

Nivelación: Exámen Final

(parte práctica)

Docentes: Ignacio Urteaga

Autor: Ing. Gustavo Bustamante

Bases de Datos Relacionales	3
Programación Lógica	8
Programación en Python	10
Probabilidad y Estadística	13

Bases de Datos Relacionales

Construir una consulta SQL que permita agrupar en un único registro los datos de un alumno con todas sus calificaciones y demás atributos, a partir de las siguientes tablas:

Alumnos				
idAlumno	EdadActual	Sexo	FechaIngreso	Egresado
1	35	M	2008-03-01	si
2	30	F	2010-03-01	no
3	45	M	2002-03-01	si
4	32	F	2012-03-01	si
5	25	F	2018-03-01	no

Asignaturas			
idAsignatura	Titulo	CargaHoraria	Año
1	Analisis 1	160	1
2	Fisica 1	240	2
3	Mecanica 1	160	3
4	Electronica	480	4

Calificaciones				
idCalificaciones	idAlumno	idAsignatura	Fecha	Calificacion
1	1	1	2008-09-01	9
2	2	1	2011-10-01	9
3	3	2	2004-03-01	4
4	4	2	2013-03-01	6
5	5	3	2020-12-01	7
6	5	3	2020-12-01	5
7	4	3	2013-03-01	6
8	4	1	2013-03-01	6

(En la tabla “Calificaciones” se adicionaron filas para verificar el correcto funcionamiento de la consulta realizada)

La consulta debe ser realizada para las 4 asignaturas del plan de estudios, quedándose con la máxima calificación (si la hubiera rendido varias veces), siendo la salida esperada:

Salida esperada						
idAlumno	EdadAlIngreso	Sexo	Egresado	Asignatura	MesesDesdeIngreso	Calificacion

SOLUCIÓN

Creación de las 3 tablas dato:

```
CREATE TABLE Alumnos (
    idAlumno INT NOT NULL AUTO_INCREMENT,
    EdadActual INT NOT NULL,
    Sexo ENUM('M', 'F') DEFAULT NULL,
    FechaIngreso DATE DEFAULT NULL,
    Egresado ENUM('si', 'no') DEFAULT NULL,
    PRIMARY KEY (idAlumno)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;





CREATE TABLE Asignaturas (
    idAsignatura INT NOT NULL AUTO_INCREMENT,
    Titulo VARCHAR(45) DEFAULT NULL,
    CargaHoraria INT NOT NULL,
    Año INT NOT NULL,
    PRIMARY KEY (idAsignatura)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE Calificaciones (
    idCalificaciones INT NOT NULL AUTO_INCREMENT,
    idAlumno INT NOT NULL,
    idAsignatura INT NOT NULL,
    Fecha DATE DEFAULT NULL,
    Calificacion INT NOT NULL,
    PRIMARY KEY (idCalificaciones),
    FOREIGN KEY (idAlumno) REFERENCES Alumnos(idAlumno),
    FOREIGN KEY (idAsignatura) REFERENCES asignaturas(idAsignatura)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Carga de datos tabla “Alumnos”

```
INSERT INTO alumnos (EdadActual, Sexo, FechaIngreso, Egresado)
VALUES ('35', 'M', '2008-03-01', 'si');
INSERT INTO alumnos (EdadActual, Sexo, FechaIngreso, Egresado)
VALUES ('30', 'F', '2010-03-01', 'no');
INSERT INTO alumnos (EdadActual, Sexo, FechaIngreso, Egresado)
VALUES ('45', 'M', '2002-03-01', 'si');
INSERT INTO alumnos (EdadActual, Sexo, FechaIngreso, Egresado)
VALUES ('32', 'F', '2012-03-01', 'si');
INSERT INTO alumnos (EdadActual, Sexo, FechaIngreso, Egresado)
VALUES ('25', 'F', '2018-03-01', 'no');

SELECT * FROM tienda.alumnos;
```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:    Export/Import: 					
	idAlumno	EdadActual	Sexo	FechaIngreso	Egresado
▶	1	35	M	2008-03-01	si
	2	30	F	2010-03-01	no
	3	45	M	2002-03-01	si
	4	32	F	2012-03-01	si
	5	25	F	2018-03-01	no

Carga de datos tabla “Asignaturas”

```
INSERT INTO asignaturas (Titulo, CargaHoraria, Año)
VALUES ('Analisis 1', '160', '1');
INSERT INTO asignaturas (Titulo, CargaHoraria, Año)
VALUES ('Fisica 1', '240', '2');
INSERT INTO asignaturas (Titulo, CargaHoraria, Año)
VALUES ('Mecanica 1', '160', '3');
INSERT INTO asignaturas (Titulo, CargaHoraria, Año)
VALUES ('Electronica', '480', '4');

SELECT * FROM tienda.asignaturas;
```

idAsignatura	Titulo	CargaHoraria	Año
1	Analisis 1	160	1
2	Fisica 1	240	2
3	Mecanica 1	160	3
4	Electronica	480	4

Carga de datos tabla “Calificaciones”

```
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('1', '1', '2008-09-01', '9');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('2', '1', '2011-10-01', '9');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('3', '2', '2004-03-01', '4');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('4', '2', '2013-03-01', '6');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('5', '3', '2020-12-01', '7');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('5', '3', '2020-12-01', '5');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('4', '3', '2013-03-01', '6');
INSERT INTO calificaciones (idAlumno, idAsignatura, Fecha, Calificacion)
VALUES ('4', '1', '2013-03-01', '6');

SELECT * FROM tienda.calificaciones;
```

idCalificaciones	idAlumno	idAsignatura	Fecha	Calificacion
1	1	1	2008-09-01	9
2	2	1	2011-10-01	9
3	3	2	2004-03-01	4
4	4	2	2013-03-01	6
5	5	3	2020-12-01	7
6	5	3	2020-12-01	5
7	4	3	2013-03-01	6
8	4	1	2013-03-01	6

Funciones para el cálculo de años y meses:

```
# ----- Funcion calculo de años -----
DELIMITER //
CREATE FUNCTION anios(edad int, date1 date) RETURNS int DETERMINISTIC
BEGIN
    DECLARE date2 DATE;
    Select current_date() into date2;
    RETURN edad - (year(date2) - year(date1));
END //
# ----- Funcion calculo de meses -----
DELIMITER //
CREATE FUNCTION meses(date1 date, date2 date) RETURNS int DETERMINISTIC
BEGIN
    DECLARE meses INT; DECLARE anios INT; DECLARE salida INT;

    SET meses = month(date1) - month(date2),
    anios = year(date1) - year(date2);
    IF meses = 0 THEN
        IF anios = 0 THEN
            SET salida = 0;
        ELSE
            SET salida = anios * 12;
        END IF;
    ELSE
        IF anios = 0 THEN
            SET salida = meses;
        ELSE
            SET salida = anios * 12 + meses;
        END IF;
    END IF;
    RETURN salida;
END //
```

Consulta:

```
CREATE TABLE consulta
AS SELECT
Alumnos.idAlumno,
anios(EdadActual, FechaIngreso) AS EdadAlingreso,
Sexo,
Egresado,
Titulo,
meses(Fecha, FechaIngreso) AS MesesDesdeIngreso,
Calificacion
FROM Alumnos, Calificaciones, Asignaturas
WHERE Alumnos.idAlumno = 5
AND Calificaciones.idAlumno = 5
AND Asignaturas.idAsignatura = Calificaciones.idAsignatura;

SELECT * FROM tienda.consulta;
```

Salida:

```
CREATE VIEW vista1
AS SELECT
idAlumno, EdadAlingreso, Sexo, Egresado, Titulo, MesesDesdeIngreso,
MAX(Calificacion) as Calificacion
FROM consulta
WHERE Titulo in (SELECT Titulo FROM consulta
GROUP BY Titulo
HAVING COUNT(*) > 1);

CREATE VIEW vista2
AS SELECT
idAlumno, EdadAlingreso, Sexo, Egresado, Titulo, MesesDesdeIngreso, Calificacion
FROM consulta
WHERE Titulo in (SELECT Titulo FROM consulta
GROUP BY Titulo
HAVING COUNT(*) = 1);

CREATE VIEW vista3
as SELECT idAlumno, EdadAlingreso, Sexo, Egresado, Titulo, MesesDesdeIngreso,
Calificacion FROM vista1
WHERE idAlumno is not null
union all
SELECT idAlumno, EdadAlingreso, Sexo, Egresado, Titulo, MesesDesdeIngreso,
Calificacion FROM vista2
WHERE idAlumno is not null;

SELECT * FROM tienda.vista3;
```

Distintas consultas Realizadas: (ver tablas con valores datos)

- Materia Repetida, mismo Alumno = conserva la mayor nota

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	idAlumno	EdadAlingreso	Sexo	Egresado	Titulo	MesesDesdeIngreso	Calificacion
	5	21	F	no	Mecanica 1	33	7

- Distintas Materias, mismo alumno

Result Grid

Filter Rows:

Export:

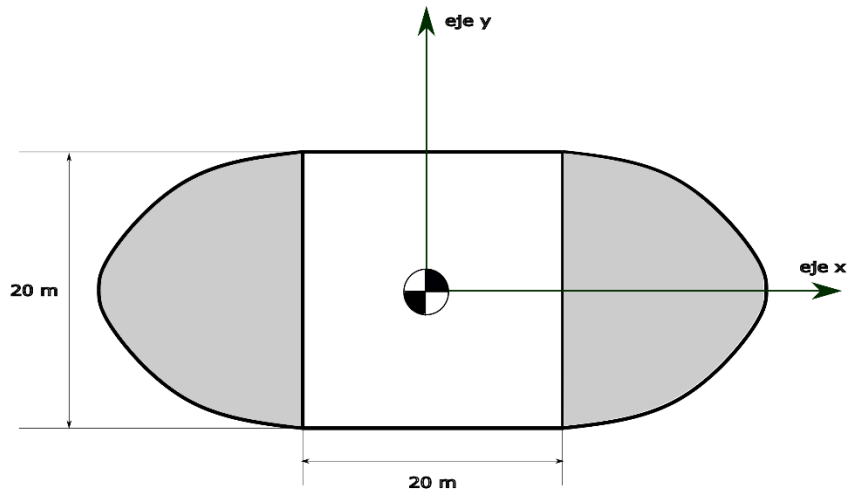
Wrap Cell Content:

	idAlumno	EdadAlingreso	Sexo	Egresado	Titulo	MesesDesdeIngreso	Calificacion
	4	22	F	si	Analisis 1	12	6
	4	22	F	si	Fisica 1	12	6
	4	22	F	si	Mecanica 1	12	6

Programación Lógica

Un barco porta contenedores acomoda en su cubierta 10 contenedores. Se pide verificar el equilibrio del barco, conociendo los siguientes datos:

- Barco: 20 x 20 x 5 (largo, ancho, altura).
- Contenedores : 5 x 2 x 2 (largo, ancho, altura) y el peso de cada uno, el cual será distinto para cada contenedor.

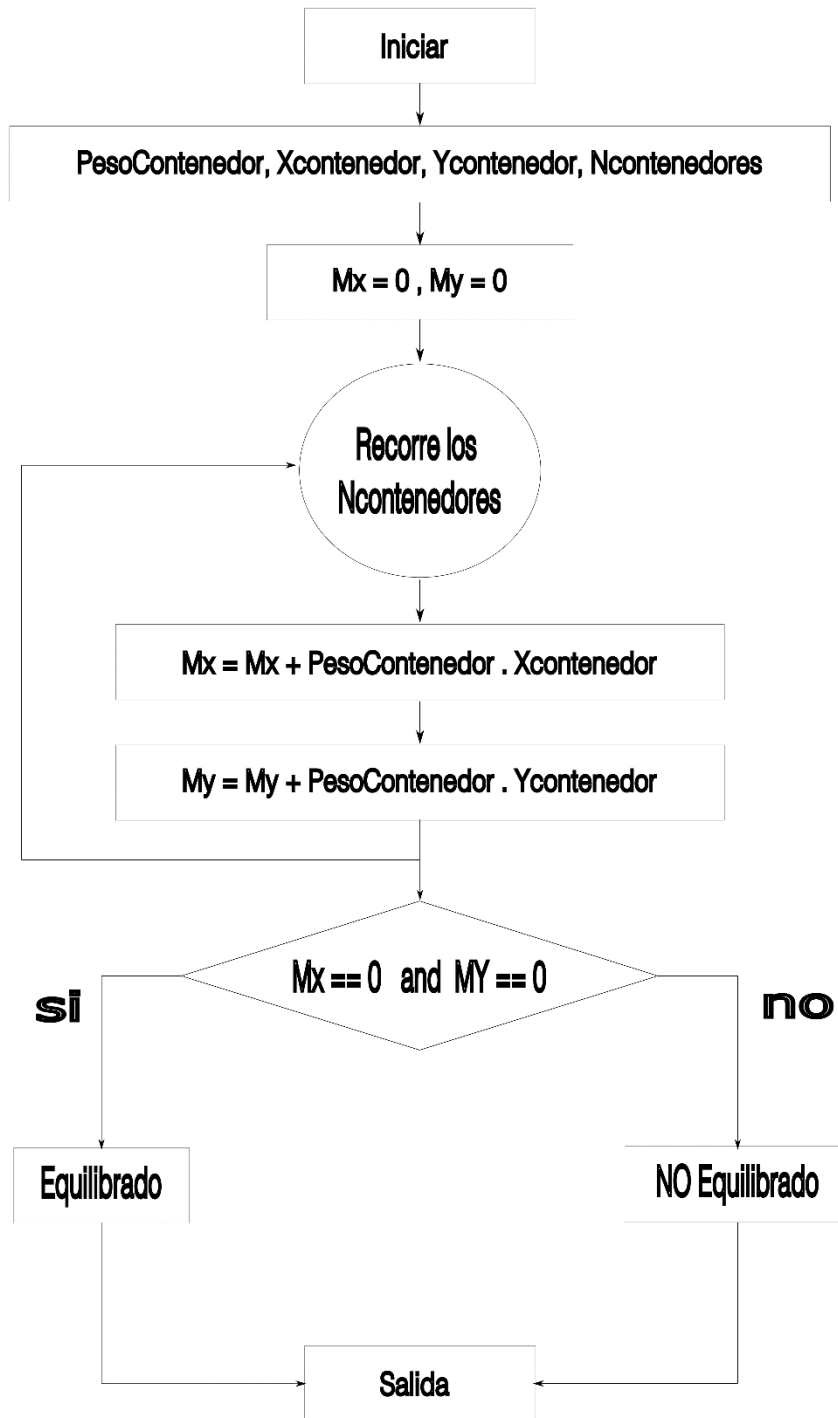


SOLUCION

Estableciendo un sistema de ejes coordenados coincidente con el centro de masa del barco, se proceden a calcular una sumatoria de Momentos de fuerza según las direcciones respecto del centro. Para el equilibrio se debe verificar:

$$\sum \text{Peso} \cdot \text{distancia}_{\text{eje } x} \cong 0 \quad \text{y} \quad \sum \text{Peso} \cdot \text{distancia}_{\text{eje } y} \cong 0$$

El algoritmo para la resolución queda representado con el siguiente diagrama:



Programación en Python

Asuma que tiene los datos de los contenedores (peso, coordenada x, coordenada y) del problema anterior en un archivo plano separado por comas (CSV)

Cargue en variables los datos que necesite para resolver el problema y realice la estimación del problema anterior.

SOLUCION

Para la solución del problema se establecerán varios casos de estudio.

- Los archivos .csv serán leídos mediante la librería “pandas”
- Se define una función “Equilibrio” para la solución de la consulta
- Se establecen 3 casos de estudio, uno con los contenedores en equilibrio y dos estudios con los contenedores ubicados en distintas configuraciones espaciales dentro de la cubierta del barco.

La función de solución tiene la forma:

```
✓ [1] import pandas as pd
0 s

✓ [9] def Equilibrio(datos):
0 s
    peso = datos['peso']
    x = datos['x']
    y = datos['y']

    Mx, My = 0, 0
    for i in range(len(peso)):
        Mx += peso[i] * x[i]
        My += peso[i] * y[i]

    if Mx == 0 and My == 0:
        return 'Equilibrado'
    else:
        return 'NO Equilibrado'
```

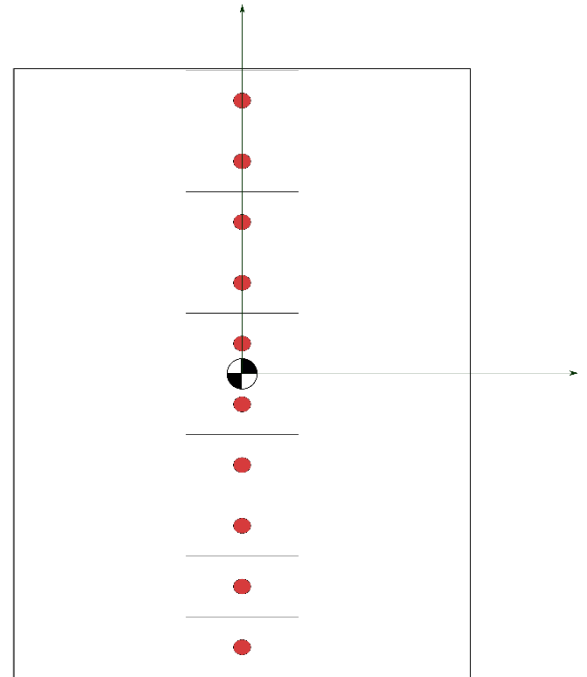
Caso 0:

En este caso todos los contenedores se encuentran en línea, de manera tal que su coordenada en “y” sea igual a cero. Todos los contenedores tienen el mismo peso.

```
✓ [14] datos0 = pd.read_csv('caso00.csv')
0s print(datos0)
print('-----')
Equilibrio(datos0)
```

	peso	x	y
0	20	9	0
1	20	7	0
2	20	5	0
3	20	3	0
4	20	1	0
5	20	-1	0
6	20	-3	0
7	20	-5	0
8	20	-7	0
9	20	-9	0

'Equilibrado'



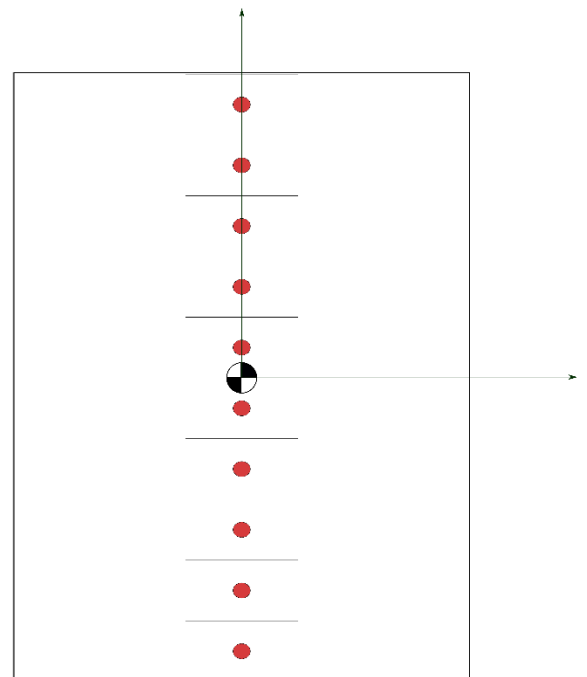
Caso 1:

En este caso todos los contenedores se encuentran en línea, de manera tal que su coordenada en “y” sea igual a cero. Todos los contenedores tienen distinto peso.

```
✓ [15] datos1 = pd.read_csv('caso01.csv')
0s print(datos1)
print('-----')
Equilibrio(datos1)
```

	peso	x	y
0	20.0	9	0
1	18.0	7	0
2	5.0	5	0
3	15.0	3	0
4	18.0	1	0
5	21.0	-1	0
6	19.5	-3	0
7	6.0	-5	0
8	13.0	-7	0
9	11.0	-9	0

'NO Equilibrado'



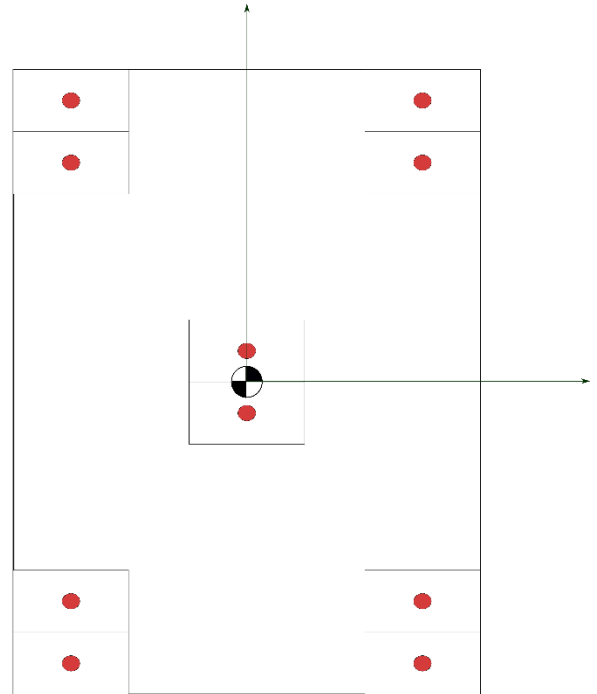
Caso 2:

En este caso los contenedores están distribuidos a lo largo de la cubierta y todos tienen pesos distintos

```

✓ 0s ▶ datos2 = pd.read_csv('caso02.csv')
print(datos2)
print('-----')
Equilibrio(datos2)

   peso  x  y
0  20.0 -7.5  9
1  18.0 -7.5  7
2   5.0  7.5  9
3  15.0  7.5  7
4  18.0  1.0  0
5  21.0 -1.0  0
6  19.5 -7.5 -9
7   6.0 -7.5 -7
8  13.0  7.5 -9
9  11.0  7.5 -7
-----
'NO Equilibrado'
    
```



Probabilidad y Estadística

Sabiendo que para el problema anterior, el promedio de los contenedores pesa 19.8 toneladas y el desvío estándar es de 3 toneladas.

Asuma que la distribución de probabilidades de los contenedores es una gaussiana.

1. Calcule la probabilidad de encontrar un contenedor de 30 toneladas o más.
2. Los contenedores de tres barcos se descargan en puerto en tres grupos separados. Seleccionamos tres contenedores al azar de cada grupo. Obtenemos tres contenedores de 30 toneladas. Resulta esto compatible con un nivel de confianza del 90% que los contenedores vengan del barco cuyo contenedor promedio pesaba 19.8 toneladas con un desvío estándar de 3 toneladas.

```
✓ [21] from scipy import stats
0s      import numpy as np
```

```
✓ [22] # Prueba 't de student'
0s      promedio = 19.8
      valor = 30
      sd = 3
      n = 10          # Nº de contenedores

      # prueba t
      t = (promedio - valor) / (sd / np.sqrt(n))
```

```
✓ [23] print('El estadígrafo es:')
0s      print(t)
```

```
El estadígrafo es:
-10.75174404457249
```

```
✓ [24] # Probabilidad de que un contenedor pese más de 30 toneladas con esos datos
0s      prob = stats.norm.cdf(t) * 100
      print('Probabilidad de que un contenedor pese más de 30 toneladas con esos datos:')
      print(prob)
```

```
Probabilidad de que un contenedor pese más de 30 toneladas con esos datos:
2.9075768407925505e-25
```

Se obtiene una probabilidad muy baja de tener un contenedor con un peso de más de 30 toneladas con los datos proporcionados.