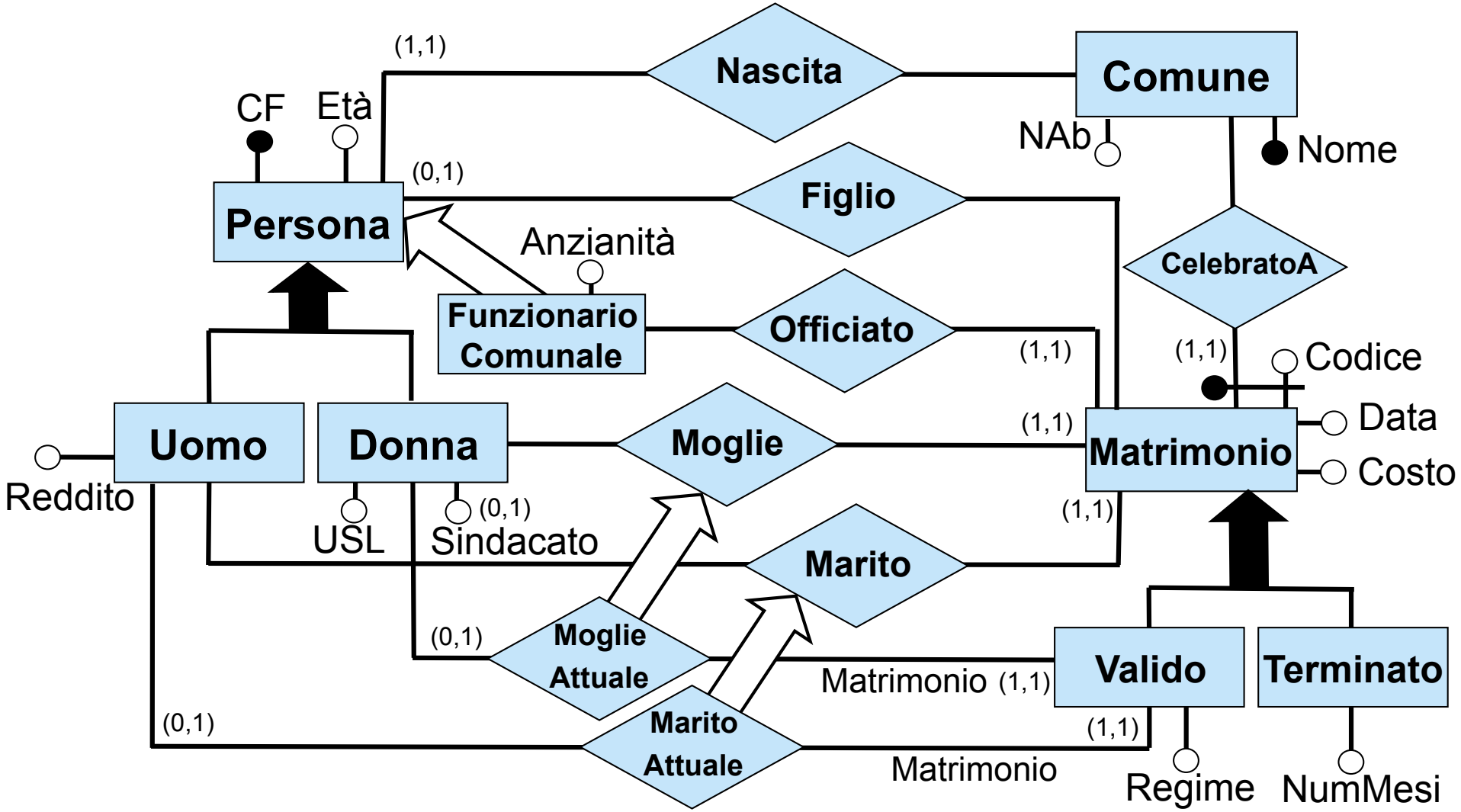
Basi di dati

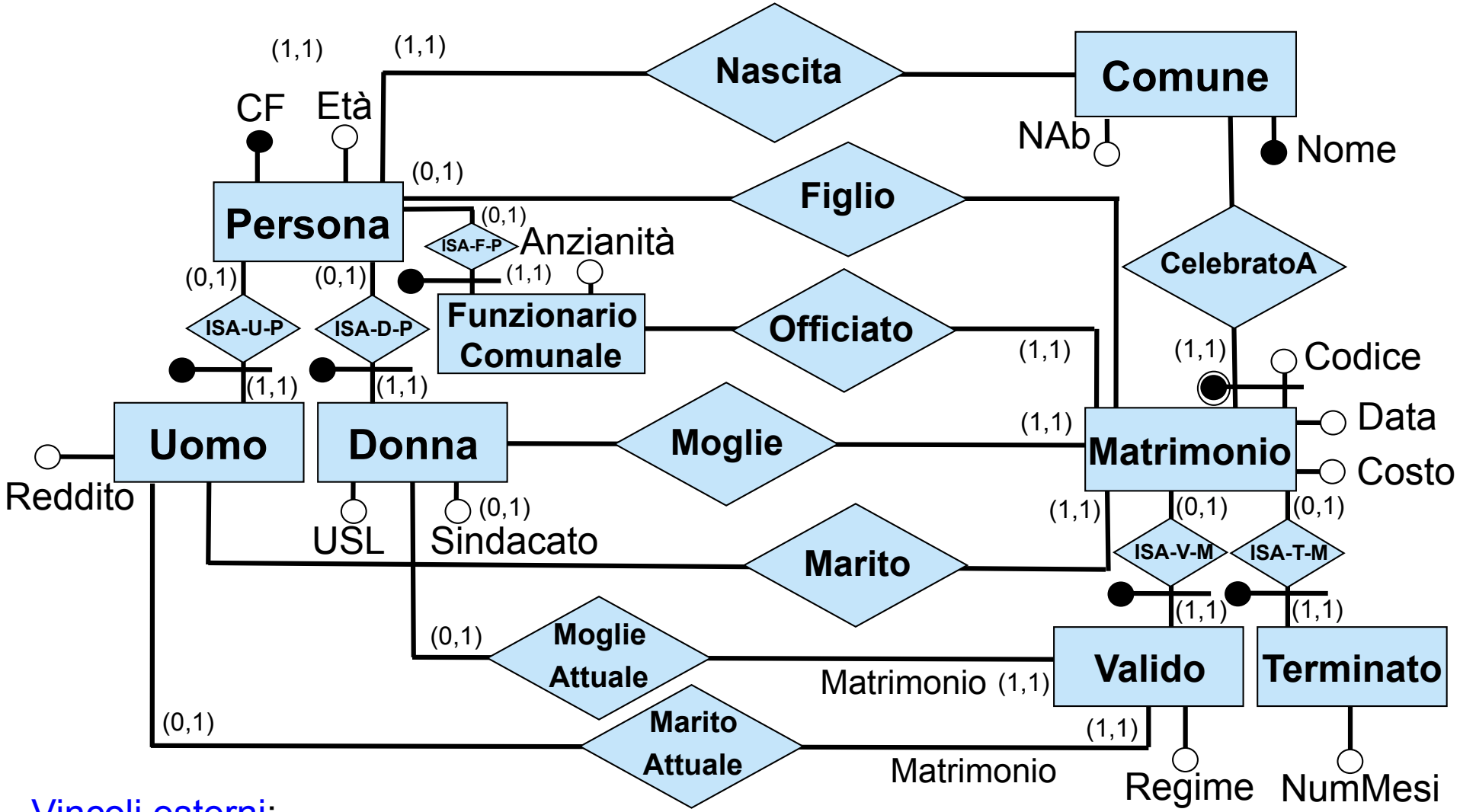
Appello del 26-02-2010

Anno Accademico 2009/10

Problema 1 - Schema ER



Problema 2 - Schema ER ristrutturato



Vincoli esterni:

- Ogni istanza di Persona partecipa ad esattamente una tra ISA-U-P e ISA-D-P
- Ogni istanza di Matrimonio partecipa ad esattamente una tra ISA-V-M e ISA-T-M
- Ogni istanza di MaritoAttuale è istanza di Marito
- Ogni istanza di MoglieAttuale è istanza di Moglie

Problema 2 - Schema logico dalla traduzione

Persona(CF, età)

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

Uomo(CF, reddito)

foreign key: Uomo[CF] ⊆ Persona[CF]

Donna(CF, usl, sindacato*)

foreign key: Donna[CF] ⊆ Persona[CF]

FunzionarioComunale(CF, anzianità)

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

Comune(nome, numab)

Matrimonio(codice, comune, data, costo)

foreign key: Matrimonio[comune] ⊆ Comune[nome]

foreign key: Matrimonio[codice,comune] ⊆ Officiato[codmat,comunemat]

foreign key: Matrimonio[codice,comune] ⊆ Marito[codmat,comunemat]

foreign key: Matrimonio[codice,comune] ⊆ Moglie[codmat,comunemat]

Valido(codice, comune, regime)

foreign key: Valido[codice,comune] ⊆ Matrimonio[codice,comune]

foreign key: Valido[codice,comune] ⊆ MaritoAttuale[codmat,comunemat]

foreign key: Valido[codice,comune] ⊆ MoglieAttuale[codmat,comunemat]

Terminato(codice, comune, nummesi)

foreign key: Terminato[codice,comune] ⊆ Matrimonio[codice,comune]

Problema 2 - Schema logico dalla traduzione

Nascita(persona, comune)

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

foreign key: Nascita[comune] ⊆ Comune[nome]

Figlio(persona,codmat,comunemat)

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

foreign key: Figlio[codmat,comunemat] ⊆ Matrimonio[codice,comune]

Officiato(funzionario, codmat, comunemat)

foreign key: Officiato[funzionario] ⊆ FunzionarioComunale[CF]

foreign key: Officiato[codmat,comunemat] ⊆ Matrimonio[codice,comune]

Marito(uomo, codmat,comunemat)

foreign key: Marito[codmat,comunemat] ⊆ Matrimonio[codice,comune]

foreign key: Marito[uomo] ⊆ Uomo[CF]

Moglie(donna,codmat,comunemat)

foreign key: Moglie[codmat,comunemat] ⊆ Matrimonio[codice,comune]

foreign key: Moglie[donna] ⊆ Donna[CF]

MaritoAttuale(uomo, codmat, comunemat)

foreign key: MaritoAttuale[uomo,codmat,comunemat] ⊆ Marito[uomo,codice,comune]

foreign key: MaritoAttuale[codmat,comunemat] ⊆ Valido[codice,comune]

chiave: uomo

Problema 2 - Schema logico dalla traduzione

MoglieAttuale(donna, codmat,comunemat)

foreign key: MoglieAttuale[donna,codmat,comunemat] ⊆ Moglie[donna,codmat,comunemat]

foreign key: MoglieAttuale[codmat,comunemat] ⊆ Valido[codice,comune]

chiave: donna

Vincoli esterni

Persona[CF] = Uomo[CF] ∪ Donna[CF]

Uomo[CF] ∩ Donna[CF] = ∅

Matrimonio[comune,codice] = Valido[codice,comune] ∪ Terminato[codice,comune]

Valido[codice,comune] ∩ Terminato[codice,comune] = ∅

Problema 2 - Schema logico ristrutturato

Per evitare valori nulli, si effettua una decomposizione mista della relazione

Donna(CF, usl, sindacato*)

sostituendola con le due relazioni:

Donna(CD,usl)

foreign key: Donna[CF] ⊆ Persona[CF]

SindacatoDonna(CF, sindacato)

foreign key: SindacatoDonna[CF] ⊆ Donna[CF]

Problema 2 - Schema logico ristrutturato

Per tenere conto del fatto che quando si accede ad un matrimonio valido si vuole sempre conoscere non solo il regime finanziario, ma anche chi sono il marito e la moglie, si esegue un accorpamento: le relazioni MaritoAttuale e MoglieAttuale spariscono, perché vengono accorpate alla relazione Valido, che quindi diventa:

Valido(codmat, comunemat, regime, moglie, marito)

- foreign key: Valido[codmat,comunemat,moglieattuale] ⊆ Moglie[codmat,comunemat,donna]
- foreign key: Valido[codmat,comunemat,maritoattuale] ⊆ Moglie[codmat,comunemat,uomo]
- chiave: moglie
- chiave: marito

Si noti che non abbiamo inserito esplicitamente il vincolo di foreign key

- foreign key: Valido[codmat, comunemat]] ⊆ Matrimonio[codice,comune]

perché è ora implicato dai due vincoli

- foreign key: Valido[codmat,comunemat,moglieattuale]] ⊆ Moglie[codmat,comunemat,donna]
- foreign key: Moglie[codmat,comunemat]] ⊆ Matrimonio[codice,comune]

Problema 3

Lo schema relazionale al quale si riferisce l'esercizio è:

Partita(sqc, sqt, goalc, goalt)

Luogo(squadra, città)

1. Mostrare i dati delle partite in cui la squadra di casa ha sede in Roma ed è risultata vincitrice.

***Soluzione**: È sufficiente un banale join tra "Partita" e "Luogo", con una selezione per tenere conto della condizione della sede in Roma e della condizione di vincitrice (che si traduce in una condizione sui goal segnati)*

```
select  Partita.sqc, Partita.sqt, Partita.goalc, Partita.goalt
from    Partita, Luogo
where   Partita.sqc = Luogo.squadra and
        Luogo.città= 'Roma' and
        goalc > goalt
```

Problema 3

2. Calcolare le città dove la Sampdoria non ha mai perso in trasferta.

***Soluzione**: Una città C farà parte del risultato se non esiste alcuna delle squadra che giocano nella città C che battuto la Sampdoria in una partita in cui la Sampdoria stessa giocava in trasferta.*

```
select Luogo.città
from   Luogo L1
where  not exists (select L2.squadra
                   from Partita, Luogo L2
                   where Partita.sqc = L2.squadra and
                         sqt = 'Sampdoria' and
                         goalc > goalt and
                         L2.città = L1.città)
```

Problema 3

3. Per ogni squadra s che ha giocato almeno 10 derby (cioè partite tra squadre della stessa città), calcolare la media dei goal segnati da s nei derby in cui s era la squadra di casa.

***Soluzione**: È sufficiente calcolare il join tra tra "Partita", "Luogo" e ancora "Luogo" in modo da selezionare solo i derby, e poi aggregare il join in gruppi basati sulla squadra di casa, calcolando la media richiesta, ed usando la clausola "having" per filtrare solo i gruppi che sono costituiti da almeno 10 elementi.*

```
select Partita.sqc, avg(Partita.goalc)
from Partita, Luogo L1, Luogo L2
Where  Partita.sqc = L1.squadra and
       Partita.sqt = L2.squadra and
       L1.città = L2.città
group by Partita.sqc
having count(*) >= 10
```

Problema 4

Proviamo a rispondere positivamente alla domanda, ovvero cerchiamo un'istanza dello schema (S2) che *non* sia istanza dello schema (S2). Se riusciremo a trovarla, avremo risposto positivamente alla domanda. Se invece non riusciremo a trovarla, avremo gli elementi per motivare la risposta negativa.

(S1) differisce da (S2) solo per il vincolo di identificazione su R. Il vincolo di identificazione definito in (S1) viene violato se esistono due istanze r1 ed r2 di R che sono collegate alle stesse istanze di A e B tramite le relazioni R1 e R2. Definiamo allora l'istanza I in modo da violare il vincolo di identificazione di (S1), ovvero mettiamo in I due istanze r1 e r2 di R legate alle stesse istanza a e b di A e B tramite le relazioni R1 ed R2. Se I è l'istanza che cerchiamo (ovvero una istanza di (S2) che non è istanza di (S1)), allora occorre fare in modo che I sia istanza di (S2), ovvero che rispetti il vincolo di identificazione di (S2). È facile verificare che, per fare in modo che I rispetti tale vincolo, è sufficiente assegnare ad r1 un valore per l'attributo V diverso rispetto al valore assegnato ad r2 per lo stesso attributo. Ne segue che la risposta alla domanda è positiva, ed una istanza di (S2) che non è istanza di (S1) è l'istanza I definita così:

```
Istanze(I,R) = { r1, r2 }
Istanze(I,A) = { a }
Istanze(I,B) = { b }
Istanze(I,R1) = { <A:a,R:r1>, <A:a,R:r2> }
Istanze(I,R2) = { <B:b,R:r1>, <B:b,R:r2> }
Istanze(I,V) = { <r1,1>, <r2,2> }
```