Datalog. Prácticas

Práctica 1. Diseña una base de datos con Datalog para realizar consultas sobre la información de una academia partiendo de la siguiente base de datos extensional con el siguiente formato:

```
% estudiantes(código de estudiante,nombre).
estudiantes(111,pepe).
estudiantes(222,juan).
estudiantes(333,ana).
estudiantes(444,maria).
estudiantes(555,carlos).

% asignatura(código de la asignatura,nombre de la asignatura,número de créditos).
asignatura(c1,matematicas,20).
asignatura(c2,fisica,25).
asignatura(c3,quimica,10).

% matriculaciones(código de estudiante,código de la asignatura,nota,
% año de matriculación).
matriculaciones(111,c1,6,2020).
matriculaciones(111,c2,7,2020).
matriculaciones(333,c1,6,2021).
matriculaciones(444,c3,9,2021).
matriculaciones(444,c2,10,2021).
matriculaciones(444,c1,5,2020).
matriculaciones(111,c3,9,2022).
matriculaciones(333,c2,10,2022).
matriculaciones(333,c3,9,2022).
matriculaciones(555,c3,9,2022).
matriculaciones(444,c2,10,2022).
matriculaciones(444,c1,5,2022).
```

Diseña las siguientes consultas:

- `listado(N,Asig,A)`: asignaturas matriculadas (Asig) por el alumno (N) en el año (A). Ejemplo:

```
DES> listado(N,Asig,A).

{
  listado(ana,fisica,2022),
  listado(ana,matematicas,2021),
  listado(ana,quimica,2022),
  listado(carlos,quimica,2022),
  listado(maria,fisica,2021),
  listado(maria,fisica,2022),
  listado(maria,matematicas,2020),
  listado(maria,matematicas,2022),
  listado(maria,quimica,2021),
  listado(pepe,fisica,2020),
  listado(pepe,matematicas,2020),
  listado(pepe,quimica,2022)
}
Info: 12 tuples computed.
```

- `numeroMatriculados(A,Num)`: número de alumnos matriculados (Num) an el año (A). Ejemplo:

```
DES> numeroMatriculados(A,Num).
```

```
{
  numeroMatriculados(2020,3),
  numeroMatriculados(2021,3),
  numeroMatriculados(2022,6)
}
Info: 3 tuples computed.
```

- `maximaNotaAsig(A,Asig,N)`: nota máxima (N) de la asignatura (Asig) en el año (A). Ejemplo:

```
DES> maximaNotaAsig(A,Asig,N).

{
  maximaNotaAsig(2020,fisica,7),
  maximaNotaAsig(2020,matematicas,6),
  maximaNotaAsig(2021,fisica,10),
  maximaNotaAsig(2021,matematicas,6),
  maximaNotaAsig(2021,quimica,9),
  maximaNotaAsig(2022,fisica,10),
  maximaNotaAsig(2022,matematicas,5),
  maximaNotaAsig(2022,quimica,9)
}
Info: 8 tuples computed.
```

- `asigCreditos(Asig)`: Asig es la asignatura con más créditos. Ejemplo:

```
DES> asigCreditos(Asig).

{
  asigCreditos(fisica)
}
Info: 1 tuple computed.
```

Solución:

```
%Listado de alumnos matriculados en un año:

listado(N,Asig,A):- estudiantes(Nif,N),matriculaciones(Nif,Cod,_,A),
    asignatura(Cod,Asig,_).

%Asignatura con más créditos:

creditos(Cre):- group_by(
        asignatura(_,Asig,C),
        [],
        Cre=max(C)
).
asigCreditos(Asig):- asignatura(_,Asig,C),creditos(Cre),C=Cre.

%Número de alumos matriculados por año:

numeroMatriculados(A,Num):- group_by(
        listado(A,N,_),
        [A],
        Num=count(N)
```

```
).
%Nombre de alumno con máxima nota para un año y asignatura

notasAsigAnio(N,Asig,A,Nota):-asignatura(Cod,Asig,_),estudiantes(Nif,N),
    matriculaciones(Nif,Cod,Nota,A).

maximaNotaAsig(A,Asig,N):- group_by(
        notasAsigAnio(_,Asig,A,Nota),
        [Asig,A],
        N=max(Nota)
).

%Numero de matriculas de honor (nota igual a 10)

matricula(N,Asig,A):-notasAsigAnio(N,Asig,A,10).
numMatriculas(M):- group_by(
        matricula(N,Asig,A),
        [],
        M=count(N)
).
```

Práctica 2. Un cadena de supermercados necesita hacer 'minería de datos' sobre las compras de sus clientes. Básicamente, le interesa saber qué productos van asociados a las compras más caras y qué productos se compran casi siempre juntos.

Diremos que existe asociación entre dos productos: A y B, si B aparece, al menos, en el 70 % de los tickets en los que aparece A. Por sencillez, suponemos que cada producto aparece una vez en cada ticket.

La base de datos extensional tiene que tener el formato que se adjunta en el anexo. Diseña las siguientes consultas:

1. `coste_ticket(C,Total)`: `Total` es el coste total del ticket de código `C`. Ejemplo:

```
DES> coste_ticket(C,Total).
{
  coste_ticket(111,61),
  coste_ticket(222,33),
  coste_ticket(333,32),
  coste_ticket(444,34),
  coste_ticket(555,130)
}
Info: 5 tuples computed.
```

2. `ticket_mas_caro(C)`: `C` es el código del ticket más caro. Ejemplo:

```
DES> ticket_mas_caro(C).
{
  ticket_mas_caro(555)
}
Info: 1 tuple computed.
```

3. `productos_premium(P)`: `P` es un producto que aparece en aquellos tickets cuyo coste total es el doble que el coste medio de todos los tickets. Ejemplo:

```
DES> productos_premium(P).
{
  productos_premium(11),
  productos_premium(55)
}
Info: 2 tuples computed.
```

4. `asociacion(P1,P2)`: existe una asociación entre el producto `P1` y `P2`. Ejemplo:

```
DES> asociacion(P1,P2).
{
  asociacion(22,11),
  asociacion(22,33),
  asociacion(33,11),
  asociacion(33,22),
  asociacion(44,11),
  asociacion(44,22),
  asociacion(44,33),
  asociacion(44,55),
  asociacion(55,11),
  asociacion(55,22),
  asociacion(55,33),
  asociacion(55,44)
}
Info: 12 tuples computed.
```

Solución:

```
%Valor de un ticket

coste_item(Cod,Coste):- ticket(Cod,Cod2,U),precio_por_unidad(Cod2,P),Coste is U*P.
coste_ticket(C,Total):- group_by(
        coste_item(C,Coste),
        [C],
        Total=sum(Coste)
).
```

```
%Ticket mas caro
coste_mas_caro(Total):- group_by(
        coste_ticket(_,Total2),
        [],
        Total = max(Total2)
).
ticket_mas_caro(C):- coste_ticket(C,T),coste_mas_caro(Taux),T=Taux.
```

```
%Coste medio del ticket
coste_medio(Total):- group_by(
        coste_ticket(_,Total2),
        [],
        Total = avg(Total2)
).
```

```
%Tickets "premium"
tickets_premium(C):- coste_ticket(C,Total), coste_medio(Total2), Total > 2*Total2.
%Productos que aparecen en los tickets más caros
productos_premium(P):- ticket(C,P,_),tickets_premium(C).
```

```
%Asociaciones de productos
num_veces(P,Num):- group_by(
        ticket(C,P,U),
        [P],
        Num = count(U)
).
mismo_ticket(P1,P2,C):- ticket(C,P1,_),ticket(C,P2,_), P1 \= P2.
num_veces_cond(P2,P1,Num):- group_by(
        mismo_ticket(P2,P1,C),
        [P1,P2],
        Num = count(C)
).
asociacion(P1,P2):- num_veces(P1,N1),num_veces_cond(P2,P1,N2),
    Umbral is N1*0.7,N2 > Umbral, P1 \= P2.
```

Práctica 3. Consideremos la siguiente base de datos extensional que almacena información sobre una competición deportiva:

```
%equipo(id_equipo,nombre_equipo)
equipo(atm,'Atletico de Madrid').
equipo(bcn,'Barcelona').
equipo(rm,'Real Madrid').
equipo(sev,'Sevilla').

%itempartido(id_partido,id_equipoA,goles_A,id_equipoB,goles_B,espectadores)
itemPartido(1,atm,1,rm,0,30).
itemPartido(2,sev,2,bcn,0,15).
itemPartido(3,bcn,1,atm,1,34).
itemPartido(4,sev,1,rm,2,11).
itemPartido(5,atm,3,sev,0,50).
itemPartido(6,bcn,1,rm,2,11).
itemPartido(7,rm,1,atm,2,32).
itemPartido(8,bcn,2,sev,2,14).
itemPartido(9,atm,3,bcn,1,29).
itemPartido(10,rm,1,sev,3,11).
itemPartido(11,sev,3,atm,1,55).
itemPartido(12,rm,1,bcn,3,10).
```

Observaciones:

- El orden de aparición de `id_equipoA` y `id_equipoB` en `itemPartido` es relevante. `id_equipoA` es el equipo local y `id_equipoB` el equipo visisante. Por ejemplo, en `itemPartido(1,atm,1,rm,0,30) atm` es el equipo local y `rm` el equipo visitante.
- El número de espectadores se almacena en miles.

Diseña las siguientes reglas con las cabeceras que se indican:

- `partido(N1,G1,N2,G2)`:-..., se ha jugado un partido entre *N1* y *N2* con el resultado *N1 G2 - N2 G2*.
- `empate(P,N1,N2)`:-..., en el partido *P* ha habido un empate entre *N1* y *N2*.
- `asistenciaTotal(S)`:-..., *S* es el número total de asistentes en toda la competición.
- `numeroVitoriasLocales(L)`:-..., *L* es el número de victorias locales.
- `asistenciaMediaEquipoLocal(Id1,M)`:-..., para un identificador de equipo *Id1*, *M* es la asistencia media cuando actúa como local.
- `asistenciaMediaEquipoLocalNombre(N1,M)`:-..., para un nombre de equipo *N1*, *M* es la asistencia media cuando actúa como local.

Solución:

```
%0,5 puntos:
partido(N1,G1,N2,G2):-equipo(Id1,N1),equipo(Id2,N2),
                      itemPartido(_,Id1,G1,Id2,G2,_).

%0,5 puntos:
empate(P,N1,N2):-equipo(Id1,N1),equipo(Id2,N2),
                 itemPartido(P,Id1,G1,Id2,G2,_),G1=G2.

%1 punto:
asistenciaTotal(S):-group_by(
```

```
            itemPartido(I,N1,G1,N2,G2,A),
            [],
            S=sum(A)).

%1 punto:
numeroVitoriasLocales(L):- group_by(
            itemPartido(I,X,A,Y,B,_),A>B,
            [],
            L=count(I)
            ).

%1 punto:
asistenciaMediaEquipoLocal(Id1,M):- group_by(
            itemPartido(_,Id1,_,_,_,A),
            [Id1],
            M=avg(A)
            ).
asistenciaMediaEquipoLocalNombre(N1,M):- equipo(Id1,N1),
                                         asistenciaMediaEquipoLocal(Id1,M).
```