PRACTICA COMPLEMENTARIA 13
PROGRAMACIÓN CON SQL PARTE 2

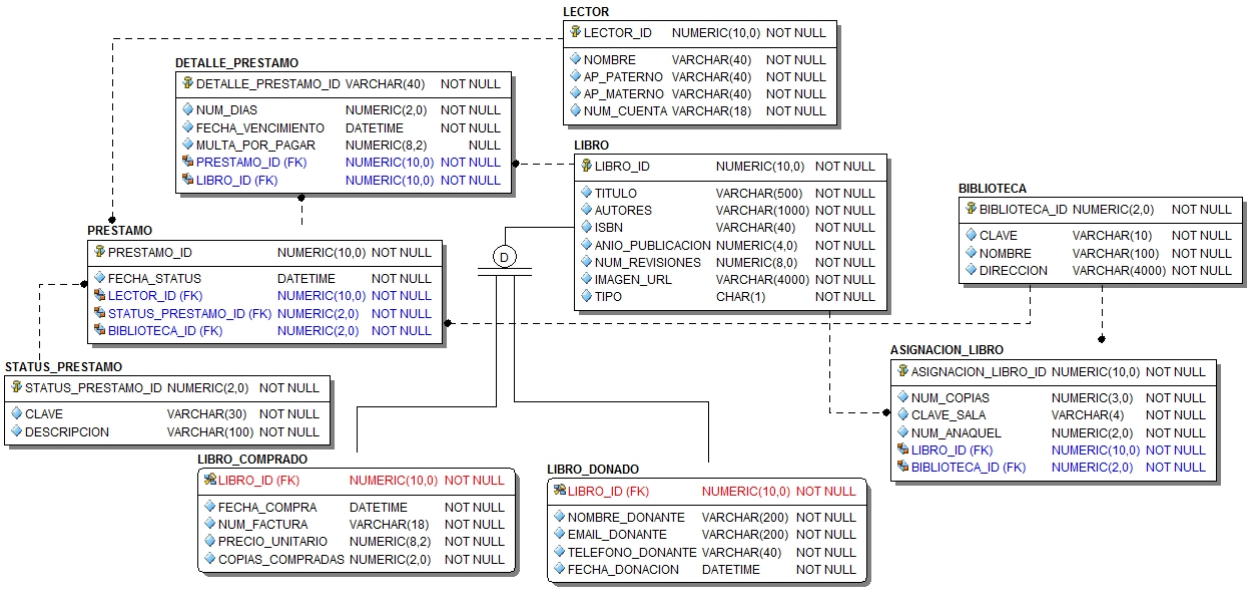
El reporte se entrega en equipos formado por máximo 2 integrantes.

1.1. OBJETIVO:

Poner en práctica los conceptos de programación PL/SQL para realizar el correcto manejo de cursores, procedimientos almacenados y objetos LOB.

1.2. PREPARACIÓN DE LA BASE DE DATOS.

Considerar el siguiente modelo relacional de una base de datos empleada para administrar bibliotecas.



1.2.1. Creación del usuario.

- Crear un script llamado `s-01-creacion-usuario.sql` El script deberá contener el código necesario para crear un usuario llamado `<iniciales>_p1302_biblio`, donde: `<iniciales>` corresponde con las primeras 4 letras de los integrantes del equipo: letra inicial de los apellidos del primer integrante, letra inicial de los apellidos del segundo integrante. Si es individual, emplear las iniciales del nombre completo. No olvidar agregar el encabezado al archivo.
- Asignarle los privilegios necesarios para poder crear los objetos del diagrama relacional. Adicionalmente asignarle el privilegio `create procedure`.

1.2.1. Creación de objetos

- De la carpeta compartida correspondiente a esta práctica, descargar un script llamado `s-02-biblio-ddl.sql` el cual será el encargado de crear los objetos del modelo relacional anterior.

1.2.1. Carga inicial

Para cada tabla se proporciona un script de carga inicial que puede ser descargado del sitio Web Mockaroo (generador de datos SQL). El Script se puede descargar directamente del navegador web empleando la dirección http, o a través de una terminal empleando el comando `curl` que será encargado de obtener los datos y guardarlos en el script SQL correspondiente. Se sugiere abrir una terminal, colocarse en el directorio donde se encuentran todos los scripts SQL y ejecutar las siguientes instrucciones.

```
curl "https://api.mockaroo.com/api/94e76f80?count=1000&key=b71bb7e0" > "s-03-lector.sql"
curl "https://api.mockaroo.com/api/dc41b390?count=56&key=b71bb7e0" > "s-03-biblioteca.sql"
curl "https://api.mockaroo.com/api/aae7bd80?count=100&key=b71bb7e0" > "s-03-libro.sql"
curl "https://api.mockaroo.com/api/6dc64cd0?count=80&key=b71bb7e0" > "s-03-libro-comprado.sql"
curl "https://api.mockaroo.com/api/4272bff0?count=20&key=b71bb7e0" > "s-03-libro-donado.sql"
curl "https://api.mockaroo.com/api/75dc3b20?count=1000&key=b71bb7e0" > "s-03-asignacion-libro.sql"
curl "https://api.mockaroo.com/api/eb68dec0?count=6&key=b71bb7e0" > "s-03-status-prestamo.sql"
curl "https://api.mockaroo.com/api/b97f9cb0?count=1000&key=b71bb7e0" > "s-03-prestamo.sql"
curl "https://api.mockaroo.com/api/fa0d7e80?count=1000&key=b71bb7e0" > "s-03-detalle-prestamo.sql"
```

1.2.1. Ejecución de scripts.

- Crear un script `s-00-main.sql` que invoque a cada uno de los scripts anteriores empleando los usuarios correspondientes. Ejecutar y verificar que no existan errores.

Ejemplo:

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Archivo principal

--si ocurre un error, se hace rollback de los datos y
--se sale de SQL *Plus
whenever  sqlerror exit rollback

Prompt creando usuario jrc_p1302_biblio
@s-01-creacion-usuario.sql

Prompt conectando como usuario jrc_p1302_biblio
connect jrc_p1302_biblio

Prompt creando objetos
@s-02-biblio-ddl.sql

set define off

!curl "https://api.mockaroo.com/api/94e76f80?count=1000&key=b71bb7e0" > "s-03-lector.sql"
!curl "https://api.mockaroo.com/api/dc41b390?count=56&key=b71bb7e0" > "s-03-biblioteca.sql"
!curl "https://api.mockaroo.com/api/aae7bd80?count=100&key=b71bb7e0" > "s-03-libro.sql"
!curl "https://api.mockaroo.com/api/6dc64cd0?count=80&key=b71bb7e0" > "s-03-libro-comprado.sql"
!curl "https://api.mockaroo.com/api/4272bff0?count=20&key=b71bb7e0" > "s-03-libro-donado.sql"
!curl "https://api.mockaroo.com/api/75dc3b20?count=1000&key=b71bb7e0" > "s-03-asignacion-libro.sql"
!curl "https://api.mockaroo.com/api/eb68dec0?count=6&key=b71bb7e0" > "s-03-status-prestamo.sql"
!curl "https://api.mockaroo.com/api/b97f9cb0?count=1000&key=b71bb7e0" > "s-03-prestamo.sql"
!curl "https://api.mockaroo.com/api/fa0d7e80?count=1000&key=b71bb7e0" > "s-03-detalle-prestamo.sql"

Prompt realizando la carga de datos
@s-03-lector.sql
@s-03-biblioteca.sql
@s-03-libro.sql
@s-03-libro-comprado.sql
@s-03-libro-donado.sql
@s-03-asignacion-libro.sql
@s-03-status-prestamo.sql
@s-03-prestamo.sql
@s-03-detalle-prestamo.sql

set define on

Prompt confirmando cambios
commit;


--Si se encuentra un error, no se sale de SQL *Plus
--no se hace commit ni rollback, es decir, se
--regresa al estado original.
whenever sqlerror continue none

Prompt Listo!
```

1.3. EJERCICIOS DE PROGRAMACIÓN

No olvidar formatear correctamente el código empleando los tipos de fuentes y tamaños recomendados en prácticas anteriores.

1.3.1. Ejercicio 1

Crear un script `s-04-ejercicio-corrige-status.sql`. Generar un procedimiento almacenado con la siguiente estructura:

```
create or replace procedure p_corrige_status (p_num_expirado out number ,
       p_num_con_multa out number, p_num_en_curso out number) is
```

El procedimiento deberá realizar las siguientes acciones:

- Para cada uno de los prestamos registrados en la base de datos se necesita validar si el valor de `status_prestamo_id` es el correcto.
- Para verificar si el status asignado es el correcto se deberán aplicar las siguientes reglas:
    - Emplear una tabla temporal llamada `t_detalle_prestamo` que contenta la misma estructura que `detalle_prestamo`. Los datos de la tabla temporal se deben preservar, aunque se haga `commit`.

```
create global temporary table t_detalle_prestamo(
 detalle_prestamo_id       varchar2(40)     not null,
    num_dias               number(2, 0)     not null,
    fecha_vencimiento      timestamp(6)     not null,
    multa_por_pagar        number(8, 2),
    prestamo_id            number(10, 0)    not null,
    libro_id               number(10, 0)    not null,
    constraint t_detalle_prestamo_pk primary key (detalle_prestamo_id)
) on commit preserve rows;
```

- El código de esta tabla puede ser incluido en el script `s-00-main.sql`
- Para cada préstamo se deberá realizar una consulta en `detalle_prestamo` y recuperar el detalle de cada préstamo (lista de libros que fueron prestados). Estos registros deberán ser insertados en la tabla temporal.
- Posterior al a inserción, se deberán aplicar las siguientes reglas para validar el status asignado al préstamo:
    - Si alguno de los libros del préstamo tiene un importe de multa mayor a cero, el status del préstamo debe actualizarse a 5 (CON MULTA). No olvidar actualizar la fecha de status con la fecha del sistema.
    - En caso de no contar con multas asignadas, se considera la fecha de vencimiento con respecto a la fecha actual.
        - Al menos un libro tiene la fecha de vencimiento expirada: El status debe actualizarse a 4 (EXPIRADO).
        - En caso contrario, el status debe actualizarse a 2 (EN CURSO).
    - Al final de cada iteración se deberán eliminar los datos de la tabla temporal para optimizar el uso de memoria.
- Notar que el procedimiento acepta 3 parámetros de salida en donde se deberá actualizar el número de préstamos que actualizaron su status a CON MULTA, EXPIRADO o EN CURSO respectivamente.
- Para verificar el funcionamiento de este procedimiento, crear un programa anónimo PL/SQL que invoque al procedimiento y genere una salida similar a la siguiente:

```
creando bloque pl/sql para ejecutar Procedimiento p_corrige_status
Iniciando correccion de status
============= Resultados: ==============
Cambios a Expirados: x
Cambios a Multados:  x
Cambios a En curso:  x
Total de cambios:    x
```

- EL programa anónimo puede incluirse en el mismo script, justo después de la definición del procedimiento.

C1. Incluir en el reporte únicamente el código del procedimiento y .la salida del bloque PL/SQL

1.3.2. Ejercicio 2

Generar un procedimiento almacenado en un script llamado `s-05-ejercicio-consulta-bibliotecas.sql` El script deberá definir un procedimiento llamado `p_consulta_bibliotecas`. Deberá aceptar como parámetro de entrada la clave de una biblioteca. El procedimiento deberá realizar una consulta de todas las bibliotecas y de sus libros asignados con base a la clave proporcionada. Las columnas que debe llevar la consulta son:

- Identificador de la biblioteca
- Clave
- Nombre
- Número total de libros que tiene asignado cada biblioteca.
- Identificador de libro asignado a la biblioteca.
- ISBN del libro
- Título del libro
- Tipo del libro
- Precio unitario (en caso que el libro sea comprado)
- Nombre del donante (en caso que el libro sea donado)
- Total de préstamos. Corresponde con el número de veces que el libro ha sido prestado sin importar la biblioteca.

Los datos obtenidos deberán almacenarse en una tabla temporal con la siguiente estructura:

```
create global temporary table t_reporte_biblioteca (
 biblioteca_id number (10,0) not null,
 clave varchar2(10) not null,
 nombre varchar2(4000) not null,
 total_libros_asignados number(10,0) not null,
 libro_id number(10,0) not null,
 isbn varchar2(40) not null,
 titulo varchar2(500) not null,
 tipo char not null,
 precio_unitario number(8,2),
 nombre_donante varchar2(200),
 total_prestamos number(10,0) not null
) on commit preserve rows;
```

- La definición de esta tabla puede agregarse en el archivo main.

Generar un programa anónimo PL/SQL empleado para invocar el procedimiento anterior. El programa deberá seleccionar 5 bibliotecas al azar e invocar al procedimiento.
Nota: al término de cada consulta, el programa deberá eliminar el contenido de la tabla temporal para no mezclar los datos con los de la consulta anterior.
Deberá generar una salida similar a la siguiente:

```
=========Generando reporte de bibliotecas ==========
Registros exportados para biblioteca: B07383   -   xx
Registros exportados para biblioteca: B02063   -   xx
Registros exportados para biblioteca: B04276   -   xx
Registros exportados para biblioteca: B74376   -   xx
Registros exportados para biblioteca: B20724   -   xx
Haciendo commit
```

C2. Incluir en el reporte únicamente el código del procedimiento y la salida del programa anónimo.

1.3.3. Ejercicio 3

En este ejercicio se revisará el manejo de datos BLOB.
Observar que la tabla `libro` define un campo llamado `imagen_url`. Los valores de este atributo contienen direcciones válidas en las que se encuentran las imágenes de los libros. Haciendo uso de esta característica, la idea de este ejercicio es descargar la imagen, guardarlas en un directorio y posteriormente, cargarlas a la base de datos. Para ello se deberá seguir el siguiente procedimiento:

- Las imágenes deberán ser almacenadas en una nueva tabla llamada `imagen_libro` que cuenta con la siguiente estructura:

```
create table  libro_imagen(
 libro_id number(10,0) not null,
 imagen_url varchar2(4000) not null,
 nombre_archivo varchar2(1000) not null,
 comando varchar2(4000) not null,
 imagen blob
);
```

- Agregar la definición de esta tabla en el archivo main.
- Crear un script llamado `s-06-ejercicio-genera-shell-script.sql`. El archivo define un bloque anónimo y obtiene una muestra aleatoria de aproximadamente 10 libros cuyas imágenes serán obtenidas y guardadas en la tabla anterior.
- Para cada libro seleccionado el script realiza las siguientes acciones:
    - Genera un nuevo registro en la tabla `libro_imagen`.

- En el campo `imagen_url` se guarda la dirección en internet donde se encuentra la imagen.
- En el campo `nombre_archivo` se guarda el nombre del archivo de la imagen que será descargado. Todas las imágenes serán descargadas en una carpeta llamada imágenes que se creará en el mismo directorio donde se encuentran los scripts.
- En el campo comando, se guarda la instrucción a ejecutar empleando el comando `curl` y será el encargado de descargar la imagen. Por ejemplo: `curl https://images.gr-assets.com/books/1361039443m/41865.jpg > imagenes/img-41865.jpg`

- Como siguiente paso, el script generará un archivo llamado `s-06-ejercicio-descarga_imagenes.sh`. En este script se encuentra la lista de imágenes que se van a descargar empleando el comando `curl`.
- El siguiente paso es ejecutar el script s`-06-ejercicio-descarga_imagenes.sh` para obtener las imágenes de internet.
- Al terminar su ejecución se deberá contar con un directorio imágenes y dentro de este, se encontrarán las imágenes de los libros (revisar la carpeta para confirmar).

Ejemplo: Revisar y modificar el script con el usuario correspondiente.

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Script encargado de obtener imagenes de los libros.

Prompt conectando como usuario jrc_p1302_biblio
connect jrc_p1302_biblio
set serveroutput on
Prompt generando shell script para descargar imagenes de libros.

declare
 cursor cur_muestra_imagenes is
 select libro_id, imagen_url
 from libro sample(10);
 v_comando varchar2(4000);
 v_nombre_archivo varchar2(1000);
begin
 delete from libro_imagen;
 for r in cur_muestra_imagenes loop
  v_nombre_archivo := 'img-'||substr(r.imagen_url,instr(r.imagen_url,'/',-1,1)+1);
  --esta variable contiene el comando a ejecutar para obtener la imagen
  -- Ejemplo: curl https://images.gr-assets.com/books/1361039443m/41865.jpg > imagenes/img-41865.jpg
  -- las imagenes serán almacenadas en un directorio llamado imagenes.
  v_comando := 'curl '||r.imagen_url||' > imagenes/'||v_nombre_archivo;
  insert into libro_imagen(libro_id,imagen_url,nombre_archivo,comando,imagen)
  values(r.libro_id,r.imagen_url,v_nombre_archivo,v_comando,empty_blob());
 end loop;
end;
/
Prompt haciendo commit
commit;

set pagesize 100
set linesize 1000
set echo off;
set feedback off;
set heading off;
!mkdir imagenes

--en este archivo se guardará el contenido de la columna comando que contiene las instrucciones
-- para descargar las imágenes.  Se hace uso de un spool.

spool "s-06-ejercicio-descarga_imagenes.sh"

select comando from libro_imagen;

spool off;
set echo on
set feedback on
set heading on

Prompt Shell script generado, obteniendo las imagenes
!rm -rf imagenes
!mkdir -p imagenes
!sh s-06-ejercicio-descarga_imagenes.sh

Prompt Listo!
exit
```

Una vez que las imágenes han sido descargadas, el siguiente paso es crear un programa PL/SQL que lea los archivos y su contenido sea guardado en la columna `imagen` (blob) de la tabla `libro_imagen`.

- Crear un script llamado `s-07-actualiza-imagen.sql`. El archivo contendrá la definición de un procedimiento llamado `p_actualiza_imagen` Este procedimiento deberá realizar las siguientes acciones:
  - Se realiza una conexión como usuario sys. El objetivo es crear un objeto tipo directory llamado `DATA_DIR` y que apunta a `/tmp/bd`.
  - Este objeto será empleado para conocer la ubicación de las imágenes dentro del programa PL/SQL.
  - Se otorgan privilegios para que el usuario de la práctica pueda realizar operaciones de lectura y escritura.
  - Se realiza ahora una conexión empleando el usuario de la práctica para poder crear el procedimiento almacenado.
  - En el procedimiento, se realiza la iteración haciendo uso de un curso sobre todos los registros contenidos en la tabla `libro_imagen`
  - Por cara registro se deberá leer el archivo que contiene la imagen del libro.
  - Haciendo uso de la función `loadblobfromfile`, se realiza la carga del contenido del archivo y se actualiza en el campo `imagen` de la tabla.
- Observar las instrucciones posteriores a la creación del procedimiento. Se ejecuta una serie de comandos a nivel del sistema operativo para copiar las imágenes descargadas hacia el directorio /tmp/bd.
- El objetivo del paso anterior es garantizar que el procedimiento encontrará a los archivos en la ruta asociada al objeto directory `DATA_DIR`.
- Al final, el script realiza una consulta de la tabla `libro_imagen` para confirmar que las imágenes fueron cargadas en la BD. Se consulta la longitud del campo BLOB para confirmar que existen datos en cada registro.

C3. Incluir en el reporte únicamente la salida de ejecución de este script.

Ejemplo: (Revisar y modificar el nombre de usuario).

```
--@Autor:           Jorge Rodriguez
--@Fecha creación:  dd/mm/yyyy
--@Descripción:     Script encargado de actualizar imagenes en la BD.

--whenever sqlerror exit rollback
set serveroutput on
Prompt Actualizando imagenes del libros
Prompt conectando como sys
connect sys as sysdba

Prompt creando objeto DATA_DIR
create or replace directory data_dir as '/tmp/bd';

grant read,write on directory data_dir to jrc_p1302_biblio;

Prompt conectando como usuario jrc_p1302_biblio
connect jrc_p1302_biblio

Prompt creando procedimiento para actualizar imagenes.

create or replace procedure p_actualiza_imagen  is
  v_bfile bfile;
  v_src_offset number := 1;
  v_dest_offset number := 1;
  v_dest_blob blob;
  v_src_length number;
  v_dest_length number;
  v_nombre_archivo varchar2(1000);

  cursor cur_libro_imagen is
    select libro_id,imagen,nombre_archivo
    from libro_imagen;

begin

  for r in cur_libro_imagen loop
    v_src_offset  := 1;
    v_dest_offset := 1;
    dbms_output.put_line('cargando imagen para '||r.nombre_archivo);
    v_bfile := bfilename('DATA_DIR', r.nombre_archivo);

    if dbms_lob.fileexists(v_bfile) = 1 and not dbms_lob.isopen(v_bfile) = 1 then dbms_lob.open(
      v_bfile, dbms_lob.lob_readonly);
    else raise_application_error(-20001, 'El archivo '
      || r.nombre_archivo
      ||' no existe en el directorio DATA_DIR'
      || ' o el archivo esta abierto');
    end if;
    select imagen into v_dest_blob
    from libro_imagen
    where libro_id = r.libro_id
    for update;
```

```
    dbms_lob.loadblobfromfile(
      dest_lob    => v_dest_blob,
      src_bfile   => v_bfile,
      amount      => dbms_lob.getlength(v_bfile),
      dest_offset => v_dest_offset,
      src_offset  => v_src_offset);

    dbms_lob.close(v_bfile);
    v_src_length := dbms_lob.getlength(v_bfile);
    v_dest_length := dbms_lob.getlength(v_dest_blob);

    if v_src_length = v_dest_length then
      dbms_output.put_line('Escritura correcta, bytes escritos: '
        || v_src_length);
    else raise_application_error(-20002, 'Error al escribir datos.\n'
      || ' Se esperaba escribir '
      || v_src_length
      || ' Pero solo se escribio '
      || v_dest_length);
    end if;
  end loop;
end;
/
show errors

Prompt copiando imagenes
!rm -rf /tmp/bd
!mkdir -p /tmp/bd
!chmod 777 /tmp/bd
!cp imagenes/img-* /tmp/bd
!chmod 755 /tmp/bd/img-*

Prompt invocando procedimiento
exec p_actualiza_imagen

commit;
Prompt Mostrando resultados
col nombre_archivo format a30
select libro_id,nombre_archivo,dbms_lob.getlength(imagen) as longitud_imagen
from libro_imagen;
Prompt Listo!
```

1.4. CONTENIDO DEL REPORTE.

Para realizar la evaluación de la práctica se deberá anexar en la última página del reporte la rúbrica correspondiente:

- Rúbrica para el grupo de laboratorio.
- Rúbrica para el grupo de teoría plan 2010
- Rúbrica para grupo de teoría plan 2016 inscritos en otro grupo de laboratorio

Imprimir alguna de las siguientes páginas de este documento e incluirla en el reporte. Prácticas que no incluyan esta tabla se considerarán como prácticas no entregadas. La rúbrica permite conocer a detalle los criterios empleados para asignar la calificación final.

PRACTICA 13

Rubrica para grupo del laboratorio

Contenido	Puntaje Obtenido		Observaciones
Carátula *	0P	**5P**	
objetivos e Introducción *	0P	**5P**	
Actividades Práctica complementaria			
C1. Código del procedimiento y salida de ejecución. Ejercicio 1	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**25P**	En esta ocasión no existe validador.
C2. Código del procedimiento y salida de ejecución. Ejercicio 2	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**25P**	
C3. Únicamente salida del script de carga de imágenes. Ejercicio 3.	Salida incorrecta 0P	**30P**	
Conclusiones, comentarios, recomendaciones *	0P	**5P**	
Bibliografía *	0P	**5P**	

* Ver Rubrica general de prácticas para mayores detalles en cuanto a los requisitos que debe cumplir el elemento de evaluación y los puntajes asignados.

PRACTICA 13

Rubrica para grupo de teoría plan 2010

Contenido	Puntaje Obtenido		Observaciones
Carátula *	0P	**5P**	
objetivos e Introducción *	0P	**5P**	
Actividades Práctica complementaria			
C1. Código del procedimiento y salida de ejecución. Ejercicio 1	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**25P**	En esta ocasión no existe validador.
C2. Código del procedimiento y salida de ejecución. Ejercicio 2	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**25P**	
C3. Únicamente salida del script de carga de imágenes. Ejercicio 3.	Salida incorrecta 0P	**30P**	
Conclusiones, comentarios, recomendaciones *	0P	**5P**	
Bibliografía *	0P	**5P**	

* Ver Rubrica general de prácticas para mayores detalles en cuanto a los requisitos que debe cumplir el elemento de evaluación y los puntajes asignados.

PRACTICA 13

Rubrica para grupo de teoría plan 2016 inscritos en otro grupo de laboratorio

Contenido	Puntaje Obtenido		Observaciones
Carátula *	0P	**5P**	
Actividades Práctica complementaria			
C1. Código del procedimiento y salida de ejecución. Ejercicio 1	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**30P**	En esta ocasión no existe validador.
C2. Código del procedimiento y salida de ejecución. Ejercicio 2	0P Formato incorrecto, código incompleto, errores de compilación o salida incorrecta.	**30P**	
C3. Únicamente salida del script de carga de imágenes. Ejercicio 3.	Salida incorrecta 0P	**30P**	
Conclusiones, comentarios, recomendaciones *	0P	**5P**	

* Ver Rubrica general de prácticas para mayores detalles en cuanto a los requisitos que debe cumplir el elemento de evaluación y los puntajes asignados.