PRÁCTICA 6
TRANSPARENCIA DE DISTRIBUCIÓN – MAPEOS LOCALES.

Esta práctica puede realizarse de forma individual o en equipo de máximo 2 integrantes.

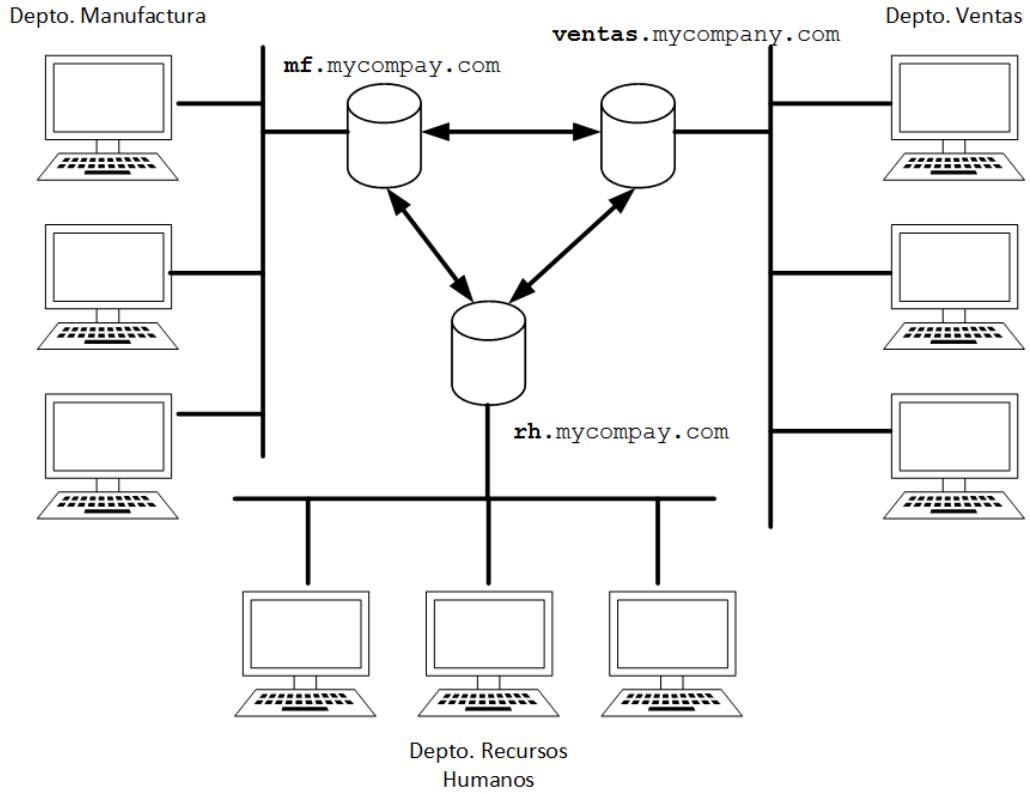
1.1. OBJETIVO.

Comprender la forma en la que se realiza la configuración de una base de datos Oracle para implementar el concepto de Transparencia de distribución en su primer nivel: Mapeos Locales. La implementación de este nivel se realizará a través del uso de PDBs y ligas (database links) para establecer una comunicación bidireccional.

1.2. ARQUITECTURA DE UNA BASE DE DATOS DISTRIBUIDA ORACLE

Existen 2 variantes de una BD distribuida. En un ***sistema de bases de datos distribuido homogéneo*** cada base de datos es una base de datos Oracle. En un ***sistema de bases de datos distribuido heterogéneo***, al menos una base de datos no es Oracle. Ambas variantes emplean la ***arquitectura cliente servidor*** para realizar el procesamiento de consultas.

Ejemplo:



En la figura anterior el sistema distribuido está formado por 3 bases de datos: `mf`, `rh` y `ventas`. Desde el punto de vista del usuario, la ubicación y la plataforma empleada por cada base de datos es totalmente transparente, de tal forma que es posible ejecutar sentencias como `select * from cliente`.

1.2.1. Base de datos distribuida vs Procesamiento distribuido.

Son 2 conceptos relacionados pero diferentes.

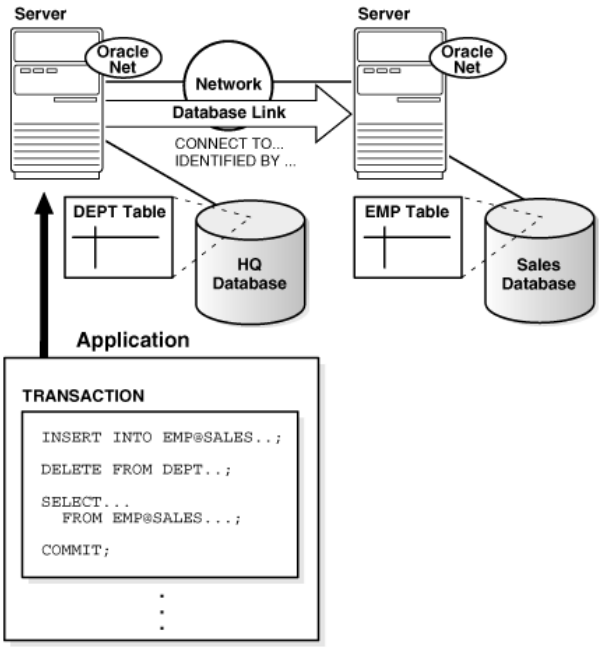
- *Base de datos distribuida*: Conjunto de bases de datos que pertenecen a un sistema distribuido que se presentan al usuario final como una sola fuente de datos.
- *Procesamiento distribuido*: Se refiere a la división de las tareas que realiza un sistema hacia diferentes computadoras o máquinas a lo largo de una red. El ejemplo típico es la distribución cliente – servidor en la que el procesamiento asociado con la vista y presentación de los resultados se delega a la máquina cliente mientras que las actividades relacionadas con la ejecución de sentencias SQL y acceso a datos las realiza el servidor.

Una base de datos distribuida Oracle hace uso de ***una arquitectura de procesamiento distribuido***. Un servidor de bases de datos puede actuar como cliente cuando requiere acceder a los datos de otro servidor en la red, y a la vez puede actuar como servidor cuando otros servidores le solicitan datos.

1.2.2. Arquitectura de base de datos Cliente – Servidor.

Un ***servidor de bases de datos*** está representado por el software que administra la base de datos Oracle. Un ***cliente*** es una aplicación que solicita datos al servidor.

Ejemplo:

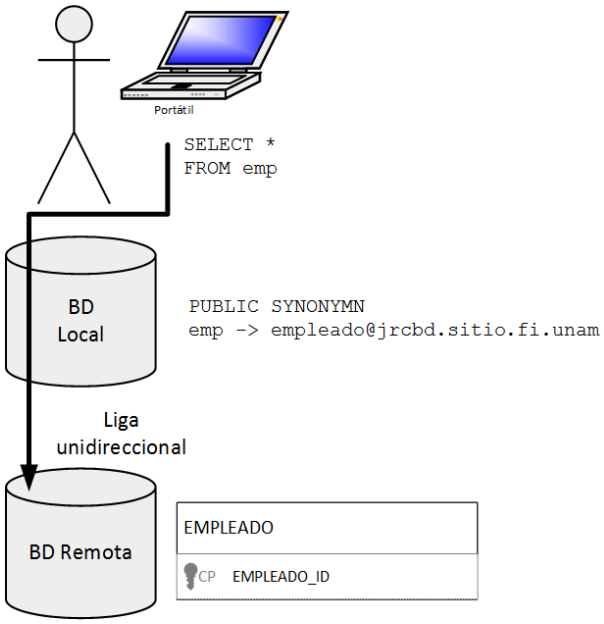


1.3. DATABASE LINKS.

- Representa un concepto fundamental dentro de una base de datos distribuida Oracle.
- Una liga es una conexión entre 2 servidores de bases de datos físicos que permite a clientes acceder a los datos de manera lógica.
- Una liga puede visualizarse como un puntero ***unidireccional*** de un servidor Oracle a otro. La definición de este puntero (liga) se almacena en el diccionario de datos.
- Existe una variante para este concepto: Una liga ***global***. En este caso, la definición de la liga se almacena en un directorio de servicios. Este directorio de servicios se encuentra en un servidor de directorios existente en la red lo que permite administrar todas las ligas que pudieran existir en un sistema distribuido de forma centralizada.
- Para hacer uso de la liga, se debe estar conectado a la base de datos local que contiene la definición de la liga.
- Existen 2 tipos de ligas
  - Privadas: Solo el usuario que la creó puede tener acceso a ella.
  - Públicas: Todos los usuarios pueden tener acceso a ella.
- Para que la conexión se pueda realizar de manera exitosa, cada base de datos dentro del sistema distribuido debe tener un ***nombre global único***.

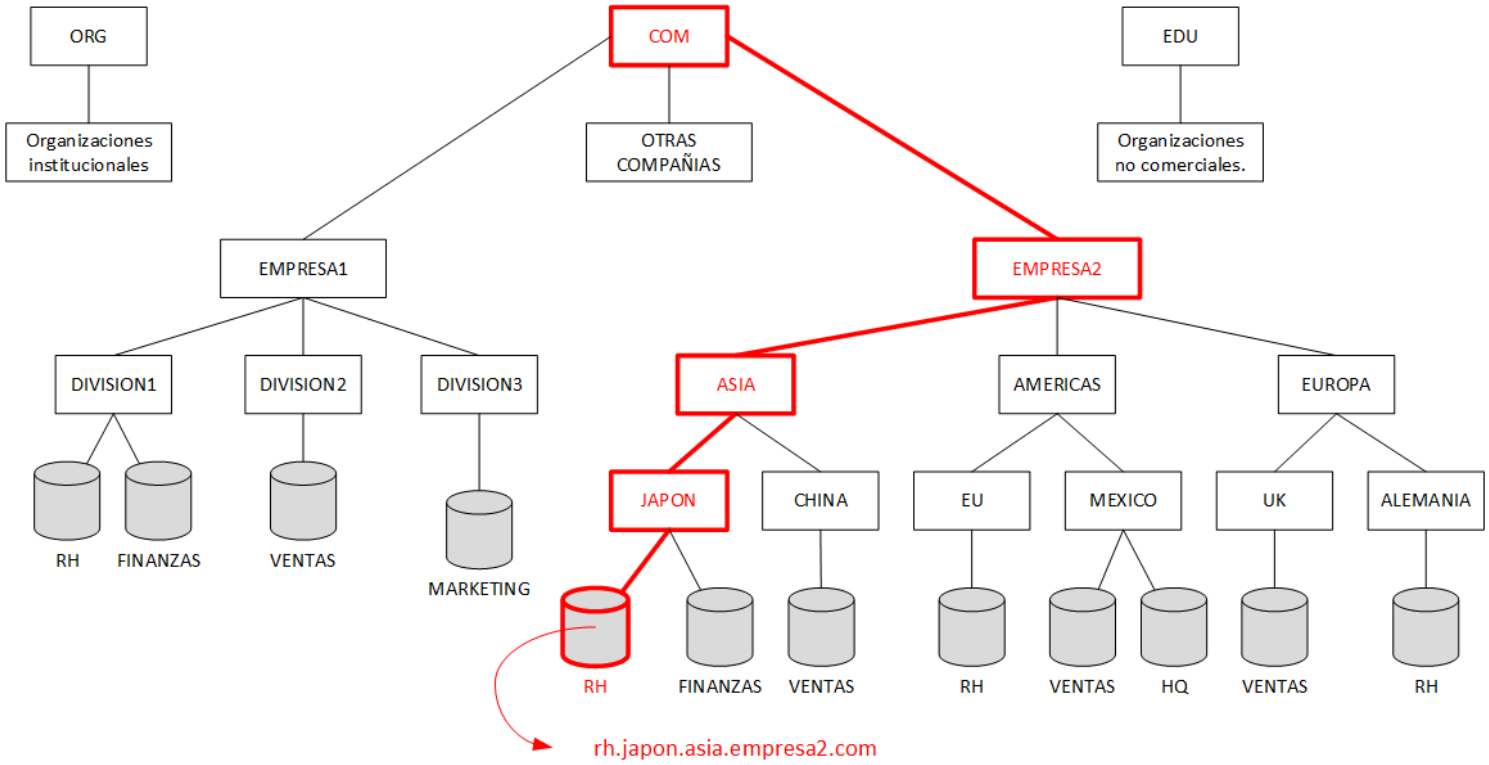
Ejemplo:

`jrcbd``.fi.unam` representa el nombre global de la base de datos, mientras que **`jrbd`** representa el ***identificador del sistema de la base de datos*** (SID), `fi.unam` representa el llamado ***dominio de red***.



A nivel general, se puede tener una estructura global de nombres únicos como se muestra en la figura siguiente.

- Observar la base de datos señalada, su nombre global y único es `rh.japon.asia.empresa2.com` (iniciando en las hojas de jerarquía o árbol de la figura).
- Como se puede observar, en una red pueden existir 2 bases de datos con el mismo nombre (`RH`), pero con nombre global único.
- En Oracle se emplean los parámetros `db_name` y `db_domain` que identifican de manera única a una base de datos.



- Observar que el nombre completo de la BD se lee de abajo hacia arriba (de las hojas del árbol hacia arriba).
- Es importante destacar que el valor del parámetro `db_name` es el mismo para cada PDB. Por lo tanto, para determinar el nombre global único de cada PDB se emplean los valores de los parámetros `con_name` y `db_domain` en lugar de los parámetros `db_name` Y `db_domain`.
- Para cada una de las PBDs (hasta 4 nodos) que integran a la BDD, generar un diagrama jerárquico similar al de la figura de la sección anterior (marcado en rojo) que muestre los nombres globales de cada PDB. Ejemplo: `jrcbd_s1.fi.unam` y `jrcbd_s2.fi.unam` **C1. Incluir en el reporte** el diagrama generado. **Notar que si la práctica se hace en equipo se tendría un solo diagrama con 4 nodos.**

1.3.1. Autenticación de usuarios empleando ligas.

Existe una clasificación más de las ligas que toma como criterio la forma en la que se autentican los usuarios:

- *Connected User link*: Se emplean los mismos datos de autenticación del usuario en la base de datos local para autenticarlo en la base de datos remota. Esto significa que en la base remota debe existir un usuario con el mismo nombre y password que en la base local. Lo anterior simplifica su creación ya que ***no requiere*** la especificación de usuarios y passwords en el código que se emplea para crear la liga. ***Esta será la técnica a emplear en el curso por simplicidad***.

- *Fixed user link:* Para conectarse a la base remota se emplea el usuario y password que se especifican al crear la liga (usuario y password fijos). Por ejemplo, el usuario `juan`, se conecta a la base remota `rh` con el usuario/password `hr_user/hr_user` que fueron especificados al crear la liga.

- *Current user link:* En este escenario el usuario se conecta como un ***usuario global*** a la base remota. Este usuario se considera global dentro del contexto, por ejemplo, de un procedimiento almacenado. No requiere almacenar usuario/password en la definición de la liga. Por ejemplo, el usuario `juan` tiene acceso a un procedimiento almacenado del usuario `paco` en la base remota. El usuario `juan` se conecta a la base remota como usuario `paco` para acceder al procedimiento almacenado, y de ahí realizar el acceso a datos. Nota: El usuario paco debe ser un usuario global el cual es autenticado y autorizado por un servicio de directorios, por ejemplo, Oracle Internet Directory.

1.3.2. Shared database links

- Se considera a una liga como compartida cuando múltiples procesos de usuario (user process) pueden hacer uso de una misma liga de manera simultánea. La base remota puede estar configurada ya sea en modo compartido o modo dedicado.
- Lo anterior puede aplicarse a escenarios como los siguientes:
  - Diferentes usuarios que acceden al mismo objeto a través de la misma liga pueden compartir la conexión.
  - Cuando un usuario (sesión) requiere conectarse a una base remota a partir de un "server process" (SP1) asociado a dicha sesión, este proceso puede reutilizar conexiones que se hayan establecido anteriormente a la base remota. Esto ocurre solo cuando se emplea el mismo SP1 y la misma liga. Si la liga no es compartida, las conexiones hacia una base remota no son compartidas entre las diferentes sesiones existentes en la base local.
- Investigar el significado de los siguientes conceptos en cuanto a la forma en la que se atienden las peticiones (de forma dedicada o compartida) **C2. Incluir en el reporte** la siguiente tabla:

Configuración de una BD Oracle	Descripción	Ventajas	Desventajas
Modo dedicado			
Modo compartido			

En esta práctica se realizará la simulación de una BDD empleando las PDBs creadas en prácticas anteriores. Cada PDB representará a un nodo de la BDD. Si la práctica se desarrolla en equipo se tendrán hasta 4 nodos (4PDBs). La comunicación se realizará únicamente entre PDBs de cada integrante del equipo. Posteriormente se realizarán conexiones entre diferentes equipos (BDD real).

1.4. CONFIGURACIÓN DE PRIVILEGIOS

- Crear un script SQL llamado `s-01-<iniciales>-privilegios-usuarios.sql` El script deberá realizar las siguientes acciones por cada integrante:
- Conectarse a cada PDB como usuario SYS
- Ejecutar las instrucciones necesarias para otorgar los privilegios al usuario creado en la práctica anterior requeridos para realizar las actividades de esta práctica. No olvidar agregar el encabezado a cada script, similar a los de la práctica anterior.
- Los privilegios que se emplearán en esta práctica son:
  - Otorgar permisos para crear ligas empleado el privilegio `create database link`.
  - Crear procedimientos almacenados: `create procedure`.
- Ejecutar el script. Seguir los mismos lineamientos de ejecución de scripts ilustrados en la práctica anterior.

1.5. CREACIÓN DE LIGAS.

En esta sección se describen todas las actividades necesarias para crear ligas. Las actividades de esta sección se deberán realizar de forma individual, es decir, por cada integrante del equipo.

1.5.1. Creación de usuarios en las PDBs

- Se hará uso de los usuarios creados en la práctica anterior y el esquema de fragmentación existente en cada PDB.

1.5.2. Sintaxis para crear una liga.

```
create [ shared] [public] database link dblink
[ connect TO
   { current user | user identified by password [dblink authentication]}
   | dblink_authentication
]...
[using connect_string];
```

Donde `bdlink_authentication` tiene la siguiente sintaxis:

```
authenticated by user identified by password
```

- *dblink* indica el nombre de la liga. Se recomienda emplear el nombre global de la base remota o PDB. Si solo se especifica el nombre de la base de datos, automáticamente se agrega el nombre del dominio de la base local.
- Si se omite PUBLIC, se creará una liga privada.
- La accesibilidad de los datos en la base remota depende de la identidad que utilice la liga al conectarse a la base remota:
  - Si se especifica `connect to user identified by password`, la liga se conecta con el usuario y password especificados.
  - Si se especifica `connect to current_user`, como se mencionó anteriormente, se realiza una conexión empleando un usuario global.
  - Si se omiten las 2 opciones anteriores, se realiza una conexión con los mismos datos de la sesión local.
- *dblink_authentication*: se especifica esta cláusula solo si se crean ligas compartidas. Se especifica el usuario y password que se emplearán para autenticar en la base remota.

- *USING 'connect string'* en esta cláusula se especifica el nombre del servicio de la base remota. Un nombre de servicio es una cadena que mapea a un servidor de base de datos ya sea local o en algún otro servidor. En Oracle se emplea el archivo `tnsnames.ora` localizado en `$ORACLE_HOME/network/admin`. Notar que el nombre del servicio debe aparecer entre comillas simples.
- En la práctica anterior, se realizó el procedimiento correspondiente empleando `netmgr` para crear los nombres de servicio para las PBDs. El siguiente ejemplo muestra parte de la configuración del archivo `tnsnames.ora`, la cadena marcada en negro corresponde al nombre del servicio y es la cadena que debe configurarse con la cláusula **`USING`**.

```
JRCBD_S1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = jrcbd_s1.fi.unam)
    )
  )
```

- Empleando el usuario creado en cada PDB, crear una liga privada que apunte a la otra PDB. Es decir, se crearán 2 ligas para establecer una comunicación bidireccional. Emplear como nombre de la liga el ***nombre global de la PDB***. Recordar: usar la técnica *Connected User link* en la que no se requiere especificar username y password dentro de la instrucción `create database link`.

Ejemplo:

Suponer que se va a crear una liga en `jrcbd_s1.fi.unam` para conectarse `a jrcbd_s2.fi.unam`. El nombre de la liga será el mismo que el nombre global de la BD remota. En este caso el nombre será: `jrcbd_s2.fi.unam`

En forma gráfica:

En `jrcbd_s2` se deberá aplicar el mismo procedimiento para que la comunicación sea bidireccional:

- Generar un solo script SQL por integrante llamado `s-02-<iniciales>-creacion-ligas.sql` Ejecutar el script.

Ejemplo:

```
prompt Creando liga en jrcbd_s1 hacia jrcbd_s2
connect jorge/jorge@jrcbd_s1
create database link jrcbd_s2.fi.unam using 'JRCBD_S2';

prompt Creando liga en jrcbd_s2 hacia jrcbd_s1
connect jorge/jorge@jrcbd_s2
--completar

prompt Listo.
exit
```

- Hasta este punto las PDBs de cada integrante están listas para poder comunicarse de forma bidireccional.

1.6. TRANSPARENCIA DE MAPEO LOCAL.

- Recordando de la práctica anterior, para poder consultar los datos en una PDB remota, se requiere conectarse a dicha PDB empleando el comando `connect`.
- Las creaciones de ligas permiten implementar el nivel de transparencia de distribución de mapeo local.
- Lo anterior significa que ahora será posible acceder a los datos de una PDB remota sin tener que hacer uso del comando `connect` y realizar conexiones de forma manual.

Para comprobar lo anterior, realizar las siguientes actividades.

- Generar un script SQL por cada PDB llamado `s-03-<SID>-consultas.sql`
- Observar el término <SID>. Su valor corresponde al SID de las PDBs de cada integrante. (Hasta 4 archivos en caso de ser en equipo). Ejemplo: `s-03-`**`jrcbd_s1`**`-consultas.sql`
- A partir de este momento se inicia el uso de programación **PL/SQL** para realizar algunas actividades de las práctica actual y futuras. Se recomienda ampliamente revisar los apuntes del **tema 10** de Bases de Datos ubicada en la carpeta compartida BD.
- El script deberá contener un programa PL/SQL que realice lo siguiente:

1. Ejecutar una sentencia SQL que realice el conteo de registros en todos los fragmentos haciendo uso de las ligas creadas. Como se mencionó anteriormente, ya no será necesario conectarse de forma manual a la PDB remota. Para realizar el conteo de los registros en todos los fragmentos se deberá emplear la expresión de reconstrucción de cada tabla global.
2. El resultado de la consulta será asignado a una variable.
3. Imprimir el valor de las variables al final.

Ejemplo:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Script encargado de realizar consultas para comprobar ligas

prompt conectando a jrcbd_s1
connect editorial bdd@jrcbd s1
prompt Realizando conteo de registros
set serveroutput on
--La consulta se lanza en JRCBD_S1
declare
 v_num_paises number;
 --completar
begin

 dbms_output.put_line('Realizando consulta empleando ligas');

 --consultando paises
 select count(*) as into v_num_paises
 from (
  select pais id
  from  F_JRC_PAIS_1
  union all
  select pais_id
  from  F_JRC_PAIS_2@jrcbd_s2.fi.unam
 ) q1;
```

```
 -- realizar el mismo procedimiento para las demás tablas.

 dbms_output.put_line('Resultado del conteo de registros');
 dbms_output.put_line('Paises:         '||v_num_paises);
 --completar con las demás tablas.
/
Prompt Listo
exit
```

- En este ejemplo se ilustra la consulta para 2 fragmentos `f_jrc_pais_1` y `f_jrc_pais_2` ubicados en `jrcbd_s1` y `jrcbd_s2` respectivamente.
- Observar que solo se consulta el campo correspondiente a la PK para minimizar los datos a transmitir ya que solo se desea hacer un conteo de registros.
- Observar que el resultado de la consulta se asigna a la variable `v_num_paises`.
- Observar que se emplea el operador `union all` como expresión de reconstrucción para recuperar los registros de todos los fragmentos. En este caso, la unión de 2 fragmentos distribuidos de forma horizontal. Se usa `union all` ya que no es necesario filtrar duplicados ya que no debe existir registros duplicados entre fragmentos.
- Observar el uso de la liga **`jrcbd_s2.fi.unam`** para acceder al sitio remoto desde `jrcbd_s1`. Si la consulta se ejecutara en `jrcbd_s2`, se tendría lo siguiente:

```
--consultando paises
select count(*) as into v_num_paises
from (
 select pais id
 from  f_jrc_pais_1@jrcbd_s1.fi.unam
 union all
 select pais_id
 from  f jrc pais 2
) q1;
```

- En este caso la liga se especifica en el fragmento 1 ya que la consulta se lanza desde `jrcbd_s2`.
- Al final del programa se imprime el valor de las variables. La salida será algo similar al siguiente ejemplo:

```
idle> start s-03-jrcbd_s2-consultas.sql
conectando a jrcbd s2
Connected.
Realizando conteo de registros
Realizando consulta empleando ligas

Resultado del conteo de registros
=================================
Paises:                       2
Suscriptores:                 3
Articulos:                    2
Revistas:                     2
Articulo Revista:             2
Pago suscriptor:              2
```

C3. Incluir en el reporte el código únicamente para la tabla `suscriptor` por cada integrante y la salida de ejecución del script.

1.7. Manejo de datos binarios en Bases de datos distribuidas.

Hasta el momento no se ha considerado el manejo de datos binarios, en especial para tipos de datos CLOB y BLOB. El manejo de este tipo de información toma especial atención en especial en BDD. Si se intenta realizar la siguiente consulta se obtendrá un error:

```
my_user@jrcbd_s1> select * from f_jrc_articulo_1@jrcbd_s2;
ERROR:
ORA-22992: cannot use LOB locators selected from remote tables
no rows selected
```

Este error se produce debido a que se intenta consultar los datos del fragmento remoto `f_jrc_articulo_1` que se encuentra en `jrcbd_s2`. Dicho fragmento contiene un atributo tipo BLOB.
Internamente en un campo tipo BLOB se almacena un puntero al objeto binario (lob locator). Los punteros se vuelven inválidos fuera de su base de datos, por lo que en un sitio remoto carecerían de significado. Por esta razón se produce este error. En las siguientes prácticas se resolverá este inconveniente. En esta práctica solo se aprenderá a importar y exportar datos binarios de un directorio a un campo BLOB en la base de datos.

En esta sección se muestran los principales conceptos que ofrece Oracle para hacer un eficiente y correcto manejo.

1.7.1. Uso de Secure files.

En versiones anteriores de Oracle, se soportaba un solo tipo de almacenamiento para objetos tipo LOB (Large Objects), como son CLOB y BLOB. A este esquema se le conoce como ***BasicFiles LOBs*** . A partir de Oracle 11g se crea una nueva estrategia de almacenamiento llamada ***Secure Files***, y en Oracle 12c se vuelve la opción por default.

Esta nueva estrategia ofrece una mejora considerable con respecto a la anterior y ofrece un desempeño similar o mejor a los sistemas de un sistema de archivos de red tradicional. Adicionalmente, Secure File ofrece la posibilidad de habilitar compresión, cifrado y de-duplicación. De-duplicación permite identificar registros con valores idénticos de la columna CLOB o BLOB. Si esto ocurre, el valor solo se almacena una vez y es compartido por todos los registros.

Ejemplo:

```
create table t1(
    id number(10,0),
    image blob not null
) lob(image) store as securefile;
```

- Observar el uso de la función `lob`. Como se mencionó anteriormente el uso de secure files representa la técnica por defaut. El ejemplo muestra la sintaxis para especificar el tipo de almacenamiento de forma explícita.

1.7.2. Manejo de datos binarios empleando LOBs (Large Objects)

Antes de revisar la solución para resolver el problema de transferencia de datos binarios entre fragmentos, se revisa en esta sección la forma en la que se realizará el manejo de datos binarios. Básicamente se tienen 2 etapas:

A. Cargar un archivo binario ubicado en un directorio del servidor a una tabla que define un campo tipo BLOB.
B. Realizar la operación inversa: exportar un objeto BLOB de la Base de Datos hacia un archivo binario en algún directorio del servidor.

Para realizar estas 2 tareas es necesario realizar programación PL/SQL.

1.7.2.1. Carga de archivos binarios a la Base de Datos.

La estrategia para realizar esta tarea es la siguiente:

- Se tendrá un archivo en un directorio que se desea guardar en la BD.
- Del lado de la BD existirá un procedimiento almacenado que aceptará los siguientes parámetros:
    - Directorio donde se encuentra el archivo a cargar
    - Nombre del archivo a cargar
    - Nombre de la tabla donde se guardará el objeto binario.
    - Nombre del campo tipo BLOB en el que se debe almacenar el objeto binario.

Adicional a estos parámetros se necesita resolver la siguiente pregunta: ¿A qué registro se le asociará el objeto binario? Para ello será necesario 2 parámetros más:

- Nombre del(los) campo(s) que forman a la PK de la tabla. Esto permite identificar de forma única al registro en el que se guardará el objeto binario.
- Valor de(los) campo(s) que forman a la PK.

En realidad, el proceso de carga no es complicado. La principal complejidad se debe a que este programa recibe nombres de tablas y campos, y por lo tanto debe formar sentencias ***SQL dinámicas*** para poder almacenar el objeto binario.
Por ejemplo: suponer que se tienen las siguientes variables con sus valores:

```
v_directory_name = 'PDF_DIR'
v_src_file_name  =  miArchivo.pdf'
v_table_name     =  'ARTICULO_PDF',
v_blob_column_name = 'PDF',
v_pk1_column_name  = 'ARTICULO_ID',
v_pk1_column_value  = 15
v_pk2_column_name  = 'NUM_PDF'
v_pk2_column_value 20
```

Se puede observar que existe una tabla llamada ARTICULO_PDF que tiene una llave primaria compuesta ARTICULO_ID y NUM_PDF con los valores 15 y 20 respectivamente (para efectos de las prácticas se consideran tablas hasta con 2 atributos máximo como PK). El objeto binario se deberá guardar en el campo PDF. Con esta información el programa deberá leer el contenido del archivo y generar una sentencia SQL dinámica similar a la siguiente:

```
update articulo_pdf
set pdf = v dato binario
where articulo_id = 15
and num_pdf = 20
```

Un punto importante de seguridad a considerar en este programa es la llamada inyección de SQL. Al recibir cadenas como valores de parámetros para formar SQL dinámico, el programa puede ser vulnerable a ataques. ¿Qué pasaría si el parámetro que recibe el nombre de la tabla se le asigna el valor `'drop database'`? Por lo anterior, el programa PL/SQL se debe diseñar de tal forma que sea inmune a estos ataques.

El programa PL/SQL que implementa estos requerimientos se encuentra en la carpeta de esta práctica: `s-00-carga-blob-en-bd.sql` Abrir el archivo, leer cuidadosamente los comentarios.

1.7.2.2. Exportar objetos BLOB a un archivo binario.

- El segundo paso es realizar la operación contraria: leer un objeto BLOB almacenado en la BD y guardarlo en un archivo binario en un directorio del servidor.
- La estrategia es similar a la anterior. El siguiente programa PL/SQL muestra una solución simple con los siguientes inconvenientes:
  - No soporta llaves primarias compuestas
  - Permite inyección de SQL.
- Crear un script llamado `s-00-guarda-blob-en-archivo.sql` El script deberá contener una nueva versión del siguiente código que implemente los requerimientos anteriores. Es decir:
  - El procedimiento deberá aceptar 2 parámetros más. El primero se empleará para indicar el nombre del segundo campo que forma a la PK, y el segundo parámetro se empleará para indicar el valor del segundo campo que se emplea como PK.
  - El código debe ser inmune a inyección de SQL.

```
--@Author Jorge A. Rodriguez C
--Procedimiento empleado para exportar un el contenido de una columna BLOB a
--un archivo. Se emplea el valor de la llave primaria para localizar el registro.
--Instrucciones:
-- 1. Como SYS crear un directorio virtual en el que se guardará el archivo.
--    Por ejemplo:
--       create or replace directory tmp_dir as '/tmp/bdd';
-- 2. Como SYS otorgar permisos de lectura y escritura al usuario que va
--    ejecutar el script.
--    grant read,write on directory tmp_dir to <my_user>;
-- 3. Suponer que se desea leer el contenido de la columna pdf de la tabla:
--    create table my_table(
--     id number(2,0) ,
--     pdf blob
--   );
-- 4.  Invocar el procedimiento:
--
--  exec save_file_from_blob('TMP_DIR','revista3.pdf','my_table','pdf','id','5');
--
--    parámetro 1:  nombre del directorio virtual creado anteriormente
--    parámetro 2:  nombre del archivo dentro del directorio configurado
--    parámetro 3:  nombre de la tabla que contiene lacolumna a leer
--    parámetro 4:  nombre de la columna tipo blob.
--    parámetro 5:  nombre de la columna que actua como PK.
--    parámetro 6:  valor de la PK que se emplea para localizar al registro cuyo
--                  valor de la columna de tipo blob se va a leer.
```

```
create or replace procedure guarda_blob_en_archivo (
    v_directory_name      in varchar2,
    v_dest_file_name      in varchar2,
    v_table_name          in varchar2,
    v_blob_column_name    in varchar2,
    v_pk_column_name      in varchar2,
    v_pk_column_value     in varchar2) is

    v_file utl_file.FILE_TYPE;
    v_buffer_size number :=32767;
    v_buffer RAW(32767);
    v_blob blob;
    v_blob_length number;
    v_position number;
    v_query varchar2(2000);

    begin
        v_query := 'select '
            ||v_blob_column_name
            ||' into :ph_blob '
            ||' from '
            ||v_table_name
            ||' where '
            ||v_pk_column_name
            ||'='
            ||v_pk_column_value;

        --ejecuta query dinámico
        execute immediate v_query into v_blob;
        v_blob_length := dbms_lob.getlength(v_blob);
        v_position := 1;
        v_file := utl_file.fopen(v_directory_name,v_dest_file_name,'wb',32767);

        --lee el archivos por partes hasta completar
        while v_position < v_blob_length loop
            dbms_lob.read(v_blob,v_buffer_size,v_position,v_buffer);
            UTL_FILE.put_raw(v_file,v_buffer,TRUE);
            v_position := v_position+v_buffer_size;
        end loop;
        utl_file.fclose(v_file);

     -- cierra el archivo en caso de error y relanza la excepción.
    exception when others then
        --cerrar v_file en caso de error.
        if utl_file.is_open(v_file) then
            utl_file.fclose(v_file);
        end if;
        raise;
    end;
    /
show errors;
```

En la nueva versión hacer uso de 'using' para asociar a los placeholders correspondientes a los valores de la PK:

```
--ejecuta query dinámico
if v_pk2_column_name is not null then
  execute immediate v_query into v_blob using
    v_pk1_column_value,v_pk2_column_value;
else
   execute immediate v_query into v_blob using
    v_pk1_column_value;
end if;
```

- **C4. Incluir en el reporte** únicamente las líneas que se marcan en negritas en el código anterior con las correcciones aplicadas.

1.8. PREPARAR LA LECTURA Y CARGA DE ARCHIVOS BINARIOS.

Para realizar la carga de archivos binarios en las tablas de la práctica es necesario realizar algunas configuraciones. Se hará uso de archivos PDFs para simular la carga de datos binarios en columnas tipo BLOB. Estos archivos deberán descargarse de la carpeta de la práctica y se llaman `m_archivo*.pdf`. No deberán emplearse archivos diferentes ya que el validador asume su uso. Realizar las siguientes acciones para preparar la lectura y carga de estos archivos.

- Descargar los archivos PDF, copiarlos en la misma carpeta donde se encuentran los scripts de la práctica.
- Crear un script por integrante `s-04-prepara-carga-archivos.sql`. El script deberá realizar las siguientes acciones:

1. Crear el directorio `/tmp/bdd/p06` a nivel de sistema operativo. Para ello, dentro del mismo programa se puede invocar al comando del sistema operativo `mkdir`. En SQL*Plus es posible ejecutar comandos del sistema operativo ante poniendo el símbolo "!"
2. Copiar los archivos PDFs al directorio `/tmp/bdd/p06`, nuevamente, empleando instrucciones del sistema operativo.
3. Cambiar los permisos de los archivos PDF para que cualquier usuario pueda leerlos.
4. Cambiar los permisos al directorio p06 para que cualquier usuario y grupo pueda leer y escribir (777)

Realizar los siguientes pasos en cada PDB:

5. Conectarse como usuario `sys`, crear un objeto `directory` que apunte a `/tmp/bdd/p06` empleado para hacer la lectura y escritura de archivos PDF. No olvidar que esta instrucción no provoca la creación del directorio realizado anteriormente.
6. Otorgar los privilegios necesarios para que el usuario `editorial_bdd` pueda leer y escribir del directorio anterior.
7. Conectarse con el usuario de la práctica `editorial_bdd` e invocar los procedimientos almacenados creados anteriormente. Es decir, invocar la ejecución de `s-00-carga-blob-en-bd.sql` y `s-00-guarda-blob-en-archivo.sql`
8. Invocar al procedimiento `carga_blob_en_bd` las veces que sean necesarias para realizar la carga de los archivos con base a las siguientes tablas de asignación.

ARTICULO

ARTICULO_ID	PDF
1	m_archivo_1.pdf
2	m_archivo_2.pdf

PAGO_SUSCRIPTOR

NUM_PAGO	SUSCRIPTOR_ID	RECIBO_PAGO
1	1	m_archivo_3.pdf
70	2	m_archivo_4.pdf

9. Generar las consultas SQL para confirmar la carga de los datos. Para la columna tipo `blob` mostrar únicamente su longitud.
10. Realizar la operación inversa, es decir, invocar el procedimiento `guarda_blob_en_archivo` para exportar el archivo PDF en el mismo directorio configurado en el paso 1. Los nombres de los archivos a asignar son:

Nombre original	Nombre para ser exportado en disco
m_archivo_1.pdf	m_export_archivo_1.pdf
m_archivo_2.pdf	m_export_archivo_2.pdf
m_archivo_3.pdf	m_export_archivo_3.pdf
m_archivo_4.pdf	m_export_archivo_4.pdf

11. Ejecutar la instrucción `ls -l` que permita mostrar los archivos PDF exportados de la base de datos.

Ejemplo:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Carga y Export de archivos PDF en el sitio JRCBD_S1

whenever sqlerror exit rollback;

--ruta donde se ubicarán los archivos PDFs
define p_pdf_path='/tmp/bdd/p06'
set verify off

Prompt Limpiando y creando directorio en /tmp/bdd/p06
!rm -rf &p_pdf_path
!mkdir -p &p pdf path
!chmod 777 &p_pdf_path

 -- Se asume que los PDFs se encuentran en el mismo directorio donde se
 -- ejecuta este script.
!cp m_archivo_*.pdf &p_pdf_path
!chmod 755 &p pdf path/m archivo *.pdf

--jrcbd_s1
Prompt conectando a jrcbd_s1 como SYS para crear objetos tipo directory
accept p_sys_password default 'system' prompt 'Password de sys [system]: ' hide
connect sys@jrcbd s1/&p sys password as sysdba

Prompt creando un objeto DIRECTORY para acceder al directorio /tmp/bdd/p06
create or replace directory tmp_dir as '&p_pdf_path';
grant read,write on directory tmp dir to editorial bdd;

--jrcbd_s2

--Seguir el mismo procedimiento para jrcbd_s2
```

```
------------------ Cargando datos  en jrcbd_s1 ----------------------

Prompt conectando a jrcbd_s1 con usuario editorial_bdd para cargar datos binarios
connect editorial_bdd/editorial_bdd@jrcbd_s1

/*
 * En este sitio  se cargarán los siguientes archivos.
 * F_JRC_PAGO_SUSCRIPTOR_1
 * NUM_PAGO  SUSCRIPTOR_ID   RECIBO_PAGO
 * ---------  -------------   ----------
 * 1                      1   m_archivo_3.pdf
 *
 */
Prompt ejecutando procedimientos para hacer import/export de datos BLOB
@s-00-carga-blob-en-bd.sql
@s-00-guarda-blob-en-archivo.sql

Prompt cargando datos binarios en jrcbd s1

begin
  carga_blob_en_bd('TMP_DIR','m_archivo_3.pdf','f_jrc_pago_suscriptor_1',
  'recibo pago','num pago','1','suscriptor id','1');
end;
/

Prompt mostrando el tamaño de los objetos BLOB en BD.

Prompt para f jrc pago suscriptor 1:
select num_pago,suscriptor_id,dbms_lob.getlength(recibo_pago) as longitud
from f jrc pago suscriptor 1;

Prompt salvando datos BLOB  en directorio TMP_DIR

begin
  guarda blob en archivo('TMP DIR','m export archivo 3.pdf',
    'f_jrc_pago_suscriptor_1','recibo_pago','num_pago','1','suscriptor_id','1');
end;
/

Prompt mostrando el contenido del directorio para validar la existencia del archivo.
!ls -l &p pdf path/m export archivo *.pdf


------------------ Cargando datos  en jrcbd_s2 ----------------------

--Seguir el mismo procedimiento para jrcbd_s2

Prompt Listo!
disconnect
```

1.9. VALIDACIÓN DE RESULTADOS.

- De la carpeta de la práctica, obtener los siguientes archivos:
- De la carpeta de la práctica obtener todos los scripts de validación.
- Ejecutar el siguiente script para iniciar la validación de resultados. ***NO*** ejecutar con el usuario Oracle o root.

```
mi_usuario:$ sqlplus /nolog
start s-05-validacion-main.sql
```

- El script solicitará ciertos datos para poder realizar la validación. En caso de existir errores, revisar los mensajes y corregir.
- **C5. Incluir en el reporte** el resultado.
- Si la práctica se hace en equipo, ejecutar el script en la BD de cada integrante. **El reporte deberá incluir una salida por integrante.**

1.10. CONTENIDO DEL REPORTE.

- Elementos comunes. Para mayores detalles revisar el documento de ***instrucciones generales*** para realizar las prácticas.
- C1. Diagrama Jerárquico.
- C2. Investigación modo dedicado/modo compartido
- C3. Código únicamente para `suscriptor` y ejecución del script `s-03-<SID>-consultas.sql`
- C4. Actualización de fragmentos de código para el procedimiento `guarda_blob_en_archivo`
- C5. Salida de ejecución del script de validación `s-05-validacion-main.sql`