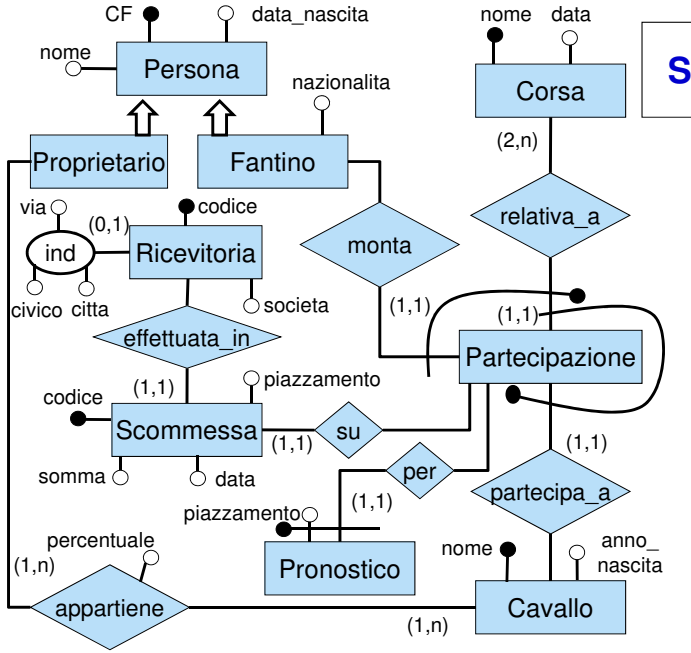
Basi di Dati ed Ingegneria del Software
modulo di Basi di Dati

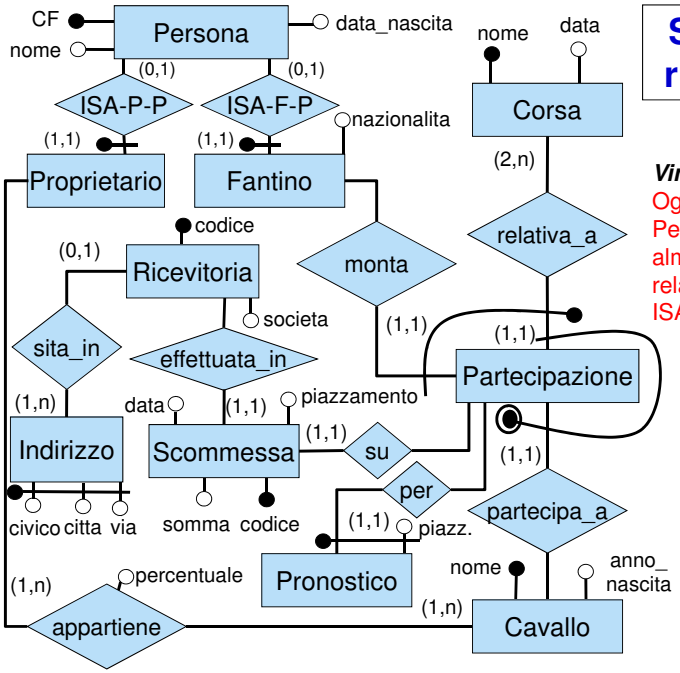
Soluzione del compito del 24/01/2013

Schema ER



Vincoli esterni: *Non esistono persone che non sono né fantini né proprietari di cavalli.*

Schema ER ristrutturato



Vincoli esterni:
Ogni istanza di Persona partecipa da almeno una fra le relazioni ISA-P-P ed ISA-F-P

Schema logico (parte 1)

Ricevitoria(codice, societa)

Sita_in(codice,via,civico,citta)
- foreign key: Sita_in[codice] ⊆ Ricevitoria[codice]
- foreign key: Sita_in[via,civico,citta] ⊆ Indirizzo[via,civico,citta]

Indirizzo(via,civico,citta)
- inclusione: Indirizzo[via,civico,citta] ⊆ Sita_in[via,civico,citta]

Scommessa(codice, data, somma, piazzamento)
- foreign key: Scommessa[codice] ⊆ Effettuata_in[scommessa]
- foreign key: Scommessa[codice] ⊆ Su[scommessa]

Effettuata_in(scommessa,ricevitoria)
- foreign key: Effettuata_in[scommessa] ⊆ Scommessa[codice]
- foreign key: Effettuata_in[ricevitoria] ⊆ Ricevitoria[codice]

Persona(CF, nome,dataNascita)

Proprietario(CF)
- foreign key: Proprietario[CF] ⊆ Persona[CF]
- inclusione: Proprietario[CF] ⊆ Appartiene[proprietario]

Schema logico (parte 2)

Appartiene(cavallo, proprietario,percentuale)
foreign key: Appartiene[cavallo] ⊆ Cavallo[nome]
foreign key: Appartiene[proprietario] ⊆ Proprietario[CF]
Corsa(nome, data)
Partecipazione(corsa, cavallo)
foreign key: Partecipazione[cavallo] ⊆ Cavallo[nome]
foreign key: Partecipazione[corsa] ⊆ Corsa[nome]
Fantino(CF, nazionalita)
foreign key: Fantino[CF] ⊆ Persona[CF]
Cavallo(nome, annoNascita)
inclusione: Cavallo[nome] ⊆ Appartiene[cavallo]
Monta(corsa,cavallo,fantino)
foreign key: Monta[corsa,cavallo] ⊆ Partecipazione[corsa,cavallo]
foreign key: Monta[fantino] ⊆ Fantino[CF]
unique: corsa, fantino

Deriva dall'identificazione esterna non principale su Partecipazione

Vincoli sullo schema logico

- inclusione: Persona[CF] ⊆ Fantino[CF] ∪ Proprietario[CF]
- Per ogni tupla t ∈ Corsa esistono almeno due tuple t1,t2 ∈ Partecipazione tali che t1[corsa]=t2[corsa]=t[nome] *(corrisponde alla cardinalità (2,n) sul ruolo R1 della ER-relazione relativa-a)*

Ristrutturazione dello schema logico

Ogni volta che si accede ad una scommessa si vuole conoscere il codice della ricevitoria presso cui e stata effettuata.

Per risparmiare un join nella precedente interrogazione, si effettua un accorpamento forte fra le relazioni Scommessa ed Effettuata_in

ScommessaNew(codice, data, somma, piazzamento, ricevitoria)
foreign key: Scommessa[codice] ⊆ Effettuata_in[scommessa]
foreign key: Scommessa[codice] ⊆ Su[scommessa]
foreign key: ScommessaNew[ricevitoria] ⊆ Ricevitoria[codice]

Oltre alla precedente ristrutturazione si può effettuare un accorpamento debole fra Sita_in e Indirizzo per eliminare una relazione ricostruibile. Il risultato è uno schema uguale al precedente in cui si elimina la relazione (ricostruibile) Indirizzo.

Problema 3

1. Calcolare il codice ed il prezzo dei prodotti consegnati a clienti di Roma dopo il 01/05/2012

```
Select  p.codProdotto, p.prezzo
From prodotto p, consegna c, cliente cl
Where p.codProdotto = c.prodotto and
      c.cliente = cl.codCliente and
      c.Data > '2012-05-01' and
      cl.citta = 'Roma'
```

Problema 3

2. Calcolare le consegne (codProdotto,codCliente,data,quantita) eettuate nella stessa data presso lo stesso cliente

```
Select c1.prodotto, c1.cliente, c1.data
From consegna c1, consegna c2
Where c1.cliente = c2.cliente and
      c1.data = c2.data and
      c1.prodotto <> c2.prodotto
```


Problema 3

3. Calcolare la spesa complessiva sostenuta da ciascun cliente per tutte le consegne a lui destinate, ma solo per clienti che abbiano avuto almeno 10 consegne (si noti che per ogni consegna la spesa e data dalla quantità di prodotto consegnato moltiplicata per il prezzo di quel prodotto);

```
Select sum(p.prezzo*c.quantita)
From consegna c, prodotto p
Where c.prodotto = p.codice
Group by c.cliente
Having count(*) > 10
```


Problema 3

4. Calcolare i prodotti (basta restituirne i codici) che non sono mai stati consegnati a clienti di Roma.

```
Select p.codProdotto
From prodotto p
Where not exists (  Select *
                    From consegna c, cliente cl
                    Where c.cliente = cl.codCliente
                          and c.prodotto = p.codProdotto
                          and cl.citta = 'Roma' )
```