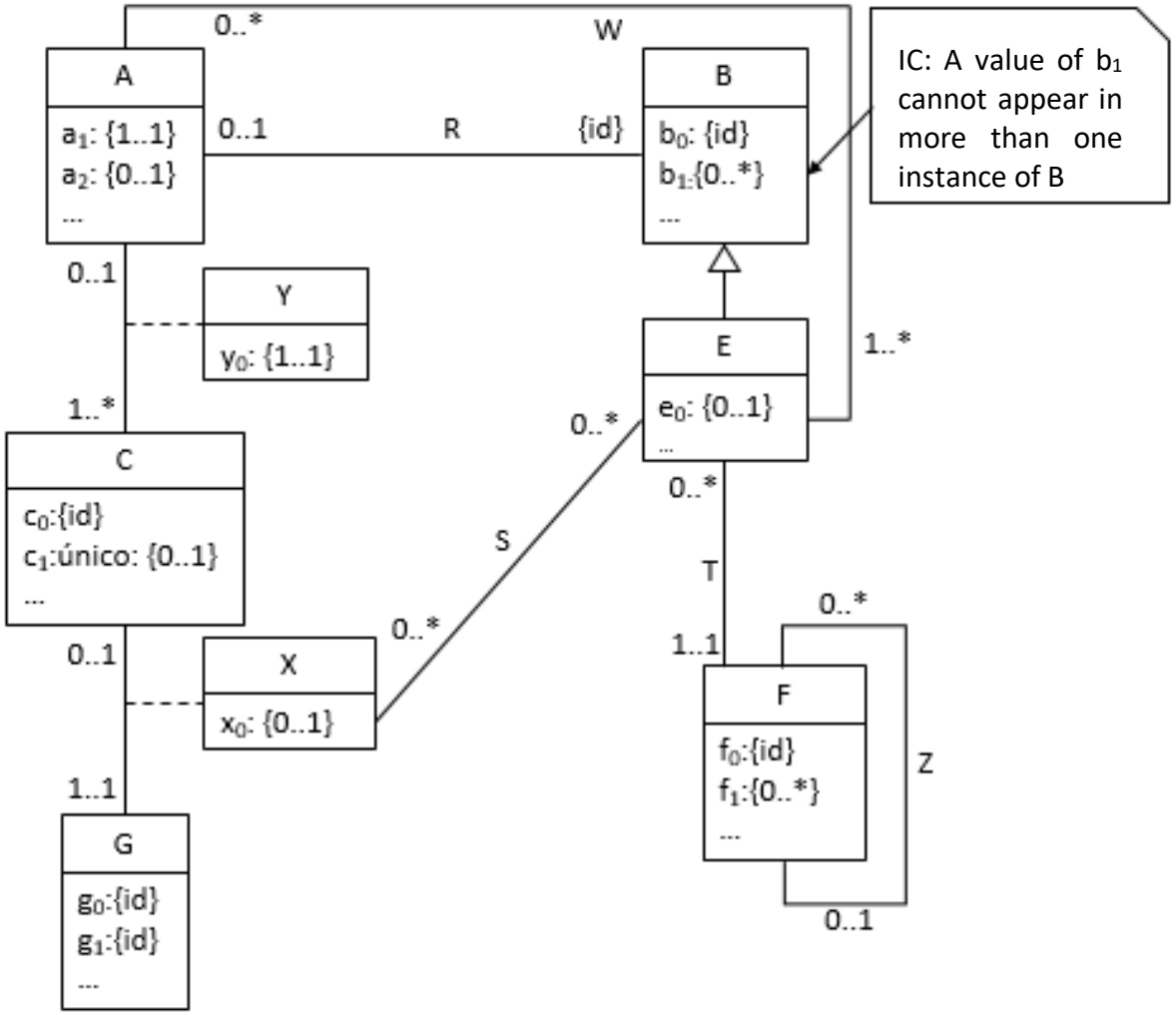
BASES DE DATOS Y SISTEMAS DE INFORMACIÓN" EXAM 23/01/2018

UNITS 4 AND 5

1. **(1.5 points).** 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). Note that it is not necessary to indicate the domains of the attributes.



2. **(0.5 points).** Consider the following relational schema:

R(A: A_type, B: B_type, C: C_type, D: D_type, E: E_type)

PK: {A, B}
NNV: {C, D, E}

A_type: char(10);
B_type: int;
E_type: record of {F: char(10), G: char(10)}
C_type: set of integers ;
D_type: char(10);

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

{B} → {D} {F} → {G}

3. **(1.5 points)** 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).

The section of Monumental Trees of the city wishes to design a database to manage the pruning and maintenance of those trees, then the information system is described.

Monumental trees are uniquely coded and some of them are known by a name that is also unique. It is mandatory to know its age, its location and the species to which it belongs. The species are also encoded and their scientific name (which is unique) must be known. If the species has a common name it must be also stored. It is necessary to have information about how the trees of the species should be cared for.

Tree pruning is planned in one-day sessions. Each one-day session includes a set of trees (at least one). For each pruning it is mandatory to know the code (which is unique), the day it will be carried out, and the crew of gardeners who will carry out the work.

Each crew of gardeners is numbered in a unique way, has a level and consist of between 1 and 5 gardeners of which one is the supervisor (a gardener belongs to one crew at most). Each gardener has a worker number (which is unique), a name, a bank account number, an address, and a telephone number. Crews can not be assigned to two prunings on the same day.

The material available (items) is coded, has a name (lifting baskets, mechanical saws, trucks to collect waste, ...) and it is important to know how it works and how many units of each item there are. The items can be reserved for a pruning session indicating the reserved quantity of each of item. It is not possible to reserve more units of an item than exists.

In addition to programmed pruning, the trees can suffer incidents (falling branches, fires, ...) that require an unprogrammed work. For each incident, which is identified by a number that is unique for each day, we want to know the tree or trees involved and a description of the problem. If necessary, one of the gardeners who is not part of any crew can prepare a report of one tree. The report must contain the affected tree, the incident, the date and the result of the visual analysis. It is possible that this analysis leads to possible economic compensation for a citizen who has been harmed by the incident. We must store for each compensation its code (which identifies it) the economic compensation, and the citizen who will receive it (ID, name and email).

4. **(0.5 points)** Consider the following relational schema:

```
CREATE TABLE Assistant (
SSN CHAR(10)CONSTRAINT pk_assistant PRIMARY KEY DEFERRABLE,
name VARCHAR(50) NOT NULL,
phone CHAR(10) NOT NULL,
num INTEGER DEFAULT 0 NOT NULL)

CREATE TABLE Trip (
id char(5) CONSTRAINT pk_trip PRIMARY KEY DEFERRABLE,
SSN CHAR(10) CONSTRAINT fk_trip_asst REFERENCES Assistant(SSN) DEFERRABLE,
tdate DATE NOT NULL)
```

where:

- The **ASSISTANT** relation stores the trip assistants data: *SSN*, *name*, *phone,* and *num* (where *num* contains the total amount of trips assigned to the assistant).
- the **TRIP** relation stores the information associated to the trips: *id* of the trip, *SSN* of the assistant assigned to the trip, and the date (t*date)* of the trip.

Implement a trigger in Oracle PL-SQL to check that the value of the *num* attribute in the ASSISTANT relation is adecuate (it has the necesary value) when a new ASSISTANT is INSERTED.

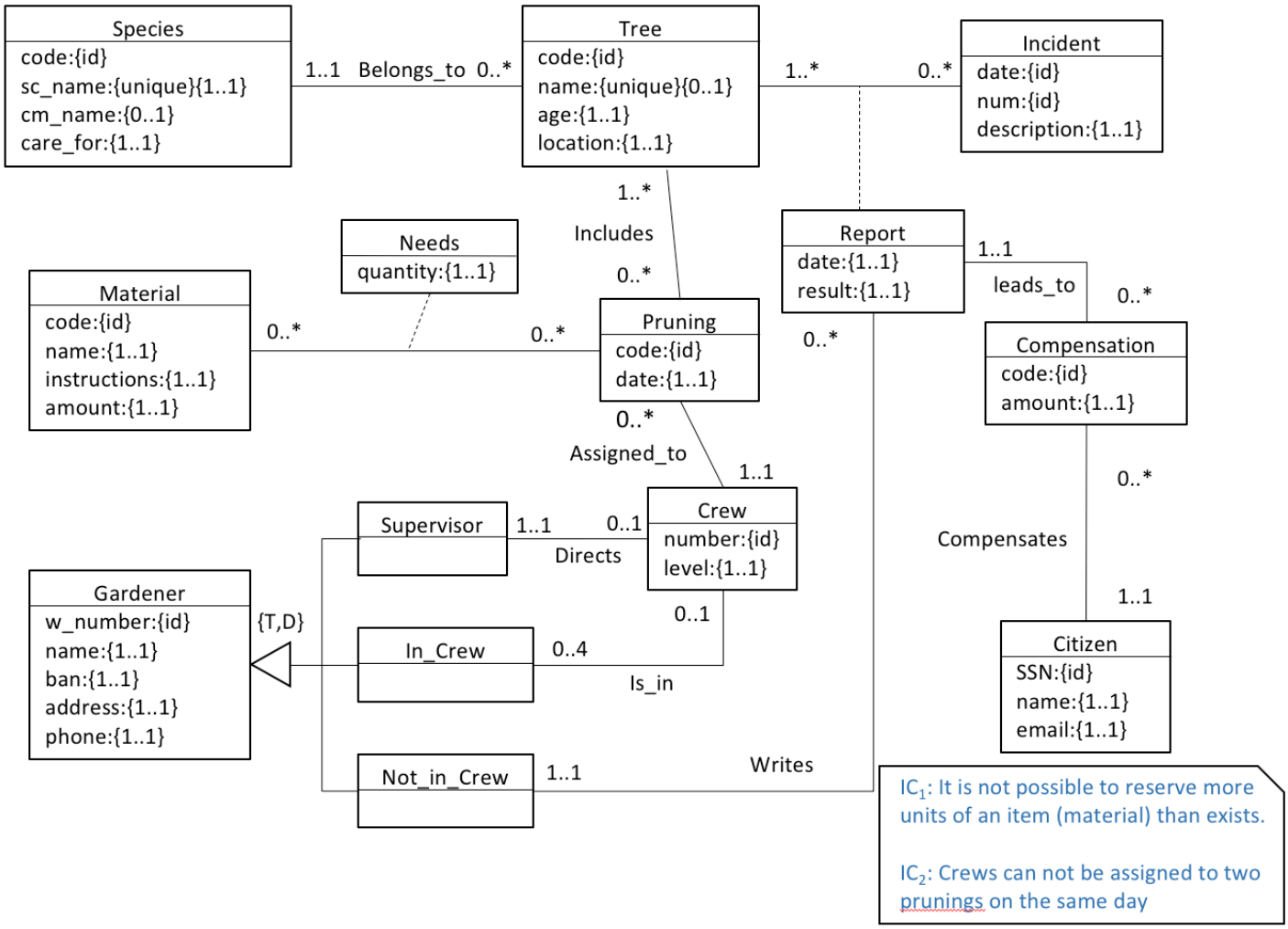
1.-

$B(b_0, \ldots)$ PK: $\{b_0\}$ $B\text{-}B1(b_0, b_1)$ PK: $\{b_1\}$ FK: $\{b_0\} \rightarrow B(b_0)$ NNV: $\{b_0\}$	$C(c_0, c_1, \ldots, g_0, g_1, x_0)$ PK: $\{c_0\}$ Unique: $\{c_1\}$ FK: $\{g_0, g_1\} \rightarrow G(g_0,g_1)$ NNV: $\{g_0, g_1\}$ Unique: $\{g_0, g_1\}$ G(g0, g1,...} PK: {g0, g1}
$A(b_0, a_1, a_2, \ldots)$ PK: $\{b_0\}$ FK: $\{b_0\} \rightarrow B(b_0)$ NNV: $\{a_1\}$ $Y(b_0, c_0, y_0)$ PK: $\{c_0\}$ NNV: $\{b_0\}$ FK: $\{b_0\} \rightarrow A(b_0)$ FK: $\{c_0\} \rightarrow C(c_0)$ NNV: $\{y_0\}$ $IC_{min\_A\_in\ Y}$: All value of b_0 in A must appear in b_0 of Y.	$E(b_0, e_0, \ldots, f_0)$ PK: $\{b_0\}$ FK: $\{b_0\} \rightarrow B(b_0)$ FK: $\{f_0\} \rightarrow F(f_0)$ NNV: $\{f_0\}$ $S(b_0, c_0)$ PK: $\{b_0, c_0\}$ FK: $\{b_0\} \rightarrow E(b_0)$ FK: $\{c_0\} \rightarrow C(c_0)$
$F(f_0, \ldots, f_0\_Z$ PK: $\{f_0\}$ FK: $\{f_0\_Z\} \rightarrow F(f_0)$ $F\text{-}F1(f_0, f_1)$ PK: $\{f_0, f_1\}$ FK: $\{f_0\} \rightarrow F(f_0)$	$W(b_0, b_0\_A)$ PK: $\{b_0, b_0\_A\}$ FK: $\{b_0\} \rightarrow E(b_0)$ FK: $\{b_0\_A\} \rightarrow A(b_0)$ $IC_{min\_A\_in\ W}$: All value of b_0 in A must appear in b_0 of W.

2.-

R(A: A_type, B: B_type, F: F_type) PK: {A, B} NNV: {F} FK: {B} → R-D (B) FK: {F} → R-G (F)	R-D(B: B_type, D: D_type) PK: {B} NNV: {D}
R-G(F: F_type, G: G_type) PK: {F} NNV: {G}	R-C(A: A_type, B: B_type, C: integer) PK: {A, B, C} FK: {A, B} → R(A, B)
IC: • All pair (A, B) in R must be in R-C • All value of B in R-D must be in R • All value of B in R-F must be in R	

3.-



4.-

```
/*This trigger can be BEFORE or AFTER/*

CREATE OR REPLACE TRIGGER T_ins_assist
BEFORE INSERT ON Assistant
FOR EACH ROW
WHEN new.num <> 0
BEGIN
  RAISE_APPLICATION_ERROR(…);
END;


/* This solution can only be BEFORE/*

CREATE OR REPLACE TRIGGER T_ins_assist
BEFORE INSERT ON Assistant
FOR EACH ROW
WHEN new.num <> 0
BEGIN
  :new.num:=0;
END;
```