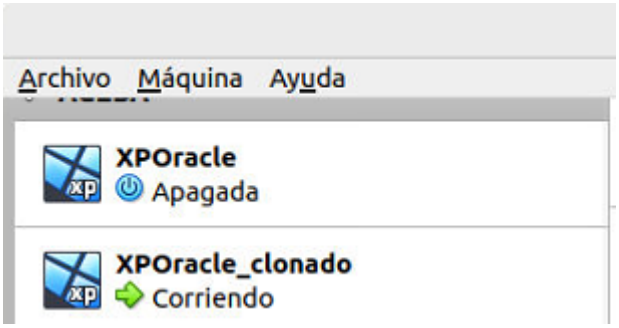
Ejercicio bases de datos distribuidas:

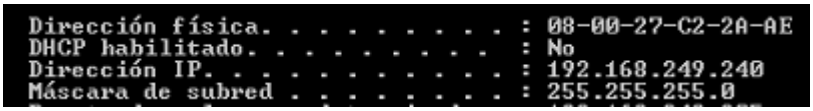
1) Genera una copia de la máquina virtual oracle, le cambias la dirección IP, y asegúrate que ambas máquinas virtuales tienen conectividad entre sí:

Hemos clonado la máquina virtual, y al clon lo hemos llamado “XPOracle_clonado”:



IP de máquina original:

IP de máquina clon:



Hacemos ping de máquina original (192.168.249.242) al clon (192.168.249.240):

Hacemos también ping de la máquina clon a la original:

Es decir, hay conectividad entre ambas máquinas.

2) Crear links entre los oracles de las dos máquinas virtuales, si es necesario editar el fichero tsnames.ora

Creamos en la máquina virtual original el enlace *enlaceorig*:

```
create database link enlaceorig connect to system identified by system
using '(description = (address = (protocol = TCP)(host = VIKI) (port=
1521)) (connect_data = (sid = ASIR)))';
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

Enlace con la base de datos creado.

Igualmente, creamos en la máquina virtual clon el enlace *enlaceclon*:

```
create database link enlaceclon connect to system identified by system
using '(description = (address = (protocol = TCP)(host = winxp) (port=
1521)) (connect_data = (sid = ASIR)))';
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

Enlace con la base de datos creado.

Añadimos en el fichero tnsnames.ora en la máquina original las líneas (C:\oracle\ora92\network\admin\tnsnames.ora):

```
BBDD_CONECTAR_MAQ_VIRT =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = "VIKI")(PORT = "1521"))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ASIR)
    )
  )
```

Y en la máquina clon:

```
BBDD_CONECTAR_MAQ_VIRT2 =

  (DESCRIPTION =

    (ADDRESS_LIST =

      (ADDRESS = (PROTOCOL = TCP)(HOST = "winxp")(PORT = "1521"))

    )

    (CONNECT_DATA =

      (SERVICE_NAME = ASIR)

    )

  )
```

3) Con la utilidad de tnsping probar que es posible conectar la original a la copia

En la máquina original:

```
C:\Documents and Settings\usuario>tnsping BBDD_CONECTAR_MAQ_VIRT 2

TNS Ping Utility for 32-bit Windows: Version 9.2.0.1.0 - Production on 01-DIC-20
22 17:27:00

Copyright (c) 1997 Oracle Corporation.  All rights reserved.

Archivos de par@metros utilizados:
C:\oracle\ora92\network\admin\sqlnet.ora


Adaptador TNSNAMES utilizado para resolver el alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)
(HOST = VIKI)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = ASIR)))
Realizado correctamente (10 mseg)
Realizado correctamente (0 mseg)
```

4) Crear un usuario user1 con los permisos “Create database link”, “create public database link” y el rol resource:

```
create user user1 identified by user1;
```

Ejecutar | Guardar Archivo de Comandos | Limpia

Usuario creado.

```
grant create database link to user1;
```

Ejecutar | Guardar Archivo de Comandos | Limpiar

Concesión terminada correctamente.

```
grant create public database link to user1;
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Par

Concesión terminada correctamente.

```
GRANT RESOURCE TO USER1;
```

Ejecutar | Guardar Archivo de Comandos

Concesión terminada correctamente.

5) Crear un usuario user2 en la máquina copiada con los permisos “create session” y el rol resource

```
CREATE USER USER2 IDENTIFIED BY USER2;
```

Ejecutar | Guardar Archivo de Comandos | Lin

Usuario creado.

```
GRANT CREATE SESSION TO USER2;
```

Ejecutar | Guardar Archivo de Comandos

Concesión terminada correctamente.

```
GRANT RESOURCE TO USER2;
```

Ejecutar | Guardar Archivo de Comandos

Concesión terminada correctamente.

6) Crear una tabla llamada clientes en la máquina original, en el esquema de usuario1 con los campos cod_cliente, nombre y dirección, e insertar dos registros.

CREATE TABLE CLIENTES (COD_CLIENTE INT PRIMARY KEY,
NOMBRE VARCHAR2(50) NOT NULL,
DIRECCION VARCHAR2(100) NOT NULL);
INSERT INTO CLIENTES VALUES (1, ‘SANDRA A’, ‘SEVILLA’);
INSERT INTO CLIENTES VALUES (2, ‘JOSE B’, ‘MADRID’);

```
connect user1/user1;
```

Ejecutar | Guardar Archivo de Coman

Conectado.

```
create table CLIENTES (COD_CLIENTE INT PRIMARY KEY,
NOMBRE VARCHAR2(50) NOT NULL,
DIRECCION VARCHAR2(100) NOT NULL);
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla |

Tabla creada.

```
INSERT INTO CLIENTES VALUES (1, 'SANDRA A', 'SEVILLA');
INSERT INTO CLIENTES VALUES (2, 'JOSE B', 'MADRID');
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Canc

1 fila creada.

1 fila creada

7) Cree una tabla llamada mascotas en la máquina copiada, en el esquema user 2, con los campos código de mascota, tipo raza y nombre (la clave ajena es el código del dueño). Insertar dos mascotas para cada cliente, por tanto si arriba hay dos clientes, abajo cuatro mascotas

CREATE TABLE MASCOTAS (CODIGO_MAS INT PRIMARY KEY,
RAZA VARCHAR2(15) NOT NULL,
NOMBRE_M VARCHAR2(20) NOT NULL,
DUENO INT NOT NULL);
INSERT INTO MASCOTAS VALUES (1, 'DALMATA', 'BECKY', 1);
INSERT INTO MASCOTAS VALUES (2, 'DESCON', 'BOLITA', 1);
INSERT INTO MASCOTAS VALUES (3, 'YORKSHIRE', 'NIEBLA', 2);
INSERT INTO MASCOTAS VALUES (4, 'PASTORALEMAN', 'CAPITAN', 2);

```
CREATE TABLE MASCOTAS (CODIGO_MAS INT PRIMARY KEY,
RAZA VARCHAR2(15) NOT NULL,
NOMBRE_M VARCHAR2(20) NOT NULL,
DUENO INT NOT NULL);
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla

Tabla creada.

```
INSERT INTO MASCOTAS VALUES (1, 'DALMATA', 'BECKY', 1);
INSERT INTO MASCOTAS VALUES (2, 'DESCON', 'BOLITA', 1);
INSERT INTO MASCOTAS VALUES (3, 'YORKSHIRE', 'NIEBLA', 2);
INSERT INTO MASCOTAS VALUES (4, 'PASTORALEMAN', 'CAPITAN', 2);
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

1 fila creada.

1 fila creada.

1 fila creada.

1 fila creada.

8) Ejecuta una consulta que devuelva un listado con las mascotas de cada cliente, incluyendo la dirección y el nombre del cliente.

SELECT * FROM USER1.CLIENTES CLI, (SELECT * FROM USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO;

```
SELECT * FROM USER1.CLIENTES CLI, (SELECT * FROM
USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO;
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

COD_CLIENTE	NOMBRE	DIRECCION	CODIGO_MAS	RAZA	NOMBRE_M	DUENO
1	SANDRA A	SEVILLA	1	DALMATA	BECKY	1
1	SANDRA A	SEVILLA	2	DESCON	BOLITA	1
2	JOSE B	MADRID	3	YORKSHIRE	NIEBLA	2
2	JOSE B	MADRID	4	PASTORALEMAN	CAPITAN	2

```
SELECT * FROM USER1.CLIENTES CLI, (SELECT * FROM
USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO;
```

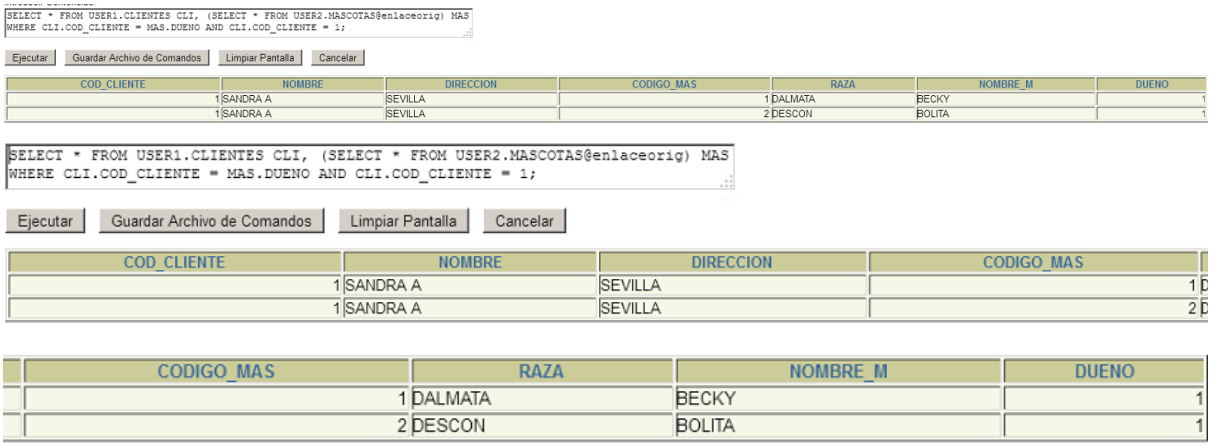
Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

COD_CLIENTE	NOMBRE	DIRECCION	CODIGO_MAS
1	SANDRA A	SEVILLA	1
1	SANDRA A	SEVILLA	2
2	JOSE B	MADRID	3
2	JOSE B	MADRID	4

CODIGO_MAS	RAZA	NOMBRE_M	DUENO
1	DALMATA	BECKY	1
2	DESCON	BOLITA	1
3	YORKSHIRE	NIEBLA	2
4	PASTORALEMAN	CAPITAN	2

9) Crea ahora la misma consulta, pero utilizando tablas derivadas, para atraer solamente las mascotas del primer cliente.

SELECT * FROM USER1.CLIENTES CLI, (SELECT * FROM USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO AND CLI.COD_CLIENTE = 1;



```
SELECT * FROM USER1.CLIENTES CLI, (SELECT * FROM USER2.MASCOTAS@enlaceorig) MAS
WHERE CLI.COD_CLIENTE = MAS.DUENO AND CLI.COD_CLIENTE = 1;
```

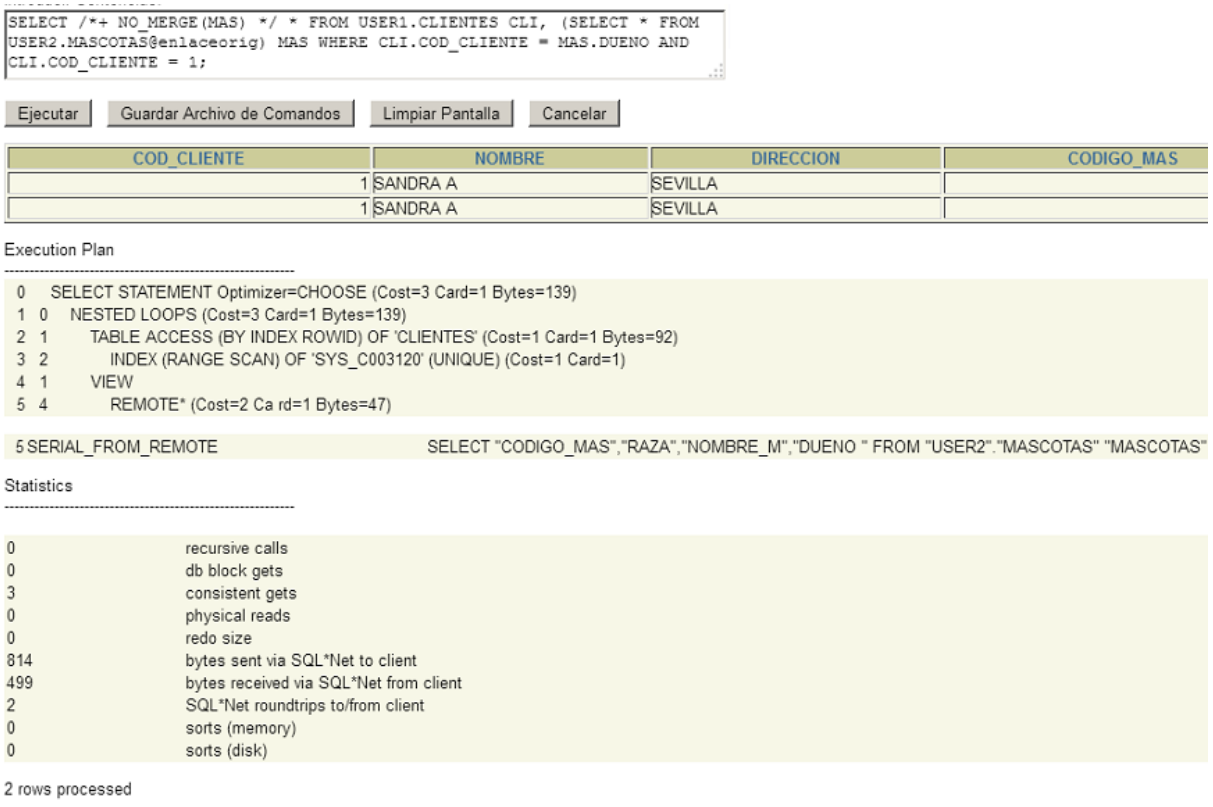
Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

COD_CLIENTE	NOMBRE	DIRECCION	CODIGO_MAS
1	SANDRA A	SEVILLA	1
1	SANDRA A	SEVILLA	2

CODIGO_MAS	RAZA	NOMBRE_M	DUENO
1	DALMATA	BECKY	1
2	DESCON	BOLITA	1

10) Ejecuta la consulta anterior usando los HINTS, no_merge y driving_site, comparando sus planes de ejecución con el comando explain plan.

CON HINT NO_MERGE:



```
SELECT /*+ NO_MERGE(MAS) */ * FROM USER1.CLIENTES CLI, (SELECT * FROM
USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO AND
CLI.COD_CLIENTE = 1;
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

COD_CLIENTE	NOMBRE	DIRECCION	CODIGO_MAS
1	SANDRA A	SEVILLA	
1	SANDRA A	SEVILLA	

Execution Plan

```
0   SELECT STATEMENT Optimizer=CHOOSE (Cost=3 Card=1 Bytes=139)
1  0   NESTED LOOPS (Cost=3 Card=1 Bytes=139)
2  1     TABLE ACCESS (BY INDEX ROWID) OF 'CLIENTES' (Cost=1 Card=1 Bytes=92)
3  2       INDEX (RANGE SCAN) OF 'SYS_C003120' (UNIQUE) (Cost=1 Card=1)
4  1     VIEW
5  4       REMOTE* (Cost=2 Ca rd=1 Bytes=47)

5 SERIAL_FROM_REMOTE                SELECT "CODIGO_MAS","RAZA","NOMBRE_M","DUENO " FROM "USER2"."MASCOTAS" "MASCOTAS"
```

Statistics

```
0          recursive calls
0          db block gets
3          consistent gets
0          physical reads
0          redo size
814        bytes sent via SQL*Net to client
499        bytes received via SQL*Net from client
2          SQL*Net roundtrips to/from client
0          sorts (memory)
0          sorts (disk)
```

2 rows processed

CON HINT DRIVING_SITE:

```
SELECT /*+ DRIVING_SITE(MAS) */ * FROM USER1.CLIENTES CLI, (SELECT *
FROM USER2.MASCOTAS@enlaceorig) MAS WHERE CLI.COD_CLIENTE = MAS.DUENO
AND CLI.COD_CLIENTE = 1;
```

Ejecutar | Guardar Archivo de Comandos | Limpiar Pantalla | Cancelar

COD_CLIENTE	NOMBRE	DIRECCION	CODIGO_MAS
1	SANDRA A	SEVILLA	
1	SANDRA A	SEVILLA	

Execution Plan

```
0      SELECT STATEMENT (REMOTE) Optimizer=CHOOSE
1    0   MERGE JOIN
2    1     SORT (JOIN)
3    2       TABLE ACCESS (FULL ) OF 'MASCOTAS'
4    1     SORT (JOIN)
5    4       REMOTE*
```

5 SERIAL_FROM_REMOTE SELECT "COD_CLIENTE","NOMBRE","DIRECCION" FR OM "USER1"."CLIENTES" "A2" WHERE "CO

Statistics

```
5          recursive calls
0          db block gets
2          consistent gets
0          physical reads
0          redo size
811        bytes sent via SQL*Net to client
499        bytes received via SQL*Net from client
2          SQL*Net roundtrips to/from client
0          sorts (memory)
0          sorts (disk)
```

2 rows processed

Vemos que si ejecutamos la consulta con el hint DRIVING_SITE, tenemos 2 consistent gets, mientras que si lo hacemos con el hint NO_MERGE - los consistent gets son 3; esto quiere decir que con el hint DRIVING_SITE en este caso (que indica en qué nodo se tiene que ejecutar la consulta) la consulta es más rápida y, por tanto, más optimizada.