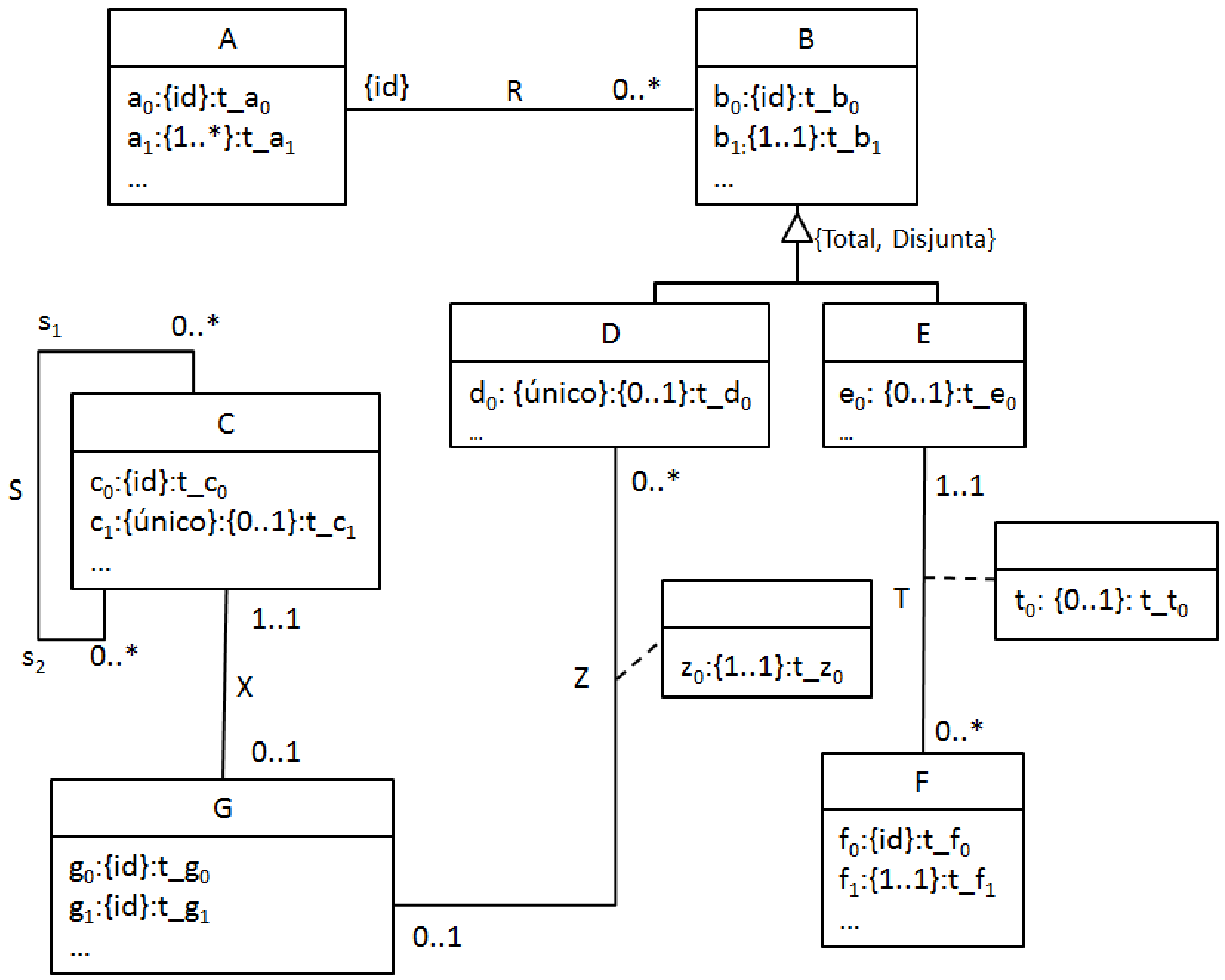
UNITS 4 AND 5

1. Perform the logical design of the following UML class diagram in order to obtain the equivalent set of relations in the relational model. Those constraints that cannot be expressed graphically can be written in natural language (English or Spanish). (Where “único” means “unique” and “Disjunta” means “Disjoint).

(1.5 points).



2. Design a UML class diagram for the following information system. Those constraints that cannot be expressed graphically can be written in natural language (English or Spanish).
(1.5 points)

We want to design a database for a car rental company.

The company has several branch offices around the country. Each branch office has a code, which identifies it, and one address (two offices cannot share the same address). Each branch office has different vehicles to be rented. We want to know the license plate of each vehicle, which identifies it, its brand, and its model. There are two types of vehicles: cars (we know how many seats has each car) and vans (we know the cargo capacity of each van). A vehicle is classified according to its category. Each category has a name (its identifier) and a daily rental price.

In the branch offices there are employees (identified by their SSN), with a name, a surname, home address, city, zip code and company position. All this information is required.

Customers can make reservations. Customers are identified by their SSN. Their name, surname, home address, city, zip code, date of driver's license, and number of one credit card are also required for all the customers.

Each reservation is coded with a unique value. To make a reservation we also need information about the customer, the category of the vehicle, the date and office of pickup, and the date and office of return. All this information is mandatory. A customer cannot make two reservations with the same pickup date.

When a customer pickup the vehicle (delivery), we must record the delivery employee, the assigned vehicle, and the km and the amount of allocated fuel. Note that each delivery corresponds to a single reservation.

Finally, when the vehicle is returned, we must register the employee receiving the car, the final km, the amount of fuel remaining, and a list of possible damages suffered by the vehicle. Note that each return corresponds to a single delivery.

3. Consider the following relational schema:

   R(A: char, B: char, C: int, D: char, E: int, F: char, G: int, H: char)

   PK: {A, B, C}

   NNV: {D, E, F, G, H}

   From the dependencies shown below, transform the relation to a set of relations in third normal form.

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

(0.5 points).

4. Consider the following relational schema:

```
CREATE TABLE Subject (
   code CHAR(5) CONSTRAINT pk_subject PRIMARY KEY DEFERRABLE,
   name VARCHAR(50) NOT NULL,
   description  VARCHAR(50)  NOT NULL,
   minimum_age INTEGER NOT NULL);

CREATE TABLE Student (
   st_code char(5) CONSTRAINT pk_student PRIMARY KEY DEFERRABLE,
   name VARCHAR(50) NOT NULL,
   age INTEGER NOT NULL,
   subject CHAR(10) NOT NULL
     CONSTRAINT fk_student_subject REFERENCES Subject(code)
                  DEFERRABLE);
```

Where:

- The **Subject** relation stores the information related to one subject: code, name, description and the minimum age required to be enrolled in the subject.
- The **Student** relation stores the information relating one student to the subjects he/she is interested in: student code (st_code), student name, student age, and the code of the subject he/she is interested in.

Implement a trigger in Oracle PL-SQL that will be activated when a new student is inserted. The trigger will check that the student age is valid for that subject. If the student age in lower than the minimum age for the subject, the trigger will prevent the insertion of the student.

(0.5 points).

1.-

A(a0:t_a0, …)
CP:{a0}

A1(a0:t_a0, a1:t_a1)
CP:{a0, a1}
CAj:{a0} → A

RI$_{\text{MinA1}}$: Todo valor que aparezcan en el atributo a0 de A debe aparecer en el atributo a0 de A1.

B(a0:t_a0, b0:t_b0, b1:t_b1,…,)
CP:{a0, b0}
CAj:{a0} → A
VNN:{b1}

D(a0:t_a0, b0:t_b0, d0:t_d0, …)
CP:{a0, b0}
CAj:{a0, b0} → B
Uni:{d0}

E(a0:t_a0, b0:t_b0, e0:t_e0, …)
CP:{a0, b0}
CAj:{a0, b0} → B

RI$_{\text{Disjunta}}$: Ningún par de valores que aparezca en los atributos {a0, b0} de D puede aparecer en valores del par de atributos {a0, b0} de E.

RI$_{\text{Total}}$: Todo par de valores que aparezca en los atributos {a0, b0} de B debe aparecer en los atributos {a0, b0} de D o de E**.**

F(f0:t_f0, f1:t_f1, …, a0:t_a0, b0:t_b0, t0: :t_t0)
CP{f0}
VNN:{f1}
CAj:{a0, b0} → E
VNN:{a0, b0}

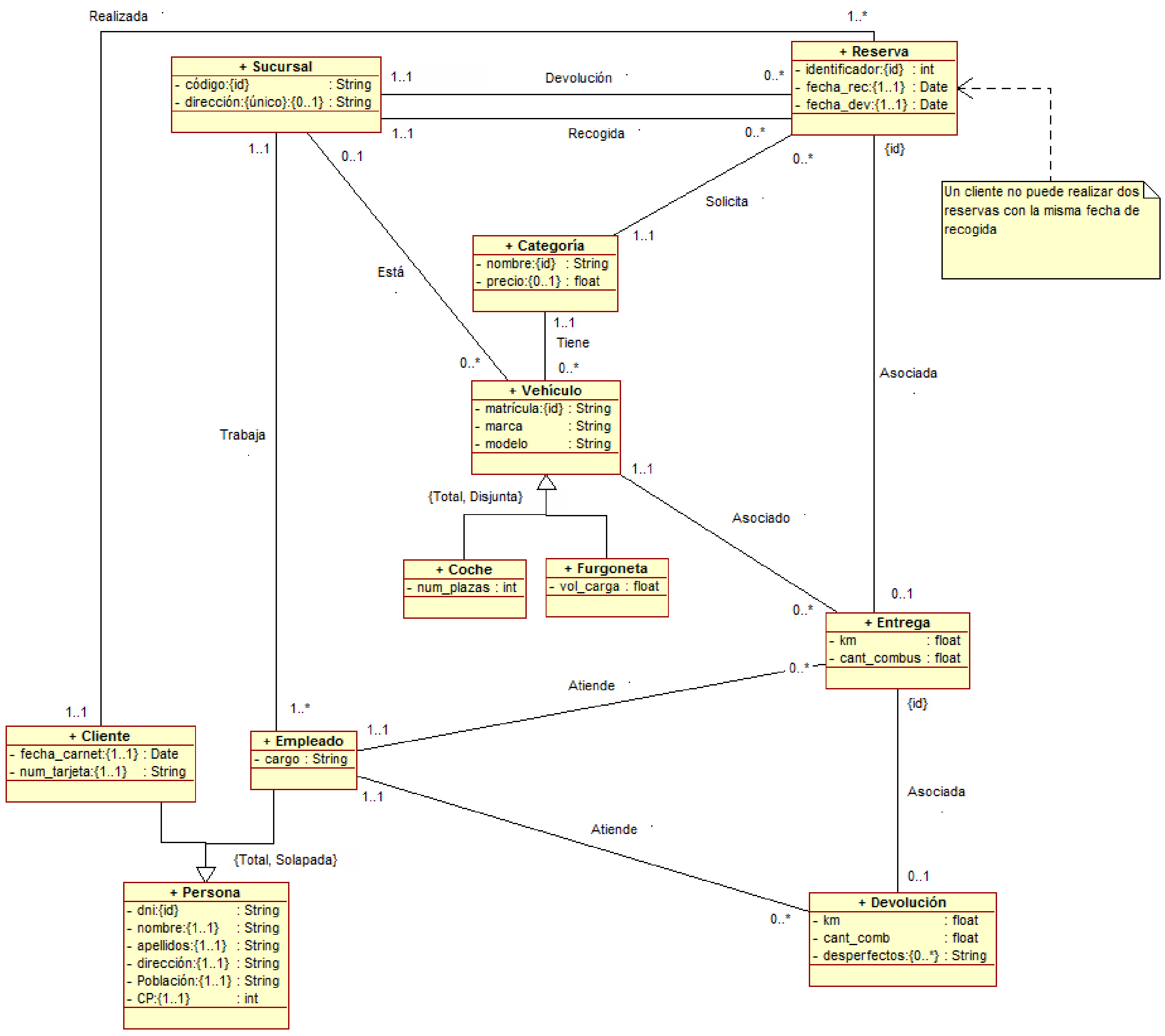
G(g0:t_g0, g1:t_g1, …, c0:t_c0)
CP{g0, g1}
Uni:{c0}
VNN:{c0}
CAj:{c0} → C

Z(a0:t_a0, b0:t_b0, g0:t_g0, g1:t_g1, z0:t_z0)
CP{a0, b0}
VNN:{z0}
CAj:{g0, g1} → G
CAj:{a0, b0} → D
VNN:{g0, g1}

C(c0:t_c0, c1:t_c1, …)
CP{c0}
Uni:{c1}

S(c0_s1:t_c0, c0_s2:t_c0)
CP:{c0_s1, c0_s2}
CAj:{c0_s1} → C(c0)
CAj:{c0_s2} → C(c0)

2.-


Realizada
1..*
+ Sucursal
- código:{id} : String
- dirección:{único}:{0..1} : String
1..1
Devolución
0..*
1..1
Recogida
0..*
+ Reserva
- identificador:{id} : int
- fecha_rec:{1..1} : Date
- fecha_dev:{1..1} : Date
{id}
Un cliente no puede realizar dos reservas con la misma fecha de recogida
1..1
0..1
0..*
Solicita
1..1
Está
+ Categoría
- nombre:{id} : String
- precio:{0..1} : float
1..1
Tiene
0..*
0..*
+ Vehículo
- matrícula:{id} : String
- marca : String
- modelo : String
Trabaja
Asociada
1..1
{Total, Disjunta}
Asociado
+ Coche
- num_plazas : int
+ Furgoneta
- vol_carga : float
0..1
0..*
+ Entrega
- km : float
- cant_combus : float
0..*
Atiende
{id}
1..1
1..*
1..1
+ Cliente
- fecha_carnet:{1..1} : Date
- num_tarjeta:{1..1} : String
+ Empleado
- cargo : String
1..1
Asociada
Atiende
{Total, Solapada}
0..1
+ Persona
- dni:{id} : String
- nombre:{1..1} : String
- apellidos:{1..1} : String
- dirección:{1..1} : String
- Población:{1..1} : String
- CP:{1..1} : int
0..*
+ Devolución
- km : float
- cant_comb : float
- desperfectos:{0..*} : String

3.-

R(A: char, B: char, C: int, G: int)

PK: {A, B, C}

NNV: {G}

FK:{A} → R1

FK:{B, C} → R2

FK:{G} → R4

R1(A: char, D: char)

PK: {A}

NNV: {D}

R2(B: char, C: int, E: int)

PK: {B, C}

NNV: {E}

FK:{E} → R3

R3(E: int, F: char)

PK: {E}

NNV: {F}

R4(G: int, H: char)

PK: {G}

NNV: {H}

Al value of A in R1 must be in R
Al value B,C in R2 must be in R
Al value of E in R3 must be in R2
Al value of G in R4 must be in R

4.-

```
CREATE OR REPLACE TRIGGER T_ins_student
BEFORE INSERT ON Student
FOR EACH ROW
DECLARE
    minim INTEGER;
BEGIN
    SELECT minimum_age INTO minim
    FROM Subject WHERE code = :new.subject;
IF minim > :new.age THEN
    RAISE_APPLICATION_ERROR(-20000, ‘The student
    agecannot be lower than the subject minimum age’);
 END IF;
END;
```