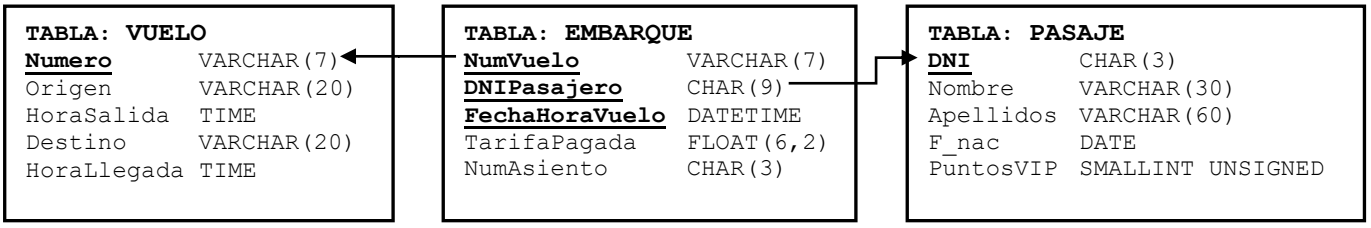
Módulo: BASES DE DATOS
I.E.S.: Aguadulce **C.F.G.S. : DAW (Desarrollo de Aplicaciones Web)**
Fecha: Miércoles 2 de Junio de 2021 **Hora de Comienzo: 15:30h**
Examen: SEGUNDO PARCIAL **Duración: 2:30 horas**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

SEGUNDO PARCIAL

PARTE PRÁCTICA (7 PUNTOS)

Una compañía aérea quiere informatizar su servicio de embarque del pasaje en sus vuelos y para ello tenemos entre las diferentes tablas de la base de datos las siguientes:



MUY IMPORTANTE: LEE las siguientes consideraciones antes de realizar las consultas:

- **NumVuelo** es clave foránea que referencia a la clave primaria de VUELO.
- **DNIPasajero** es clave foránea que referencia a la clave primaria de PASAJE.
- Las claves primarias de cada tabla están subrayadas y en negrita.

1.- Realizar las siguientes consultas para que funcionen correctamente con MySQL (1,5 p.):

a) Obtener un listado de todos los vuelos con destino Alicante. **(0,2 p.)**
b) Obtener por cada vuelo cuyo origen sea Madrid, el número de vuelo, la fecha y hora del vuelo y la cantidad total de pasajeros embarcados ordenando el resultado de mayor a menor cantidad de pasaje. **(0,4 p.)**
c) Obtener todos los datos del pasaje que haya embarcado en alguno de los vuelos con la menor tarifa pagada de todos. **(0,4 p.)**
d) Obtener un listado con el número de vuelo, el origen, el destino, la duración del vuelo (que es la diferencia entre las horas) y la recaudación de cada vuelo (que es la suma total de todas las tarifas pagadas por cada vuelo) siempre y cuando esa recaudación supere los 5000 euros. **(0,5 p.)**

SOLUCIÓN:

```
a) SELECT * FROM VUELO WHERE Destino='Alicante';

b) SELECT V.Numero, E.FechaHora, COUNT(E.DNIPasajero) AS CantidadPasaje
   FROM VUELO V, EMBARQUE E
   WHERE V.Numero = E.NumVuelo
   AND V.Origen = 'Madrid'
   GROUP BY V.Numero, E.FechaHora
   ORDER BY CantidadPasaje DESC;

c) SELECT P.*, FROM PASAJE P, EMBARQUE E
   WHERE E.DNIPasajero = P.DNI
   AND E.TarifaPagada IN (SELECT MIN(TarifaPagada) FROM EMBARQUE);

d) SELECT V.Numero, V.Origen, V.Destino,
   TIMEDIFF(V.HoraLlegada, V.HoraSalida) AS Duracion,
   SUM(E.TarifaPagada) AS RecaudaciónVuelo
   FROM VUELO V, EMBARQUE E,
   WHERE V.Numero = E.NumVuelo
   GROUP BY V.Numero
   HAVING RecaudacionVuelo > 5000;
```

2.- Realizar las siguientes sentencias de actualización, inserción y borrado de registros para que funcionen correctamente en MySQL (1,5 p.):

a) Insertar un nuevo registro en el pasaje con tus propios datos y sin valor en los puntos VIP. **(0,3 p.)**
b) Eliminar los pasajeros que no han volado desde el año 2010. **(0,4 p.)**
c) Incrementar en diez los puntosVIP de los pasajeros embarcados en el vuelo número IB-5030 del 19 de marzo de 2020. **(0,4 p.)**
d) Insertar en la tabla PUENTE_AEREO aquellos pasajeros que han embarcado más de 3 veces en los vuelos entre Barcelona y Madrid y viceversa. **(0,4 p.)**
(NOTA: La tabla PUENTE_AEREO tiene la misma estructura de campos y tipos de datos que la tabla PASAJE)

SOLUCIÓN:

```
a) INSERT INTO PASAJE
   VALUES ('11111111X', 'MiNombre', 'Mis Apellidos', '1980-10-05', NULL);

b) DELETE FROM PASAJE
   WHERE DNI NOT IN
        (SELECT DISTINCT DNIPasajero
         FROM EMBARQUE
         WHERE YEAR(FechaHoraVuelo)>2010);


c) UPDATE PASAJE P
   SET P.PuntosVIP = P.PuntosVIP + 10
   WHERE P.DNI IN (SELECT DNIPasajero
                   FROM EMBARQUE
                   WHERE NumVuelo = 'IB-5030'
                   AND FechaHoraVuelo = '2020-03-19');

d) INSERT INTO PUENTE_AEREO
   SELECT DISTINCT P.* FROM PASAJE P, EMBARQUE E, VUELO V
   WHERE P.DNI = E.DNIPasajero
   AND V.Numero = E.NumVuelo
   AND V.Origen IN ('Barcelona', 'Madrid')
   AND V.Destino IN ('Barcelona', 'Madrid')
   GROUP BY P.DNI
   HAVING COUNT(P.DNI)>3;
```

3.- Teniendo en cuenta las mismas tablas de los ejercicios anteriores pero con tipos de datos Oracle, hay que realizar los siguientes apartados utilizando el lenguaje PL/SQL:

NOTA: Al final del examen tienes un ANEXO con las estructuras básicas que pueden servirte de ayuda.

a) Crear un procedimiento almacenado que reciba como parámetro el número de vuelo y la fecha y hora del mismo y devuelva el DNI, nombre, apellidos, número de asiento y tarifa pagada de cada uno de los pasajeros que han embarcado en dicho vuelo. **(1,75 p.)**

```
CREATE OR REPLACE PROCEDURE listado_pasaje_vuelo(pVuelo varchar2, pFecHora DATETIME)
AS

/*Declaración de cursor */
CURSOR cListadoPasaje IS

SELECT P.DNI, P.Nombre, P.Apellidos, E.NumAsiento, E.TarifaPagada
FROM   PASAJE  P, EMBARQUE E
WHERE  P.DNI = E.DNIPasajero
AND     E.NumVuelo = pVuelo
AND    E.FechaHoraVuelo = pFecHora;

/* Declaración de variables */
pasajero     cListadoPasaje%ROWTYPE;

BEGIN
/*Muestra por pantalla */
DBMS_OUTPUT.PUT_LINE ('Listado de pasajeros del vuelo: ' || pVuelo ||
                        ' con fecha: ' || pFecHora);

/*Abre el cursor */
OPEN cListadoPasaje;

/*Lee la primera fila recuperada por el cursor y la almacena en la variable pasajero*/
FETCH cListadoPasaje INTO pasajero;

/* Bucle de control, mientras el cursor devuelva resultados se mostrará por pantalla el valor
recuperado y se leerá la siguiente fila del cursor */
WHILE cListadoPasaje%FOUND LOOP
       DBMS_OUTPUT.PUT_LINE(pasajero.DNI || '-'
                            || pasajero.Nombre || '-'
                            || pasajero.Apellidos || '-'
                            || pasajero.NumAsiento || '-'
                            || pasajero.TarifaPagada);

FETCH cListadoPasaje INTO pasajero;
END LOOP;

/*Cierra el cursor */
CLOSE cListadoPasaje;

END listado_pasaje_vuelo;
```

b) Crear una función que reciba como parámetro el DNI de un pasajero y devuelva la suma total de las tarifas pagadas en todos sus vuelos. **(1,25 p.)**

```
CREATE OR REPLACE FUNCTION total_gastado(pDNI VARCHAR2)

/*Devuelve el valor de  tipo numérico */
RETURN NUMBER
IS
/*Declaración de variables */
cantidad NUMBER;

BEGIN
/* Suma de las tarifas pagadas por el pasajero */

SELECT SUM(TarifaPagada) into cantidad
FROM EMBARQUE
WHERE DNIPasajero = pDNI;

/*Valor de retorno de la función */
RETURN cantidad;

END total_gastado;
```

4.- Dado el siguiente código donde se crea un objeto utilizando Oracle, realiza los siguientes apartados escribiendo el código necesario para que funcionen correctamente en Oracle: (1 punto)

a) Crear una tabla llamada Catalogo de objetos Armario. **(0,3 p.)**

```
CREATE TABLE Catalogo OF Armario;
```

b) Insertar una instancia de un objeto Armario en la tabla creada anteriormente con los datos que desees introducir para cada atributo. **(0,3 p.)**

```
INSERT INTO Catalogo VALUES ('12345', 'Modelo XL',
      'Armario grande de color blanco y madera pino' 240 ,125, 50);
```

c) Obtener por pantalla el volumen de la instancia de Armario insertada anteriormente en la tabla Catalogo utilizando la función del objeto Armario que devuelve el volumen. **(0,4 p.)**

```
select c.getVolumen() from catalogo c where codigo='12345'
/* IMPORTANTE poner el alias c para que funcione la sentencia */
```

O bien con un bloque de código como el siguiente:

```
DECLARE
a1 Armario;
BEGIN
SELECT VALUE(c) INTO a1 FROM Catalogo c WHERE codigo='12345';
DBMS_OUTPUT.PUT_LINE('El volumen es: ' || a1.getVolumen);
END;
```

```
CREATE OR REPLACE TYPE Armario AS OBJECT(
    Codigo VARCHAR2(5),
    Nombre VARCHAR2(40),
    Descripcion VARCHAR2(200),
    Altura NUMBER(3),
    Anchura NUMBER(3),
    Profundidad NUMBER(3),

    MEMBER FUNCTION getVolumen RETURN NUMBER
);

CREATE OR REPLACE TYPE BODY Armario AS

       MEMBER FUNCTION getVolumen RETURN NUMBER IS
       BEGIN
              RETURN (SELF.Altura * SELF.Anchura * SELF.Profundidad);
       END getVolumen;
END;
```

ANEXO: Estructuras PL/SQL que pueden ser útiles para la realización del examen práctico.

Procedimiento en PL/SQL

```
CREATE[OR REPLACE] PROCEDURE nombre_procedimiento([parámetros])
IS
[DECLARE]
        [<variables locales>]
BEGIN
        <código del procedimiento>
[EXCEPTION]
END [nombre_procedimiento];
```

Función en PL/SQL

```
CREATE [OR REPLACE] FUNCTION nombre_función([parámetros])
RETURN tipodato
IS
[DECLARE]
        [<variables locales>]
BEGIN
        <código de la función>
RETURN <valor>;
[EXCEPTION]
END [nombre_función];
```

Disparador en PL/SQL

```
CREATE[OR REPLACE]TRIGGER nombre_trigger
{BEFORE|AFTER|INSTEAD OF}
{INSERT|DELETE|UPDATE [OF <atributo]} ON <tabla>
[FOR EACH ROW|STATEMENT]
[WHEN condición]
[DECLARE]
        [<variables locales>]
BEGIN
        <código del trigger>
[EXCEPTION]
END[nombre_trigger];
```

Declaración y varias formas de manejar un cursor en PL/SQL.

```
CURSOR prueba IS
Sentencia de consulta;
```

```
LOOP
FETCH prueba INTO mi_registro;
EXIT WHEN prueba%NOTFOUND;
--- se procesa el registro
END LOOP;
```

```
FETCH prueba INTO mi_registro;
WHILE prueba%FOUND LOOP
DBMS_OUTPUT.PUT_LINE('Hola');
FETCH prueba INTO mi_registro;
END LOOP;
```

```
FOR loop_contador IN [REVERSE]
 lim_inf .. lim_sup LOOP
Instrucciones ;
DBMS_OUTPUT.PUT_LINE('Hola');
END LOOP;
```

Estructura de control en PL/SQL

```
IF condicion THEN
DBMS_OUTPUT.PUT_LINE('una cosa');
ELSE
DBMS_OUTPUT.PUT_LINE('otra cosa');
END IF;
```