# Soluzione agli esercizi

Basi Di Dati

Università degli Studi di Palermo

99 pag.

# Capitolo 2

## Esercizio 2.1

Considerare le informazioni per la gestione dei prestiti di una biblioteca personale. Il proprietario presta libri ai suoi amici, che indica semplicemente attraverso i rispettivi nomi o soprannomi (così da evitare omonimie) e fa riferimento ai libri attraverso i titoli (non possiede 2 libri con lo stesso titolo). Quando presta un libro, prende nota della data prevista di restituzione. Definire uno schema di relazione per rappresentare queste informazioni, individuando opportuni domini per i vari attributi e mostrarne un'istanza in forma tabellare. Indicare la chiave (o le chiavi) della relazione.

**Soluzione:**

Queste informazioni possono essere rappresentate da una sola relazione contente i prestiti, perché non ci sono altre informazioni su amici e libri oltre ai nomi e ai titoli.
Un possibile schema è il seguente:

PRESTITO (Titolo, Nome, DataRestituzione)

Questi attributi denotano rispettivamente il titolo del libro, il nome o il soprannome dell'amico e la data di restituzione prevista del libro. La chiave è "Titolo" perché non possiede libri con lo stesso nome, quindi ogni libro è unico. Un amico invece può avere più libri e restituirli in date differenti.

Questo è un esempio in forma tabellare della relazione:

| **Titolo** | **Nome** | **DataRestituzione** |
|---|---|---|
| Il signore degli anelli | Vittorio | 12/12/2003 |
| Timeline | Danilo | 10/08/2003 |
| L'ombra dello scorpione | Angelo | 05/11/2003 |
| Piccolo mondo antico | Valerio | 15/04/2004 |

## Esercizio 2.2

Rappresentare per mezzo di una o più relazioni le informazioni contenute nell'orario delle partenze di una stazione ferroviaria: numero, orario, destinazione finale, categoria, fermate intermedie, di tutti i treni in partenza.

**Soluzione:**

Ecco un possibile schema:

PARTENZE (Numero, Orario, Destinazione, Categoria)
FERMATE (Treno, Stazione, Orario)

La relazione PARTENZE rappresenta tutte le partenze della stazione; contiene il numero di treno che è la chiave, l'orario, la destinazione finale e la categoria.
Le fermate sono rappresentate dalla seconda relazione FERMATE, perché il numero di fermate cambia per ogni treno, rendendo impossibile la rappresentazione delle fermate in PARTENZE, che

deve avere un numero fisso di attributi. La chiave di questa relazione è composta da due attributi, "Treno" e "Stazione", che indicano il numero di treno e le stazioni in cui si fermano. È necessario introdurre un vincolo di integrità referenziale tra "Treno" in FERMATE e "Numero" in PARTENZE.

## Esercizio 2.3

Definire uno schema di basi di dati per organizzare le informazioni di un'azienda che ha impiegati (ognuno con codice fiscale, cognome, nome e data di nascita), filiali (con codice, sede e direttore, che è un impiegato). Ogni lavoratore lavora presso una filiale. Indicare le chiavi e i vincoli di integrità referenziale dello schema. Mostrare un'istanza della base di dati e verificare che soddisfa i vincoli.

**Soluzione:**

Un possibile schema per questo database è:

IMPIEGATO (CodiceFiscale, Cognome, Nome, DataDiNascita, Filiale)
FILIALI (Codice, Sede, Direttore)

Le chiavi sono "CodiceFiscale" per la relazione IMPIEGATO e "Codice" per la relazione FILIALI.
I vincoli di integrità referenziale sono due:
- tra l'attributo "Filiale" di IMPIEGATO e "Codice" di FILIALI
- tra l'attributo "Direttore" di FILIALI e "CodiceFiscale" di IMPIEGATO

IMPIEGATO

| **CodiceFiscale** | **Cognome** | **Nome** | **DataDiNascita** | **Filiale** |
|---|---|---|---|---|
| SLVPTR54B578H | Salvi | Pietro | 15/04/54 | A231 |
| RSSNDR60T524S | Rossi | Andrea | 18/03/60 | A231 |
| RSSNTN58Z127B | Rossi | Antonio | 20/09/58 | A574 |
| BNCGVN52B421V | Bianchi | Giovanni | 21/11/52 | B421 |
| VRDTMM55A481L | Verdi | Tommaso | 12/01/55 | B421 |

FILIALE

| **Codice** | **Sede** | **Direttore** |
|---|---|---|
| A231 | Milano 2 | RSSNDR60T524S |
| A574 | Milano 3 | RSSNTN58Z127B |
| B421 | Roma | BNCGVN52B421V |
| C538 | Los Angeles | NULL |

Ogni valore dell'attributo "Filiale" di IMPIEGATO compare in "Codice" di FILIALE; in modo simile succede tra "Direttore" di FILIALE e "CodiceFiscale" di IMPIEGATO.

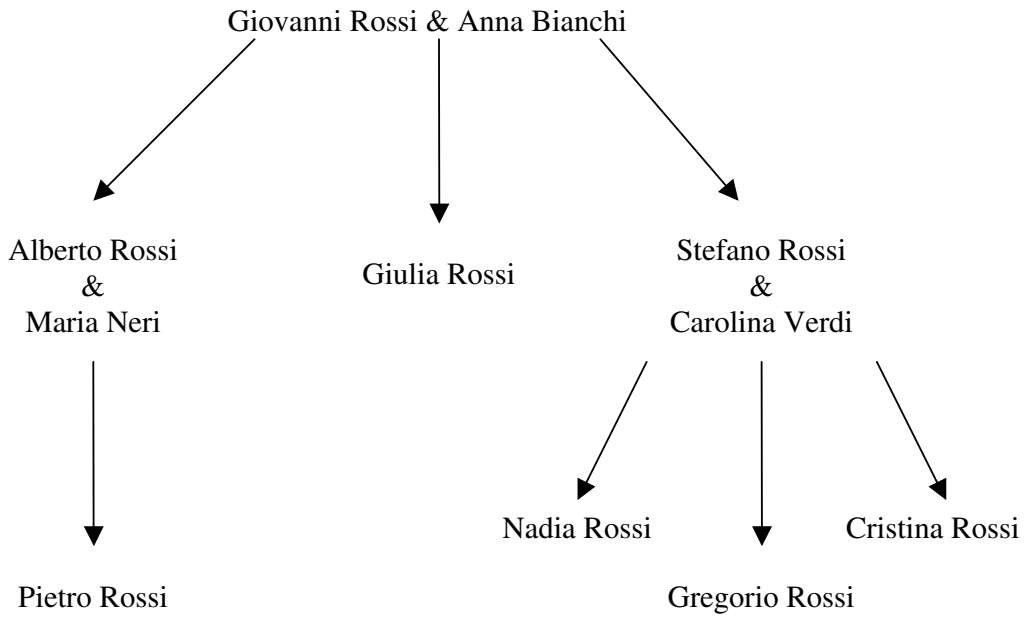
Comunque, è possibile ammettere un valore "Codice" di FILIALE che non è presente in "Filiale" di IMPIEGATO (vedi C538; nessuno lavora in quella filiale) ed un valore nullo per "Direttore" in FILIALE. Questa situazione significa per esempio che la filiale è appena stata aperta e che per il momento non ha impiegati.
Chiaramente, se "Codice" avesse un riferimento in "Filiale" il valore per il "Direttore" deve essere presente.

## Esercizio 2.4

Un albero genealogico rappresenta, in forma grafica, la struttura di una famiglia (o più famiglie, quando è ben articolato). Mostrare come si possa rappresentare, in una base di dati relazionale, un albero genealogico, cominciando eventualmente da una struttura semplificata, in cui si rappresentano solo le discendenze in linea maschile (cioè i figli vengono rappresentati solo per i componenti di sesso maschile) oppure solo quelle in linea femminile.

**Soluzione:**

Un tipico albero genealogico può essere simile a questo:




Queste informazioni possono essere rappresentate nel database:

MATRIMONIO (Marito, Moglie)
PATERNITÀ (Padre, Figlio)

Questo schema implica che ogni persona abbia un unico nome.
La famiglia vista sopra diventa:

MATRIMONIO

| **Marito** | **Moglie** |
|---|---|
| Giovanni Rossi | Anna Bianchi |
| Alberto Rossi | Maria Neri |
| Stefano Rossi | Carolina Verdi |

PATERNITÀ

| Padre | Figlio |
|---|---|
| Giovanni Rossi | Alberto Rossi |
| Giovanni Rossi | Giulia Rossi |
| Giovanni Rossi | Stefano Rossi |
| Alberto Rossi | Pietro Rossi |
| Stefano Rossi | Nadia Rossi |
| Stefano Rossi | Gregorio Rossi |
| Stefano Rossi | Cristina Rossi |

Per rappresentare anche i rappresentanti femminili è necessario aggiungere un'altra relazione per MATERNITÀ.

## Esercizio 2.5

Per ciascuno degli esercizi 2.1 – 2.4, valutare le eventuali esigenze di rappresentazioni di valori nulli, con i benefici e le difficoltà connesse.

**Soluzione:**

Nello schema dell'esercizio 2.1, i valori nulli possono essere ammessi sull'attributo DataRestituzione, perché è possibile prestare un libro senza aver fissato una precisa data di restituzione; sarebbe difficile accettare valori nulli sull'attributo "Nome", perché di solito è necessario sapere chi ha il libro. L'attributo "Libro" è la chiave e quindi non può avere valori nulli.

Nell'esercizio 2.2, oltre che alle chiavi, è difficile assegnare valori nulli a qualsiasi altro attributo, perché tutte le informazioni che contengono sono molto importanti per i viaggiatori.

Nell'esercizio 2.3, è interessante sottolineare che un valore nullo può essere ammesso nell'attributo "Direttore" nella relazione FILIALE, se il rispettivo valore "Codice" non ha riferimenti in "Filiale" nella relazione IMPIEGATO; questa situazione potrebbe significare, per esempio, che la filiale è appena stata creata e che al momento non ha impiegati. Naturalmente, se "Codice" ha un riferimento in "Filiale", il valore di "Direttore" deve essere presente.
Ovviamente possiamo immaginare valori nulli sugli attributi "Cognome", "Nome" o "DataDiNascita", ma è molto strano che queste informazioni non siano conosciute.

Nell'esercizio 2.4 si possono ammettere valori nulli sull'attributo "Moglie" in MATRIMONIO se consideriamo solo la linea maschile della famiglia.

## Esercizio 2.6

Descrivere in linguaggio naturale le informazioni organizzate nella base dati in figura 2.23.

PAZIENTI

| Cod | Cognome | Nome |
|---|---|---|
| A102 | Necchi | Luca |
| B372 | Rossini | Piero |
| B543 | Missoni | Nadia |
| B444 | Missoni | Luigi |
| S555 | Rossetti | Gino |

RICOVERI

| Paziente | Inizio | Fine | Reparto |
|---|---|---|---|
| A102 | 2/05/94 | 9/05/94 | A |
| A102 | 2/12/94 | 2/01/95 | A |
| S555 | 5/10/94 | 3/12/94 | B |
| B444 | 1/12/94 | 2/01/95 | B |
| S555 | 5/10/94 | 1/11/94 | A |

MEDICI

| Matr | Cognome | Nome | Reparto |
|---|---|---|---|
| 203 | Neri | Piero | A |
| 574 | Bisi | Mario | B |
| 431 | Bargio | Sergio | B |
| 530 | Belli | Nicola | C |
| 405 | Mizzi | Nicola | A |
| 201 | Monti | Mario | A |

REPARTI

| Cod | Nome | Primario |
|---|---|---|
| A | Chirurgia | 203 |
| B | Medicina | 574 |
| C | Pediatria | 530 |

**Soluzione:**

Questo database è per un ospedale o per una clinica.
La relazione PAZIENTI contiene le informazioni riguardanti le persone che sono state ammesse almeno una volta. Le persone sono identificate da un codice.
La relazione RICOVERI contiene tutti i ricoveri fatti nell'ospedale. Per ogni ricovero abbiamo il paziente (identificato dal codice), la data di ammissione e di dimissione e il reparto in cui il paziente è stato ricoverato.
La relazione MEDICI contiene le informazioni dei dottori che lavorano per l'ospedale e fornisce il cognome, il nome e il reparto. Il reparto è indicato da un codice. Ogni medico è identificato da un numero di matricola.
La relazione REPARTI descrive i vari reparti dell'ospedale, mostrando per ognuno di essi il nome del reparto e il primario che ne è a capo (attraverso un riferimento alla relazione MEDICI). I reparti sono identificati con un codice (A,B,C)

## Esercizio 2.7

Individuare le chiavi e i vincoli di integrità referenziale che sussistono nella base di dati di figura 2.23 e che è ragionevole assumere siano soddisfatti da tutte le basi di dati sullo stesso schema. Individuare anche gli attributi sui quali possa essere sensato ammettere valori nulli.

**Soluzione:**

Le chiavi sono:
- "Cod" per la relazione PAZIENTI
- "Paziente" e "Inizio" per la relazione RICOVERI
- "Matr" per la relazione MEDICI
- "Cod" per la relazione REPARTI

La scelta fatta sulla relazione RICOVERI presume che un paziente possa essere ricoverato solo una volta nello stesso giorno.
Se supponiamo che questa ipotesi non venga soddisfatta, e che un paziente possa essere ammesso due o più volte nello stesso giorno, la relazione non sarebbe corretta. Infatti due o più ricoveri nello stesso giorno e nello stesso reparto dovrebbero avere anche la stessa data di dimissione, e così sarebbe rappresentata nella stessa riga nella relazione.

I vincoli di integrità che esistono nel database sono tra l'attributo "Paziente" in RICOVERI e "Cod" in PAZIENTI, tra "Reparto" nella relazione RICOVERI e "Cod" nella relazione REPARTI, tra "Primario" in REPARTI e "Matr" nella relazione MEDICI e infine tra "Reparto" in MEDICI e "Cod" in REPARTI.

I valori nulli possono essere ammessi negli attributi "Cognome" e "Nome" nella relazione PAZIENTI, "Fine" nella relazione RICOVERI, "Cognome" e "Nome" nella relazione MEDICI e "Nome" nella relazione REPARTI. Tutti questi attributi non sono chiavi e non hanno nessun vincolo di integrità referenziale.

## Esercizio 2.8

Definire uno schema di basi di dati che organizzi i dati necessari a generare la pagina dei programmi radiofonici di un quotidiano, con stazioni, ore e titoli dei programmi; per ogni stazione sono memorizzati, oltre al nome, anche la frequenza di trasmissione e la sede.

**Soluzione:**

Una possibile soluzione è:

STAZIONE(Nome, Frequenza, Sede)
PROGRAMMA(Titolo, Stazione, Orario)

Questo schema presume che lo stesso titolo di un programma non possa essere utilizzato da due stazioni differenti. Se questo dovesse accadere, il campo chiave per PROGRAMMA dovrebbe essere composto da Titolo e Stazione

# Capitolo 3

## Esercizio 3.1

Considerare lo schema di base di dati contenente le relazioni:

```
Film(CodiceFilm, Titolo, Regista, Anno, CostoNoleggio)
Artisti(CodiceAttore,Cognome, Nome, Sesso, DataNascita,
        Nazionalità)
Interpretazioni(CodiceFilm,CodiceAttore, Personaggio)
```

1. Mostrare una base di dati su questo schema per la quale i join fra le varie relazioni siano tutti completi.
2. Supponendo che esistano due vincoli di integrità referenziale fra la relazione `Interpretazioni` e le altre due, discutere i possibili casi di join non completo.
3. Mostrare un prodotto cartesiano che coinvolga relazioni in questa base di dati.
4. Mostrare una base di dati per la quale uno (o più) dei join sia vuoto.

## Soluzione:

**FILM**

| CodiceFilm | Titolo | Regista | Anno | CostoNoleggio |
|---|---|---|---|---|
| 145684 | Armageddon | 15434 | 1997 | 5000000 |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 |
| 563822 | Ronin | 34573 | 1997 | 2000000 |

**ARTISTI**

| CodiceAttore | Cognome | Nome | Sesso | DataNascita | Nazionalità |
|---|---|---|---|---|---|
| 67532 | Benigni | Roberto | M | 14/03/1950 | Italiana |
| 12456 | DeNiro | Robert | M | 22/04/1951 | Americana |
| 45673 | Braschi | Nicoletta | F | 1/05/1954 | Italiana |
| 67777 | Willis | Bruce | M | 3/2/1959 | Americana |
| 12345 | Tyler | Liv | F | 18/02/1962 | Americana |

**INTERPRETAZIONI**

| CodiceFilm | CodiceAttore | Personaggio |
|---|---|---|
| 457343 | 67532 | Guido |
| 457343 | 45673 | Dora |
| 145684 | 67777 | Harry |
| 145684 | 12345 | Grace |
| 563822 | 12456 | Sam |

1. In questa instanza della base di dati, i due join (naturali) tra INTERPRETAZIONI e ARTISTI e tra INTERPRETAZIONI e FILM sono completi (in quanto non vi sono tuple che non partecipano al risultato)
2. Se inserissimo tutti i registi nella tabella ARTISTI, allora il join con INTERPRETAZIONI non sarebbe completo. Il vincolo di integrità referenziale in questo schema è posto su `CodiceAttore` e su `CodiceFilm`. Nella tabella INTERPRETAZIONI non ha senso una tupla in cui i valori `CodiceFilm` e `CodiceAttore` non abbiano corrispondenza nelle tabelle FILM e ARTISTI. È

comunque possibile ammettere tuple della tabella ARTISTI senza corrispondenza in INTERPRETAZIONI (ad esempio, nel caso dell'introduzione dei registi tra gli artisti). Questa situazione causa join incompleti.

3. Un esempio di prodotto cartesiano tra ARTISTI e FILM sulla base di dati è qui riportato:

| CodiceFilm | Titolo | Reg. | Anno | CostoNoleggio | CodiceAttore | Cognome | Nome | Sesso | DataNascita | Nazionalità |
|---|---|---|---|---|---|---|---|---|---|---|
| 145684 | Armageddon | 15434 | 1997 | 5000000 | 67532 | Benigni | Roberto | M | 14/03/50 | Italiana |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 | 67532 | Benigni | Roberto | M | 14/03/50 | Italiana |
| 563822 | Ronin | 34573 | 1997 | 2000000 | 67532 | Benigni | Roberto | M | 14/03/50 | Italiana |
| 145684 | Armageddon | 15434 | 1997 | 5000000 | 12456 | DeNiro | Robert | M | 22/04/51 | Americana |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 | 12456 | DeNiro | Robert | M | 22/04/51 | Americana |
| 563822 | Ronin | 34573 | 1997 | 2000000 | 12456 | DeNiro | Robert | M | 22/04/51 | Americana |
| 145684 | Armageddon | 15434 | 1997 | 5000000 | 45673 | Braschi | Nicoletta | F | 1/05/54 | Italiana |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 | 45673 | Braschi | Nicoletta | F | 1/05/54 | Italiana |
| 563822 | Ronin | 34573 | 1997 | 2000000 | 45673 | Braschi | Nicoletta | F | 1/05/54 | Italiana |
| 145684 | Armageddon | 15434 | 1997 | 5000000 | 67777 | Willis | Bruce | M | 3/2/59 | Americana |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 | 67777 | Willis | Bruce | M | 3/2/59 | Americana |
| 563822 | Ronin | 34573 | 1997 | 2000000 | 67777 | Willis | Bruce | M | 3/2/59 | Americana |
| 145684 | Armageddon | 15434 | 1997 | 5000000 | 12345 | Tyler | Liv | F | 18/02/62 | Americana |
| 457343 | La vita è bella | 67532 | 1998 | 1000000 | 12345 | Tyler | Liv | F | 18/02/62 | Americana |
| 563822 | Ronin | 34573 | 1997 | 2000000 | 12345 | Tyler | Liv | F | 18/02/62 | Americana |

4. Un esempio di Base di dati con join vuoti può essere il seguente: i valori nella tabella INTERPRETAZIONI non hanno corrispondenza nelle altre tabelle:

FILM

| CodiceF | Titolo | Regista | Anno | CostoNoleggio |
|---|---|---|---|---|
| 145684 | Armageddon | 15434 | 1997 | 5000000 |
| 457343 | La vita e bella | 67532 | 1998 | 1000000 |
| 563822 | Ronin | 34573 | 1997 | 2000000 |

ARTISTI

| CodiceAttore | Cognome | Nome | Sesso | DataNascita | Nazionalità |
|---|---|---|---|---|---|
| 67532 | Benigni | Roberto | M | 14/03/1950 | Italiana |
| 12456 | DeNiro | Robert | M | 22/04/1951 | Americana |
| 45673 | Braschi | Nicoletta | F | 1/05/1954 | Italiana |
| 67777 | Willis | Bruce | M | 3/2/1959 | Americana |
| 12345 | Tyler | Liv | F | 18/02/1962 | Americana |

INTERPRETAZIONI

| CodiceFilm | CodiceAttore | Personaggio |
|---|---|---|
| 478384 | 67500 | Peter |
| 467343 | 42223 | Dora |
| 185682 | 67754 | Harry |
| 945684 | 99845 | John |
| 963822 | 12000 | Mark |

# Esercizio 3.2

Con riferimento allo schema nell'esercizio 3.1, formulare in algebra relazionale, in calcolo su domini, in calcolo su tuple e in Datalog le interrogazioni che trovano:

1. i titoli dei film nei quali Henry Fonda sia stato interprete;
2. i titoli dei film per i quali il regista sia stato anche interprete;
3. i titoli dei film in cui gli attori noti siano tutti dello stesso sesso.

## Soluzione:

1.

Algebra Relazionale:

$\Pi_{Titolo}$( FILM $\triangleright\triangleleft$ ($\sigma_{(Nome="Henry") \wedge (Cognome="Fonda")}$ (ARTISTI) $\triangleright\triangleleft$ INTERPRETAZIONI) )

Calcolo dei Domini:

{ Titolo: t | FILM (CodiceFilm: fn, Titolo: t,Regista: d, Anno: y,CostoNoleggio: pc) $\wedge$
ARTISTI (CodiceAttore: an, Cognome:cogn , Nome: n, Sesso: s,
DataNascita:b, Nazionalità:naz) $\wedge$
INTERPRETAZIONI (CodiceFilm: fn, CodiceAttore: an, Personaggio: ch ) $\wedge$
(cogn= "Fonda") $\wedge$ (n= "Henry") }

Calcolo delle Tuple:

{ F.titolo | F(FILM), A(ARTISTI), I(INTERPRETAZIONI) |
F.CodiceFilm = I.CodiceFilm $\wedge$ A.CodiceAttore =I. CodiceAttore $\wedge$
A.Cognome= "Fonda" $\wedge$ A.Nome= "Henry" }

Datalog:

FILMCONFONDA (Titolo: t) $\leftarrow$
FILM (CodiceFilm: fn, Titolo: t, Regista: d, Anno: y, CostoNoleggio: pc),
ARTISTI ( CodiceAttore: an, Cognome: "Fonda", Nome: "Henry", Sesso: s,DataNascita: b,
Nazionalità : naz),
INTERPRETAZIONI ( CodiceFilm : fn, CodiceAttore : an, Personaggio: ch )

2.

Algebra Relazionale:

$\Pi_{Titolo}$ ($\sigma_{(Regista=CodiceAttore)}$(INTERPRETAZIONI$\triangleright\triangleleft$ FILM))

Calcolo dei Domini:

{ Titolo: t | FILM ( CodiceFilm : fn, Titolo: t, Regista: d, Anno: y, CostoNoleggio: pc) $\wedge$
INTERPRETAZIONI( CodiceFilm : fn, CodiceAttore:d, Personaggio: ch ) }

Calcolo delle tuple:

{ F.Titolo | F(FILM), I(INTERPRETAZIONI) |
F. CodiceFilm =I. CodiceFilm $\wedge$ F.Regista=I.CodiceAttore}

Datalog:

FILMCONREGISTAARTISTA( Titolo: t) ←
FILM (CodiceFilm:fn, Titolo:t,Regista:d, Anno: y,CostoNoleggio: pc)
INTERPRETAZIONI ( CodiceFilm : fn, CodiceAttore: d, Personaggio: ch )

3.

Algebra Relazionale:

$\Pi_{Titolo}$ (FILM) -
$\Pi_{Titolo}$ (FILM) $\triangleright\triangleleft \sigma_{Sex<>Sex1}$( (ARTISTI $\triangleright\triangleleft$ INTERPRETAZIONI) $\triangleright\triangleleft$
$\rho_{Sex1\leftarrow Sex}(\Pi_{CodiceFilm,Sex}$ (ARTISTI $\triangleright\triangleleft$ INTERPRETAZIONI)) )

Calcolo dei Domini:

{ Titolo: t | FILM (CodiceFilm: fn, Titolo:t, Regista:d, Anno: y, CostoNoleggio: pc) ∧
¬∃t1(∃d1(∃y1(∃pd1(FILM( CodiceFilm: fn,Titolo:t1,Regista:d1, Anno: y1, CostoNoleggio:pd1 ) ∧
ARTISTI (CodiceAttore: an1,Cognome: sur1,Nome: n1, Sesso: s1,
DataNascita: b1, Nazionalità :nat1) ∧
ARTISTI (CodiceAttore: an2,Cognome: sur2,Nome: n2, Sesso: s2,
DataNascita: b2, Nazionalità :nat2) ∧
INTERPRETAZIONI (CodiceFilm: fn, CodiceAttore: an1, Personaggio: ch1)
∧
INTERPRETAZIONI (CodiceFilm: fn, CodiceAttore: an2, Personaggio: ch2)
∧
(s1 ≠ s2) )))) }

Calcolo delle Tuple:

{ F.Titolo | F(FILM) |
¬(∃F1(FILM)(∃A1(ARTISTI) (∃A2(ARTISTI) (∃I1(INTERPRETAZIONI)
(∃I2( INTERPRETAZIONI ) ∧
A1.CodiceAttore=I1.CodiceAttore∧ F1.CodiceFilm=I1.CodiceFilm ∧
A2.CodiceAttore=I2. CodiceAttore ∧ I1.CodiceFilm =I2.CodiceFilm ∧
A1.Sesso≠A2.Sesso ))))) }

Datalog:

SESSODIVERSO (CodiceFilm: fn) ←
FILM (CodiceFilm: fn, Titolo: t,Regista: d, Anno: y, CostoNoleggio: pc),
ARTISTI (CodiceAttore: an1, Cognome:sur1 , Nome: n1, Sesso: s1, DataNascita:b1,
Nazionalità :nat1),
ARTISTI (CodiceAttore: an2, Cognome:sur2 , Nome: n2, Sesso: s2, DataNascita:b2,
Nazionalità :nat2),
INTERPRETAZIONI (FilmNumber: fn, CodiceAttore: an1, Personaggio: ch1),
INTERPRETAZIONI (FilmNumber: fn, CodiceAttore: an2, Personaggio: ch2),
(s1 ≠ s2)

STESSOSESSO(Titolo: t) ←
FILM (CodiceFilm: fn, Titolo: t,Regista: d, Anno: y, CostoNoleggio: pc),
NOT SESSODIVERSO (CodiceFilm: fn)

# Esercizio 3.3

Si consideri lo schema di base di dati che contiene le seguenti relazioni:

DEPUTATI (Codice, Cognome, Nome, Commissione, Provincia, Collegio)
COLLEGI (Provincia, Numero, Nome)
PROVINCE (Sigla, Nome, Regione)
REGIONI (Codice, Nome)
COMMISSIONI (Numero, Nome, Presidente)

Formulare in algebra relazionale, in calcolo dei domini e in calcolo delle tuple le seguenti interrogazioni:

1. Trovare nome e cognome dei presidenti di commissioni cui partecipa almeno un deputato eletto in una provincia della Sicilia;
2. Trovare nome e cognome dei deputati della commissione Bilancio;
3. Trovare nome, cognome e provincia di elezione dei deputati della commissione Bilancio;
4. Trovare nome, cognome, provincia e regione di elezione dei deputati della commissione Bilancio;
5. Trovare le regioni in cui vi sia un solo collegio, indicando nome e cognome del deputato ivi eletto;
6. Trovare i collegi di una stessa regione in cui siano stati eletti deputati con lo stesso nome proprio.

## Soluzione:

1.

Algebra Relazionale:

$$\Pi_{Nom,\ Cogn}((\rho_{Nom,Cogn \leftarrow Nome,Cognome}(DEPUTATI)) \rhd\lhd_{Presidente=Codice} (COMMISSIONI \rhd\lhd_{Numero=Comm} (\rho_{Comm \leftarrow Commissione}(DEPUTATI \rhd\lhd_{Provincia=Sigla} (PROVINCE \rhd\lhd_{Regione=Codice} (\sigma_{Nome='Sicilia'}(REGIONI)))))))$$

Calcolo dei Domini:

$$\{ Cognome: surp, Nome: fnp \mid DEPUTATI(Codice: np, Cognome: surp, Nome: fnp, Commissione: cmp, Provincia: sigl, Collegio: cstp) \wedge COMMISSIONI(Numero: cm, Nome: nameC, Presidente: np) \wedge COLLEGI(Provincia: usl22, Numero: usl23, Nome: usl32) \wedge PROVINCE(Sigla: sigl, Nome: usl1, Regione: co) \wedge REGIONI(Codice:co, Nome: RegName) \wedge (RegName = "Sicilia") \}$$

Calcolo delle Tuple:

$$\{ PRES.(Cognome, Nome) \mid PRES(DEPUTATI), D(Deputati), COM(COMMISSIONI), COL(COLLEGI), P(PROVINCE), R(REGIONI) \mid (PRES.Codice=COM.Presidente) \wedge (COM.Numero=D.Numero) \wedge (D.Provincia=P.Sigla) \wedge (P.Regione=R.Codice) \wedge (R.Nome = "Sicilia") \}$$

2.

Algebra Relazionale:

$$\Pi_{NomeC,Cognome}\ ((\rho_{NomeC \leftarrow Nome}(\ DEPUTATI)) \triangleright\triangleleft_{Commissione=Numero} (\sigma_{Nome="Bilancio"}(\ COMMISSIONE)))$$

Calcolo dei Domini:

{Nome: fn, Cognome: sur |
DEPUTATI(Codice: n, Cognome: sur, Nome: fn,
Commissione: cm, Provincia:co,Collegio: cst) ∧
COMMISSIONE(Numero: cm, Nome: nomeC, Presidente: np) ∧
(nomeC= "Bilancio") }

Calcolo delle tuple:

{D.(Nome,Cognome) | D(DEPUTATI), C(COMMISSIONI) |
(D.Commissione=C.Numero) ∧ (C.Nome= "Bilancio") }

3.

Algebra Relazionale:

$$\Pi_{NomeC,Cognome,nom1}\ (\ (\rho_{Nom1 \leftarrow Nome}(\ PROVINCIA)) \triangleright\triangleleft_{Sigla=Provincia} ((\rho_{Nome1 \leftarrow Nome}(DEPUTATI)) \triangleright\triangleleft_{Commissione=Numero} (\sigma_{Nome="Bilancio"}(\ COMMISSIONE)))$$

Calcolo dei Domini:

{Nome: fn, Cognome: sur, Provincia: prv |
DEPUTATI(Codice: n, Cognome: sur, Nome: fn,
Commissione: cm, Provincia:co,Collegio: cst) ∧
PROVINCE(Sigla: usl1 ,Nome: prv ,Regione: co)∧
COMMISSIONE(Numero: cm, Nome: nameC, Presidente: usl2) ∧
(nameC= "Bilancio") }

Calcolo delle tuple:

{D.(Nome,Cognome), P.(Nome), | D(DEPUTATI),
C(COMMISSIONI), P(PROVINCE) |
(D.Commissione=C.Numero) ∧ (C.Nome= "Bilancio") ∧
(D.Provincia= P.Sigla) }

4.

Algebra Relazionale:

$$\Pi_{NomeC,Cognome,nom1,rege}\ (\ (\rho_{Rege \leftarrow Nome}(\ REGIONE)) \triangleright\triangleleft_{Codice=Regione} (\rho_{Nom1 \leftarrow Nome}(\ PROVINCIA)) \triangleright\triangleleft_{Sigla=Provincia} ((\rho_{Nome1 \leftarrow Nome}(DEPUTATI)) \triangleright\triangleleft_{Commissione=Numero} (\sigma_{Nome="Bilancio"}(\ COMMISSIONE)))$$

Calcolo dei Domini:

{Nome: fn, Cognome: sur, Provincia: prv, Regione:rege |
DEPUTATI(Codice: n, Cognome: sur, Nome: fn,
Commissione: cm, Provincia:co,Collegio: cst) ∧
PROVINCE(Sigla: usl1 ,Nome: prv ,Regione: co)∧
REGIONI(Codice:co, Nome: rege )∧
COMMISSIONE(Numero: cm, Nome: nameC, Presidente: usl2) ∧
(nameC= "Bilancio") }

Calcolo delle tuple:

{D.(Nome,Cognome), P.(Nome), R.(Nome) | D(DEPUTATI),
C(COMMISSIONI), R(REGIONI), P(PROVINCE) |
(D.Commissione=C.Numero) ∧ (C.Nome= "Bilancio") ∧
(D.Provincia= P.Sigla)∧ (P.Regione= R.Codice) }

5. Algebra Relazionale:

$\Pi_{\text{RegioneC,Nome, Cognome}}$
(DEPUTATI$\triangleright\triangleleft_{\text{Provincia=ProvinciaC} \wedge \text{ Collegio=NumeroC}}$
(REGIONI$\triangleright\triangleleft_{\text{Codice=RegioneC}}$
(($\rho_{\text{SiglaC,NomeC,RegioneC,NumeroC, Nome1C}\leftarrow\text{Sigla,Nome,Regione,Numero, Nome1}}$
(PROVINCE$\triangleright\triangleleft_{\text{Sigla=Provincia}}(\rho_{\text{Nome1}\leftarrow\text{Nome}}$COLLEGI))))
-
$\Pi_{\text{SiglaC,NomeC,RegioneC,NumeroC, Nome1C}}$
((PROVINCE$\triangleright\triangleleft_{\text{Sigla=Provincia}}(\rho_{\text{Nome1}\leftarrow\text{Nome}}$COLLEGI))
$\triangleright\triangleleft_{\text{Regione=RegioneC}}$
($\rho_{\text{SiglaC,NomeC,RegioneC,NumeroC, Nome1C}\leftarrow\text{Sigla,Nome,Regione,Numero, Nome1}}$
(PROVINCE$\triangleright\triangleleft_{\text{Sigla=Provincia}}(\rho_{\text{Nome1}\leftarrow\text{Nome}}$COLLEGI))))))

Calcolo dei Domini:

{Nome: fn, Cognome: sur, Regione:rege |
DEPUTATI(Codice: n, Cognome: sur, Nome: fn,
Commissione: cm, Provincia:prov,Collegio: coll) ∧
PROVINCE(Sigla: prov,Nome: prv ,Regione: co)∧
REGIONI(Codice:co, Nome: rege )∧
PROVINCE(Sigla: n1 ,Nome: nom13 ,Regione: co)∧
COLLEGI(Provincia: n1 ,Numero: coll,Nome: usl31)∧
¬( ∃ n1(∃ co PROVINCE(Sigla: n1 ,Nome: nom1 ,Regione: co)∧
COLLEGI(Provincia: n1 ,Numero: usl11 ,Nome: usl21)∧
PROVINCE(Sigla: n21 ,Nome: nom2 ,Regione: co)∧
COLLEGI(Provincia: n2 ,Numero: usl12 ,Nome: usl22) ∧ usl11 <> usl12}

Calcolo delle tuple:

{D.(Nome,Cognome),R.(Nome) | D(DEPUTATI),
C(COLLEGI), R(REGIONI), C1(COLLEGI),
P1(PROVINCE), C2(COLLEGI), P2(PROVINCE) |
(D.Collegio=C.Numero)
¬(∃ P1(PROVINCE) ((P2.Sigla=C2.Provincia) ∧ (P1.Sigla=C1.Provincia) ∧
(P1.Regione=P2.Regione) ∧ (C1.nome ≠ C2.nome) ) }

6. Algebra Relazionale:

$(\Pi_{NomeColl1}$
$(\sigma_{NomeD1=NomeD2}$
$(\rho_{NomeD2,NomeColl2,Regione2\leftarrow,NomeD,Nome,Regione}$
$(\Pi_{Regione,NomeD,Provincia,CollegioNome}(\rho_{NomeD,ProvinciaD\leftarrow Nome,Provincia}$ DEPUTATI )
$\rhd\lhd_{(Provincia=ProvinciaD \wedge Collegio=Numero)}$
(COLLEGI$\rhd\lhd_{Provincia=Sigla}(\rho_{NomeP\leftarrow Nome}$ PROVINCE)))))
$\rhd\lhd_{Regione2=Regione1}$
$(\rho_{NomeD1,NomeColl1,Regione1\leftarrow,NomeD,Nome,Regione}$
$(\Pi_{Regione,NomeD,Provincia,CollegioNome}(\rho_{NomeD,ProvinciaD\leftarrow Nome,Provincia}$ DEPUTATI )
$\rhd\lhd_{(Provincia=ProvinciaD \wedge Collegio=Numero)}$
(COLLEGI$\rhd\lhd_{Provincia=Sigla}(\rho_{NomeP\leftarrow Nome}$ PROVINCE))))))))

Calcolo dei Domini:

{Nome: nomecoll |
DEPUTATI(Codice: n1, Cognome: sur1, Nome: fn,
Commissione: cm1, Provincia:n1,Collegio: coll1) ∧
PROVINCE(Sigla: n1,Nome: prv1 ,Regione: rege)∧
COLLEGI(Provincia: n1 ,Numero: coll1 ,Nome: nomecoll1)∧
DEPUTATI(Codice: n1, Cognome: sur2, Nome: fn,
Commissione: cm2, Provincia:n2,Collegio: coll2) ∧
PROVINCE(Sigla: n2,Nome: prv2 ,Regione: rege)∧
COLLEGI(Provincia: n2 ,Numero: coll2,Nome: Nomecoll2)}

Calcolo delle tuple:

{C1.(Nome) | D1(DEPUTATI),P1(PROVINCE), C1(COLLEGI),
D2(DEPUTATI),P2(PROVINCE), C2(COLLEGI) |
(C1.Provincia=P1.Sigla) ∧ (C2.Provincia=P2.Sigla)∧
(D1.Provincia= C1.Provincia)∧(D1.Collegio= C1.Numero)∧
(D2.Provincia= C2.Provincia)∧(D2.Collegio= C2.Numero)∧
(P2.Regione=P1.Regione) ∧ (D1.Nome=D2.Nome) }

# Esercizio 3.4

Mostrare come le interrogazioni nell'esercizio 3.3 possano trarre vantaggio, nella specifica, dalla definizione di viste.

## Soluzione:

I punti 2, 3 e 4 dell'esercizio 3.3 sono molto simili. Per rendere più veloci le interrogazioni è utile definire una vista con le parti in comune, inserendo in una tabella COMBILANCIO tutte le informazioni necessarie:

$$\begin{aligned}&\text{COMBILANCIO} = (\\&(\rho_{\text{Rege}\leftarrow\text{Nome}}(\text{ REGIONE}))\rhd\lhd_{\text{Codice=Regione}}\\&(\rho_{\text{Nom1}\leftarrow\text{Nome}}(\text{ PROVINCIA}))\rhd\lhd_{\text{Sigla=Provincia}}\\&((\rho_{\text{Nome1}\leftarrow\text{Nome}}(\text{DEPUTATI}))\rhd\lhd_{\text{Commissione=Numero}}\\&(\sigma_{\text{Nome="Bilancio"}}(\text{ COMMISSIONE})))\end{aligned}$$

A questo punto le interrogazioni dell'esercizio 3.3 si risolvono più semplicemente:

3.2 $\Pi_{\text{NomeC,Cognome}}$ (COMBILANCIO)
3.3 $\Pi_{\text{NomeC,Cognome,nom1}}$ (COMBILANCIO)
3.4 $\Pi_{\text{NomeC,Cognome,nom1,rege}}$ (COMBILANCIO)

# Esercizio 3.5

Si consideri lo schema di base di dati sulle relazioni:

MATERIE (Codice, Facoltà, Denominazione, Professore)
STUDENTI (Matricola, Cognome, Nome, Facoltà)
PROFESSORI (Matricola, Cognome, Nome)
ESAMI (Studente, Materia, Voto, Data)
PIANIDISTUDIO (Studente, Materia, Anno)

Formulare, in algebra relazionale, in calcolo su domini, in calcolo su tuple e in Datalog le interrogazioni che producono:

1. gli studenti che hanno riportato in almeno un esame una votazione pari a 30, mostrando, per ciascuno di essi, nome e cognome e data della prima di tali occasioni;
2. per ogni insegnamento della facoltà di ingegneria, gli studenti che hanno superato l'esame nell'ultima seduta svolta;
3. gli studenti che hanno superato tutti gli esami previsti dal rispettivo piano di studio;
4. per ogni insegnamento della facoltà di lettere, lo studente (o gli studenti) che hanno superato l'esame con il voto più alto;
5. gli studenti che hanno in piano di studio solo gli insegnamenti della propria facoltà;
6. nome e cognome degli studenti che hanno sostenuto almeno un esame con un professore che ha il loro stesso nome proprio.

## Soluzione:

1)

Algebra Relazionale:

$$\Pi_{Nome,Cognome,\ Data}$$
$$( (\Pi_{Studente,Data}\ (\sigma_{Voto=\ '30'}(ESAMI)) -$$
$$\Pi_{Studente,Data}((\sigma_{Voto=\ '30'}(ESAMI) \rhd\lhd_{(Studente=Studente1)\wedge(Data>Data1)}$$
$$\rho_{Studente1,Corso1,Voto1,Data1\leftarrow\ Studente,Corso,Voto,Data}\sigma_{Voto=\ '30'}(ESAMI)))$$
$$\rhd\lhd_{Studente=Matricola}(STUDENTE)\ )$$

Calcolo dei Domini:

{Nome: fn, Cognome: sur, Data: d |
STUDENTE(Matricola: n, Nome: fn, Cognome: sur, Facoltà: f) ∧
ESAMI(studente: n, Corso: c, Voto: g, Data: d ) ∧(g= '30') ∧
¬(∃c1 (∃ d1 (ESAMI (Studente: n, Corso: c1, Voto: g, Data: d1) ∧ (d1<d)))) }

Calcolo delle Tuple:

{ S.(Nome,Cognome), E.Data | S (STUDENTE), E(ESAMI) |
(S.matricola=E.Studente) ∧ (E.voto= '30') ∧
¬(∃ E1(ESAMI) ((E1.Studente=S.matricola) ∧(E1.voto= '30') ∧ (E1.Data < E.Data))) }

Datalog:

OTHERA (Studente: n, Data: d) ←
ESAMI(Studente: n, Corso: c, Voto: '30', Data: d ),
ESAMI(Studente: n, Corso: c1, Voto: '30', Data: d1 ),
(d > d1)

FIRSTA (Nome: fn, Cognome: sur, Data: d) ←
STUDENTE(Matricola: n, Nome: fn, Cognome: sur, Facoltà: f),
ESAMI(Studente: n, Corso: c, Voto: '30', Data: d ),
not OTHERA(Studente: n, Data: d)

2)

Algebra Relazionale:

$$\sigma_{\ (Voto\geq 18\ )\wedge(Facoltà='Ingegneria')}$$
$$(\Pi_{Cognome,Nome,\ Materia,Voto,Data}$$
$$(STUDENTI\rhd\lhd_{Studente=Matricola}(\Pi_{Studente,Materia,\ Voto,\ Data}\ (\sigma_{(D\geq Data)}$$
$$(ESAMI\rhd\lhd_{Materia=M\ (}\rho_{\ S,M,V,D\leftarrow Studente,Materia,\ Voto,\ Data}(ESAMI)\ ))))))$$

Calcolo Domini:

{Nome: fn, Cognome: sur, Materia: c, Voto: g, Data:d |
STUDENTE(Matricola: sn, Nome: fn, Cognome: sur, Facoltà: f) ∧
ESAMI(studente: sn, Materia: c, Voto: g, Data: d ) ∧
ESAMI(studente: sn, Materia: c, Voto: v1, Data: d1) ∧
(f='Ingegneria') ∧ (d ≥ d1 ) }

Calcolo delle Tuple:

{ S.(Nome,Cognome, Materia,Voto, Data) |
S (STUDENTI), E(ESAMI),E1(ESAMI) |
(E.Materia=E1.Materia) ∧ (E.Data> E1.Data) ∧ (E.Studente=S.Matricola) ∧
(Facoltà="Ingegneria") ∧ (E.Voto≥ 18) }

Datalog:

BRAVISTUDENTI(Nome: fn, Cognome: sur, Materia: c, Voto: g , Data: d) ←
STUDENTI(Matricola: sn, Nome: fn, Cognome: sur, Facoltà: sf),
ESAMI(Studente: sn, Materia: c, Voto: g, Data: d ),
ESAMI(Studente: sn, Materia: c, Voto: g1, Data: d1 ),
(d ≥ d1),(sf = 'Ingegneria'), (g ≥ 18)

3)

Algebra Relazionale:

$\Pi_{Studente}$(PIANIDISTUDIO)-
($\Pi_{Studente}$($\Pi_{Studente,Materia}$(PIANIDISTUDIO) -$\Pi_{Studente,\ Materia}$ (ESAMI)))

Calcolo dei Domini:

{ Studente: n | PIANIDISTUDIO (Studente:n, Materia :c , Anno: y) ∧
(∀c1(∀y1(∃g1,∃d1( ¬( PIANIDISTUDIO (Studente: n, Materia: c1, Anno: y1) ∨
(ESAMI(Studente: n, Materia: c1, Voto: g1, Data: d1)) )))) }

Calcolo delle Tuple:

{ S.Studente | P( PIANIDISTUDIO ) | (∀P( PIANIDISTUDIO ) (∃E(ESAMI)
(¬(S.Studente=P.Studente) ∨ ((P.Studente=E.Studente)∧(S1.Materia=E.Materia)) )) }

Datalog:

ESAMISTUDENTE(Studente: n, Materia: c) ←
ESAMI(Studente: n, Materia: c, Voto: g, Data:d )

ESAMIINCOMPLETISTUDENTE(Studente: n) ←
PIANIDISTUDIO (Studente:n, Materia :c , Anno: y),
NOT ESAMISTUDENTE (Studente: n, Materia : c)

TUTTIESAMI(Studente: n) ← PIANIDISTUDIO (Studente:n, Materia :c , Anno: y)
NOT ESAMIINCOMPLETISTUDENTE(Studente: n)

4)

Algebra Relazionale :

$\Pi_{Studente,Materia}(ESAMI \triangleright\triangleleft_{Materia=Codice} \sigma_{Facoltà= \text{“Lettere”}} (MATERIE)) -$
$\Pi_{Studente,Materia}\sigma_{(Voto< Voto1)\wedge(Facoltà= \text{“Lettere”})}(MATERIE \triangleright\triangleleft_{Codicer= Materia} ESAMI$
$\triangleright\triangleleft_{Materia= Materia1} \rho_{Studente1, Materia1 , Voto1, Data1 \leftarrow Studente, Materia , Voto, Data}(ESAMI) )$

Calcolo dei Domini:

{ Studente: sn, Materia: c | ESAMI(Studente: sn, Materia : c, Voto: g, Data: d ) ∧
MATERIE(Codice: c, Facoltà: f, Denominazione: ct, Professore: t) ∧ (f= "Lettere" ) ∧
¬(∃sn1(∃d1 (∃ g1( ESAMI( Studente: sn1, Materia: c, Date: d1, Voto: g1) ∧ (g1> g) )))) }

Calcolo delle Tuple:

{ E.Studente, E.Materia | E(ESAMI), M(MATERIE) |
(E. Materia =M.Codice) ∧ (M.Facoltà= "Lettere") ∧
((∃E1(ESAMI) ¬(E1. Materia =E. Materia ) ∧ (E1.Voto > E.Voto) ))

Datalog:

STUDENTINONMIGLIORI(Studente: sn, Materia: c) ←
ESAMI(Studente: sn, Materia : c, Voto: g, Data: d),
MATERIE(Number: c, Facoltà : "Lettere", Denominazione: ct, Professore: t),
ESAMI(Studente: sn1, Materia : c, Voto: g1, Data: d1),
(g < g1)

STUDENTIMIGLIORI(Studente: sn, Materia : c) ←
ESAMI(Studente: sn, Materia : c, Voto: g, Data: d),
MATERIE(Codice: c, Facoltà: " Lettere ", Denominazione : ct, Professore : t),
NOT STUDENTINONMIGLIORI(Studente: sn, Materia : c)

5)

Algebra Relazionale:

$\Pi_{Studente}(PIANIDISTUDIO)-$
$\Pi_{Studente}(\sigma_{SfFacoltà \neq Facoltà)}(\rho_{SMatricola,SFacoltà \leftarrow Matricolar,Facoltà}(STUDENTI)$
$\triangleright\triangleleft_{SMatricols=Studente}$ PIANIDISTUDIO
$\triangleright\triangleleft_{Materia=Codice}$ MATERIE ))

Calcolo dei Domini:

{ Studente: n | PIANIDISTUDIO (Studente:n, Materia:c , Anno: y)∧
¬(∃fn(∃sur,∃sf(∃f,(∃ct,∃t(
STUDENTI(Matricola: n, Nome: fn, Cognome: sur, Facoltà: sf) ∧
MATERIE (Codice: c, Facoltà: f, Denominazione: ct, Professore: t)∧(sf≠f) ))))) }

Calcolo delle Tuple:

{ SP.Studente | SP( PIANIDISTUDIO ) | ¬(∃S( STUDENTI ) (∃C( MATERIE )
((SP.Materia=C.Codice)∧(S.Matricola=SP.Studente)∧(C.Facoltà≠S.Facoltà)) )) }

Datalog:

DIFFERENTEFACOLTA(Studente: n) ←
PIANIDISTUDIO (Studente:n, Materia:c , Anno: y),
STUDENTS(Matricola: n, Nome: fn, Cognome: sur, Facoltà: sf) ,
MATERIE (Codice: c, Facoltà: f, Denominazione: ct, Professore: t),
(sf≠f)

STESSAFACOLTA(Studente: n) ←
PIANIDISTUDIO (Studente:n, Materia :c , Anno: y),
NOT DIFFERENTEFACOLTA(Studente: n)

6)

Algebra Relazionale:

$\Pi_{Nome,Cognome}$ ($\sigma_{Nome=NomeP}$(
(ESAMI$\triangleright\triangleleft_{Studente= Matricola}$ STUDENTI)$\triangleright\triangleleft_{Materia=Codice}$
($\rho_{NomeP,CognomeP \leftarrow Nome,Cognome}$(PROFESSORI $\triangleright\triangleleft_{Matricola=Professore}$ MATERIE))))

Calcolo dei Domini:

{ Nome: fn, Cognome: sur |
STUDENTI(Matricola: n, Nome: fn, Cognome: sur, Facoltà: sf) ∧
ESAMI(Studente: n, Materia: c, Voto: g, Data: d ) ∧
MATERIE( Codice: c, Facoltà: f, Denominazione: ct, Professore: t)∧
PROFESSORI( Matricola: t, Nome:fn, Cognome: tfn) }

Calcolo delle Tuple:

{ S.(Nome,Cognome) |
S(STUDENTI), E (ESAMI), M(MATERIE), P(PROFESSORI) |
(S.Matricola=E.Studente)∧(E.Materia=M.Codice) ∧ (M.Professore=P.Matricola) ∧
(P.Nome=S.Nome) }

Datalog:

STESSONOME ( Nome: fn, Cognome: sur) ←
STUDENTI(Matricola: n, Nome: fn, Cognome: sur, Facoltà: sf) ,
ESAMI(Studente: n, Materia: c, Voto: g, Data: d ) ,
MATERIE( Codice: c, Facoltà: f, Denominazione: ct, Professore: t),
PROFESSORI( Matricola: t, Cognome:surf, Nome: fn)

## Esercizio 3.6

Con riferimento al seguente schema di base di dati:

CITTÀ (Nome, Regione, Abitanti)
ATTRAVERSAMENTI (Città, Fiume)
FIUMI (Fiume, Lunghezza)

formulare, in algebra relazionale,in calcolo sui domini, in calcolo su tuple e in Datalog le seguenti interrogazioni:

1. Visualizza nome, regione e abitanti per le città che hanno più di 50000 abitanti e sono attraversate dal Po oppure dall'Adige;
2. Trovare le città che sono attraversate da (almeno) due fiumi, visualizzando il nome della città e quello del più lungo di tali fiumi.

### Soluzione:

1) Algebra Relazionale:

$\Pi_{Nome,\ Regione,\ Abitanti}\ (\ \sigma_{(Fiume="Po")\vee(Fiume="Adige")}(ATTRAVERSAMENTO) \triangleright\triangleleft_{Citta=Nome}$
$\sigma_{Abitanti>50000}(CITTA)\ )$

Calcolo dei Domini:

{ Nome: n, Regione: reg, Abitanti: p | CITTA (Name: n, Regione : reg, Abitanti : p) ∧
ATTRAVERSAMENTO ( Città: n, Fiume: r )∧ (p > 50000) ∧
((r= "Po") ∨ ( r= "Adige")) }

Calcolo delle Tuple:

{ C.(Nome,Regione, Abitanti ) | C(CITTA), A (ATTRAVERSAMENTO) |
(C.Nome= A.Città) ∧ (C. Abitanti > 50000) ∧
((A.Fiume= "Po") ∨ (A.Fiume= "Adige")) }

Datalog:

POEADIGE (Città: c) ←
ATTRAVERSAMENTO (Città: c, Fiume: r),
(r = "Po")

2) Algebra Relazionale:

$\Pi_{Città,Fiume}\ (\sigma_{Fiume \neq Fiume\ 1}\ (ATTRAVERSAMENTO \rhd\lhd_{Città=Città1}$
$\rho_{Città1,Fiume\ 1 \leftarrow Città,\ Fiume}\ (\ ATTRAVERSAMENTO\ )))\ -$

$\Pi_{Città,Fiume}\ (\sigma_{(Fiume \neq Fiume1) \wedge (Lughezza\ <\ Lunghezza1)}((FIUMI \rhd\lhd ATTRAVERSAMENTO\ ) \rhd\lhd_{Città=Città1}$
$\rho_{Città1,Fiume1,\ Lunghezza1 \leftarrow Città,Fiume,\ Lunghezza}(\ FIUMI\ \rhd\lhd\ ATTRAVERSAMENTO))\ ))$

Calcolo dei Domini:

{ Città: n, Fiume: r | ATTRAVERSAMENTO (Città: n, Fiume: r) ∧
ATTRAVERSAMENTO (Città: n, Fiume: r1) ∧ (r≠r1) ∧
FIUMI(Fiume: r, Lunghezza: l ) ∧
(∀r2(∀l2¬(( ATTRAVERSAMENTO (Città: n, Fiume: r2)∧
FIUMI (Fiume: r2, Lunghezza: l2)) ∨ (l2<l) ))) }

Calcolo delle Tuple:

{C.( Città , Fiume ) |
C( ATTRAVERSAMENTO ), C1( ATTRAVERSAMENTO ), F( FIUMI ) |
(C. Città =C1. Città ) ∧ (C. Fiume ≠ C1. Fiume ) ∧ (C. Fiume =F. Fiume ) ∧
(∀C2( ATTRAVERSAMENTO ) (∀F2( FIUMI ) (C2.Città ≠C. Città )
∨( (F2. Fiume =F. Fiume )∧(F2. Lunghezza < F. Lunghezza ) ) ) )) }

Datalog:

FIUMECORTO ( Città : c, Fiume : r) ←
ATTRAVERSAMENTO ( Città : c, Fiume : r),
FIUMI ( Fiume : r, Lunghezza : l ),
ATTRAVERSAMENTO ( Città : c, Fiume : r1),
FIUMI ( Fiume : r1, Lunghezza : l1 ),
(l < l1)

FIUMELUNGO ( Città : c, Fiume : r) ←
ATTRAVERSAMENTO ( Città : c, Fiume : r),
ATTRAVERSAMENTO ( Città : c, Fiume : r1),
(r ≠r1),
NOT FIUMECORTO ( Città : c, Fiume : r)

# Esercizio 3.7

Con riferimento al seguente schema di base di dati:

AFFLUENZA (Affluente, Fiume)
FIUMI (Fiume, Lunghezza)

formulare l'interrogazione in Datalog che trova tutti gli affluenti, diretti e indiretti dell'Adige.

## Soluzione:

TUTTIAFFLUENTI (Affluenti: t) ←
AFFLUENZA ( Affluenti : t, Fiume: "Adige" )

TUTTIAFFLUENTI ( Affluenti : t ) ←
AFFLUENZA ( Affluenti : t, Fiume:r ),
TUTTIAFFLUENTI ( Affluenti : r)

# Esercizio 3.8

Si consideri lo schema relazionale composto dalle seguenti relazioni:

PROFESSORI (Codice, Cognome, Nome)
CORSI ( Codice, Denominazione, Professore)
STUDENTI (Matricola, Cognome, Nome)
ESAMI (Studente, Corso, Data, Voto)

Formulare, con riferimento a tale schema, le espressioni dell'algebra, del calcolo relazionale su tuple e del Datalog, che producano:

1. Gli esami superati dallo studente Pico della Mirandola (supposto unico), con indicazione, per ciascuno, della denominazione del corso, del voto e del cognome del professore;
2. i professori che tengono due corsi (e non più di due), con indicazione di cognome e nome del professore e denominazione dei due corsi.

## Soluzione:

**1)** Algebra Relazionale:

$$\Pi_{\text{Denominazione,Voto,CognomeP,NomeP}}( \sigma_{\text{Cognome= "della Mirandola"} \wedge \text{ Nome= "Pico"}} (\text{STUDENTI} \rhd\lhd_{\text{Matricola=Studente}} \text{ESAMI} \rhd\lhd_{\text{Corso=CCodice}} \rho_{\text{CCodice}\leftarrow\text{Codice}}(\text{CORSI}) \rhd\lhd_{\text{Professore=TCodice}} \rho_{\text{TCodice,T CognomeP,NomeP}\leftarrow\text{Codice, Cognome,Nome}} (\text{PROFESSORI})))$$

Calcolo delle Tuple:

{ C.Denominazione, E.Voto, P.(Cognome,Nome) | C( CORSI ),
E( ESAMI ), P( PROFESSORI ), S( STUDENTI) |
(S.Matricola=E.Studente) ∧ (E.Corso=C.Codice) ∧ ( C.Professore= P.Codice) ∧
(S.Cognome= "Della Mirandola" ) ∧ (S.Nome= "Pico" )

Datalog:

ESAMIPICO( Denominazione: cn, Voto: g, Cognome: tsur, Nome: tname ) ←
STUDENTI(Matricola: n, Cognome: "Della Mirandola", Nome: "Pico"),
ESAMI ( Studente: n, Corso: c, Voto: g, Data: d),
CORSI (Codice: c, Denominazione:cn, Professore: t),
PROFESSORI (Codice: t, Cognome: tsur, Nome: tname )

**2)** Algebra Relazionale:

$\Pi_{\text{Cognome,Nome,Denominazione, Denominazione1}}$
$(\Pi_{\text{Professore,Denominazione,Denominazione1}}$ (CORSI $\triangleright\triangleleft_{(\text{,Professore=,Professore1})\wedge(\text{Codice}\neq\text{(Codice)}}$
$\rho_{\text{Codice1,Denominazione1,Professore1}\leftarrow\text{ Codice,Denominazione, Professore}}$ (CORSI) ) -
$\Pi_{\text{Professore,Denominazione, Denominazione1}}$ ($\sigma_{\text{Codice}\neq\text{Codice2}}$(CORSI $\triangleright\triangleleft_{(\text{Professore=Professore1})\wedge(\text{Codice}\neq\text{Codice1})}$
$\rho_{\text{Codice1, Denominazione1 ,Professore1}\leftarrow\text{ Codice, Denominazione , Professore}}$ (CORSI)
$\triangleright\triangleleft_{(\text{Professore1=Professore2})\wedge(\text{Codice 1}\neq\text{ Codice 2})\ \wedge(\text{Codice}\neq\text{ Codice 2})}$
$\rho_{\text{ Codice 2,Denominazione2,Professore2}\leftarrow\text{ Codice,Denominazione, Professore}}$ (CORSI)))
$\triangleright\triangleleft_{\text{Professore=PCodice}}$ $\rho_{\text{PCodice}\leftarrow\text{Codice}}$ (PROFESSORI))

Calcolo delle tuple:
{ P.(Nome,Cognome), C.Denominazione, C1. Denominazione |
P( PROFESSORI ), C ( CORSI ), C1( CORSI ) |
(P.Codice)=(C. Professore ) ∧ (P. Codice )=(C1. Professore ) ∧
(C. Codice . Codice ) ∧
¬(∃ C2( CORSI ) ((C2.Professore=P. Codice ) ∧
(C2. Codice ≠ C. Codice ) ∧(C2. Codice ≠ C1. Codice )∧(C. Codice ≠ C1. Codice ) ) }

Datalog:

PIUDIDUE ( Professore : t ) ←
CORSI ( Codice : n, Denominazione : cn, Professore : t ),
CORSI ( Codice : n1, Denominazione : cn1, Professore : t ),
CORSI ( Codice : n1, Denominazione : cn1, Professore : t ),
(n≠n1),(n≠n2),(n1≠n2)

ESATTAMENTEDUE ( Nome : fn, Cognome: sn, Corso1 : cn1, Corso2 : cn2 ) ←
PROFESSORI ( Codice : t, Nome: fn, Cognome: sn),
CORSI ( Codice : n1, Denominazione : cn1, Professore : t ),
CORSI ( Codice : n2, Denominazione : cn2, Professore : t ),
(n1≠n2),
NOT PIUDIDUE( Professore : t)

# Esercizio 3.9

Considerare uno schema relazionale contenente le relazioni

$R_1(ABC)$, $R_2(DG)$, $R_3(EF)$

Formulare in calcolo relazionale su tuple e su domini l'interrogazione realizzata in algebra relazionale dalla seguente espressione:

$(R_3 \rhd\lhd_{G=E} R_2) \cup ( \rho_{DG \leftarrow AC} ( \Pi_{ACEF} (R_1 \rhd\lhd_{B=F} R_3 )))$

## Soluzione:

Questa espressione non è esprimibile in calcolo sulle tuple a causa dell'unione tra due diverse tabelle.
In calcolo sui domini l'espressione diventa:

$\{ D: d, G: g, E: e, F: f \mid R_3(E:e, F:f) \wedge ( (R_2( D: d, G: g) \wedge (g=e)) \vee (R_1(A: d, B: b, C: g) \wedge (b=f))) \}$

# Esercizio 3.10

Con riferimento allo schema dell'esercizio 3.9, formulare in algebra relazionale le interrogazioni realizzate in calcolo su domini dalle seguenti espressioni:

$\{ H: g, B: b \mid R_1( A: a, B: b, C: c) \wedge R_2( D: c, G: g) \}$
$\{ A: a, B: b \mid R_2(D:a , G: b) \wedge R_3 (E: a, F: b) \}$
$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge$
$\quad \exists a' (R_1(A: a' , B: b, C: c) \wedge a \neq a' ) \}$
$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge$
$\quad \forall a' (\neg R_1(A: a' , B: b, C: c) \vee a=a' ) \}$
$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge$
$\quad \neg\exists a'( R_1(A: a' , B: b, C: c) \wedge a \neq a' ) \}$

## Soluzione:

$\rho_{H \leftarrow G} (\Pi_{BG} ( R_1 \rhd\lhd_{C=D} R_2 ))$
$\rho_{AB \leftarrow DG} ( \Pi_{DG} (R_2 \rhd\lhd_{(D=E) \wedge (G=F)} R_3 ))$
$\Pi_{AB}( \sigma_{A \neq A1} ( R_1 \rhd\lhd_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$
$\Pi_{AB} (R_1) - \Pi_{AB}( \sigma_{A \neq A1} ( R_1 \rhd\lhd_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$
$\Pi_{AB} (R_1) - \Pi_{AB}( \sigma_{A \neq A1} ( R_1 \rhd\lhd_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$

# Esercizio 3.11

Trasformare la seguente espressione dall'algebra, che fa riferimento allo schema

$$R_1(AB),\ R_2\ (CDE),\ R_3(FGH)$$

con l'obiettivo di ridurre le dimensioni dei risultati intermedi:

$\Pi_{ADH}\ (\ \sigma_{(B=C)\wedge(E=F)\wedge(A>20)\wedge(G=10)}\ ((\ R_1 \rhd\lhd R_3)\ \rhd\lhd\ R_2\ )$

## Soluzione:

$\Pi_{ADH}\ (\ \sigma_{A>20}(R_1) \rhd\lhd_{B=C}\ \Pi_{CDH}\ (R_2 \rhd\lhd_{E=F}\ \Pi_{FH}\ (\sigma_{G=10}(R_3)))\ )$

# Capitolo 4

## Esercizio 4.1

Ordinare i seguenti domini in base al valore massimo rappresentabile, supponendo che `integer` abbia una rappresentazione a 32 bit e `smallint` a 16 bit: `numeric(12,4)`, `decimal(10)`, `decimal(9)`, `integer`, `smallint`, `decimal(6,1)`.

### Soluzione:

| Dominio | Valore Massimo |
|---|---|
| 1. `decimal(10)` | 9999999999 |
| 2. `integer` | 4294967296 |
| 3. `decimal(9)` | 999999999 |
| 4. `numeric(12,4)` | 99999999.9999 |
| 5. `decimal(6,1)` | 99999.9 |
| 6. `smallint` | 65536 |

## Esercizio 4.2

Definire un attributo che permetta di rappresentare stringhe di lunghezza massima pari a 256 caratteri, su cui non sono ammessi valori nulli e con valore di default "sconosciuto".

### Soluzione:

```
Create domain STRING as character varying (256) default 'sconosciuto'
not null
```

# Esercizio 4.3

Dare le definizioni SQL delle tre tabelle

```
FONDISTA(Nome, Nazione, Età)
GAREGGIA(NomeFondista, NomeGara, Piazzamento)
GARA(Nome, Luogo, Nazione, Lunghezza)
```

rappresentando in particolare i vincoli di foreign key della tabella `GAREGGIA`.

## Soluzione:

```
Create Table FONDISTA
(
Nome     character(20)      primary key,
Nazione  character(30),
Età      smallint
)

Create table GARA
(
Nome     character(20)    primary key,
Luogo    character(20),
Nazione  character(20),
Lunghezza        integer
)

Create table GAREGGIA
(
NomeFondista     character(20)   references FONDISTA(Nome),
NomeGara         character(20),
Piazzamento      smallint,
primary key (NomeFondista, NomeGara),
foreign key (NomeGara)    references GARA(Nome)
)
```

# Esercizio 4.4

Dare le definizioni SQL delle tabelle

```
AUTORE (Nome, Cognome, DataNascita, Nazionalità)
LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)
```

Per il vincolo foreign key specificare una politica di `cascade` sulla cancellazione e di `set null` sulle modifiche.

## Soluzione:

```
Create table AUTORE
(
Nome              character(20),
Cognome           character(20),
DataNascita       date,
Nazionalità       character(20),
primary key(Nome, Cognome)
)

Create table LIBRO
(
TitoloLibro       character(30)    primary key,
NomeAutore        character(20),
CognomeAutore     character(20),
Lingua            character(20),
foreign key (NomeAutore, CognomeAutore)
                  references AUTORE(Nome, Cognome)
                  on delete cascade
                  on update set NULL
)
```

# Esercizio 4.5

Dato lo schema dell'esercizio precedente, spiegare cosa può capitare con l'esecuzione dei seguenti comandi di aggiornamento:

```
delete from AUTORE where Cognome = 'Rossi'

update LIBRO set NomeAutore= 'Umberto'
          where CognomeAutore = 'Eco'

insert into AUTORE(Nome,Cognome) values('Antonio','Bianchi')

update AUTORE set Nome = 'Italo' where Cognome = 'Calvino'
```

## Soluzione:

1. Il comando cancella dalla tabella AUTORE tutte le tuple con Cognome = 'Rossi'. A causa della politica `cascade` anche le tuple di LIBRO con CognomeAutore = 'Rossi' verranno eliminate.

2. Il comando non è corretto: Nome e Cognome sono attributi della tabella AUTORE e non della tabella LIBRO
3. Il comando aggiunge una nuova tupla alla tabella AUTORE. Non ha alcun effetto sulla tabella LIBRO
4. Le tuple di AUTORE con Cognome = Calvino vengono aggiornate a Nome = Italo. A causa della politica `set null` gli attributi NomeAutore e CognomeAutore delle tuple di Libro con CognomeAutore = Calvino vengono posti a `NULL`.

## Esercizio 4.6

Date le definizioni:

```
create domain Dominio as integer default 10
create table Tabella (Attributo Dominio default 5)
```

indicare cosa avviene in seguito ai comandi:

```
1. alter table Tabella alter column Attributo drop default
2. alter domain Dominio drop default
3. drop domain Dominio
```

### Soluzione:

1. Il comando cancella il valore di default di Attributo. Il valore di default dopo il comando sarà quello impostato in Dominio, ossia 10.
2. Il comando cancella il valore di default di Dominio. Dopo l'operazione il valore di default sarà `NULL`
3. Il comando cancella l'intero dominio Domain. In Tabella il dominio di Attributo diventerà `integer`

## Esercizio 4.7

Dato il seguente schema:

```
AEROPORTO(Città, Nazione,NumPiste)
VOLO(IdVolo,GiornoSett,CittàPart,OraPart,
                  CittàArr,OraArr,TipoAereo)
AEREO(TipoAereo,NumPasseggeri,QtaMerci)
```

scrivere le interrogazioni SQL che permettono di determinare:

1. Le città con un aeroporto di cui non è noto il numero di piste;
2. Le nazioni da cui parte e arriva il volo con codice AZ274;
3. I tipi di aereo usati nei voli che partono da Torino;
4. I tipi di aereo e il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile, visualizzare solamente il tipo;
5. Le città da cui partono voli internazionali;
6. Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente;
7. Il numero di voli internazionali che partono il giovedì da Napoli;
8. Il numero di voli internazionali che partono ogni settimana da città italiane (farlo in due modi, facendo comparire o meno nel risultato gli aeroporti senza voli internazionali);
9. Le città francesi da cui partono più di venti voli alla settimana diretti in Italia;

10. Gli aeroporti italiani che hanno solo voli interni. Rappresentare questa interrogazione in quattro modi:
    a) con operatori insiemistici;
    b) con un interrogazione nidificata con l'operatore `not in`;
    c) con un interrogazione nidificata con l'operatore `not exists`;
    d) con l'outer join e l'operatore di conteggio
11. Esprimere l'interrogazione pure in algebra relazionale;
12. Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri;
13. Il massimo numero di passeggeri che possono arrivare in un aeroporto italiano dalla Francia il Giovedì (se vi sono più voli, si devono sommare i passeggeri)

## Soluzione:

```
1.      select Città
        from AEROPORTO
        where NumPiste is NULL

2.      select A1.Nazione, A2.Nazione
        from AEROPORTO as A1 join VOLO on A1.Città=CittàArr
        join AEROPORTO as A2 on CittàPart=A2.Città
        where IdVolo= 'AZ274'

3.      select TipoAereo
        from VOLO
        where CittàPart='Torino'

4.      select VOLO.TipoAereo, NumPasseggeri
        from VOLO left join AEREO
        on  VOLO.TipoAereo=aereo.TipoAereo
        where CittàPart= 'Torino'

5.      select CittàPart
        from AEROPORTO as A1 join VOLO on CittàPart=A1.Città
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione <> A2.Nazione

6.      select CittàPart
        from VOLO
        where CittàArr= 'Bologna'
        order by CittàPart

7.      select count(*)
        from VOLO join AEROPORTO on CittàArr=Città
        where CittàPart = 'Napoli' and Nazione <> 'Italia' and
        GiornoSett= 'Giovedì'

8.  a.  select count(*), CittàPart
        from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
        group by CittàPart
```

b.
```
select count(CittàArr)
from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
group by CittàPart
```

9.
```
select CittàPart
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione='Francia' and A2.Nazione= 'Italia'
group by CittàPart
Having count(*) >20
```

10. a.
```
select CittàPart
from VOLO join AEROPORTO on CittàPart=Città
where Nazione = 'Italia'
except
select CittàPart
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
join AEROPORTO as A2 on CittàArr=A2.Città
where (A1.Nazione=' Italia ' and A2.Nazione<>' Italia ' )
```

b.
```
select CittàPart
from VOLO join AEROPORTO on CittàPart=Città
where Nazione= 'Italia' and CittàPart not in
( select CittàPart
from AEROPORTO as A1 join VOLO on
A1.Città=CittàPart join AEROPORTO as A2 on CittàArr=A2.Città
where  A1.Nazione='Italia' and
A2.Nazione<> 'Italia'  )
```

c.
```
select CittàPart
from VOLO join AEROPORTO as A1 on CittàPart=Città
where Nazione= 'Italia' and
not exists ( select * from VOLO join AEROPORTO as A2
on A2.Città=CittàArr
where A1.Città=CittàPart and
A2.Nazione<>'Italia'  )
```

d.
```
select CittàPart
from AEROPORTO as A1 join VOLO on
A1.Città=CittàPart left join AEROPORTO as A2 on
(CittàArr=A2.Città and A2.Nazione='Italia')
where A1.Nazione='Italia'
group by CittàPart
having count (district A2.Nazione)= 1 )
```

e. $\Pi_{\text{Cittàpart}}\ \sigma_{\text{Nazione='Italia'}}(\text{AEROPORTO} \triangleright\!\triangleleft_{\text{Città=CittàPart}} \text{VOLO})$
$-$
$\Pi_{\text{CittàPart}}\ \sigma_{\text{Nazione='Italia'}}\ (\text{AEROPORTO} \triangleright\!\triangleleft_{\text{Città=CittàPart}} \text{VOLO} \triangleright\!\triangleleft_{\text{CittàArr=CittàK}}\ \rho_{\text{CittàK,NazioneK,nK}\leftarrow\text{Città,Nazione,NumPiste}}(\sigma_{\text{Nazione}\neq\text{'Italia'}}(\text{AEROPORTO}))))$

11.
```
select CittàPart
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=      (select
                           max( NumPasseggeri )from AEREO )
union
select CittàArr
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=      (select max( NumPasseggeri )
                           from AEREO)
```

12.
```
create view Passeggeri(Numero)
as select sum ( NumPasseggeri )
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
        join AEROPORTO as A2 on A2.Città=CittàArr
        join AEREO on VOLO.TipoAereo=Aereo.TipoAereo
where A1.Nazione='Francia' and A2.Nazione='Italia'
    and GiornoSett='Giovedì'
group by A2.Città

select max(Numero)
from Passeggeri
```

## Esercizio 4.8

Dato il seguente schema:

```
DISCO(NroSerie, TitoloAlbum,Anno,Prezzo)
CONTIENE(NroSerieDisco,CodiceReg,NroProg)
ESECUZIONE(CodiceReg,TitoloCanz,Anno)
AUTORE(Nome,TitoloCanzone)
CANTANTE(NomeCantante,CodiceReg)
```

formulare le interrogazioni SQL che permettono di determinare:

1. I cantautori (persone che hanno cantato e scritto la stessa canzone) il cui nome inizia per 'D';
2. I titoli dei dischi che contengono canzoni di cui non si conosce l'anno di registrazione;
3. I pezzi del disco con numero di serie 78574, ordinati per numero progressivo, con indicazione degli interpreti per i pezzi che hanno associato un cantante;
4. Gli autori e i cantanti puri, ovvero autori che non hanno mai registrato una canzone e cantanti che non hanno mai scritto una canzone;
5. I cantanti del disco che contiene il maggior numero di canzoni;
6. Gli autori solisti di "collezioni di successi" (dischi in cui tutte le canzoni sono di un solo cantante e in cui almeno tre registrazioni sono di anni precedenti la pubblicazione del disco);
7. I cantanti che non hanno mai registrato una canzone come solisti;
8. I cantanti che non hanno mai inciso un disco in cui comparissero come unici cantanti;
9. I cantanti che hanno sempre registrato canzoni come solisti.

**Soluzione:**

1.
```
select NomeCantante
from  CANTANTE join ESECUZIONE on
CANTANTE.CodiceReg=ESECUZIONE.CodiceReg
join AUTORE on ESECUZIONE.TitoloCanz=AUTORE.TitoloCanzone
where Nome=NomeCantante and Nome like 'd%'
```

2.
```
select TitoloAlbum
from DISCO join CONTIENE
        on DISCO.NroSerie=CONTIENE.NroSerieDisco
        join ESECUZIONE on
        CONTIENE.CodiceReg=ESECUZIONE.CodiceReg
where ESECUZIONE.anno is NULL
```

3.
```
select NroProg, TitoloCanz, NomeCantante
from (CONTIENE left join CANTANTE on
        CONTIENE.NroSerieDisco=CANTANTE.CodiceReg)
        join ESECUZIONE
        on CONTIENE.codiceReg= ESECUZIONE.CodiceReg
where NroSerieDisco=78574
order by NroProg
```

4.
```
select Nome
from AUTORE
where Nome not in
        ( select NomeCantante
          from CANTANTE )
union
select NomeCantante
from CANTANTE
where NomeCantante not in
      ( select Nome
                  from AUTORE )
```

5.
```
create view DiscoNumerato (NroSerieDisco,NumCanzoni)
as select  NroSerieDisco , count(*)
from CONTIENE
group by NumSerieDisco

select NomeCantante
from CANTANTE join CONTIENE on
                CANTANTE.CodiceReg=CONTIENE.CodiceReg
                join DiscoNumerato on

CONTIENE.NroSerieDisco=DiscoNumerato.NroSerieDisco
where NumCanzoni= ( select max (NumCanzoni)
                            from DiscoNumerato)
```

6.
```
select NroSerie
from DISCO
where NroSerie not in
( select NroSerieDisco
  from CONTIENE join CANTANTE as S1 on
  CONTIENE.CodiceReg=S1.CodiceReg
  join CANTANTE as S2 on
  CONTIENE.CodiceReg =S2.CodiceReg
  where   S1.NomeCantante<>S2.NomeCantante )
   and NroSerie in
 ( select NroSerieDisco
  from (CONTIENE join ESECUZIONE on
   CodiceReg= CodiceReg ) join DISCO on
  DISCO.NroSerie=CONTIENE.NroSerieDisco)
   where ESECUZIONE.Anno<DISCO.Anno
  group by NroSerieDisco
  having count(*) >=3 )
```

7.
```
select distinct NomeCantante
from CANTANTE
where NomeC not in
     ( select S1.NomeCantante
       from CANTANTE as S1
       where CodiceReg not in
        ( select CodiceReg
          from CANTANTE S2
          where S2.NomeCantante <> S1.NomeCantante ) )
```

8.
```
Create view CantantiUnDisco (NomeCantante) as
select NomeCantante
from CONTIENE join CANTANTE on
         CONTIENE.CodiceReg=CANTANTE.CodiceReg
where NroSerieDisco not in
     (  select NroSerieDisco
        from CONTIENE join CANTANTE as S1 on
                     CONTIENE.CodiceReg=S1.CodiceReg
                      join CANTANTE as S2 on
                     CodiceReg=S2.CodiceReg
        where S1.NomeCantante<>S2.NomeCantante  )

select NomeCantante
from CANTANTE
where NomeCantante not in (select NomeCantante
                           from CantantiUnDisco)
```

9.
```
select NomeCantante
from CANTANTE
where NomeCantante not in
( select S1.NomeCantante
from CANTANTE as S1 join ESECUZIONE on
CodiceReg=S1.CodiceReg join CANTANTE as S2 on
CodiceReg=S2.CodiceReg )
where S1.NomeCantante<> S2.NomeCantante )
```

# Esercizio 4.9

Dare una sequenza di comandi di aggiornamento che modifichi l'attributo Stipendio della tabella Impiegato, aumentando del 10% gli stipendi sotto i 30 mila € diminuendo del 5% gli stipendi sopra i 30 mila € (gli stipendi di 30.000 € rimangono invariati).

## Soluzione:

```
update Impiegato set Stipendio= Stipendio*0.5
where Stipendio < 30000

update Impiegato set  Stipedio= Stipendio*0.95
where Stipendio > 30000

update Impiegato set Stipendio= Stipendio*2.2
where Stipendio < 15000
```

# Esercizio 4.10

Definire sulla tabella `Impiegato` il vincolo che il dipartimento `Amministrazione` abbia meno di 100 dipendenti, con uno stipendio medio superiore ai 40 mila €.

## Soluzione:

```
check (100 >= ( select count(*)
                 from Impiegato
                where Dipartimento='Amministrazione' )
       and 40000 <= ( select avg(Stipendio)
                      from Impiegato
                      where
                      Dipartimento='Amministrazione'))
```

# Esercizio 4.11

Definire a livello di schema il vincolo che il massimo degli stipendi degli impiegati di dipartimenti con sede a Firenze sia minore dello stipendio di tutti gli impiegati del dipartimento Direzione.

## Soluzione:

```
create assertion ControlloSalari
  check ( not exists(     select *
                          from Impiegato join Dipartimento on
                          Impiegato.Dipartimento=Dipartimento.Nome
                          where Dipartimento.Città='Firenze' and
                          Stipendio > (    select min(Stipendio)
                                           from Impiegato
                                           where
                                           Dipartimento='Direzione')
          )          )
```

## Esercizio 4.12

Definire una vista che mostra per ogni dipartimento il valore medio degli stipendi superiori alla media del dipartimento

### Soluzione:

```
create view SalariSopraMedia (Dipartimento,Stipendio) as
select Dipartimento, avg(Stipendio)
from I
where Stipendio > ( select avg(Stipendio)
                          from Impiegato as I )
                    where I.Dipartimento=I.Dipartimento )
group by Dipartimento
```

## Esercizio 4.13

Tramite la definizione di una vista permettere all'utente "`Carlo`" di accedere al contenuto di `Impiegato`, escludendo l'attributo S`tipendio`.

### Soluzione:

Ipotizzando la tabella Impiegato

```
Impiegato(Codice, Nome, Cognome, Stipendio, Dipartimento)
create view ImpiegatoRistretto
(Codice,Nome,Cognome,Dipartimento) as
select  Codice,Nome,Cognome,Dipartimento
from Impiegato

grant select on ImpiegatoRistretto to Carlo
```

## Esercizio 4.14

Descrivere l'effetto delle seguenti istruzioni: quali autorizzazioni sono presenti dopo ciascuna istruzione? (ciascuna linea è preceduta dal nome dall'utente che esegue il comando)

```
Stefano:        grant select on Table1 to Paolo,Riccardo
                        with grant option
Paolo:          grant select on Table1 to Piero
Riccardo:       grant select on Table1 to Piero with grant option
Stefano:        revoke select on Table1 from Paolo cascade
Piero:          grant select on Table1 to Paolo
Stefano:        revoke select on Table1 from Riccardo cascade
```

## Soluzione:

1. Stefano concede a Paolo e a Riccardo l'autorizzazione di `select` e di concedere a loro volta l'autorizzazione
2. Paolo concede a Piero l'autorizzazione di `select`
3. Riccardo concede a Piero l'autorizzazione di `select` e di `grant`. Ora Piero ha 2 diverse autorizzazioni sulla tabella.
4. Stefano revoca l'autorizzazione data a Paolo. A causa dell'attributo `cascade` anche Piero perde le autorizzazioni concesse da Paolo ma continua ad avere quella concessa da Riccardo.
5. Ora Paolo può di nuovo accedere alla tabella grazie all'autorizzazione concessa da Piero
6. Stefano revoca l'autorizzazione di Riccardo e tramite `cascade` anche di Piero e di Paolo. Ora solo Stefano ha autorizzazioni sulla tabella.

# Capitolo 6

## Esercizio 6.1

Realizzare una procedura in un linguaggio di programmazione di alto livello che tramite SQL Embedded elimina dalla tabella DIPARTIMENTO l'elemento che ha il nome che viene fornito come parametro alla procedura.

### Soluzione:

Soluzione in C:

```
#include<stdlib.h>
main()
{
char Nome1;
char Nome2;
begin declare section;
         char Nome;
exec sql end declare section;

exec sql declare DipCursore for
         select Nome
         from Dipartimento;
exec sql open DipCursore;

do {
exec sql fetch DipCursore into
         :Nome1;
if (Nome1 == Nome2)
         exec sql delete from Dipartimento
         where nome = :Nome1;
}while (sqlca.sqlcode==0)
}
```

# Esercizio 6.2

Realizzare un programma in un linguaggio di programmazione di alto livello che tramite SQL Embedded costruisce una videata in cui si presentano le caratteristiche di ogni dipartimento seguito dall'elenco degli impiegati che lavorano nel dipartimento, ordinati per cognome

## Soluzione:

```
#include<stdlib.h>
main()
{

exec sql begin declare section;
        char Nome[20], Città[20], CognomeImpiegato[20],
                NomeImpiegato[20];
        int NumeroDip;
exec sql end declare section;

exec sql declare DipCursore cursor for
        select Nome, Citta, NumDip
        from DIPARTIMENTO;

exec sql declare ImpCursore cursor for
        select CognomeImpiegato, NomeImpiegato
        from Impiegato join Dipartimento on
        Impiegato.Dipartimento=Dipartimento.:Nome
        order by CognomeImpiegato;
exec sql open DipCursore;
do
{
exec sql fetch DipCursore into
        :Nome, :Città, :NumeroDip;
printf("Dipartimento: ", Nome, " situato in: ",Città,
        " Numero Dipendenti: ",NumeroDip." "Dipendenti:");
        exec sql open ImpCursore;
        do
        {
        exec sql fetch ImpCursore into
                :CognomeImpiegato, :NomeImpiegato;
        printf( CognomeImpiegato," ", NomeImpiegato);
        }while(sqlca.sqlcode==0);
        exec sql close cursor ImpCursore;
}while(sqlca.sqlcode==0);
exec sql close cursor DipCursore;
}
```

# Esercizio 6.3

Realizzare l'esercizio precedente usando ADO

**Soluzione:**

```
#include<stdlib.h>
main()
{

conn ADODB.Connection;
impiegati ADODB.Recordset;
dipartimenti ADODB.Recordset;
comandoSQL ADODB.Command;
buffer string;

conn = new ADODB.connection;
conn.open("Server","Utente","Password");

dipartimenti = New ADODB.Recordset;
buffer="select Nome, Citta, NumDip from DIPARTIMENTO";
comandoSQL.CommandText = buffer;
dipartimenti.Open ComandoSQL(conn);




do{
        printf("Dipartimento: ", dipartimenti!Nome, " situato in:
        ",dipartimenti.Città," Numero Dipendenti:
        ",dipartimenti.NumeroDip." "Dipendenti:");

do
{
        impiegati = New ADODB.Recordset;
        buffer = "select CognomeImpiegato, NomeImpiegato
                 from Impiegato join Dipartimento on
                 Impiegato.Dipartimento=Dipartimento.",
                 dipartimenti!Nome,
                 "order by CognomeImpiegato";
        comandoSQL.CommandText = buffer;
        impiegati.Open ComandoSQL(conn);

        printf( impiegati.CognomeImpiegato," ",
                 impiegati.NomeImpiegato);
        impiegati.movenext;
        }while(impiegati.EOF);
dipartimenti.movenext;
}while(dipartimenti.EOF);
}
```

# Esercizio 6.4

Realizzare un programma java che scandisce gli impiegati ordinati per cognome e inserisce ogni impiegato che si trova in una posizione che è un multiplo di 10 in una tabella IMPIEGATIESTRATTI

**Soluzione:**

```
import java.sql.*;
public class ImpiegatiEstratti {
public static void main(String[] arg){

connection conn = null;
try{
        // carica driver e inizializza connessione
        Class.forName("sun.jdbc.odbc.jbdcObdcDriver");
        conn = DriverManager.GettConnection("jdbc:odbc:impiegati)
        }
try{
        Statement interrogazione = conn.createStatement();
        ResultSet risultato = interrogazione.executeQuery(
                "select *
                 from IMPIEGATI"




        Statement interrogazione1 = conn.createStatement();
        ResultSet risultato1 = interrogazione1.executeQuery(
                "create table IMPIEGATIESTRATTI
                        (
                        NomeImpiegato char(20),
                        CognomeImpiegato char(20),
                        dipartimento char(20),
                        stipedio integer,
                        primary key(NomeImpiegato,CognomeImpiegato)
                        )
        i=0;
        While(risultato.next()) {

        if(i=10)
        {
        string NomeImpiegato = risultato.getString("NomeImpiegato");
        string CognomeImpiegato =          risultato.getString
("CognomeImpiegato");
        string Dipartimento= risultato.getString("Dipartimento");
        int Stipendio = risultato.getString("Stipendio");
```

```
        Statement interrogazione2 = conn.createStatement();
        ResultSet risultato2 = interrogazione2.executeQuery(
                "insert into IMPIEGATIESTRATTI
                        values(" NomeImpiegato","
                        CognomeImpiegato","
                        dipartimento","
                        stipedio",
                        )";
        i=0;
}}}}
```

## Esercizio 6.5

Realizzare un programma che accede al contenuto di una tabella

```
Capitolo(Numero, Titolo, Lunghezza)
```

che descrive i capitoli di un libro, con il numero e la dimensione delle pagine. Il programma quindi popola una tabella

```
Indice(Numero, Titolo, NumPagine)
```

in cui si presenta il numero di pagina nel quale inizia il capitolo, supponendo che il capitolo 1 inizia sulla prima pagina e che i capitoli devono iniziare su pagine dispari (eventualmente introducendo una pagina bianca alla fine del capitolo)

### Soluzione:

```
#include<stdlib.h>
main()
{
exec sql begin declare section;
        char Titolo[50];
        int Numero, Lunghezza;
exec sql end declare section;

exec sql declare CapCursore cursor for
        select Numero, Titolo, Lunghezza
        from Capitolo;
exec sql open CapCursore;
exec sql create table Indice
        (
        Numero integer primary key,
        Titolo char(50),
        NumPagine integer
        );
do{
exec sql fetch CapCursore into
        :Numero, :Titolo, :Lunghezza;
if (Lunghezza%2 != 0)
        Lunghezza = Lunghezza + 1; // Pagina Bianca
exec sql insert into Indice
        values(:Numero,:Titolo,:Lunghezza)";
}while(sqlca.sqlcode == 0)

exec sql close cursor CapCursore;
}
```

## Esercizio 6.6

Si supponga di avere le tabelle:

Magazzino(Prodotto, QtaDisp, Soglia, QtaRiordino)
OrdineInCorso(Prodotto, Qta)

Scrivere una procedura SQL che realizza il prelievo dal magazzino accettando 2 parametri, il prodotto `prod` e la quantità da prelevare `QtaPrelievo`. La Procedura deve verificare inizialmente che `QtaPrelievo` sia inferiore al valore di `QtaDisp` per il prodotto indicato. `QtaPrelievo` viene quindi sottratta al valore di `QtaDisp`. A questo punto la procedura verifica se per il prodotto `QtaDisp` risulta minore di `Soglia`, senza che in `OrdineInCorso` compaia già una tupla relativa al prodotto prelevato; se si, viene inserito un nuovo elemento nella tabella `OrdineInCorso`. Con i valori di `Prod` e del corrispondente attributo `QtaRiordino`.

## Soluzione:

```
#include<stdlib.h>
main()
{
exec sql begin declare section;
        char Prodotto, prod;
        int QtaDisp, Soglia, QtaRiordino, Qta, Qta1, i;
exec sql end declare section;

exec sql declare MagazzinoCursore cursor for
        select  Prodotto, QtaDisp, Soglia, QtaRiordino
        from Magazzino;
exec sql open MagazzinoCursore;

prod = sceltaProdotto();
Qta= sceltaQta();
i=0;

do{
exec sql fetch MagazzinoCursore into
        :Prodotto, :QtaDisp, :Soglia, :QtaRiordino;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
        printf("ERRORE – Prodotto non trovato");
        exit(1);
        }
if (QtaDisp < Qta){
        printf("ERRORE – Quantità non disponibile");
        exit(1);
        }
QtaDisp = QtaDisp – Qta;

if (QtaDisp < Soglia){
        Qtariordino = QtaRiordino + QtaDisp – Soglia;
        }
```

```
exec sql update magazzino
         set QtaDisp = :QtaDisp
         set QtaRiordino = :QtaRiordino
         where current of MagazzinoCursore;

exec sql declare OrdineCursore cursor for
         select  Prodotto, Qta
         from OrdineInCorso;
exec sql open OrdineCursore ;

i=0;
do{
exec sql fetch  OrdineCursore into
         :Prodotto, :Qta1;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
         exec sql update OrdineInCorso
                  set Qta = :QtaRiordino;
                  where current of OrdineCursore;
         }
else exec sql insert into OrdineInCorso
         values(:prod,:QtaRiordino)";

}
```

# Capitolo 7

## Esercizio 7.1

Considerate lo schema ER in figura 7.27: lo schema rappresenta varie proprietà di uomini e donne.

- Correggete lo schema tenendo conto delle proprietà fondamentali delle generalizzazioni.
- Lo schema rappresenta solo le lavoratrici donne; modificare lo schema rappresentando ora tutti i lavoratori, uomini e donne.
- Tra le proprietà delle città, l'attributo `Regione` può essere visto anche come un attributo del concetto PROVINCIA. Ristrutturare lo schema in tal senso.

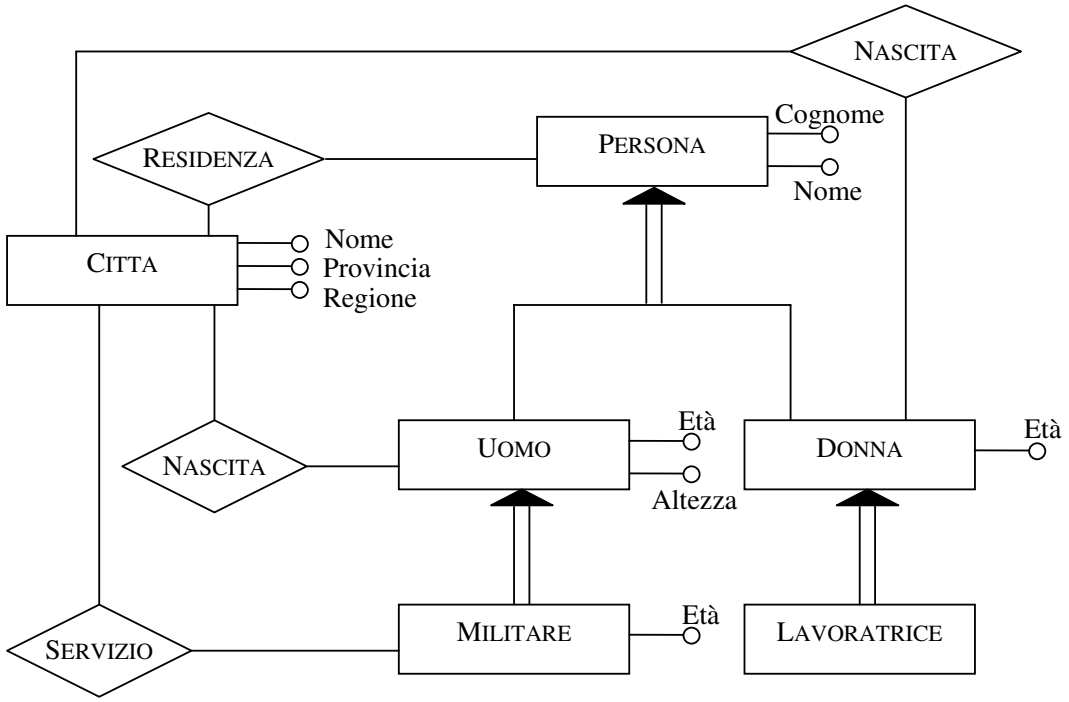
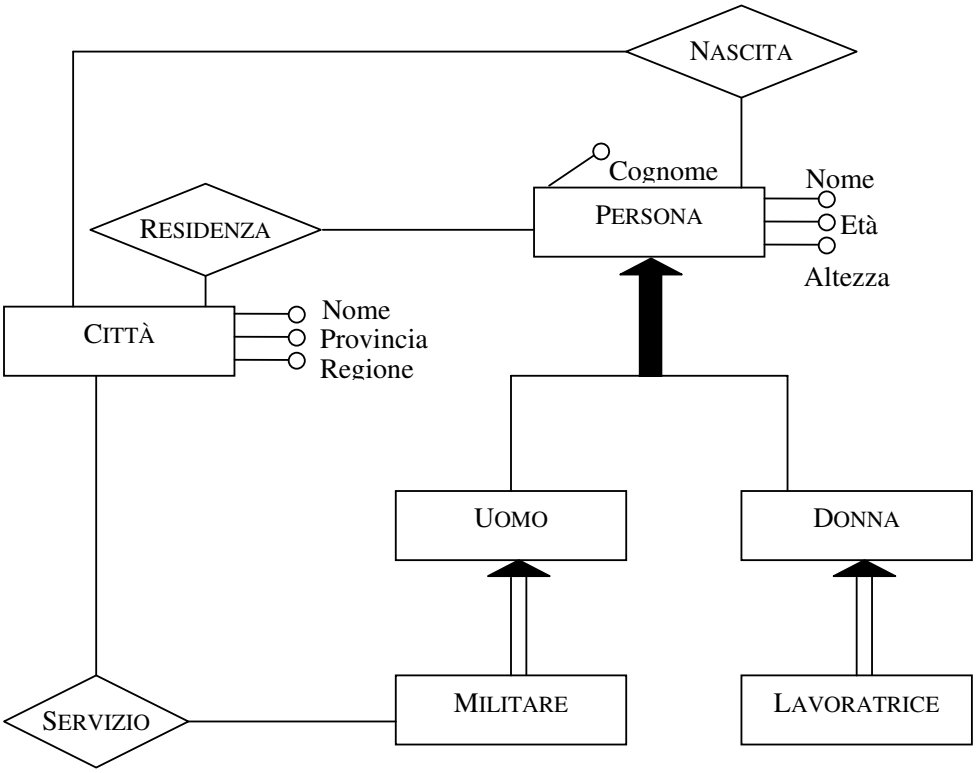


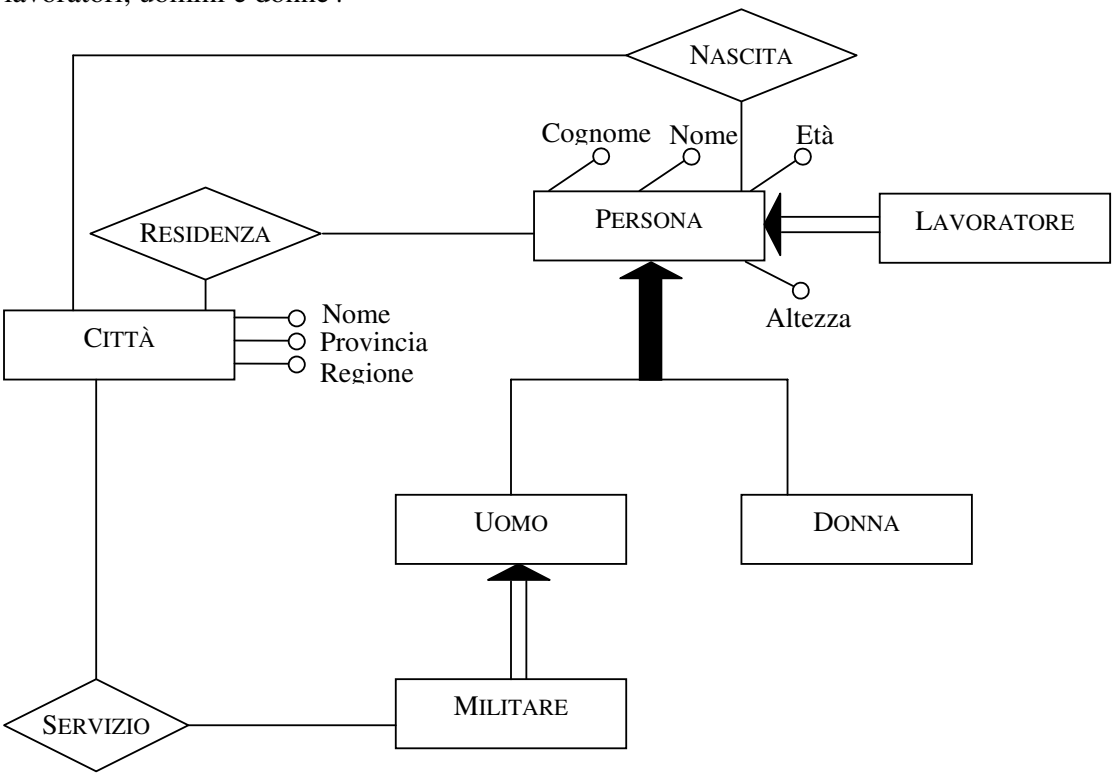

**Figura 7.27** Schema E-R per l'esercizio 7.1

**Soluzione:**

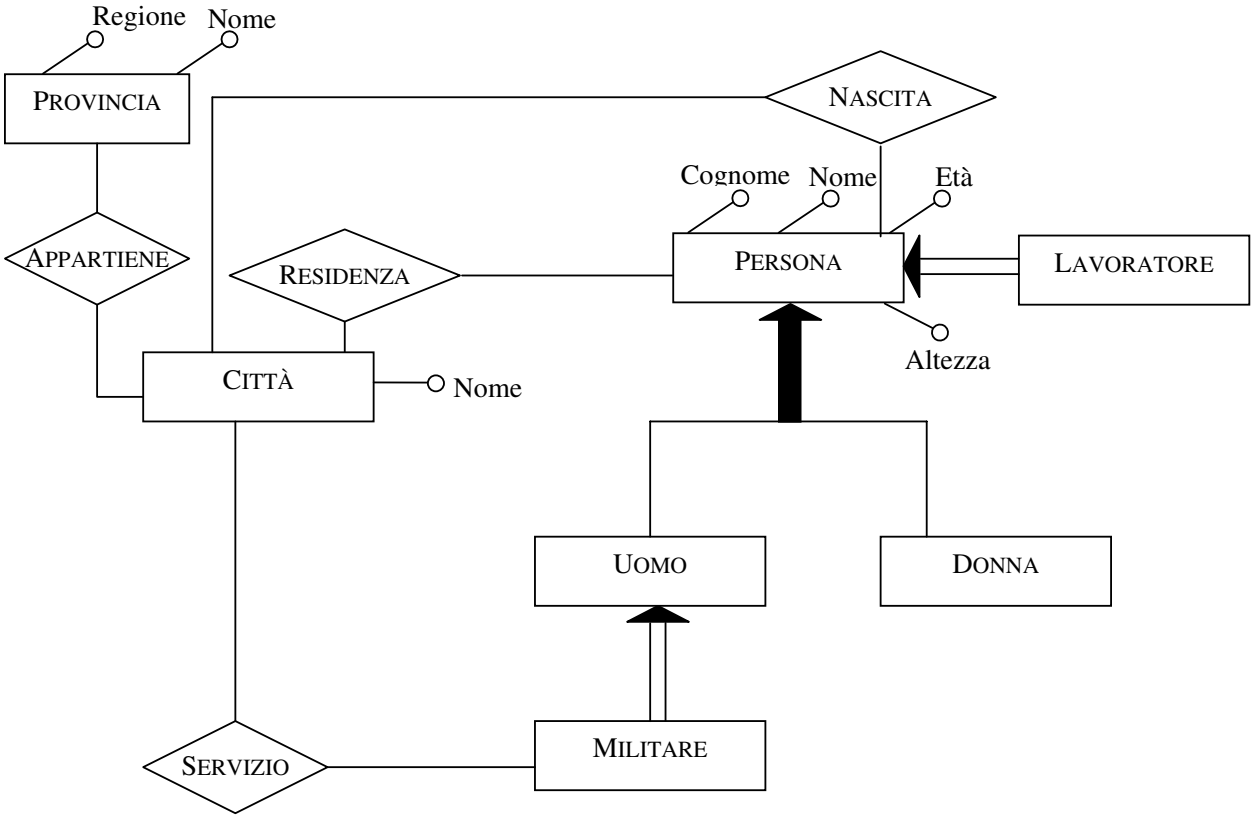
1) Correggete lo schema tenendo conto delle proprietà fondamentali delle generalizzazioni




2) Lo schema rappresenta solo le lavoratrici donne; modificare lo schema rappresentando ora tutti i lavoratori, uomini e donne .



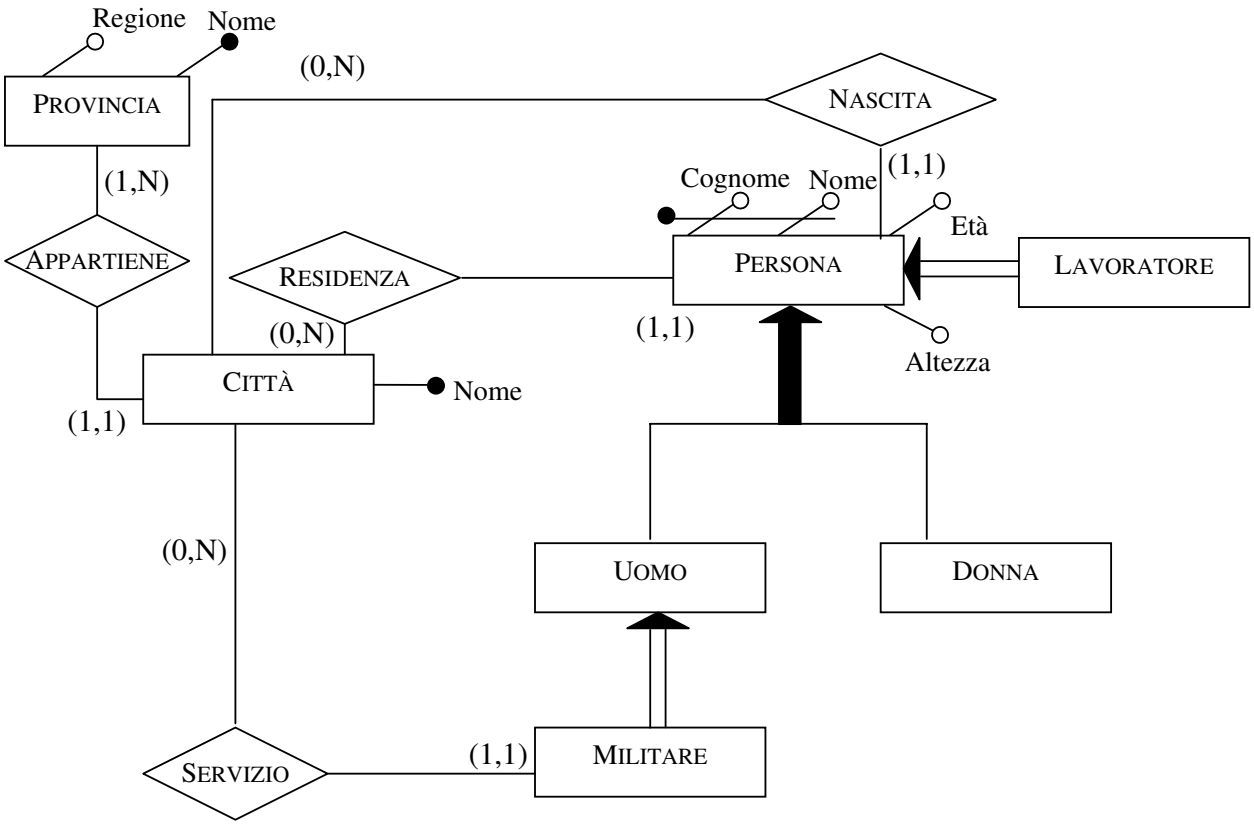
3) Tra le proprietà delle città, l'attributo `Regione` può essere visto anche come un attributo del concetto PROVINCIA. Ristrutturare lo schema in tal senso.



## Esercizio 7.2

Aggiungere le cardinalità minime e massime allo schema prodotto nell'esercizio 7.1 e gli identificatori principali. Dire se esistono dei vincoli di integrità sullo schema che non possono essere espressi con il modello Entità-Relazione.

**Soluzione:**




I vincoli che non possono essere espressi nello schema Entità-Relazione sono:

- L'età degli uomini che svolgono il servizio militare deve essere superiore ai 18 anni.
- I lavoratori devono avere almeno 18 anni.
- L'altezza degli uomini che svolgono il servizio militare deve essere almeno uguale ad un minimo richiesto.

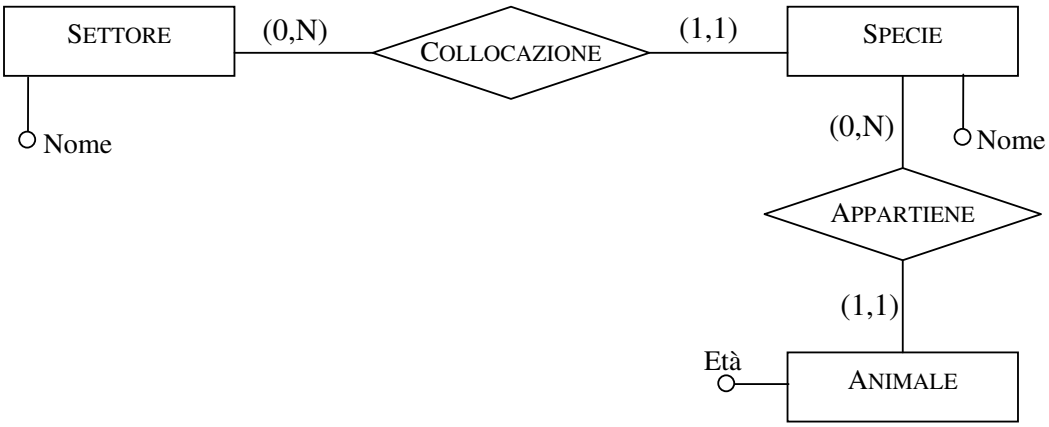
## Esercizio 7.3

Rappresentare le seguenti realtà utilizzando i costrutti del modello Entità-Relazione e introducendo solo le informazioni specificate.

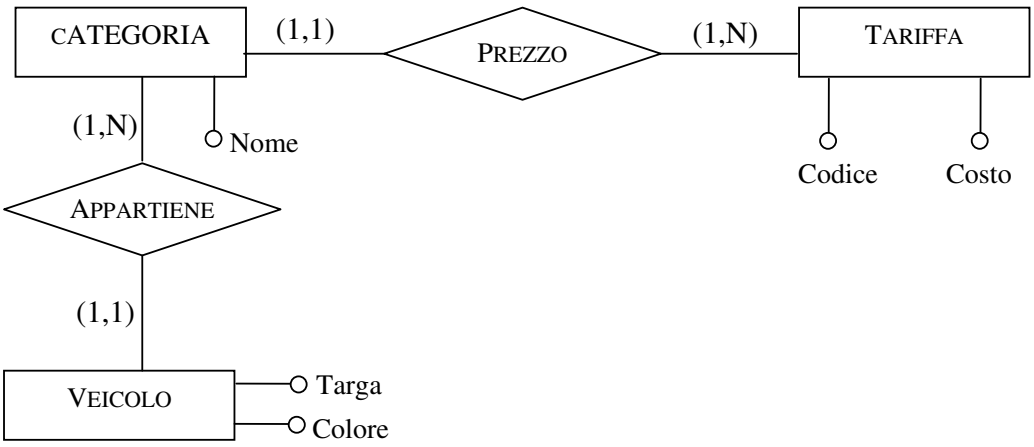
- In un giardino zoologico ci sono degli animali appartenenti a una specie e aventi una certa età; ogni specie è localizzata in un settore (avente un nome) dello zoo.
- Una agenzia di noleggio di autovetture ha un parco macchine ognuna delle quali ha una targa, un colore e fa parte di una categoria; per ogni categoria c'è una tariffa di noleggio.
- Una casa discografica produce dischi aventi un codice ed un titolo; ogni disco è inciso da uno o più cantanti, ognuno dei quali ha un nome, un indirizzo e, qualcuno, un nome d'arte.

**Soluzione:**

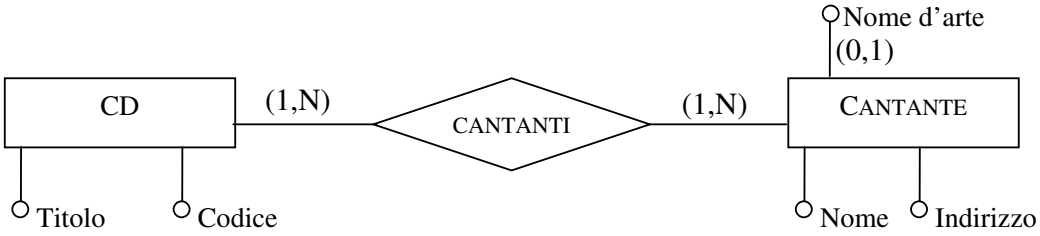
1) In un giardino zoologico ci sono degli animali appartenenti a una specie e aventi una certa età; ogni specie è localizzata in un settore (avente un nome) dello zoo.




2) Una agenzia di noleggio di autovetture ha un parco macchine ognuna delle quali ha una targa, un colore e fa parte di una categoria; per ogni categoria c'è una tariffa di noleggio.



3) Una casa discografica produce dischi aventi un codice ed un titolo; ogni disco è inciso da uno o più cantanti, ognuno dei quali ha un nome, un indirizzo e, qualcuno, un nome d'arte.

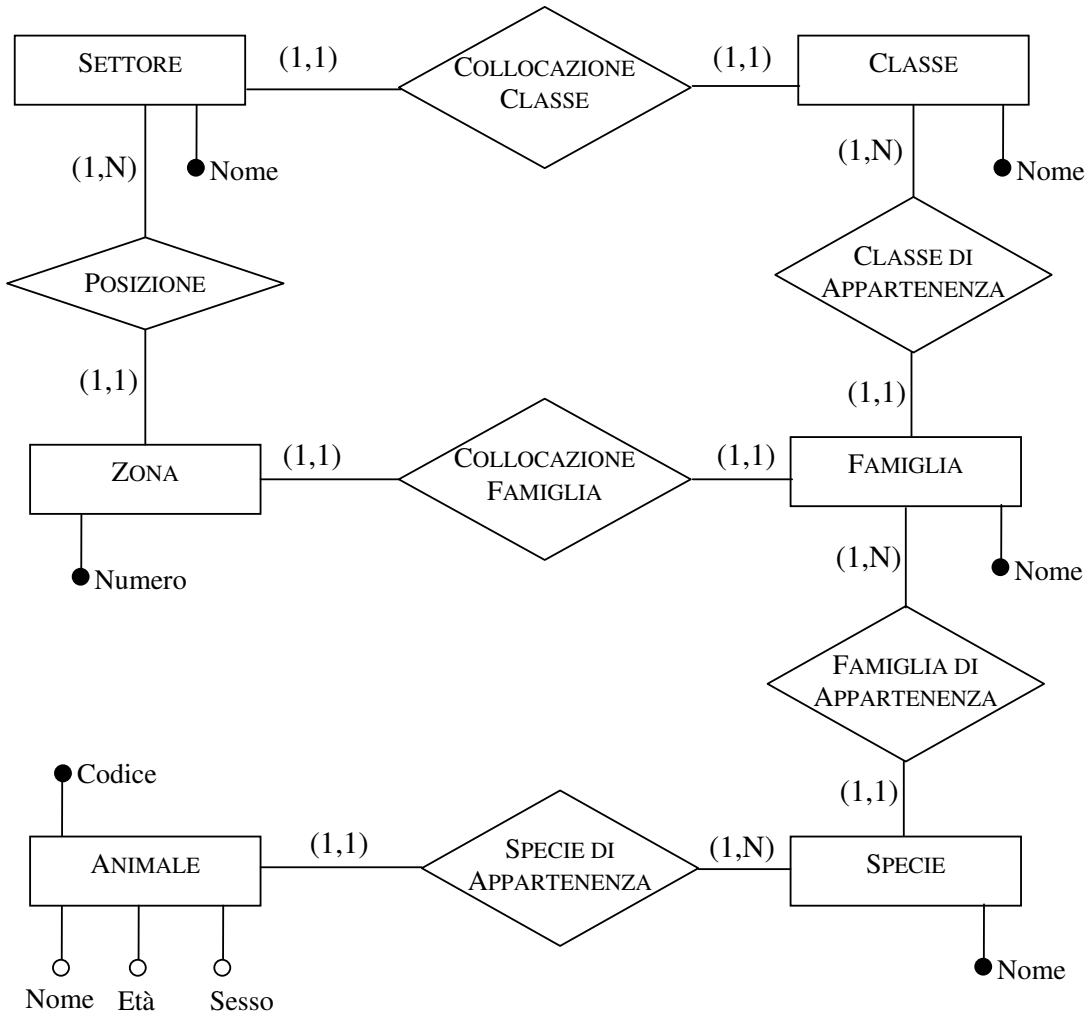



## Esercizio 7.4

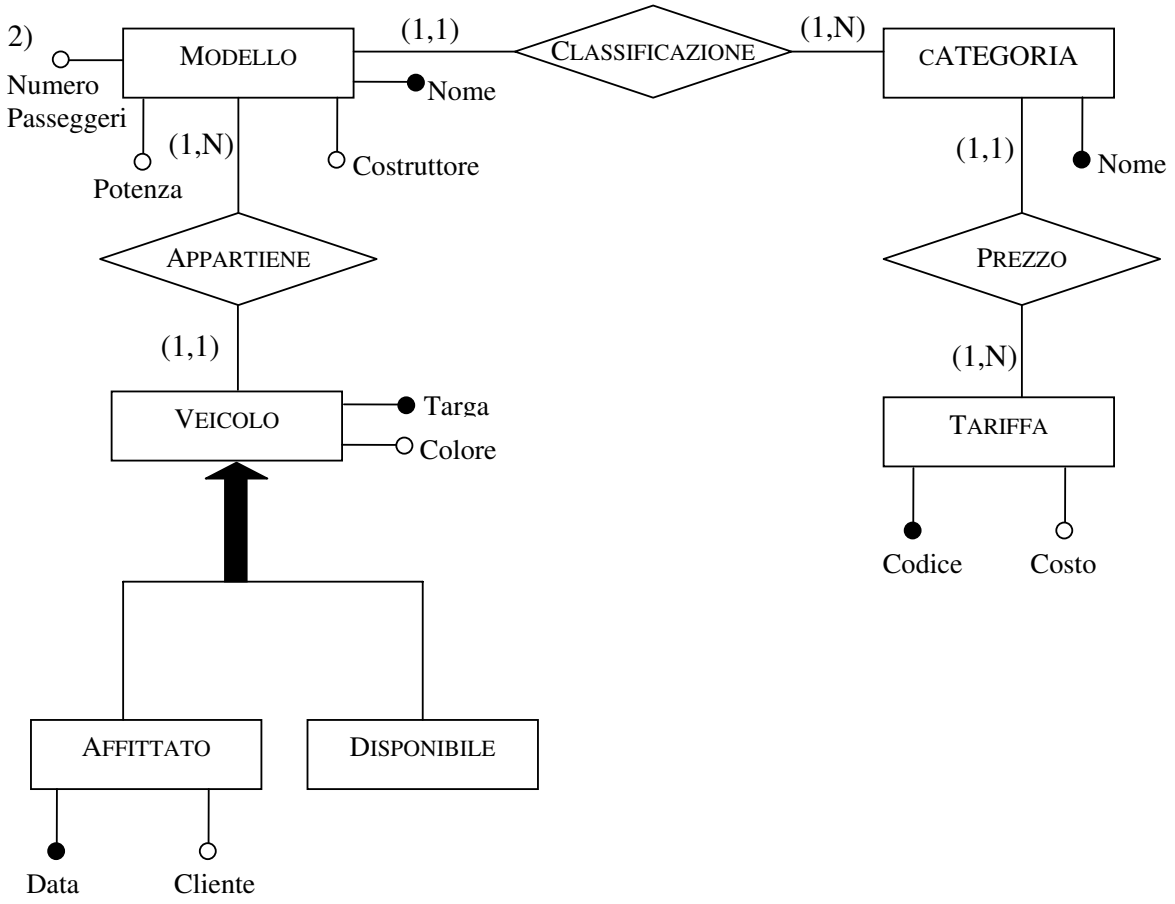
Completare i frammenti di schema prodotti nell'esercizio precedente con ulteriori informazioni, basandosi sulle proprie conoscenze o facendo delle ipotesi sulle rispettive realtà di interesse.

**Soluzione:**

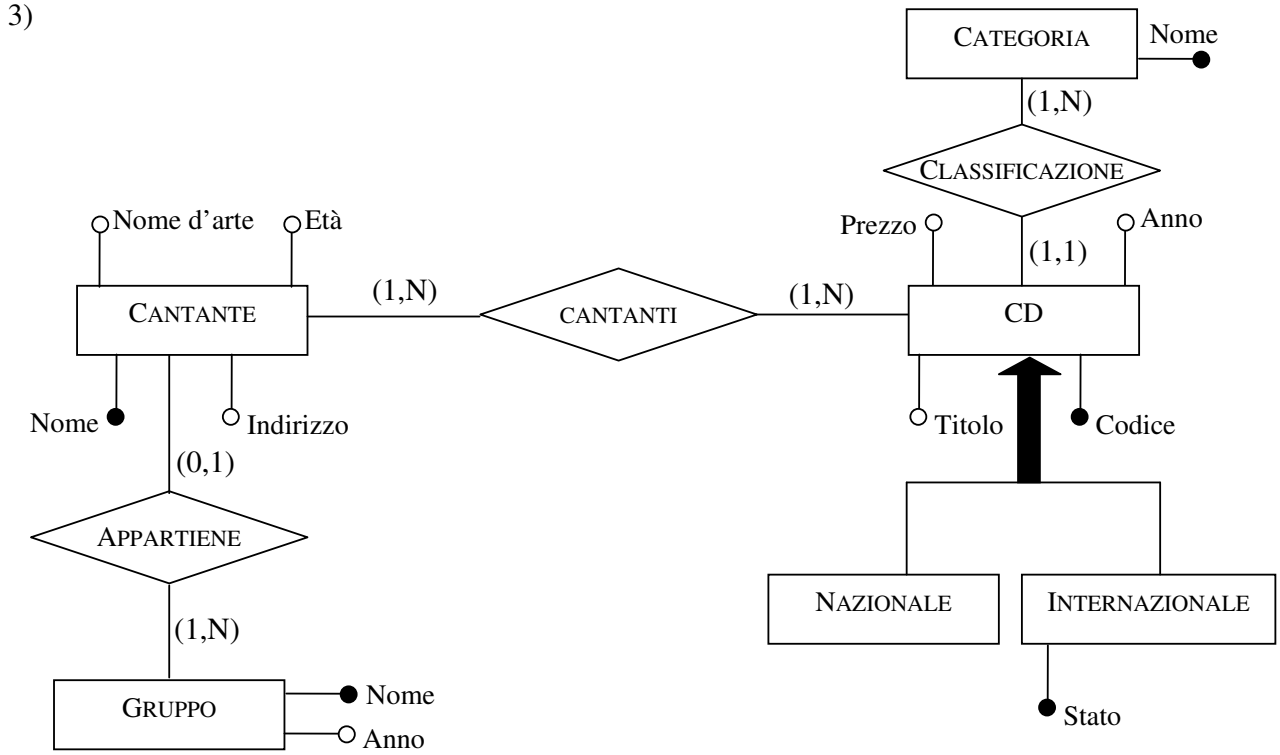
1)



2)




3)



## Esercizio 7.5

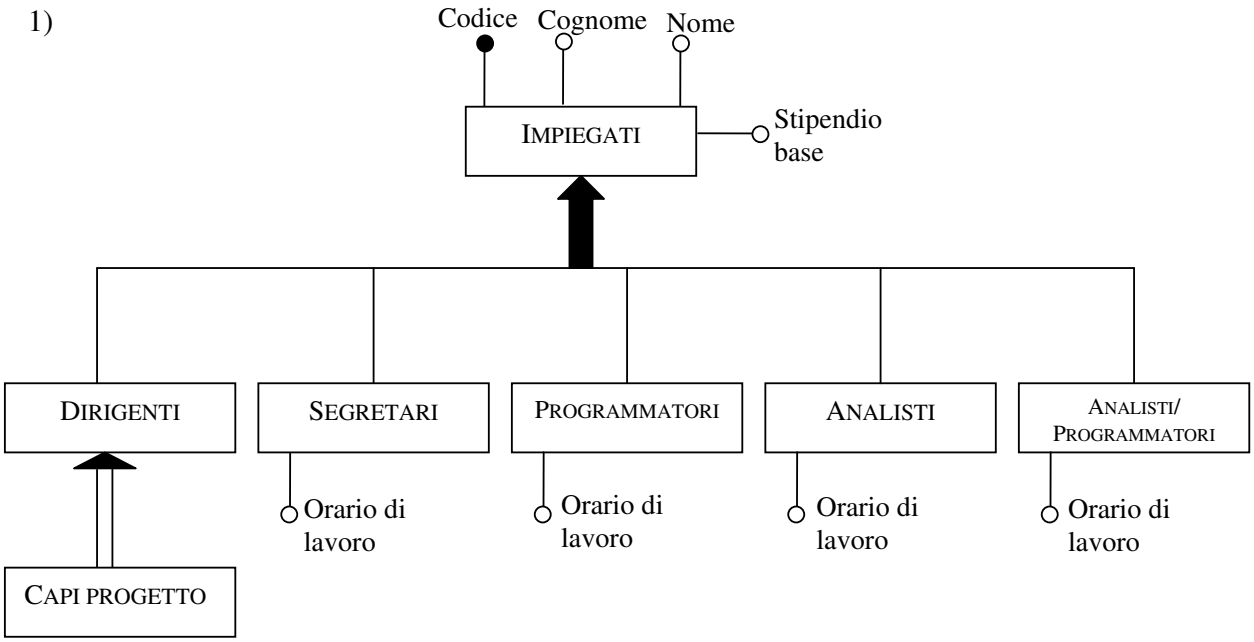
Rappresentare le seguenti classi di oggetti facendo uso, dove opportuno, del costrutto di generalizzazione del modello Entità-Relazione. Indicare nei vari casi, gli attributi delle varie entità e il tipo di generalizzazione, risolvendo i casi di sovrapposizione.

- Gli impiegati di una azienda si dividono in dirigenti, programmatori, analisti, capi progetto e segretari. Ci sono analisti che sono anche programmatori. I capi progetto devono essere dirigenti. Gli impiegati hanno un codice, un nome e un cognome. Ogni categoria di impiegato ha un proprio stipendio base. Ogni impiegato, tranne i dirigenti, ha un orario di lavoro.
- Una compagnia aerea offre voli che possiedono un numero che identifica la tratta (per esempio, Roma-Milano), una data (25 marzo 2001), un orario di partenza (ore 8:00) e uno di arrivo (ore 9:00), un aeroporto di partenza e uno di destinazione. Ci sono voli nazionali e internazionali. I voli internazionali possono avere uno o più scali. Dei voli passati è di interesse l'orario reale di partenza e di arrivo (per esempio, con riferimento al volo suddetto, ore 8:05 e 9:07), di quelli futuri è di interesse il numero di posti disponibili.
- Una casa automobilistica produce veicoli che possono essere automobili, motocicli, camion e trattori. I veicoli sono identificati da un numero di telaio e hanno un nome (per esempio, Punto), una cilindrata e un colore. Le automobili si suddividono in utilitarie (lunghezza sotto i due metri e mezzo) e familiari (lunghezza sopra i due metri e mezzo). Vengono anche classificate in base alla cilindrata: piccola (fino a 1200 cc), media (da 1200 cc a 2000cc) e grossa cilindrata(sopra i 2000 cc). I motocicli si suddividono in motorini (cilindrata sotto i 125 cc) e moto (cilindrata sopra i 125 cc). I camion hanno un peso e possono avere un rimorchio.

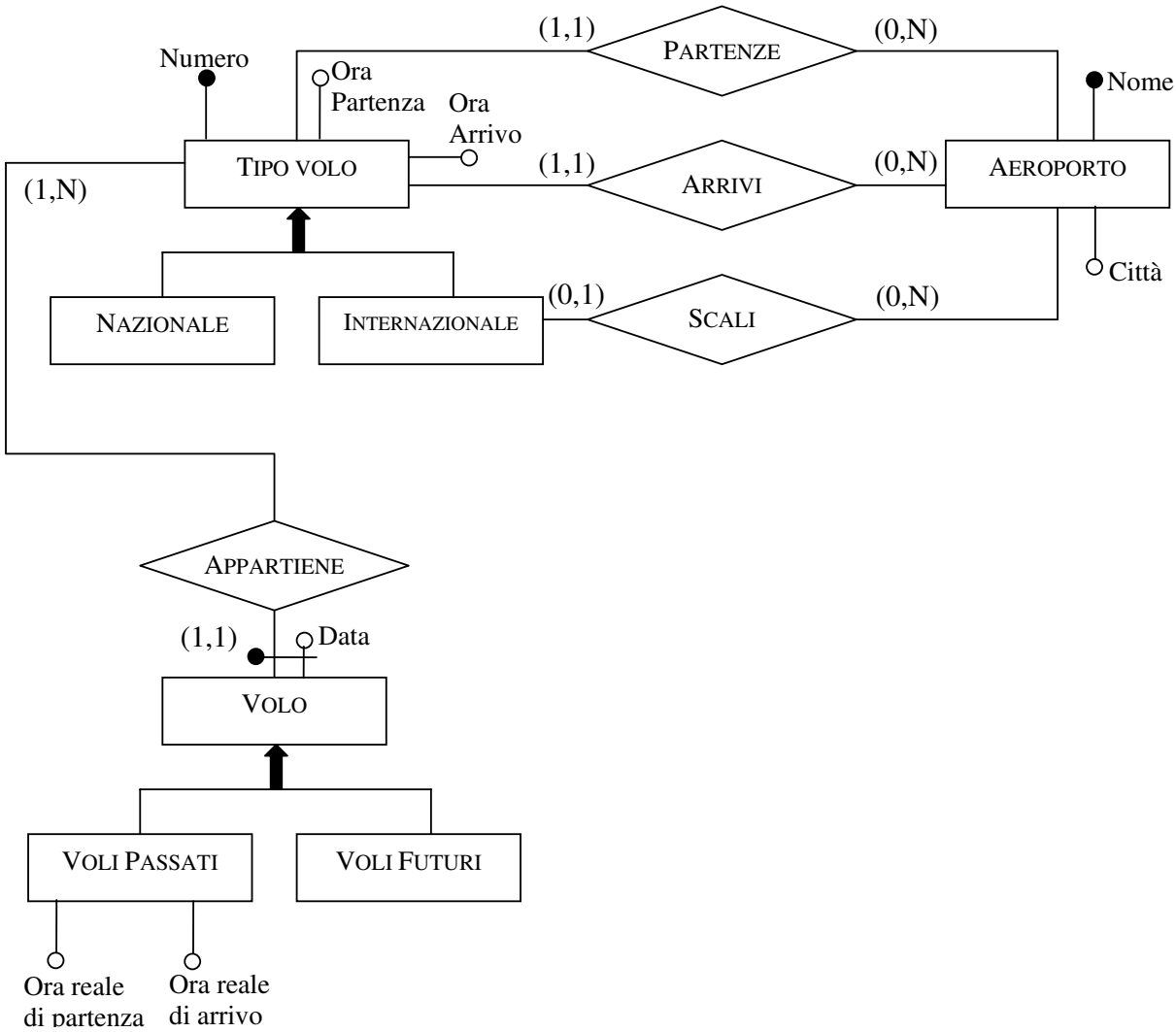
**Soluzione:**

Tutte le generalizzazioni sono esclusive

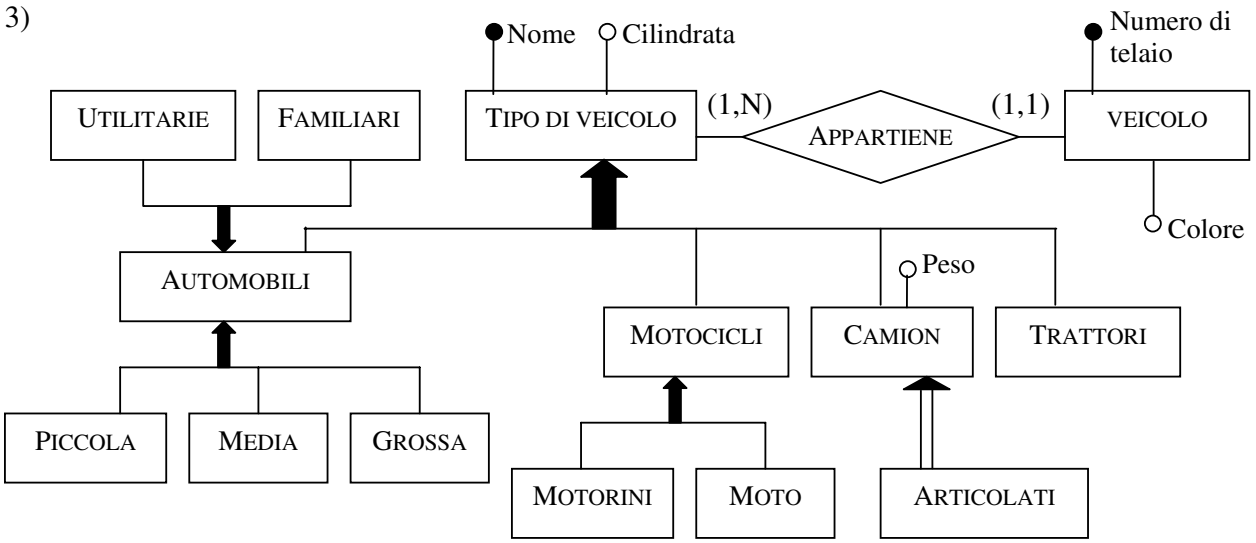
1)



2)




3)



## Esercizio 7.6

Si consideri lo schema Entità-Relazione in figura 7.28. Descrivere le informazioni che esso rappresenta utilizzando il linguaggio naturale.

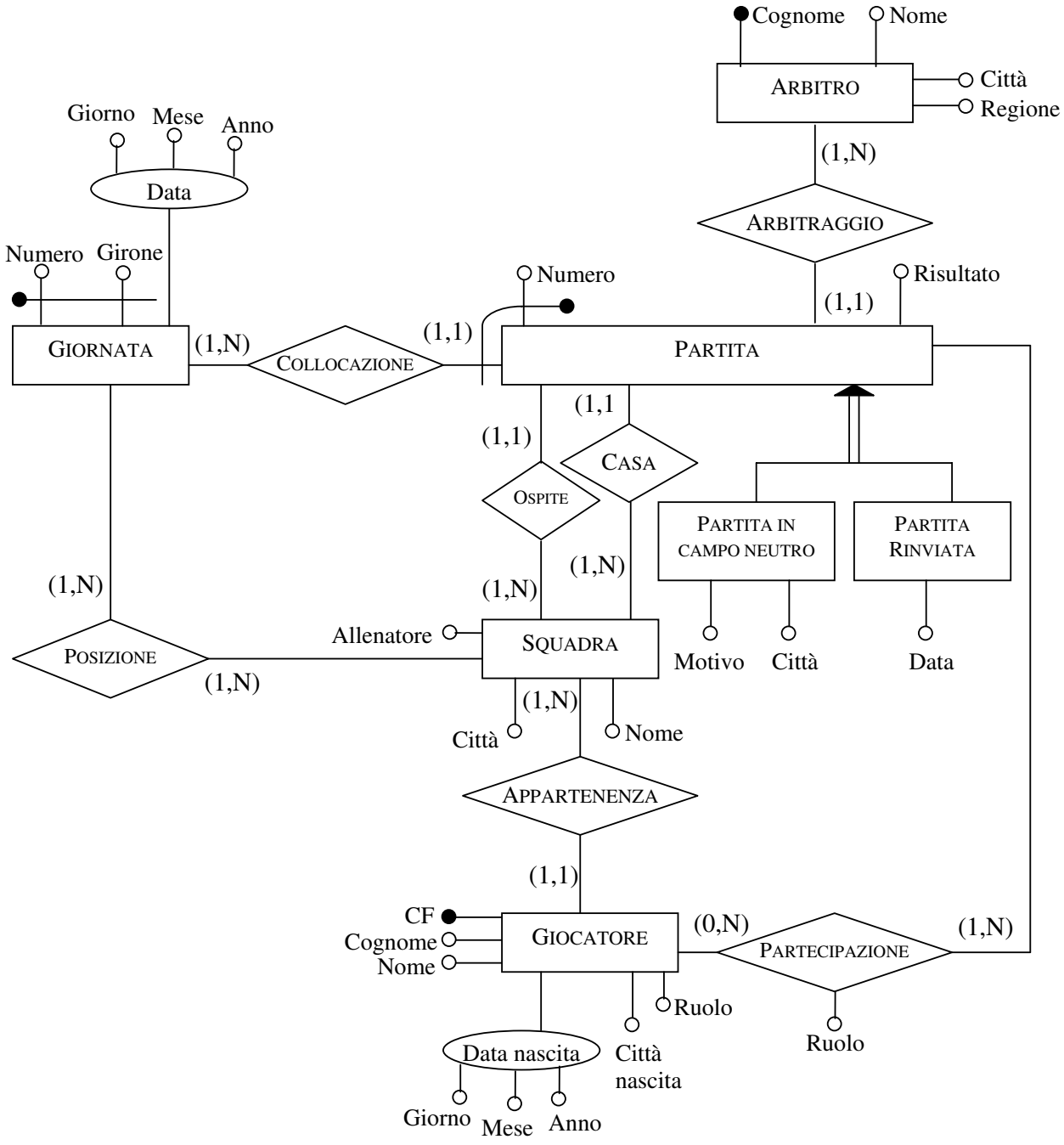



**Figura 7.28** Schema E-R per l'esercizio 7.6

**Soluzione:**

Lo schema contiene le informazioni di un campionato (per esempio un campionato di calcio).
L'entità SQUADRA rappresenta tutte le squadre del campionato, indicando per ognuna di esse il nome, la città e il nome dell'allenatore. L'entità GIOCATORE rappresenta i giocatori delle squadre: ogni giocatore ha un contratto con una sola squadra e ogni squadra ha più giocatori.
I giocatori sono identificati dal loro Codice Fiscale (CF) e per ognuno di essi è indicato il nome, il cognome, il ruolo nella squadra, la città di nascita e la data di nascita.
Lo schema contiene anche informazioni sulle partite del campionato con l'entità PARTITA. Una partita è identificata con un numero (che deve essere differente per tutte le partite dello stesso giorno) e con un riferimento al giorno (attraverso la relazione COLLOCAZIONE e l'entità GIORNATA).
Le relazioni CASA e OSPITE rappresentano le due squadre che giocano la partita: per ogni partita è indicato il risultato e l'arbitro, con la relazione ARBITRAGGIO tra PARTITA e ARBITRO; questa entità rappresenta tutti gli arbitri del campionato e per ognuno di essi è indicato il Nome, il Cognome, la Città e la Regione. Un arbitro è rappresentato solo se ha arbitrato almeno una partita.
Una partita può essere giocata su campo neutrale o può essere rinviata ad un'altra data (ma questi due eventi non sono ammessi contemporaneamente nello schema).
La relazione Partecipazione rappresenta il fatto che un giocatore abbia giocato in una partita, la sua posizione (che può essere diversa dalla sua solita). Lo schema non esprime la condizione che i giocatori che giocano una partita devono avere un contratto con una delle due squadre.
L'entità GIORNATA rappresenta la giornata del campionato. Sono identificate con Numero e Girone.
La relazione Posizione dà il punteggio di ogni squadra in ogni giornata.

## Esercizio 7.7

Tradurre in regole aziendali le seguenti proprietà sui dati lo schema di figura 7.28.

- Non ci possono essere più di 5 giocatori in una squadra che giocano nello stesso ruolo.
- Una squadra guadagna 3 punti se vince, 1 se pareggia e 0 se perde.
- Se una squadra gioca in casa una partita, allora è ospite nella partita successiva

Produrre quindi una documentazione completa per tale schema.

**Soluzione:**

RA1) In una squadra, il numero di giocatori con la stessa posizione **DEVE ESSERE** inferiore a cinque.
RA2) Il numero di punti guadagnato da una squadra in una partita **È OTTENUTO** sottraendo il punteggio della giornata della partita dal punteggio che aveva nella giornata precedente.
RA3) Il numero di punti guadagnato da una squadra che vince una partita **DEVE ESSERE** 3.
RA4) Il numero di punti guadagnato da una squadra che pareggia una partita **DEVE ESSERE** 1.
RA5) Il numero di punti guadagnato da una squadra che perde una partita **DEVE ESSERE** 0.
RA6) La prossima partita di una squadra **È OTTENUTA** ricercando, tra tutti gli incontri della prossima giornata, l'unico che coinvolge la squadra.
RA6) La prossima partita di una squadra che ha giocato come ospite **DEVE ESSERE** in casa.

Si osserva, incidentalmente, che l'ultima regola non può essere rispettata da tornei "all'italiana" con n (pari) squadre in cui ogni squadra incontra tutte le altre squadre in n-1 turni di campionato, a meno che non vi siano solo 2 squadre. Si invita il lettore a dimostrare questa impossibilità.

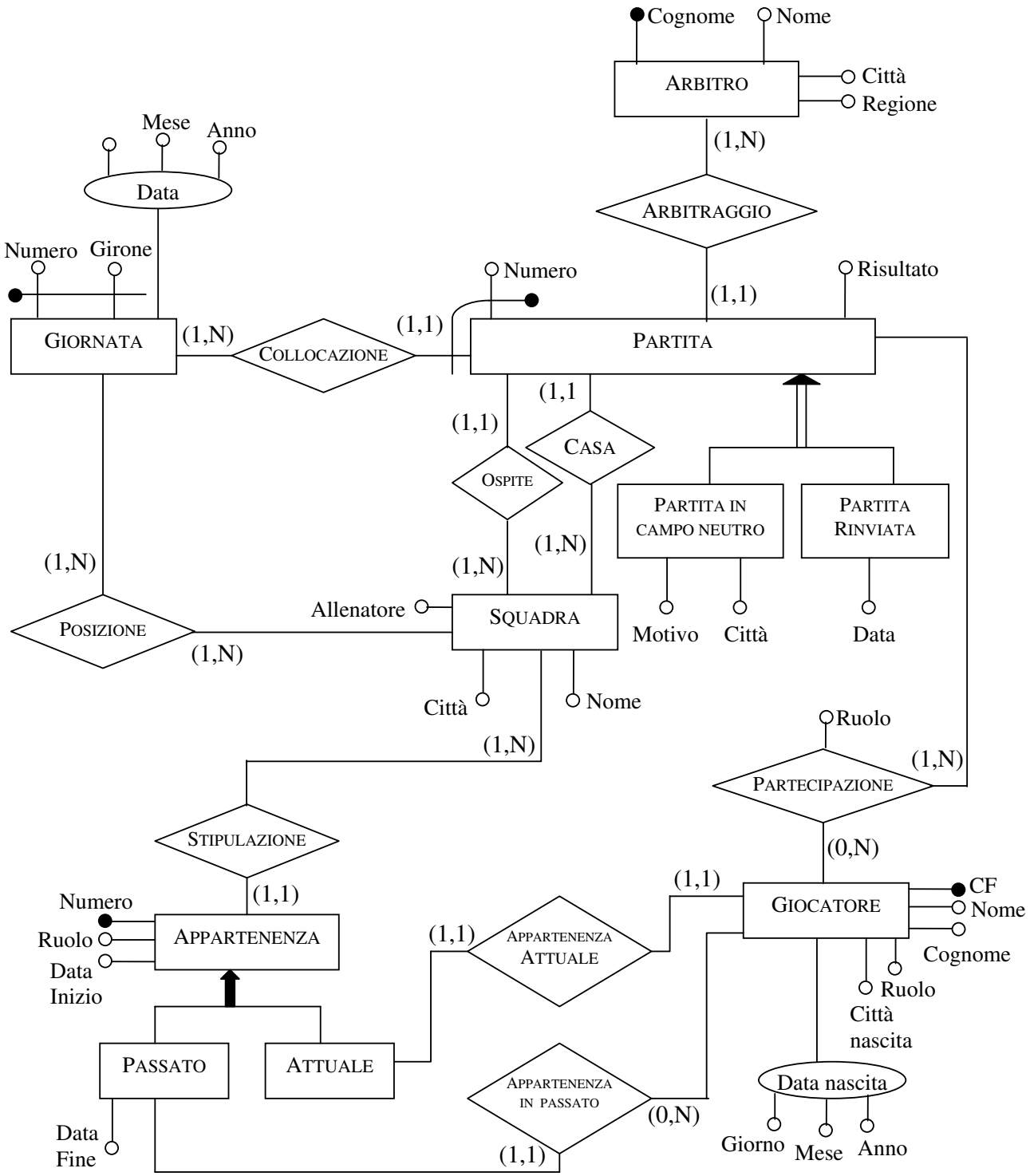
**DIZIONARIO DEI DATI**

| Entità | Descrizione | Attributi | Identificatore |
|---|---|---|---|
| Squadra | Una squadra che gioca nel campionato | Nome, Città, Allenatore | Nome |
| Giocatore | Giocatore che gioca in una squadra | Codice Fiscale, Cognome, Nome, Ruolo, Città di nascita, Data di nascita (Giorno, Mese, Anno) | Codice Fiscale |
| Partita | Una partita giocata durante il campionato | Numero, Risultato | Numero + Giornata (identificatore esterno) |
| Partita in campo neutro | Una partita giocata su un campo neutrale | Motivo, Città, Numero, Risultato | Numero + Giornata (identificatore esterno) |
| Partita rinviata | Una partita che è stata rinviata ad un'altra data | Data, Numero, Risultato | Numero + Giornata (identificatore esterno) |
| Giornata | Una giornata del campionato | Numero, Girone, Data (Giorno, Mese, Anno) | Numero, Girone |
| Arbitro | Un arbitro del campionato | Nome, Cognome, Città, Regione | Cognome |
| Arbitraggio | Associa una partita con il rispettivo arbitro | Arbitro, Partita | |
| Collocazione | Associa una partita con la rispettiva giornata di campionato. È necessaria per identificare una partita | Partita, Giornata | |
| Casa | Associa una partita con una squadra: rappresenta la squadra che gioca la partita in casa | Partita, Squadra | |
| Ospite | Associa una partita con una squadra: rappresenta la squadra che gioca la partita fuori casa | Partita, Squadra | |
| Posizione | Associa una giornata con una squadra: rappresenta (dando il punteggio) la posizione della squadra dopo ogni giornata | Giornata, Squadra | Punteggio |
| Appartenenza | Associa una squadra con un giocatore: rappresenta il fatto che un giocatore gioca attualmente con una squadra | Squadra, Giocatore | |
| Partecipazione | Associa un giocatore con una partita: rappresenta il fatto che un giocatore ha giocato in una partita. Può aver giocato in una posizione diversa dalla sua abitale. | Partita, Giocatore | Posizione |

## Esercizio 7.8

Modificare lo schema Entità-Relazione in figura 7.28 in maniera da descrivere anche i rapporti passati tra giocatori e squadre con dati di inizio e fine del rapporto e il ruolo principale ricoperto da ogni giocatore in ogni squadra. È possibile che un giocatore abbia diversi rapporti con la stessa squadra in periodi diversi. Per i rapporti in corso si vuole conoscere la data di inizio.

**Soluzione:**



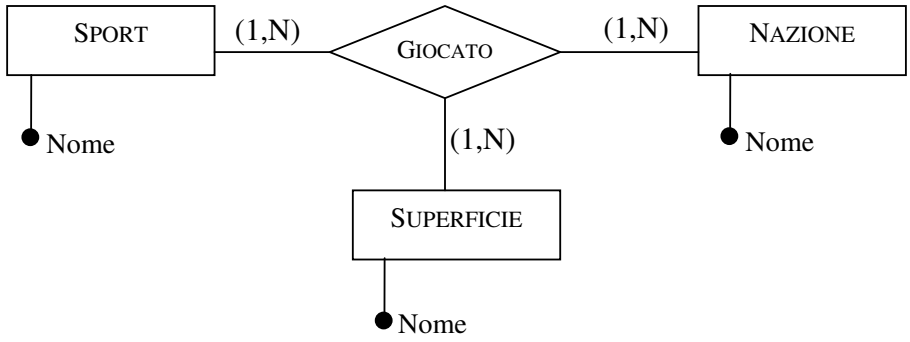
## Esercizio 7.9

In ciascuno dei seguenti casi, si fa riferimento a due o più entità definite in uno schema Entità-Relazione e a un concetto che le coinvolge. Specificare i relativi frammenti di schema, definendo i costrutti (una o più relazioni e, se necessario, ulteriori entità con il relativo identificatore) necessari a rappresentare il concetto, mantenendo le entità indicate e introducendo solo gli attributi richiesti esplicitamente.

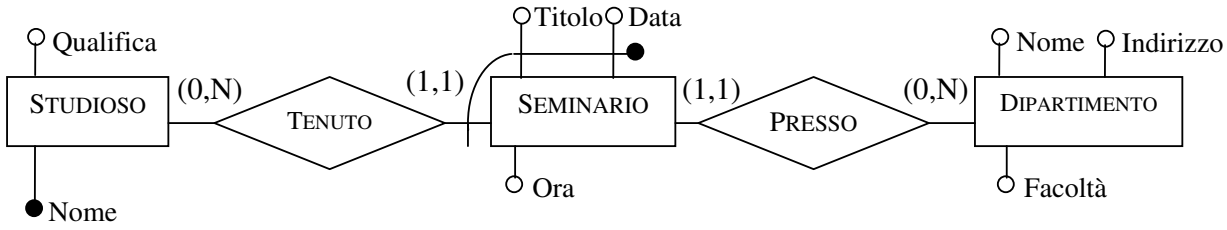
- Entità: Sport, nazione e superficie. Concetto: il fatto che uno sport si pratichi in una certa nazione su una certa superficie (ad esempio, il tennis si gioca sull'erba in Inghilterra e in Australia, sulla terra rossa in Italia e in Francia, sul sintetico in USA, Italia e Francia; il calcio sull'erba in Italia, sul sintetico e sull'erba in USA, sull'erba in Inghilterra).
- Entità: studioso e dipartimento. Concetto: il fatto che lo studioso abbia tenuto seminari presso il dipartimento. Per ogni seminario è necessario rappresentare data, ora e titolo, con il vincolo che uno studioso non possa tenere più seminari nello stesso giorno.
- Entità: professionista e azienda. Concetto: il fatto che il professionista abbia svolto consulenze per l'azienda. È necessario rappresentare il numero di consulenze effettuate dal professionista per ciascuna azienda, con il relativo costo totale.

**Soluzione:**

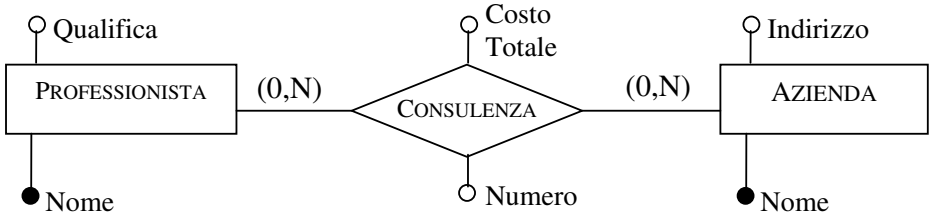
1)




2)




3)



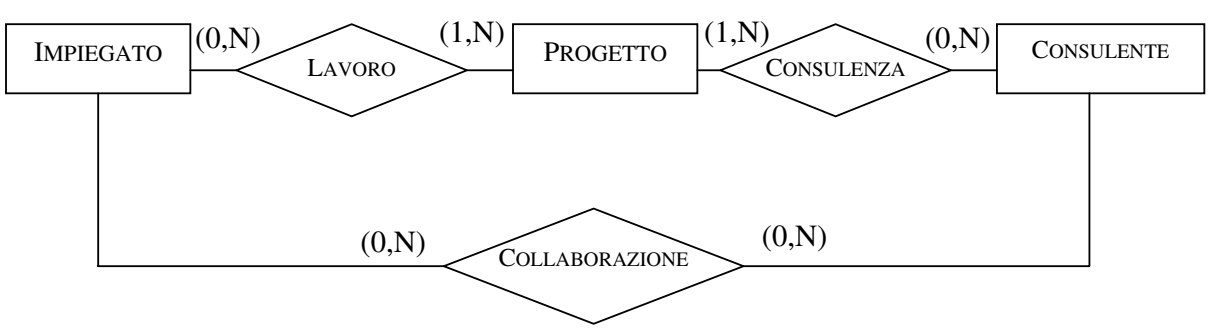
## Esercizio 7.10

Si consideri una relazione ternaria che coinvolge le seguenti entità: IMPIEGATO, PROGETTO e CONSULENTE. Indicare in quali dei seguenti casi (e, in caso affermativo, come) è opportuno sostituire a tale relazione due (o tre) relazioni binarie.

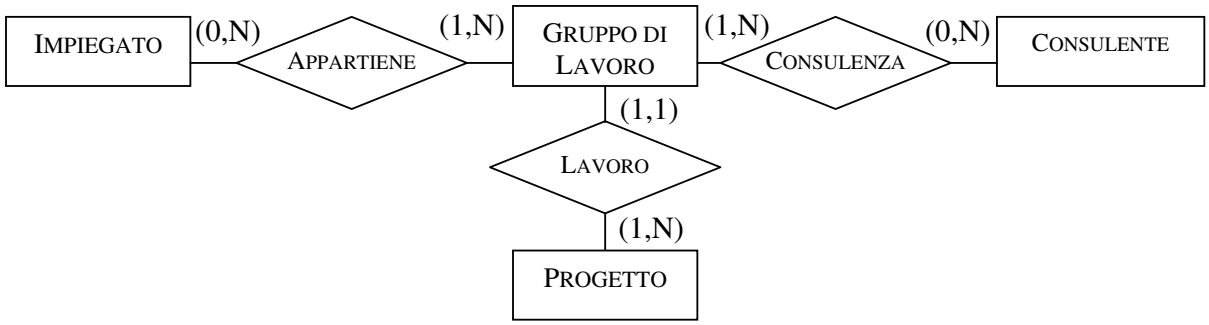
1. Ogni impiegato è coinvolto in zero o più progetti e interagisce con zero o più consulenti. Ogni consulente è coinvolto in zero o più progetti e interagisce con zero o più impiegati. Ogni progetto coinvolge uno o più impiegati e uno o più consulenti (che possono non interagire tra di loro). Un impiegato e un consulente collaborano nell'ambito di un progetto se e solo se essi collaborano fra loro e sono entrambi coinvolti nello stesso progetto.
2. Ogni impiegato è coinvolto in zero o più progetti, in ciascuno dei quali interagisce con uno o più consulenti (che possono essere diversi da progetto a progetto e che possono in generale essere un sottoinsieme dei consulenti coinvolti nel progetto). Ogni consulente è coinvolto in zero o più progetti, in ciascuno dei quali interagisce con uno o più impiegati (che possono essere diversi da progetto a progetto e che possono in generale essere un sottoinsieme degli impiegati coinvolti nel progetto). Ogni progetto coinvolge una o più coppie impiegato-consulente.
3. Ogni impiegato è coinvolto in zero o più progetti. Ogni consulente è coinvolto in zero o più progetti. Ogni progetto coinvolge uno o più impiegati e uno o più consulenti. Un impiegato e un consulente interagiscono se e solo se esiste almeno un progetto in cui siano entrambi coinvolti.

**Soluzione:**

1)




2)




3)

# Capitolo 8

## Esercizio 8.1

Si desidera automatizzare il sistema di prestiti di una biblioteca.
Le specifiche del sistema, acquisite attraverso un'intervista con il bibliotecario, sono quelle riportate in figura 8.15. Analizzare tali specifiche, filtrare le ambiguità presenti e poi raggrupparle in modo omogeneo. Prestare particolare attenzione alla differenza esistente tra il concetto di *libro* e di *copia* di libro. Individuare i collegamenti esistenti tra i vari gruppi di specifiche così ottenuti.

| **Biblioteche** |
|---|
| *I lettori che frequentano la biblioteca hanno una tessera su cui è scritto il nome e l'indirizzo ed effettuano richieste di prestito per i libri che sono catalogati nella biblioteca. I libri hanno un titolo, una lista di autori e possono esistere in diverse copie. Tutti i libri contenuti nella biblioteca sono identificati da un codice. A seguito di una richiesta viene dapprima consultato l'archivio dei libri disponibili (cioè non in prestito). Se il libro è disponibile, si procede alla ricerca del volume negli scaffali; il testo viene poi classificato come in prestito. Acquisito il volume, viene consegnato al lettore, che procede alla consultazione. Terminata la consultazione, il libro viene restituito, reinserito in biblioteca e nuovamente classificato come disponibile. Per un prestito si tiene nota degli orari e delle date di acquisizione e di riconsegna.* |

**Figura 8.15** Specifiche per l'esercizio 8.1

**Soluzione:**

| Termine | Descrizione | Sinonimo | Collegamenti |
|---|---|---|---|
| Lettore | Una persona che prende in prestito libri dalla biblioteca | Utente | Copia, Prestito |
| Libro | Tipo di libro presente in biblioteca. La biblioteca ha una o più copie di uno stesso libro. | | Copia |
| Copia | Ogni copia di un libro presente in biblioteca. Può essere prestato a un lettore. | Libro, Testo, Volume | Libro, Lettore, Prestito |
| Prestito | Un prestito fatto a un lettore: ogni prestito si riferisce ad una copia di un libro. | | Lettore, Copia |

FRASI RELATIVE AI LETTORI:

I lettori che frequentano la biblioteca hanno una tessera su cui è scritto il nome e l'indirizzo ed effettuano richieste di prestito per i libri che sono catalogati nella biblioteca.

FRASI RELATIVE AI LIBRI:

I libri hanno un titolo, una lista di autori e possono esistere in diverse copie.

FRASI RELATIVE ALLE COPIE:

Tutti i libri contenuti nella biblioteca sono identificati da un codice.
A seguito di una richiesta viene dapprima consultato l'archivio dei libri disponibili (cioè non in prestito).
Se il libro è disponibile, si procede alla ricerca del volume negli scaffali;

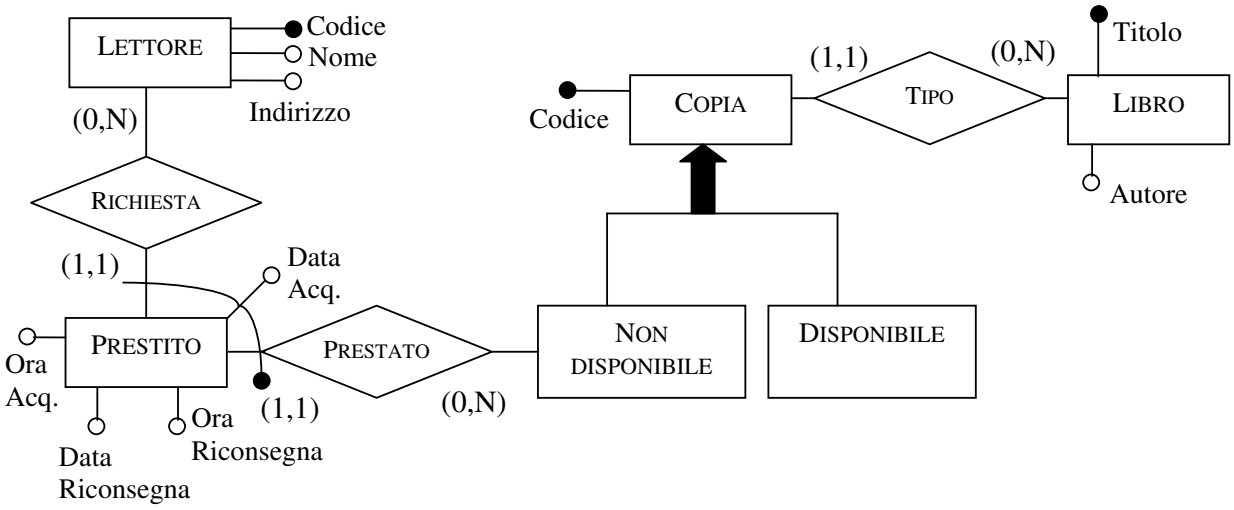
FRASI RELATIVE AI PRESTITI:

Acquisito il volume, viene consegnato al lettore, che procede alla consultazione.
il testo viene poi classificato come in prestito.
Per un prestito si tiene nota degli orari e delle date di acquisizione e di riconsegna.

## Esercizio 8.2

Rappresentare le specifiche dell'esercizio precedente (dopo la fase di riorganizzazione) con uno schema del modello Entità-Relazione.

**Soluzione:**



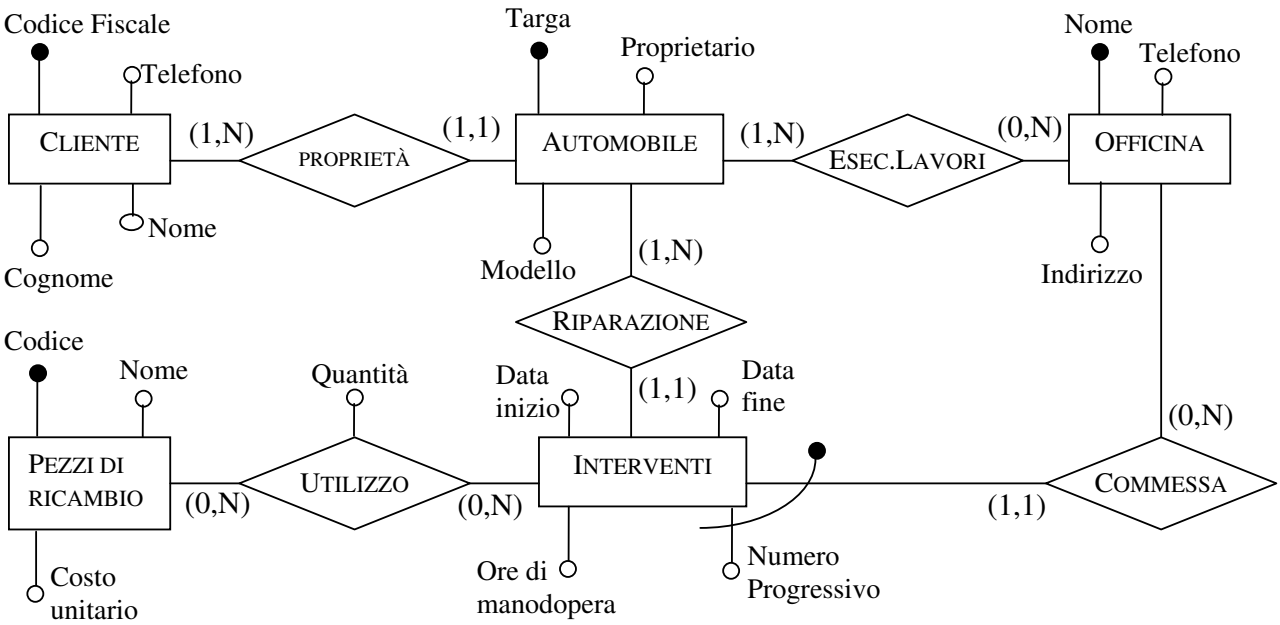
## Esercizio 8.3

Definire uno schema Entità-Relazione che descriva i dati di una applicazione relativa a una catena di officine. Sono di interesse le seguenti informazioni.

- Le officine, con nome (identificante), indirizzo e telefono.
- Le automobili, con targa (identificante) e modello (una stringa di caratteri senza ulteriore struttura) e proprietario.
- I clienti (proprietari di automobili), con codice fiscale, cognome, nome e telefono. Ogni cliente può essere proprietario di più automobili.
- Gli "interventi" di manutenzione, ognuno effettuato presso un'officina e con numero progressivo (unico nell'ambito della rispettiva officina), date di inizio e di fine, pezzi di ricambio utilizzati (con le rispettive quantità) e numero di ore di manodopera.
- I pezzi di ricambio, con codice, nome e costo unitario.

Indicare le cardinalità delle relazioni e (almeno) un identificatore per ciascuna entità.

**Soluzione:**



## Esercizio 8.4

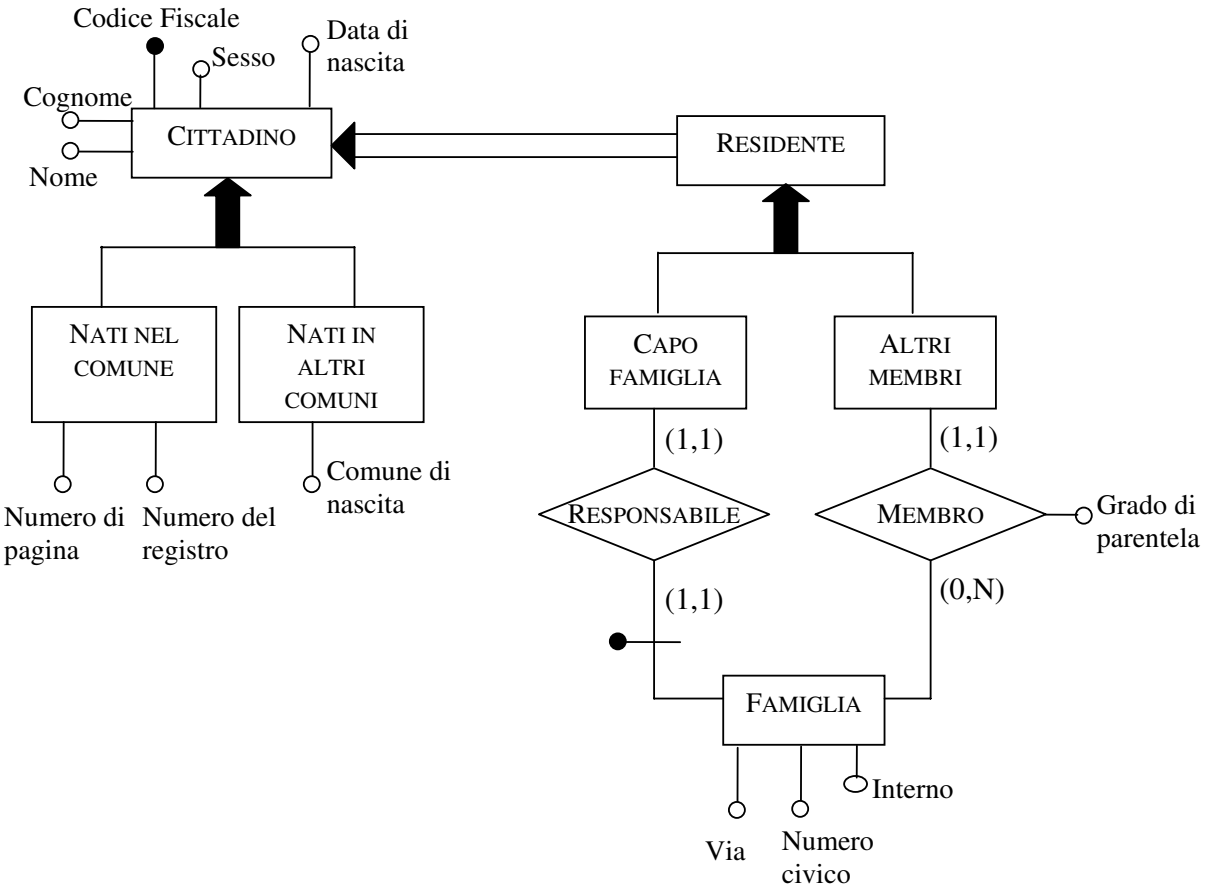
Definire uno schema E-R che descriva i dati di una applicazione relativa all'anagrafe del comune di Chissadove, con cittadini e famiglie. Vanno memorizzate:

- Informazioni sui cittadini nati nel comune e su quelli residenti in esso; ogni cittadino è identificato dal codice fiscale e ha cognome, nome, sesso e data di nascita; inoltre:
  - Per i nati nel comune, sono registrati anche gli estremi di registrazione (numero del registro e pagina)
  - Per i nati in altri comuni, è registrato il comune di nascita

- Informazioni sulle famiglie residenti, ognuna delle quali ha uno e un solo capofamiglia e zero o più membri, per ognuno dei quali è indicato (con la sigla) il grado di parentela (coniuge, figlio, genitore o altro); ogni cittadino residente appartiene ad una e una sola famiglia; tutti i membri di una famiglia hanno lo stesso domicilio (via, numero civico, interno)

Cercare di procedere secondo la strategia inside-out. Al termine, verificare le qualità dello schema ottenuto.

**Soluzione:**




Usando la strategia inside-out , la creazione di questo schema parte con l'entità CITTADINO; attorno ad esso, possiamo aggiungere le due specializzazioni e le specializzazioni di RESIDENTE (questa specializzazione rappresenta i residenti, ovunque siano nati).
Infine, possiamo aggiungere l'entità FAMIGLIA con le due relazioni Responsabile e Membro.

Questo schema è corretto e completo perché rappresenta tutte le specifiche con i costrutti corretti.

## Esercizio 8.5

Analizzare le specifiche relative a partite di un campionato di calcio riportate in figura8.16 e costruite un glossario dei termini ad esse relativo.

| **Campionato di calcio** |
|---|
| *Per ogni partita, descrivere il girone e la giornata in cui si è svolta, il numero progressivo nella giornata (es. prima partita, seconda partita, ecc), la data, con giorno , mese e anno, le squadre coinvolte nella partita, con nome, città della squadra e allenatore, e infine per ciascuna squadra se ha giocato in casa. Si vogliono conoscere i giocatori che giocano in ogni squadra con i loro nomi e cognomi, la loro data di nascita e il loro ruolo principale. Si vuole conoscere, per ogni partita, i giocatori che hanno giocato, i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita) e nome, cognome, città e regione di nascita dell'arbitro della partita. Distinguere le partite giocate regolarmente da quelle rinviate. Per quelle rinviate, rappresentare la data in cui si sono effettivamente giocate. Distinguere anche le partite giocate in una città diversa da quella della squadra ospitante; per queste si vuole rappresentare la città in cui si svolgono, nonché il motivo della variazione di sede. Dei giocatori interessa anche la città di nascita.* |

**Figura 8.16** Specifiche per l'esercizio 8.5

**Soluzione:**

**Glossario dei termini**

| Termine | Descrizione | Sinonimo | Collegamento |
|---|---|---|---|
| Partita | Una partita giocata nel torneo; può essere rinviata o giocata in campo neutrale | | Giocatore, Squadra, Giornata, Arbitro |
| Giornata | In una giornata si giocano molte partite. Ogni giornata ha la sua data (giorno, mese e anno) | | Partita, Squadra |
| Squadra | Una squadra che gioca nel campionato | | Giocatore, Partita, Giornata |
| Giocatore | Un giocatore che gioca in una squadra; è importante conoscere in quali partite ha giocato ed in quali posizioni | | Squadra, Partita |
| Arbitro | Un arbitro che arbitra una partita del campionato | | Partita |

## Esercizio 8.6

Dopo aver riorganizzato in gruppi omogenei le specifiche dell'esercizio precedente, rappresentarle con il modello Entità-Relazione, procedendo in maniera top-down per livelli di astrazione successiva a partire da uno schema scheletro iniziale. Si osservi che lo schema in figura 7.28 rappresenta una possibile soluzione di questo esercizio.

**Soluzione:**

FRASI RELATIVE ALLA PARTITA E ALLA GIORNATA

Per ogni partita, descrivere il girone e la giornata in cui si è svolta, il numero progressivo nella giornata (es. prima partita, seconda partita, ecc), la data, con giorno, mese e anno.
Distinguere le partite giocate regolarmente da quelle rinviate. Per quelle rinviate, rappresentare la data in cui si sono effettivamente giocate
Distinguere anche le partite giocate in una città diversa da quella della squadra ospitante; per queste si vuole rappresentare la città in cui si svolgono, nonché il motivo della variazione di sede

FRASI RELATIVE ALL'ARBITRO

Si vuole conoscere, per ogni partita, nome, cognome, città e regione di nascita dell'arbitro della partita

FRASI RELATIVE ALLE SQUADRE

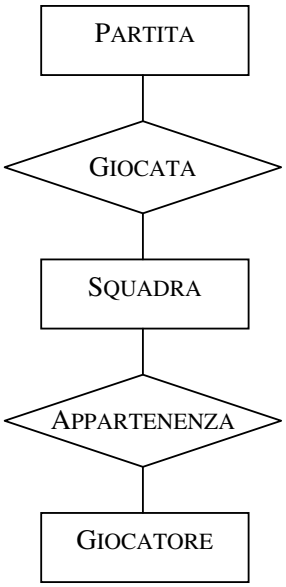
Per ogni partita, descrivere le squadre coinvolte nella partita, con nome, città della squadra e allenatore, e infine per ciascuna squadra se ha giocato in casa.
Memorizziamo, per ogni giornata, quanti punti ha ogni squadra.

FRASI RELATIVE AI GIOCATORI

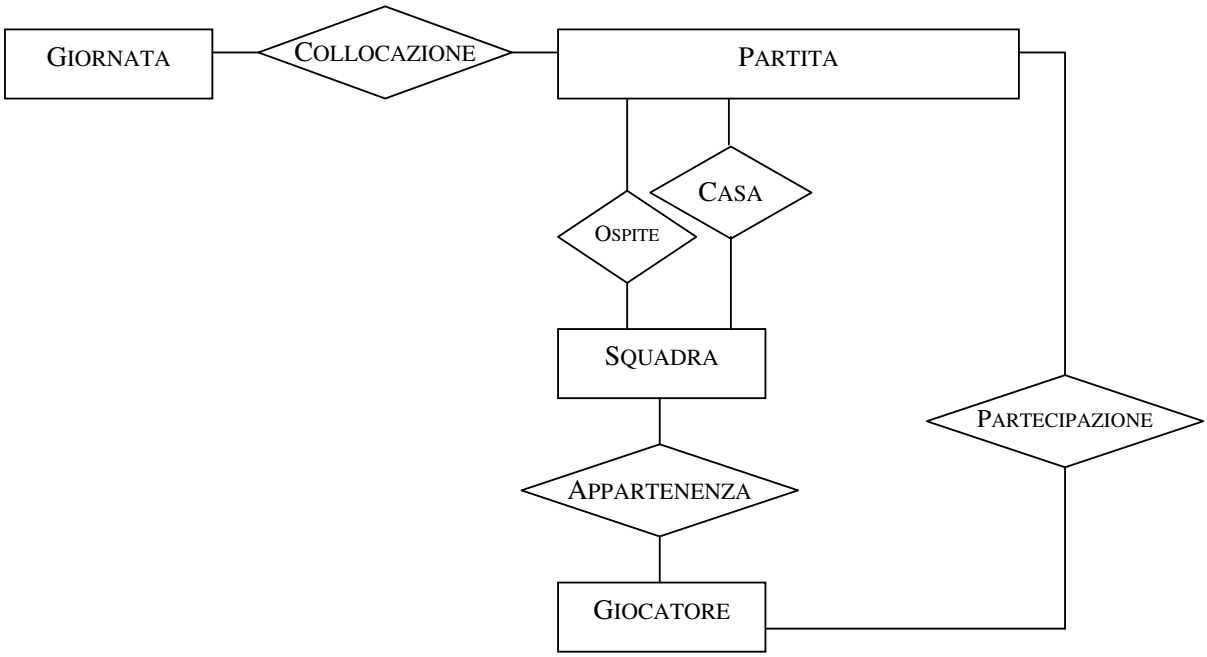
Si vogliono conoscere i giocatori che giocano in ogni squadra con i loro nomi e cognomi, la loro data di nascita e il loro ruolo principale. Si vogliono conoscere, per ogni partita, i giocatori che hanno giocato, i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita).
Per ogni giocatore siamo interessati alla città di nascita.

I seguenti schemi rappresentano i passi per la costruzione dello schema finale, usando la strategia top-down.

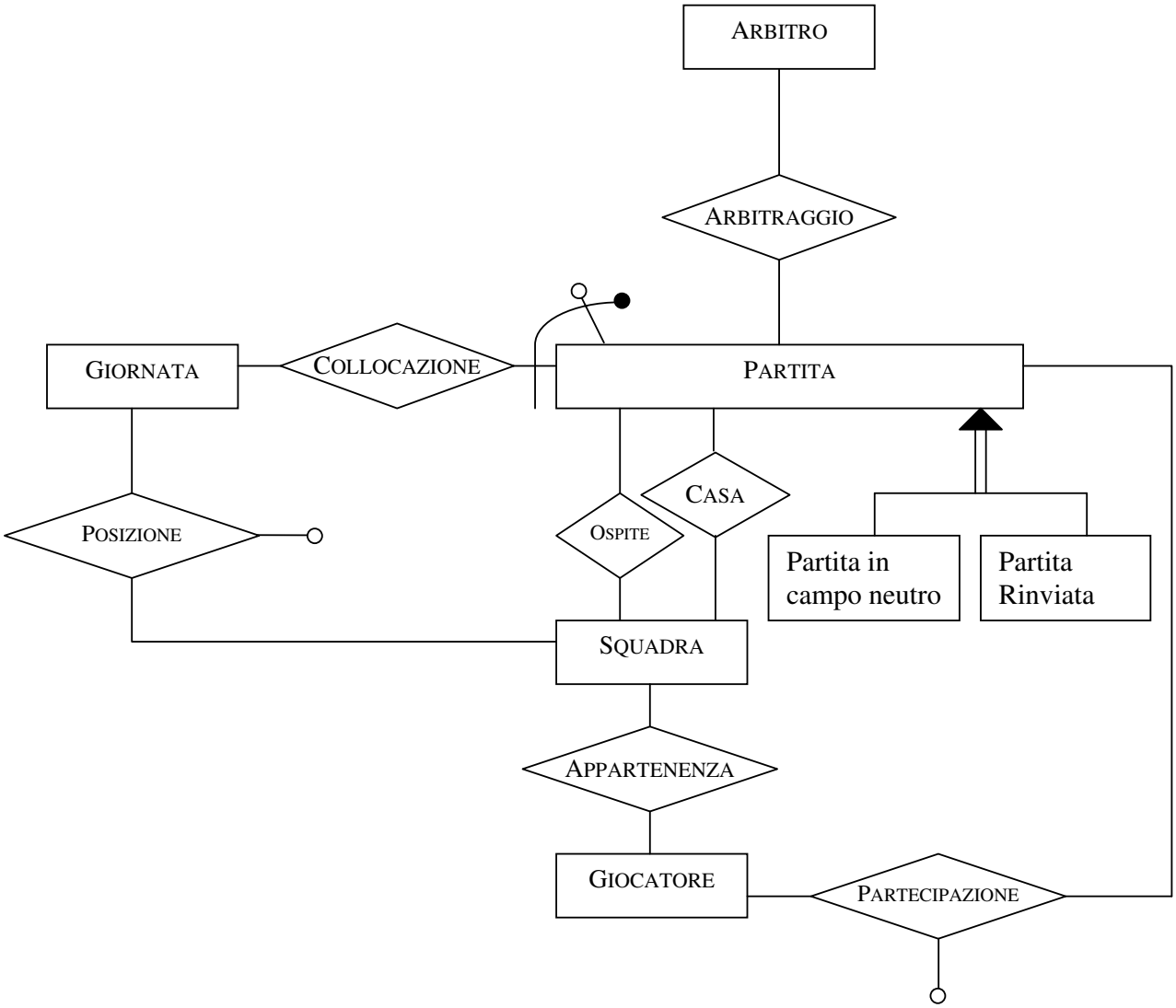
1) Schema Skeleton



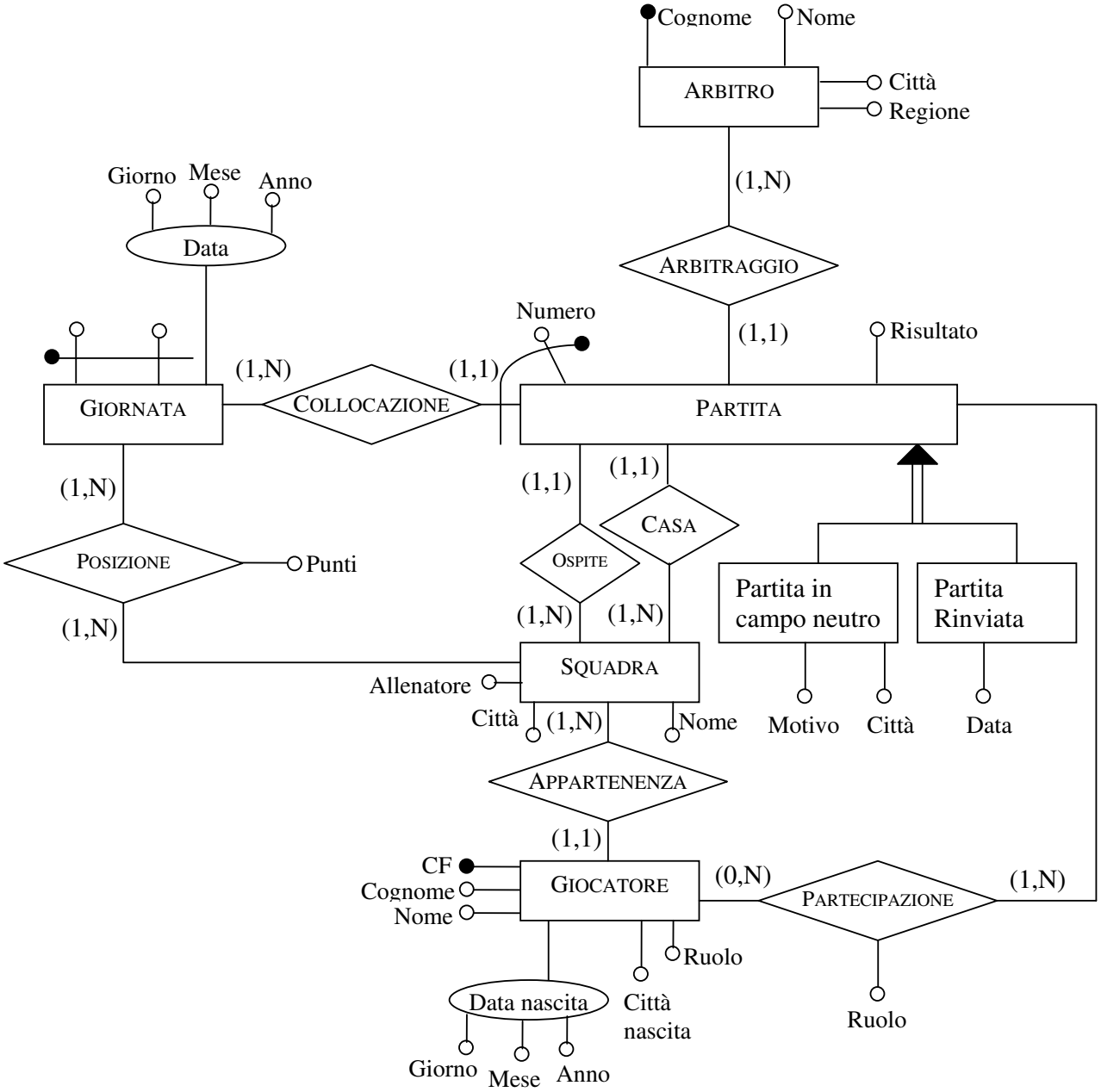

2)



3)



4)



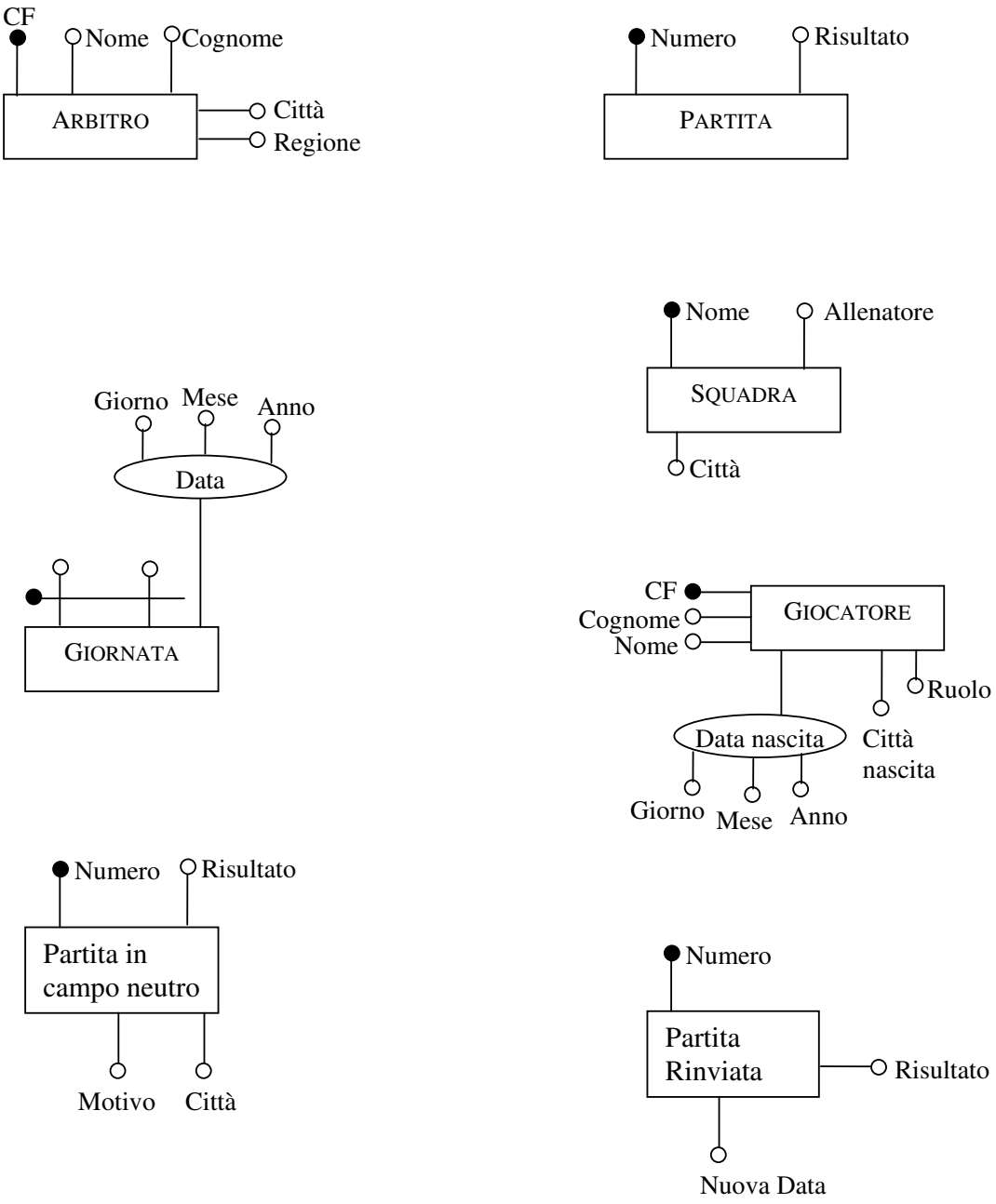
## Esercizio 8.7

Provare a rappresentare di nuovo le specifiche dell'esercizio 8.5 con uno schema Entità-Relazione, procedendo però in maniera bottom-up: costruire frammenti di schema separati che descrivono le varie componenti omogenee delle specifiche e poi procedere per integrazione dei vari schemi. Confrontare il risultato con lo schema ottenuto nell'esercizio 8.6.

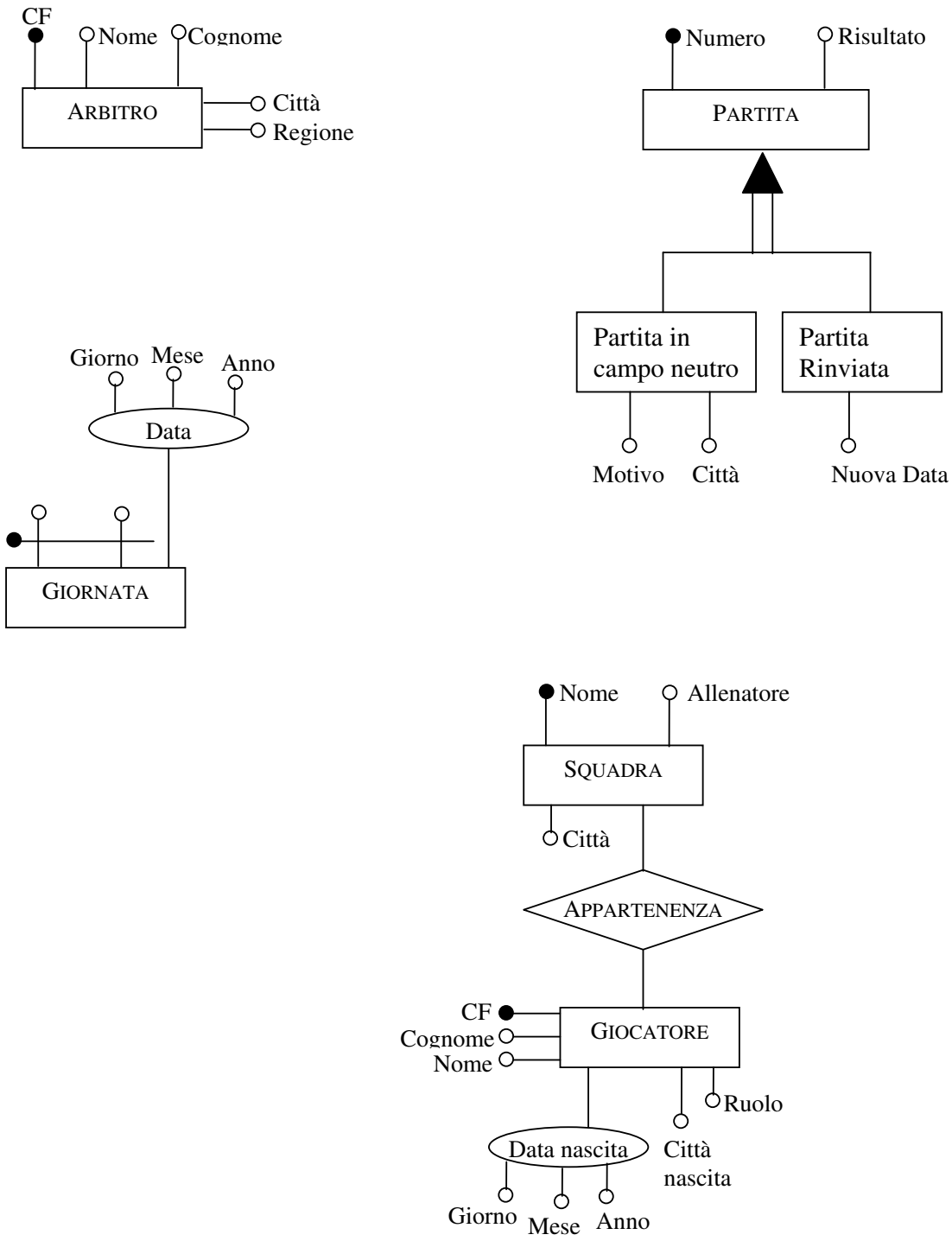
**Soluzione:**

I passi seguenti rappresentano lo schema finale utilizzando la strategia bottom-up.
Gli schemi così ottenuti sono simili a quelli dell'esercizio precedente.

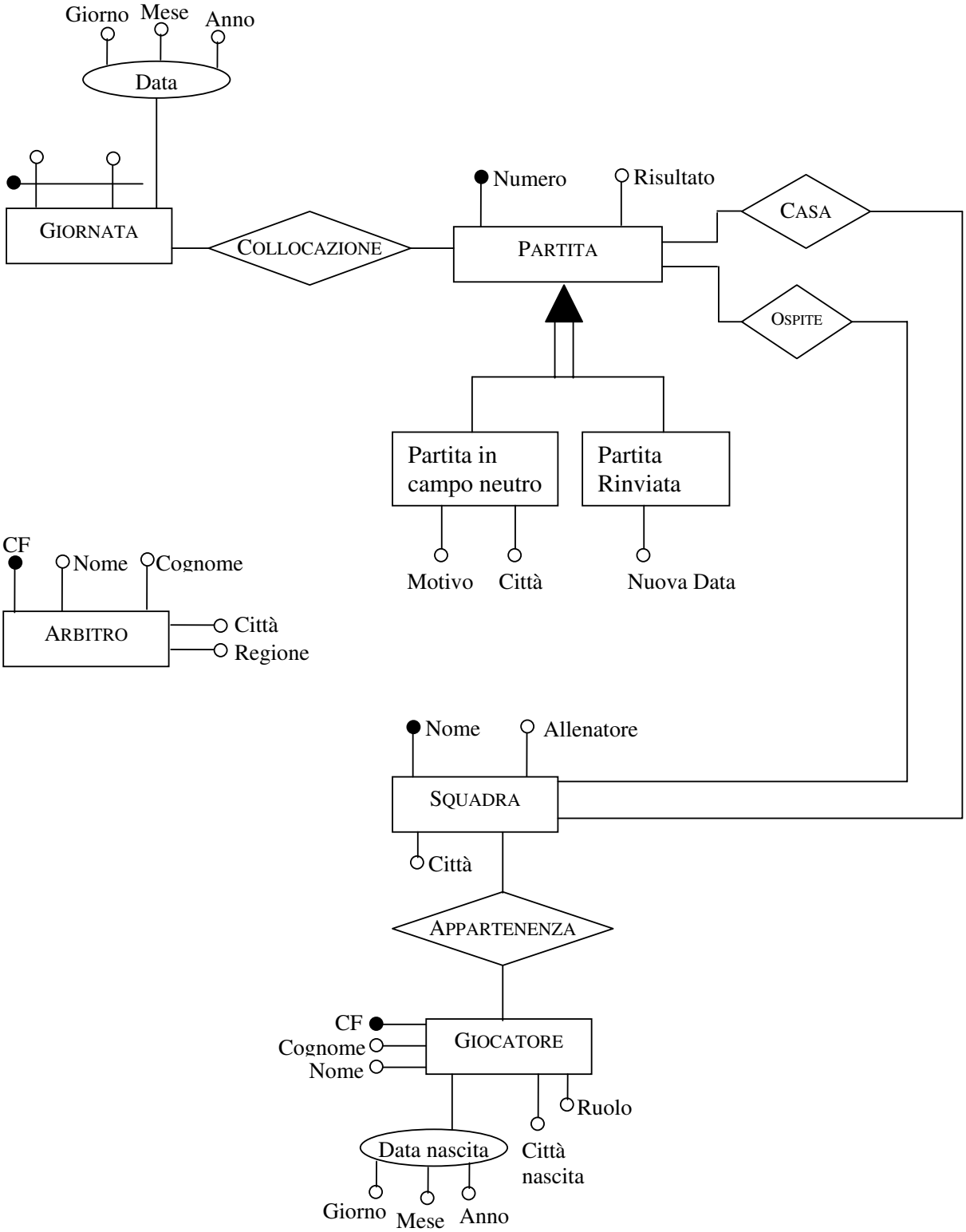
1)



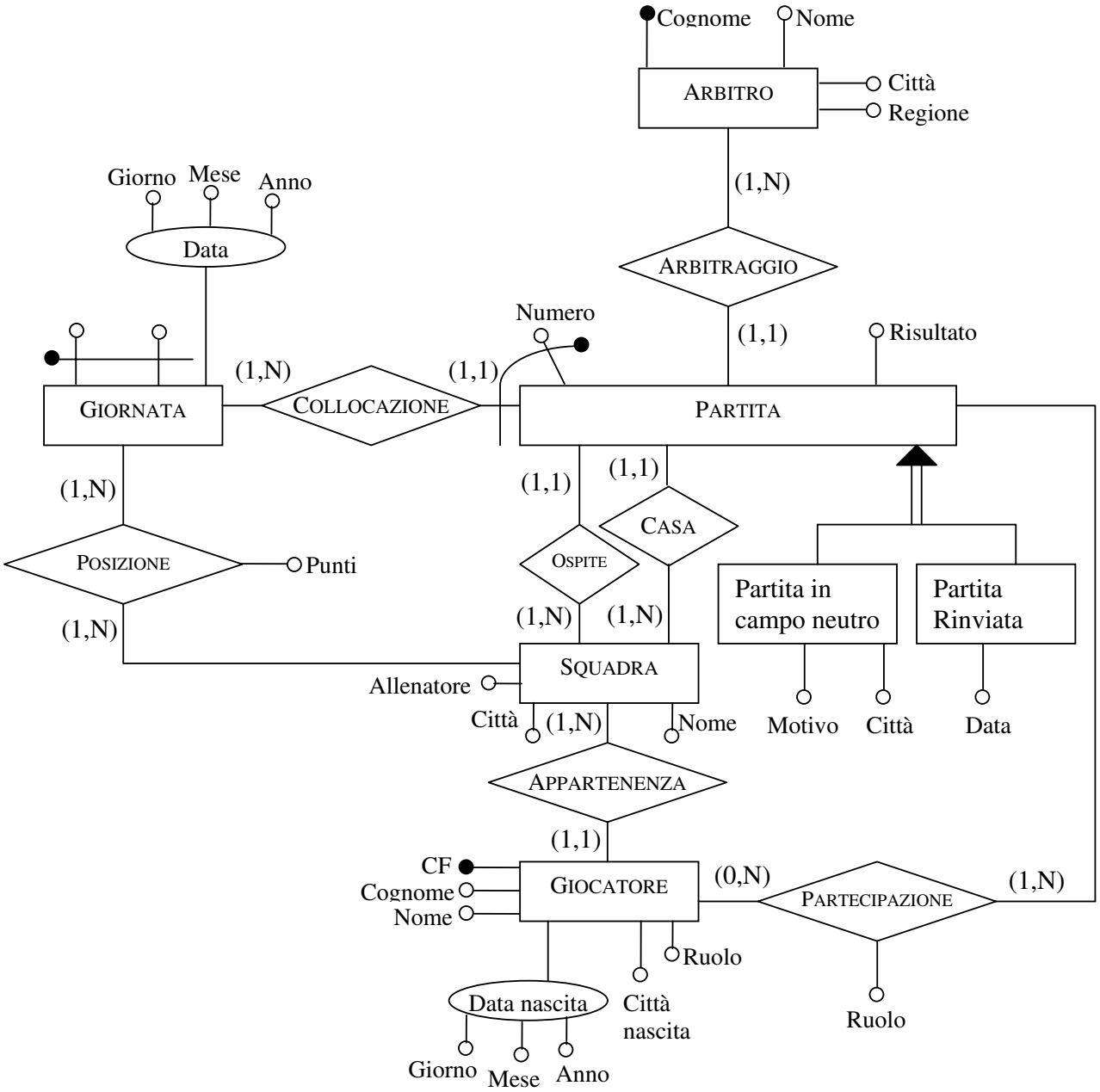
2)



3)



4)



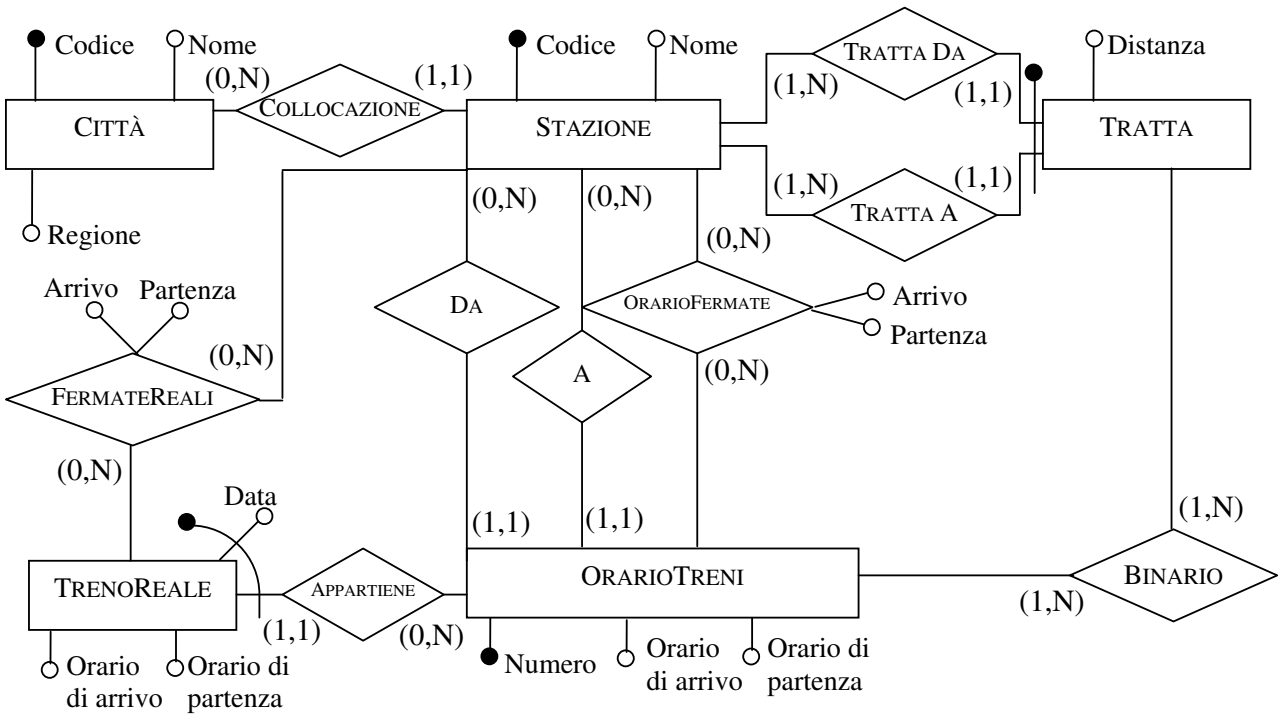
## Esercizio 8.8

Si vuole effettuare una operazione di re*verse-engeneering*, ovvero si vuole ricostruire, a partire da una base di dati relazionale, una sua rappresentazione concettuale con il modello Entità-Relazione. La base di dati è relativa a una applicazione su treni e stazioni ferroviarie ed è composta dalle seguenti relazioni:

- STAZIONE(Codice, Nome, Città), con il vincolo di integrità referenziale fra l'attributo Città e la relazione CITTÀ;
- CITTÀ(Codice, Nome, Regione);
- TRATTA(Da, A,Distanza) con i vincoli di integrità referenziale tra l'attributo Da e la relazione STAZIONE e tra l'attributo A e la relazione STAZIONE; questa relazione contiene tutte e sole le coppie di stazioni connesse da una linea in modo diretto (cioè senza stazioni intermedie);
- ORARIOTRENI(Numero, Da, A, OrarioDiPartenza, OrarioDiArrivo) con vincoli di integrità referenziale tra l'attributo Da e la relazione STAZIONE e tra l'attributo A e la relazione STAZIONE;
- TRATTETRENO(NumeroTreno, Da, A) con vincoli di integrità referenziale tra l'attributo NumeroTreno e la relazione ORARIOTRENI e tra gli attributi Da e A e la relazione TRATTA;
- ORARIOFERMATE(NumeroTreno, Stazione, Arrivo, Partenza) con il vincolo di integrità referenziale tra l'attributo numero treno e la relazione OrarioTreni e tra l'attributo Stazione e la relazione STAZIONE;
- TRENOREALE(Numero, Data, OrarioDiPartenza, OrarioDiArrivo) con il vincolo di integrità referenziale tra l'attributo Numerto e la relazione ORARIOTRENI;
- FERMATEREALI(NumeroTreno, Data, Stazione, Arrivo, Partenza) con il vincolo di integrità referenziale tra gli attributi NumeroTreno e Stazione e la relazione ORARIOFERMATE.

Segnalare eventuali ridondanze. In particolare, qualora si tratti di relazioni derivate.

**Soluzione:**



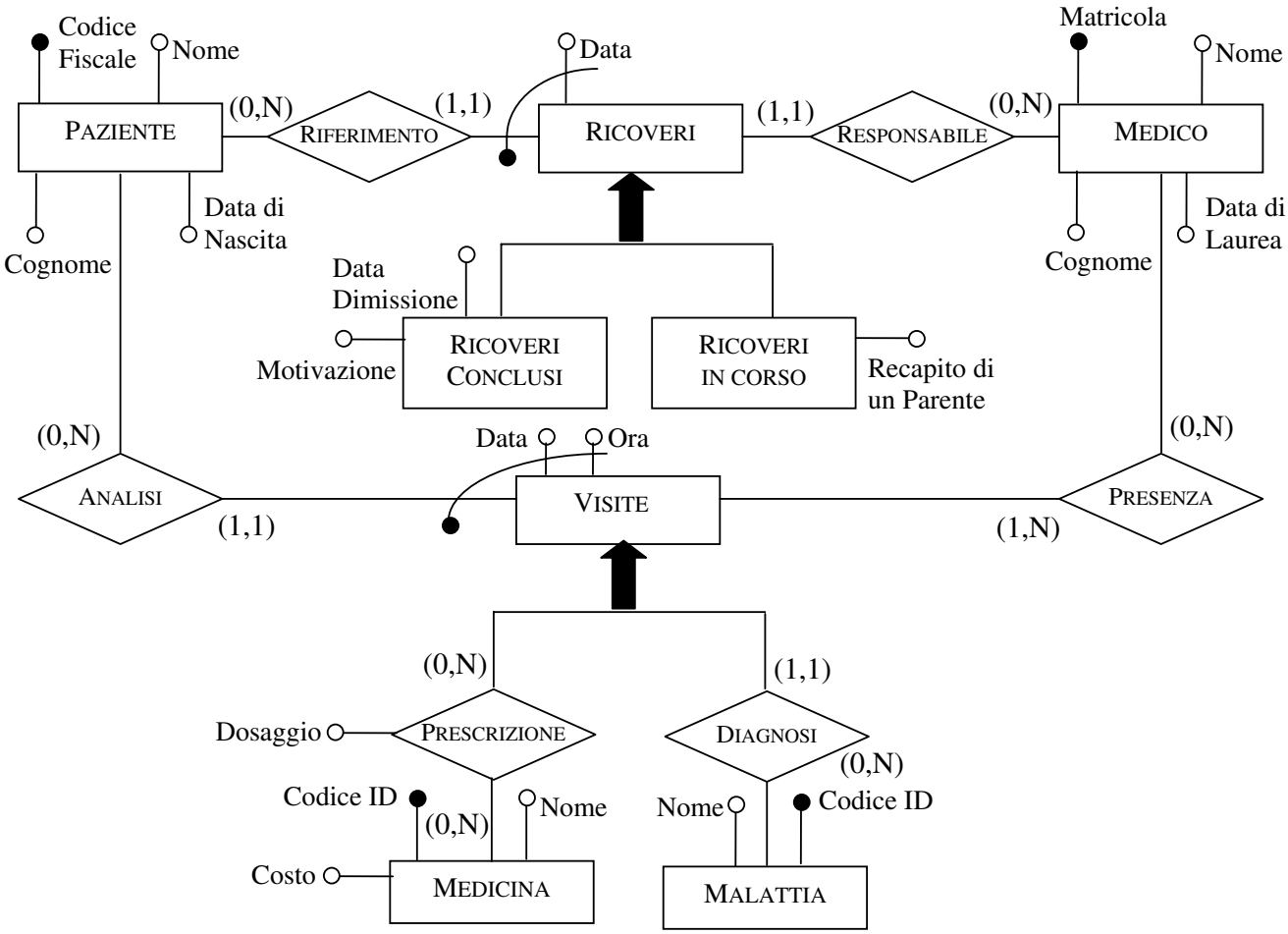
La relazione BINARIO contiene informazioni ridondanti, perché il binario può essere ottenuto dalle relazioni Da, A, OrarioFermate. Comunque questa informazione potrebbe essere un po' più difficile, e le informazioni sui binari potrebbero essere richieste spesso nel database; così la ridondanza è utile per migliorare le prestazioni, inoltre il concetto di tratta è importante in questo contesto, e così è corretto rappresentarlo nello schema E-R.

## Esercizio 8.9

Definire uno schema Entità-Relazione che descriva i dati di una applicazione relativa ad un reparto ospedaliero. Sono di interesse le seguenti informazioni.

- I pazienti, con codice fiscale, nome, cognome e data di nascita.
- I ricoveri dei pazienti, ognuno con data di inizio (identificante nell'ambito dei ricoveri di ciascun paziente) e medico curante; inoltre, per i ricoveri conclusi, la data di conclusione e la motivazione (dimissione, trasferimento, etc.), e, per i ricoveri in corso, il recapito di un parente (che si può assumere sia semplicemente una stringa)
- I medici, con numero di matricola, cognome, nome e data di laurea.
- Le visite, con la data, l'ora, i medici visitanti, le medicine prescritte (con le relative quantità) e le malattie diagnosticate; ogni visita è identificata dal paziente coinvolto, dalla data e dall'ora.
- Per ogni medicina sono rilevanti un codice identificativo, un nome e un costo.
- Per ogni malattia sono rilevanti un codice identificativo e un nome.

**Soluzione:**



## Esercizio 8.10

Definire uno schema Entità-Relazione che descriva i dati di una applicazione relativa all'archivio di un amministratore di condomini, secondo le seguenti specifiche (semplificate rispetto a molte realtà).

- Ogni condominio ha un nome (che lo identifica) e un indirizzo e comprende una o più *scale*, ognuna delle quali comprende un insieme di appartamenti.
- Se il condominio comprende più scale, ad ogni scala sono associati:

– Un codice (es: scala "A") che la identifica insieme al nome del condominio;
– Un valore, detto *quota della scala*, che rappresenta, in millesimi, la frazione delle spese del condominio che sono complessivamente di competenza degli appartamenti compresi nella scala.

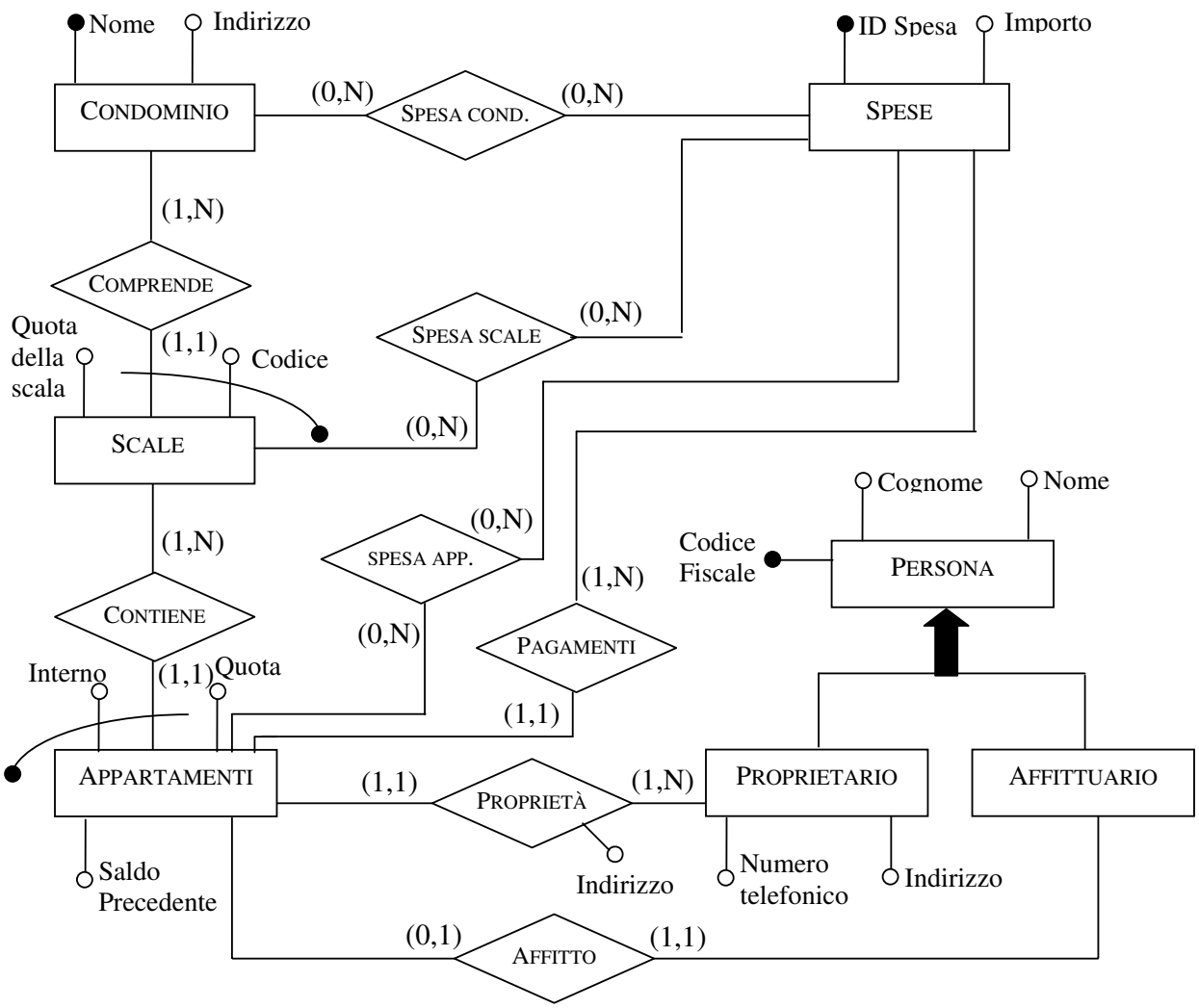
- Ogni appartamento è identificato, nel rispettivo condominio, dalla scala (se esiste) e da un numero (l'*interno*). Ad ogni appartamento è associata una quota (ancora espressa in millesimi), che indica la frazione della spese (della scala) che sono di competenza dell'appartamento.
- Ogni appartamento ha un proprietario per il quale sono di interesse il nome, il cognome, il codice fiscale e l'indirizzo al quale deve essere inviata la corrispondenza relativa all'appartamento. Ogni persona ha un solo codice fiscale, ma potendo essere proprietario di più appartamenti, potrebbe anche avere indirizzi diversi per appartamenti diversi. Di solito, anche chi è proprietario di molti appartamenti ha comunque solo uno o pochi indirizzi. In molti casi, l'indirizzo del proprietario coincide con quello del condominio.
- Per la parte contabile, è necessario tenere traccia delle spese sostenute dal condominio e dei pagamenti effettuati dai proprietari.

– Ogni spesa è associata ad un intero condominio, oppure ad una scala o ad un singolo appartamento.
– Ogni pagamento è relativo ad uno e un solo appartamento.

Nella base di dati vengono mantenuti pagamenti e spese relativi all'esercizio finanziario in corso (di durata annuale) mentre gli esercizi precedenti vengono sintetizzati attraverso un singolo valore (il *saldo precedente*) per ciascun appartamento che indica il debito o il credito del proprietario. In ogni istante esiste un *saldo corrente* per ciascun appartamento, definito come somma algebrica del saldo precedente e dei pagamenti (positivi) e delle spese addebitate (negative).

Se e quando lo si ritiene opportuno, introdurre codici identificativi sintetici.

**Soluzione:**



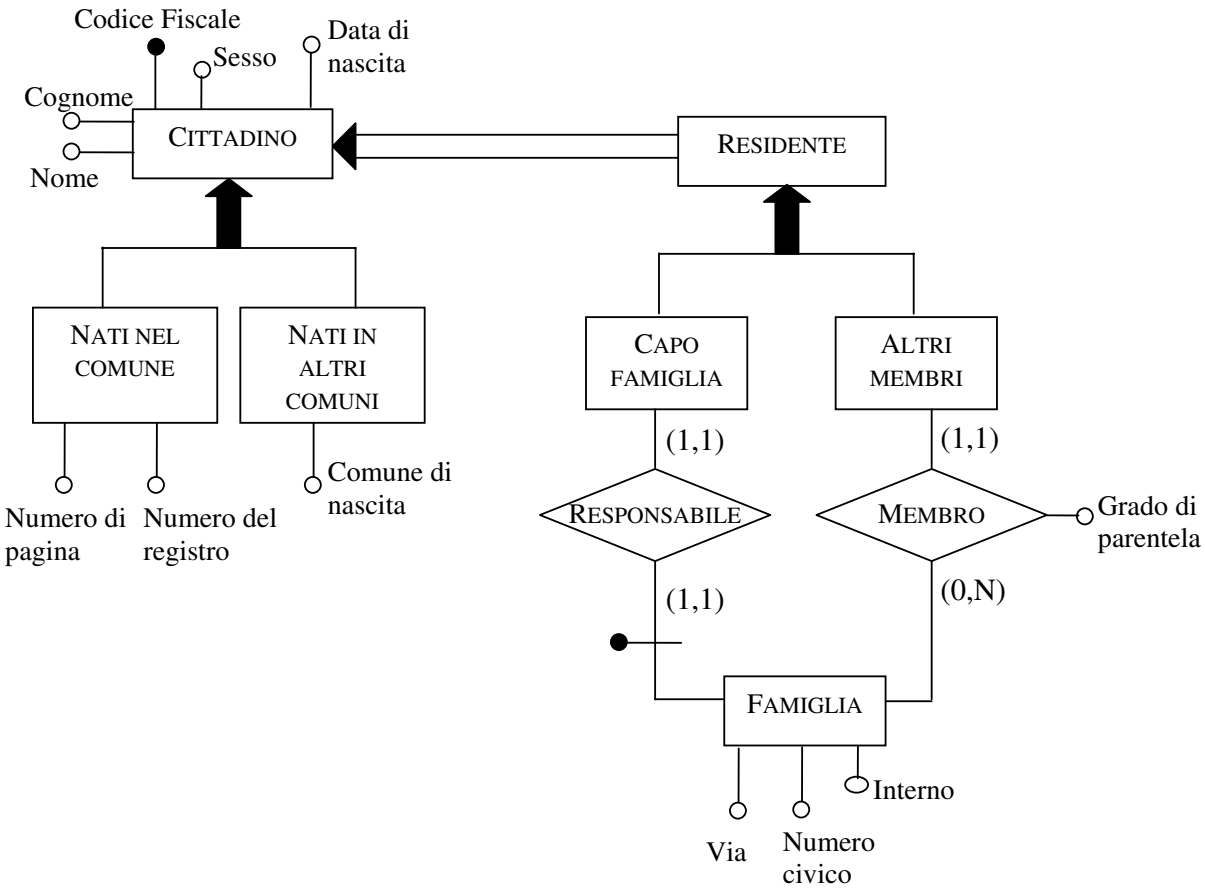
# Capitolo 9

## Esercizio 9.1

Si consideri lo schema Entità-Relazione ottenuto come soluzione dell'esercizio 8.4. Fare delle ipotesi sul volume dei dati e sulle operazioni possibili su questi dati e, sulla base di queste ipotesi, effettuare le necessarie ristrutturazioni dello schema. Effettuare poi la traduzione verso il modello relazionale.

**Soluzione:**

Questo è lo schema prodotto nell'esercizio 8.4



Le seguenti tavole contengono ipotesi sui volumi e sulle operazioni:

**Volumi:**

| Concetto | Tipo | Volume |
|---|---|---|
| Cittadino | E | 1.100.000 |
| Nati nel comune | E | 1.000.000 |
| Nati in altri comuni | E | 100.000 |
| Residente | E | 1.000.000 |
| Capo famiglia | E | 250.000 |
| Altri membri | E | 750.000 |
| Famiglia | E | 250.000 |
| Responsabile | R | 250.000 |
| Membro | R | 750.000 |

**Operazioni:**

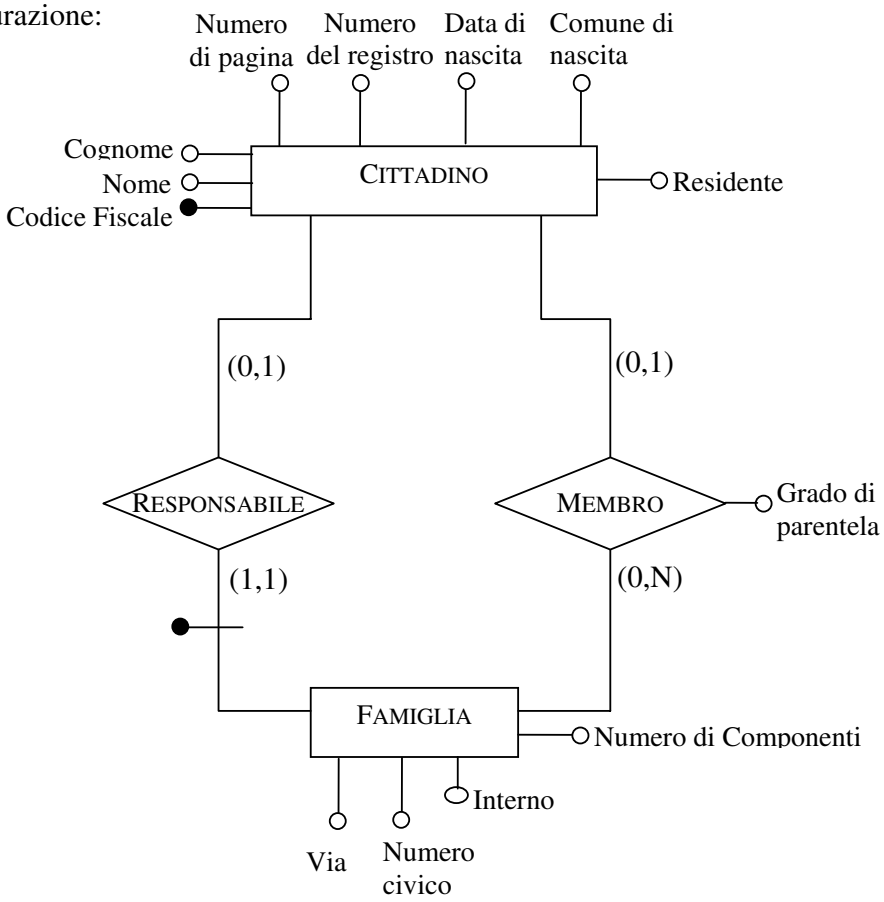
| Operazione | Descrizione | Frequenza | Tipo |
|---|---|---|---|
| 1 | Aggiungere un nuovo cittadino nato nel comune | 100 al giorno | I |
| 2 | Aggiungere un nuovo cittadino residente nel comune ma nato in un altro comune | 20 al giorno | I |
| 3 | Aggiungere una nuova famiglia | 20 al giorno | I |
| 4 | Cancellare un cittadino | 100 al giorno | I |
| 5 | Cancellare una famiglia | 5 al giorno | I |
| 6 | Visualizzare il numero di cittadini residenti nel comune | 1 al giorno | B |
| 7 | Visualizzare un numero di residenti uomini e donne | 1 al giorno | B |

Potrebbe essere utile per aggiungere un attributo ridondante "Numero di Componenti" all'entità FAMIGLIA. Senza questo attributo, l'operazione 6 ha bisogno di 1.000.000 di accessi in lettura all'entità RESIDENTE ogni giorno. Con questo attributo ridondante, l'operazione 6 ha bisogno di soli 250.000 accessi in lettura all'entità FAMIGLIA.
Comunque, la presenza di questo attributo cambia il costo delle operazioni 1, 2 e 4; infatti queste 3 operazioni hanno ora bisogno, oltre agli accessi che già avevano, anche di un accesso in lettura a CAPO FAMIGLIA (o ad ALTRI MEMBRI), un accesso a RESPONSABILE (o a MEMBRO), un accesso in lettura ed uno in scrittura all'entità FAMIGLIA (per aggiornare l'attributo "Numero di componenti").

Supponendo che un accesso in scrittura abbia il costo di 2 accessi in lettura, il costo totale è (1+1+1+2)*90 + (1+1+1+2)*20 + (1+1+1+2)*100 = 1.050
La frequenza dell'operazione 1 è 90 perché non tutti i cittadini nati nel comune sono residenti, ma solo il 90%.

Così, il vantaggio dell'attributo ridondante è 750.000 – 1.050 = 748.950 accessi al giorno.

Ristrutturazione:




Traduzioni:

CITTADINO(<u>Codice Fiscale</u>, Cognome, Nome, Numero di pagina, Numero del registro, Data di nascita, Comune di nascita, Residente)

FAMIGLIA(<u>Capo Famiglia</u>, Via, Numero civico, Interno, Numero di Componenti) con vincolo di integrità referenziale tra **Capo Famiglia** e la relazione CITTADINO.

MEMBRO(<u>Cittadino</u>, <u>Famiglia</u>, Grado di parentela) ) con vincolo di integrità referenziale tra **Cittadino** e la relazione CITTADINO e tra **Famiglia** e la relazione FAMIGLIA.

## Esercizio 9.2

Tradurre lo schema Entità-Relazione che abbiamo più volte incontrato sul personale di un'azienda (riportato per comodità in figura 9.35) in uno schema del modello relazionale.

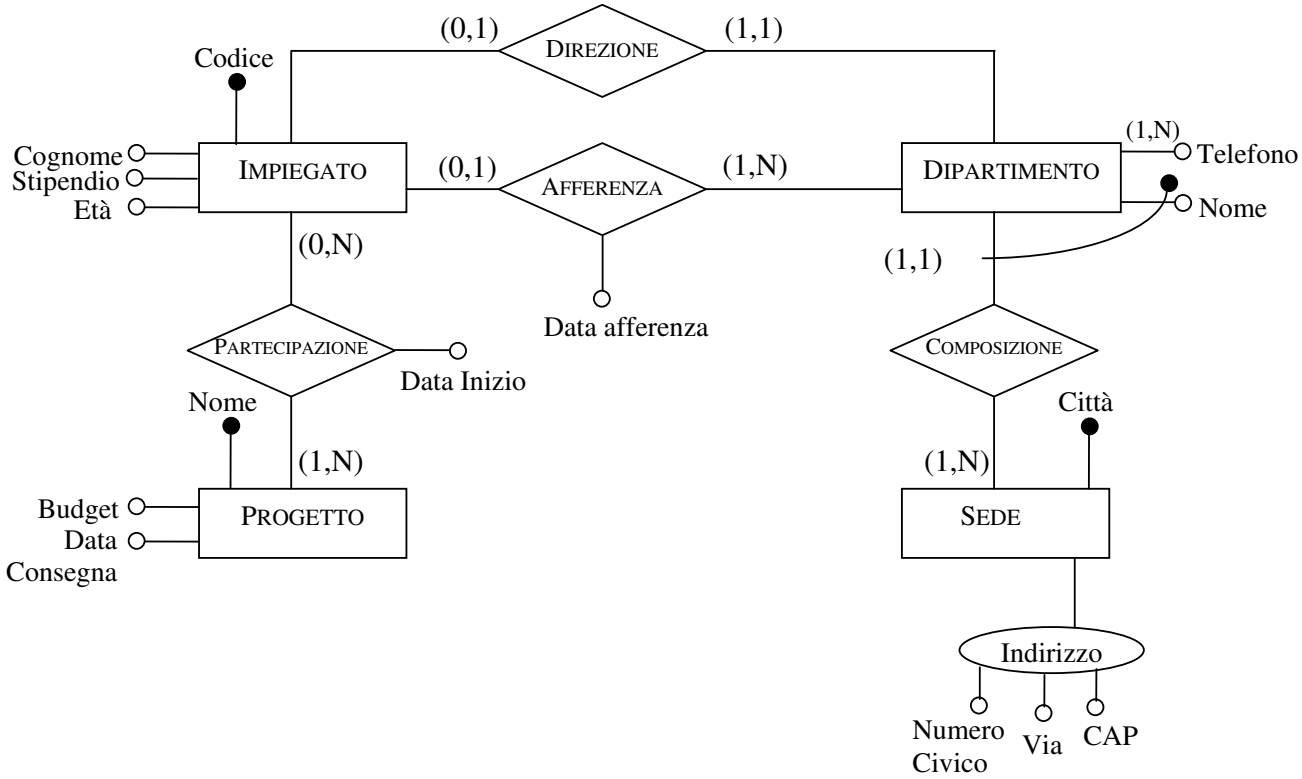



**Figura 9.35** Uno schema E-R sul personale di un'azienda

**Soluzione:**

IMPIEGATO(Codice, Cognome, Stipendio, Età, Dipartimento, Sede, Data afferenza), con vincolo di integrità referenziale tra **Dipartimento** e la relazione DIPARTIMENTO, e tra **Sede** e la relazione SEDE.

DIPARTIMENTO(Nome, Sede) con vincolo di integrità referenziale tra **Sede** e la relazione SEDE.

TELEFONO(Dipartimento, Numero), con vincolo di integrità referenziale tra **Dipartimento** e la relazione DIPARTIMENTO.

SEDE(Città, CAP, Via, Numero Civico)

PROGETTO(Nome, Budget, Data Consegna)

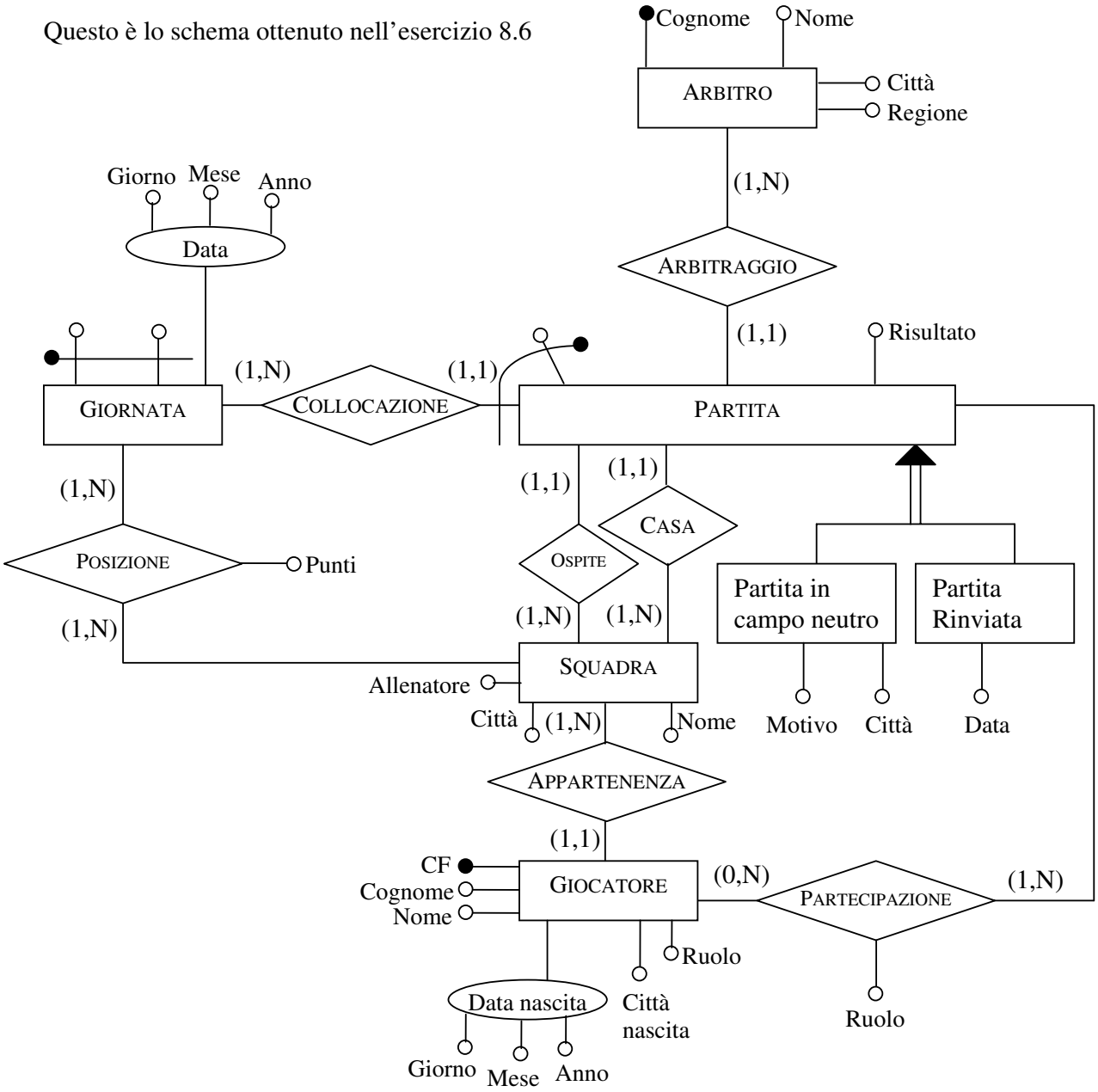
PARTECIPAZIONE(Impiegato, Progetto, Data Inizio) con vincolo di integrità referenziale tra **Impiegato** e la relazione IMPIEGATO e tra **Progetto** e la relazione PROGETTO.

## Esercizio 9.3

Tradurre lo schema Entità-Relazione ottenuto nell'esercizio 8.6 in uno schema del modello relazionale.

**Soluzione:**

Questo è lo schema ottenuto nell'esercizio 8.6



**Traduzioni**:

ARBITRO(Cognome, Nome, Città, Regione)

GIORNATA(Numero, Serie, Giorno, Mese, Anno)

SQUADRA(Nome, Città, Allenatore)

GIOCATORE(Codice Fiscale, Cognome, Nome, Ruolo, Città di Nascita, Squadra) con vincolo di integrità referenziale tra **Squadra** e la relazione SQUADRA.

PARTITA(Numero, DNumero, DSerie, Risultato, Arbitro, Casa, Ospite) con vincoli di integrità referenziale tra **DNumero** e **DSerie** e la relazione GIORNATA, tra **Arbitro** e ARBITRO e tra **Casa** e **Ospite** con la relazione SQUADRA.

PARTITA IN CAMPO NEUTRO(Partita, Numero, Serie, Motivo, Città) con vincoli di integrità referenziale tra **Partita**, **Numero** e **Serie** con la relazione PARTITA.

PARTITA RINVIATA(Partita, Numero, Serie, Data) con vincoli di integrità referenziale tra **Partita**, **Numero** e **Serie** con la relazione PARTITA.

POSIZIONE(Squadra, Numero, Serie, Punteggio) con vincoli di integrità referenziale tra **Squadra** e la relazione SQUADRA e tra **Numero** e **Serie** e la relazione GIORNATA.

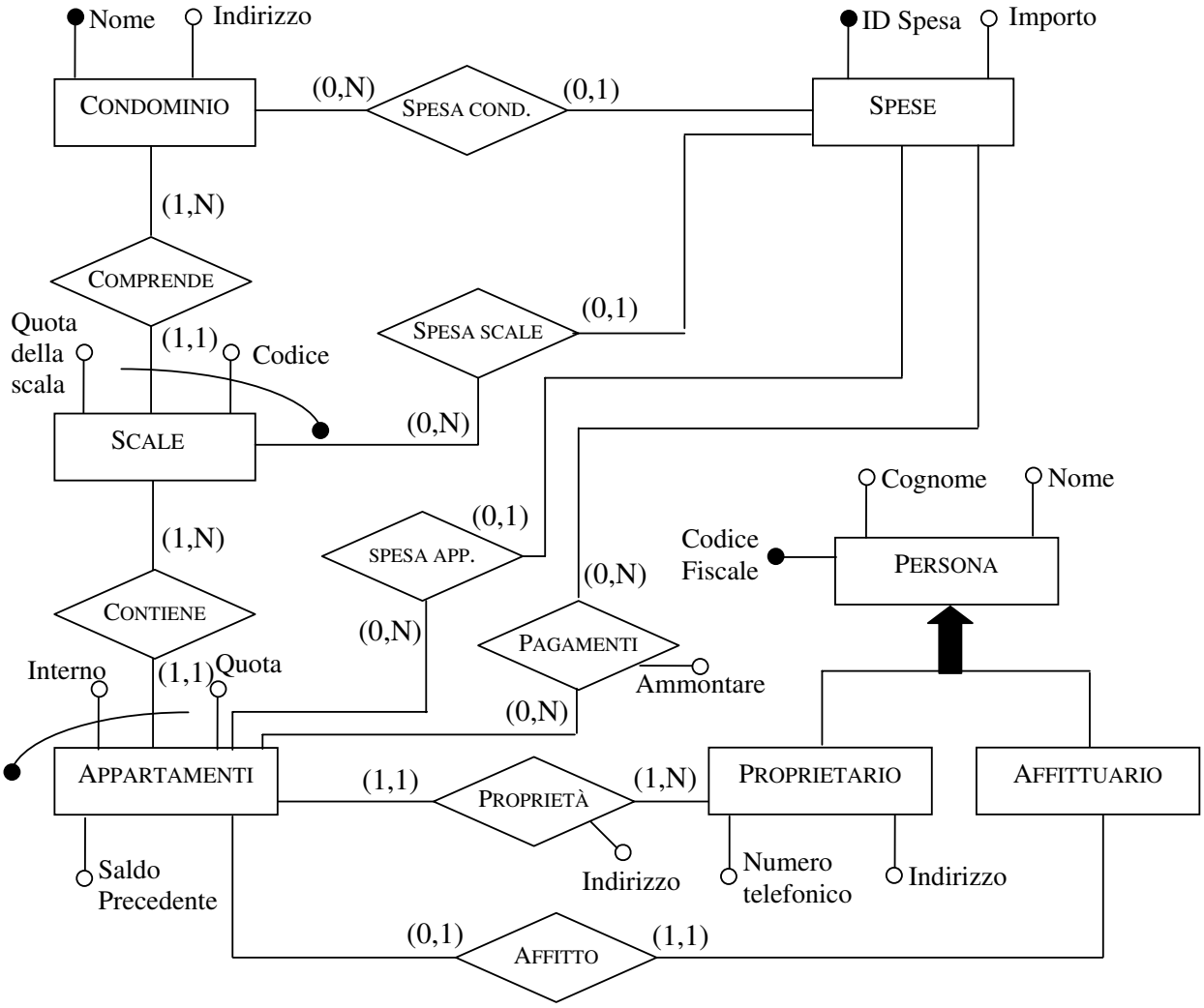
PARTECIPAZIONE(Giocatore, Partita, Numero, Serie, Ruolo) con vincoli di integrità referenziale tra **Giocatore** e la relazione GIOCATORE e tra **Partita**, **Numero**, **Serie** e la relazione PARTITA.

## Esercizio 9.4

Definire uno schema logico relazionale corrispondente allo schema E-R ottenuto nell'esercizio 8 .10. Per la fase di ristrutturazione, indicare le possibili alternative e sceglierne poi una, facendo assunzioni sui parametri quantitativi. Come riferimento per i parametri principali, assumere che la base di dati riguardi cento condomini, mediamente con cinque scale ciascuno, e che ogni scala abbia mediamente venti appartamenti e che le registrazioni principali siano la registrazione di una spesa (cinquanta all'anno per condominio più dieci per scala e cinque per appartamento) e di un pagamento (dieci all'anno per appartamento); annualmente viene stilato il bilancio di ciascun condominio, con il totale degli accrediti e degli addebiti per ciascun appartamento e quindi il calcolo del nuovo saldo (la stampa di ciascun bilancio deve essere organizzata per scale e ordinata).

**Soluzione:**

Questo è lo schema ottenuto nell'esercizio 8.10



Supponendo di avere 100 condomini i volumi del database sono:

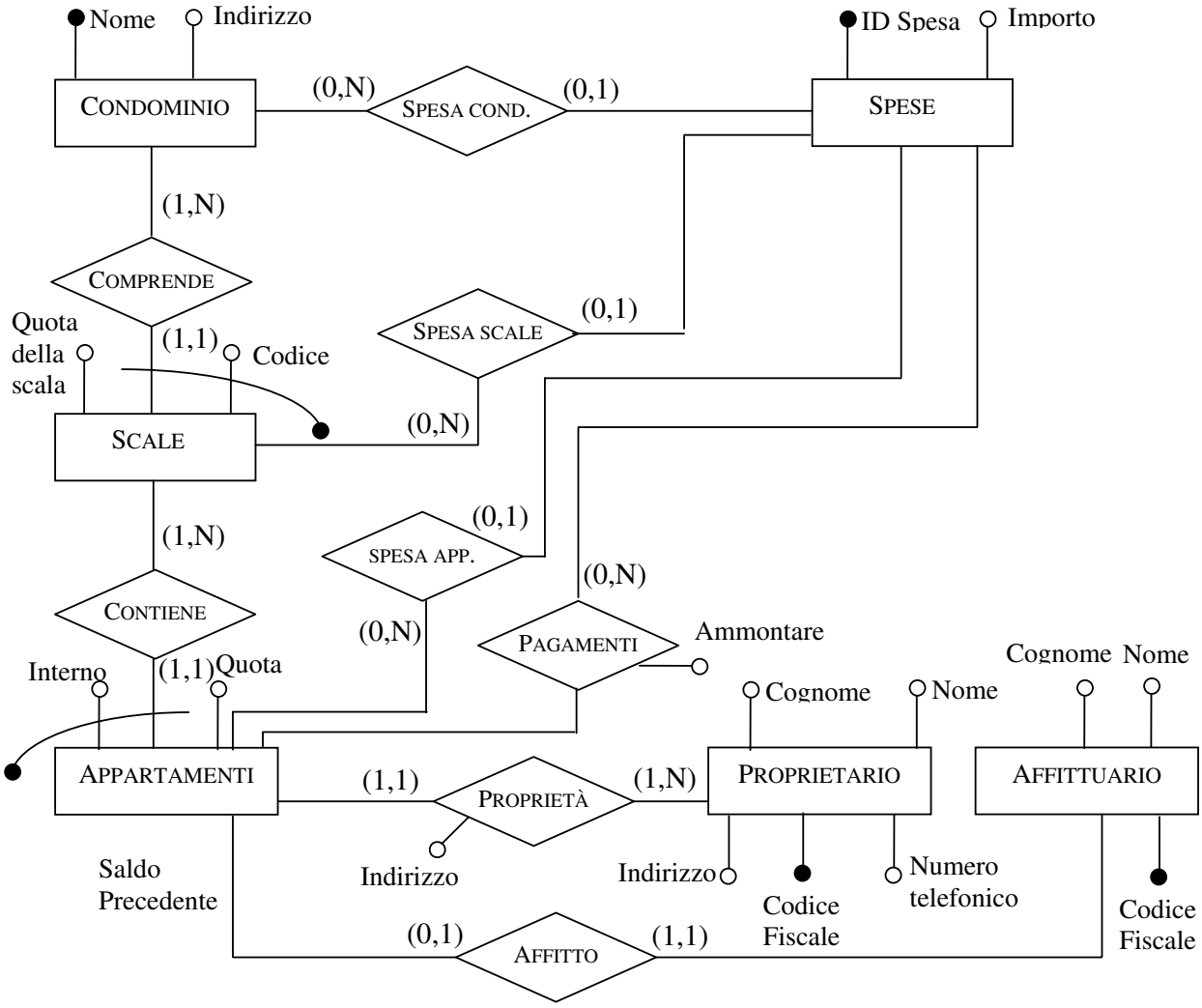
| **Concetto** | **Tipo** | **Volume** |
|---|---|---|
| Condominio | E | 100 |
| Scale | E | 500 |
| Appartamento | E | 10.000 |
| Spese | E | 60.000 |
| Pagamento | E | 100.000 |
| Persona | E | 10.000 |
| Proprietario | E | 8000 |
| Comprende | R | 500 |
| Contiene | R | 10.000 |
| Spesa cond | R | 5.000 |
| Spesa scale | R | 5.000 |
| Spesa App | R | 50.000 |
| Pagamenti | R | 100.000 |

Viste le tavole dei volumi e le operazioni effettuate, possiamo procedere nella ristrutturazione dello schema.

Eliminazione delle gerarchie:

Nello schema è presente una sola gerarchia ed è relativa alle persone ed è totale ed esclusiva, in quanto una persona se è proprietaria di un appartamento non ne è contemporaneamente anche affittuario, quindi si decide di lasciare due entità distinte: l'entità PROPRIETARIO e AFFITTUARIO.

Il nuovo schema sarà il seguente:



Traduzione nel modello relazionale:

CONDOMINIO(Nome, Indirizzo)
SPESE(ID Spesa, Importo)
PROPRIETARIO(Codice Fiscale, Cognome, Nome, Indirizzo, Numero Telefonico)
AFFITTUARIO(Codice Fiscale, Cognome, Nome)

SCALE(Codice, Nome, Quote della scala)
APPARTAMENTI(Interno, Codice, Nome, Quota, CF Proprietario, CF Affittuario)

SPESA COND(ID Spesa, Nome)
SPESA SCALE(ID Spesa, Nome, Codice)
SPESA APP(ID Spesa, Nome, Codice, Interno)
PAGAMENTI(ID Spesa, Nome, Codice, Interno, Ammontare)

## Esercizio 9.5

Tradurre lo schema Entità-Relazione in figura 9.36 in uno schema di basi di dati relazionale. Per ciascuna relazione (dello schema relazionale) si indichi la chiave (che si può supporre unica) e, per ciascun attributo, si specifichi se sono ammessi valori nulli (supponendo che gli attributi dello schema E-R non ammettano valori nulli).

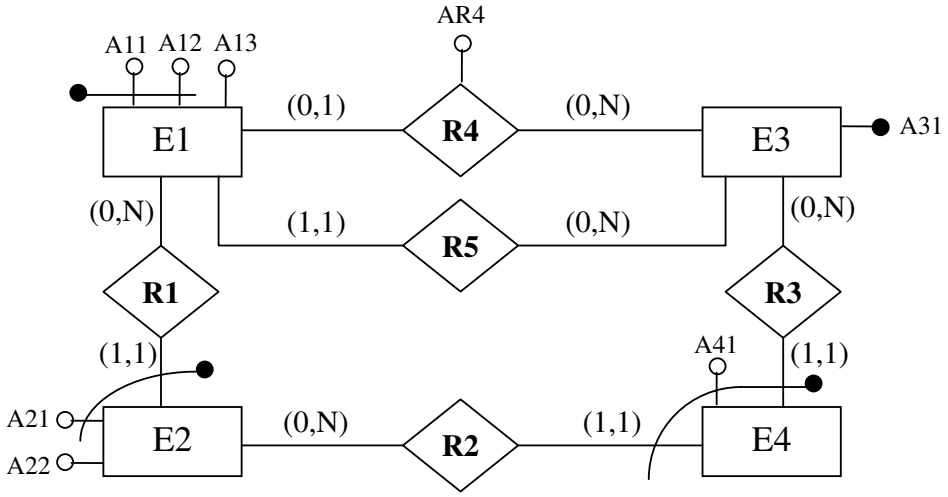



**Figura 9.36** Uno schema E-R da tradurre

**Soluzione:**

Per prima cosa, traduciamo ciascuna entità con una relazione. La traduzione delle entità dotate di identificatore interno è immediata.
E1(A11, A13, A13)
E3(A31)

Traduciamo ora le entità con le identificazioni esterne. Otteniamo le seguenti relazioni:
E2(A21, A11, A12, A22)
E4(A41, A31, A21, A11, A12)

Passiamo ora alla traduzione delle associazioni. Le associazioni R1, R2 e R3 sono già state tradotte come conseguenza dell'identificazione esterna di E2 ed E4.

- Per tradurre R4, introduciamo con opportune ridenominazioni gli attributi che identificano E3 tra quelli di E1, nonché l'attributo AR4 proprio di R4; in pratica, introduciamo A31R4 e AR4. Data la natura della relazione (0,N), per questi attributi sono ammessi valori nulli.
- Per tradurre R5, analogamente al caso precedente, introduciamo A31R5 in E1. In questo caso non sono ammessi valori nulli.

Lo schema relazionale ottenuto è il seguente:
E1(A11, A13, A13, A31R4*, AR4*, A31R5)
E2(A31)
E2(A21, A11, A12, A22)
E4(A41, A31, A21, A11, A12)

## Esercizio 9.6

Sia dato il seguente schema Entità-Relazione in figura 9.37. Ristrutturare lo schema, eliminando le gerarchie, supponendo che le operazioni più significative siano le seguenti, ciascuna eseguita 10 volte al giorno:

**Operazione 1:** Accesso agli attributi $A_{21}$, $A_{22}$, $A_{11}$, $A_{12}$, $A_{13}$ dell'entità $E_2$;
**Operazione 2:** Accesso agli attributi $A_{41}$, $A_{42}$, $A_{31}$, $A_{11}$, $A_{12}$, $A_{13}$ dell'entità $E_4$
**Operazione 3** Accesso agli attributi $A_{51}$,$A_{52}$, $A_{31}$ , $A_{11}$, $A_{12}$, $A_{13}$ dell'entità $E_5$;

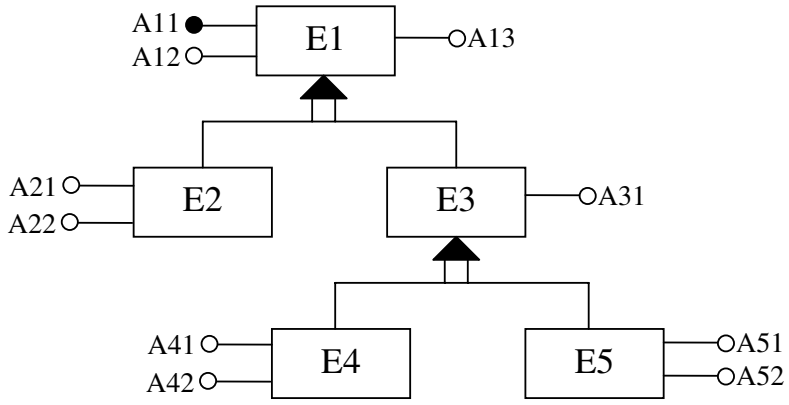



**Figura 9.37** Uno schema E-R con generalizzazioni

**Soluzione:**

Lo schema precedente può essere ristrutturato in vari modi, dipendenti dal volume delle tabelle e dalla complessità che si vuole ottenere.

Tenendo conto delle possibili sovrapposizioni tra le popolazioni E2, E4, E5, la soluzione migliore consiste nel ristrutturare la gerarchia in un'unica entità. Questa soluzione è la più semplice in assoluto ed ha il pregio di avere solamente 30 accessi giornalieri. Di contro, l'entità ha la presenza di valori nulli e il conseguente spreco di memoria.

E(<u>A11</u>, A12, A13, A21*, A22*, A31*, A41*, A42*, A51*, A52*)

## Esercizio 9.7

Si consideri lo schema concettuale di Figura 9.38, che descrive i dati di conti correnti bancari. Si osservi che un cliente può essere titolare di più conti correnti e che uno stesso conto corrente può essere intestato a diversi clienti. Si supponga che su questi dati, sono definite le seguenti operazioni principali:

**Operazione 1:** Apri un conto corrente ad un cliente.
**Operazione 2:** Leggi il saldo totale di un cliente.
**Operazione 3:** Leggi il saldo di un conto.
**Operazione 4:** Ritira i soldi da un conto con una transazione allo sportello.
**Operazione 5:** Deposita i soldi in un conto con una transazione allo sportello.
**Operazione 6:** Mostra le ultime 10 transazioni di un conto.
**Operazione 7:** Registra transazione esterna per un conto.
**Operazione 8:** Prepara rapporto mensile dei conti.
**Operazione 9:** Trova il numero dei conti posseduti da un cliente.
**Operazione10:** Mostra le transazioni degli ultimi 3 mesi dei conti delle società con saldo negativo.

Si supponga infine che, in fase operativa, i dati di carico per questa applicazione bancaria siano quelli riportati in figura 9.39.
Effettuare la fase di progettazione logica sullo schema E-R tenendo conto dei dati forniti. Nella fase di ristrutturazione si tenga conto del fatto che sullo schema esistono due ridondanze: Gli attributi **Saldo Totale** e **Numero di Conti** dell'entità CLIENTE. Essi possono infatti essere derivati dall'associazione TITOLARITÀ e dall'entità CONTO.

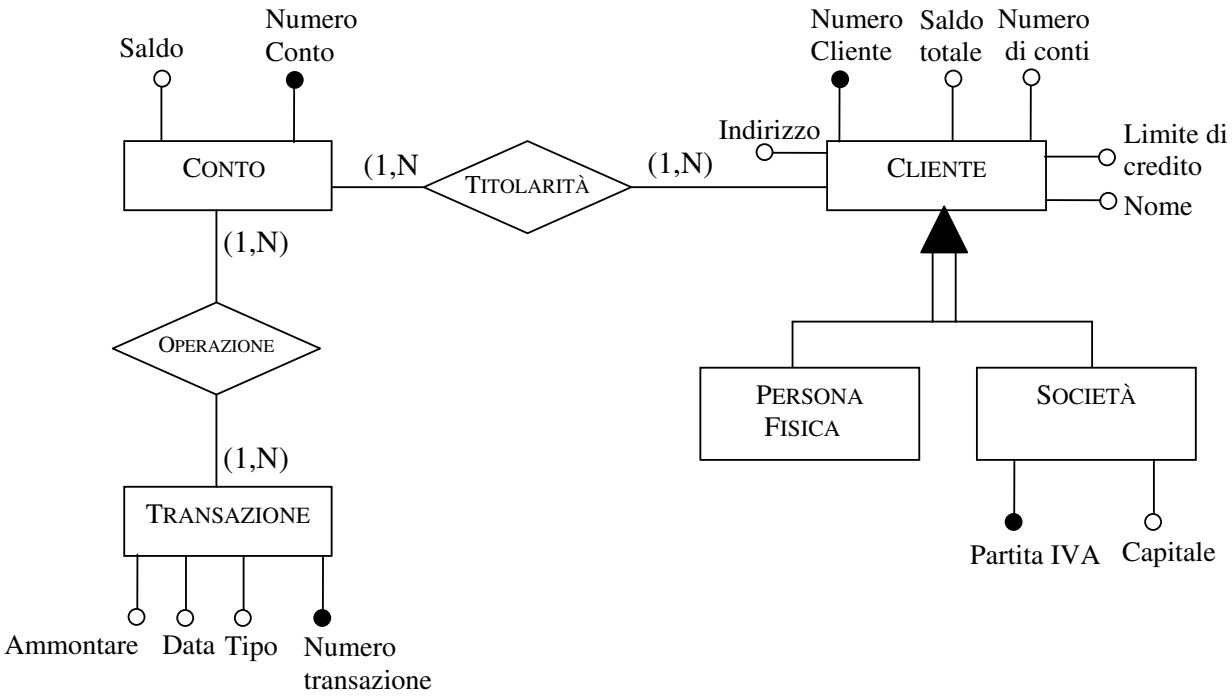



**Figura 9.38** Uno schema E-R da tradurre

**Tavola dei volumi**

| Concetto | Tipo | Volume |
|---|---|---|
| Cliente | E | 15000 |
| Conto | E | 20000 |
| Transazione | E | 600000 |
| Persona Fisica | E | 14000 |
| Società | E | 1000 |
| Titolarità | R | 30000 |
| Operazione | R | 800000 |

**Tavola delle operazioni**

| Operazione | Tipo | Frequenza |
|---|---|---|
| Op. 1 | I | 100/giorno |
| Op. 2 | I | 2000/giorno |
| Op. 3 | I | 1000/giorno |
| Op. 4 | I | 2000/giorno |
| Op. 5 | I | 1000/giorno |
| Op. 6 | I | 200/giorno |
| Op. 7 | B | 1500/giorno |
| Op. 8 | B | 1/mese |
| Op. 9 | B | 75/giorno |
| Op. 10 | I | 20/giorno |

**Tavola 9.39** Tavole dei volumi e delle operazioni per lo schema in figura 8.38

**Soluzione:**

**Analisi delle ridondanze**:
Nello schema esistono 2 dati ridondanti: **Saldo totale** e **Numero di Conti**.

Saldo totale:
Ipotizzando che l'attributo saldo totale sia di tipo float (32 bit e quindi 4 bytes per ogni occorrenza), l'utilizzo di questo dato richiederebbe 4*15.000 bytes, con un utilizzo di memoria pari a 60 KB.
Le operazioni coinvolte con questo dato sono la 2, la 4, la 5, la 7 e la 8.
Si procede analizzando il costo per ognuna di queste operazioni, non conteggiando l'operazione 8 che viene svolta in batch una sola volta al mese.

- Con dato ridondante
  Per l'operazione 2 abbiamo: 1x2.000 accessi in lettura = 2.000 accessi al giorno
  Per l'operazione 4 abbiamo: 1x2.000 accessi in lettura (leggo il saldo totale) + 2x2.000 accessi in scrittura (scrivo il nuovo saldo) = 6.000 accessi al giorno
  Per l'operazione 5 abbiamo: 1x1.000 accessi in lettura (leggo il saldo totale) + 2x1.000 accessi in scrittura (scrivo il nuovo saldo) = 3.000 accessi al giorno
  Per l'operazione 7 abbiamo: 1x1.500 accessi in lettura (leggo il saldo totale) + 2x1.500 accessi in scrittura (scrivo il nuovo saldo) = 4.500 accessi al giorno

- Senza dato ridondante
  Per l'operazione 2 abbiamo: ipotizzando che la query di ricerca dei conti di un cliente abbia un costo pari a 20 accessi in lettura, moltiplicato per 2.000 operazioni al giorno, ottengo 40.000 accessi al giorno.
  Per l'operazione 4 abbiamo: Nessun accesso aggiuntivo
  Per l'operazione 5 abbiamo: Nessun accesso aggiuntivo
  Per l'operazione 7 abbiamo: Nessun accesso aggiuntivo

In conclusione, il dato ridondante ho 15.000 accessi, mentre senza il dato ridondante ho 40.000 accessi al giorno. Il dato ridondante mi fa risparmiare 25.000 accessi a fronte di 60 KB di memoria.

Numero di conti:
Per quanto riguarda il numero di conti posseduto da un cliente, l'utilizzo di memoria col dato ridondante è di 1 byte per cliente, che equivale a 15 KB di memoria.
Le operazioni coinvolte sono la 1 e la 9.
Anche senza svolgere i conti, si può notare che l'utilizzo del dato è di sole 75 volte al giorno e in modalità batch con l'operazione 9.
Il conseguente miglioramento di efficienza sarà nell'ordine di un migliaio di accessi in meno al giorno. Sarà quindi a discrezione del progettista l'utilizzo o meno del dato.
In questo caso ipotizziamo quindi di non ritenerlo necessario.

**Eliminazione delle gerarchie**:

Nello schema è presente una sola gerarchia relativa all'entità CLIENTE, che viene distinto in PERSONA FISICA o SOCIETÀ. L'entità SOCIETÀ ha gli attributi Partita IVA e Capitale che la distinguono. L'unica operazione che fa una distinzione sul tipo di cliente è la numero 10.
Visto lo scarso numero di operazioni e il poco spazio necessario per accorpare le due entità, si decide di accorpare gli attributi Partita IVA e Capitale in Cliente. Sarà l'attributo Partita IVA ad identificare un cliente come società.

**Scelta degli identificatori principali**:

Gli identificatori sono Numero transazione per l'entità TRANSAZIONE, Numero Conto per l'entità CONTO.
Per quanto riguarda l'entità CLIENTE, l'identificatore è l'attributo Numero cliente; l'attributo Partita Iva identifica le società e se presente deve essere univoco.

**Schema relazionale**

TRANSAZIONE(Numero transazione, Tipo, Data, Ammontare)
CONTO(Numero Conto, Saldo)
CLIENTE(Numero cliente, Saldo Totale, Limite di credito, Nome, Indirizzo, Partita IVA*, Capitale*)
OPERAZIONE(Numero conto, Numero transazione)
TITOLARITÀ(Numero conto, Numero cliente)

# Capitolo 10

## Esercizio 10.1

Considerare lo relazione in figura 10.19 e individuare le proprietà della corrispondente applicazione. Individuare inoltre eventuali ridondanze e anomalie nella relazione.

| Docente | Dipartimento | Facoltà | Preside | Corso |
|---|---|---|---|---|
| Verdi | Matematica | Ingegneria | Neri | Analisi |
| Verdi | Matematica | Ingegneria | Neri | Geometria |
| Rossi | Fisica | Ingegneria | Neri | Analisi |
| Rossi | Fisica | Scienze | Bruni | Analisi |
| Bruni | Fisica | Scienze | Bruni | Fisica |

**Figura 10.19** Relazione per l'esercizio 10.1

**Soluzione:**

Una chiave per questa relazione è **Dipartimento, Facoltà, Corso**; una dipendenza funzionale che non riguarda la chiave **Facoltà** $\rightarrow$ **Preside**: questa dipendenza funzionale introduce una ridondanza nella relazione, perché per ogni corso nella stessa Facoltà, il Preside deve essere ripetuto.
La relazione ha un'anomalia di aggiornamento, perché se cambiamo il preside di una facoltà dobbiamo aggiornare tutte le tuple che contengono questa informazione, e non solamente una tupla.
La relazione contiene anche un'anomalia di cancellazione, perché se cancelliamo il preside di una facoltà, perdiamo anche tutte le informazioni sui docenti di quel dipartimento.

## Esercizio 10.2

Individuare la chiave e le dipendenze funzionali della relazione considerata nell'Esercizio 10.1 e individuare poi una decomposizione in forma normale di Boyce e Codd.

**Soluzione:**

Una chiave per questa relazione è **Dipartimento, Facoltà, Corso.**
Anche gli attributi **Docente, Facoltà, Corso** sembrano formare una chiave in questa relazione, ma generalmente parlando questo non è corretto perché lo stesso docente può insegnare lo stesso corso in differenti dipartimenti di una Facoltà.

Decomposizione:

| Docente | Dipartimento | Facoltà | Corso |
|---|---|---|---|
| Verdi | Matematica | Ingegneria | Analisi |
| Verdi | Matematica | Ingegneria | Geometria |
| Rossi | Fisica | Ingegneria | Analisi |
| Rossi | Fisica | Scienze | Analisi |
| Bruni | Fisica | Scienze | Fisica |

| Facoltà | Preside |
|---|---|
| Ingegneria | Neri |
| Scienze | Bruni |

Questa decomposizione è corretta perché, con un join tra le due relazioni, otteniamo tutte le tuple della relazione originaria.
Inoltre, la decomposizione risolve il problema delle anomalie, perché è in forma normale di Boyce-Codd.

## Esercizio 10.3

Si consideri la relazione riportata in figura 10.20 che rappresenta alcune informazioni sui prodotti di una falegnameria e i relativi componenti. Vengono indicati: il tipo del componente di un prodotto (attributo **Tipo**), la quantità del componente necessaria per un certo prodotto (attributo **Q**), il prezzo unitario del componente di un certo prodotto (attributo **PC**), il fornitore del componente (attributo **Fornitore**) e il prezzo totale del singolo prodotto (attributo **PT**). Individuare le dipendenze funzionali e la chiave di questa relazione.

| Prodotto | Componente | Tipo | Q | PC | Fornitore | PT |
|---|---|---|---|---|---|---|
| Libreria | Legno | Noce | 50 | 10.000 | Forrest | 400.000 |
| Libreria | Bulloni | B212 | 200 | 100 | Bolt | 400.000 |
| Libreria | Vetro | Cristal | 3 | 5.000 | Clean | 400.000 |
| Scaffale | Legno | Mogano | 5 | 15.000 | Forrest | 300.000 |
| Scaffale | Bulloni | B212 | 250 | 100 | Bolt | 300.000 |
| Scaffale | Bulloni | B412 | 150 | 300 | Bolt | 300.000 |
| Scrivania | Legno | Noce | 10 | 8.000 | Wood | 250.000 |
| Scrivania | Maniglie | H621 | 10 | 20.000 | Bolt | 250.000 |
| Tavolo | Legno | Noce | 4 | 10.000 | Forrest | 200.000 |

**Figura 10.20** Una relazione contenente dati di una falegnameria

**Soluzione:**

Supponendo che un Tipo si riferisca solamente ad un componente, una chiave per la relazione è **Prodotto, Tipo;** così tutti gli attributi che contengono **Prodotto, Tipo** sono superchiavi per la relazione.
Gli attributi **Q** e **PC** sembrano un'altra chiave, ma potrebbe non essere vero in tutte le istanze di questo database.
Un'altra chiave apparente è **Tipo, PT**

Le dipendenze funzionali sono:

- **Prodotto → PT**
- **Tipo, Fornitore → PC**
- **Tipo → Componente**

## Esercizio 10.4

Con riferimento alla relazione in Figura 10.20 si considerino le seguenti operazioni di aggiornamento:

- Inserimento di un nuovo prodotto;
- Cancellazione di un prodotto;
- Aggiunta di una componente a un prodotto;
- Modifica del prezzo di un prodotto.

Discutere i tipi di anomalia che possono essere causati da tali operazioni.

**Soluzione:**

1) L'inserimento di un nuovo prodotto richiede l'aggiunta di una tupla per ogni tipo di componente. Il prezzo del componente, che è in funzione del prodotto, deve essere ripetuto in ogni tupla.. Anche il prezzo di un componente può essere ridondante perché lo stesso tipo di componente, con lo stesso fornitore è usato per altri prodotti, il prezzo del componente è già presente nella relazione. Questa è un'anomalia di inserimento.
2) La cancellazione di un prodotto implica che tutte le tuple che si riferiscono al prodotto devono essere cancellate; così se un prodotto ha più di un componente, la cancellazione di un prodotto implica la cancellazione di molte tuple; inoltre questa operazione cancella informazioni sui fornitori di componenti: se non ci sono altre tuple che si riferiscono a quei fornitori, le informazioni su di loro andranno perse. Questa è un'anomalia di cancellazione.
3) L'aggiunta di un nuovo componente implica l'aggiunta di una nuova tupla nella relazione. Questa è un'altra anomalia di aggiornamento perché, come per il punto 1, il prezzo totale e (eventualmente) il prezzo del componente devono essere ripetuti.
4) La modifica del prezzo di un prodotto produce un'anomalia di aggiornamento, perché l'aggiornamento di un attributo implica l'aggiornamento di più tuple nella relazione (una tupla per ogni tipo di componente dello stesso prodotto).

## Esercizio 10.5

Si consideri sempre la relazione in figura 10.20. Descrivere le ridondanze presenti e individuare una decomposizione della relazione che non presenti tali ridondanze. Fornire infine l'istanza dello schema così ottenuto, corrispondente all'istanza originale. Verificare poi che sia possibile ricostruire l'istanza originale a partire da tale istanza.

**Soluzione:**

Le ridondanze presenti nella relazione sono riferite alle dipendenze funzionali. Gli attributi ridondanti sono:

- **PT**: che è ripetuto in ogni tupla che si riferisce allo stesso prodotto.
- **PC**: che è ripetuto in ogni tupla che ha lo stesso valore in **Tipo** e **Fornitore**.
- **Componente**: che è ripetuto in ogni tupla che ha lo stesso **Tipo**.

Una possibile decomposizione è:

R1

| Prodotto | Tipo | Q | Fornitore |
|---|---|---|---|
| Libreria | Noce | 50 | Forrest |
| Libreria | B212 | 200 | Bolt |
| Libreria | Cristal | 3 | Clean |
| Scaffale | Mogano | 5 | Forrest |
| Scaffale | B212 | 250 | Bolt |
| Scaffale | B412 | 150 | Bolt |
| Scrivania | Noce | 10 | Wood |
| Scrivania | H621 | 10 | Bolt |
| Tavolo | Noce | 4 | Forrest |

R2

| Prodotto | PT |
|---|---|
| Libreria | 400.000 |
| Scaffale | 300.000 |
| Scrivania | 250.000 |
| Tavolo | 200.000 |

R3

| Tipo | Componente |
|---|---|
| Noce | Legno |
| B212 | Bulloni |
| B412 | Bulloni |
| Cristal | Vetro |
| Mogano | Legno |
| H621 | Maniglie |

R4

| Fornitore | Tipo | PC |
|---|---|---|
| Forrest | Noce | 10.000 |
| Bolt | B212 | 100 |
| Clean | Cristal | 5.000 |
| Forrest | Mogano | 15.000 |
| Bolt | B412 | 300 |
| Wood | Noce | 8.000 |
| Bolt | H621 | 20.000 |

La relazione R1 ha la chiave originaria della relazione, ma non contiene ridondanze. Le relazioni R2, R3 e R4 hanno le chiavi sul lato sinistro delle dipendenze funzionali (vedi Esercizio 10.3).
Facendo il join su queste chiavi si possono ricostruire esattamente le informazioni dello schema originario.
Tutte le dipendenze sono preservate nella decomposizione, perché ognuna di loro è rappresentata con una relazione differente.

## Esercizio 10.6

Si consideri lo schema della relazione in figura 10.21: La chiave di questa relazione è costituita dagli attributi Titolo e Copia, e su di essa è definita la dipendenza **Titolo → Autore Genere**. Verificare se lo schema è o meno in terza forma normale e, in caso negativo, decomporlo opportunamente.

| Titolo | Autore | Genere | Copia | Scaffale |
|---|---|---|---|---|
| Decamerone | Boccaccia | Novelle | 1 | A75 |
| Divina Commedia | Dante | Poema | 1 | A90 |
| Divina Commedia | Dante | Poema | 2 | A90 |
| I Malavoglia | Verga | Romanzo | 1 | A90 |
| I Malavoglia | Verga | Romanzo | 2 | A75 |
| I Promessi Sposi | Manzoni | Romanzo | 1 | B10 |
| Adelchi | Manzoni | Tragedia | 1 | B20 |

**Figura 10.21** Relazione per l'Esercizio 10.6

**Soluzione:**

La relazione non è in terza forma normale perché il lato destro della dipendenza funzionale **Titolo → Autore Genere** non è parte della chiave.
Una possibile decomposizione è:

R1

| Titolo | Copia | Scaffale |
|---|---|---|
| Decamerone | 1 | A75 |
| Divina Commedia | 1 | A90 |
| Divina Commedia | 2 | A90 |
| I Malavoglia | 1 | A90 |
| I Malavoglia | 2 | A75 |
| I Promessi Sposi | 1 | B10 |
| Adelchi | 1 | B20 |

R2

| Titolo | Autore | Genere |
|---|---|---|
| Decamerone | Boccaccia | Novelle |
| Divina Commedia | Dante | Poema |
| I Malavoglia | Verga | Romanzo |
| I Promessi Sposi | Manzoni | Romanzo |
| Adelchi | Manzoni | Tragedia |

La relazione è in forma normale di Boyce-Codd, perché la chiave per R2 è **Titolo**, che è anche il lato sinistro della dipendenza funzionale.

## Esercizio 10.7

Si consideri lo schema Entità-Relazione in figura 10.22. Sui dati descritti da questo schema valgono le seguenti proprietà:

- Un giocatore può giocare per una sola squadra (o per nessuna);
- Un allenatore può allenare una sola squadra (o nessuna);
- Una squadra ha un solo allenatore, diversi giocatori e appartiene a un'unica città.

Verificare se lo schema soddisfa la forma normale di Boyce-Codd e, in caso negativo, ristrutturarlo in un nuovo schema in maniera che soddisfi tale forma normale.

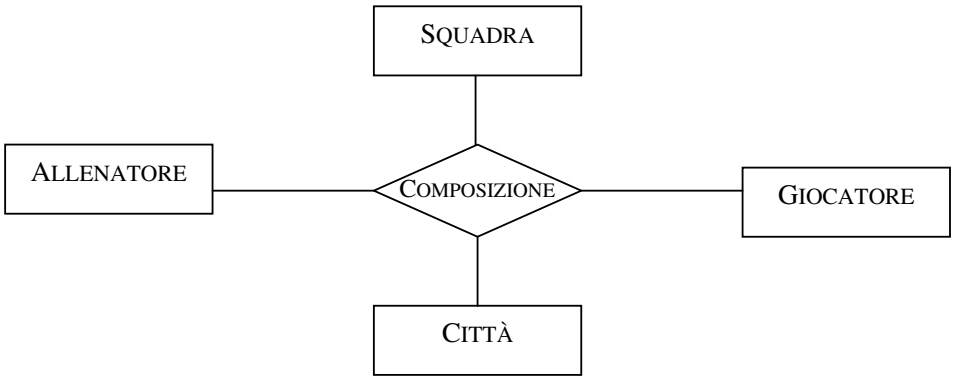



**Figura 10.22** Uno schema da sottoporre alla verifica di normalizzazione

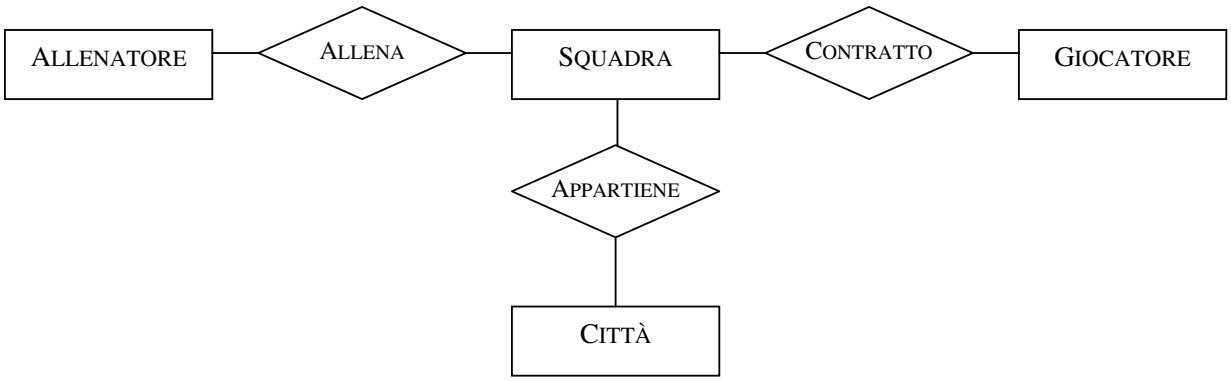
**Soluzione:**

Le dipendenze funzionali presenti nello schema sono:

- **Giocatore → Squadra**.
- **Allenatore → Squadra**
- **Squadra → Città**

La chiave per la relazione Composizione è Giocatore e così lo schema non è in forma normale di Boyce-Codd.

Una possibile ristrutturazione è:




In questo nuovo schema ci sono solamente relazioni binarie, e quindi rispetta la forma normale di Boyce-Codd.

## Esercizio 10.8

Consideriamo la relazione in figura 10.23 e le sue seguenti possibile decomposizioni:

- **Reparto, Cognome** in una relazione e **Cognome, Nome, Indirizzo** nell'altra;
- **Reparto, Cognome, Nome** in una relazione e **Nome, Indirizzo** nell'altra;
- **Reparto, Cognome, Nome** in una relazione e **Cognome, Nome, Indirizzo** nell'altra;

Individuare, con riferimento sia all'istanza specifica sia all'insieme delle istanze sullo stesso schema (con le proprietà naturalmente associate), quali di tali decomposizioni sono senza perdita.

| Reparto | Cognome | Nome | Indirizzo |
|---|---|---|---|
| Vendite | Rossi | Mario | Via Po 20 |
| Acquisti | Rossi | Mario | Via Po 20 |
| Bilancio | Neri | Luca | Via Taro 12 |
| Personale | Rossi | Luigi | Via Taro 12 |

**Figura 10.23** Relazione per l'Esercizio 10.8

**Soluzione:**

La chiave di questa relazione è **Reparto**. Assumiamo che le persone siano identificate dal Cognome e dal Nome.
La relazione ha una dipendenza funzionale: **Cognome, Nome** → **Indirizzo**

1) Questa soluzione non è corretta in generale; il join tra le due relazioni produce informazioni spurie. Infatti l'attributo **Cognome** non identifica una persona, e il join associerà a un Reparto tutte le persone con lo stesso cognome. Con questa istanza otterremo:

| Reparto | Cognome |
|---|---|
| Vendite | Rossi |
| Acquisti | Rossi |
| Bilancio | Neri |
| Personale | Rossi |

| Cognome | Nome | Indirizzo |
|---|---|---|
| Rossi | Mario | Via Po 20 |
| Neri | Luca | Via Taro 12 |
| Rossi | Luigi | Via Taro 12 |

| Reparto | Cognome | Nome | Indirizzo |
|---|---|---|---|
| Vendite | Rossi | Mario | Via Po 20 |
| Vendite | Rossi | Luigi | Via Taro 12 |
| Acquisti | Rossi | Mario | Via Po 20 |
| Acquisti | Rossi | Luigi | Via Taro 12 |
| Bilancio | Neri | Luca | Via Taro 12 |
| Personale | Rossi | Luigi | Via Taro 12 |
| Personale | Rossi | Mario | Via Po 20 |

2) Questa decomposizione è corretta in questa particolare istanza del database, perché non ci sono due persone con lo stesso nome e così il join tra le due relazioni da ancora la relazione originaria, ma generalmente parlando il **Nome** non identifica una persona e il loro join può dare una relazione con delle informazioni spurie.

3) Questa decomposizione è sempre corretta perché entrambi gli attributi **Cognome** e **Nome** sono presenti nelle relazioni, e la seconda relazione ha **Cognome** e **Nome** come chiave.
Questa decomposizione produce un database in forma normale di Boyce-Codd.

| Reparto | Cognome | Nome |
|---|---|---|
| Vendite | Rossi | Mario |
| Acquisti | Rossi | Mario |
| Bilancio | Neri | Luca |
| Personale | Rossi | Luigi |

| Cognome | Nome | Indirizzo |
|---|---|---|
| Rossi | Mario | Via Po 20 |
| Neri | Luca | Via taro 12 |
| Rossi | Luigi | Via taro 12 |

| Reparto | Cognome | Nome | Indirizzo |
|---|---|---|---|
| Vendite | Rossi | Mario | Via Po 20 |
| Acquisti | Rossi | Mario | Via Po 20 |
| Bilancio | Neri | Luca | Via taro 12 |
| Personale | Rossi | Luigi | Via taro 12 |

## Esercizio 10.9

Consideriamo nuovamente la relazione in figura 10.23. Individuare quali delle seguenti decomposizioni conservano le sue dipendenze:

- Una relazione sugli attributi **Reparto, Cognome** e **Nome** e l'altra sugli attributi **Cognome** e **Indirizzo**.
- Una relazione su **Reparto, Cognome** e **Nome** e l'altra su **Cognome, Nome** e **Indirizzo.**
- Una relazione su **Reparto** e **Indirizzo** e l'altra su **Reparto, Cognome** e **Nome**.

**Soluzione:**

1) Questa decomposizione non conserva la dipendenza **Cognome, Nome** → **Indirizzo,** perché gli attributi coinvolti sono suddivisi tra le due relazioni. Così, se abbiamo bisogno, per esempio, di cambiare l'indirizzo riferito al reparto "Vendite", possiamo fare questa operazione soltanto cambiando tutte le tuple della seconda relazione che hanno come Cognome "Rossi"; ma questo non è corretto perché ci sono due persone con lo stesso cognome e solo una è riferita al reparto "Vendite". Anche questa decomposizione non è corretta.

2) Questa decomposizione è corretta, perché la seconda relazione contiene tutti gli attributi della dipendenza funzionale.

3) Questa decomposizione è sbagliata, perché come nel punto 1, gli attributi della dipendenza funzionale sono divisi in due relazioni; in questo caso è possibile aggiungere tuple nella prima relazione, associando ai reparti un indirizzo che non si riferisce alla persona corretta (mentre è impossibile nella relazione originaria).