Segundo trabajo de bases de datos (25%)

Fecha de entrega: Miércoles 12 de mayo de 2021 hasta las **5pm**. Trabajos recibidos después de esa hora no se califican. Los trabajos deben ser enviados a este correo: *fjmoreno@unal.edu.co*
- Trabajos enviados "por accidente" a otros correos no se calificarán.
- Envíe **un solo** archivo comprimido con todo el trabajo.
- Para facilitar la identificación, el nombre del archivo comprimido debe tener el primer apellido de cada integrante del grupo. Ejemplo: **Lipa_Electra_Jolie.zip**
- No se reciben ni se califican versiones "mejoradas". **No envíe varias versiones**. No envíe varios programas (soluciones). **Solo un programa por grupo**.
- Grupos máximo de **tres** estudiantes.

Nota: Los progamas deben estar comentados, explicando si es necesario en cada línea de código o grupo de líneas (cada 2, 3, 4, o 5 líneas de código) que se está haciendo. Trabajos copiados parcial o totalmente, se calificarán con cero para todos los integrantes involucrados sin excepción. **Programas no explicados ni comentados se calificarán con cero, así hagan lo solicitado**.

Cualquier duda consultarla con el profesor. El monitor les puede ayudar con recomendaciones pero su labor no es hacerles el trabajo **ni está autorizado para cambiar las condiciones del trabajo**.

Realice un programa en el lenguaje que desee, puede ser Java, Python u otro. El programa recibe como entrada una muestra de datos (una relación) que tendrá **3** o **4** atributos. Se garantiza que la muestra de datos corresponde a una relación (es decir, no tiene, por ejemplo, tuplas repetidas). La muestra de datos puede tener **cualquier** número de tuplas, pero por simplicidad solamente tendrá 3 o 4 atributos. Usted decide como recibir la muestra de datos en su programa: debe recibir los nombres de los atributos y las tuplas.

a) Lo primero que debe hacer el programa es analizar la muestra de datos y **descubrir TODAS** las **dependencias funcionales COMPLETAS** que se **cumplen** EN LA MUESTRA. **No incluya** dependencias funcionales generadas por reflexión, **ni** por aumento **ni** por autodeterminación. El programa debe imprimir las dependencias funcionales COMPLETAS. Por simplicidad **NO** tenga en cuenta dependencias funcionales donde el lado izquierdo o derecho sea el conjunto vacío.

b) A continuación el programa debe imprimir **todas** las claves candidatas de la relación ingresada. Note que en el **peor** de los casos, la relación ingresada tendrá al menos una clave candidata (que en el **peor** de los casos serán todos los atributos de la relación). Por supuesto, cada clave candidata debe cumplir unicidad e irreducibilidad.

c) El programa debe imprimir el **cierre mínimo** del conjunto de las dependencias funcionales completas encontradas en el literal a).

d) Si la relación ingresada tiene **UNA SOLA** clave candidata (que de hecho será entonces la clave primaria), proceda a aplicar el algoritmo de descomposición 3NF explicado en clase (se explica en la clase del martes 27 de abril). El programa debe imprimir las relaciones resultantes de la descomposición cada una con sus tuplas correspondientes. Si la relación ingresada tiene más de una clave candidada, imprima: "El algoritmo de descomposición no aplica".

Veamos tres **ejemplos** concretos.

Nota: Su programa debe funcionar para **cualquier** muestra de datos que tenga 3 o 4 atributos, es decir, no solamente para los siguientes ejemplos.

Ejemplo 1.
Supongamos que se ingresa la siguiente muestra de datos:

cédula	**nombre**	**salario**
256	Lina	100
943	Carlos	110
119	Juan	50
221	Juan	60
333	María	50
888	Juan	50

Según esta muestra de datos tenemos:
a) Las dependencias funcionales completas son:
cédula --> nombre
cédula --> salario
b) Hay una sola clave candidata:
cédula
c) El cierre mínimo de las dependencias funcionales completas del punto a) son:
cédula --> nombre
cédula --> salario
d) En este caso al aplicar el algoritmo de descomposición 3NF, la relación **no** se parte, el resultado es la misma muestra de datos ingresada, es decir, el programa debe imprimir en este literal lo siguiente:

cédula	**nombre**	**salario**
256	Lina	100
943	Carlos	110
119	Juan	50
221	Juan	60

333 María 50
888 Juan 50

Ejemplo 2.
Supongamos que se ingresa la siguiente muestra de datos:

nit	pdto	color	cantidad
10	p1	azul	50
10	p2	azul	50
10	p3	verde	10
10	p4	azul	80
10	p5	rojo	50
20	p1	azul	60
20	p3	verde	50
30	p1	azul	50

Según esta muestra de datos tenemos:
a) Las dependencias funcionales completas son:
{nit, pdto} --> cantidad
pdto --> color
b) Hay una sola clave candidata:
{nit, pdto}
c) El cierre mínimo de las dependencias funcionales completas del punto a) son:
{nit, pdto} --> cantidad
pdto --> color
d) En este caso al aplicar el algoritmo de descomposición 3NF, se generan dos particiones, es decir, el programa debe imprimir en este literal lo siguiente:

Partición 1:

nit	pdto	cantidad
10	p1	50
10	p2	50
10	p3	10
10	p4	80
10	p5	50
20	p1	60
20	p3	50
30	p1	50

Partición 2:

pdto	color
p1	azul
p2	azul
p3	verde
p4	azul
p5	rojo

Ejemplo 3.
Supongamos que se ingresa la siguiente muestra de datos:

código	**país**	**idioma**
1	Francia	Francés
3	Francia	Francés
6	USA	Inglés
8	UK	Inglés
9	UK	Inglés

Según esta muestra de datos tenemos:
a) Las dependencias funcionales completas son:
código --> país
código --> idioma
país --> idioma
b) Hay una sola clave candidata:
código
c) El cierre mínimo de las dependencias funcionales completas del punto a) son:
código --> país
país --> idioma
d) En este caso al aplicar el algoritmo de descomposición 3NF, se generan dos particiones, es decir, el programa debe imprimir en este literal lo siguiente:

Partición 1

código	**país**
1	Francia
3	Francia
6	USA
8	UK
9	UK

Partición 2

país	**idioma**
Francia	Francés
USA	Inglés
UK	Inglés