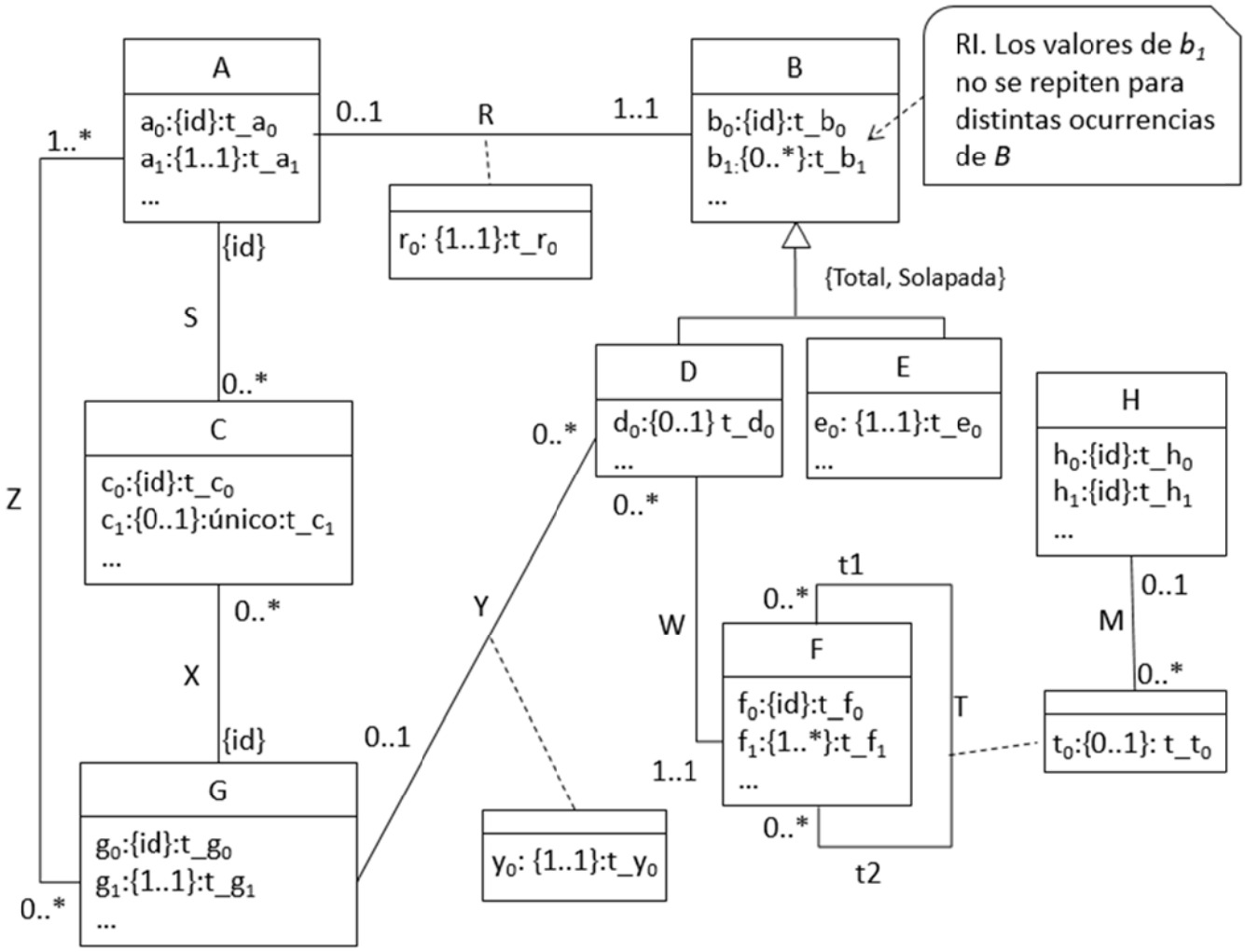
EXAMEN DE BASES DE DATOS Y SISTEMAS DE INFORMACIÓN 10/01/2018

PARCIAL 2: UNIDADES DIDÁCTICAS 4 Y 5

1. Realice el diseño lógico del siguiente diagrama de clases en UML para obtener un conjunto equivalente de relaciones del modelo relacional. Las restricciones que no pueda expresar en el esquema relacional, escríbalas en lenguaje natural **(1'5 puntos).**



2. Sea el siguiente esquema de relación:

R(A: int, B: int, C: char, D: int, E: txt, F: char, G: int, H: char)
CP: {A, B, C}
VNN: {D, E, F, G, H}

A partir de las dependencias que aparecen a continuación, transforme la relación a un conjunto de relaciones en tercera forma normal **(0'5 puntos).**

{A} → {F} {B, C} → {D} {F} → {G} {D} → {H}

3. Diseñe un diagrama de clases en UML para el sistema de información que se describe a continuación. Las restricciones que no se puedan expresar gráficamente, escríbalas en lenguaje natural. **(1'5 puntos)**

Se desea guardar la información de las distintas ediciones de un circuito de carreras populares. Cada año se celebra una edición del circuito que consta de 10 carreras. El conjunto de carreras que forman parte del circuito puede variar de una edición a otra. El personal técnico de la organización será el encargado de dar de alta en el sistema cada nueva edición del circuito usando para ello el año de la edición (que la identifica). Además, se asignará a la edición del circuito las carreras que lo componen ese año. Si alguna de las carreras es nueva, se dará de alta en el sistema con una denominación, que es única, y una ubicación que es obligatoria. Cada edición de una carrera se identifica por un número de edición que es único para cada carrera, además se debe conocer la fecha y la hora de celebración, una distancia en metros y la entidad organizadora. De la entidad organizadora interesa conocer su nombre, que se utilizará para identificarla, así como una persona de contacto: DNI, nombre, apellidos, teléfonos y correo electrónico (todos estos valores serán obligatorios).
Los corredores se inscriben a una edición del circuito pagando un precio de inscripción antes de la finalización del plazo de inscripción de la edición. La inscripción es válida para todas las carreras del circuito. Tanto el precio como el plazo de inscripción deben conocerse para cada edición del circuito. Durante la inscripción, el corredor tendrá que introducir sus datos personales (DNI, nombre, apellidos, fecha de nacimiento y sexo), datos de contacto (teléfonos y correo electrónico) y los datos de su tarjeta de crédito (que no puede ser usada por otro corredor). Todos estos datos son obligatorios. Si todo funciona correctamente, el sistema asignará al corredor la categoría (en función de su sexo y edad en el momento de la inscripción), y su número de dorsal y de chip asignados para esa edición del circuito.
Los corredores se podrán inscribir a cada edición como miembros de un club. Del club se almacenará su nombre y su representante. Algunos clubs son federados y disponen de una identificación federativa y una razón social. Durante una edición del circuito un corredor no podrá cambiar de club, pero sí podrá inscribirse a través de distintos clubs en distintas ediciones del circuito.
El personal técnico publicará los resultados de cada carrera al finalizar la misma. Para cada corredor participante se indicará su posición en la clasificación general de la carrera, su posición en la categoría, tiempo oficial, tiempo real, ritmo medio por km oficial y real. La clasificación podrá ser consultada tanto por los corredores como por el personal técnico de la organización.

4. Dado el siguiente esquema de una base de datos

```
CREATE TABLE Monitor (
DNI   CHAR(10)   CONSTRAINT   pk_monitor   PRIMARY   KEY DEFERRABLE,
nombre VARCHAR(50) NOT NULL,
telefono CHAR(10)  NOT NULL,
num INTEGER DEFAULT 0 NOT NULL)

CREATE TABLE Viaje (
id char(5)  CONSTRAINT  pk_viaje  PRIMARY  KEY  DEFERRABLE,
DNI CHAR(10) CONSTRAINT  fk_viaje_monitor REFERENCES monitor(DNI)  DEFERRABLE,
fecha DATE NOT NULL)
```

Donde:

- La relación **Monitor** almacena los datos del monitor: *DNI*, *nombre*, *teléfono* y *num*, el número de viajes que tiene asignado el monitor.
- La relación **Viaje** almacena la información de un viaje: *id* del viaje, *DNI* del monitor asignado al viaje y *fecha* del viaje.

Implementar un disparador (*trigger*) en PL-SQL de Oracle para mantener actualizado el atributo *num* del monitor para la operación de insertar un nuevo viaje. **(0'5 puntos)**

SOLUCIONES:

1.

A(a0:t_a0, a1:t_a1, …, b0:t_b0, r0:t_r0)
CP:{a0}
VNN:{a1, b0, r0}
CAj:{b0} → B
Uni:{b0}

B(b0:t_b0,…,)
CP:{b0}

B1(b1:t_b1, b0:t_b0)
CP:{b1}
VNN:{b0}
CAj:{b0} → B

C(c0:t_c0, c1:t_c1, …, a0:t_a0, g0:t_g0)
CP{c0, a0, g0}
Uni:{c1}
CAj:{a0} → A
CAj:{g0} → G

D(b0:t_b0, d0:t_d0, …, f0:t_f0)
CP:{b0}
CAj:{b0} → B
CAj:{f0} → F(f0)
VNN:{f0}

E(b0:t_b0, e0:t_e0, …)
CP:{b0}
CAj:{b0} → B
VNN:{e0}

G(g0:t_g0, g1:t_g1, …)
CP{g0}
VNN:{g1}

H(h0:t_h0, h1:t_h1, …)
CP{h0, h1}

F(f0:t_f0, …,)
CP{f0}

F1(f1:t_f1, f0:t_f0)
CP:{f1, f0}
CAj:{f0} → F

Y(g0:t_g0, b0:t_b0, y0:t_y0)
CP:{b0}
CAj:{g0} → G(g0)
CAj:{b0} → D(b0)
VNN:{g0, y0}

Z(a0:t_a0, g0:t_g0)
CP{a0, g0}
CAj:{a0} → A(a0)
CAj:{g0} → G(g0)

T(f0_t1:t_f0, f0_t2:t_f0, t0:t_t0, h0:t_h0, h1:t_h1)
CP{f0_t1, f0_t2}
CAj:{f0_t1} → F(f0)
CAj:{f0_t2} → F(f0)
CAj:{h0, h1} → H(h0, h1)

$RI_{min\_G\_en\_Z}$: Todos los valores que aparezcan en el atributo g0 de G deben aparecer en el atributo g0 de Z.

RI_{min_f1}: Todos los valores que aparezcan en el atributo f0 de F deben aparecer en el atributo f0 de F1.

RI_{Total}: Los valores que aparezcan en el atributo b0 de B debe aparecer en el atributo b0 de D o en el b0 de E.

2.

R(A: int, B: int, C: char, E: txt)
CP: {A, B, C}
VNN: {E}
CAj: {A}→R1
CAj: {B,C}→R2

R1(A: int, F: char)
CP: {A}
VNN: {F}
CAj: {F}→R12

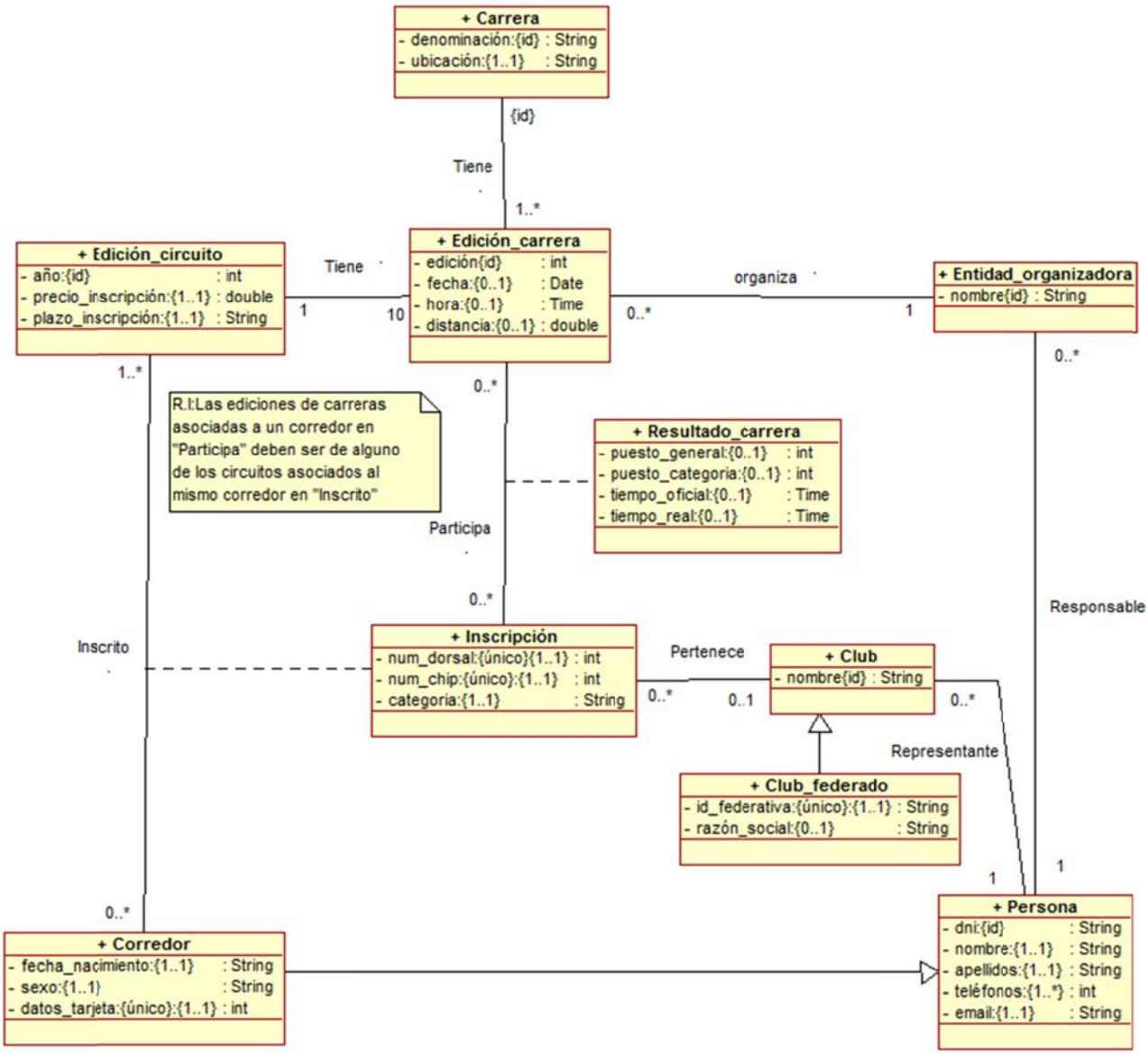
R12(F: char, G: int)
CP: {F}
VNN: {G}

R2(B: int, C: char, D: int)
CP: {B, C}
VNN: {D}
CAj: {D}→R22

R22(D: int, H: char)
CP: {D}
VNN: {H}

Todo valor de A en R1 debe de existir en R.
Todo valor de F en R12 debe de existir en R1.
Todo par de valores de B y C en R2 debe de existir en R.
Todo valor de D en R22 debe de existir en R2.

3.



4.

```
CREATE OR REPLACE TRIGGER T_ins_viaje
AFTER INSERT ON viaje
FOR EACH ROW
WHEN new.DNI IS NOT NULL
BEGIN
UPDATE Monitor SET num = num+1
WHERE DNI = :new.DNI;
END;
```