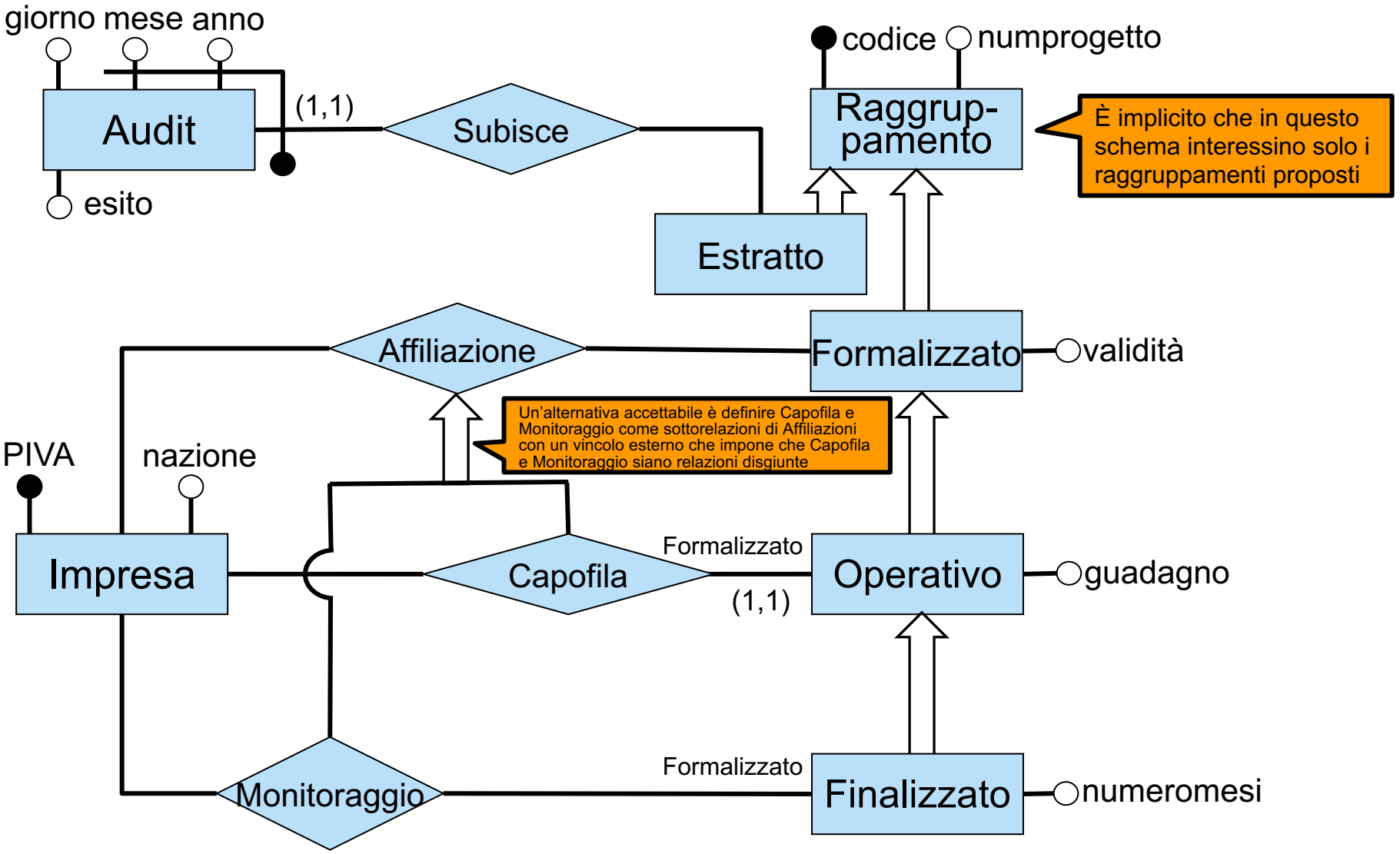
Basi di dati

Soluzione dei problemi proposti nell'appello del 23-02-2024 Compito B

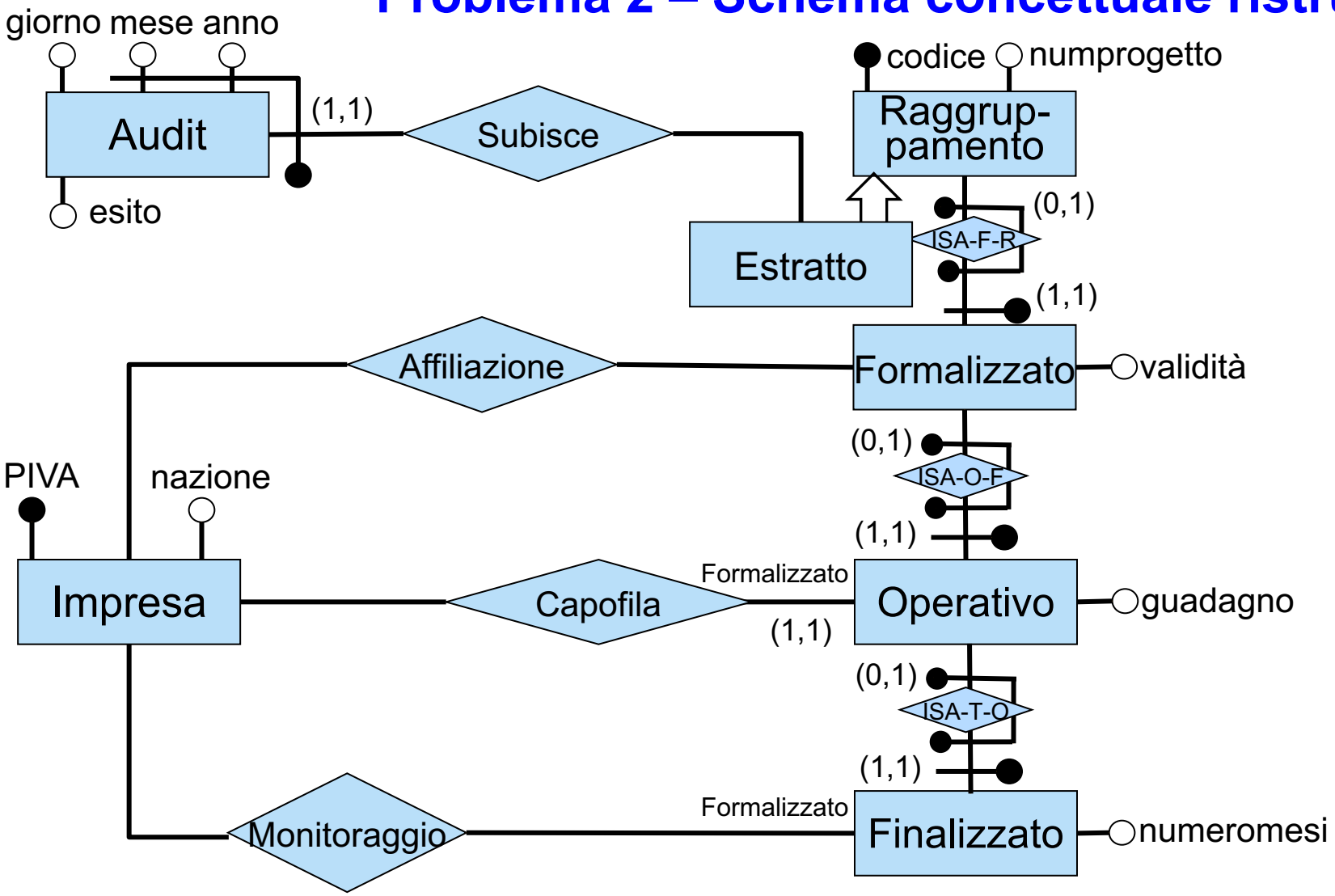
Maurizio Lenzerini

Anno Accademico 2023/24

Problema 1 – Schema concettuale



Problema 2 – Schema concettuale ristrutturato



Vincolo esterno: - per ogni <Impresa:m,Formalizzato:t> in Capofila tale che <Operativo:t,Formalizzato:f> è in ISA-O-F, si ha che <Impresa:m,Formalizzato:f> è in Affiliazione;
- per ogni <Impresa:m,Formalizzato:t> in Monitoraggio tale che <Terminato:t,Operativo:v> in è ISA-T-O e <Operativo:v,Formalizzato,f> è in ISA:O-F, si ha che <Impresa:m,Formalizzato:v> non è in Capofila e <Impresa:m,Formalizzato:f> è in Affiliazione.

Problema 2 – Schema logico da traduzione diretta

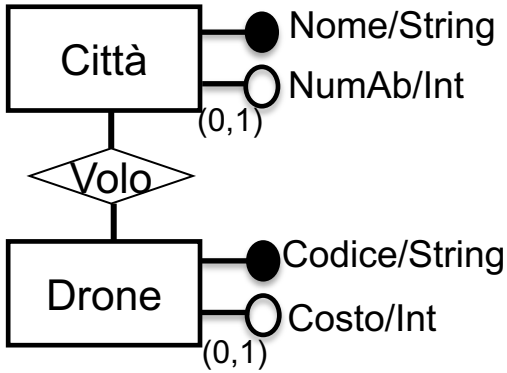
Raggruppamento(codice,numprogetto)
Estratto(codice)
  foreign key: Estratto[codice] $\subseteq$ Raggruppamento[codice]
Formalizzato(codice,validità)
  foreign key: Formalizzato[codice] $\subseteq$ Raggruppamento[codice]
Operativo(codice,guadagno)
  foreign key: Operativo[codice] $\subseteq$ Formalizzato[codice]
  foreign key: Operativo[codice] $\subseteq$ Capofila[codice]
Finalizzato(codice,numeromesi)
  foreign key: Finalizzato[codice] $\subseteq$ Operativo[codice]
Impresa(piva,nazione)
Affiliazione(formalizzato,impresa)
  foreign key : Affiliazione[formalizzato] $\subseteq$ Formalizzato[codice]
  foreign key : Affiliazione[impresa] $\subseteq$ Impresa[piva]
Capofila(formalizzato,impresa)
  foreign key: Capofila[formalizzato] $\subseteq$ Operativo[codice]
  foreign key: Capofila[formalizzato,impresa] $\subseteq$ Affiliazione[formalizzato,impresa]
  disgiuntezza: Capofila[formalizzato,impresa] $\cap$ Monitoraggio[formalizzato,impresa] = $\emptyset$
Monitoraggio(formalizzato,impresa)
  foreign key: Monitoraggio[formalizzato] $\subseteq$ Terminato[codice]
  foreign key: Monitoraggio[formalizzato,impresa] $\subseteq$ Affiliazione[formalizzato,impresa]
Audit(estratto,mese,anno,giorno,esito)
  foreign key: Audit[estratto] $\subseteq$ Estratto[codice]

Problema 2 – schema logico ristrutturato

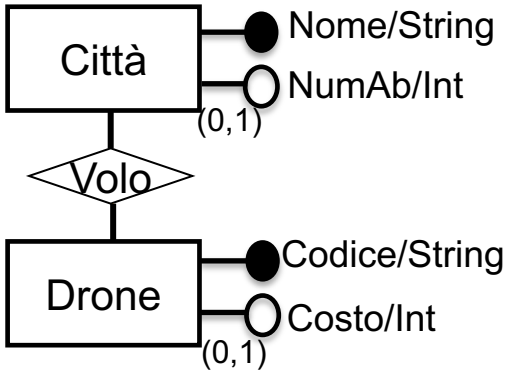
L'indicazione di progetto suggerisce di accorpare Raggruppamento e Formalizzato debolmente accoppiati e poi di accorpare anche la relazione risultante e Operativo, a loro volta debolmente accoppiati.

Raggruppamento(codice,numprogetto,validità*,capofila*,guadagno*)
  vincolo di tupla: capofila is null se e solo se guadagno is null and
                    se validità is null allora capofila is null
  foreign key: Raggruppamento[capofila] ⊆ Impresa[piva]
  foreign key: Raggruppamento[codice,capofila] ⊆ Affiliazione[formalizzato,impresa]
Estratto(codice)
  foreign key: Estratto[codice] ⊆ Raggruppamento[codice]
Finalizzato(codice,numeromesi)
  foreign key: Finalizzato[codice] ⊆ (select codice
                                     from Raggruppamento where capofila is not null)
Impresa(piva,nazione)
Affiliazione(formalizzato,impresa)
  foreign key: Affiliazione[formalizzato] ⊆ (select codice
                                            from Raggruppamento where validità is not null)
  foreign key: Affiliazione[impresa] ⊆ Impresa[piva]
Monitoraggio(formalizzato,impresa)
  foreign key: Monitoraggioformalizzato,impresa] ⊆ Affiliazione[formalizzato,impresa]
  foreign key: Monitoraggio[formalizzato] ⊆ Terminato[codice]
  disgiuntezza: Monitoraggio[formalizzato,impresa] ∩ Capofila[formalizzato,impresa] = ∅
Audit(estratto,mese,anno,giorno,esito)
  foreign key: Audit[estratto] ⊆ Estratto[codice]

Problema 3 – soluzione



schema concettuale di partenza



schema concettuale ristrutturato

schema logico prodotto dalla traduzione diretta

Città(Nome,NumAb*)

Drone(Codice,Costo*)

Volo(Drone,Città)

foreign key: Volo[Drone] $\subseteq$ Drone[Codice]

foreign key: Volo[Città] $\subseteq$ Città[Nome]

schema logico prodotto dalla traduzione diretta espresso in SQL

```
create table Città (
   nome varchar primary key,
   numAb varchar);

create table Drone (
   codice varchar primary key,
   costo integer);

create table Volo (
   drone varchar,
   città varchar,
   foreign key (drone) references Drone,
   foreign key (città) reference Città);
```

Problema 3 – soluzione

Per seguire l'indicazione di progetto dobbiamo fare in modo che il sistema limiti al massimo il rifiuto delle operazioni di inserimento, cancellazione ed aggiornamento richieste dagli utenti. Ricordiamo che il sistema rifiuta un'operazione quando la base di dati risultante dalla esecuzione di tale operazione viola almeno un vincolo di integrità.

1. Vincoli di chiave: un tale vincolo sulla relazione R (Città, Drone o Volo) può essere violato da un inserimento su R e da un aggiornamento su R: se tali operazioni causano queste violazioni, esse verranno rifiutate dal sistema e ciò non può essere evitato.
2. Vincolo di foreign key: il vincolo di foreign key da Volo a Città (oppure da Volo a Drone) viene violato nelle seguenti situazioni:
   a) quando si cancella una tupla t di Città (oppure di Drone) referenziata da Volo; per evitare il rifiuto di questa operazione è sufficiente definire il vincolo "on delete cascade" in modo che vengano cancellate le tuple di Volo che referenziano t;
   b) quando si aggiorna il codice di una tupla t di Città (oppure di Drone) referenziata da Volo; per evitare il rifiuto di questa operazione è sufficiente definire il vincolo "on update cascade" in modo che venga aggiornato il codice delle tuple di Volo che referenziano t;
   c) quando si inserisce in Volo una tupla t che viola il vincolo di foreign key da Volo a Città (oppure da Volo a Drone) perché il valore dell'attributo "città" della tupla t non compare nell'attributo "nome" di Città, oppure perché il valore dell'attributo "drone" della tupla t non compare nell'attributo codice di Drone; questa situazione si può gestire con un trigger che, se necessario, inserisce l'opportuna tupla in Città con il valore null nell'attributo "numAb" e, se necessario, inserisce l'opportuna tupla in Drone con il valore null nell'attributo "costo".
   d) quando si aggiorna una tupla t di Volo ed il nuovo valore assegnato all'attributo "città" (oppure "drone") non compare nell'attributo "nome" di Città (oppure il nuovo valore assegnato all'attributo "drone" non compare nell'attributo "codice" di Drone); questa situazione si può gestire con lo stesso trigger menzionato prima.

Problema 3 - soluzione

Scriviamo il codice SQL che definisce lo schema logico che risulta dalle decisioni sudddette.

```
create table Città (
   nome varchar primary key,
   numAb integer);

create table Drone (
   codice varchar primary key,
   costo integer);
```

```
create table Volo (
   drone varchar,
   città varchar,
   primary key (drone,città)
   foreign key (drone) references Drone
       on delete cascade on update cascade
   foreign key (città) references Città
       on delete cascade on update cascade);
```

```
create function aggiustaVolo() returns trigger as
$$
BEGIN
   IF NEW.drone not in (select codice from Drone)
   THEN insert into Drone values (NEW.codice,null);
   END IF;
   IF NEW.città not in (select nome from Città)
   THEN insert into Città values (NEW.nome,null);
   END IF;
   return NEW;
END;
$$ language plpgsql;
```

```
create trigger triggerInserisciVolo
before insert on Volo
    for each row execute procedure
          aggiustaVolo();

create trigger triggerAggiornaVolo
before update on Volo
    for each row execute procedure
          aggiustaVolo();
```

Problema 4 – testo e soluzione

Testo: Sia B una base di dati con la relazione Autore(scrittore,libro) sugli autori di libri e la relazione Libro(codice,genere) sui libri ed il loro genere. Sappiamo che ogni libro è scritto da almeno un autore ed è quindi soddisfatto il vincolo di inclusione da codice di Libro a libro di Autore. (3.1) Scrivere una query in SQL che calcoli per ogni libro di genere "giallo" il codice del libro ed il numero dei suoi autori. (3.2) Scrivere una query che calcoli per ogni autore il nome dell'autore ed il massimo numero di co-autori che ha avuto nella scrittura dei suoi libri.

(3.1)

```
select b.codice, count(*)
from Libro b join Autore a on b.codice = a.libro
where b.genere = "giallo"
group by b.codice
```

(3.2)

```
select a1.scrittore, max(a2.nco) - 1
from Autore a1 join (select a2.libro, count(*) nco from Autore group by libro) a2
        on a1.libro = a2.libro
group by a1.scrittore
```

Problema 5 – soluzione

(4.1) Sì, esiste una istanza dello schema *S* in cui la relazione Acquisto ha due istanze. Definiamo l'istanza I così (dove p1 è diverso da p2 e d1 è diverso da d2):

I(Banda) = { b }
I(Direttore) = { d }
I(Festa) = { f }
I(Paese) = { p1, p2 }
I(Contratto) = { }
I(Concerto) = { <Paese:p1, Banda:b, Direttore:d, Festa:f>,
<Paese:p2, Banda:b, Direttore:d, Festa:f>,
I(Data) = { <<Paese:p1, Banda:b, Direttore:d, Festa:f>, d1>,
<<Paese:p2, Banda:b, Direttore:d, Festa:f>, d2> }

Si noti che, siccome p1 è diverso da p2, le istanze di Concerto in I sono diverse, perché differiscono in almeno un ruolo (se questa proprietà non fosse rispettata, I non sarebbe una istanza dello schema S). Inoltre, si vede chiaramente che I soddisfa tutti i vincoli di S, compreso il vincolo di identificazione su Concerto, perché sebbene le due tuple di Concerto abbiano le stesse istanze nei ruoli Direttore e Banda, esse differiscono nel valore della data. Rimane da dimostrare che I ha il minimo numero possibile di istanze di entità. Osserviamo che Contratto non ha istanze ma, siccome Concerto non è vuota, nessuna entità partecipante ad Concerto può essere vuota. In I, Banda, Direttore e Festa hanno ciascuna una sola istanza, ossia il minimo possibile, mentre Paese ha due istanze. Potevamo avere meno istanze dell'entità Paese? No, perché senza queste due istanze (o due istanze in Festa) non si potrebbero formare in I due istanze della relazione Concerto. Abbiamo quindi dimostrato che esiste una istanza di S in cui la relazione Concerto ha due istanze ed abbiamo illustrato una tale istanza in cui il numero di istanze di entità è il minimo possibile.

Problema 5 – soluzione

(4.2)
Chiamiamo ora S lo schema in cui abbiamo aggiunto il vincolo di cardinalità (2,2) sul ruolo Banda di Concerto e sia I una istanze di *S* in cui Paese ha due istanze. Notiamo subito che, visto il vincolo di cardinalità minima 1 nel ruolo Paese di Concerto, in I la relazione Concerto ha almeno una istanza. Osserviamo che i vincoli di cardinalità (2,2) sui ruoli Direttore e Banda di Concerto fanno sì che in ogni istanza dello schema *S* in cui la relazione Concerto non è vuota il numero di istanze di Concerto è la metà sia del numero di istanze di Direttore sia del numero di istanze di Banda, da cui segue che il numero di istanze di Banda è uguale al numero di istanze di Direttore. Quindi abbiamo che

(α) in I, il numero di istanze di Banda è uguale al numero di istanze di Direttore.

Ma attenzione: a causa del vincolo (2,2) sul ruolo Banda di Contratto, la relazione Contratto non sarà vuota in I e quindi abbiamo che in I il numero di istanze di Banda è la metà del numero di istanze di Contratto e, poiché il vincolo di cardinalità sul ruolo Direttore di Contratto è (0,1), il numero di istanze di Direttore è maggiore o uguale al numero di istanze di Contratto, da cui segue che

(β) in I, il numero di istanze di Banda è al massimo uguale alla metà del numero di istanze di Direttore.

Poiché **(α) e (β) sono chiaramente in contraddizione**, abbiamo dimostrato che se aggiungiamo ad *S* il vincolo di cardinalità (2,2) sul ruolo Banda della relazione Contratto, non esiste alcuna istanza dello schema modificato in cui l'entità Paese ha due istanze.