texto from v$instance";
                    cm.CommandType = CommandType.Text;
                    string instancia = cm.ExecuteScalar().ToString();
                    lblinstancia.Text = "Instancia: " + instancia;
                    cm.Dispose();
                }
            }
            catch (Exception e1)
            {
                Console.WriteLine(e1.Message);
            }
        }

    }
}
```

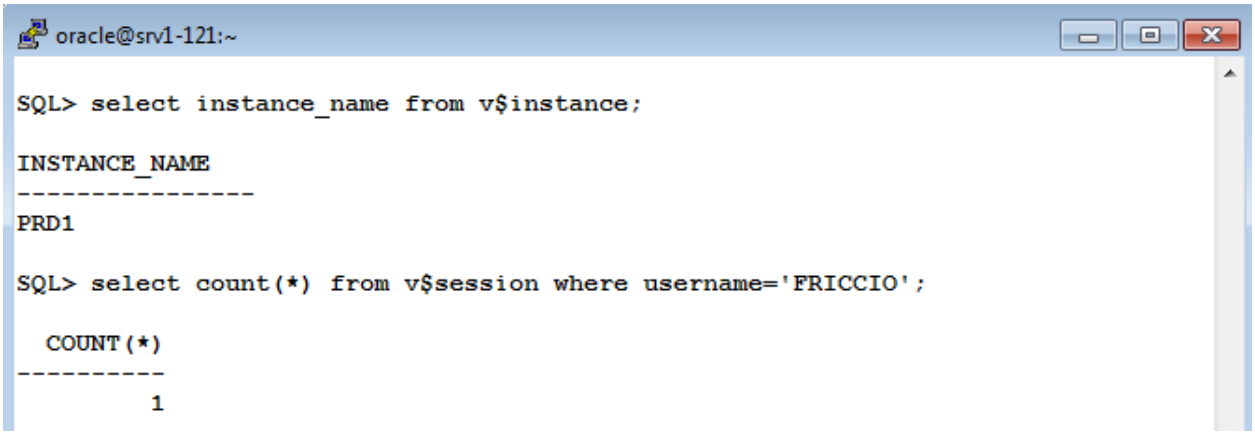
Es importante observar que nosotros somos responsable del manejo de la conexión una vez que se haya disparado el evento al recibirse un mensaje FAN. FCF no creará conexiones por nosotros.

Validando la aplicación:

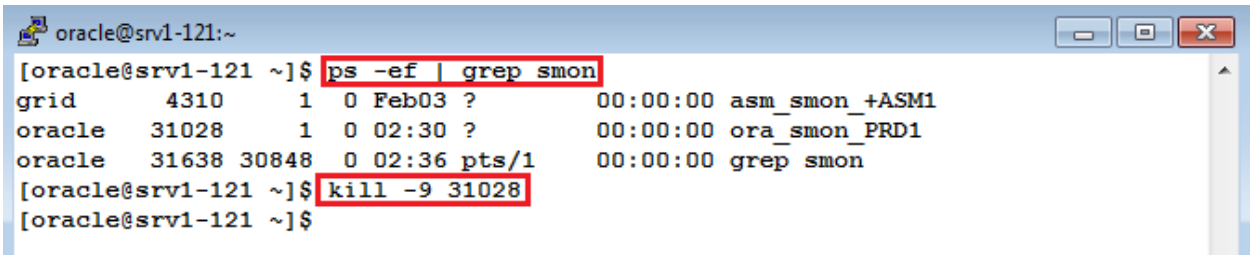
Al iniciar la aplicación veremos que se ha generado una conexión de base de datos a la instancia PRD1.



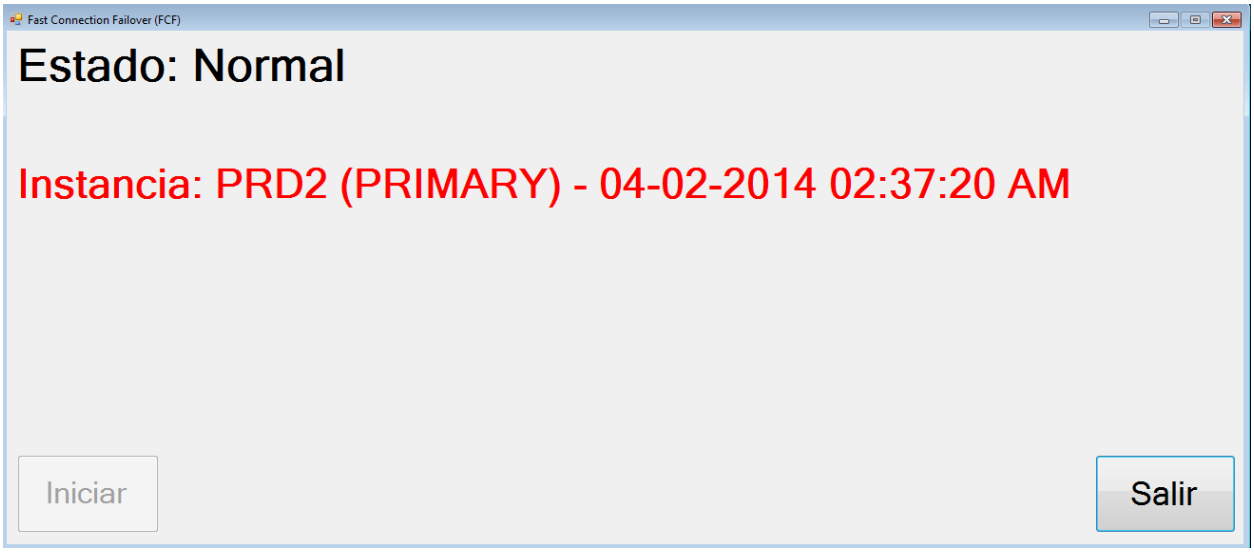
En la instancia 1 se evidencia que efectivamente se encuentra la sesión de base de datos.



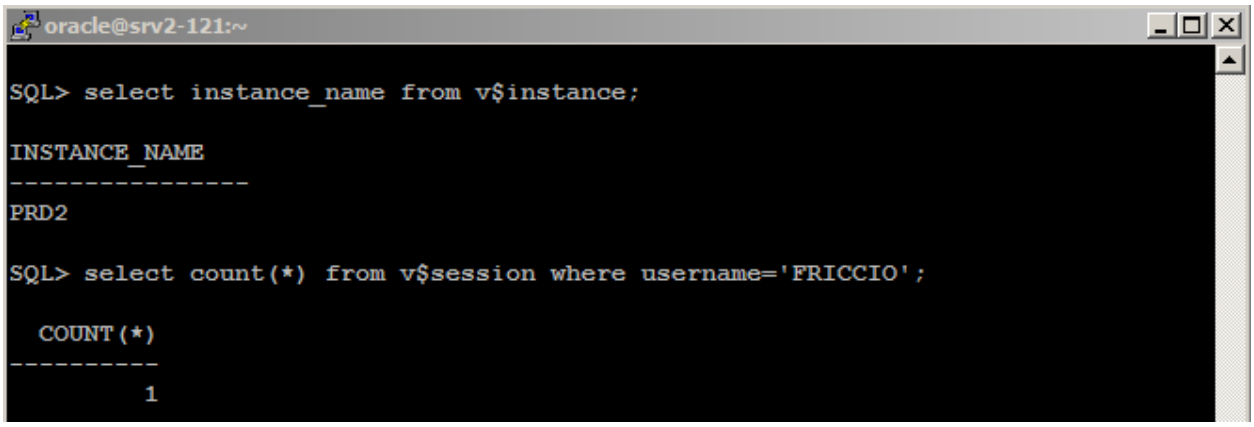
Luego procedemos a matar la instancia PRD1 con la finaldiad de validar que el failover de conexión se realice automáticamente.



Veremos que la aplicación automáticamente comienza a responder sin problemas pero conectado a la instancia PRD2.



En la instancia 2 se evidencia que efectivamente se encuentra la sesión de base de datos.



Conclusiones

Se puede apreciar que Oracle FCF complementa nuestras soluciones de Alta Disponibilidad proporcionándonos un failover de conexión de base de datos a nuestras aplicaciones; ocasionando que nuestros usuarios no perciban el incidente ocurrido en la base de datos.

FCF permite que nuestras aplicaciones que utilizan pool de conexiones de base de datos se beneficien enormemente con el feature de carga de balanceo y actualizadas gracias a las notificaciones FAN, haciendo que nuestra aplicación sea más eficiente al ir distribuyendo las conexiones nuevas sobre las instancias más óptimas además que todo requerimiento de base de datos que envié la aplicación será atendida por conexiones que estén conectadas sobre las instancias que ofrecen mejor desempeño.

Publicado por Ing. Francisco Riccio. Es un IT Architect en IBM Perú e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application & Base de Datos.

e-mail: francisco@friccio.com

web: www.friccio.com