Resolução Exame de Recurso 2017/18

Grupo I.1

I.1.1.a)

I.1.1.b) Solução sem serem permitidos nulos

Stands(código_i,nome,morada,abertura,fecho)
Oficinas(código_i,nome,morada,abertura,fecho)

Veículos(matrícula,marca,modelo,cor)

Vendas(matrícula,código_i)
Chave estrangeira: código_i referindo Stands(código_i)
Notação alternativa para indicar chave estrangeira código_i -> Stands(código_i)

Marcações(código_i,num,data,hora,matrícula)
CEs: código_i -> Oficinas(código_i) e matrícula -> Veículos(matrícula)

Reparações(código_r, data, hora, matrícula, código_i)
CEs: código_i -> Oficinas(código_i) e matrícula -> Veículos(matrícula)

Reparações_Marcadas(código_r, código_i, num)
CEs: código_r -> Reparações(código_r) e (código_i, num) -> Marcações(código_i, num)

Mecânicos(código_m, nome, especialidade)

Horas_reparações(código_m, código_r, horas)
CEs: código_r -> Reparações(código_r) e código_m -> Mecânicos (código_m)

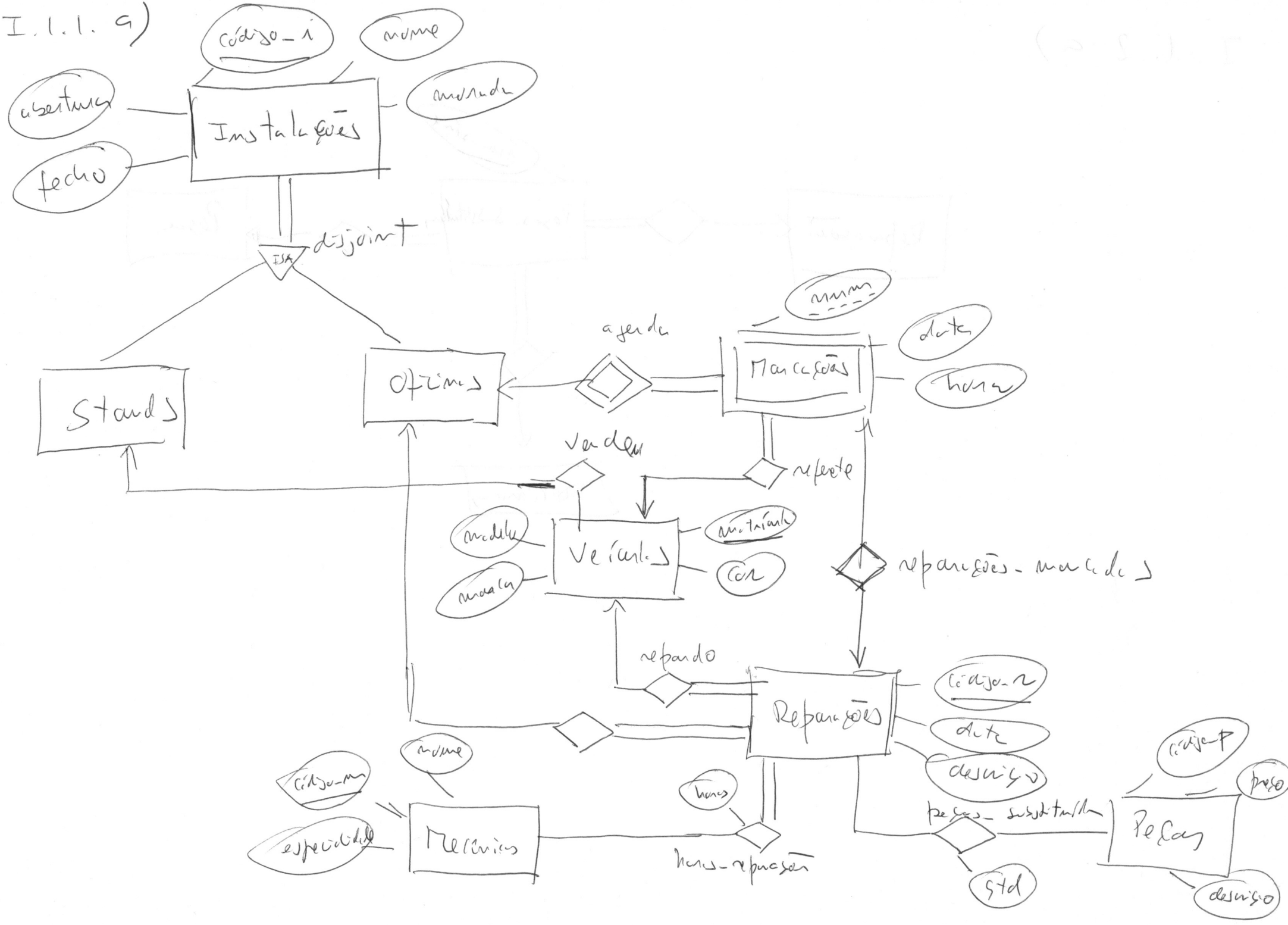
Peças(código_p, descrição,preço)
Peças_substituídas(código_p, código_r, qtd)
CEs: código_r -> Reparações(código_r) e código_p -> Peças (código_p)

I.1.2.a)

I.1.2.b)

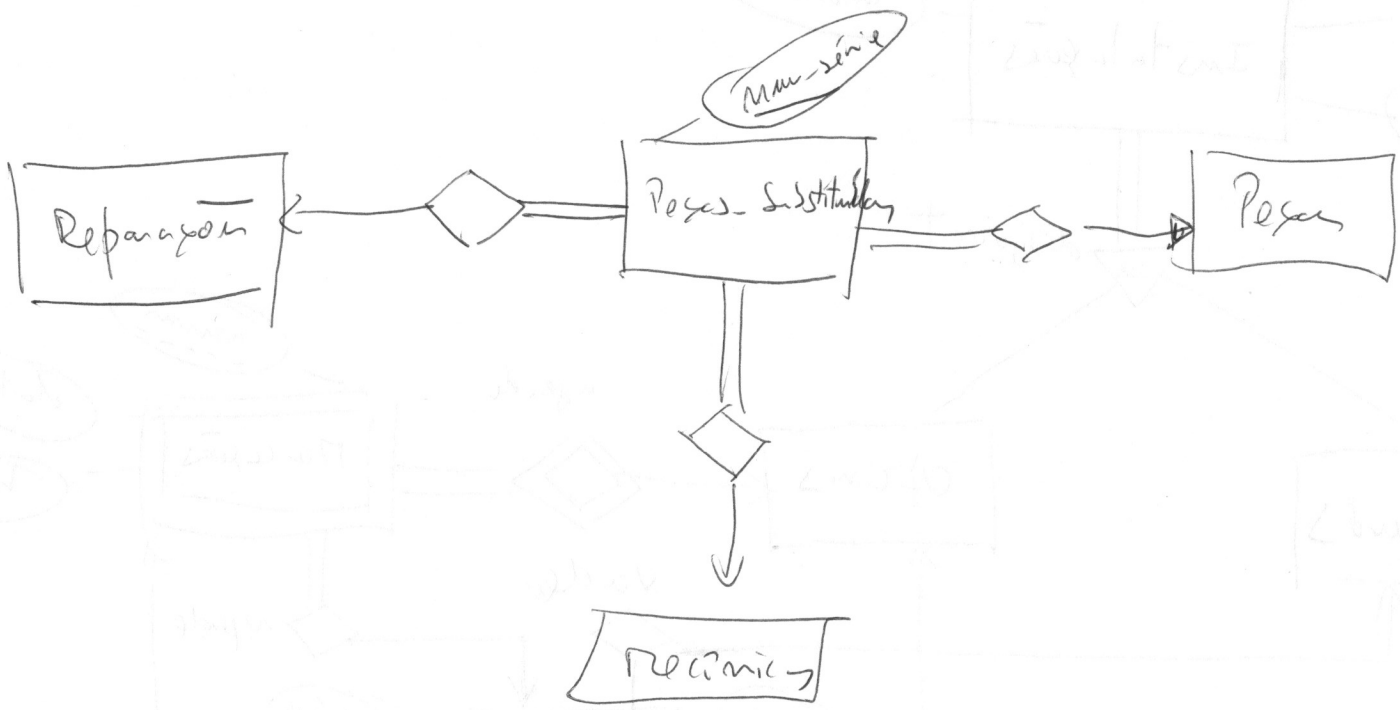
Peças_substituídas (num_série, código_r, código_p, código_m)
CEs: código_r -> Reparações(código_r), código_m -> Mecânicos (código_m) e
código_p -> Peças (código_p)

I.1.1. a)



código_i
nome
abertura
morada
Instalações
fecho
disjoint
ISA
Stands
Oficinas
agenda
num
data
Marcações
hora
vender
referente
modelo
matrícula
Veículos
marca
cor
reparações_marcadas
repondo
Reparações
código_r
data
descrição
nome
código_m
especialidade
Mecânicos
horas
horas_reparação
peças_substituídas
Peças
código_p
preço
qtd
descrição

I. 1. 2. a)



Grupo I.2

I.2.a) As duas chaves candidatas são: {Matrícula,DataRegisto} e {Chassis,DataRegisto}

Justificação (com as abreviaturas óbvias):

MD é super-chave: MD^+ = MDNPUCG
MD é minimal:

- M+ = MCG <> MDNPUCG
- D+ = D <> MDNPUCG

CD é super-chave: CD^+ = MDNPUCG
CD é minimal:

- C+ = CMG <> MDNPUCG
- D+ = D <> MDNPUCG

I.2.b) Uma cobertura canónica Fc é a seguinte:

MD -> PU
M -> CG
C -> M
P -> N

Ou seja, basta provar que G é dispensável na dependência C -> MG e que não há mais atributos dispensáveis.
NOTA: para sermos rigorosos teríamos de provar que todos os atributos não são dispensáveis. Basta escrever na vossa solução a prova que um deles não é dispensável, assumo que vocês verificaram todos os outros. Vamos verificar que M não é dispensável na primeira dependência funcional. Na vossa justificação incluam sempre um caso à direita e outro à esquerda.

C é dispensável à direita em C->MG. Para isso, calcula-se C^+ em Fc e vemos se o resultado inclui MG. Como C^+ = CMG logo C é dispensável à direita.

M não é dispensável à esquerda em F. Calcula-se D^+ = D em F e como o resultado não inclui PU logo M não é dispensável à esquerda.

I.2.c) O esquema não está na forma normal de Boyce-Codd. Por exemplo, a dependência funcional P -> N não é trivial e o seu lado esquerdo não é super-chave pois P+ = PN, não se obtendo todos os atributos. Vamos então decompor R:

R(MDCGPUN) decompondo com P->N obtém-se:
R1(PN)
R2(MDCGPU)

R1(PN) está na FNBC pois só tem dois atributos. Mas R2 não se encontra na FNBC pois por exemplo M+ = MCG obtendo alguns atributos, para além de M, em R2 mas não todos.

Decompondo R2(MDCGPU) com M -> CG obtém-se:
R3(MCG) e
R4(MDPU)

Vamos verificar que R3 está na FNBC (como tem 3 atributos, basta testar subconjuntos de cardinalidade até 3-2=1)

M+ = MCG (todos os atributos de R3)
C+ = MCG (todos os atributos de R3)
G+ = G (só G)

Vamos verificar que R4 está na FNBC (como tem 4 atributos, basta testar subconjuntos de cardinalidade até 4-2=2)

M+ = MCG (todos)	MD+ = R
D+ = D (só D)	MP+ = MCGPN (só MP em R4)
P+ = PN (só P, N não está em R4)	MU+ = MUCG (só MU em R4)
U+ = U (só U)	DP+ = DPN (só DP em R4)
	DU+ = DU (só D)
	PU+ = PNU (só PU)

Logo, a decomposição do esquema na FNBC é R1(PN), R3(MCG) e R4(MDPU).

I.2.d) O esquema não está na 3FN pois quer M -> CG quer P -> N violam a 3FN. Pois, ambas não são triviais, o seu lado esquerdo não são super-chaves e têm um atributo do lado direito que não pertence a uma chave candidata (G) e (N), respectivamente.

Aplicando o algoritmo à cobertura canónica Fc de I.2.b obtemos:

R1(MDPU), R2(MCG) e R3(PN). Repare-se que a dependência funcional C->M não introduz uma relação pois CM está contido em R2. Como R1, contém uma chave candidata não há necessidade de juntar mais nenhuma relação.

Grupo II.1

II.1.1.a)

$$\Pi_{matricula}\,(\sigma_{kms=0}(vendas)) - \Pi_{matricula}\,(\sigma_{kms>0}(vendas))$$

```
SELECT matrícula
FROM Vendas
WHERE kms = 0
MINUS
SELECT matrícula
FROM Vendas
WHERE kms > 0
```

II.1.1.b)

$$\Pi_{NomeMarca}\left[\sigma_{kms=0}(vendas) \bowtie \sigma_{Nome='CarroNovo' \wedge data='2017'(empresas)} \bowtie veiculos \bowtie marcas\right]$$

```
SELECT NomeMarca
FROM Empresas NATURAL INNER JOIN Vendas
                    NATURAL INNER JOIN Veículos
                    NATURAL INNER JOIN Marcas
WHERE Kms=0 AND Nome='Carro Novo' AND data = '2017'
```

II.1.1.c)

$$\Pi_{NomeMarca,NomeModelo,Nome}\left\{\begin{array}{c}\Pi_{Chassis,NIF}\,[vendas \bowtie \sigma_{localidade='Lisboa'}(stands)] \\ \bowtie\,(pessoas \bigcup empresas) \\ \bowtie veiculos \bowtie marcas \\ \bowtie \rho_{x(idMarca,idModelo,NomeModelo,AnoModelo)}(modelos)\end{array}\right\}$$

```
SELECT p.Nome, v.NomeMarca, v.Nome as NomeModelo
FROM (vendas NATURAL INNER JOIN veículos
                    NATURAL INNER JOIN modelos
                    NATURAL INNER JOIN marcas ) v
        INNER JOIN ( SELECT * FROM empresas UNION SELECT * FROM pessoas) p USING (NIF)
        INNER JOIN stands USING (idStand)
WHERE localidade = 'Lisboa'
```

II.1.2.a)

```
SELECT COUNT(DISTINCT matrícula)
FROM vendas NATURAL INNER JOIN stands
WHERE kms > 100000 AND localidade = 'Lisboa'
```

II.1.2.b)

```
SELECT idStand
FROM vendas
GROUP BY idStand
HAVING COUNT(*) > 1000
```

II.1.2.c)

```
WITH vendas_marca(idMarca,Num) AS
( SELECT idMarca,COUNT(*)
  FROM vendas NATURAL INNER JOIN  Vendas
  WHERE Kms=0 AND data = '2017'
  GROUP BY idMarca
)
SELECT idMarca
FROM vendas_marca WHERE Num = (SELECT MAX(Num) FROM vendas_marca)
```

Grupo II.2

II.2.1.a)

```
ALTER TABLE empresas ADD CONSTRAINT c1 UNIQUE (Nome)

CREATE ASSERTION nome_igual CHECK
NOT EXISTS (SELECT Nome FROM pessoas INTERSECT SELECT nome FROM empresas)
```

II.2.1.b)

```
CREATE ASSERTION só_uma_matrícula CHECK
NOT EXISTS (SELECT Chassis
            FROM Vendas
            GROUP BY Chassis
            HAVING COUNT(DISTINCT matrícula) > 1 )
```

II.2.1.c)

```
CREATE ASSERTION comprador_existe CHECK
NOT EXISTS
(SELECT NIF
 FROM Vendas
 WHERE NIF NOT IN( SELECT NIF FROM pessoas UNION SELECT NIF FROM empresas )
)
```

II.2.2.a)

```
CREATE TRIGGER primeira_venda AFTER INSERT ON Vendas
REFERENCING new row as nrow
FOR EACH ROW WHEN (Kms=0)
BEGIN
    INSERT INTO transmissões VALUES(nrow.matrícula,1,nrow.data,0);
END;
```

II.2.2.b)

```
CREATE TRIGGER vendas_subsequentes AFTER INSERT ON Vendas
REFERENCING new row as nrow
FOR EACH ROW WHEN (Kms>0)
DECLARE problema NUMBER;
BEGIN
        SELECT COUNT(*) INTO problema
        FROM transmissões t
        WHERE t.matrícula = nrow.matrícula AND nrow.data > t.data AND nrow.kms < t.kms;
        IF problema > 0 THEN
            raise_application_error(-20000,'Mudou conta-quilómetros!')
        ELSE
            UPDATE transmissões
            SET numero = numero + 1, data = nrow.data, kms = nrow.kms
            WHERE matrícula = nrow.matrícula
        END IF;
END;
```

Grupo III.1

III.1.1.a)

q1(NomeModelo) :- stands(_,_,_,IdM), marcas(IdM,'UMM',_), modelos(IdM,_,NomeModelo,_).

III.1.1.b)

q2(Nome) :- modelos(IdMarca,IdModelo,Nome,_), not nunca_vendidos(IdMarca,IdModelo).
modelo_vendido(I1,I2) :- vendas(C, _, _, _, _, _), veículos(C,_,I1,I2).

III.1.1.c)

q3(IdStand) :- stands(IdStand,_,_,IdMarca), marcaFA(IdMarca).
marcaFA(IdMarca) :- marcas(IdMarca,_,'França').
marcaFA(IdMarca) :- marcas(IdMarca,_,'Alemanha').

Grupo III.2

III.2.1)

A utilização de transacções no modo serializable consegue garantir quando se verifica que o NIF da pessoa ou da empresa existe, esta tabela não é alterada enquanto se faz a inserção da venda. Caso contrário, poderia ficar-se num estado inconsistente.

III.2.2.a)

Pode ser 1 (executando T1 seguido de T2) ou 2 (execução de T2 seguida de T1)

III.2.2.b) Sim, é serializável de conflito. O único conflito que existe é entre a instrução B1 e a instrução A2, não formando qualquer ciclo o grafo de dependências.

Grupo III.3

III.3.1)

```
<concessionários>
	<marca id="S">
		<nome>Sado</nome>
		<stand>
			<nome>Carros Bons</nome>
			<morada cp="2900">Rua Rápida, 5, Setúbal</morada>
		</stand>
		<oficina>
			<oficial/>
			<morada cp="2900">Rua Rápida, 5, Setúbal</morada>
		</oficina>
		<oficina>
			<morada cp="2000">Rua Partida, 7, Santarém</morada>
		</oficina>
	</marca>
	<marca id="U">
		<nome>UMM</nome>
		<stand>
			 <nome>Carros portugueses</nome>
			 <morada cp="1000">Rua Fangio, 7, Lisboa</morada>
		</stand>
	</marca>
</concessionários>
```

III.3.2.a) //*[morada/@cp >= 1000 and morada/@cp <= 1999]
III.3.2.b) /concessionários/marca[stand and oficina]
III.3.2.c) /concessionários/marca[stand/morada = oficina/morada]

III.3.3) Só se indicam as alterações/acrescentos

```
<!ELEMENT marca (nome,modelos,stand+,oficina*)>
<!ELEMENT modelos (modelo*)>
<!ATTLIST modelo id ID #REQUIRED >
<!ELEMENT stand (nome, morada,veículo+) >
<!ELEMENT veículo (EMPTY)>
<!ATTLIST veículo modelo IDREF #REQUIRED
                  matrícula CDATA #REQUIRED
                  preço CDATA #REQUIRED >
```

III.3.4) Produz uma lista por marca dos respectivos stands e oficinas “não oficiais” com nome e morada no atributo b e nome no conteúdo do elemento nome

```
<lista>
<marca a="Sado"><nome b="Carros bons Rua Rápida, 5, Setúbal ">Carros
bons</nome></marca>
	<marca a="Sado"><nome b="Rua Partida 7, Santarém"/>
	<marca a="UMM"><nome b=" Carros portugueses Rua Fangio 7, Lisboa">Carros
	portugueses</nome></marca>
</lista>
```