Bases de Datos
Curso 2015-2016
Grado en Ingeniería del Software
Examen Septiembre

Nombre: _____

Se debe entregar esta hoja

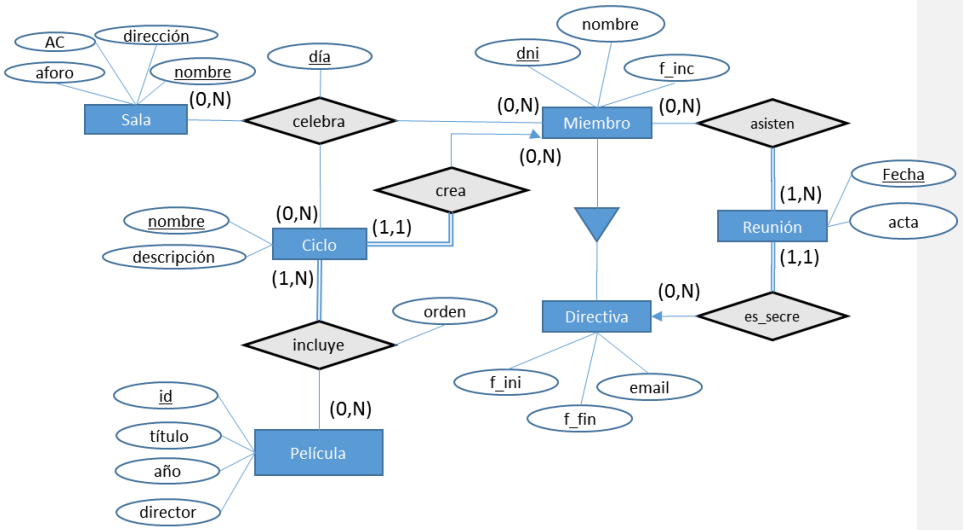
1**) (4 puntos) A partir de la información sobre la BD que se describe más abajo, se pide:**

a) **(2 puntos) El esquema entidad-relación, incluyendo atributos, claves, restricciones de cardinalidad y participación.**
b) **(1,5 puntos) Pasar al modelo relacional haciendo uso de las transformaciones apropiadas e indicando las restricciones de integridad referencial resultantes.**
c) **(0,5 puntos) Indicar otras restricciones de integridad del modelo que no se hayan podido implementar en los modelos anteriores, o que se hayan perdido en la transformación.**

Se desea diseñar la base de datos para un cine club de una ciudad. El cine club se encarga de proyectar ciclos de cine en distintos lugares de la ciudad y además la base de datos debe recoger información también de las reuniones entre los miembros del cine club. Esto es lo que nos piden reflejar en la BD.

- En primer lugar necesitamos guardar información de los miembros del cine club. Concretamente, guardaremos su nombre, su DNI y la fecha de incorporación.
- Como es natural también guardaremos información de las películas disponibles en el archivo del cine club. De cada película debemos almacenar en la base de datos un identificador, el título, el año en que fue estrenada y el nombre del director.
- En la base de datos también almacenaremos las salas disponibles de la ciudad. De cada una guardaremos el nombre, la dirección, su aforo y si tiene aire acondicionado.
- En este cine club cualquier miembro puede organizar un ciclo de películas. De cada ciclo se guarda su nombre (que será único), la descripción del mismo (es decir, sobre qué trata el ciclo), la lista de películas que componen el ciclo, incluyendo el orden el que serán proyectadas en el ciclo, y también qué miembro creó el ciclo. Los ciclos duran un único día y se celebran en una única sala, pero es habitual repetir algunos ciclos exitosos o reponer ciclos antiguos. Por ello, la base de datos debe registrar el sitio y la fecha donde se proyectó cada ciclo, junto con el miembro del cine club que moderó la tertulia posterior (no es necesariamente la misma persona que creó el ciclo y puede ser una persona distinta cada vez que se proyecta el ciclo).
- Por otro lado, algunos miembros del cine club pertenecen a la directiva del mismo. Necesitamos guardar la fecha de inicio y de fin del periodo en que dicho miembro forma parte de la directiva (la de fin en el momento en que se produzca) y su correo electrónico. Ten en cuenta que una persona que ha sido de la directiva no puede volver a serlo
- Por último, también tenemos que guardar información de las reuniones de los miembros del cine club. Debemos guardar la fecha y la hora en qué se celebró la

reunion, los miembros del cine club que asitieron a la misma, el acta (un objeto binario que se almacenará cuando sea escrita) y qué miembro de la directiva ejerció de secretario.



Una alternativa posible sería reflejar la relación entre Miembro y Directiva como una relación 1:1 con participación (0,1) y (1,1) respectivamente.

b) Modelo relacional

película(<u>id</u>, título, año, director)
sala(<u>nombre</u>, dirección, aforo, ac)
ciclo(<u>nombre</u>, descripción, creador)
pelis_ciclo(<u>ciclo,id_pelicula</u>, orden)
pases(<u>ciclo, sala,moderador, dia</u>) ó pases(<u>ciclo,día</u>,sala,moderador)
miembro(<u>dni</u>, nombre, f_inc)
directiva(<u>dni</u>, f_ini, f_fin*, email)
reunión(<u>fecha</u>, acta, secretario)
asistentes(<u>f_reu, asistente</u>)

$\prod_{creador}$(ciclo) <u>C</u> en $\prod_{dni}$(miembro)
$\prod_{dni}$(directiva) <u>C</u> en $\prod_{dni}$(miembro)
$\prod_{moderador}$(pases) <u>C</u> en $\prod_{dni}$(miembro)
$\prod_{secretario}$(reunión) <u>C</u> en $\prod_{dni}$(directiva)
$\prod_{ciclo}$(pases) <u>C</u> en $\prod_{nombre}$(ciclo)
$\prod_{sala}$(pases) <u>C</u> en $\prod_{sala}$(nombre)
$\prod_{asistente}$(asistentes) <u>C</u> en $\prod_{dni}$(miembro)
$\prod_{f\_reu}$(asistentes) <u>C</u> en $\prod_{fecha}$(reunión)

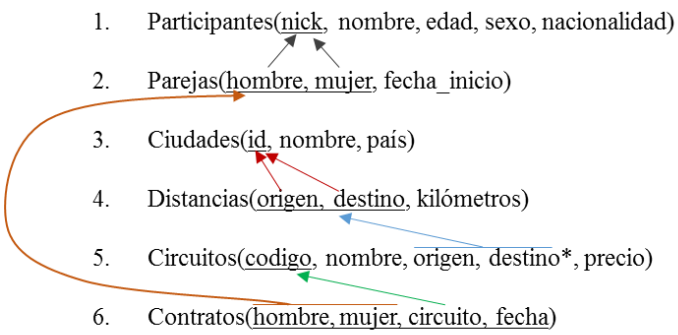
c) Restricciones de integridad perdidas

- El orden de las películas en un ciclo debe ser consecutivo y empezando en 1, como se desprende del enunciado.

- Que cada reunión tenga al menos un asistente (participación mínima de "miembro" en "asisten") y cada ciclo una película (participación mínima de "ciclo" en "incluye").
- En directiva la f_ini debe ser posterior a f_inc del correspondiente miembro en miembro.
- En directiva que f_fin sea posterior a f_ini.
- La fecha de las reuniones debe estar dentro del periodo en que el directivo que actúa de secretario está en activo (entre f_ini y f_fin).
- De manera similar con la proyección de un ciclo, la fecha en la que se proyecta debe ser posterior a la fecha de incorporación de dicho miembro. IDEM con la fecha en la que se proyecta el ciclo y la fecha de incorporación del miembro que la creó.

Estas restricciones de integridad de los datos no se pueden plasmar con el esquema relacional resultante, por lo que habrá que usar otros mecanismos como disparadores, para que en la base de datos se asegure el cumplimiento de estas restricciones y no se almacenen datos que las violen.

2**) (6 puntos) Dado el siguiente modelo relacional**



Y las siguientes restricciones:

a) El sexo se representa mediante un carácter: 'H' ⇔ Hombre, 'M' ⇔ Mujer.
b) La relación *Parejas* registra las parejas entre un hombre y una mujer en la base de datos junto con la fecha de inicio de dicha relación. El atributo 'hombre' se corresponde con el 'nick' de un participante de sexo masculino. El atributo 'mujer' se corresponde con el 'nick' de un participante de sexo femenino.
c) La relación *Distancias* registra la información de la distancia en kilómetros (entero positivo) entre dos ciudades. La distancia D entre las ciudades A y B ha de ser igual tanto si A actúa como origen o como destino. Por ejemplo, en la relación *Distancias* pueden existir las tuplas (A, B, 100) y (B, A, 100) o solo una de ellas.
d) La relación *Circuitos* recoge la información de los distintos circuitos vacacionales que puede contratar una pareja. No se admiten valores de precio negativo. Para circuitos con precio desconocido se registrará 0.
e) La relación *Contratos* registra los circuitos contratados por una pareja en una determinada fecha.

Comentado [JA1]: OJO: Aquí esto está bien, pero habría que aclarar si se guardan ambas en la base de datos o no. Y en caso de que fuese no, tendría que poner el where más elaboradoen plan: where (origen=A AND destino=B) OR (origen=B AND destino=A)

- **(1 puntos) Supuesto creadas las tablas 1, 2 y 3, escribe las sentencias SQL que permitan crear las tablas 4, 5 y 6, incluyendo las restricciones de integridad referencial y las otras restricciones de integridad descritas.**
- **(0.5 puntos) Indica cuáles son las restricciones de integridad que no has podido incluir en las sentencias de creación de las tablas.**
- **(0.75 puntos) Disparadores:**
  - Crea un disparador que sólo permita insertar tuplas válidas en la tabla Parejas según las restricciones impuestas anteriormente. Si la tupla a insertar no es válida, generar una excepción "RAISE_APPLICATION_ERROR", define el código y el mensaje de error asociado que consideres adecuado.

- **Realizar las siguientes consultas:**

a) (0.5) Se dice que una pareja es 'débil' si la diferencia de edad es mayor a 10 años. Listar los nombres de las parejas débiles (nombre del hombre, nombre de la mujer) y la diferencia de edad.
b) (0.5) Obtener el nombre de la ciudad que aparece como origen en el mayor número de circuitos.
c) (0.5) Obtener un listado de ciudades que no son destino de ningún circuito.

d) (0.75) Obtener un listado de circuitos cuyo precio sea superior al precio medio de los circuitos con el mismo origen.

~~d)~~e) (0.75) Listado de circuitos clasificados de la siguiente manera: Circuitos cortos (distancia entre la ciudad de origen y la ciudad de destino es inferior a 300 km), circuitos gran recorrido (distancia entre ciudades de origen y destino es superior a 2.000 km) y circuitos estándar (resto). Los resultados deben estar ordenados por tipo de circuito. La salida será algo del estilo:

Origen	Destino	Tipo circuito	Distancia
Roma	Nápoles	Corto	230
Madrid	Toledo	Corto	75
Madrid	Barcelona	Estándar	630
Génova	Rennes	Estándar	912
Madrid	Tokio	Gran recorrido	10.648

~~e)~~f)(0.75) Listado de los circuitos considerados rentables en el año 2016. Los circuitos rentables son aquellos cuyo importe facturado en un año supera los 100.000 euros. Ordenar por importe facturado.

Con formato: Numerado + Nivel: 1 + Estilo de numeración: a, b, c, … + Iniciar en: 1 + Alineación: Izquierda + Alineación: 0 cm + Tabulación después de: 0,63 cm + Sangría: 0,63 cm

```
create table distancias(
  origen      VARCHAR(10),
  destino     VARCHAR(10),
  kilometros INTEGER NOT NULL CHECK (kilometros>=0),
  PRIMARY KEY (origen, destino),
  FOREIGN KEY (origen) REFERENCES ciudades(id),
  FOREIGN KEY (destino) REFERENCES ciudades(id),
  Check (origen <> destino)
);


create  table circuitos(
  codigo      VARCHAR(5) PRIMARY KEY,
  nombre      VARCHAR(50) NOT NULL,
  origen      VARCHAR(10) NOT NULL,
  destino     VARCHAR(10) ,
  precio      NUMBER(4,2) default 0 CHECK (precio>0),
  FOREIGN KEY (origen, destino) REFERENCES distancias(origen, destino)
);


create  table contratos(
  hombre    VARCHAR(10) NOT NULL,
  mujer     VARCHAR(10) NOT NULL,
  circuito VARCHAR(5) NOT NULL,
```

```
  fecha    DATE NOT NULL,
  PRIMARY KEY (hombre, mujer, circuito, fecha),
  FOREIGN KEY (hombre, mujer) REFERENCES parejas(hombre, mujer),
  FOREIGN KEY (circuito) REFERENCES circuitos(codigo)
);
```

- Restricciones que no se han podido incluir:

En la tabla distancias, la distancia entre las ciudades A y B sea igual a la distancia entre B y A.

Que la fecha del contrato sea posterior a la fecha de inicio de la pareja.

- Trigger:

```
CREATE OR REPLACE TRIGGER t_parejas
before INSERT ON parejas
     FOR EACH ROW
DECLARE
  var_h participantes.sexo%TYPE;
  var_m participantes.sexo%TYPE;
BEGIN
     select sexo into var_h
        from participantes
        where nick = :new.hombre;
     select sexo into var_m
        from participantes
        where nick = :new.mujer;
     IF var_h == var_m THEN
          RAISE_APPLICATION_ERROR(-20000, 'NICKS INCORRECTOS');
     END IF;
 END;
```

```
--Consultas:

-- a)
select P1.nombre, P2.nombre, abs(P1.edad - P2.edad)
from parejas P, participantes P1, participantes P2
where P.hombre = P1.nick and P.mujer = P2.nick and
      abs(P1.edad - P2.edad) > 10;

-- b)
select ciudades.nombre, count(*)
from circuitos C JOIN ciudades ON origen=id
group by ciudades.nombre
having count(*) >= all (select count(*)
                        from  circuitos Cir
                        group by Cir.origen
                                );

-- c) primera versión
SELECT  id, NOMBRE
FROM CIUDADES
WHERE id not IN (SELECT DESTINO
                 FROM CIRCUITOS
                 where destino is not null);
-- En la consulta anterior podemos usar también
```

```
--  WHERE id <> ALL (...

-- c) segunda versión. Consulta correlacionada
SELECT  id, NOMBRE
FROM CIUDADES C
WHERE not exists (SELECT DESTINO
                  FROM CIRCUITOS Cir
                  where Cir.destino =  C.id);


-- d)
select *
from circuitos Cir
where precio > (select avg(precio)
                from circuitos C
                                where C.origen = Cir.origen);

-- e)
select C.codigo, C.origen, C.destino, case when D.kilometros < 300
then 'Corto'
            when D.kilometros > 2000 then 'Largo'
            else 'Std'
            end as Tipo , D.KILOMETROS
      from circuitos C, distancias D
      where (D.origen = C.origen and D.destino = C.destino)
union
select C.codigo, C.origen, C.destino, 'Corto', 0
      from circuitos C
      where C.destino is null;

-- OJO: en circuitos no está el nombre de las ciudades. Si queremos
ofrecerlo como en el modelo, debemos incluir en el from del primer
select dos veces la tabla ciudad (una como origen y otra como
destino) para hacer dos reuniones con circuito y poder así extraer
los nombres.



-- f)

select C.circuito, C.nombre, sum(precio) as Importe
  from contratos C, circuitos Cir
  where C.fecha BETWEEN TO_DATE ('2016/01/01', 'yyyy/mm/dd')
        AND TO_DATE ('2016/12/31', 'yyyy/mm/dd') and Cir.codigo =
C.circuito
  group by C.circuito, C.nombre
  having sum(precio) > 100000
order by importe;
```