PRÁCTICA 5
IMPLEMENTACIÓN DEL ESQUEMA DE FRAGMENTACIÓN

Esta práctica se puede realizar de forma individual o en equipo de máximo 2 personas.

1.1. OBJETIVO.

Implementar el esquema de fragmentación realizado en la práctica anterior empleando las PDBs creadas en la práctica 3.

1.2. CONCEPTOS BÁSICOS PARA TRABAJAR CON SQL *PLUS, LINEAMIENTOS DE PROGRAMACIÓN.

- Revisar los conceptos y los lineamientos para trabajar con los ambientes de bases de datos explicados en el documento ubicado en la carpeta compartida `BD/practicas/practica07/practica7-previo.pdf` Leer el documento en caso de no haberlo realizado anteriormente. No es necesario incluir en el reporte las actividades que en él se indican.
- En el archivo anterior se recomienda personalizar el prompt `SQL>` , en especial para mostrar datos de la conexión. En Bases de datos distribuidas adquiere mayor relevancia en especial cuando se trabaja con varias PDBs. Se recomienda actualizar el archivo `gloging.sql` para incluir la PDB y el usuario de la siguiente manera:

```
define prompt_name=idle
col global_name new_value prompt_name
col global_name noprint
set heading off
set termout off

select lower(sys_context('userenv','current_user')
    ||'@'||sys_context('userenv','con_name')) global_name
from dual;
set sqlprompt '&prompt_name > '
set heading on
set termout on
col global_name print
```

- En este código, se define una variable `prompt_name` que contendrá el valor del prompt a mostrar. El comando `col` define las características de una columna llamada `global_name`. La expresión `new_value prompt_name` en la línea 2 especifica que el valor de la columna será asignado con el valor de la variable `prompt_name`.
- En las líneas 4 y 5 se deshabilita la impresión de los valores de la columna ya que no se requieren en este escenario.
- La función `sys_context` permite obtener los valores de los parámetros de la base de datos. En este caso se obtienen los valores de los parámetros `current_user` (muestra el usuario conectado), y `con_name` (muestra el nombre del contenedor: PDB). Observar que el resultado de la sentencia `select` se asigna a la columna definida anteriormente: `global_name`.
- Se asigna el valor del prompt empleando la expresión `&prompt_name` la cual contiene el valor obtenido en el punto anterior.
- Finalmente se habilita nuevamente la impresión de los valores de las columnas.

1.3. CREACIÓN DE USUARIOS

- Crear un script llamado `s-01-<iniciales>-creacion-usuarios.sql`
- `<iniciales>` corresponde al valor de las iniciales de cada integrante (las mismas empleadas en la práctica anterior).
- El script deberá conectarse a cada PDB y crear un usuario llamado `editorial_bdd`
- El usuario deberá contar con las siguientes configuraciones:
    - Cuota ilimitada en el tablespace USERS.
    - Privilegios para crear sesión, tablas, procedimientos y secuencias (no asignar más privilegios de los necesarios).
    - Password: se recomienda el mismo que el nombre de usuario por simplicidad y propósitos del curso.
- Si la práctica se realiza en equipo se tendrán 2 archivos, de lo contrario se tendrá uno solo.

1.3.1. Ejemplo, script de creación de usuarios.

```
--@Autor:           Jorge A. Rodríguez C
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Creación de usuarios para la máquina pc-jrc.

prompt Conectandose a jrcbd_s1 como usuario SYS
connect sys@jrcbd_s1 as sysdba
prompt creando usuario editorial_bdd
--completar

prompt conectandose a jrcbd_s2 como usuario SYS
--completar
prompt creando usuario editorial_bdd
--completar

prompt Listo
exit
```

- Ejecutar el script.

1.3.2. Ejemplo de ejecución:

```
[jorge@jrc-ora-pc scripts]$ sqlplus /nolog
SQL*Plus: Release 12.1.0.2.0 Production on Mon Mar 19 21:54:16 2018
Copyright (c) 1982, 2014, Oracle.  All rights reserved.

idle> @s-01-jrc-creacion-usuarios.sql

Conectandose a jrcbd_s1 como usuario SYS
Enter password:
Connected.
creando usuario editorial_bdd
User created.
otorgando permisos minimos necesarios
Grant succeeded.

conectandose a jrcbd s2 como usuario SYS
Enter password:
Connected.
User created.
otorgando permisos minimos necesarios
Grant succeeded.
```

- En la línea 1, observar que no se emplea el usuario `oracle` para ejecutar el script. Se recomienda ejecutarlo con el usuario ordinario del sistema operativo y cambiarse al directorio donde se encuentran los scripts de la práctica. No copiar los archivos dentro de los directorios cuyo dueño es el usuario oracle. Por ejemplo, en alguna carpeta a partir de `/u01` o `/home/oracle`.
- En la línea 1, observar que se emplea `/nolog` para entrar a SQL plus sin autenticar. Esto debido a que el script se encarga de conectarse a la PDB correspondiente.
- Finalmente, empleando el comando `start` se ejecuta el script. En Oracle se puede emplear también '@' para invocar la ejecución de un script.
- Para comprobar que la creación de usuarios fue correcta, se puede ejecutar la siguiente instrucción:

```
[jorge@pc-jrc caso2]$ sqlplus editorial bdd/editorial bdd@jrcbd s1
```

- Forma segura ocultando el password:

```
[jorge@pc-jrc caso1]$ sqlplus editorial_bdd@jrcbd_s1
```

- Al entrar en sesión se muestra el prompt personalizado que con el usuario y la PDB actual.

```
editorial_bdd@jrcbd_s1>
```

1.4. IMPLEMENTACIÓN DE FRAGMENTOS – CÓDIGO DDL.

- Por cada integrante y por cada nodo se deberá crear un script SQL con la siguiente nomenclatura: `s-02-<iniciales>-n{1|2}-ddl.sql`
- El script contendrá la definición de los fragmentos que serán creados en cada PDB. Se tendrán 2 archivos si la práctica es individual y 4 si es en equipo.
- Asignar un nombre adecuado y entendible a cada restricción de referencia. El contar con un nombre claro permitirá identificar fácilmente posibles errores de integridad referencial. Por convención el nombre de la restricción de referencia es:

`<nombre_tabla_hija>_<nombre_campo_fk>`

- En Oracle los nombres de los objetos no deben sobrepasar los 30 caracteres. Para evitar este detalle, omitir los datos del número de fragmento e iniciales. Por ejemplo, una restricción de referencia `pais -> oficina` el nombre de la restricción será: `oficina_pais_id_fk`
- Para los nombres de las restricciones de llave primaria emplear la convención `<nombre_tabla_pk>`
- Abreviar únicamente cuando se exceda de los 30 caracteres.

1.4.1. Creación de los objetos en cada PDB.

- Crear un archivo por integrante llamado `s-03-<iniciales>-main-ddl.sql`
- El archivo deberá conectarse a cada PDB y ejecutar los scripts de creación de objetos. Se obtendrá 1 script si la práctica es individual y 2 si se realiza en equipo.

1.5. EXPLORACIÓN DEL DICCIONARIO DE DATOS.

Ejecutar las siguientes sentencias que muestran las características de los objetos creados en cada PDB. Las sentencias se deben ejecutar en ambos PDBs por cada integrante.

1.5.1. Lista de fragmentos.

- Generar un archivo llamado `s-04-<iniciales>-consulta-fragmentos.sql`
- El script deberá conectarse a cada PDB y deberá mostrar la lista de fragmentos (tablas) que fueron creadas. **C1. Incluir en el reporte** el contenido del script y el resultado obtenido. Ordenar la lista por el nombre del fragmento.
- Si la práctica es en equipo, incluir el contenido y salida de un solo integrante.

Ejemplo.

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:    Consulta de fragmentos creados en jrc-pc
Prompt Conectando a  S1 - jrcbd s1
connect editorial_bdd/editorial_bdd@jrcbd_s1
Prompt mostrando lista de fragmentos

--completar

Prompt Conectando a  S1 - jrcbd s2
connect editorial_bdd/editorial_bdd@jrcbd_s2
Prompt mostrando lista de fragmentos

--completar

Prompt Listo!
exit
```

1.5.2. Referencias de integridad.

- Generar un archivo llamado `s-05-<iniciales>-consulta-restricciones.sql`
- El script deberá conectarse a cada PDB y deberá mostrar las relaciones de referencia que existen entre los fragmentos de cada PDB. Los campos a mostrar son: Nombre de la tabla hija, nombre de la restricción de referencia, nombre de la tabla padre y tipo de restricción.
- Ordenar la lista por el nombre de la tabla padre
- Tips:
  - En Oracle la vista `user_constraints` contiene los datos referentes a las restricciones de una tabla.
  - El atributo `constraint_type` define el tipo de restricción, que en este caso solo interesan las restricciones de referencia: 'R'
  - El campo `r_constraint_name` contiene el nombre de la restricción de la llave primaria de la tabla padre. Este campo puede ser empleado como condición de `join` para obtener el nombre de la tabla padre.
- Si la práctica es en equipo, existirán 2 scripts.

Ejemplo:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Consulta de restricciones de referencia en jrc-pc

Prompt Conectando a  S1 - jrcbd_s1
connect editorial_bdd/editorial_bdd@jrcbd_s1
--ejecuta la misma consulta en ambas pdbs
@s-05-consulta-restricciones.sql

Prompt Conectando a  S2 - jrcbd_s2
connect editorial_bdd/editorial_bdd@jrcbd_s1
--ejecuta la misma consulta en ambas pdbs
@s-05-consulta-restricciones.sql

Prompt Listo!
exit
```

- En este ejemplo se ha creado un script adicional Debido a que la consulta es idéntica en cada sitio, es posible reutilizarlo para evitar repetir el código en cada nodo.

Ejemplo:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Consulta de restricciones de referencia

Prompt mostrando lista de restricciones de referencia

col tabla_padre format A30
col tabla_hija format A30
col nombre_restriccion format A30
set linesize 200

-- select ...
```

- Observar el comando `col` empleado para dar formato a las columnas y poder visualizar los resultados correctamente.
- Notar que el código del script es exactamente el mismo a ejecutar en los 2 nodos.
- Para evitar ejecutar manualmente el script e ambos nodos o para evitar duplicar el código en cada nodo, se recomienda crear un script adicional `s-05-<iniciales>-consulta-restricciones-main.sql` que se conecte a cada sitio e invoque al script anterior:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Consulta de restricciones de referencia en jrc-pc

Prompt Conectando a  S1 - jrcbd_s1
connect bancos_bdd/editorial_bdd@jrcbd_s1
--ejecuta la misma consulta en ambas pdbs
@s-05-jrc-consulta-restricciones.sql

Prompt Conectando a  S2 - jrcbd_s2
connect bancos bdd/editorial bdd@jrcbd s1
--ejecuta la misma consulta en ambas pdbs
@s-05-jrc-consulta-restricciones.sql

Prompt Listo!
exit
```

- **C2. Incluir en el reporte** el contenido del script `s-05-<iniciales>-consulta-restricciones.sql` y el resultado de ejecutar `s-05-<iniciales>-consulta-restricciones-main.sql`

1.6. CARGA INICIAL DE DATOS.

Debido a que hasta este punto no se cuenta con ningún nivel de transparencia implementado, la carga y consulta de datos se debe realizar de forma manual en cada PDB. La implementación de los distintos tipos y niveles de transparencia se implementarán en prácticas posteriores.

- Crear un script `s-06-<iniciales>-carga.sql` El script deberá contener las sentencias `insert` necesarias a ejecutar en cada PDB para cargar la siguiente información.
- El script deberá conectarse a cada PDB y realizar las inserciones de forma manual considerando el esquema de fragmentación.

PAIS

PAIS_ID	CLAVE	NOMBRE	REGION
1	MX	MEXICO	A
2	JAP	JAPON	B

SUSCRIPTOR

SUSCRIPTOR_ID	NOMBRE	AP_PATERNO	AP_MATERNO	FECHA	PAIS_ID	NUM_TARJETA
1	OMAR	LOPEZ	MENDEZ	01/01/2017	1	5420900754028724
2	LALO	KIM	LUNA	01/01/2016	2	5800807976301529
3	LUCY	ZAMORA	PEREZ	01/01/2015	2	6202870129036021

ARTICULO

ARTICULO_ID	TITULO	RESUMEN	TEXTO	PDF
1	LOS SISMOS	Estudio y origen de los sismos	Texto de ejemplo para el artículo	Invocar a la función `empty_blob()` En prácticas posteriores se verá el manejo de datos binarios en BDD.
2	FAUNA MARINA	Estudio de la fauna marina de México.	Texto de ejemplo para el artículo	`empty_blob()`

REVISTA

REVISTA_ID	FOLIO	TITULO	FECHA	PDF	REV ADICIONAL
1	90001	Premier	01/03/2017	`empty_blob()`	Null
2	90002	TI en la UNAM	01/09/2017	`empty_blob()`	1

ARTICULO_REVISTA

ARTICULO_REVISTA_ID	ARTICULO_ID	REVISTA_ID	FECHA	CALIFICACION
1	1	1	01/02/2017	9
2	2	2	01/08/2017	10

PAGO_SUSCRIPTOR

NUM_PAGO	SUSCRIPTOR_ID	FECHA_PAGO	IMPORTE	RECIBO_PAGO
1	1	01/02/2017	989.67	`empty_blob()`
70	2	01/08/2017	1000.55	`empty_blob()`

Ejemplo:

El siguiente pseudo código muestra la estructura recomendada de este script.

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Archivo de carga inicial en jrc-pc

Prompt Conectando a  S1 - jrcbd_s1
connect editorial bdd/editorial bdd@jrcbd s1

--si ocurre un error, la ejecución se detiene.
whenever sqlerror exit rollback;

Prompt limpiando.
--delete from ...

Prompt Cargando datos
--insert into ..

--hacer commit al terminar
commit;

Prompt Conectando a  S2 - jrcbd_s2
connect editorial_bdd/editorial_bdd@jrcbd_s2

--si ocurre un error, la ejecución se detiene.
whenever sqlerror exit rollback;

Prompt limpiando.
--delete from ...

Prompt Cargando datos
--insert into ..

--hacer commit al terminar
commit;
Prompt Listo!
exit
```

- Observar que se agregan sentencias `delete`. Esto permite ejecutar el script N veces sin provocar errores de duplicidad ya que primero se realiza una limpieza de las tablas y posteriormente se realiza la carga inicial. Esto funciona correctamente ya que las tablas se encuentran vacías.

1.6.1. Conteo de registros.

- Generar una sentencia SQL que muestre el conteo de todos los registros en cada PDB. Generar un script llamado `s-07-<iniciales>-consulta-datos.sql`
- Esta consulta permitirá validar de forma manual el complimiento de las 3 reglas de fragmentación.
- El script deberá conectarse a ambas PDBs y generar una consulta similar a la siguiente. Si la práctica es en equipo, se requieren 2 archivos.

1.6.1.1. Ejemplo, conteo de registros.

```
idle> start s-07-jrc-consulta-datos.sql
conectando a sitio s1
Connected.
Realizando conteo de registros

PAIS_1      SUSCRIPTOR_1  SUSCRIPTOR_2 SUSCRIPTOR_3 ARTICULO_1  REVISTA_1  REVISTA_2  PAGO_SUSCRIPTOR_1
---------- ------------  ------------ ------------ ----------  ---------- ----------  -----------------
1          3             1            1            2           2          1           1

conectando a sitio s2
Connected.
Realizando conteo de registros

. . .
. . .
. . .
```

- Las consultas deben emplear los mismos alias y orden de consultas mostrados
- Se deberá obtener un solo registro por cada PDB
- Tip: emplear subqueries en la cláusula SELECT.
- **C3. Incluir en el reporte** el código y el resultado de ejecución. Si la práctica es en equipo, incluir el contenido y salida de un solo integrante.

1.7. VALIDACIÓN DE RESULTADOS.

- De la carpeta de la práctica, obtener los siguientes archivos:

```
s-00-header-validacion.sql
s-00-funciones-validacion.plb
s-08-validacion-main.sql
s-08-validacion-pdb-create.plb
s-08-validacion-pdb-execute.sql
```

- Ejecutar el archivo `s-08-validacion-main.sql` El script solicitará ciertos datos para poder realizar la validación. En caso de existir errores, revisar los mensajes y corregir. **C4. 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.**

1.8. CONTENIDO DEL REPORTE.

- Elementos comunes. Para mayores detalles revisar el documento de ***instrucciones generales*** para realizar las prácticas.
- **C1**. Código y salida del script `s-04-<iniciales>-consulta-fragmentos.sql`
- **C2.** Código y salida del script `s-05-<iniciales>-consulta-restricciones-main.sql` y `s-05-<iniciales>-consulta-restricciones.sql`
- **C3.** Código y salida del script `s-07-<iniciales>-consultas.sql`
- **C4.** Salida de ejecución del script de validación `s-08-validacion-main.sql`