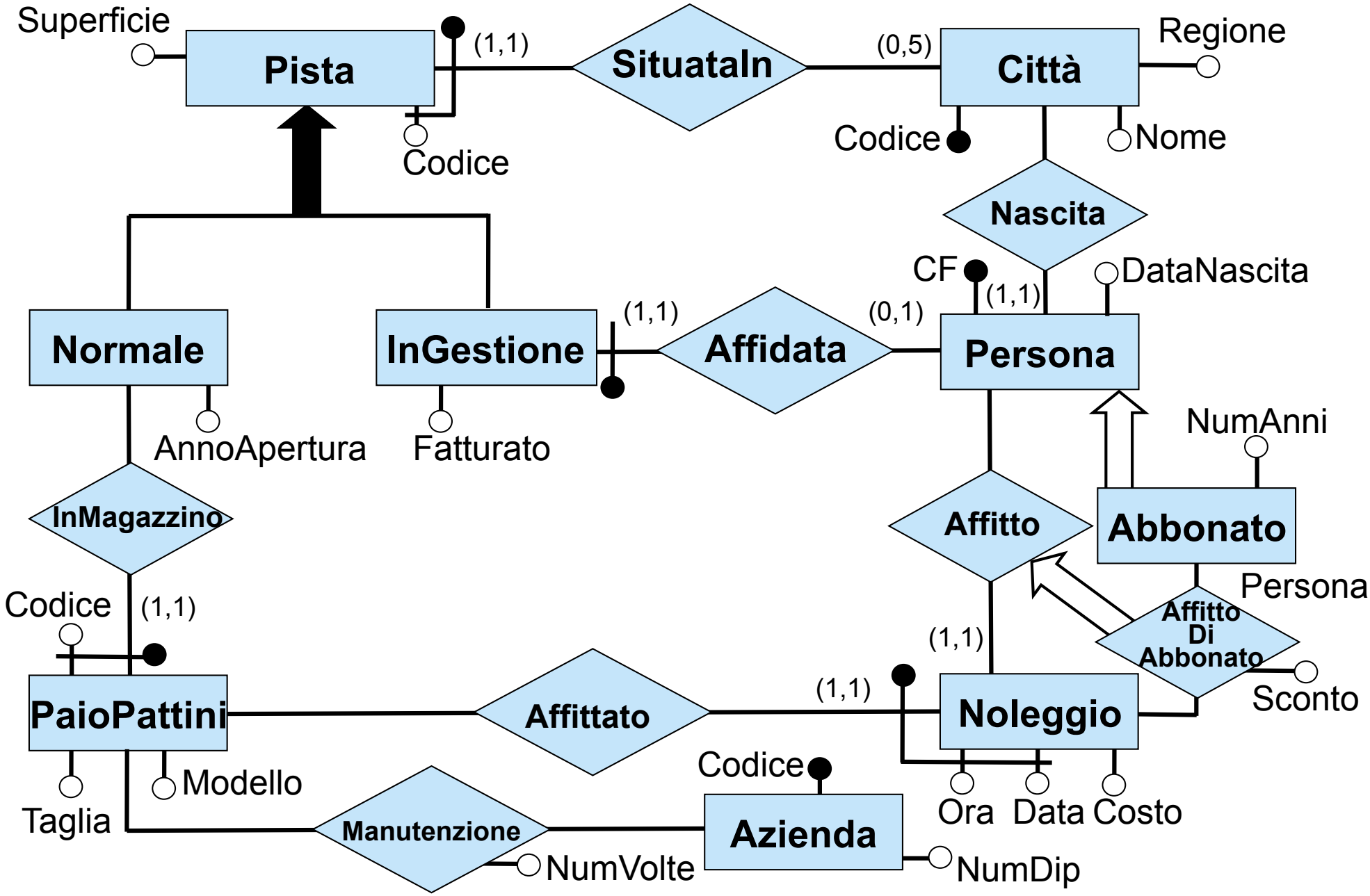
Basi di dati

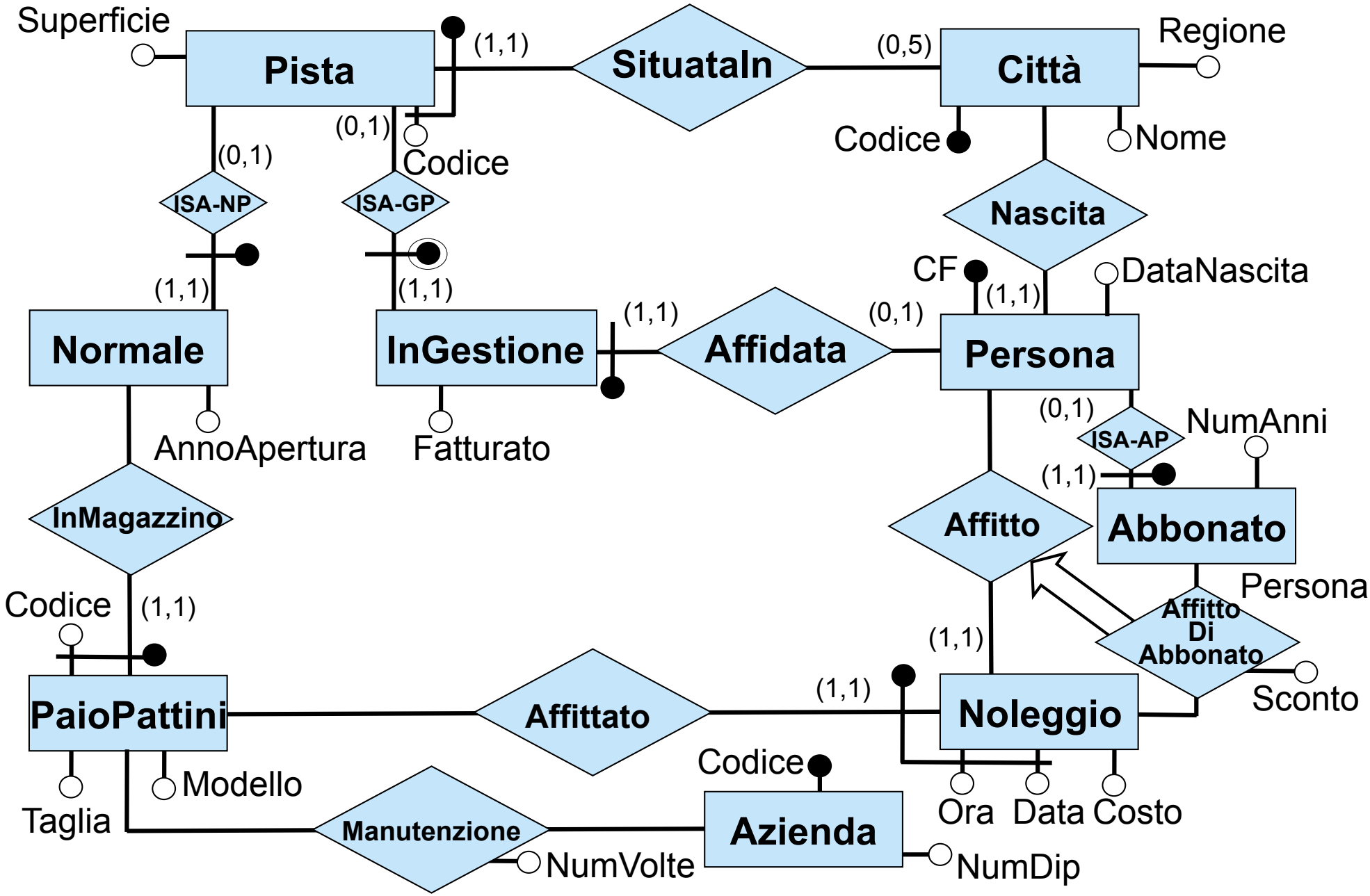
Appello del 10-01-2012
Soluzione del compito A

Anno Accademico 2011/12

Problema 1 - Schema ER



Problema 2 - Schema ER ristrutturato (1)



Problema 2 - Schema ER ristrutturato (2)

Vincoli esterni:

- (Vincolo di generalizzazione) ogni istanza di Pista partecipa o alla relazione ISA-NP o alla relazione ISA-GP, ma non a tutte e due.

- Per ogni istanza <Noleggio:n,Abbonato:a> di AffittoDiAbbonato, detta p l'istanza di Persona tale che <Abbonato:a,Persona:p> è istanza di ISA-AP, si ha che <Noleggio:n,Persona:p> è istanza di Affitto.

Problema 2 - Schema logico dalla traduzione

Città(codice, nome, regione)

Pista(codice, città, superficie)

inclusione: Pista[città] ⊆ Città[codice]

Normale(codice, città, annoApertura)

foreign key: Normale[codpista,cittàpista] ⊆ Pista[codice,città]

InGestione(codice, città, fatturato)

foreign key: InGestione[codice,città] ⊆ Pista[codice,città]

foreign key: InGestione[codice,città] ⊆ Affidata[codpista,cittàpista]

PaioPattini(codice, codpista, cittàpista, taglia, modello)

foreign key: PaioPattini[codpista,cittàpista] ⊆ Normale[codice,città]

Azienda(codice, numDip)

Manutenzione(codPattini, codpista, cittàpista, azienda, numVolte)

foreign key: Manutenzione[codPattini,codpista,cittàpista] ⊆
PaioPattini[codice,codpista,cittàpista]

foreign key: Manutenzione[azienda] ⊆ Azienda[codice]

Affidata(codpista, cittàpista, persona)

foreign key: Affidata[codpista,cittàpista] ⊆ InGestione[codice,città]

foreign key: Affidata[persona] ⊆ Persona[CF]

chiave: persona

Problema 2 - Schema logico dalla traduzione

Persona(CF, dataNascita)

foreign key: Persona[CF] ⊆ Nascita[persona]

Nascita(persona, città)

foreign key: Nascita[persona] ⊆ Persona[CF]

foreign key: Nascita[città] ⊆ Città[codice]

Noleggio(ora, data, codPattini, codpista, cittàpista, costo)

foreign key: Noleggio[codPattini,codpista,cittàpista] ⊆
PaioPattini[codice,codpista,cittàpista]

foreign key: Noleggio[ora,data,paioPattini,codpista,cittàpista] ⊆
Affitto[ora,data,paioPattini,codpista,cittàpista]

Affitto(ora, data, codPattini, codpista, cittàpista, persona)

foreign key: Affitto[ora,data,codPattini,codpista,cittàpista] ⊆
Noleggio[ora,data,codPattini,codpista,cittàpista]

foreign key: Affitto[persona] ⊆ Persona[CF]

Abbonato(CF, numAnni)

foreign key: Abbonato[CF] ⊆ Persona[CF]

AffittoDiAbbonato(ora, data, paioPattini, codpista, cittàpista, abbonato, sconto)

foreign key: AffittodiAbbonato[ora,data,paioPattini,codpista,cittàpista,abbonato] ⊆
Affitto[ora,data,paioPattini,codpista,cittàpista,persona]

foreign key: AffittodiAbbonato[abbonato] ⊆ Abbonato[CF]

Problema 2 - Schema logico dalla traduzione

Vincoli esterni:

1. Pista[codice,città] $\subseteq$ Normale[codice,città] $\cup$ InGestione[codice,città]
2. Normale[codice,città] $\cap$ InGestione[codice,città] = $\varnothing$
3. Nessun valore che compare in Città[codice] compare più di 5 volte in Pista[città]; questo vincolo si può rappresentare in SQL inserendo nella create table di Città il vincolo:

   check (5 >= select count(*) from Pista where Pista.città = codice)

Problema 2 – Ristrutturazione schema logico

Per rispondere alla indicazione che dati un paio di pattini, una data ed un'ora, si vuole spesso conoscere la persona che ha noleggiato quel paio di pattini in quella data e in quell'ora, occorre accorpare la relazione "Affitto" nella relazione "Noleggio" (e quindi trasformare la foreign key che prima era definita tra "Affitto" e "Persona" in una foreign key tra "Noleggio" e "Persona" e trasformare la foreign key che prima era definita tra "AffittoDiAbbonato" e "Affitto" in una foreign key tra "AffittoDiAbbonato" e "Noleggio"):

Noleggio(ora, data, codPattini, codista, cittàpista, costo, persona)

foreign key: Noleggio[paioPattini,codpista,cittàpista] ⊆
PaioPattini[codice,codpista,cittàpista]

foreign key: Noleggio[persona] ⊆ Persona[CF]

AffittoDiAbbonato(ora, data, paioPattini, codpista, cittàpista, abbonato, sconto)

foreign key: AffittodiAbbonato[ora,data,paioPattini,codpista,cittàpista,abbonato] ⊆
Noleggio[ora,data,paioPattini,codpista,cittàpista,persona]

foreign key: AffittodiAbbonato[abbonato] ⊆ Abbonato[CF]

Problema 2 - Schema logico finale

Città(codice, nome, regione)

Pista(codice, città, superficie)

inclusione: Pista[città] ⊆ Città[codice]

Normale(codpista, cittàpista, annoApertura)

foreign key: Normale[codpista,cittàpista] ⊆ Pista[codice,città]

InGestione(codice, città, fatturato)

foreign key: InGestione[codice,città] ⊆ Pista[codice,città]

foreign key: InGestione[codice,città] ⊆ Affidata[codpista,cittàpista]

PaioPattini(codice, codpista, cittàpista, taglia, modello)

foreign key: PaioPattini[codpista,cittàpista] ⊆ Normale[codice,città]

Azienda(codice, numDip)

Manutenzione(codPattini, codpista, cittàpista, azienda, numVolte)

foreign key: Manutenzione[codPattini,codpista,cittàpista] ⊆
PaioPattini[codice,codpista,cittàpista]

foreign key: Manutenzione[azienda] ⊆ Azienda[codice]

Affidata(codpista, cittàpista, persona)

foreign key: Affidata[codpista,cittàpista] ⊆ InGestione[codice,città]

foreign key: Affidata[persona] ⊆ Persona[CF]

chiave: persona

Problema 2 - Schema logico finale

Persona(CF, dataNascita)

foreign key: Persona[CF] ⊆ Nascita[persona]

Nascita(persona, città)

foreign key: Nascita[persona] ⊆ Persona[CF]

foreign key: Nascita[città] ⊆ Città[codice]

Noleggio(ora, data, codPattini, codpista, cittàpista, costo, persona)

foreign key: Noleggio[paioPattini,codpista,cittàpista] ⊆
PaioPattini[codice,codpista,cittàpista]

foreign key: Noleggio[persona] ⊆ Persona[CF]

Abbonato(CF, numAnni)

foreign key: Abbonato[CF] ⊆ Persona[CF]

AffittoDiAbbonato(ora, data, paioPattini, codpista, cittàpista, abbonato, sconto)

foreign key: AffittodiAbbonato[ora,data,paioPattini,codpista,cittàpista,abbonato] ⊆
Noleggio[ora,data,paioPattini,codpista,cittàpista,persona]

foreign key: AffittodiAbbonato[abbonato] ⊆ Abbonato[CF]

Vincoli esterni:

1. Pista[codice,città] ⊆ Normale[codice,città] ∪ InGestione[codice,città]
2. Normale[codice,città] ∩ InGestione[codice,città] = ∅
3. Nella create table di Città:
   check (5 >= select count(*) from Pista where Pista.città = codice)

Problema 3

Si consideri uno schema relazionale in cui la relazione

Festa(Codice, Organizzatore, Giorno, Mese, Anno, Quartiere)

memorizza, per un insieme di feste, il codice della festa, il codice fiscale dell'organizzatore, il giorno, il mese, l'anno ed il quartiere in cui si sono tenute e la relazione

Persona(CF, Quartiere)

memorizza, per ogni persona, il codice fiscale ed in quartiere in cui vive.

1. Per ogni festa del 2011 tenutasi nel quartiere dell'organizzatore, calcolare il codice della festa ed il giorno e il mese in cui si è tenuta.

***Soluzione**: È sufficiente un banale join tra "Festa" e "Persona".*

```
select Festa.Codice, Festa.Giorno, Festa.Mese
from  Festa, Persona
where Festa.Quartiere = Persona.Quartiere and
      Festa.Organizzatore = Persona.CF and
      Festa.Anno = 2011
```

Problema 3

2. Calcolare i quartieri in cui non si sono tenute feste dal 2000 in poi.

***Soluzione**: Si calcola il complemento dell'insieme dei quartieri in cui si sono tenute feste dal 2000 in poi. Come insieme dei quartieri si considera l'unione dei quartieri che compaiono nella relazione "Festa" (attributo "Quartiere") con quelli che compaiono nella relazione "Persona" (attributo "Quartiere").*

```
select Festa.Quartiere
from  Festa
where Festa. Quartiere not in
      ( select Festa.Quartiere
        from Festa
        where Anno >= 2000 )
union
select Persona. Quartiere
from  Persona
where Persona.Quartiere not in
      ( select Festa. Quartiere
        from Festa
        where Anno >= 2000 )
```

Problema 3

3. Si chiamano “nostrane” le feste organizzate in un quartiere da un organizzatore che vive in quel quartiere. Per ogni quartiere in cui si sono tenute almeno 10 feste nostrane dal 2005, calcolare quante sono state in tale quartiere le feste nostrane tenute dal 2005.

***Soluzione**: Le feste nostrane si calcolano con un equi-join tra “Festa” e “Persona” sull’attributo “Quartiere” e su Organizzatore=CF. È sufficiente selezionare le tuple delle feste nostrane che hanno “Anno” maggiore o uguale di 2005, aggregare tali tuple su “Quartiere”, ed usare la clausola “having” per filtrare solo i gruppi che hanno almeno 10 tuple (siccome “Codice” è chiave per “Festa” non serve usare la clausola distinct nel conteggio).*

```
select Festa.Quartiere, count(Festa.Codice)
from Festa, Persona
where Festa.Organizzatore = Persona.CF and
      Festa.Quartiere = Persona.Quartiere and
      Festa.Anno >= 2005
group by Festa.Quartiere
having count(Festa.Codice) >= 10
```

Problema 4

Un vincolo di integrità è una condizione che si esprime a livello di schema della base di dati e che si intende debba essere soddisfatta da tutte le istanze della base di dati.

Il vincolo di integrità

> *“dal 1990 in poi, nel quartiere “Gianicolo” non si possono tenere feste nel mese di maggio”*

è un vincolo intra-relazionale, e più in particolare un vincolo di tupla, che si esprime in SQL mediante la seguente clausola “check” dentro la “create table” relativa alla tabella “Fiera”:

```
create table Festa(...
...
check (Quartiere <> “Gianicolo” or Anno < 1990 or Mese <>
      “Maggio”)
...
)
```