

# Esercizi BDD



## ► ESERCITAZIONE 14

### DOMANDA 1

1. Sì, un'istanza di  $I$  è

$$\text{ISTANZE}(I, R) = \{< A:a, B:b >\}$$

$$\text{ISTANZE}(I, A) = \{a\}$$

$$\text{ISTANZE}(I, B) = \{b\}$$

2. Non esistono istanze di  $S$  in cui  $Q$  ha almeno una istanza perché le istanze di  $Q$  sono un sottoinsieme delle istanze di  $R$  e quindi non esistono 2 istanze di  $R$  con le stesse istanze di  $C$ . Quindi l'impossibilità è dovuta dall'incongruenza delle 2 cardinalità minime.

### DOMANDA 2

Considero un'istanza  $I$ :

$$\text{ISTANZE}(I, A) = \{a_1, \dots\}$$

$$\text{ISTANZE}(I, B) = \{b_1, b_2, \dots\}$$

$$\text{ISTANZE}(I, C) = \{c_1, \dots\}$$

$$\text{ISTANZE}(I, R_1) = \{< A:a_1, B:b_1 >, < A:a_1, B:b_2 >, \dots\}$$

$$\text{ISTANZE}(I, R_2) = \{< A:a_1, B:b_1 >, \dots\}$$

Scrivendo parte delle istanze ci accorgiamo che ogni istanza di  $B$  partecipa in  $R_2$  ed abbiano bisogno di 2 istanze di  $B$  per ogni  $A$  ma per ogni istanza di  $B$  abbiano bisogno di un'istanza di  $A$  che partecipa in  $R_2$  quindi non esiste alcuna istanza di  $S$ .

### DOMANDA 3

1. No perché dovendo  $E$  partecipare in almeno 2 istanze di  $R$  ho bisogno di 2 istanze di  $D$  e quindi di  $E$ .

2. Sì, vediamo l'istanza  $I$

$$\text{ISTANZE}(I, E) = \{e_1, e_2\}$$

$$\text{ISTANZE}(I, D) = \{d_1, d_2\}$$

$$\text{ISTANZE}(I, R) = \{< E:e_1, D:d_1 >, < E:e_1, D:d_2 >, < E:e_2, D:d_1 >, < E:e_2, D:d_2 >\}$$

$$\text{ISTANZE}(I, Q) = \{< E:e_1, D:d_1 >, < E:e_2, D:d_2 >\}$$

*Sia  $e_1$  che  $e_2$  devono partecipare 2 volte in  $R$*

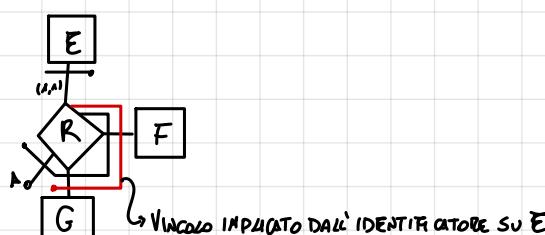
*Possano ognuno partecipare 2 volte perché la cardinalità è legata alla partecipazione nel ruolo*

### DOMANDA 4

1. Sì e sì sono equivalenti perché ho che la cardinalità di  $C$  in  $Q$  è  $(1,1)$  quindi ho che ogni istanza di  $C$  partecipa 1 ed 1 sola volta in  $R$  e  $Q$ . Siccome in  $S_1$  si ha l'istante  $R$  e  $Q$  avrà due in  $Q$  le stesse istanze di  $R$  e quindi  $S_1$  ed  $S_2$  sono equivalenti.

### DOMANDA 5

No, il vincolo non è essenziale infatti



### DOMANDA 6

$M(P,N)$

Inclusione:  $M(P) \subseteq F(M)$

$F(H,M,D,G)$

foreign key:  $F(M) \subseteq M(P)$

→ solo questo perché la cardinalità  $(1,1)$  implica l'identificatore in  $R$  sulle

$R(E,H,M)$

foreign key:  $R(H,M) \subseteq F(H,M)$

foreign key:  $R(E) \subseteq E(C)$

} *l'identificatore principale*

$E(C,B,A)$

foreign key:  $E(C) \subseteq R(E)$

Vincolo esterno: nel join tra  $R$  ed  $E$  con la condizione  $R.E=R.C$  non esistono due tuple di  $E$  con stessa combinazione di valori di  $\langle A, H, M \rangle$ .

→ *il vincolo esterno serve per rappresentare il vincolo su  $E$*

## DOMANDA 7

Lo schema relazione non è corretto.  
Quello corretto è:

$F(H, M1, M2, G)$   
foreign key:  $F(M1) \subseteq M(N)$   
foreign key:  $F(M2) \subseteq M(N)$

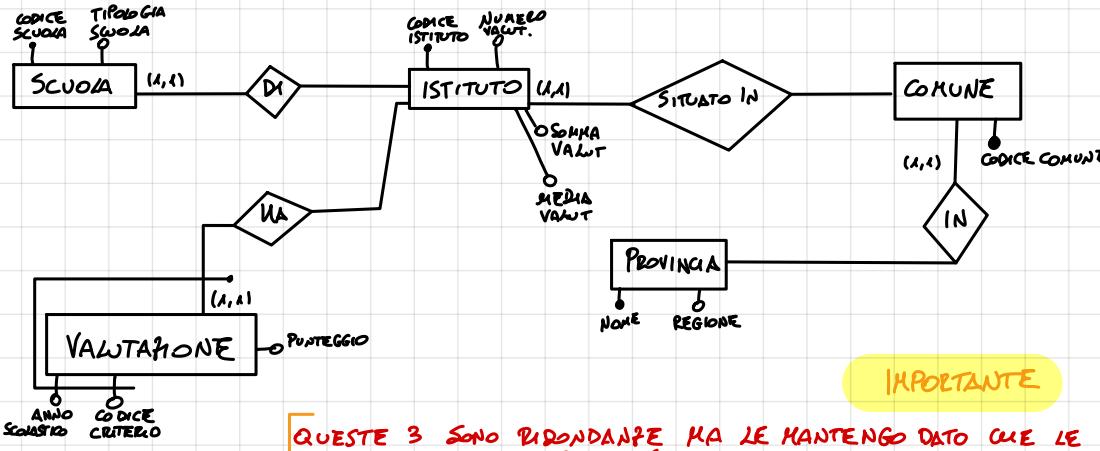
$M(N, P)$   
inclusione:  $M(N) \subseteq F(M2)$

Vincolo esterno:  
nel join tra E ed R con condizione  $E.C = R.C$  non esistono due tuple con stessa combinazione di  $\langle A, H, M1, M2 \rangle$ .

E quindi le proprietà non rappresentate sono:  
1. gli attributi M1 ed M2 sono foreign key referenziati all'identificatore di M  
2. cardinalità minima 1 per M nel ruolo M2 in Q

## ESERCIZIO DI PROGETTAZIONE

1.



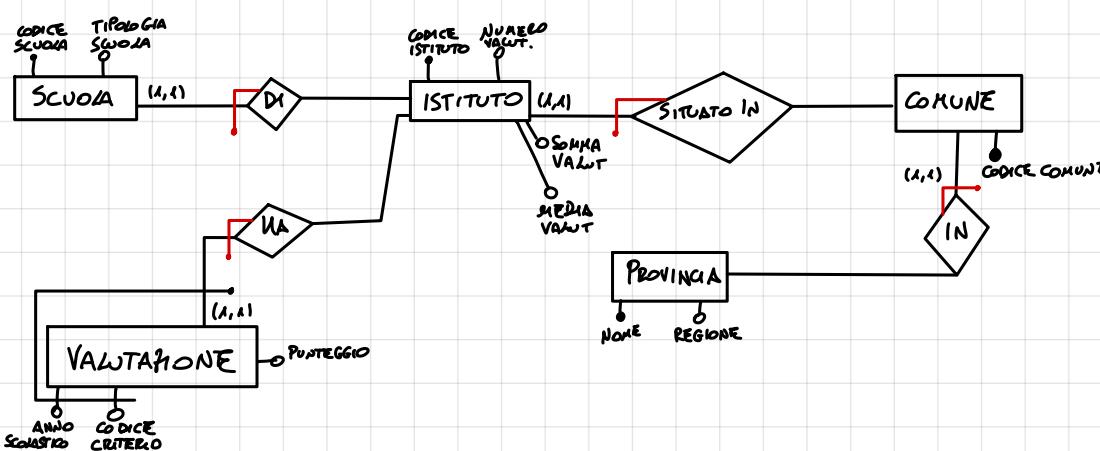
IMPORTANTE

QUESTE 3 SONO RIDONDANZE MA LE MANTENGO DATO CHE LE SPECIFICHE SULLE OPERAZIONI MI DICONO CHE ACCEDO SPesso AL NUMERO, MEDIA E SOMMA DELLE VALUTAZIONI. QUESTO È IMPATTO DAL PUNTO 1 DELLE SPECIFICHE SULLE RELAZIONI!

Vincoli esterni:

1. data un'istanza di Istituto l'attributo numerovalut. è il numero di istanze della relazione Ha in cui partecipa l'istanza di Istituto
2. data un'istanza di Istituto l'attributo sommavalut. è il valore assunto dalla somma dei valori dell'attributo punteggio associato a tutte le istanze dell'entità Valutazione che partecipano in Ha con l'istanza di Istituto
3. data un'istanza di Istituto l'attributo mediavalut. è il valore dato da sommavalut. / numerovalut.

## 2. RISTRUTTURAZIONE DELLO SCHEMA, TRADUZIONE DIRETTA E RISTRUTTURAZIONE DELLO SCHEMA RELAZIONALE



Traduzione diretta:

Scuola(codicescuola, tipologiascuola)  
foreign key: Scuola(codicescuola) ⊂ Di(scuola)

\* NON SONO TUTTI E 3 PRIMARY KEY PERCHÉ HO L'IDENTIFICATORE ESPLICATO E QUINDI SOLO QUELLO È PRIMARY KEY

Istituto(codiceistituto, numerovalut., sommavalut., mediavalut.)  
foreign key: Istituto(codiceistituto) ⊂ Situatoln(istituto)

Di(scuola, istituto)  
foreign key: Di(scuola) ⊂ Scuola(codicescuola)  
foreign key: Di(istituto) ⊂ Istituto(codiceistituto)

Situatoln(istituto, comune)  
foreign key: Situatoln(istituto) ⊂ Istituto(codiceistituto)  
foreign key: Situatoln(comune) ⊂ Comune(codicecomune)

Comune(codicecomune)  
foreign key: Comune(codicecomune) ⊂ In(comune)

In(comune, provincia)  
foreign key: In(comune) ⊂ Comune(codicecomune)  
foreign key: In(provincia) ⊂ Provincia(nome)

Provincia(nome, regione)

Valutazione(annoscolastico, codicecriterio, codiceistituto, punteggio)  
foreign key: Valutazione(codiceistituto) ⊂ Istituto(codiceistituto)

Vincoli esterni:

1. per ogni tupla t di Istituto, t.numerovalutazioni = select count(\*) from Valutazione where istituto = t.codiceistituto

→ TRADUCO I VINCOLI ESTERNI SFRUTTANDO LO SCHEMA RELAZIONALE APPENA OTTENUTO

2. per ogni tupla t di Istituto, t.sommavalutazioni = select sum(punteggio) from Valutazione where istituto = t.codiceistituto

3. per ogni tupla t di Istituto, t.mediavalutazioni = select sum(punteggio)/count(\*) from Valutazione where istituto = t.codiceistituto

Ristrutturazione dello schema relazionale:

La specifica 2 richiede l'accorpamento tra Scuola e Di quindi

Scuola(codicescuola, Istituto, tipologiascuola)  
foreign key: Scuola(Istituto) C Istituto(codiceistituto)

La specifica 3 richiede l'accorpamento tra Istituto e Situatoln quindi

Istituto(codiceistituto, comune, numerovalut., sommavalut., mediavalut.)  
foreign key: Istituto(comune) C Comune(codicecomune)

La specifica 4 richiede l'accopramento tra Comune ed In quindi

Comune(codicecomune, provincia)  
foreign key: Comune(provincia) C Provincia(nome)

La specifica 5 richiede l'accorpamento richiedere l'accorpamento tra Comune e Provincia:

Comune (codicecomune, provincia, regione)

Vincolo esterno aggiuntivo:

Non esistono due tuple di Comune con stessa provincia e diversa regione.

## ► Esercitazione 15

### DOMANDA 1

1. No, B da sola non è una chiave perchè si dice solo che non esistono 2 tuple che nella coppia  $\langle A, B \rangle$  hanno sia stesso valore di A e stesso valore di B e che nella coppia  $\langle C, B \rangle$  non esistono 2 tuple che hanno sia stesso valore di C che di B mentre posso avere 2 tuple che hanno stessa B.
2. Come superchiavi della relazione ho:  $\langle A, B, C \rangle$ ,  $\langle A, B, D \rangle$ ,  $\langle B, C, D \rangle$ ,  $\langle A, B, C, D \rangle$ ,  $\langle A, B \rangle$ ,  $\langle B, C \rangle$  ANCHE UNA CHIAVE È UNA SUPERCHIAVE DATO CHE È UN PARTICOLARE TIPO DI SUPERCHIAVE
3. Sceglierrei come chiave primaria la chiave  $\langle A, B \rangle$  perchè nella chiave  $\langle C, B \rangle$  ho che C può essere NULL ma, nel caso di una chiave primaria, non posso avere valori NULL.

### DOMANDA 3

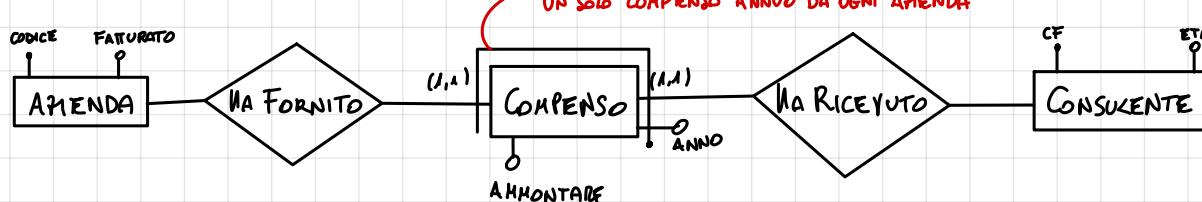
```
SELECT c1.giorno
FROM contagia AS c1
WHERE c1.regione = 'Lazio' AND NOT EXISTS(SELECT c2.giorno
                                            FROM contagia AS c2
                                            WHERE c2.regione = 'Lazio' AND c2.numinfetti < (SELECT c3.numinfetti
                                                FROM contagia AS c3
                                                WHERE c3.giorno = c2.giorno - 1 AND c3.regione = 'Lazio')
                                            AND c2.giorno < c1.giorno)

AND NOT EXISTS(SELECT c4.giorno
                FROM contagia AS c4
                WHERE c4.regione = 'Lazio' AND c4.numinfetti > (SELECT c5.numinfetti
                    FROM contagia AS c5
                    WHERE c5.giorno = c4.giorno - 1 AND c5.regione = 'Lazio')
                AND c4.giorno > c1.giorno)
```

### DOMANDA 4

Non esistono istanze dello schema S in cui l'insieme delle istanze R2 non è vuoto. Infatti abbiamo che ogni istanza di E deve partecipare almeno una volta in R1 e partecipa 1 ed 1 sola volta in R. Innanzitutto si ha che il ruolo E in R1 eredita la cardinalità massima 1 dal ruolo E in R quindi la cardinalità (1,n) diventa (1,1) dato che le istanze di R1 sono un sottoinsieme delle istanze di R. Siccome R1 ed R2 sono in generalizzazione l'insieme delle istanze delle due relazioni sono disgiunti ma tutte le istanze in R appartengono ad R1 per la cardinalità e quindi R2 sarà sempre vuoto.

### DOMANDA 5



### DOMANDA 6

Schema logico derivante dalla traduzione diretta:

Provincia(nome, regione)  
inclusione: Provincia(nome) C Automobile(provincia)

Automobile(targa, provincia, anno)  
foreign key: Automobile(provincia) C Provincia(nome)

AutoElettrica(codice, autonomia)  
foreign key: AutoElettrica(codice) C ISA-E-A(AutoElettrica)

ISA-E-A(autoel, automobile, provincia)  
foreign key: ISA-E-A(autoel) C AutoElettrica(codice)  
foreign key: ISA-E-A(automobile, provincia) C Automobile(targa, provincia)  
chiave: automobile, provincia

Schema logico ristrutturato per soddisfare le specifiche sulle operazioni:

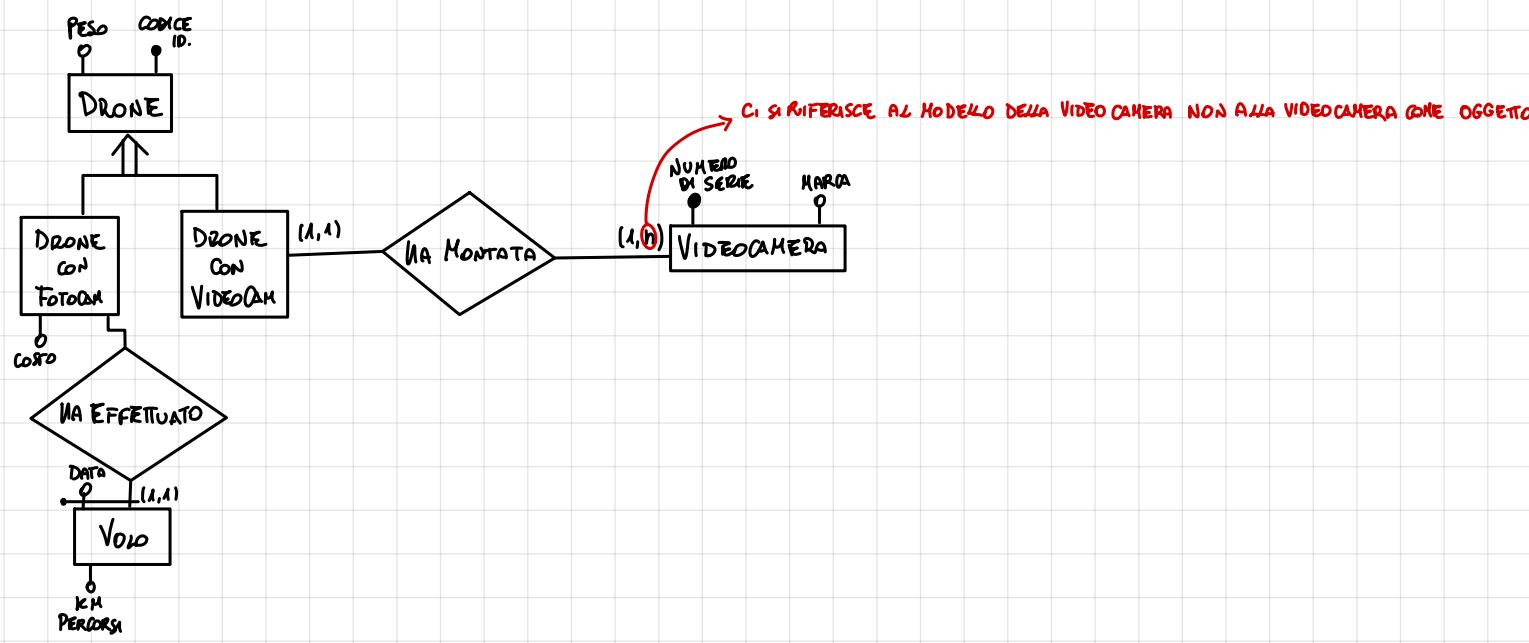
Inizialmente serve un accorpamento tra AutoElettrica ed ISA-E-A

AutoElettrica(codice, autonomia, automobile, provincia)  
foreign key: AutoElettrica(automobile, provincia) C Automobile(targa, provincia)  
chiave: automobile, provincia

Da cui poi

AutoElettrica(codice, autonomia, targa, anno)  
foreign key: AutoElettrica(targa, anno) C Automobile(targa, anno)  
chiave: targa

## DOMANDA 7

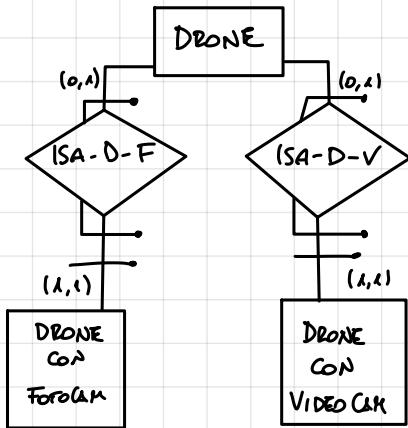


## DOMANDA 8

### RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

EVRONO LA GENERALIZZAZIONE

ESPLICATO UN VINCOLO IMPLICATO



VINCOLO ESTERNO:

$$\text{DRONE CON FOTOCAM} \cap \text{DRONE CON VIDEOCAM} = \emptyset$$

Traduzione diretta dello schema concettuale

Drone(codiceid, peso)

DroneConFotocam(codiceid, costo)

foreign key: DroneConFotocam(codiceid) ⊑ Drone(codiceid)  
vincolo di generalizzazione: l'intersezione tra DroneConFotocam(codiceid) e DroneConVideocam(codiceid) è vuota

IMPORTANTE

QUINDI NON VA DEFINITO UN VINCOLO ESTERNO MA SI INSERISCE UN VINCOLO IN UNA DELLE TABELLE

DroneConVideocam(codiceid)

foreign key: DroneConVideocam(codiceid) ⊑ Drone(codiceid)  
foreign key: DroneConVideocam(codiceid) ⊑ HaMontata(drone)

Volo(drone, data, km\_percorsi)

foreign key: Volo(drone) ⊑ DroneConFotocam(codiceid)

HaMontata(drone, videocamera)

foreign key: HaMontata(drone) ⊑ DroneConVideocam(codiceid)

foreign key: HaMontata(videocamera) ⊑ Videocamera(numeroserie)

Videocamera(numeroserie, marca)

inclusione: Videocamera(numeroserie) ⊑ HaMontata(videocamera)

Ristrutturazione dello schema logico per rispettare le specifiche sulle operazioni

Innanzitutto svolgo un accorpamento tra DroneConVideocam e Videocamera con la cancellazione della relazione HaMontata

DroneConVideocam(codiceid, numeroserie, marca)

foreign key: DroneConVideocam(codiceid) ⊑ Drone(codiceid)

chiave: numeroserie

Faccio ora un accorpamento tra Drone e DroneConVideocam permettendo che numeroserie e marca possano essere NULL ma se uno non è NULL l'altro non può esserlo

Drone(codiceid, peso, flag(numeroserie\*. marca\*))

chiave: numeroserie

Con flag = true numeroserie e marca devono essere diversi da NULL mentre con flag = false numero serie e marca devono essere uguali a NULL.

→ VINCOLO NECESSARIO PER MANTENERE CONSISTENZA

Bisogna quindi aggiungere dei vincoli aggiuntivi

DroneConFotocam(codiceid, costo)

foreign key: DroneConFotocam(codiceid) ⊑ Drone(codiceid)

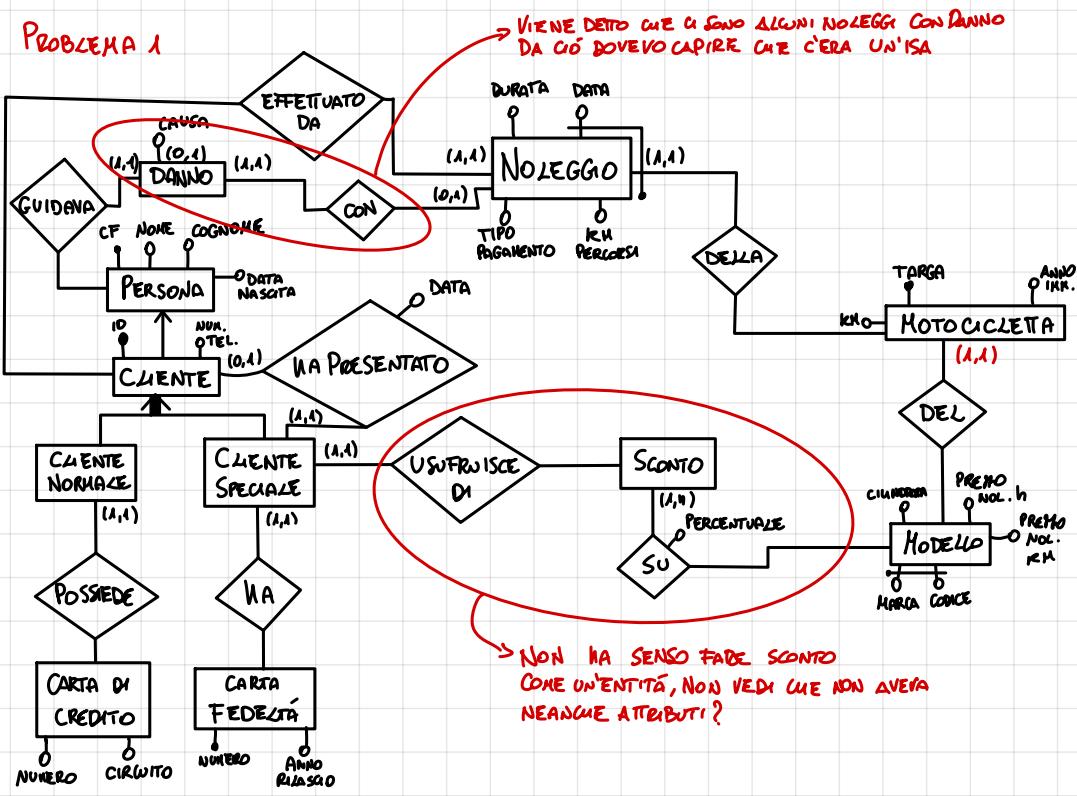
vincolo: DroneConFotocam(codiceid) ⊑ SELECT codice FROM drone WHERE flag = false

→ RAPPRESENTA IL VINCOLO DI GENERALIZZAZIONE

Volo(drone, data, km\_percorsi)

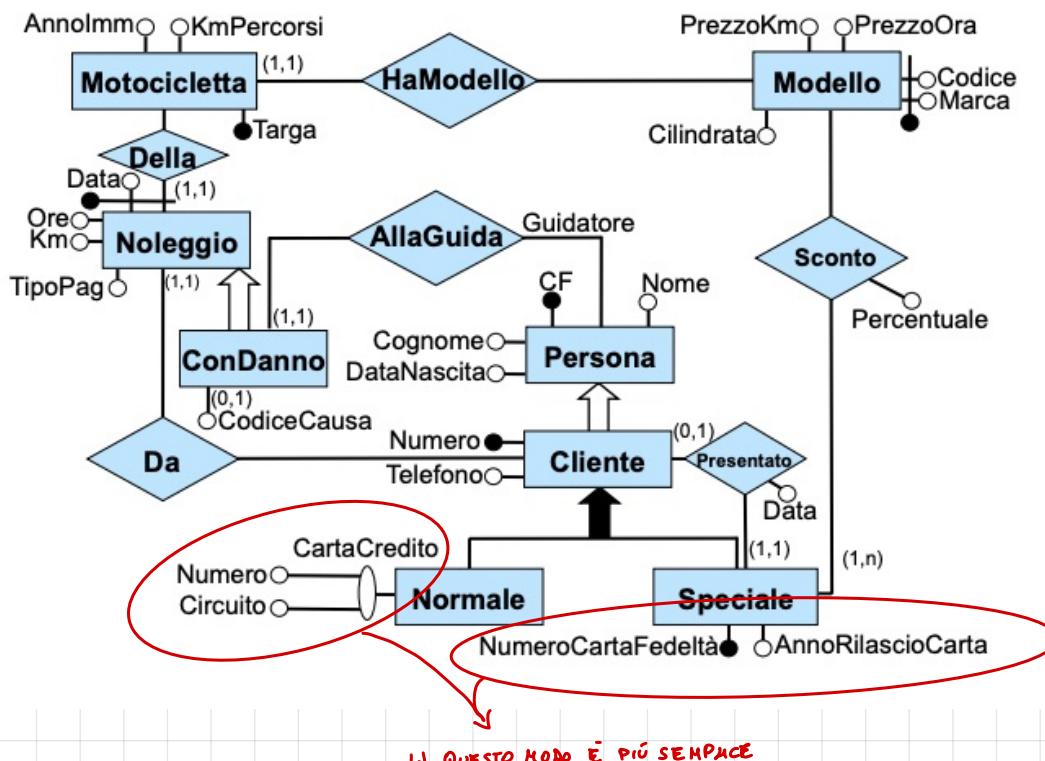
foreign key: Volo(drone) ⊑ DroneConFotocam(codiceid)

## PROBLEMA 1



SOLUZIONE PROFESSORE

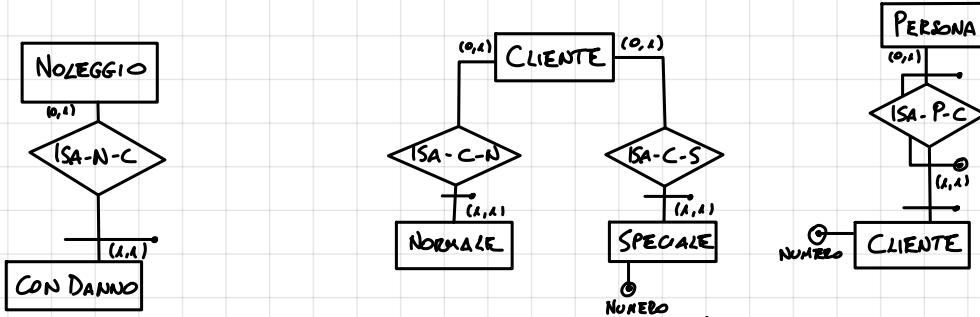
## Problema 1 – Schema ER



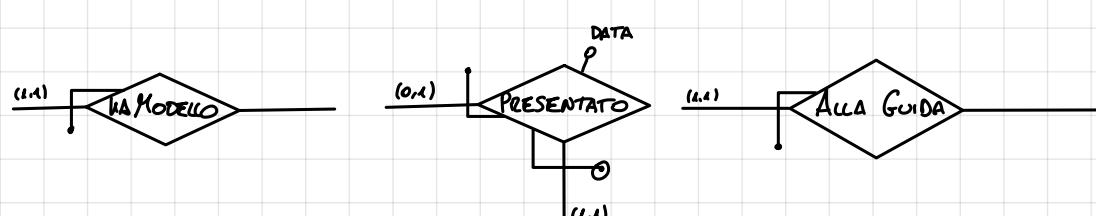
## PROBLEMA 2

RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

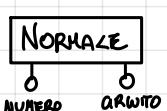
- ELEMINAZIONE DELLA GENERALIZZAZIONE ED ISA



- ESPLICAZIONE VINCOLI IMPLIKATI



- ELEMINAZIONE ATTRIBUTO COMPOSTO



VINCOLO ESTERNO: OGNI ISTANZA DI CLIENTE È NORMALE O SPECIALE. NORMALE E SPECIALE SONO ENTITÀ DISGIUNTE. → Non dimenticare il vincolo di GENERALIZZAZIONE.

## PROGETTAZIONE SCHEMA LOGICO

Modello(codice, marca, cilindrata, prezzoOra, prezzoKM)

Motocicletta(targa, annoLMM, kmPercorsi)

foreign key: Motocicletta(targa) ⊂ HaModello(motocicletta)

HaModello(motocicletta, codice, marca)

foreign key: HaModello(motocicletta) ⊂ Motocicletta(targa)

foreign key: HaModello(codice, marca) ⊂ Modello(codice, marca)

Noleggio(targa, data, ora, km, tipopag)

foreign key: Noleggio(targa) ⊂ Motocicletta(targa)

foreign key: Noleggio(targa, data) ⊂ Da(targa, data)

ConDanno(targa, data, codiceCausa\*)

foreign key: ConDanno(targa, data) ⊂ Noleggio(targa, data)

foreign key: ConDanno(targa, data) ⊂ AllaGuida(targa, data)

Persona(CF, nome, cognome, dataNascita)

AllaGuida(targa, data, CF)

foreign key: AllaGuida(targa, data) ⊂ ConDanno(targa, data)

foreign key: AllaGuida(CF) ⊂ Persona(CF)

Cliente(numero, telefono)

vincolo di generalizzazione: Cliente(numero) = Normale(numero) U ISA-C-S(numeroCliente)

foreign key: Cliente(numero) ⊂ ISA-P-C(numero)

ISA-P-C(numero, CF)

foreign key: ISA-P-C(numero) ⊂ Cliente(numero)

foreign key: ISA-P-C(CF) ⊂ Persona(CF)

chiave: CF

Da(targa, data, numero)

foreign key: Da(targa, data) Noleggio(targa, data)

foreign key: Da(numero) ⊂ Cliente(numero)

Normale(cliente, numeroCarta, circuitoCarta)

vincolo di generalizzazione: L'intersezione tra Normale(cliente) e ISA-C-S(numerocliente) è vuota

foreign key: Normale(cliente) ⊂ Cliente(numero)

Speciale(numeroCartaFedeltà, annoRilascioCarta)

foreign key: Speciale(numeroCartaFedeltà) ⊂ ISA-C-S(numeroCartaFedeltà)

foreign key: Speciale(numeroCartaFedeltà) ⊂ Presentato(numeroCartaFedeltà)

inclusione: Speciale(numeroCartaFedeltà) ⊂ Sconto(cliente)

ISA-C-S(numeroCartaFedeltà, numeroCliente)

foreign key: ISA-C-S(numeroCartaFedeltà) ⊂ Speciale(numeroCartaFedeltà)

chiave: numeroCliente

Sconto(cliente, codice, marca, percentuale)

foreign key: Sconto(cliente) ⊂ Speciale(numeroCartaFedeltà)

foreign key: Sconto(codice, marca) ⊂ Modello(codice, marca)

Presentato(cliente, numeroCartaFedeltà, data)

foreign key: Presentato(cliente) ⊂ Cliente(numero)

foreign key: Presentato(numeroCartaFedeltà) ⊂ Speciale(numeroCartaFedeltà)

chiave: cliente

## RISTRUTTURAZIONE SCHEMA LOGICO

Dalle specifiche ci viene detto che ai clienti si accede mediante il numero identificativo quindi si necessita di inserire in numero identificativo anche nella tabella Speciale dove non è presente e quindi si deve fare un'accorpamento tra Speciale e ISA-C-S:

Speciale(numeroCartaFedeltà, numeroCliente, annoRilascioCarta)

foreign key: Speciale(numeroCartaFedeltà) ⊂ Presentato(numeroCartaFedeltà)

foreign key: Speciale(numeroCliente) ⊂ Cliente(numero)

inclusione: Speciale(numeroCartaFedeltà) ⊂ Sconto(cliente)

chiave: numeroCliente

Per la specifica 2 sulle operazioni si richiede un accorpamento tra ConDanno e AllaGuida ma prima bisogna separare tra i noleggi con codiceCausa non NULL e noleggi con codiceCausa NULL per poi fare l'accorpamento, per ConDannoSenzaCausa rimane la relazione AllaGuida:

ConDannoConCausa(targa, data, codiceCausa, persona)

foreign key: ConDanno(targa, data) ⊂ Noleggio(targa, data)

foreign key: ConDanno(persona) ⊂ Persona(CF)

ConDannoSenzaCausa(targa, data)

vincolo: L'intersezione tra ConDannoSenzaCausa(targa, data) e ConDannoConCausa(targa, data) è vuota

foreign key: ConDanno(targa, data) ⊂ Noleggio(targa, data)

foreign key: ConDannoSenzaCausa(targa, data) ⊂ AllaGuidaSenzaCausa(targa, data)

AllaGuidaSenzaCausa(targa, data, CF)

foreign key: AllaGuidaSenzaCausa(targa, data) ⊂ ConDannoSenzaCausa(targa, data)

foreign key: AllaGuidaSenzaCausa(CF) ⊂ Persona(CF)

## PROBLEMA 3

1. SELECT film.regista, film.spettatori  
FROM film  
WHERE anno >= 2000 AND anno <= 2010 AND (paese = 'Italia' OR paese = 'Francia' OR paese = 'Germania')

2. SELECT genere.genrefilm, sum(film.spettatori)  
FROM genere JOIN film ON genere.regista = film.regista  
GROUP BY genere.genrefilm

3. SELECT film.anno  
FROM film JOIN genere ON film.regista = genere.regista  
WHERE film.spettatori > 1000

EXCEPT

SELECT film.anno  
FROM film JOIN genere ON film.regista = genere.regista  
WHERE genere.genrefilm == 'horror'

## PROBLEMA 4

Abbiamo i seguenti vincoli di integrità:

1. Vincoli di cardinalità (1,1) sul ruolo E in R1 ed R2

2. Vincolo di identificazione su ruolo E in R1 ed R2

3. Vincolo di cardinalità (1,1) sull'attributo V

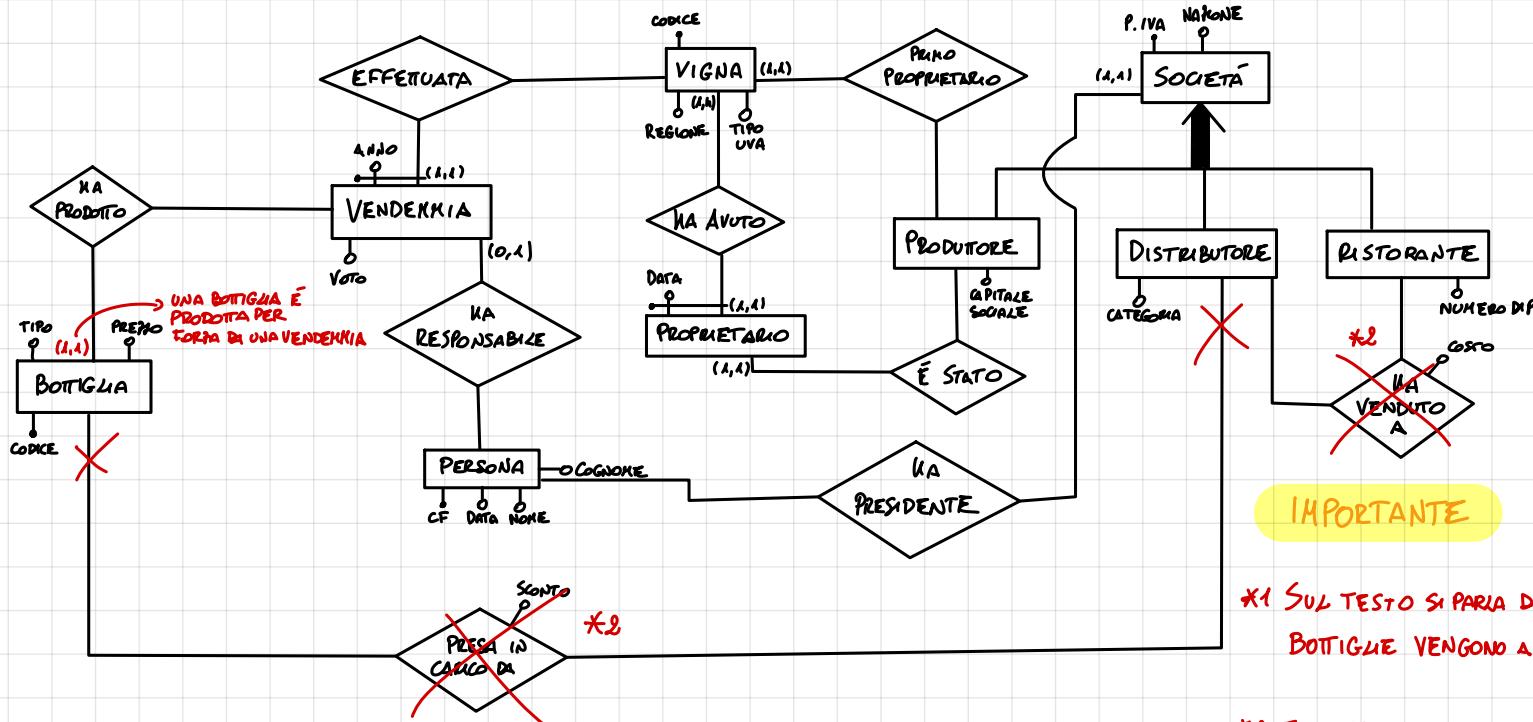
Un'istanza dello schema che non soddisfa i vincoli è:

Istanze(I,E) = {e}  
 Istanze(I,A) = {a}  
 Istanze(I,B) = {b}  
 Istanze(I,R1) = {}  
 Istanze(I,R2) = {<e,b>}

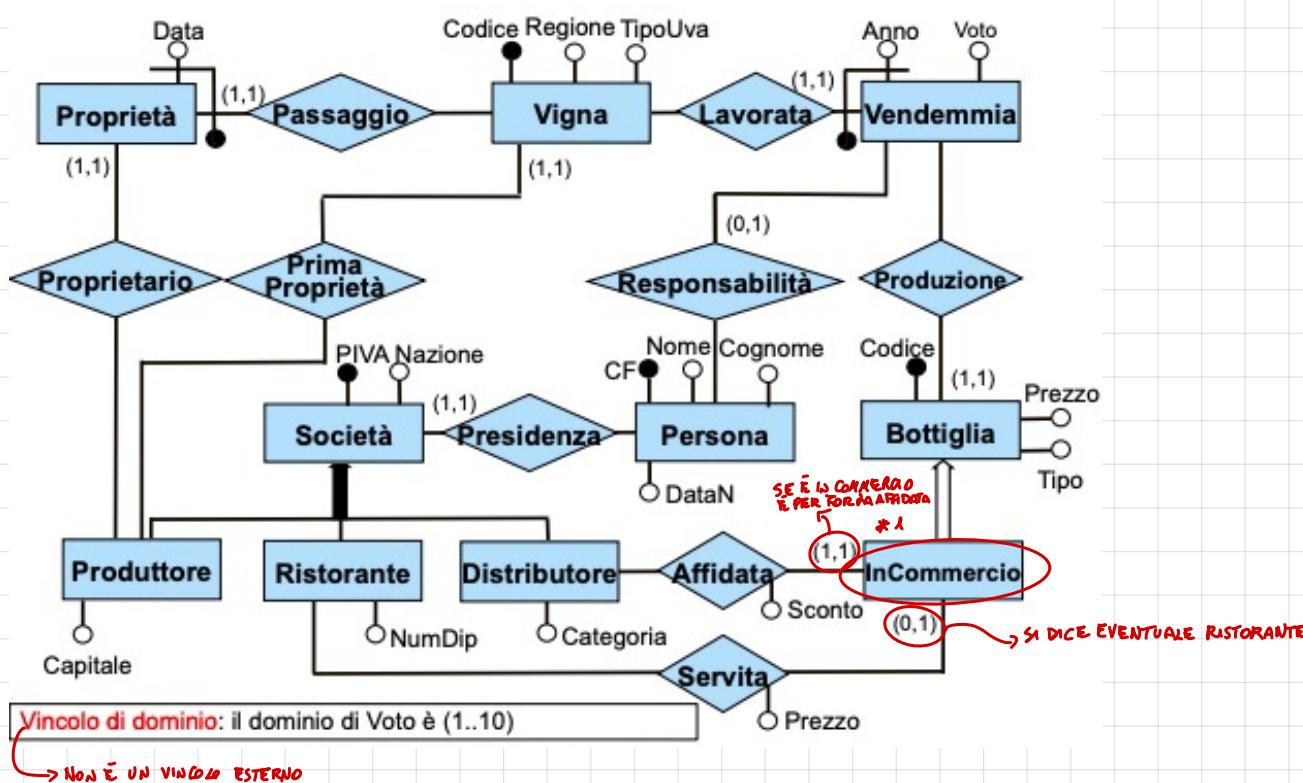
Vediamo che l'istanza e dell'entità E non partecipa in R1, abbiamo quindi una violazione del vincolo di cardinalità (1,1).

► ESAME 25 GENNAIO 2019

PROBLEMA 1



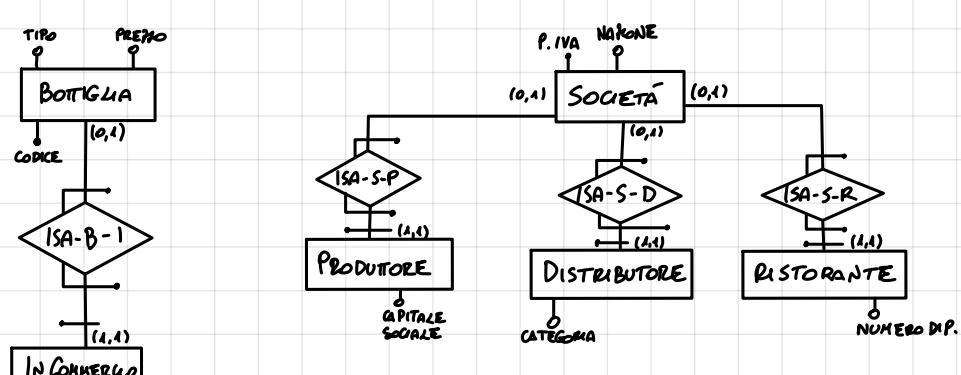
SOLUZIONE DEL PROFESSORE



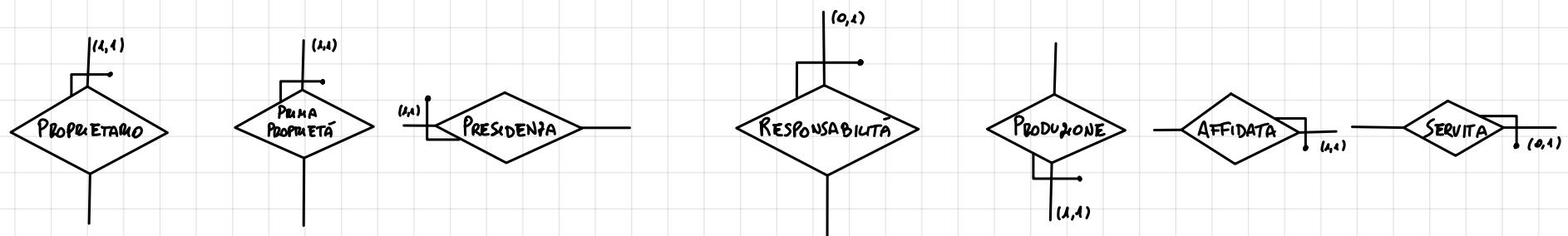
PROBLEMA 2

RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

• EKLINAZIONE ISA E GENERALIZZAZIONI



## • ESPLICAZIONE VINCOLI IMPLICATI



*RICORDATI: SEMPRE QUESTO VINCOLO*

VINCOLO DI GENERALIZZAZIONE:

UN'ISTANZA DI SOCIETÀ PARTECIPA AD UNA ED UNA SOLO DELLE 3 RELAZIONI ISA.

## TRADUZIONE DIRETTA

Vigna(codice, regione, tipouva)  
foreign key: Vigna(codice) ⊑ PrimaProprietà(vigna)

Proprietà(vigna, data)  
foreign key: Proprietà(vigna) ⊑ Vigna(codice)  
foreign key: Proprietà(vigna, data) ⊑ Proprietario(vigna, data)

Vendemmia(vigna, anno, voto)  
foreign key: Vendemmia(vigna) ⊑ Vigna(codice)

Società(piva, nazione)  
foreign key: Società(piva) ⊑ Presidenza(società)  
vincolo di generalizzazione: Società(piva) = Produttore(piva) U Ristorante(piva) U Distributore(piva) e l'intersezione tra Produttore(piva), Ristorante(piva) e Distributore(piva) è vuota

Persona(CF, nome, cognome, dataN)

Presidenza(società, persona)  
foreign key: Presidenza(società) ⊑ Società(piva)  
foreign key: Presidenza(persona) ⊑ Persona(CF)

Responsabilità(vendemmia, anno, persona)  
foreign key: Responsabilità(vendemmia, anno) ⊑ Vendemmia(vigna, anno)  
foreign key: Responsabilità(persona) ⊑ Persona(CF)

Produttore(piva, capitale)  
foreign key: Produttore(piva) ⊑ Società(piva)

Ristorante(piva, numdip)  
foreign key: Ristorante(piva) ⊑ Società(piva)

Distributore(piva, categoria)  
foreign key: Distributore(piva) ⊑ Società(piva)

PrimaProprietà(vigna, produttore)  
foreign key: PrimaProprietà(vigna) ⊑ Vigna(codice)  
foreign key: PrimaProprietà(produttore) ⊑ Produttore(piva)

Proprietario(vigna, data, piva)  
foreign key: Proprietario(vigna, data) ⊑ Produttore(piva)  
foreign key: Proprietario(vigna, data) ⊑ Proprietà(vigna, data)

Bottiglia(codice, prezzo, tipo)  
foreign key: Bottiglia(codice) ⊑ Produzione(bottiglia)

Produzione(bottiglia, vigna, anno)  
foreign key: Produzione(bottiglia) ⊑ Bottiglia(codice)  
foreign key: Produzione(vigna, anno) ⊑ Vendemmia(vigna, anno)

InCommercio(bottiglia)  
foreign key: InCommercio(bottiglia) ⊑ Bottiglia(codice)  
foreign key: InCommercio(bottiglia) ⊑ Affidata(bottiglia)

Affidata(bottiglia, distributore, sconto)  
foreign key: Affidata(bottiglia) ⊑ InCommercio(bottiglia)  
foreign key: Affidata(distributore) ⊑ Distributore(piva)

Servita(bottiglia, piva, prezzo)  
foreign key: Servita(bottiglia) ⊑ Ristorante(piva)  
foreign key: Servita(bottiglia) ⊑ InCommercio(bottiglia)

## RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

Dalla specifica 1 sulle operazioni si richiede inizialmente l'accorpamento tra Bottiglia e Produzione

Bottiglia(codice, vigna, anno, prezzo, tipo)  
foreign key: Bottiglia(vigna, anno) ⊑ Vendemmia(vigna, anno)

Va fatto ora un accorpamento tra InCommercio, Affidata e Servita

InCommercio(bottiglia, distributore, sconto, flag, piva\*, prezzo\*)  
foreign key: InCommercio(bottiglia) ⊑ Bottiglia(codice)  
foreign key: InCommercio(distributore) ⊑ Distributore(piva)  
foreign key: InCommercio(piva) ⊑ Ristorante(piva)  
vincolo: Se flag = true allora piva != NULL e prezzo != NULL, se flag = false allora piva = NULL e prezzo = NULL

In conclusione si accoppa Bottiglia ed InCommercio

Bottiglia(codice, vigna, anno, prezzo, tipo, distributore\*, sconto\*, piva\*, prezzocomm\*)  
foreign key: Bottiglia(vigna, anno) ⊑ Vendemmia(vigna, anno)

inclusione: Bottiglia(distributore) ⊑ (SELECT piva FROM Società where Tipo = 'distributore')  
inclusione: Bottiglia(piva) ⊑ (SELECT piva FROM Società where Tipo = 'ristorante')

vincolo: Se distributore è NULL allora anche sconto è NULL

vincolo: Se piva è NULL allora anche prezzocomm è NULL

vincolo: Se distributore è NULL allora anche piva è NULL

Dalla specifica 2 bisogna accoppare Società con Produttore, Ristorante e Distributore

*qui viene detto che si vuole conoscere il tipo in modo diretto per questo serve questo attributo*  
Società(piva, nazione, tipo, capitale\*, numdip\*, categoria\*)  
vincolo: tipo = 'produttore' OR tipo = 'distributore' OR tipo = 'ristorante'  
vincolo: capitale != NULL se tipo = 'produttore'  
vincolo: numdip != NULL se tipo = 'ristorante'  
vincolo: categoria != NULL se tipo = 'distributore'

*È NECESSARIO FARLO IN QUESTO MODO PERCHÉ LA PIVA DEVE ESSERE DI UN DISTRIBUTORE O DI UN RISTORANTE, NON DI UNO QUALSIASI*

### PROBLEMA 3

```
SELECT c1.regione, c1.nome
FROM citta AS c1
WHERE c1.numabit > (SELECT avg(c2.numabit) FROM citta AS c2 WHERE c2.regione = c1.regione)
```

### PROBLEMA 4



### PROBLEMA 5

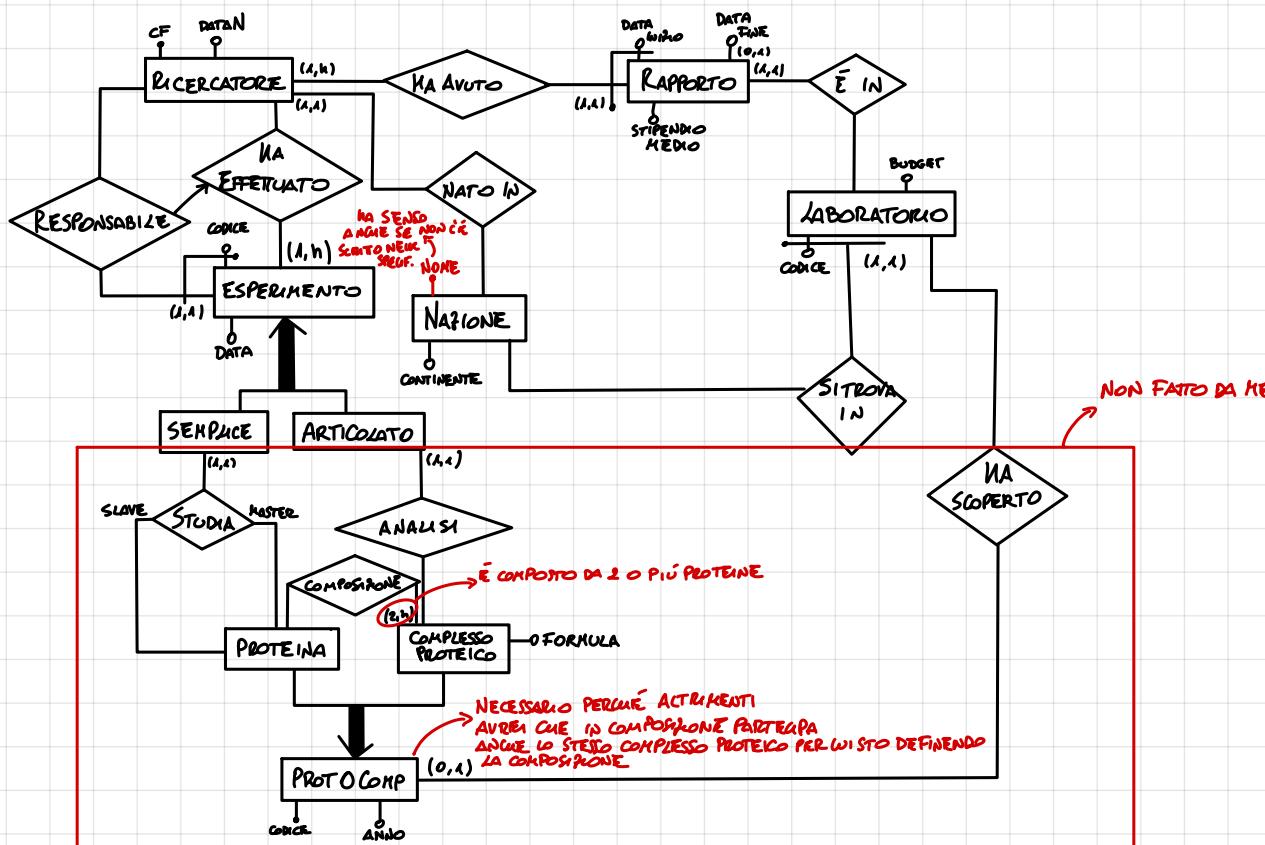
2. Si, esiste ed è la seguente:

```
Istanze(l,E) = {e1}
Istanze(l,F) = {e1}
Istanze(l,R) = {<c1, e1>}
Istanze(l,Q) = {<c1,e1>}
Istanze(l,C) = {c1}
```

3. No, non esiste. Ciò è dovuto al fatto che un'istanza di C non può partecipare più di 1 volta in R e quindi il ruolo C nella relazione Q acquisisce la cardinalità massima 1. Ciò implica che serve un'istanza di C per ogni F che partecipa a Q.

► ESAME 20 FEBBRAIO 2019

### PROBLEMA 1



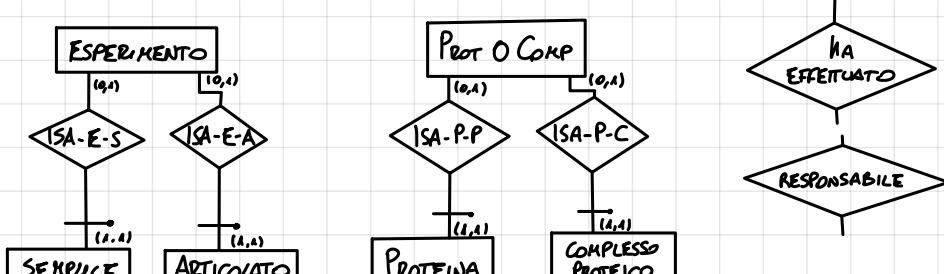
Vincoli esterni:

- Non esistono due istanze di Rapporto con datafine = NULL associate alla stessa istanza di Ricercatore
- Per ogni istanza di Rapporto in cui datafine != NULL si ha datainizio <= datafine e che datafine è minore di datainizio di tutte le istanze di Rapporto associate al Ricercatore che hanno datainizio maggiore dell'istanza di Rapporto che sto considerando
- Per le istanze di Rapporto associate ad un Ricercatore l'istanza con datafine = NULL ha datainizio maggiore di tutte le altre istanze di Rapporto

### PROBLEMA 2

RISTRUTTURAZIONE SCHEMA CONCETTUALE

#### • ELIMINAZIONE DELL'ISA



Vincoli esterni aggiuntivi:

- Esperimento partecipa in una ed una sola istanza di ISA-E-S o ISA-E-A, non in tutte e due le relazioni.
- ProtoComp partecipa in una ed una sola istanza di ISA-P-P o ISA-P-C, non in tutte e due le relazioni.
- Le istanze di Responsabile sono anche istanze di HaEffettuato.

## TRADUZIONE DIRETTA → ATTENZIONE A RIPORTARE LE INCLUSIONI NELLE ENTITÀ VERSO LE RELAZIONI PER LE CARDINALITÀ

Ricercatore(CF, dataN)  
foreign key: Ricercatore(CF) ⊑ Natoln(ricercatore)  
inclusione: Ricercatore(CF) ⊑ Rapporto(CF)

Nazione(nome, continente)

Natoln(ricercatore, nazione)  
foreign key: Natoln(ricercatore) ⊑ Ricercatore(CF)  
foreign key: Natoln(nazione) ⊑ Nazione(nome)

Esperimento(responsabile, codice, data)  
foreign key: Esperimento(responsabile) ⊑ Ricercatore(CF)  
inclusione: Esperimento(responsabile, codice) ⊑ HaEffettuato(responsabile, codice)  
vincolo: Esperimento(responsabile, codice) = Semplice(responsabile, codice) U Articolato(responsabile, codice)

HaEffettuato(responsabile, codice, ricercatore)  
foreign key: HaEffettuato(responsabile, codice) ⊑ Esperimento(responsabile, codice)  
foreign key: HaEffettuato(ricercatore) ⊑ Ricercatore(CF)

Rapporto(CF, datainizio, stipendiomedio, datafine\*)  
foreign key: Rapporto(CF) ⊑ Ricercatore(CF)  
foreign key: Rapporto(CF, datainizio) ⊑ èln(CF, datainizio)

Laboratorio(codice, nazione, budget)  
foreign key: Laboratorio(nazione) ⊑ Nazione(nome)

èln(CF, datainizio, codice, nazione)  
foreign key: èln(CF, datainizio) ⊑ Rapporto(CF, datainizio)  
foreign key: èln(codice, nazione) ⊑ Laboratorio(codice, nazione)

ProtOComp(codice, anno)  
vincolo: ProtOComp(codice) = Proteina(codice) U ComplessoProteico(codice)

HaScoperto(codice, codlab, nazione)  
foreign key: HaScoperto(codice) ⊑ ProtOComp(codice)  
foreign key: HaScoperto(codlab, nazione) ⊑ Laboratorio(codice, nazione)

Semplice(responsabile, codice)  
foreign key: Semplice(responsabile, codice) ⊑ Esperimento(responsabile, codice)  
foreign key: Semplice(responsabile, codice) ⊑ Studia(respesp, codesp)  
vincolo di generalizzazione: L'intersezione tra Semplice(responsabile, codice) e Articolato(responsabile, codice) è vuota

Articolato(responsabile, codice)  
foreign key: Articolato(responsabile, codice) ⊑ Esperimento(responsabile, codice)  
foreign key: Articolato(responsabile, codice) ⊑ Analisi(respesp, codesp)

Proteina(codice)  
foreign key: Proteina(codice) ⊑ ProtOComp(codice)  
vincolo di generalizzazione: L'intersezione tra Proteina(codice) e ComplessoProteico(codice) è vuota

ComplessoProteico(codice, formula)  
foreign key: ComplessoProteico(codice) ⊑ ProtOComp(codice)  
vincolo: (SELECT count(\*) FROM Composizione AS c WHERE c.codcomplprot = codice) >= 2

Composizione(codprot, codcomplprot)  
foreign key: Composizione(codprot) ⊑ Proteina(codice)  
foreign key: Composizione(codcomplprot) ⊑ ComplessoProteico(codice)

Studia(respesp, codesp, protslave, protmaster)  
foreign key: Studia(respesp, codesp) ⊑ Semplice(responsabile, codice)  
foreign key: Studia(protslave) ⊑ Proteina(codice)  
foreign key: Studia(protmaster) ⊑ Proteina(codice)

Analisi(respesp, codesp, codcompl)  
foreign key: Analisi(respesp, codesp) ⊑ Articolato(responsabile, codice)  
foreign key: Analisi(codcompl) ⊑ ComplessoProteico(codice)

Vincoli esterni:

1. Non esistono due tuple di Rapporto con datafine = NULL e con stesso CF
2. Per ogni tupla di Rapporto in cui datafine != NULL si ha datainizio <= datafine e che è minore di datainizio di tutte le tuple di Rapporto con stesso attributo CF che hanno datainizio maggiore della tupla di Rapporto che sto considerando
3. Per le tuple di Rapporto con stesso attributo CF la tupla con datafine = NULL ha datainizio maggiore di tutte le altre tuple di Rapporto con quel CF
4. Per ogni tupla t = (x,y,d) di Esperimento esiste la tupla (x,y,y) in HaEffettuato (ovvero: il responsabile di un esperimento è tra i ricercatori che lo effettuano). ]

IMPORANTE

É L'UNICO MODO PER RAPPRESENTARE QUESTO VINCOLO

## PISTRUZZIONE SCHEMA RELAZIONALE

La specifica 1 sulle operazioni richiede l'accorpamento tra Esperimento, Semplice, Articolato, Studia ed Analisi

Esperimento(responsabile, codice, data, tipo, master\*, slave\*, complprot\*)  
foreign key: Esperimento(responsabile) ⊑ Ricercatore(CF)  
inclusione: Esperimento(responsabile, codice) ⊑ HaEffettuato(responsabile, codice)  
foreign key: Esperimento(master) ⊑ Proteina(codice)  
foreign key: Esperimento(slave) ⊑ Proteina(codice)  
foreign key: Esperimento(complprot) ⊑ ComplessoProteico(codice)  
vincolo: tipo = 'Semplice' OR tipo = 'Articolato'  
vincolo: Se tipo = 'Semplice' allora master != NULL, slave != NULL, complprot = NULL.  
Se tipo = 'Articolato' allora master = NULL, slave = NULL, complprot != NULL.

La specifica 2 richiede l'accorpamento tra ProtOComp e HaScoperto

ProtOComp(codice, anno, scopertaesclusiva, codlab\*, nazlab\*)  
foreign key: ProtOComp(codlab, nazlab) ⊑ Laboratorio(codice, nazione)  
vincolo: ProtOComp(codice) = Proteina(codice) U ComplessoProteico(codice)  
vincolo: Se scopertaesclusiva = true allora codlab != NULL e nazlab != NULL. Se scopertaesclusiva = false allora codlab = NULL e nazlab = NULL.

## PROBLEMA 3

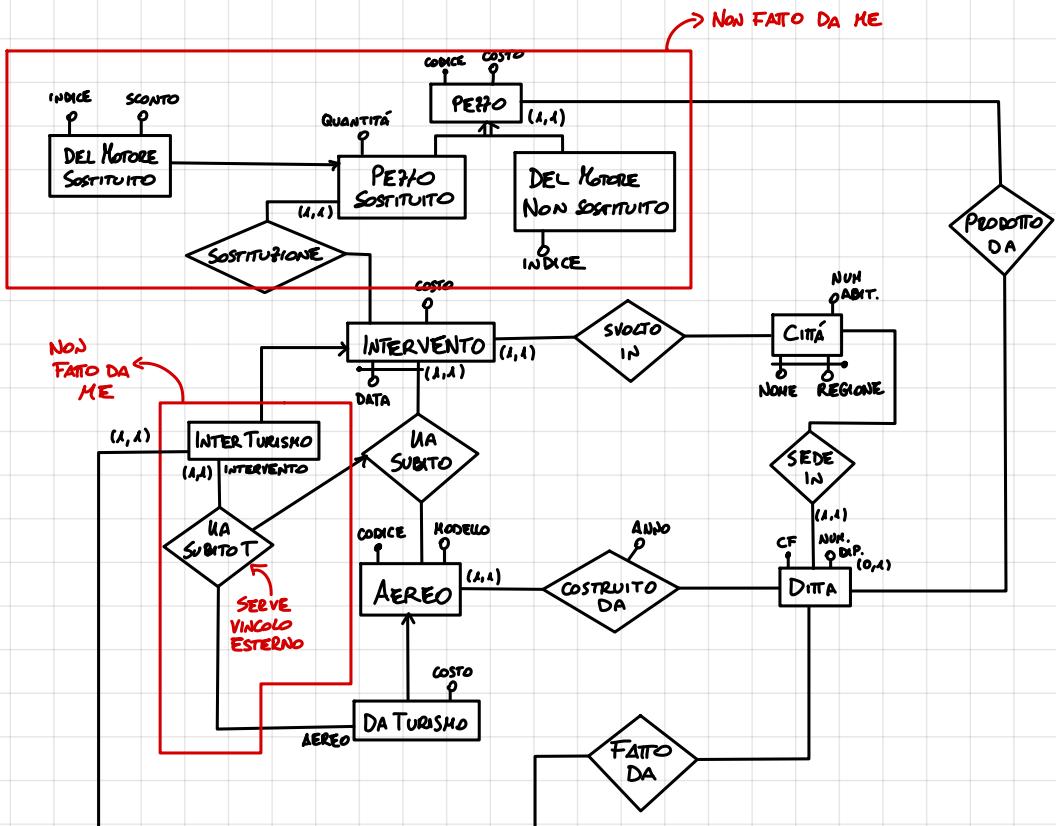
```
SELECT s1.citta, s1.codscuola  
FROM scuola s1 JOIN diplomato ON s1.codscuola = diplomato.codscuola  
GROUP BY s1.citta, s1.codscuola  
HAVING avg(diplomato.voto) >= ALL(SELECT avg(diplomato.voto) FROM scuola s2 JOIN diplomato ON s2.codscuola = diplomato.codscuola WHERE s2.citta = s1.citta GROUP BY s2.codscuola)
```

## PROBLEMA 5

Non esistono istanze dello schema in cui c'è almeno un'istanza di F1. Vediamo che il ruolo E in R ha cardinalità (1,1) quindi esiste una sola istanza di R in cui E partecipa. Questo significa che, data la cardinalità minima di 1 del ruolo E in R1 tutte le istanze di R si trovano in R1 quindi R2 è vuota. Perciò non possono esistere istanze di F1 altrimenti si violerebbe il vincolo (1..n) sul ruolo E in R2.

# ► ESAME 2 APRILE 2019

## PROBLEMA 1



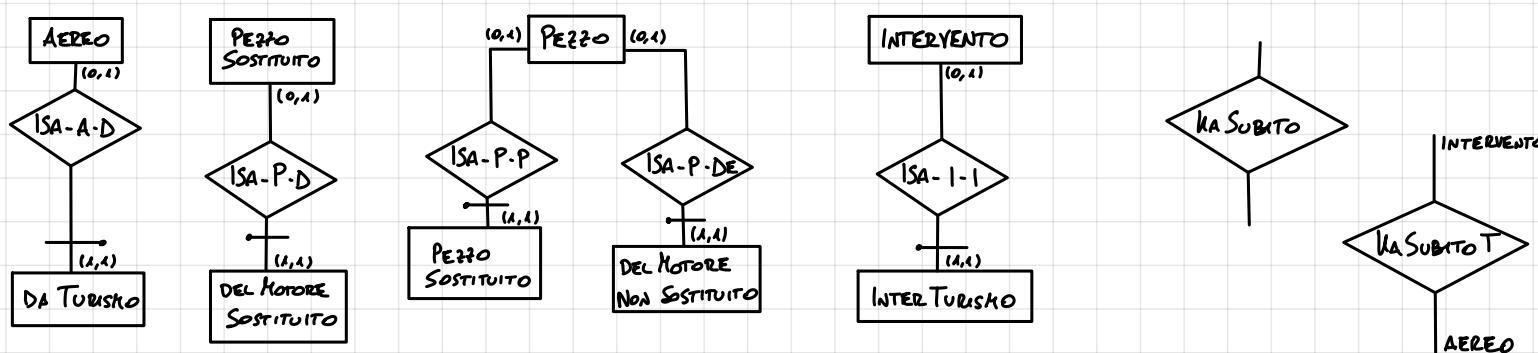
## Vincoli ESTERNI:

Ogni istanza di DaTurismo che partecipa in HaSubito partecipa anche in HaSubitoT

## PROBLEMA 2

## RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

#### • EUMINAZIONE ISA E GENERALIZAZIONI



## VINCOLO FESTEVO:

TRADE-NAME INDEX

Aereo(codice, modello)  
foreign key: Aereo(codice) C CostruitoDa(aereo)

Ditta(CF, numdip\*)  
foreign key: Ditta(CF) C Sedeln(ditta)

**CostruitoDa(aereo, ditta, anno)**  
foreign key: CostruitoDa(aereo) C Aereo(codice),  
foreign key: CostruitoDa(ditta) C Ditta(CF)

### Output

Sedeln(ditta, città, regione)  
foreign key: Sedeln(ditta) C Ditta(CF)

Intervento(aereo, data, costo)  
foreign key: Intervento(aereo) C Aereo(codice)

Svoltolin(aereo, data, città, regione)

foreign key: SvoltoIn(aereo, data) ⊂ Intervento(aereo, data)

foreign key: Pezzo(codice) C ProdottoDa(pezzo)

foreign key: ProdottoDa(pezzo) ⊂ Pezzo(codice)  
foreign key: ProdottoDa(ditta) ⊂ Ditta(CF)

DaTurismo(aereo, costo)  
foreign key: DaTurismo(aereo) ⊂ Aereo(codice)

InterTurismo(aereo, data)

HaSubitoT(codaereo, data, aereo)  
foreign key: HaSubitoT(codaereo, data) C InterTurismo(aereo, data)  
foreign key: HaSubitoT(aereo) C DaTurismo(aereo)  
vincolo: codaereo = aereo

QUESTO MI IMPICA CHE L'ISA È GIÀ SODDISFATTA DATO CHE MI ASSICURA CHE AEREO HA SUBITO UN INTERVENTO

FattoDa(aereo, data, ditta)  
foreign key: FattoDa(aereo, data) C InterTurismo(aereo, data)  
foreign key: FattoDa(ditta) C Ditta(CF)

PezzoSostituito(pezzo, quantità)  
foreign key: PezzoSostituito(pezzo) C Pezzo(codice)  
foreign key: PezzoSostituito(pezzo) C Sostituzione(pezzo)  
vincolo di generalizzazione: L'intersezione tra PezzoSostituito(pezzo) e DelMotoreNonSostituito(pezzo) è vuota

DelMotoreNonSostituito(pezzo, indice)  
foreign key: DelMotoreNonSostituito(pezzo) C Pezzo(codice)

DelMotoreSostituito(pezzo, indice, sconto)  
foreign key: DelMotoreSostituito(pezzo) C PezzoSostituito(pezzo)

Sostituzione(pezzo, aereo, data)  
foreign key: Sostituzione(pezzo) C PezzoSostituito(pezzo)  
foreign key: Sostituzione(aereo, data) C Intervento(aereo, data)

Vincolo esterni:  
Se in Intervento l'attributo aereo acquisisce un valore presente nell'attributo codice di DaTurismo allora esiste una tupla in HaSubitoT con attributo codaereo con valore uguale all'attributo codice della tupla di DaTurismo

### RISTRUTTURAZIONE SCHEMA RELAZIONALE

La specifica 1 sulle operazioni richiede l'accorpamento tra InterTurismo e FattoDa

InterTurismo(aereo, data, costo, ditta)  
foreign key: InterTurismo(aereo) C Aereo(codice)  
foreign key: InterTurismo(ditta) C Ditta(CF)

E accorpamento tra Intervento e SvoltoIn  
Intervento(aereo, data, costo, città, regione)  
foreign key: Intervento(aereo) C Aereo(codice)  
foreign key: Intervento(città, regione) C Città(nome, regione)

In conclusione serve un accorpamento tra Intervento e InterTurismo

Intervento(aereo, data, costo, città, regione, èDaTurismo, ditta\*)  
foreign key: Intervento(aereo) C Aereo(codice)  
foreign key: Intervento(città, regione) C Città(nome, regione)  
foreign key: Intervento(ditta) C Ditta(CF)  
vincolo: Se èDaTurismo = true allora ditta != NULL. Se èDaTurismo = false allora ditta = NULL  
vincolo di dominio: èDaTurismo = true OR èDaTurismo = false

La specifica 2 sulle operazioni richiede l'accorpamento tra Aereo e CostruitoDa

Aereo(codice, modello, ditta, anno)  
foreign key: Aereo(codice) C Ditta(CF)

### PROBLEMA 3

WITH  
indicatoriUscita AS  
(SELECT codScuola, avg(voto) AS media FROM diplomato GROUP BY codScuola)  
indicatoreMedioUscita AS  
(SELECT avg(media) AS indMedio FROM indicatoriUscita)  
totaleScuola AS  
(SELECT citta, count(\*) AS totale FROM scuola GROUP BY citta)

RICORDA DI POTER USARE LE VISTE PER FACILITARTI LA COSA

SELECT citta, count(\*)\*100/(SELECT totale FROM totaleScuola WHERE citta=s.citta) AS percentuale  
FROM scuola AS s NATURAL JOIN indicatoriUscita  
WHERE media > (SELECT indMedio FROM indicatoreMedioUscita)  
GROUP BY citta

### PROBLEMA 4

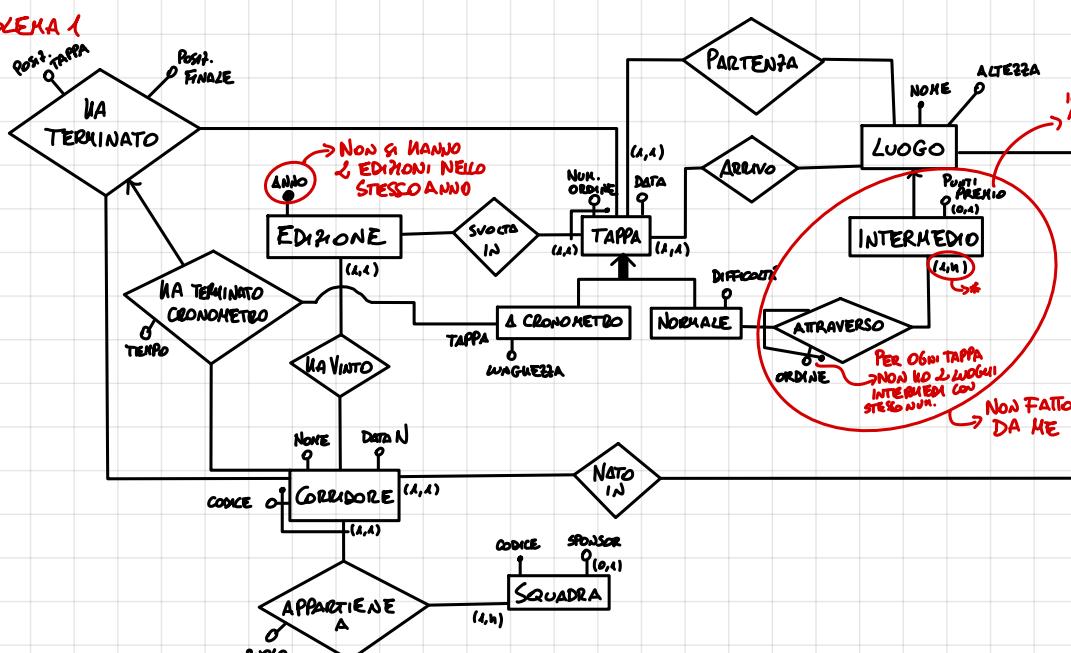
PROJ<sub>CITTÀ</sub>(SCUOLA) - PROJ<sub>CITTÀ</sub>(SCUOLA TON( PROJ<sub>CODSCUOLA</sub>(SCUOLA) - PROJ<sub>CODSCUOLA</sub>(SEL<sub>VOTO>100(DIPLOMATO))</sub>

### PROBLEMA 5

Non esiste una istanza dello schema. Abbiamo che l'istanza E partecipa 1 ed 1 una sola volta sia in R che in R1 e quindi tutte le istanze di R sono anche di R1 dato che quell'unica istanza di R in cui partecipa quell'istanza di E si troverà anche in R1.

### ► ESAME 21 GIUGNO 2019

#### PROBLEMA 1



#### VINCOLI ESTERNI:

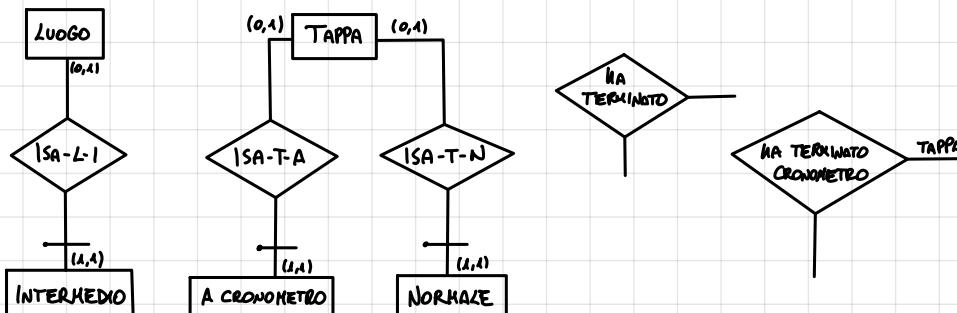
Tutte le istanze di HaTerminato in cui partecipa un'istanza di A Cronometro sono anche istanze di HaTerminatoCronometro.

\* UN LUOGO È INTERMEDIO SE ATTRAVERSATO DA ALMENO UNA TAPPA

## PROBLEMA 2

### RISTRUTTURAZIONE SCHEMA CONCETTUALE

#### • ELIMINAZIONE ISA E GENERALIZZAZIONI



#### VINCOLI ESTERNI AGGIUNTIVI:

1. Tappa partecipa in una ed una sola delle relazioni ISA-T-A e ISA-T-N.
2. Tutte le istanze di HaTerminatoCronometro sono anche istanze di HaTerminato.

#### TRADUZIONE DIRETTA

Squadra(codice, sponsor\*)  
inclusione: Squadra(codice) ⊂ Corridore(squadra)

Corridore(codice, squadra, nome, dataN, ruolo)  
foreign key: Corridore(squadra) ⊂ Squadra(codice)  
foreign key: Corridore(codice, squadra) ⊂ Natoln(codice, squadra)

Luogo(nome, altezza)

Natoln(codice, squadra, luogo)  
foreign key: Natoln(codice, squadra) ⊂ Corridore(codice, squadra)  
foreign key: Natoln(luogo) ⊂ Luogo(nome)

Edizione(anno)  
foreign key: Edizione(anno) ⊂ HaVinto(edizione)

HaVinto(edizione, codice, squadra)  
foreign key: HaVinto(edizione) ⊂ Edizione(anno)  
foreign key: HaVinto(codice, squadra) ⊂ Corridore(codice, squadra)

Tappa(numOrdine, edizione, data)  
foreign key: Tappa(edizione) ⊂ Edizione(anno)  
foreign key: Tappa(numOrdine, edizione) ⊂ Partenza(numOrdine, edizione)  
foreign key: Tappa(numOrdine, edizione) ⊂ Arrivo(numOrdine, edizione)  
vincolo di generalizzazione: Tappa(numOrdine, edizione) = ACronometro(numOrdine, edizione) U Normale(numOrdine, edizione)

HaTerminato(codice, squadra, numOrdine, edizione, posizTappa, posizFinale)  
foreign key: HaTerminato(codice, squadra) ⊂ Corridore(codice, squadra)  
foreign key: HaTerminato(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)

Partenza(numOrdine, edizione, luogo)  
foreign key: Partenza(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)  
foreign key: Partenza(luogo) ⊂ Luogo(nome)

Arrivo(numOrdine, edizione, luogo)  
foreign key: Arrivo(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)  
foreign key: Arrivo(luogo) ⊂ Luogo(nome)

ACronometro(numOrdine, edizione, lunghezza)  
foreign key: ACronometro(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)  
vincolo di generalizzazione: L'intersezione tra ACronometro(numOrdine, edizione) e Normale(numOrdine, edizione) è vuota

Normale(numOrdine, edizione, difficoltà)  
foreign key: Normale(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)

Intermedio(luogo, puntiPremio\*)  
foreign key: Intermedio(luogo) ⊂ Luogo(nome)  
inclusione: Intermedio(luogo) ⊂ Attraverso(intermedio)

Attraverso(intermedio, numOrdine, edizione, ordine)  
foreign key: Attraverso(intermedio) ⊂ Intermedio(luogo)  
foreign key: Attraverso(numOrdine, edizione) ⊂ Normale(numOrdine, edizione)

HaTerminatoCronometro(codice, squadra, numOrdine, edizione, tempo)  
foreign key: HaTerminatoCronometro(codice, squadra) ⊂ Corridore(codice, squadra)  
foreign key: HaTerminatoCronometro(numOrdine, edizione) ⊂ Tappa(numOrdine, edizione)

Vincolo esterno:  
1. Per ogni tupla in HaTerminatoCronometro c'è una tupla in HaTerminato con gli stessi valori negli attributi comuni.  
2. Per ogni tupla in HaTerminato con i valori di numOrdine ed edizione pari ad una tupla in ACronometro allora esiste una tupla in HaTerminatoCronometro con gli stessi valori negli attributi comuni.

### RISTRUTTURAZIONE SCHEMA RELAZIONALE

La specifica 1 sulle operazioni richiede un accorpamento tra Edizione ed HaVinto

Edizione(anno, codice, squadra)  
foreign key: Edizione(codice, squadra) ⊂ Corridore(codice, squadra)

La specifica 2 sulle operazioni richiede un accorpamento tra Tappa e ACronometro

Tappa(numOrdine, edizione, data, èACronometro, lunghezza\*)  
foreign key: Tappa(edizione) ⊂ Edizione(anno)  
foreign key: Tappa(numOrdine, edizione) ⊂ Partenza(numOrdine, edizione)  
foreign key: Tappa(numOrdine, edizione) ⊂ Arrivo(numOrdine, edizione)  
vincolo di generalizzazione: (SELECT numOrdine, edizione FROM Tappa WHERE èACronometro = false) ⊂ Normale(numOrdine, edizione)  
vincolo: Se èACronometro = true allora lunghezza != NULL. Se èACronometro = false allora lunghezza = NULL.  
vincolo di dominio: èACronometro = true OR èACronometro = false

Vanno poi modificate le relazioni che si riferiscono a Tappa di conseguenza a queste modifiche.

### PROBLEMA 3

```

SELECT rio1.citta, riv1.valore
FROM rivelazione riv1 JOIN rione rio1 ON riv1.cod_rione = rio1.codice
WHERE (riv1.giorno, riv1.mese, riv1.anno) = (SELECT riv2.giorno, riv2.mese, riv2.anno
                                             FROM rivelazione riv2 JOIN rione rio2 ON riv2.cod_rione = rio2.codice
                                             WHERE rio2.citta = rio1.citta
                                             ORDER BY riv2.anno, riv2.mese, riv2.giorno
                                             LIMIT 1)
  
```

### PROBLEMA 4

RIONE - PROT<sub>conce\_città</sub> (SEL VALORE>100 (RIVELAZIONE JOIN RIONE))

### PROBLEMA 5

F(B)  
inclusione: F(B) ⊆ E(F)

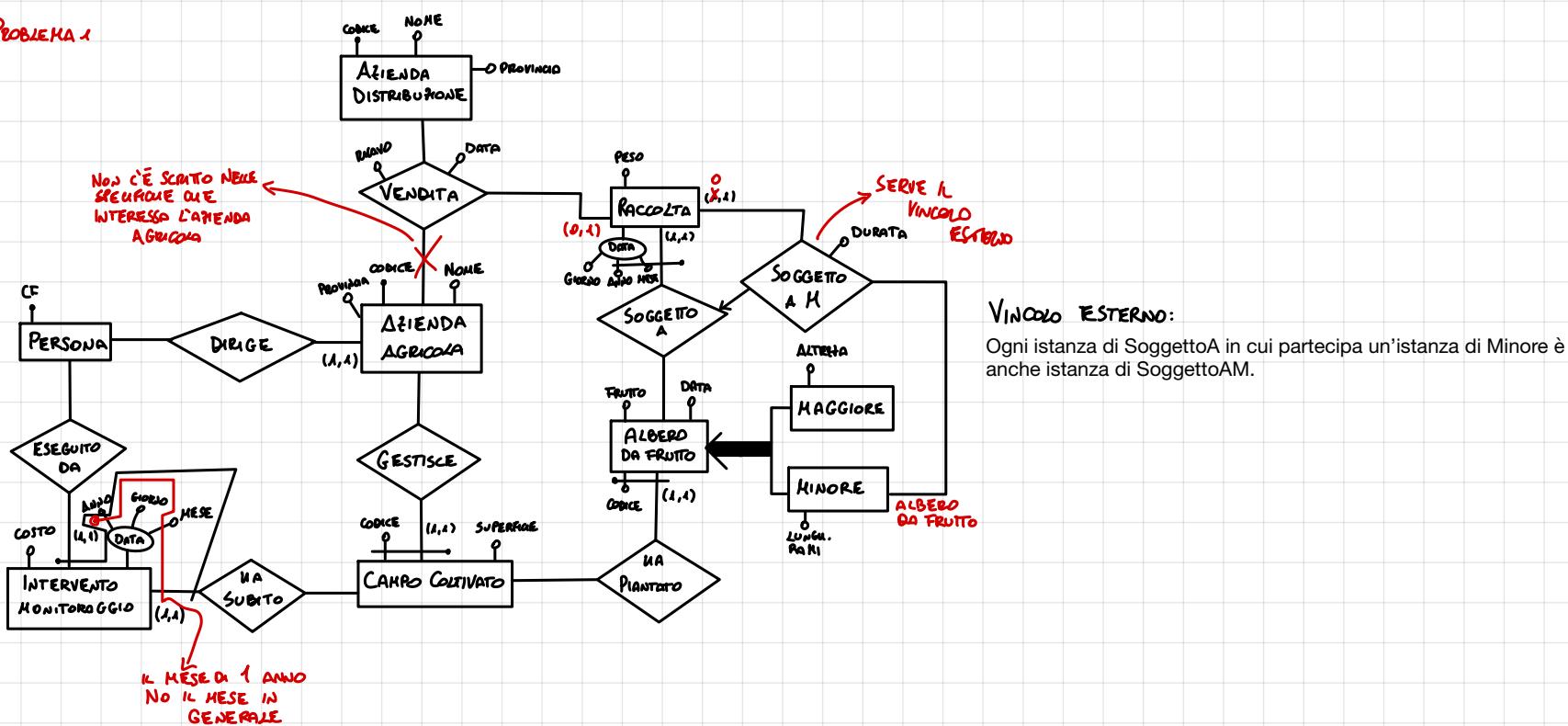
E(F,A)  
foreign key: E(F) ⊆ F(B)

G(F,A)  
foreign key: G(F,A) ⊆ E(F,A)  
vincolo di generalizzazione: L'intersezione tra G(F,A) e H(F,A) è vuota

H(F,A,C)  
foreign key: H(F,A) ⊆ E(F,A)

▷ ESAME 19 LUGLIO 2019

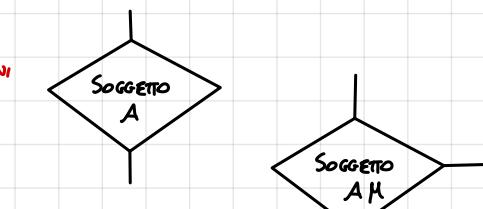
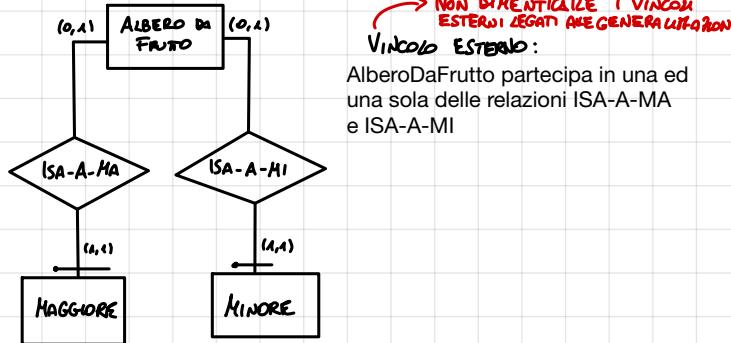
### PROBLEMA 1



### PROBLEMA 2

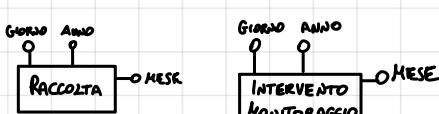
#### RISTRUTTURAZIONE SCHEMA CONCETTUALE

##### • ELIMINAZIONE ISA E GENERALIZZAZIONI



**VINCOLO ESTERNO:**  
Ogni istanza di SoggettoAM è anche istanza di SoggettoA

##### • ELIMINAZIONE ATTRIBUTI MULTIVALORE



### TRADUZIONE DIRETTA

AziendaAgricola(codice, nome, provincia)  
foreign key: AziendaAgricola(codice) ⊆ Dirige(azienda)

CampoColtivato(codice, azienda, superficie)  
foreign key: CampoColtivato(azienda) ⊆ AziendaAgricola(codice)

Persona(CF)

Dirige(azienda, persona)  
foreign key: Dirige(azienda) ⊆ AziendaAgricola(codice)  
foreign key: Dirige(persona) ⊆ Persona(CF)

InterventoMonitoraggio(codice, azienda, giorno, mese, anno, costo)

foreign key: InterventoMonitoraggio(codice, azienda) ⊑ CampoColtivato(codice, azienda)

chiave: (persona, anno, codice, azienda)

NON FA PARTE DELLA PRIMARY KEY PERCHÉ LO CARDINALITÀ (1,1) SU MONITORAGGIO

EseguitoDa(persona, codice, azienda, mese, anno)

foreign key: EseguitoDa(persona) ⊑ Persona(CF)

foreign key: EseguitoDa(codice, azienda, mese, anno) ⊑ InterventoMonitoraggio(codice, azienda, mese, anno)

chiave: (persona, codice, azienda, mese, anno) ↗ IDENTIFICATORE SU INTERVENTO MONITORAGGIO

AlberoDaFrutto(codice, codaz, codcampo, frutto, data)

foreign key: AlberoDaFrutto(codcampo, codaz) ⊑ CampoColtivato(codice, azienda)

vincolo di generalizzazione: AlberoDaFrutto(codice, codaz, codcampo) = Maggiore(codice, codaz, codcampo) ∪ Minore(codice, codaz, codcampo)

Maggiore(codice, codaz, codcampo, altezza)

foreign key: Maggiore(codice, codaz, codcampo) ⊑ AlberoDaFrutto(codice, codaz, codcampo)

vincolo di generalizzazione: L'intersezione tra Maggiore(codice, codaz, codcampo) e Minore(codice, codaz, codcampo) è vuota.

Minore(codice, codaz, codcampo, lunghRami)

foreign key: Minore(codice, codaz, codcampo) ⊑ AlberoDaFrutto(codice, codaz, codcampo)

Raccolta(codalbero, codaz, codcampo, mese, anno, giorno, peso)

foreign key: Raccolta(codalbero, codaz, codcampo) ⊑ AlberoDaFrutto(codice, codaz, codcampo)

AziendaDistribuzione(codice, nome, provincia)

Vendita(coddis, codalbero, codaz, codcampo, mese, anno, ricavo, data)

foreign key: Vendita(coddis) ⊑ AziendaDistribuzione(codice)

foreign key: Vendita(codalbero, codaz, codcampo, mese, anno) ⊑ Raccolta(codice, codaz, codcampo, mese, anno)

SoggettoAM(codalbero, codaz, codcampo, mese, anno, minore, codazm, codcampom, durata)

foreign key: SoggettoAM(codalbero, codaz, codcampo, mese, anno) ⊑ Raccolta(codalbero, codaz, codcampo, mese, anno)

foreign key: SoggettoAM(minore, codazm, codcampom) ⊑ Minore(codice, codaz, codcampo)

vincolo: (codalbero, codaz, codcampo) = (minore, codazm, codcampom)

ESPRIME GIÀ IL VINCULO  
DI ISA, QUINDI NON C'È  
BISOGNO DEL VINCULO

Vincoli esterni:

1. Per ogni tupla  $\langle c1, c2, c3, m, a, g, p \rangle$  in Raccolta di cui la tripla  $\langle c1, c2, c3 \rangle$  è contenuta in una tupla di Minore allora esiste una tupla  $\langle c1, c2, c3, m, a, c1, c2, c3, d \rangle$  in SoggettoAM.

## RISTRUTTURAZIONE DELLO SCHEMA RELAZIONALE

La specifica 1 sulle operazioni richiede l'accorpamento tra AlberoDaFrutto e Maggiore

AlberoDaFrutto(codice, codaz, codcampo, frutto, data, èMaggiore, altezza\*)

foreign key: AlberoDaFrutto(codcampo, codaz) ⊑ CampoColtivato(codice, azienda)

vincolo: Se èMaggiore = true allora altezza != NULL. Se èMaggiore = false allora altezza = NULL

vincolo di dominio: èMaggiore = true OR èMaggiore = false

vincolo di generalizzazione: (SELECT codice, codaz, codcampo FROM AlberoDaFrutto WHERE èMaggiore = false) ⊑ Minore(codice, codaz, codcampo)

Minore(codice, codaz, codcampo, lunghRami)

vincolo: (codice, codaz, codcampo) ⊑ (SELECT codice, codaz, codcampo FROM AlberoDaFrutto WHERE èMaggiore = false)

La specifica 2 sulle operazioni richiede di modificare CampoColtivato nel seguente modo

CampoColtivato(codice, azienda, superficie, persona\*, giorno\*, mese\*, anno\*)

foreign key: CampoColtivato(azienda) ⊑ AziendaAgricola(codice)

vincolo: (persona, giorno, mese, anno) = (SELECT persona, giorno, mese, anno FROM InterventoMonitoraggio WHERE InterventoMonitoraggio.codice = codice AND InterventoMonitoraggio.azienda = azienda ORDER BY (anno, mese, giorno) LIMIT 1)

vincolo: Gli attributi persona, giorno, mese ed anno non possono essere NULL se almeno uno degli attributi è diverso da NUL e non possono essere diversi da NULL se almeno uno è NULL.

## PROBLEMA 3

SELECT citta, max(valore)

FROM rione NATURAL JOIN rilevazione

WHERE (SELECT count(\*) FROM rione r1 WHERE r1.citta = rione.citta) >= 10) AND anno = 2018

GROUP BY citta

## PROBLEMA 4

PLOT(RILEVAZIONE) - PLOT ( SEL COD.RIONE,CITTA VALORE <= 50 )

## PROBLEMA 5

F(D)

inclusione: F(D) ⊑ E(F)

E(A,F,B)

foreign key: E(F) ⊑ F(D)

G(A,F)

foreign key: G(A,F) ⊑ E(A,F)

vincolo di generalizzazione: L'intersezione tra G(A,F) e H(A,F) è vuota

H(A,F,C)

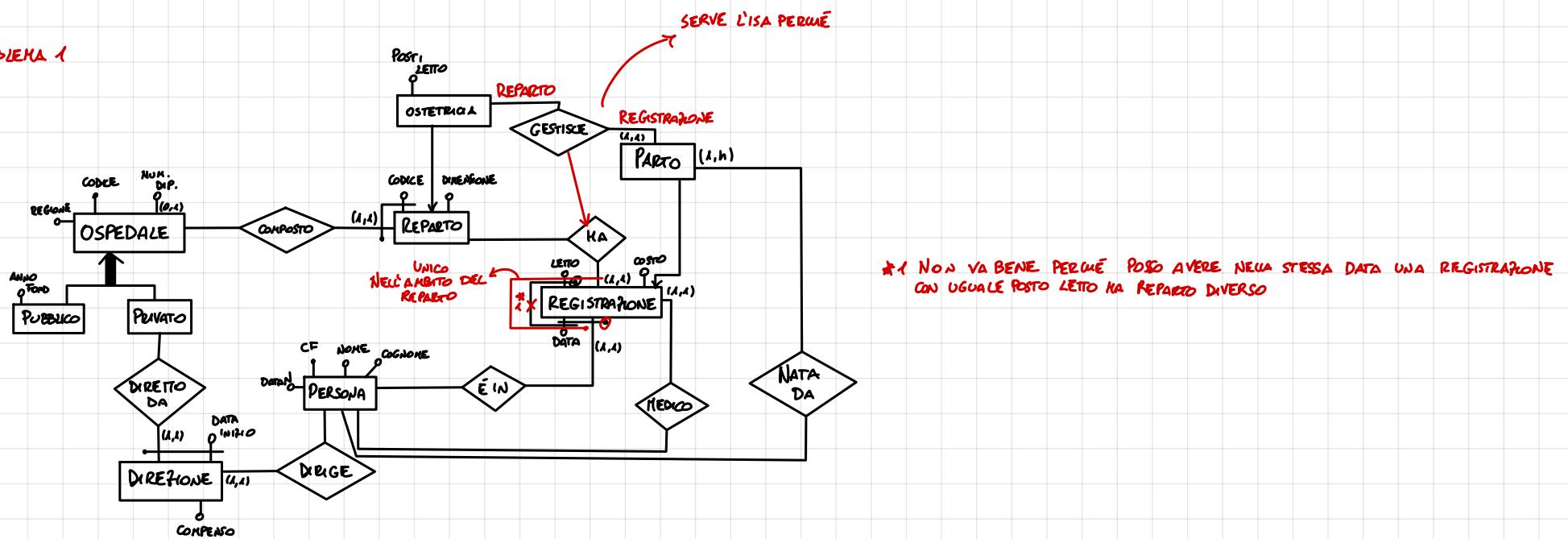
foreign key: H(A,F) ⊑ E(A,F)

Q(A,F,D)

foreign key: Q(A,F) ⊑ H(A,F)

foreign key: Q(D) ⊑ F(D)

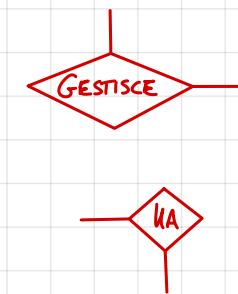
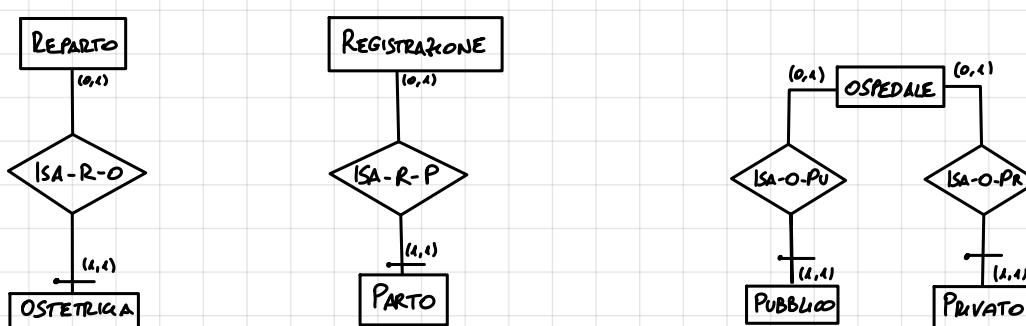
## PROBLEMA 1



## PROBLEMA 2

## RISTRUTTURAZIONE SCHEMA CONCETTUALE

- ELIMINAZIONE ISA E GENERALIZZAZIONI



## VINCOLO ESTERNO:

Ogni istanza di Ospedale partecipa in una ed una sola delle due relazioni tra ISA-O-PU e ISA-O-PR.

## VINCOLO ESTERNO:

Ogni istanza di Gestisce è anche istanza di Ha.

## TRADUZIONE DIRETTA

Ospedale(codice, regione, numDip\*)  
vincolo di generalizzazione: Ospedale(codice) = Pubblico(codice) U Privato(codice)

Reparto(codice, ospedale, dimensione)  
foreign key: Reparto(ospedale) C Ospedale(codice)

Pubblico(ospedale, annoFond)  
foreign key: Pubblico(ospedale) C Ospedale(codice)  
vincolo di generalizzazione: L'intersezione tra Pubblico(ospedale) e Privato(ospedale) è vuota

Privato(ospedale)  
foreign key: Privato(ospedale) C Ospedale(codice)

Persona(CF, nome, cognome, dataN)

Direzione(ospedale, datalnizio, compenso)  
foreign key: Direzione(ospedale) C Privato(ospedale)  
foreign key: Direzione(ospedale, datalnizio) C Dirige(ospedale, datalnizio)

Dirige(ospedale, datalnizio, persona)  
foreign key: Dirige(ospedale, datalnizio) C Direzione(ospedale, datalnizio)  
foreign key: Dirige(persona) C Persona(CF)

Registrazione(letto, data, persona, costo)  
foreign key: Registrazione(persona) C Persona(CF)  
foreign key: Registrazione(persona, data) C Ha(persona, data)  
foreign key: Registrazione(persona, data) C Medico(ricoverato, data)

Ha(persona, data, codrep, ospedale)  
foreign key: Ha(persona, data) C Registrazione(persona, data)  
foreign key: Ha(codrep, ospedale) C Reparto(codice, ospedale)  
vincolo di chiave: Nel join naturale tra Registrazione ed Ha gli attributi data, letto, codrep, ospedale compongono una chiave.

Medico(ricoverato, data, persona)  
foreign key: Medico(ricoverato, data) C Registrazione(persona, data)  
foreign key: Medico(persona) C Persona(CF)

Ostetricia(codrep, ospedale, postiLetto)  
foreign key: Ostetricia(codrep, ospedale) C Reparto(codice, ospedale)

Parto(persona, data)  
foreign key: Parto(persona, data) C Registrazione(persona, data)  
foreign key: Parto(persona, data) C Gestisce(persona, data)  
inclusione: Parto(persona, data) C NataDa(persona, data)

Gestisce(persona, data, codrep, ospedale)  
foreign key: Gestisce(persona, data) C Parto(persona, data)  
foreign key: Gestisce(codrep, ospedale) C Ostetricia(codrep, ospedale)  
vincolo: Gestisce(persona, data, codrep, ospedale) C Ha(persona, data, codrep, ospedale)

NataDa(persona, data, nato)  
foreign key: NataDa(persona, data) C Parto(persona, data)  
foreign key: NataDa(nato) C Persona(CF)

## RISTRUTTURAZIONE SCHEMA LOGICO

Per evitare i valori nulli bisogna separare Ospedale in due relazioni

Ospedale(codice, regione)

vincolo di generalizzazione: Ospedale(codice) = Pubblico(codice) U Privato(codice)

OspedaleNumDip(codice, numDip)

foreign key: OspedaleNumDip(codice) ⊑ Ospedale(codice)

## PROBLEMA 3

$\text{PROT}_{\text{PROVINCIA}}(\text{COMUNE}) - \text{PROT}_{\text{PROVINCIA}}(\text{SEL}(\text{COMUNE}))$

$\text{NONE! : PROVINCIA AND ANNO} \geq 2015$

## PROBLEMA 4

SELECT provincia, comune

FROM comune JOIN residenza ON nome = comune

WHERE salario >= ALL (SELECT salario FROM residenza r JOIN comune c ON r.comune =

c.nome WHERE c.provincia = provincia)

## PROBLEMA 5

F(D)

foreign key: F(D) ⊑ E(F)

E(A, F, B)

foreign key: E(F) ⊑ F(D)

G(A, F)

foreign key: G(A, F) ⊑ E(A, F)

Q(A, F, D)

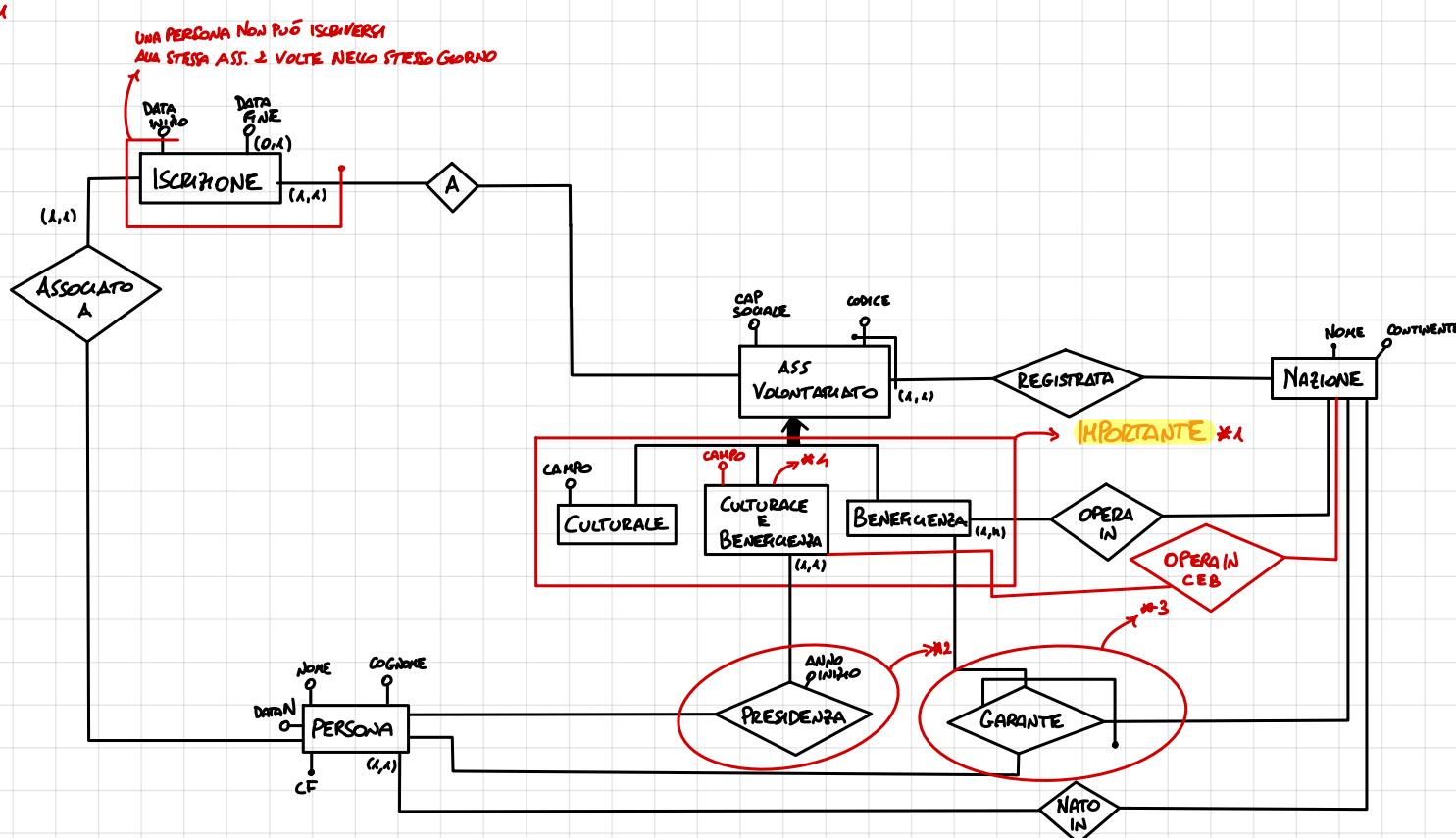
foreign key: Q(A, F) ⊑ G(A, F)

foreign key: Q(D) ⊑ F(D)

BASTA QUESTO, NON SERVE IL VINCOLO DI GENERALIZZAZIONE

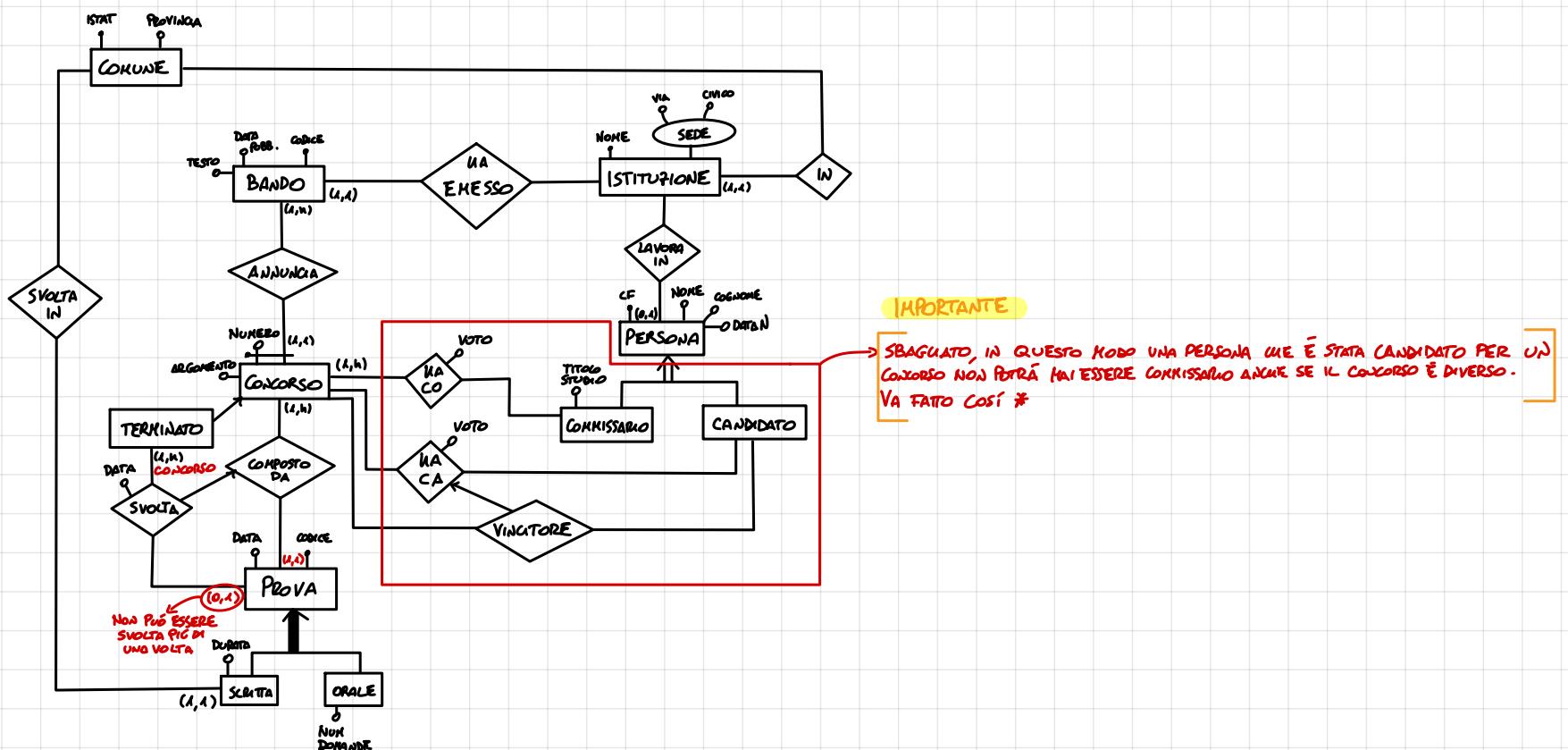
## ► ESAME 26 OTTOBRE 2020 (SOLO PROBLEMA 1)

### PROBLEMA 1

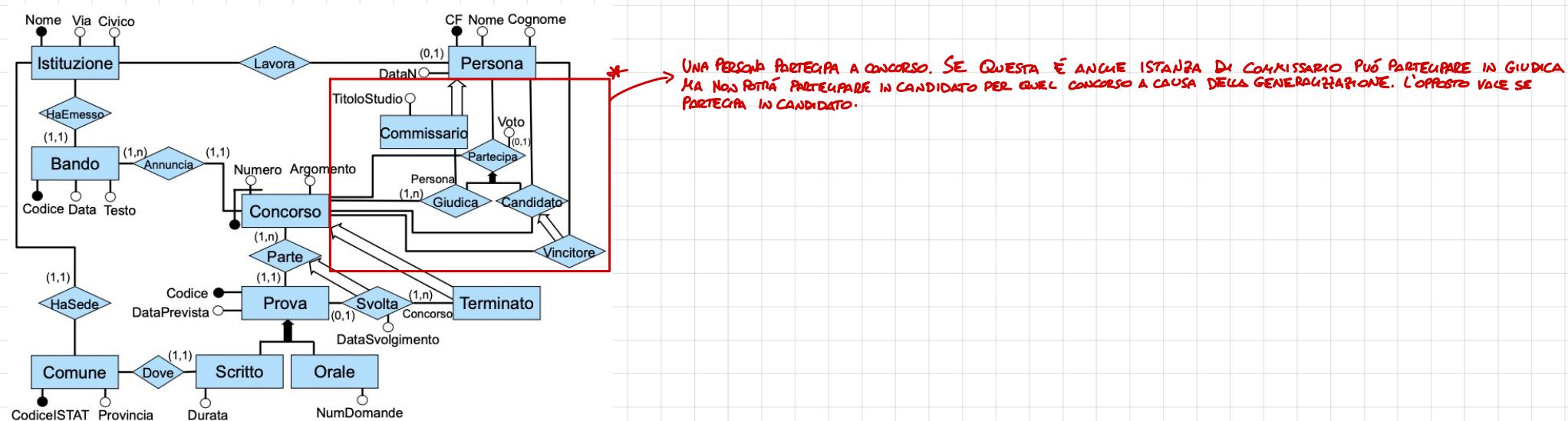


## ► ESAME 24 GENNAIO 2020

### PROBLEMA 1



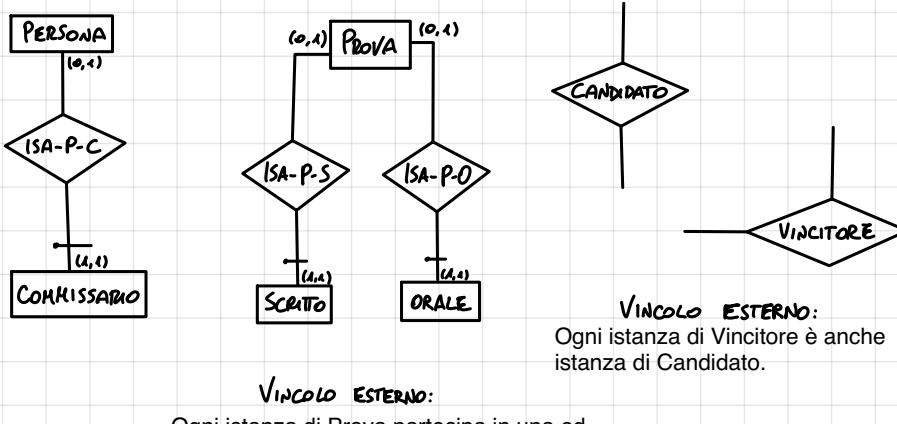
## SOLUZIONE PROFESSORE



## PROBLEMA 2

### RISTRUTTURAZIONE SCHHEMA CONCETUALE

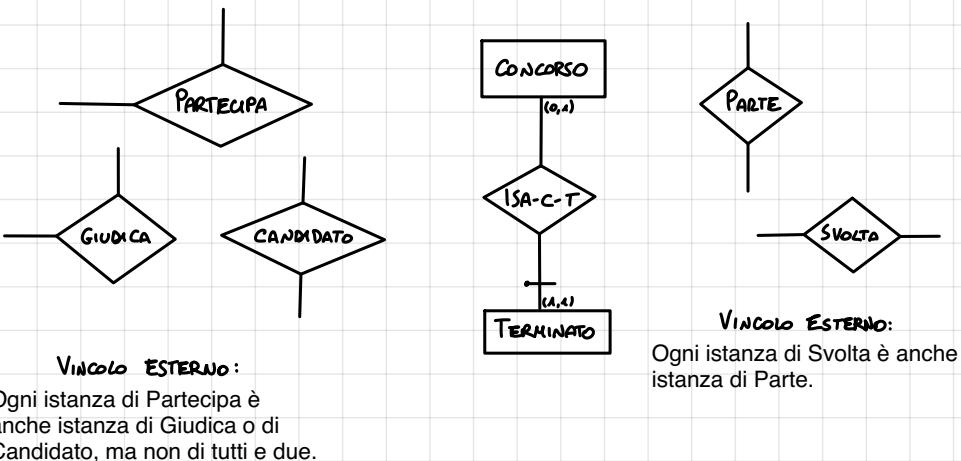
#### • ESUMINAZIONE ISA E GENERALIZZAZIONI



**VINCOLO ESTERNO:**  
Ogni istanza di Vincitore è anche istanza di Candidato.

**VINCOLO ESTERNO:**

Ogni istanza di Prova partecipa in una ed una sola delle relazioni tra ISA-P-S e ISA-P-O.



**VINCOLO ESTERNO:**  
Ogni istanza di Partecipa è anche istanza di Giudica o di Candidato, ma non di tutti e due.

**VINCOLO ESTERNO:**  
Ogni istanza di Svolta è anche istanza di Parte.

### TRADUZIONE DIRETTA

Istituzione(`nome`, `via`, `civico`)

foreign key: Istituzione(`nome`) ⊑ HaSede(`istituzione`)

Persona(`CF`, `nome`, `cognome`, `DataN`)

Lavora(`persona`, `istituzione`)

foreign key: Lavora(`persona`) ⊑ Persona(`CF`)

foreign key: Lavora(`istituzione`) ⊑ Istituzione(`nome`)

HaSede(`istituzione`, `comune`)

foreign key: HaSede(`istituzione`) ⊑ Istituzione(`nome`)

foreign key: HaSede(`comune`) ⊑ Comune(`istat`)

Comune(`istat`, `provincia`)

Bando(`codice`, `data`, `testo`)

foreign key: Bando(`codice`) ⊑ HaEmesso(`bando`)

inclusione: Bando(`codice`) ⊑ Concorso(`bando`)

HaEmesso(`bando`, `istituzione`)

foreign key: HaEmesso(`bando`) ⊑ Bando(`codice`)

foreign key: HaEmesso(`istituzione`) ⊑ Istituzione(`nome`)

Concorso(`bando`, `numero`, `argomento`)

foreign key: Concorso(`bando`) ⊑ Bando(`codice`)

inclusione: Concorso(`bando`, `numero`) ⊑ Parte(`bando`, `numero`)

inclusione: Concorso(`bando`, `numero`) ⊑ Giudica(`bando`, `numero`)

Prova(`codice`, `dataPrevista`)

foreign key: Prova(`codice`) ⊑ Parte(`prova`)

vincolo di generalizzazione: Prova(`codice`) = Scritto(`codice`) U Orale(`codice`)

Parte(`prova`, `bando`, `numero`)

foreign key: Parte(`prova`) ⊑ Prova(`codice`)

foreign key: Parte(`bando`, `numero`) ⊑ Concorso(`bando`, `numero`)

Scritto(`prova`, `durata`)

foreign key: Scritto(`prova`) ⊑ Prova(`codice`)

foreign key: Scritto(`prova`) ⊑ Dove(`prova`)

vincolo di generalizzazione: L'intersezione tra Scritto(`prova`) e Orale(`prova`) è vuota.

Dove(`prova`, `comune`)

foreign key: Dove(`prova`) ⊑ Scritto(`prova`)

foreign key: Dove(`comune`) ⊑ Comune(`istat`)

Orale(`prova`, `numDomande`)

foreign key: Orale(`prova`) ⊑ Prova(`codice`)

Terminato(`bando`, `numero`)

foreign key: Terminato(`bando`, `numero`) ⊑ Concorso(`bando`, `numero`)

inclusione: Terminato(`numero`, `bando`) ⊑ Svolta(`numero`, `bando`)

Svolta(`prova`, `bando`, `numero`, `dataSvolgimento`)

foreign key: Svolta(`prova`) ⊑ Prova(`codice`)

foreign key: Svolta(`bando`, `numero`) ⊑ Terminato(`bando`, `numero`)

inclusione: Svolta(`prova`, `bando`, `numero`) ⊑ Parte(`prova`, `bando`, `numero`)

Giudica(commissario, bando, numero)

foreign key: Giudica(commissario) ⊂ Commissario(CF)

foreign key: Giudica(bando, numero) ⊂ Concorso(bando, numero)

vincolo di generalizzazione: L'intersezione tra Giudica(commissario, bando, numero) e Candidato(persona, bando, numero) è vuota.

Candidato(persona, bando, numero)

foreign key: Candidato(persona) ⊂ Persona(CF)

foreign key: Candidato(bando, numero) ⊂ Concorso(bando, numero)

Vincitore(persona, bando, numero)

foreign key: Vincitoreo(persona) ⊂ Persona(CF)

foreign key: Vincitore(bando, numero) ⊂ Concorso(bando, numero)

foreign key: Vincitore(persona, bando, numero) ⊂ Candidato(persona, bando, numero)

Commissario(CF, titoloStudio)

foreign key: Commissario(CF) ⊂ Persona(CF)

Partecipa(persona, bando, numero, voto\*)

foreign key: Partecipa(persona) ⊂ Persona(CF)

foreign key: Partecipa(bando, numero) ⊂ Concorso(bando, numero)

vincolo di generalizzazione: Partecipa(persona, bando, numero) = Giudica(commissario, bando, numero) U Candidato(persona, bando, numero)

## RISTRUTTURAZIONE SCHEMA LOGICO

Prova(codice, dataPrevista, dataSvolgimento\*, tipoProva, durata\*, comune\*, numDomande\*)

foreign key: Prova(codice) ⊂ Parte(prova)

foreign key: Prova(comune) ⊂ Comune(istat)

vincolo di generalizzazione: Prova(codice) = Scritto(codice) U Orale(codice)

vincolo: dataSvolgimento = (SELECT dataSvolgimento FROM Svolta WHERE prova = codice)

vincolo di dominio: tipoProva = 'Scritta' OR tipoProva = 'Orale'

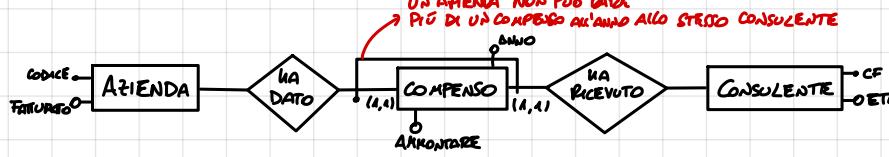
vincolo: Se tipoProva = 'Scritta' allora durata != NULL e comune != NULL e numDomande = NULL

vincolo: Se tipoProva = 'Orale' allora numDomande != NULL e durata = NULL e comune = NULL

## PROBLEMA 3

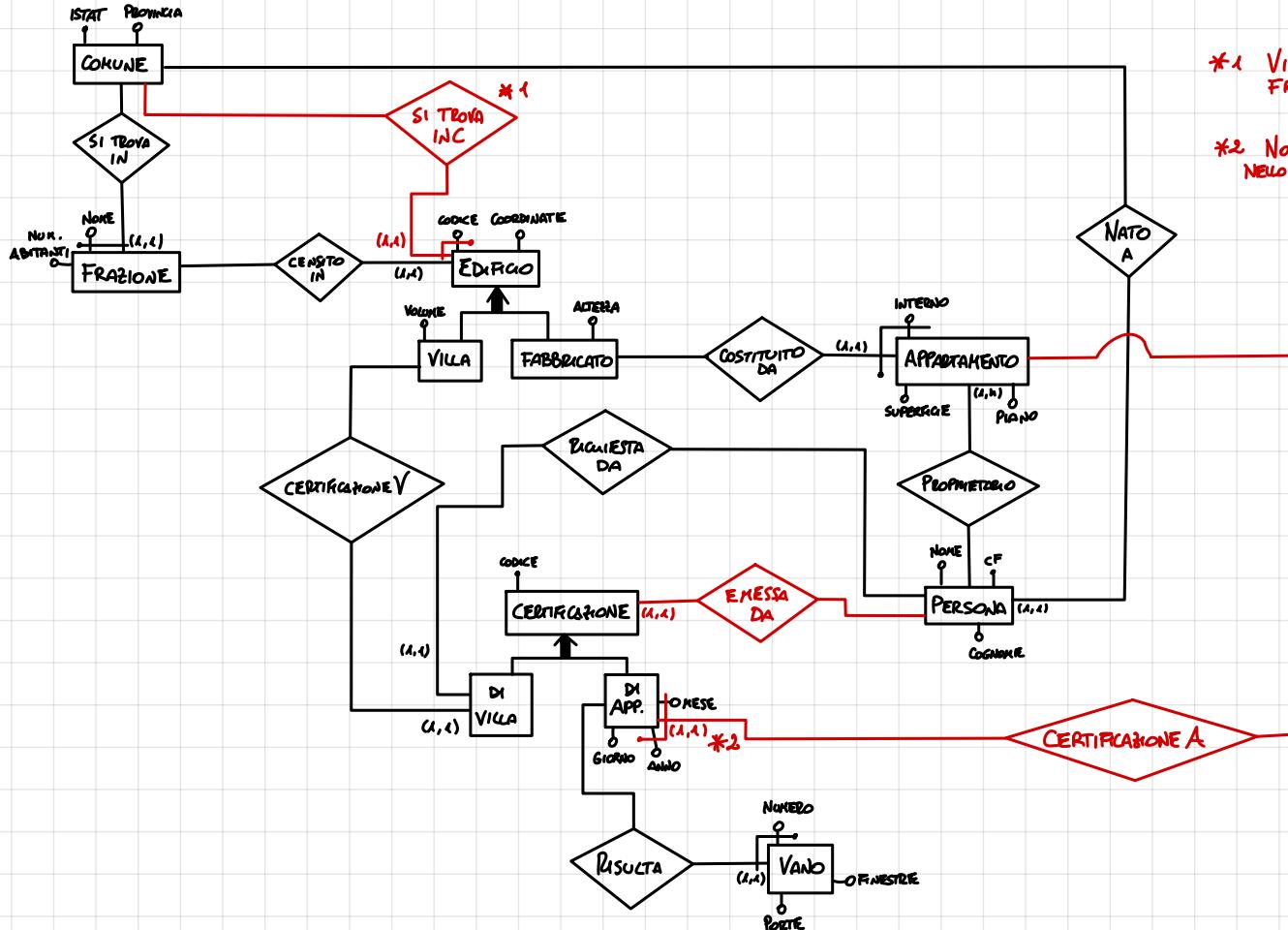
PERSONA - Prot ((PERSONA JOIN RESIDENTA) JOIN NASCITA))  
CF, DATA\_NASCITA

## PROBLEMA 4



► ESAME 18 FEBBRAIO 2020

## PROBLEMA 1



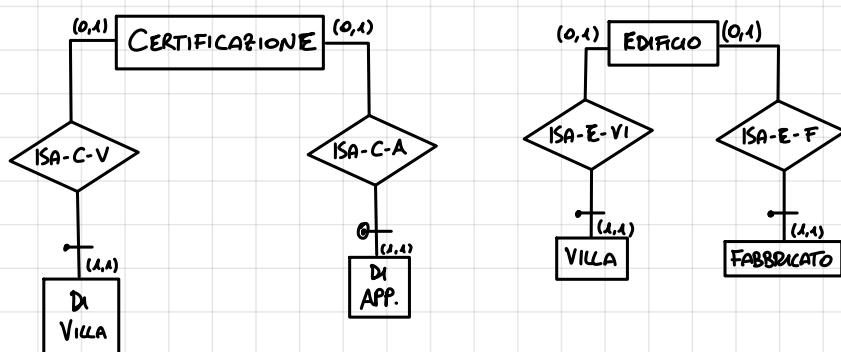
## VINCOLI ESTERNI:

- Ogni istanza di Edificio partecipa in SiTrovalnC con l'istanza di Comune in cui si trova la Frazione in cui l'istanza di Edificio è censito.

## PROBLEMA 2

### RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

#### ESEMPLIFICAZIONE ISA E GENERALIZZAZIONI



#### Vincoli esterni:

1. Certificazione partecipa in una ed una sola delle relazioni ISA-C-V e ISA-C-A
2. Edificio partecipa in una ed una sola delle relazioni ISA-E-Vi e ISA-E-F

### TRADUZIONE DIRETTA

#### Certificazione(codice)

foreign key: Certificazione(codice) ⊑ EmessaDa(certificazione)

vincolo di generalizzazione: Certificazione(codice) = DiVilla(codice) U DiApp(codice)

#### Persona(CF, nome, cognome)

foreign key: Persona(CF) ⊑ NatoA(persona)

#### EmessaDa(certificazione, persona)

foreign key: EmessaDa(certificazione) ⊑ Certificazione(codice)

foreign key: EmessaDa(persona) ⊑ Persona(CF)

#### DiVilla(codice)

foreign key: DiVilla(codice) ⊑ Certificazione(codice)

foreign key: DiVilla(codice) ⊑ RichiestaDa(certificazione)

foreign key: DiVilla(codice) ⊑ CertificazioneV(certificazione)

vincolo di generalizzazione: L'intersezione tra DiVilla(codice) e DiApp(codice) è vuota

#### DiApp(codice, giorno, mese, anno)

foreign key: DiApp(codice) ⊑ Certificazione(codice)

foreign key: DiApp(codice) ⊑ CertificazioneA(codCert)

#### RichiestaDa(certificazione, persona)

foreign key: RichiestaDa(certificazione) ⊑ DiVilla(codice)

foreign key: RichiestaDa(certificazione) ⊑ Persona(CF)

#### CertificazioneV(certificazione, codVilla, comune)

foreign key: CertificazioneV(certificazione) ⊑ DiVilla(codice)

foreign key: Certificazione(codVilla, comune) ⊑ Villa(codVilla, comune)

#### Comune(istat, provincia)

#### Frazione(nome, comune, numAbitanti)

foreign key: Frazione(comune) ⊑ Comune(istat)

#### Edificio(codice, comune, coordinate)

foreign key: Edificio(comune) ⊑ Comune(istat)

foreign key: Edificio(codice, comune) ⊑ Censitoin(codEd, comuneEd)

vincolo di generalizzazione: Edificio(codice, comune) = Villa(codice, comune) U Fabbricato(codice, comune)

vincolo: comune = (SELECT comuneFr FROM censitoin WHERE codEd = codice)

#### Censitoin(codEd, comuneEd, frazione, comuneFr)

foreign key: Censitoin(codEd, comuneEd) ⊑ Edificio(codice, comune)

foreign key: Censitoin(frazione, comuneFr) ⊑ Frazione(nome, comune)

vincolo: comuneEd = comuneFr

#### Villa(codice, comune, volume)

foreign key: Villa(codice, comune) ⊑ Edificio(codice, comune)

vincolo di generalizzazione: L'intersezione tra Villa(codice, comune) e Fabbricato(codice, comune) è vuota.

#### Fabbricato(codice, comune, altezza)

foreign key: Fabbricato(codice, comune) ⊑ Edificio(codice, comune)

#### NatoA(persona, comune)

foreign key: NatoA(persona) ⊑ Persona(CF)

foreign key: NatoA(comune) ⊑ Comune(istat)

#### Appartamento(codFab, comune, interno, superficie, piano)

foreign key: Appartamento(codFab, comune) ⊑ Fabbricato(codice, comune)

foreign key: Appartamento(codFab, comune, interno) ⊑ Proprietario(codFab, comune, interno)

#### Proprietario(codFab, comune, interno, persona)

foreign key: Proprietario(codFab, comune, interno) ⊑ Appartamento(codFab, comune, interno)

foreign key: Proprietario(persona) ⊑ Persona(CF)

#### CertificazioneA(codCert, codFab, comune, interno)

foreign key: CertificazioneA(codCert) ⊑ DiApp(codice)

foreign key: CertificazioneA(codFab, comune, interno) ⊑ Appartamento(codFab, comune, interno)

#### Vano(codCert, numero, porte, finestre)

foreign key: Vano(codCert) ⊑ DiApp(codice)

#### Vincoli esterni:

1. Nel join tra DiApp e CertificazioneA gli attributi (mese, anno, codFab, comune, interno) sono chiave.

## RISTRUTTURAZIONE SCHEMA LOGICO

Per la specifica 1 modifco Comune nel seguente modo

Comune(istat, provincia, numAbitanti)  
vincolo: numAbitanti = (SELECT sum(numAbitanti) FROM frazione WHERE comune = istat)

Per la specifica 3 bisogna fare un accorpamento tra Edificio, Villa e Fabbricato

Edificio(codice, comune, coordinate, tipo, volume\*, altezza\*)  
foreign key: Edificio(comune) ⊂ Comune(istat)  
vincolo: comune = (SELECT comuneFr FROM censitoIN WHERE codEd = codice)  
vincolo di dominio: tipo = 'villa' OR tipo = 'fabbricato'  
vincolo: Se tipo = 'villa' allora volume != NULL e altezza = NULL  
vincolo: Se tipo = 'fabbricato' allora altezza != NULL e volume = NULL

Per la specifica 4 bisogna fare un accorpamento tra Edificio ed CensitoIn

Edificio(codice, comune, coordinate, tipo, volume\*, altezza\*, frazione, comuneFr)  
foreign key: Edificio(comune) ⊂ Comune(istat)  
vincolo: comune = (SELECT comuneFr FROM censitoIN WHERE codEd = codice)  
vincolo di dominio: tipo = 'villa' OR tipo = 'fabbricato'  
vincolo: Se tipo = 'villa' allora volume != NULL e altezza = NULL  
vincolo: Se tipo = 'fabbricato' allora altezza != NULL e volume = NULL  
foreign key: Edificio(frazione, comuneFr) ⊂ Frazione(nome, comune)

## PROBLEMA 3

1. SELECT s1.CF  
FROM socio s1  
WHERE s1.categoria <= ALL (SELECT s2.categoria FROM socio s2 JOIN partita ON s1.CF = vincente AND s2.CF = perdente)

2. La query non è corretta perchè, nel caso in cui la SELECT restituisse un valore nullo anche per una sola tupla, il confronto <= darebbe FALSE per quella tupla.

SELECT s1.CF  
FROM socio s1  
WHERE s1.categoria != NULL AND s1.categoria <= ALL (SELECT s2.categoria FROM socio s2 JOIN partita ON s1.CF = vincente AND s2.CF = perdente AND s2.categoria != NULL)

## PROBLEMA 4

1. Si esiste:  
Istanze(I,A) = {a1}  
Istanze(I,B) = {a1}  
Istanze(I,D) = {d1,d2,d3}  
Istanze(I,R1) = {<a1,d1>, <a1,d2>, <a1,d3>}  
Istanze(I,R2) = {<a1,d1>, <a1,d2>, <a1,d3>, }

2. Non esiste perchè il vincolo di cardinalità (4,4) per C non può essere soddisfatto. Infatti viene ereditata la cardinalità massima 3 da A quindi in R1 non avrò mai più di 3 istanze in R1 per un'istanza di A, da ciò si vede che non potrò mai ereditare più di 3 istanze per R3.