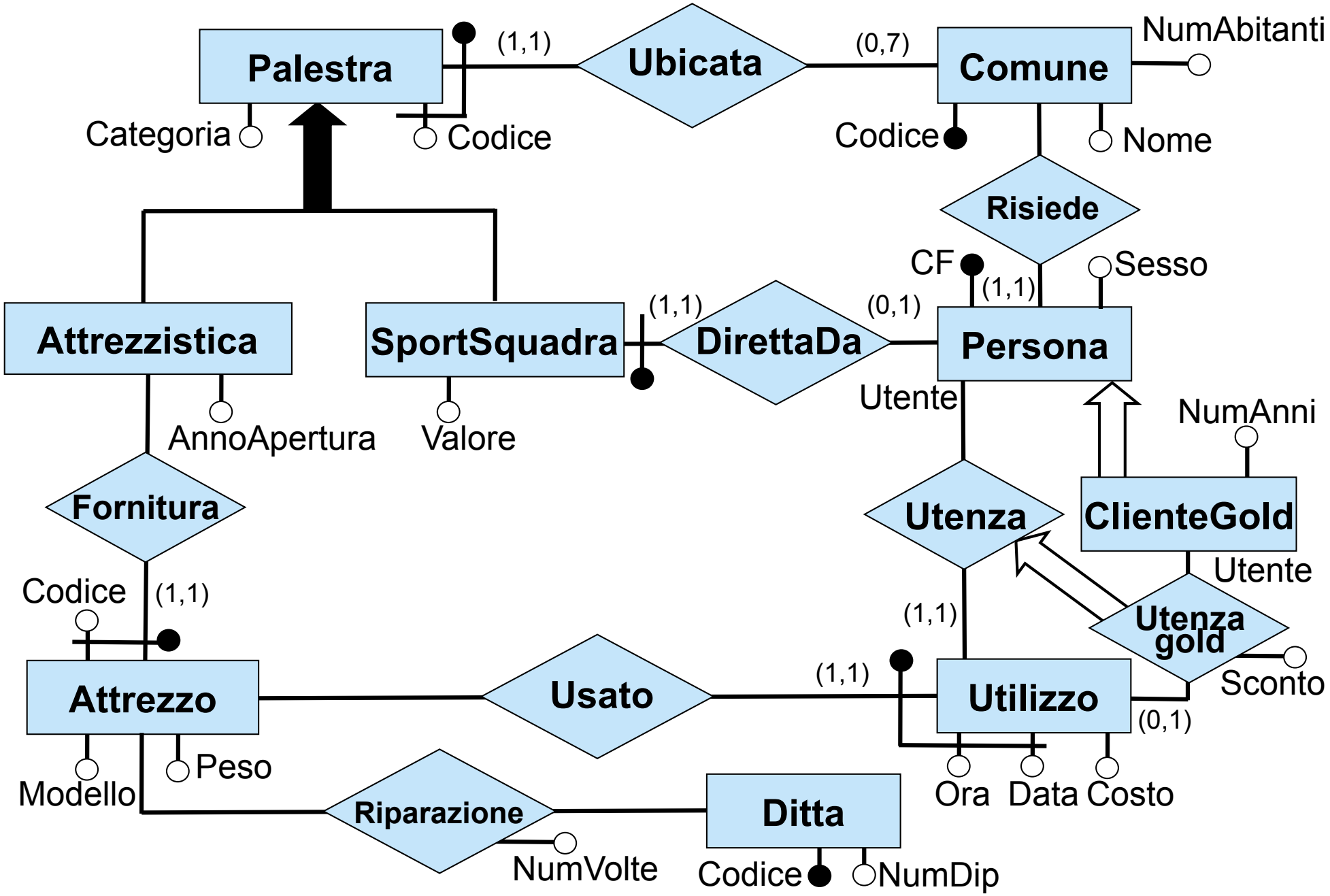
Basi di dati

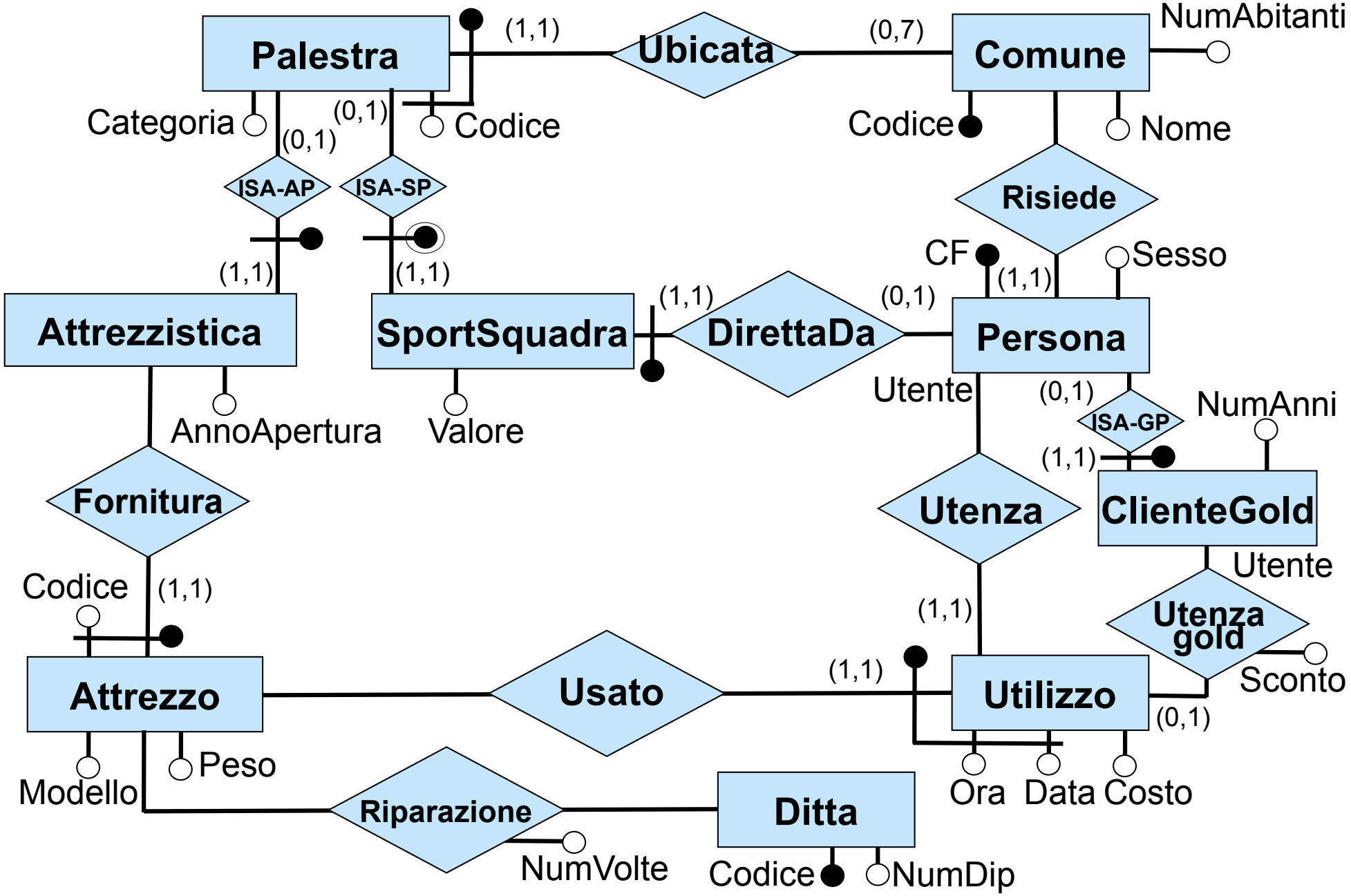
Appello del 10-01-2012
Soluzione del compito B

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 Palestra partecipa o alla relazione ISA-AP o alla relazione ISA-SP, ma non a tutte e due.

- Per ogni istanza <Utilizzo:u,ClienteGold:g> di UtenzaGold, detta p l'istanza di Persona tale che <ClienteGold:g,Persona:p> è istanza di ISA-GP, si ha che <Utilizzo:u,Persona:p> è istanza di Utenza.

Problema 2 - Schema logico dalla traduzione

Palestra(<u>codice</u>, <u>comune</u>, categoria)

foreign key: Palestra[comune] ⊆ Comune[codice]

Comune(<u>codice</u>, nome, numAbitanti)

SportSquadra(<u>codice</u>, <u>comune</u>, valore)

foreign key: SportSquadra[codice,comune] ⊆ Palestra[codice,comune]

Attrezzistica(<u>codice</u>, <u>comune</u>, annoApertura)

foreign key: Attrezzistica[codice,comune] ⊆ Palestra[codice,comune]

DirettaDa(<u>codice</u>, <u>comune</u>, persona)

foreign key: DirettaDa[codice,comune] ⊆ SportSquadra[codice,comune]

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

chiave: persona

Persona(<u>CF</u>, sesso)

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

Risiede(<u>persona</u>, comune)

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

foreign key: Risiede[comune] ⊆ Comune[codice]

Attrezzo(<u>codice</u>, <u>codPalestra</u>, <u>codComune</u>, modello, peso)

foreign key: Attrezzo[codPalestra,codComune] ⊆ Palestra[codice,comune]

Ditta(<u>codice</u>, numDipendenti)

Problema 2 - Schema logico dalla traduzione

Riparazione(codice, codPalestra, codComune, ditta, numVolte)

foreign key: Riparazione[codice,codPalestra,codComune] ⊆ Attrezzo[codice,codPalestra,codComune]

foreign key: Riparazione[ditta] ⊆ Ditta[codice]

Utilizzo(codice, codPalestra, codComune, ora, data, costo)

foreign key: Utilizzo[codice,codPalestra,codComune] ⊆ Attrezzo[codice,codPalestra,codComune]

foreign key: Utilizzo[codice,codPalestra,codComune,ora,data] ⊆ Utenza[codice,codPalestra,codComune,ora,data]

Utenza(codice, codPalestra, codComune, ora, data, persona)

foreign key: Utenza[codice,codPalestra,codComune,ora,data] ⊆ Utilizzo[codice,codPalestra,codComune,ora,data]

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

ClienteGold(CF,numAnni)

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

UtenzaGold(codice, codPalestra, codComune, ora, data, clienteGold, sconto)

foreign key: UtenzaGold[codice,codPalestra,codComune,ora,dataClienteGold] ⊆ Utenza[codice,codPalestra,codComune,ora,data,persona]

foreign key: UtenzaGold[clienteGold] ⊆ ClienteGold[CF]

Problema 2 - Schema logico dalla traduzione

Vincoli esterni:

1. Palestra[codice,comune] $\subseteq$ Attrezzistica[codice,comune] $\cup$ SportSquadra[codice,comune]
2. Attrezzistica[codice,comune] $\cap$ SportSquadra[codice,comune] = $\varnothing$
3. Nessun valore che compare in Comune[codice] compare più di 7 volte in Palestra[comune]; questo vincolo si può rappresentare in SQL inserendo nella create table di Comune il vincolo:

```
check (7 >=  select count(*)
             from Palestra
             where Palestra.comune = codice)
```

Problema 2 – Ristrutturazione schema logico

Per rispondere alla indicazione che dato un attrezzo, una data ed un'ora, si vuole spesso conoscere la persona che ha utilizzato quell'attrezzo in quella data e in quell'ora, occorre accorpare la relazione "Utenza" nella relazione "Utilizzo" (e quindi trasformare la foreign key che prima era definita tra "Utenza" e "Persona" in una foreign key tra "Utilizzo" e "Persona" e trasformare la foreign key che prima era definita tra "UtenzaGold" e "Utenza" in una foreign key tra "UtenzaGold" e "Utilizzo"):

Utilizzo(codice, codPalestra, codComune, ora, data, costo, persona)

foreign key: Utilizzo[codice,codPalestra,codComune] ⊆
Attrezzo[codice,codPalestra,codComune]

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

UtenzaGold(codice, codPalestra, codComune, ora, data, clienteGold, sconto)

foreign key: UtenzaGold[codice,codPalestra,codComune,ora,dataClienteGold] ⊆
Utilizzo[codice,codPalestra,codComune,ora,data,persona]

foreign key: UtenzaGold[clienteGold] ⊆ ClienteGold[CF]

Problema 2 - Schema logico finale

Palestra(codice, comune, categoria)

foreign key: Palestra[comune] ⊆ Comune[codice]

Comune(codice, nome, numAbitanti)

SportSquadra(codice, comune, valore)

foreign key: SportSquadra[codice,comune] ⊆ Palestra[codice,comune]

Attrezzistica(codice, comune, annoApertura)

foreign key: Attrezzistica[codice,comune] ⊆ Palestra[codice,comune]

DirettaDa(codice, comune, persona)

foreign key: DirettaDa[codice,comune] ⊆ SportSquadra[codice,comune]

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

chiave: persona

Persona(CF, sesso)

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

Risiede(persona, comune)

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

foreign key: Risiede[comune] ⊆ Comune[codice]

Attrezzo(codice, codPalestra, codComune, modello, peso)

foreign key: Attrezzo[codPalestra,codComune] ⊆ Palestra[codice,comune]

Ditta(codice, numDipendenti)

Problema 2 - Schema logico finale

Riparazione(codice, codPalestra, codComune, ditta, numVolte)

foreign key: Riparazione[codice,codPalestra,codComune] ⊆ Attrezzo[codice,codPalestra,codComune]

foreign key: Riparazione[ditta] ⊆ Ditta[codice]

Utilizzo(codice, codPalestra, codComune, ora, data, costo, persona)

foreign key: Utilizzo[codice,codPalestra,codComune] ⊆ Attrezzo[codice,codPalestra,codComune]

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

ClienteGold(CF,numAnni)

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

UtenzaGold(codice, codPalestra, codComune, ora, data, clienteGold, sconto)

foreign key: UtenzaGold[codice,codPalestra,codComune,ora,dataClienteGold] ⊆ Utilizzo[codice,codPalestra,codComune,ora,data,persona]

foreign key: UtenzaGold[clienteGold] ⊆ ClienteGold[CF]

Vincoli esterni:

1. Palestra[codice,comune] ⊆ Attrezzistica[codice,comune] ∪ SportSquadra[codice,comune]
2. Attrezzistica[codice] ∩ SportSquadra[codice] = ∅
3. Nella create table di Comune: check (7 >= select count(*) from Palestra where Palestra.comune = codice)

Problema 3

Si consideri uno schema relazionale in cui la relazione

Fiera(Codice, Mese, Anno, Città, Direttore)

memorizza, per un insieme di fiere, il codice della fiera, il mese, l'anno e la città in cui si è tenuta, e la relazione

Persona(CF, CittàResidenza)

memorizza il codice fiscale e la città di residenza di un insieme di persone.

1. Per ogni fiera del 2009 tenutasi nella città di residenza del direttore, calcolare il codice della fiera ed il mese in cui si è tenuta.

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

```
select Fiera.Codice, Fiera.Mese
from  Fiera, Persona
where Fiera.Direttore = Persona.CF and
      Fiera.Città = Persona.CittàResidenza and
      Fiera.anno= 2009
```

Problema 3

2. Calcolare le città in cui non si sono tenute fiere prima del 1998.

***Soluzione**: Si calcola il complemento dell'insieme delle città in cui si è tenuta almeno una fiera prima del 1998. Come insieme delle città si considera l'unione delle città che compaiono nella relazione "Fiera" (attributo "Città") con quelle che compaiono nella relazione "Persona" (attributo "CittàResidenza").*

```
select Fiera.Città
from  Fiera
where Fiera.Città not in
      ( select Fiera.Città
        from Fiera
        where Anno < 1998 )
union
select Persona.CittàResidenza
from  Persona
where Persona.CittàResidenza not in
      ( select Fiera.Città
        from Fiera
        where Anno < 1998 )
```

Problema 3

3. Si chiamano eterogenee le fiere tenute in un città diversa dalla città di residenza del direttore. Per ogni città in cui si sono tenute almeno 20 fiere eterogenee dal 2002, calcolare quante sono state le fiere eterogenee tenute dal 2002 in poi in tale città.

***Soluzione**: Le fiere eterogenee si calcolano con un join tra "Fiera" e "Persona" su Fiera.Città <> Persona.Città e su Fiera.Direttore=Persona.CF. È sufficiente selezionare le tuple delle fiere eterogenee che hanno "Anno" maggiore o uguale di 2002, aggregare tali tuple su "Città", ed usare la clausola "having" per filtrare solo i gruppi che hanno almeno 20 tuple (siccome "Codice" è chiave per "Fiera" non serve usare la clausola distinct nel conteggio).*

```
select Fiera.Città, count(Fiera.Codice)
From Fiera, Persona
where Fiera.Direttore = Persona.CF and
      Fiera.Città <> Persona.CittàResidenza and
      Fiera.Anno >= 2002
group by Fiera.Città
having count(Fiera.Codice) >=20
```

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 2000 in poi, nella città di Roma non si possono tenere fiere nel mese di dicembre”*

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 Fiera(...
...
check (Città <> “Roma” or Anno < 2000 or Mese <> “Dicembre”)
...
)
```