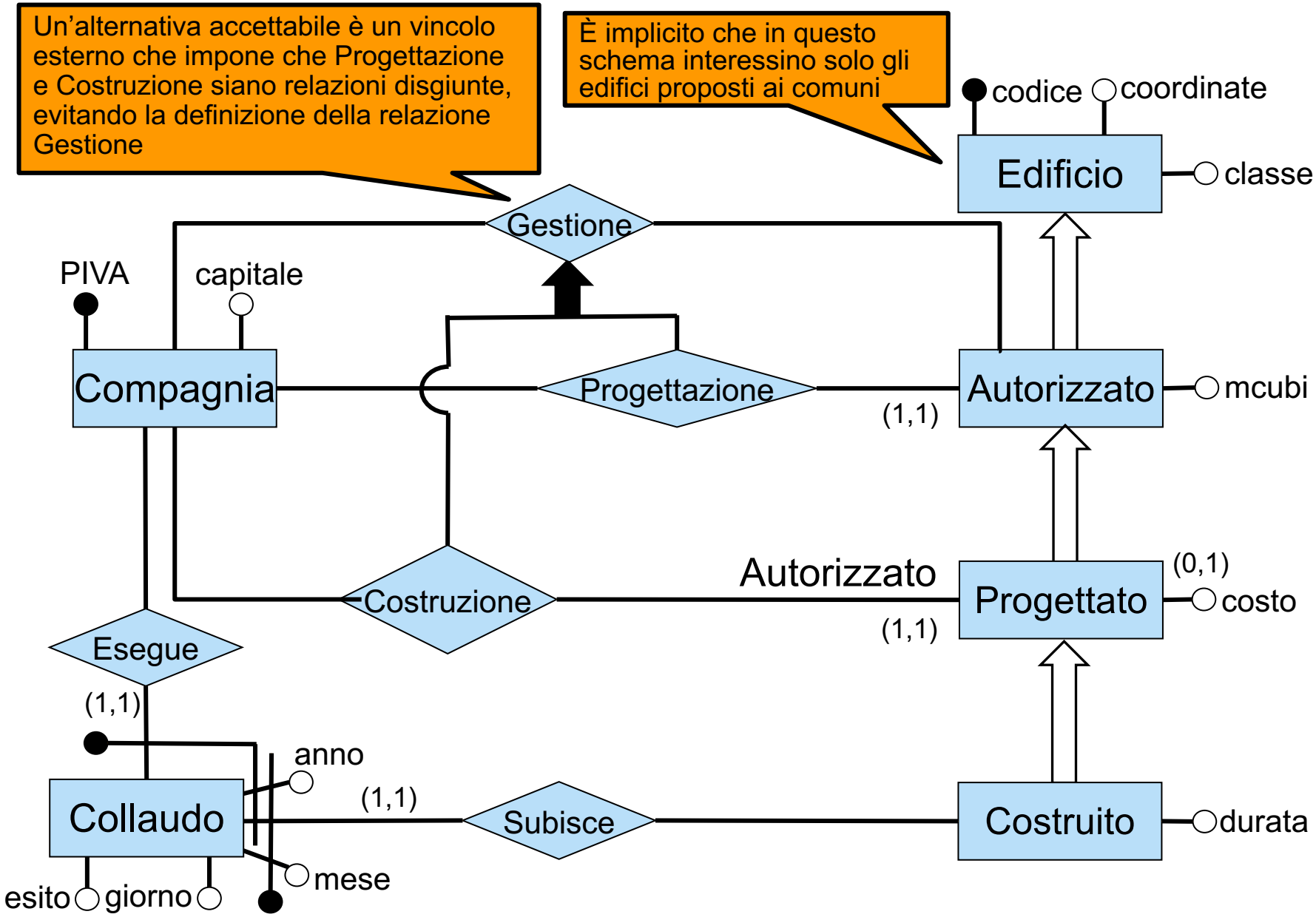
Basi di dati

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

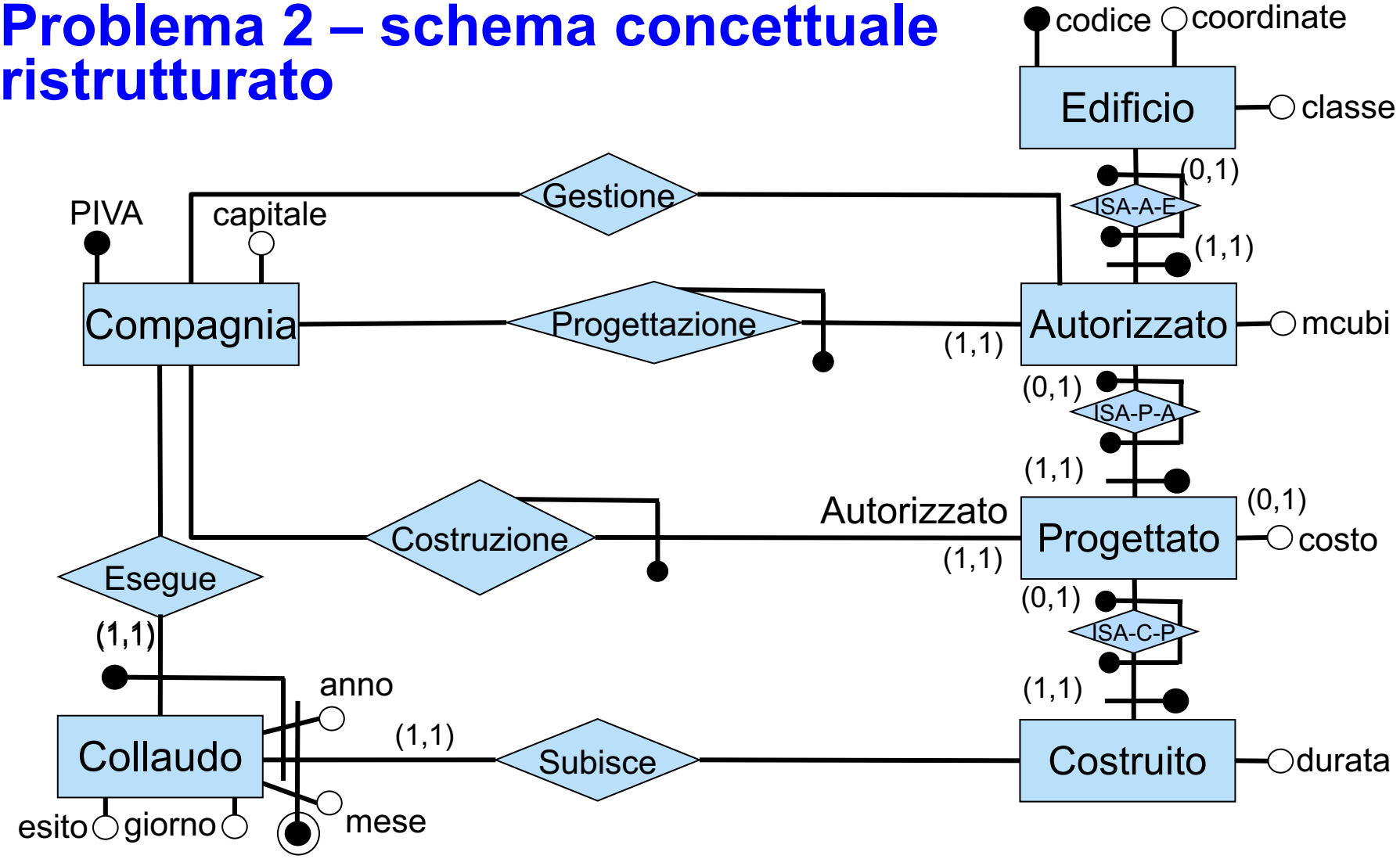
Maurizio Lenzerini

Anno Accademico 2023/24

Problema 1 – Schema concettuale



Problema 2 – schema concettuale ristrutturato



Vincolo esterno: - ogni istanza di Progettazione è anche istanza di Gestione;
- per ogni <Compagnia:c,Autorizzato:p> in Costruzione tale che <Progettato:p,Autorizzato:a> è in è ISA-P-A, si ha che <Compagnia:c,Autorizzato:a> non è in Progettazione;
- per ogni <Compagnia:c,Autorizzato:a> in Gestione ma non in Progettazione, esiste p tale che <Progettato:p,Autorizzato:a> è in ISA-P-A e <Compagnia:c,Autorizzato:p> è in Costruzione.

Problema 2 – Schema logico da traduzione diretta

Edificio(codice,coordinate,classe)
Autorizzato(codice,mcubi)
  foreign key: Autorizzato[codice] $\subseteq$ Edificio[codice]
  foreign key: Autorizzato[codice] $\subseteq$ Progettazione[Edificio]
Progettato(codice,costo*)
  foreign key: Progettato[codice] $\subseteq$ Autorizzato[codice]
  foreign key: Progettato[codice] $\subseteq$ Costruzione[Edificio]
Costruito(codice,durata)
  foreign key: Costruito[codice] $\subseteq$ Progettato[codice]
Compagnia(piva,capitale)
Gestione(autorizzato,compagnia)
  foreign key : Gestione[autorizzato] $\subseteq$ Autorizzato[codice]
  foreign key : Gestione[compagnia] $\subseteq$ Compagnia[piva]
  vincolo di gen: Gestione[edificio,compagnia] = Progettazione[edificio,compagnia] $\cup$ Costruzione[edificio,compagnia]
Progettazione(edificio,compagnia)
  foreign key: Progettazione[edificio] $\subseteq$ Autorizzato[codice]
  disgiuntezza: Progettazione[edificio,compagnia] $\cap$ Costruzione[edificio,compagnia] = $\emptyset$
Costruzione(edificio,compagnia)
  foreign key: Costruzione[edificio] $\subseteq$ Progettato[codice]
Collaudo(mese,anno,costruito,giorno,esito)
  foreign key: Collaudo[costruito] $\subseteq$ Costruito[codice]
  foreign key: Collaudo[mese,anno,costruito] $\subseteq$ Esegue[mese,anno,costruito]
Esegue(mese,anno,costruito,compagnia)
  foreign key: Esegue[mese,anno,costruito] $\subseteq$ Collaudo[mese,anno,costruito]
  foreign key: Esegue[compagnia] $\subseteq$ Compagnia[piva]

Vincolo esterno: nel join naturale tra Collaudo ed Esegue, <anno,compagnia,costruito> formano una chiave

Problema 2 – schema logico ristrutturato

L'indicazione di progetto suggerisce di accorpare Edificio ed Autorizzato debolmente accoppiati e poi di accorpare anche la relazione risultante e Costruito, a loro volta debolmente accoppiati. Inoltre, è facile vedere che possiamo eliminare la relazione inutile Gestione, ottenibile come unione delle tuple senza null di Edificio[codice,progettista] ed Edificio[codice,costruttrice].

Edificio(codice,coordinate,classe,progettista*,mcubi*,costruttrice*,costo*)
  vincolo di tupla: progettista is null se e solo se mcubi is null and
                    se progettista is null allora costruttrice is null and
                    se costruttrice is null allora costo is null
  vincolo di tupla: se costruttrice is not null allora costruttrice è diverso da progettista
  foreign key: Edificio[progettista] $\subseteq$ Compagnia[piva]
  foreign key: Edificio[costruttrice] $\subseteq$ Compagnia[piva]
Costruito(codice,durata)
  foreign key: Costruito[codice] $\subseteq$ (select codice from Edificio
                                              where costruttrice is not null)
Compagnia(piva,capitale)
Collaudo(mese,anno,costruito,esito,giorno)
  foreign key: Collaudo[costruito] $\subseteq$ Costruito[codice]
  foreign key: Collaudo[mese,anno,costruito] $\subseteq$ Esegue[mese,anno,costruito]
Esegue(mese,anno,costruito,compagnia)
  foreign key: Esegue[mese,anno,costruito] $\subseteq$ Collaudo[mese,anno,costruito]
  foreign key: Esegue[compagnia] $\subseteq$ Compagnia[piva]

Vincolo esterno: nel join naturale tra Collaudo ed Esegue, gli attributi <anno,compagnia,costruito> formano una chiave

Problema 3 – testo e soluzione

Testo: Consideriamo le relazioni Esame(studente,corso,voto) e Iscritto(studente,università), con gli ovvi significati. (3.1) Scrivere una query in SQL che calcoli quali sono gli studenti dell'università "Sapienza" che hanno una media dei voti agli esami maggiore di 25. (3.2) Scrivere una query che calcoli le università che non hanno alcun corso per il quale la media dei voti conseguiti dagli studenti di quella università negli esami di quel corso è maggiore di 28.

(3.1)

```
select e.studente
from Esame e join Iscritto t on e.studente = t.studente
where t.università = ‘‘Sapienza’’
group by e.studente
having avg(e.voto) > 25
```

(3.2)

```
select università
from Iscritto
where università not in (select t.università
                         from Esame e join Iscritto t on e.studente =t.studente
                         group by e.corso, t.università
                         having avg(e.voto) > 28)
```

Problema 4 – 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(Notaio) = { }
I(Accordo) = { }
I(Acquirente) = {a}
I(Prodotto) = {p1, p2}
I(Fornitore) = { f }
I(Acquisto) = { <Prodotto:p1, Acquirente:a, Fornitore:f>, <Prodotto:p2, Acquirente:a, Fornitore:f> }
I(Data) = { <<Prodotto:p1, Acquirente:a, Fornitore:f>, d1>,
<<Prodotto:p2, Acquirente:a, Fornitore:f>, d2> }

Si noti che, siccome p1 è diverso da p2, le istanze di Acquisto 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 Acquisto, perché sebbene le due tuple di Acquisto abbiano le stesse istanze nei ruoli Acquirente e Fornitore, esse differiscono nel valore della data. Rimane da dimostrare che I ha il minimo numero possibile di istanze di entità. Osserviamo che Notaio non ha istanze ma, siccome Acquisto non è vuota, nessuna entità partecipante ad Acquisto può essere vuota. In I, Fornitore e Acquirente hanno ciascuna una sola istanza, ossia il minimo possibile, mentre Prodotto ha due istanze. Potevamo avere meno istanze dell'entità Prodotto? No, perché senza queste due istanze non si potrebbero formare in I due istanze della relazione Acquisto. Abbiamo quindi dimostrato che esiste una istanza di S in cui la relazione Acquisto ha due istanze ed abbiamo illustrato una tale istanza in cui il numero di istanze di entità è il minimo possibile.

Problema 4 – soluzione

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

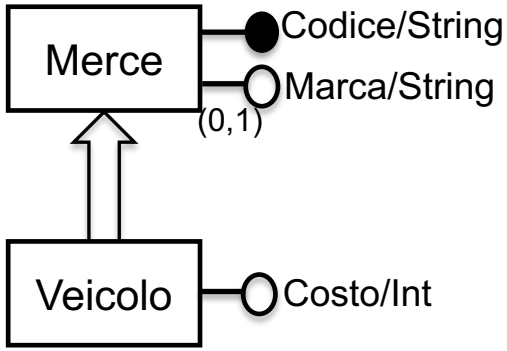
(α) in I, il numero di istanze di Fornitore è uguale al numero di istanze di Acquirente.

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

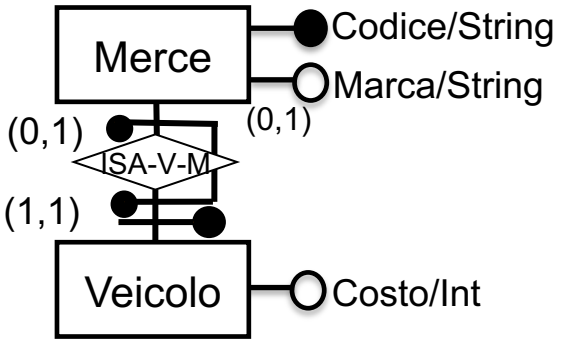
(β) in I, il numero di istanze di Fornitore è al massimo uguale alla metà del numero di istanze di Acquirente.

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

Problema 5 – soluzione



schema concettuale di partenza



schema concettuale ristrutturato

schema logico prodotto dalla traduzione diretta

Merce(<u>Codice</u>,Marca*)

Veicolo(<u>Codice</u>,Costo)

foreign key: Veicolo[codice] $\subseteq$ Merce[codice]

schema logico prodotto dalla traduzione diretta espresso in SQL

```
create table Merce (
   codice varchar primary key,
   marca varchar);

create table Veicolo (
   codice varchar primary key,
   costo integer not null,
   foreign key (codice) references Merce);
```

Problema 5 – 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 (Veicolo o Merce) 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 Veicolo a Merce viene violato nelle seguenti situazioni:

a) quando si cancella una tupla t di Merce referenziata da Veicolo; per evitare il rifiuto di questa operazione è sufficiente definire il vincolo "on delete cascade" in modo che vengano cancellate le tuple di Veicolo che referenziano t;
b) quando si aggiorna il codice di una tupla t di Merce referenziata da Veicolo; 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 Veicolo che referenziano t;
c) quando si inserisce in Veicolo una tupla t che viola il vincolo di foreign key perché il valore dell'attributo "codice" della tupla t non compare nell'attributo "codice" di Merce; questa situazione si può gestire con un trigger che inserisce l'opportuna tupla in Merce (con il valore null nell'attributo "marca")
d) quando si aggiorna una tupla t di Veicolo ed il nuovo valore assegnato all'attributo "codice" non compare nell'attributo "codice" di Merce; questa situazione si può gestire con lo stesso trigger menzionato prima, trigger che inserisce l'opportuna tupla in Merce (con il valore null nell'attributo "marca")

Problema 5 - soluzione

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

```
create table Merce (
   codice varchar primary key,
   marca varchar
);

create table Veicolo (
   codice varchar primary key,
   costo integer not null,
   foreign key (codice) references Merce on delete cascade on update cascade
);

create function inserisciMerceDaVeicolo() returns trigger as
$$
BEGIN
        IF NEW.codice not in (select codice from Merce)
        THEN insert into Merce values (NEW.codice,null);
        END IF;
        return NEW;
END;
$$ language plpgsql;

create trigger triggerInserisciVeicolo before insert on Veicolo
for each row execute procedure inserisciMerceDaVeicolo();

create trigger triggerAggiornaVeicolo before update on Veicolo
for each row execute procedure inserisciMerceDaVeicolo();
```