Ampliación de bases de datos
Grado en Ingeniería informática. Grupos B y C
Examen Segunda convocatoria. Curso 2022-2023

Ejercicio 1. (2 puntos).
Una empresa de transporte de viajeros necesita almacenar información para la gestión de su red de autobuses. La información se organiza de la siguiente forma:

- La ciudad de origen y la de destino determinan una línea. En cada línea existe al menos un trayecto (un viaje) diario.
- Cada trayecto se identifica, además, por la hora de salida. Cada autobús está asignado a una única línea y a un único trayecto. Es necesario llevar la gestión de sus plazas. Todas las plazas de los autobuses están etiquetadas con un número entre 1 y 50.
- Cada conductor está asignado a un único autobús.

Chapuzas Informáticas S.A. ha ofrecido a la empresa de transporte el siguiente diseño de la base de datos:

- Tabla 1: lineas(**origen**,**destino**,numero-trayectos-diarios)
- Tabla 2: trayectos(**origen**,**destino**,**hora-salida**,**cod-autobus**,nif-conductor,nombre-conductor,año-de-compra)

  Comentarios adicionales: `cod-autobus` es un código que identifica a un autobús, `año-de-compra` es el año en el que compañía compró el autobús.
- Tabla 3: plazas-trayectos(**origen**,**destino**,**hora-salida**,**num-plaza**,cod-autobus,tipo-de-plaza,libre)

  Comentarios adicionales: `num-plaza` es el número de plaza en cada autobús, `tipo-de-plaza` indica si es una plaza VIP o no, `libre` indica si esa plaza está libre en ese trayecto o no. Nota: para un autobús una plaza será VIP o no en todos los trayectos.

Determina si esta base de datos está en 3FN. Si no es así, indica qué descomposición de tablas es necesaria para que esté en 3FN. Por supuesto, no es válido añadir nuevos campos. Nota: un campo en negrita significa que es parte de la clave.

Soluciones:

- La tabla *líneas* no sufre cambios.
- De la tabla *trayectos* surgen las siguientes tablas:
  - autobuses(**origen**,**destino**,**hora-salida**,cod-autobus)
  - conductores-autobuses(**cod-autobus**,nif-conductor,año-de-compra)
  - info-conductores(**nif-conductor**,nombre-conductor)
- De la tabla *plazas-trayectos* surgen las siguientes tablas:
  - plazas-trayectos(**origen**,**destino**,**hora-salida**,**num-plaza**,libre)
  - autobus-plazas(**cod-autobus**,**num-plaza**,tipo-de-plaza)

Ejercicio 2. (4 puntos). Una empresa almacena información sobre sus ventas y empleados con la siguiente base de datos extensional:

```
%ventas(cod-vendedor, ciudad, provincia, mes, año, ventaMA)

%vendedores(cod-vendedor, nombre-vendedor).
```

Es posible que haya vendedores que todavía no hayan realizado ninguna venta. `ventaMA` es la suma de las ventas realizadas por un vendedor (`cod-vendedor`), en una `ciudad`, una `provincia`, en un `mes` de un `año`.

Diseña las siguientes reglas:

1. `ventas_totales(V)`: `V` es el valor de las ventas totales de la empresa. Ejemplo:

```
DES> ventas_totales(V).

{
  ventas_totales(259670)
}
Info: 1 tuple computed.
```

2. `ventas_por_vendedor(N,V)`: `V` es el valor de las ventas totales por vendedor de nombre `N`. Debe salir el nombre de todos los vendedores. Ejemplo:

```
DES> ventas_por_vendedor(N,V).

{
  ventas_por_vendedor(ana,71600),
  ventas_por_vendedor(antonio,64360),
  ventas_por_vendedor(isabel,null),
  ventas_por_vendedor(juan,4720),
  ventas_por_vendedor(maria,118990),
  ventas_por_vendedor(pedro,null)
}
Info: 6 tuples computed.
```

3. `venta_mas_alta_provincia(P,C)`: `C` es el valor de la venta (`ventaMA`) más alta para cada provincia `P`. Ejemplo:

```
DES> venta_mas_alta_provincia(P,C).

{
  venta_mas_alta_provincia(avila,17000),
  venta_mas_alta_provincia(madrid,71000),
  venta_mas_alta_provincia(toledo,31000)
}
Info: 3 tuples computed.
```

4. `mejor_vendedor_provincia(P,Cod)`: `Cod` es el código del vendedor cuya suma de ventas (`ventaMA`) es mayor para cada provincia `P`. Ejemplo:

```
DES> mejor_vendedor_provincia(P,Cod).

{
  mejor_vendedor_provincia(avila,444),
  mejor_vendedor_provincia(madrid,222),
  mejor_vendedor_provincia(toledo,333)
```

```
}
Info: 3 tuples computed.
```

Podéis encontrar en el fichero `ventas-extensinal.dl` la base extensional de estos ejemplos.

Soluciones:

```
ventas_totales(V):- group_by(
                    ventas(_,_,_,_,_,_,C),
                    [],
                    V=sum(C)
                    ).
```

```
ventas_vendedor_aux(Cod,V):- group_by(
                    ventas(Cod,_,_,_,_,_,C),
                    [Cod],
                    V=sum(C)
                    ).
ventas_por_vendedor(N,V):- lj(vendedores(Cod,N),ventas_vendedor_aux(Cod2,V),Cod=Cod2).
```

```
venta_mas_alta_provincia(P,C):- group_by(
                    ventas(_,_,P,_,_,V),
                    [P],
                    C=max(V)
                    ).
```

```
ventas_por_provincia_vendedor(Cod,P,V):- group_by(
                    ventas(Cod,_,P,_,_,C),
                    [Cod,P],
                    V=sum(C)
                    ).
%Suma de ventas mÃ¡xima por provincia y vendedor
ventas_maxima_provincia_vendedor(P,C):- group_by(
                    ventas_por_provincia_vendedor(_,P,V),
                    [P],
                    C=max(V)
                    ).
mejor_vendedor_provincia(P,Cod):- ventas_por_provincia_vendedor(Cod,P,C),
ventas_maxima_provincia_vendedor(P,C).
```

Ejercicio 3. (4 puntos). Para la gestión de una bbdd en MongoDB de una plataforma de alquiler vacacional se utiliza la siguiente estructura. Para alojamiento se almacena, entre otras, información sobre opiniones.

- Crea una base de datos llamada `vacaciones`.
- Crea una colección llamada `airbnb`.
- Introduce los datos del fichero `MongoDB.txt`.

Diseña las siguiente operaciones:

1. Mostrar solo el nombre de los propietarios (`host_name`) cuyas viviendas tienen un precio entre 100 y 150 euros y requieren un mínimo de 12 noches (`minimum_nights`). Resultado:

```
{
    "host" : {
        "host_name" : "Greta"
    }
}
```

2. Mostrar solo el id (`_id`) de todas las viviendas ordenadas descendentemente por la fecha en la que recibieron su primer comentario (`reviews`). Se asume que los comentarios se registran en orden cronológico. Resultado:

```
{
    "_id" : "1003534"
}
{
    "_id" : "1003533"
}
{
    "_id" : "1003531"
}
```

3. Mostrar solo la calle (`street`) de las viviendas cuyo primer elemento de la coordenada (`coordinates`) está entre -41 y -39 y en segundo entre 33 y 35. Resultado:

```
{
    "_id" : "1003534",
    "address" : {
        "street" : "Santa Isabel"
    }
}
```

4. Para el identificador de comentario (`reviews_id`) '4848277', introducir con una sola actualización para la clave 'scoring' (actualmente vacía) los valores 3, 2 y 5 (nota solo informativa: estos valores corresponden a limpieza, atención y situación).

Soluciones:

```
db.airbnb.find({price:{ $gt: 100, $lt:150}, minimum_nights:{$gte:12}},
{"host.host_name":1, "_id":0})
```

```
db.airbnb.find({},{_id:1}).sort({"reviews.0.date":-1})
```

```
db.airbnb.find({"address.location.coordinates.0":{$gt:-41, $lt:-39},
"address.location.coordinates.1":{$gt:33, $lt:35}},
{"address.street":1})
```

```
db.airbnb.updateOne({"reviews._id" : "4848277"},
{$push:{"reviews.$.scoring":{$each:[3,2,5]}}})
```

Duración del examen: 2:30 horas.