ESERCIZI DI BASI DI DATI
SQL
(draft)

prof. Angelo Montanari
Michela Zamparini
Alberto Guglielmetti

3 maggio 2001

Indice

Riferimenti bibliografici:

- R. Elmasri, S. Navathe, *Fundamentals of Database Systems*, 3^{rd} ed., Addison-Wesley, 2000;
- P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone, *Basi di Dati*, 2^{a} ed., McGraw-Hill, 1999.

1 SQL

Esercizio 1.1 Sia dato il seguente schema relazionale che descrive il calendario di una manifestazione sportiva a squadre nazionali:

PALAZZETTO(<u>Nome</u>, Città, Capienza)

INCONTRO(<u>NomePalazzetto,Data,Ora</u>, Squadra1, Squadra2)

NAZIONALE(<u>Nazione</u>, Continente, Livello).

Esprimere in SQL le seguenti interrogazioni (senza usare l'operatore CONTAINS e usando le funzioni aggregate solo se necessario):

(a) determinare i nomi dei palazzetti in cui non gioca nessuna nazionale asiatica;

(b) determinare la capienza complessiva dei palazzetti in cui si giocano partite di nazionali africane (ai fini della valutazione della capienza complessiva, si sommino le capienze associate a ciascuna gara, anche se più gare si svolgono nello stesso palazzetto);

(c) individuare la città (le città, se più d'una soddisfa le condizioni dell'interrogazione) in cui si trova il palazzetto in cui la squadra olandese gioca il maggior numero di partite;

(d) determinare le squadre che incontrano solo squadre dello stesso livello;

(e) determinare le squadre che giocano esattamente negli stessi palazzetti.

Soluzione.

(a)

```
SELECT Nome
FROM   PALAZZETTO
WHERE  Nome NOT IN (SELECT NomePalazzetto
                    FROM   INCONTRO, NAZIONALE
                    WHERE  Continente = 'Asia' AND (Squadra1 = Nazione
                           OR Squadra2 = Nazione));
```

(b)

```
SELECT SUM(Capienza)
FROM   PALAZZETTO JOIN INCONTRO ON Nome = NomePalazzetto
WHERE  Squadra1 IN (SELECT Nazione
                    FROM   NAZIONALE
                    WHERE  Continente = 'Africa')
       OR
       Squadra2 IN (SELECT Nazione
                    FROM   NAZIONALE
                    WHERE  Continente = 'Africa');
```

Soluzione alternativa:

```
SELECT SUM(Capienza)
FROM   PALAZZETTO INCONTRO NAZIONALE
WHERE  Nome = NomePalazzetto AND Continente = 'Africa'
       AND (Squadra1 = Nazione OR Squadra2 = Nazione);
```

(c)

```
CREATE VIEW OLANDA(Nome, Num_Partite)
  AS
  SELECT   NomePalazzetto, COUNT(*)
  FROM     INCONTRO
  WHERE    Squadra1 = 'Olanda' OR Squadra2 = 'Olanda'
  GROUP BY NomePalazzetto;

SELECT DISTINCT Città
FROM   PALAZZETTO NATURAL JOIN OLANDA
WHERE  Num_Partite >= ALL (SELECT Num_Partite
                           FROM   OLANDA);
```

L'opzione `DISTINCT` è necessaria perché ci possono essere più palazzetti, appartenenti alla stessa città, che soddisfano le condizioni dell'interrogazione.

(d)

```
SELECT Nazione
FROM   NAZIONALE N
WHERE  NOT EXISTS (SELECT *
                   FROM   INCONTRO, NAZIONALE
                   WHERE  Squadra1 = N.Nazione AND Squadra2 = Nazione
                   AND N.Livello <> Livello)
       AND
       NOT EXISTS (SELECT *
                   FROM   INCONTRO, NAZIONALE
                   WHERE  Squadra2 = N.Nazione AND Squadra1 = Nazione
                   AND N.Livello <> Livello);
```

(e)

```
SELECT N1.Nazione, N2.Nazione
FROM   NAZIONALE AS N1, NAZIONALE AS N2
WHERE  NOT EXISTS (SELECT *
                   FROM   INCONTRO I
                   WHERE  (I.Squadra1 = N1.Nazione OR I.Squadra2 = N1.Nazione)
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   INCONTRO
                                      WHERE  NomePalazzetto = I.NomePalazzetto
                                             AND
                                             (Squadra1 = N2.Nazione
                                              OR
                                              Squadra2 = N2.Nazione)))
       AND
       NOT EXISTS (SELECT *
                   FROM   INCONTRO I
                   WHERE  (I.Squadra1 = N2.Nazione OR I.Squadra2 = N2.Nazione)
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   INCONTRO
                                      WHERE  NomePalazzetto = I.NomePalazzetto
                                             AND
                                             (Squadra1 = N1.Nazione
                                              OR
                                              Squadra2 = N1.Nazione)))
       AND
       N1.Nazione <> N2.Nazione;
```

□

Esercizio 1.2 Sia dato il seguente schema relazionale:

FREQUENTA(Bambino, Gelateria)

OFFRE(Gelateria, Gusto, Quantità)

PIACE_A(Bambino, Gusto).

Si assuma che l'attributo Quantità di OFFRE indichi la quantità media di un certo gusto venduta giornalmente dalla GELATERIA. Determinare le possibili chiavi delle relazioni date ed esprimere in SQL le seguenti interrogazioni (senza usare l'operatore CONTAINS e usando le funzioni aggregate solo se necessario):

(a) determinare le gelaterie che offrono almeno un gusto originale (un gusto si dice originale se è fornito da una sola gelateria);

(b) determinare le gelaterie che offrono solo gusti originali;

(c) determinare la quantità media di ogni gusto venduta dalle gelaterie, escludendo le gelaterie che non offrono il gusto;

(d) determinare la quantità massima di ogni gusto venduta da una gelateria, limitatamente ai gusti non originali;

(e) determinare le gelaterie che offrono solo gusti che piacciono a tutti i bambini che le frequentano.

Soluzione. Le chiavi delle relazioni sono le seguenti:

- **Bambino** e **Gelateria** per FREQUENTA;
- **Gelateria** e **Gusto** per OFFRE;
- **Bambino** e **Gusto** per PIACE_A.

(a)
```
SELECT DISTINCT Gelateria
FROM   OFFRE AS O
WHERE  NOT EXISTS (SELECT *
                   FROM   OFFRE
                   WHERE  O.Gelateria <> Gelateria
                          AND
                          O.Gusto = Gusto);
```

Soluzione alternativa:

```
SELECT DISTINCT Gelateria
FROM   OFFRE AS O
WHERE  Gusto NOT IN (SELECT Gusto
                     FROM   OFFRE
                     WHERE  O.Gelateria <> Gelateria);
```

(b) Forniamo tre possibili soluzioni.

- ```
  SELECT DISTINCT Gelateria
  FROM   OFFRE
    EXCEPT
  SELECT Gelateria
  FROM   OFFRE AS O
  WHERE  EXISTS (SELECT *
                 FROM   OFFRE
                 WHERE  O.Gelateria <> Gelateria
                        AND
                        O.Gusto = Gusto);
  ```
  ```

- ```
  SELECT DISTINCT Gelateria
  FROM   OFFRE AS O
  WHERE  NOT EXISTS (SELECT *
                     FROM   OFFRE O1, OFFRE O2
                     WHERE  O1.Gelateria = O.Gelateria
                            AND
                            O1.Gelateria <> O2.Gelateria
                            AND
                            O1.Gusto = O2.Gusto);
  ```
- ```
  SELECT DISTINCT Gelateria
  FROM   OFFRE AS O
  WHERE  NOT EXISTS ((SELECT Gusto
                      FROM   OFFRE
                      WHERE  O.Gelateria = Gelateria)
                       INTERSECT
                     (SELECT Gusto
                      FROM   OFFRE
                      WHERE  O.Gelateria <> Gelateria));
  ```

(c)

```
SELECT   Gusto, AVG(Quantità)
FROM     OFFRE
GROUP BY Gusto;
```

(d)

```
SELECT DISTINCT Gusto, Quantità
FROM   OFFRE AS O
WHERE  EXISTS (SELECT *                                           #Gusti non originali
               FROM   OFFRE
               WHERE  O.Gelateria <> Gelateria
                      AND
                      O.Gusto = Gusto)
       AND
       Quantità >= ALL (SELECT Quantità                            #Quantità massima
                        FROM   OFFRE
                        WHERE  O.Gusto = Gusto);
```

(e)

```
SELECT DISTINCT Gelateria
FROM   OFFRE AS O
WHERE  NOT EXISTS (SELECT *
                   FROM   FREQUENTA AS F
                   WHERE  O.Gelateria = Gelateria
                          AND
                          EXISTS (SELECT *
                                  FROM   OFFRE
                                  WHERE  O.Gelateria = Gelateria
                                         AND
                                         Gusto NOT IN (SELECT Gusto
                                                       FROM   PIACE_A
                                                       WHERE  Bambino = F.Bambino)));
```

Oppure:

```
SELECT DISTINCT Gelateria
FROM   OFFRE AS A
WHERE  NOT EXISTS (SELECT *                             #Non esiste un gusto tale che...
```

```
FROM   OFFRE AS B
WHERE  A.Gelateria = B.Gelateria
       AND
       EXISTS (SELECT *               #Esiste un bimbo tale che...
               FROM   (FREQUENTA NATURAL JOIN OFFRE) AS C
               WHERE  C.Gelateria = B.Gelateria
                      AND
                      NOT EXISTS (SELECT *
                                  FROM   PIACE_A AS D
                                  WHERE  D.Bambino = C.Bambino
                                         AND
                                         D.Gusto = B.Gusto)));
```

□

Esercizio 1.3 Sia dato il seguente schema relazionale:

FORNITORE(<u>S#</u>, Fnome, Status, Città)

COMPONENTE(<u>C#</u>, Cnome, Colore, Peso, Città)

PROGETTO(<u>P#</u>, Pnome, Città)

FORNISCE(<u>S#, C#, P#</u>, Quantità).

Definiamo *originale* un prodotto fornito da un solo fornitore. Si esprimano in SQL i seguenti aggiornamenti:

(a) si inserisca un nuovo fornitore *S10* nella tabella FORNITORE. Il nome e la città siano rispettivamente *De Marco* e *Belluno*; lo status non sia ancora noto;

(b) si cambi il colore di tutti i componenti *rossi*, facendoli diventare *verdi*;

(c) si cancellino tutti i progetti che non vengono riforniti da alcun fornitore.

Si formulino in SQL le seguenti interrogazioni, senza usare l'operatore CONTAINS, ricorrendo alle funzioni aggregate solo se necessario e definendo, se opportuno, delle viste:

(d) determinare i nomi dei fornitori che non forniscono alcun componente originale;

(e) determinare i nomi dei fornitori che forniscono solo componenti originali;

(f) determinare le coppie di nomi di fornitori tali che l'intersezione dei componenti da loro forniti sia vuota;

(g) determinare le città in cui risiedono almeno due fornitori con status maggiore o uguale a 100, escludendo le città cui non è associato alcun progetto.

(h) determinare i nomi dei fornitori che forniscono almeno un componente originale;

(i) per ogni città, si determinino il massimo e il minimo dei pesi dei componenti ad esse associati;

(l) si determinino le città cui sono associati due o più componenti, ma non più di un progetto;

(m) si determinino i componenti (uno o più) di peso massimo e quelli (uno o più) di peso minimo;

(n) si determini lo status dei fornitori che riforniscono tutti i progetti cui viene fornito (da loro o da altri) almeno un componente di colore giallo.

Soluzione.

(a)

```
INSERT INTO FORNITORE (S#, Fnome, Città)
       VALUES ('S10', 'De Marco', 'Belluno');
```

(b)

```
UPDATE COMPONENTE
SET    Colore = 'verde'
WHERE  (SELECT *
        FROM   COMPONENTE
        WHERE  Colore = 'rosso');
```

Oppure (meglio):

```
UPDATE COMPONENTE
SET    Colore = 'verde'
WHERE  Colore = 'rosso';
```

(c) Una prima possibilità (un po' complicata):

```
DELETE FROM PROGETTO
WHERE  P# IN (SELECT P#
              FROM   PROGETTO AS PRO
              WHERE  NOT EXISTS (SELECT *
                                 FROM   FORNISCE AS FOR
                                 WHERE  PRO.P# = FOR.P#));
```

Oppure (meglio):

```
DELETE FROM PROGETTO
WHERE  P# NOT IN (SELECT P#
                  FROM   FORNISCE);
```

(d)

```
CREATE VIEW ORIGINALI(C#)
  AS
  SELECT F1.C#
  FROM   FORNISCE AS F1
  WHERE  NOT EXISTS (SELECT *
                     FROM   FORNISCE AS F2
                     WHERE  F2.C# = F1.C#
                            AND
                            F2.S# <> F1.S#);
```

L'opzione `DISTINCT` è necessaria perché, anche se originali, i componenti possono essere forniti a più progetti da uno stesso fornitore.

```
SELECT F1.S#, F1.Fnome
FROM   FORNITORE AS F1
WHERE  NOT EXISTS (SELECT *
                   FROM   FORNISCE AS F2, ORIGINALI
                   WHERE  F1.S# = F2.S#
                          AND
                          F2.C# = ORIGINALI.C#);
```

Oppure (è lo stesso):

```
SELECT F1.S#, F1.Fnome
FROM   FORNITORE AS F1
WHERE  NOT EXISTS (SELECT *
                   FROM   FORNISCE AS F2
                   WHERE  F1.S# = F2.S#
                          AND
                          F2.C# IN (SELECT *
                                    FROM   ORIGINALI));
```

In quest'ultimo caso si può evitare l'uso della vista: è sufficiente inserire la definizione della stessa al posto dell'ultima clausola `SELECT`.

(e)

```
SELECT Fnome
FROM   FORNITORE AS F
WHERE  NOT EXISTS (SELECT *
                   FROM   FORNISCE AS A, FORNISCE AS B
                   WHERE  A.S# = F.S#
                          AND
                          A.S# <> B.S#
                          AND
                          A.C# = B.C#);
```

Oppure:

```
SELECT Fnome
FROM   FORNITORE
WHERE  S# IN (SELECT S#
              FROM   FORNITORE
                EXCEPT
              SELECT A.S#
              FROM   FORNISCE AS A, FORNISCE AS B
              WHERE  A.S# <>B.S#
                     A.C# = B.C#);
```

(f)

```
SELECT A.Fnome, B.Fnome
FROM   FORNITORE AS A, FORNITORE AS B
WHERE  A.S# < B.S#
       AND
       NOT EXISTS (SELECT *
                   FROM   FORNISCE AS C, FORNISCE AS D
                   WHERE  C.S# = A.S#
                          AND
                          D.S# = B.S#
                          AND
                          C.C# = D.C#);
```

Oppure:

```
SELECT A.Fnome, B.Fnome
FROM   FORNITORE AS A, FORNITORE AS B
WHERE  A.S# < B.S#
       AND
       NOT EXISTS (SELECT *
                   FROM   COMPONENTE AS C
                   WHERE  EXISTS (SELECT *
                                  FROM   FORNISCE AS D
                                  WHERE  C.C# = D.C#
                                         AND
                                         A.S# = D.S#)
                          AND
                          EXISTS (SELECT *
                                  FROM   FORNISCE AS E
                                  WHERE  C.C# = E.C#
                                         AND
                                         B.S# = E.S#));
```

(g) Innanzitutto ci serve una vista:

```
CREATE VIEW CITTAP(Città)
  AS
  SELECT DISTINCT F.Città
  FROM   FORNITORE AS F
  WHERE  EXISTS (SELECT *
                 FROM   PROGETTO
                 WHERE  Città = F.Città);
```

Oppure, semplicemente:

```
CREATE VIEW CITTAP(Città)
  AS
  SELECT DISTINCT Città
  FROM   PROGETTO;
```

Quindi:

```
SELECT   DISTINCT Città
FROM     FORNITORE NATURAL JOIN CITTAP
WHERE    Status >= 100
GROUP BY Città
HAVING   COUNT(*) >= 2
```

Oppure, senza usare le funzioni aggregate né la vista:

```
SELECT DISTINCT Città
FROM   FORNITORE AS A, FORNITORE AS B
WHERE  A.Status >= 100
       AND
       B.Status >=100
       AND
       A.S# <> B.S#
       AND
       A.Città = B.Città
       AND
       Città IN (SELECT DISTINCT Città
                 FROM   PROGETTO);
```

(h) Riportiamo tre possibili soluzioni.

- ```
  SELECT DISTINCT Fnome
  FROM   FORNITORE NATURAL JOIN FORNISCE AS A
  WHERE  NOT EXISTS (SELECT *
                     FROM   FORNISCE
                     WHERE  S# <> A.S#
                            AND
                            C# = A.C#);
  ```
- ```
  SELECT DISTINCT Fnome
  FROM   FORNITORE NATURAL JOIN FORNISCE
  WHERE  C# NOT IN (SELECT A.C#
                    FROM   FORNISCE AS A, FORNISCE AS B
                    WHERE  A.S# <> B.S#
                           AND
                           A.C# = B.C#);
  ```

- ```
  SELECT Fnome
  FROM   FORNITORE
  WHERE  S# IN (SELECT S#
                FROM   FORNISCE
                WHERE  C# IN (SELECT C#
                              FROM   FORNISCE AS A
                              WHERE  NOT EXISTS (SELECT *
                                                 FROM   FORNISCE
                                                 WHERE  S# <> A.S#
                                                        AND
                                                        C# = A.C#)));
  ```

(i)

```
SELECT DISTINCT Città, Peso
FROM   COMPONENTE AS A
WHERE  NOT EXISTS (SELECT *
                   FROM   COMPONENTE
                   WHERE  A.Città = Città
                          AND
                          A.Peso < Peso)
  UNION
SELECT DISTINCT Città, Peso
FROM   COMPONENTE AS A
WHERE  NOT EXISTS (SELECT *
                   FROM   COMPONENTE
                   WHERE  Città = A.Città
                          AND
                          Peso < A.Peso);
```

Oppure:

```
SELECT DISTINCT Città, Peso
FROM   COMPONENTE AS A
WHERE  Peso >= ALL (SELECT Peso
                    FROM   COMPONENTE
                    WHERE  A.Città = Città)
  UNION
SELECT DISTINCT Città, Peso
FROM   COMPONENTE AS A
WHERE  Peso <= ALL (SELECT Peso
                    FROM   COMPONENTE
                    WHERE  A.Città = Città);
```

Se il massimo ed il minimo coincidono per una certa città, sarà presente una sola tupla anziché due. Una terza soluzione possibile è la seguente:

```
SELECT A.Città, A.Peso, B.Peso
FROM   COMPONENTE AS A, COMPONENTE AS B
WHERE  A.Peso <= ALL (SELECT Peso
                      FROM   COMPONENTE
                      WHERE  A.Città = Città)
       AND
       B.Peso >= ALL (SELECT Peso
                      FROM   COMPONENTE
                      WHERE  B.Città = Città)
       AND
       A.Città = B.Città;
```
```

(l)

```
SELECT A.Città
FROM   COMPONENTE AS A, COMPONENTE AS B
WHERE  A.Città = B.Città
       AND
       A.C# <> B.C#
  EXCEPT
SELECT A.Città
FROM   PROGETTO AS A, PROGETTO AS B
WHERE  A.Città = B.Città
       AND
       A.P# <> B.P#;
```

Soluzione alternativa:

```
SELECT A.Città
FROM   COMPONENTE AS A, COMPONENTE AS B
WHERE  A.Città = B.Città
       AND
       A.C# <> B.C#
  INTERSECT
SELECT Città
FROM   PROGETTO AS A
WHERE  NOT EXISTS (SELECT *
                   FROM   PROGETTO
                   WHERE  A.Città = Città
                          AND
                          A.P# <> P.#);
```

(m)

```
SELECT C#, Peso
FROM   COMPONENTE
WHERE  Peso >= ALL (SELECT Peso
                    FROM   COMPONENTE)
       OR
       Peso <= ALL (SELECT Peso
                    FROM   COMPONENTE);
```

(n)

```
SELECT S#, Status
FROM   FORNITORE AS B
WHERE  NOT EXISTS (SELECT *
                   FROM   (COMPONENTE NATURAL JOIN FORNISCE) AS C
                   WHERE  Colore = 'giallo'
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   FORNISCE
                                      WHERE  S# = B.S#
                                             AND
                                             P# = C.P#));
```

Oppure:

```
SELECT S#, Status
FROM   FORNITORE AS F
WHERE  NOT EXISTS (SELECT P#
                   FROM   FORNISCE NATURAL JOIN COMPONENTE
```

```
WHERE  Colore = 'giallo'
  EXCEPT
SELECT P#
FROM   FORNISCE
WHERE  F.S# = S#);
```

□

Esercizio 1.4 Sia dato il seguente schema relazionale:

LIBRO(<u>ISBN</u>, Autore, Titolo)

IN_VENDITA(<u>Libreria, ISBN</u>)

FREQUENTA(<u>Lettore, Libreria</u>)

È_INTERESSATO_A(<u>Lettore, ISBN</u>).

Formulare opportune interrogazioni in SQL che permettano di determinare (senza usare l'operatore CONTAINS e usando solo se necessario le funzioni aggregate):

(a) i lettori interessati a tutti i libri di Hrabal;

(b) le coppie di lettori tali che esista almeno un libro che interessi ad entrambi;

(c) le coppie di lettori che frequentano (esattamente) le stesse librerie;

(d) i lettori che frequentano almeno una libreria in cui non è in vendita alcun libro cui sono interessati;

(e) i lettori che frequentano solo librerie in cui è in vendita almeno un libro cui sono interessati.

(f) gli autori che hanno scritto il maggior numero di libri (si assuma che non vi siano autori omonimi);

(g) i libri in vendita in tutte le librerie;

(h) i titoli dei libri che interessano ad almeno un lettore, ma non a più di due;

(i) i lettori che frequentano solo librerie in cui non è in vendita alcun libro cui sono interessati;

(l) le librerie frequentate solo da lettori interessati ad almeno uno dei libri in vendita.

Soluzione.

(a)
```
SELECT A.Lettore
FROM   È_INTERESSATO_A AS A
WHERE  NOT EXISTS (SELECT *
                   FROM   LIBRO AS B
                   WHERE  B.Autore = 'Hrabal'
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   È_INTERESSATO_A
                                      WHERE  A.Lettore = Lettore
                                             AND
                                             B.ISBN = ISBN));
```

(b)
```
SELECT A.Lettore, B.Lettore
FROM   È_INTERESSATO_A AS A, È_INTERESSATO_A AS B
WHERE  A.Lettore < B.Lettore                              #ordine lessicografico
       AND
       A.ISBN = B.ISBN;
```

(c)

```
SELECT A.Lettore, B.Lettore
FROM   FREQUENTA AS A, FREQUENTA AS B
WHERE  A.Lettore < B.Lettore
       AND
       NOT EXISTS (SELECT *
                   FROM   FREQUENTA AS C
                   WHERE  (C.Lettore = A.Lettore
                           AND
                           C.Libreria NOT IN (SELECT Libreria
                                              FROM   FREQUENTA AS D
                                              WHERE  D.Lettore = B.Lettore))
                          OR
                          (C.Lettore = B.Lettore
                           AND
                           C.Libreria NOT IN (SELECT Libreria
                                              FROM   FREQUENTA AS E
                                              WHERE  E.Lettore = A.Lettore)));
```

(d)

```
SELECT DISTINCT A.Lettore
FROM   FREQUENTA AS A
WHERE  A.Libreria NOT IN (SELECT Libreria
                          FROM   IN_VENDITA NATURAL JOIN È_INTERESSATO_A
                          WHERE  Lettore = A.Lettore);
```

(e)

```
SELECT DISTINCT Lettore
FROM   FREQUENTA
  EXCEPT
SELECT DISTINCT A.Lettore
FROM   FREQUENTA AS A
WHERE  A.Libreria NOT IN (SELECT Libreria
                          FROM   IN_VENDITA NATURAL JOIN È_INTERESSATO_A
                          WHERE  Lettore = A.Lettore);
```

In realtà l'opzione DISTINCT non è necessaria perché in questo caso i duplicati vengono rimossi automaticamente. Una soluzione alternativa è la seguente:

```
SELECT Lettore
FROM   FREQUENTA
  EXCEPT
SELECT DISTINCT A.Lettore
FROM   FREQUENTA AS A
WHERE  NOT EXISTS (SELECT *
                   FROM   È_INTERESSATO_A NATURAL JOIN IN_VENDITA
                   WHERE  Lettore = A.Lettore
                          AND
                          Libreria = A.Libreria);
```

(f)

```
CREATE VIEW NUM_LIBRI (Autore, N)
  AS
  SELECT   Autore, COUNT(*)
  FROM     LIBRO
  GROUP BY Autore
```

```
SELECT Autore
FROM   NUM_LIBRI
WHERE  N >= ALL (SELECT DISTINCT N
                 FROM   NUM_LIBRI);
```

(g)

```
SELECT ISBN
FROM   LIBRO AS L
WHERE  NOT EXISTS (SELECT *
                   FROM   IN_VENDITA AS V
                   WHERE  NOT EXISTS (SELECT *
                                      FROM   IN_VENDITA
                                      WHERE  Libreria = V.Libreria
                                             AND
                                             ISBN = L.ISBN));
```

(h)

```
SELECT Titolo
FROM   LIBRO AS L
WHERE  EXISTS (SELECT *
               FROM   È_INTERESSATO_A
               WHERE  ISBN = L.ISBN)
       AND
       NOT EXISTS (SELECT *
                   FROM   È_INTERESSATO_A AS A, È_INTERESSATO_A AS B,
                          È_INTERESSATO_A AS C
                   WHERE  A.ISBN = L.ISBN
                          AND
                          B.ISBN = L.ISBN
                          AND
                          C.ISBN = L.ISBN
                          AND
                          A.Lettore <> B.Lettore
                          AND
                          B.Lettore <> C.Lettore
                          AND
                          A.Lettore <> C.Lettore);
```

(i)

```
SELECT DISTINCT Lettore
FROM   È_INTERESSATO_A
  EXCEPT
SELECT DISTINCT Lettore
FROM   IN_VENDITA AS V, FREQUENTA AS F, È_INTERESSATO_A AS I
WHERE  V.Libreria = F.Libreria
       AND
       F.Lettore = I.Lettore
       AND
       I.ISBN = V.ISBN;
```

(l)

```
SELECT DISTINCT Libreria
FROM   IN_VENDITA AS F1
WHERE  NOT EXISTS (SELECT *
                   FROM   FREQUENTA AS F2
                   WHERE  F1.Libreria = F2.Libreria
                          AND
                          NOT EXISTS (SELECT *
```

```
FROM   IN_VENDITA
WHERE  Libreria = F1.Libreria
       AND
       ISBN IN (SELECT DISTINCT ISBN
                FROM   È_INTERESSATO_A
                WHERE  Lettore = F2.Lettore)));
```

□

Esercizio 1.5 Sia dato il seguente schema relazionale:

COMPONENTI(Parte#, Componente#)

PRODUTTORI(Parte#, Produttore#).

La relazione COMPONENTI descrive le *Componenti* di ogni *Parte*. Ogni *Componente* di una data *Parte* può essere, a sua volta, scomposta in più *Componenti*. Assumiamo che la relazione COMPONENTI non sia transitiva, ossia che da "y parte di x" e "z parte di y" non segua "z parte di x". La relazione PRODUTTORI indica i *Produttori* delle varie *Parti*.

Definire preliminarmente le chiavi delle relazioni date. Successivamente, formulare opportune interrogazioni in SQL che permettano di determinare (senza usare l'operatore CONTAINS e usando solo se necessario le funzioni aggregate):

(a) tutte le parti prodotte dal produttore Liberi, le quali contengano come parte componente la parte $X20$;

(b) tutte le parti componenti delle parti prodotte dal produttore Liberi o dal produttore Allineati;

(c) i produttori che producono tutte le parti componenti della parte $X20$, ma non producono alcuna parte componente della parte $X10$;

(d) i produttori che non producono alcun componente della parte $X20$, ma che producono almeno una componente di almeno una delle componenti della parte $X20$;

(e) le coppie (x, y) di produttori tali che esiste almeno una parte prodotta da x, ma non da y, e viceversa.

Soluzione. Le chiavi delle relazioni sono costituite dalla coppia di attributi delle relazioni stesse.

(a)

```
SELECT C.Parte#
FROM   COMPONENTI AS C, PRODUTTORI AS P   #Si poteva usare un natural join.
WHERE  Produttore# = 'Liberi'
       AND
       Componente# = 'X20'
       AND
       C.Parte# = P.Parte#;               #Condizione di join.
```

(b)

```
SELECT Componente#
FROM   COMPONENTI AS C, PRODUTTORI AS P
WHERE  Produttore# = 'Liberi'
       AND
       C.Parte# = P.Parte#
  UNION
SELECT Componente#
FROM   COMPONENTI AS C, PRODUTTORI AS P
WHERE  Produttore# = 'Allineati'
       AND
       C.Parte# = P.Parte#;
```

Soluzione alternativa:

```
SELECT Componente#
FROM   COMPONENTI AS C, PRODUTTORI AS P
WHERE  (Produttore# = 'Liberi'
        OR
        Produttore# = 'Allineati')
       AND
       C.Parte# = P.Parte#;
```

(c)

```
SELECT Produttore#
FROM   PRODUTTORI AS P1
WHERE  NOT EXISTS (SELECT *
                   FROM   COMPONENTI AS C1
                   WHERE  C1.Parte# = 'X20'
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   PRODUTTORI AS P2
                                      WHERE  P1.Produttore# = P2.Produttore#
                                             AND
                                             C1.Componente# = P2.Parte#))
  EXCEPT
SELECT Produttore#
FROM   PRODUTTORI AS P, COMPONENTI AS C
WHERE  P.Parte# = C.Componente#
       AND
       C.Parte# = 'X10';
```

In questo caso non occorre specificare l'opzione `DISTINCT` nella prima clausola `SELECT` perché i doppioni vengono eliminati dalla differenza insiemistica.

(d)

```
SELECT Produttore#
FROM   PRODUTTORI AS P, COMPONENTI AS C1, COMPONENTI AS C2
WHERE  C1.Componente# = C2.Parte#                       #Almeno una componente
       AND                                              #di una componente
       C1.Parte# = 'X20'                                #di X20
       AND
       P.Parte# = C2.Componente#
  EXCEPT                                                #ma non
SELECT Produttore#
FROM   PRODUTTORI AS P, COMPONENTI AS C
WHERE  C.Parte# = 'X20'                                 #almeno una componente
       AND                                              #di X20.
       P.Parte# = C.Componente#;
```

(e)

```
SELECT P1.Produttore#, P2.Produttore#
FROM   PRODUTTORI AS P1, PRODUTTORI AS P2
WHERE  EXISTS (SELECT *
               FROM   PRODUTTORI AS P3
               WHERE  P3.Produttore# = P1.Produttore#
                      AND
                      NOT EXISTS (SELECT *
                                  FROM   PRODUTTORI AS P4
                                  WHERE  P4.Produttore# = P2.Produttore#
                                         AND
                                         P4.Parte# = P3.Parte#))
       AND
```

```
        EXISTS (SELECT *
                FROM   PRODUTTORI AS P3
                WHERE  P3.Produttore# = P2.Produttore#
                       AND
                       NOT EXISTS (SELECT *
                                   FROM   PRODUTTORI AS P4
                                   WHERE  P4.Produttore# = P1.Produttore#
                                          AND
                                          P4.Parte# = P3.Parte#));
```

La soluzione seguente invece NON va bene poiché potrebbe essere vera solo una delle due condizioni.

```
SELECT A.Produttore#, B.Produttore#
FROM   PRODUTTORI AS A, PRODUTTORI AS B
WHERE  A.Parte# NOT IN (SELECT Parte#
                        FROM   PRODUTTORI
                        WHERE  Produttore# = B.Produttore#)
  INTERSECT
SELECT A.Produttore#, B.Produttore#
FROM   PRODUTTORI AS A, PRODUTTORI AS B
WHERE  B.Parte# NOT IN (SELECT Parte#
                        FROM   PRODUTTORI
                        WHERE  Produttore# = A.Produttore#);
```

□

Esercizio 1.6 Sia dato il seguente schema relazionale:

ABITA(Pnome, Città, Via)

LAVORA(Pnome, Anome, Stipendio)

HA_SEDE_IN(Anome, Città)

DIPENDE_DA(Pnome, Snome),

dove P sta per persona, A per azienda e S per supervisore. Si assuma che un'azienda possa avere più sedi. Definire preliminarmente le chiavi delle relazioni date. Successivamente, formulare opportune interrogazioni in SQL che permettano di determinare (senza usare l'operatore CONTAINS e usando solo se necessario le funzioni aggregate):

(a) il nome di tutte le persone che non abitano nella città in cui ha una sede l'azienda per cui lavorano;

(b) il nome di tutte le persone che non lavorano per un'azienda che ha una sede a Udine o a Venezia;

(c) per ogni azienda che non ha una sede a Venezia o a Udine, il nome di tutti i supervisori;

(d) il nome di tutte le aziende che hanno una sede in tutte le città in cui ha una sede l'azienda TFR;

(e) il nome di tutti i supervisori che ricevono uno stipendio superiore a quello di tutte le persone che dipendono da loro.

Soluzione. Le chiavi delle relazioni sono le seguenti:

- **Pnome** per ABITA, escludendo casi di omonimia;
- **Pnome** per LAVORA, poiché ogni persona può lavorare solo per un'azienda e non ci sono omonimie;
- **Anome** e **Città** per HA_SEDE_IN, perché un'azienda può avere più sedi;
- **Pnome** per DIPENDE_DA, perché ogni lavoratore può avere solo un supervisore.

(a)

```
SELECT Pnome
FROM   ABITA AS A, LAVORA AS L
WHERE  A.Pnome = L.Pnome
       AND
       A.Città NOT IN (SELECT Città
                       FROM   HA_SEDE_IN
                       WHERE  Anome = L.Anome);
```

(b)

```
SELECT Pnome
FROM   LAVORA
WHERE  Anome NOT IN (SELECT Anome
                     FROM   HA_SEDE_IN
                     WHERE  Città = 'Udine'
                            OR
                            Città = 'Venezia');
```

(c)

```
SELECT Snome
FROM   DIPENDE_DA AS D, LAVORA AS L
WHERE  Snome = L.Pnome
       AND
       NOT EXISTS (SELECT *
                   FROM   HA_SEDE_IN
                   WHERE  L.Anome = Anome
                          AND
                          (Città = 'Udine'
                           OR
                           Città = 'Venezia'));
```

(d)

```
SELECT Anome
FROM   HA_SEDE_IN AS H1
WHERE  NOT EXISTS (SELECT *
                   FROM   HA_SEDE_IN AS H2
                   WHERE  H2.Anome = 'TFR'
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   HA_SEDE_IN AS H3
                                      WHERE  H3.Anome = H1.Anome
                                             AND
                                             H3.Città = H2.Città));
```

(e)

```
SELECT D.Snome
FROM   DIPENDE_DA AS D, LAVORA AS L
WHERE  D.Snome = L.Pnome
       AND
       Stipendio > ALL (SELECT Stipendio
                        FROM   DIPENDE_DA NATURAL JOIN LAVORA
                        WHERE  D.Snome = Snome);
```

□

Esercizio 1.7 Sia dato il seguente schema relazionale, relativo al calendario ciclistico di un determinato anno:

CICLISTA(Nome, Nazione, Età)

GAREGGIA(NomeCiclista, NomeGara, Piazzamento)

GARA(Nome, Nazione, Lunghezza).

Il dominio di *Piazzamento* è l'insieme dei numeri interi positivi. Definire preliminarmente le chiavi delle relazioni date. Successivamente, formulare opportune interrogazioni in SQL che permettano di determinare (senza usare l'operatore *CONTAINS* e usando solo se necessario le funzioni aggregate):

(a) il nome dei ciclisti che hanno gareggiato solo nella propria nazione;

(b) il nome dei ciclisti che provengono da una nazione in cui non si svolge alcuna gara;

(c) il nome dei ciclisti che hanno preso parte a tutte le gare svoltesi in Francia e che non hanno partecipato ad alcuna gara svoltasi in Italia;

(d) le coppie (x, y) di ciclisti tali che esista almeno una gara cui entrambi hanno partecipato e che, in tutte le gare cui entrambi hanno preso parte, x si sia piazzato meglio di y;

(e) le coppie (x, y) di ciclisti tali che esista almeno una gara cui x ha partecipato e y non ha partecipato.

Soluzione. Le chiavi delle relazioni sono le seguenti:

- **Nome** per CICLISTA;
- **NomeCiclista** e **NomeGara** per GAREGGIA;
- **Nome** per GARA.

Si noti che la scelta di **Nome** quale attributo chiave di CICLISTA (rispettivamente GARA) è una scelta obbligata, qualora si assuma (come implicitamente fatto nelle interrogazioni) che **NomeCiclista** (risp. **NomeGara**) sia chiave esterna di GAREGGIA rispetto a CICLISTA (risp. GARA).

(a)
```
SELECT Nome
FROM   CICLISTA AS C
WHERE  NOT EXISTS (SELECT *
                   FROM   GAREGGIA AS GG, GARA AS GA
                   WHERE  [GG.]NomeGara = [GA.]Nome
                          AND
                          [GG.]NomeCiclista = C.Nome
                          AND
                          C.Nazione <> [GA.]Nazione);
```

I riferimenti racchiusi tra parentesi quadre (ad esempio, [GG.]) sono stati aggiunti solo per aumentare la leggibilità dell'interrogazione e possono essere omessi.

(b)
```
SELECT Nome
FROM   CICLISTA
WHERE Nazione NOT IN (SELECT Nazione
                      FROM   GARA);
```

Oppure:

```
SELECT Nome
FROM   CICLISTA AS C
WHERE  NOT EXISTS (SELECT *
                   FROM   GARA
                   WHERE  C.Nazione = Nazione);
```

La prima soluzione è migliore della seconda in quanto l'interrogazione innestata è non correlata e, di conseguenza, può essere eseguita un'unica volta. Nella seconda soluzione, invece, l'interrogazione interna è correlata e va quindi eseguita per ogni ciclista.

(c)

```
SELECT Nome
FROM   CICLISTA AS C
WHERE  NOT EXISTS (SELECT *
                   FROM   GARA AS GA
                   WHERE  [GA.]Nazione = 'Francia'
                          AND
                          NOT EXISTS (SELECT *
                                      FROM   GAREGGIA AS GG
                                      WHERE  [GG.]NomeCiclista = C.Nome
                                             AND
                                             [GG.]NomeGara = [GA.]Nome))
       AND
       NOT EXISTS (SELECT *
                   FROM   GARA AS GA
                   WHERE  [GA.]Nazione = 'Italia'
                          AND
                          EXISTS (SELECT *
                                  FROM   GAREGGIA AS GG
                                  WHERE  [GG.]NomeCiclista = C.Nome
                                         AND
                                         [GG.]NomeGara = [GA.]Nome));
```

(d)

```
SELECT DISTINCT (GG1.NomeCiclista, GG2,NomeCiclista)
FROM   GAREGGIA AS GG1, GAREGGIA AS GG2
WHERE  GG1.NomeCiclista <> GG2.NomeCiclista
       AND
       GG1.NomeGara = GG2.NomeGara
       AND
       NOT EXISTS (SELECT *
                   FROM   GAREGGIA AS GG3, GAREGGIA AS GG4
                   WHERE  GG3.NomeGara = GG4.NomeGara
                          AND
                          GG1.NomeCiclista = GG3.NomeCiclista
                          AND
                          GG2.NomeCiclista = GG4.NomeCiclista
                          AND
                          GG3.Piazzamento > GG4.Piazzamento);
```

(e)

```
SELECT DISTINCT (GG1.NomeCiclista, GG2,NomeCiclista)
FROM GAREGGIA AS GG1, GAREGGIA AS GG2
WHERE  GG1.NomeCiclista <> GG2.NomeCiclista      #Tale condizione può essere omessa
       AND
       NOT EXISTS (SELECT *
                   FROM   GAREGGIA AS GG3
                   WHERE  GG1.NomeGara = [GG3.]NomeGara
                          AND
                          GG2.NomeCiclista = [GG3.]NomeCiclista)
```

L'interrogazione innestata è chiaramente correlata. Inoltre, si noti che `GG2` può essere sostituita con `CICLISTA`.

Una soluzione alternativa è la seguente:

```
SELECT C1.Nome, C2.Nome
FROM   CICLISTA AS C1, CICLISTA AS C2
WHERE  EXISTS (SELECT *
               FROM   GAREGGIA
               WHERE  C1.Nome = NomeCiclista
                      AND
                      NomeGara NOT IN (SELECT NomeGara
                                       FROM   GAREGGIA
                                       WHERE  C2.Nome = NomeCiclista));
```

□

2 Gestione della sicurezza

Esercizio 2.1 I privilegi che si possono esercitare su una vista sono gli stessi che si possono esercitare su una relazione, ad eccezione dei privilegi che non si applicano alle viste (alter e index). Non possono, comunque, essere concessi quei privilegi il cui esercizio comporta l'esecuzione di operazioni sulla vista che non è possibile tradurre in termini di operazioni sulla/e relazione/i componenti. Se una vista è definita su una singola relazione (vista), i priviligi che l'utente che crea la vista ha su di essa sono gli stessi che ha sulla relazione (vista), ad eccezione dei privilegi alter e index. Si consideri la relazione

AcquistoProdotti(<u>NumProdotto, Data</u>, PrezzoUnitario, Quantità, Nome)

creata da Rossi. Si supponga che:

- al tempo 10, Rossi crei una vista che, per ogni prodotto, restituisce la media del prezzo unitario;
- al tempo 16, Rossi conceda tutti i privilegi su tale vista a Neri con la grant option;
- al tempo 20, Neri conceda tutti i privilegi sulla vista a Gialli con la grant option;
- al tempo 28, Rossi conceda tutti i privilegi sulla vista a Gialli;
- al tempo 30, Rossi revochi a Neri i privilegi che gli aveva precedentemente concesso.

(a) Scrivere il comando SQL per la definizione della vista ed indicare le operazioni che Rossi può eseguire su di essa.

(b) Scrivere i comandi SQL per garantire e revocare i privilegi garantiti sulla vista.

(c) Stabilire se, dopo l'esecuzione di revoca eseguita da Rossi, l'utente Gialli può accedere alla vista e garantire ad altri utenti diritti su tale vista.

Esercizio 2.2 Si descriva l'effetto delle seguenti istruzioni; in particolare, si mettano in evidenza le autorizzazioni presenti dopo l'esecuzione di ciascuna istruzione (ogni istruzione è preceduta dall'indicazione dell'utente che la esegue):

A1: `GRANT SELECT ON R TO A2, A3 WITH GRANT OPTION`

A2: `GRANT SELECT ON R TO A4`

A3: `GRANT SELECT ON R TO A4 WITH GRANT OPTION`

A1: `REVOKE SELECT ON R FROM A2`

A4: `GRANT SELECT ON R TO A2`

A1: `REVOKE SELECT ON R FROM A3`