Sistemi Informativi T
31 gennaio 2020
Risoluzione

Tempo a disposizione: 2:30 ore

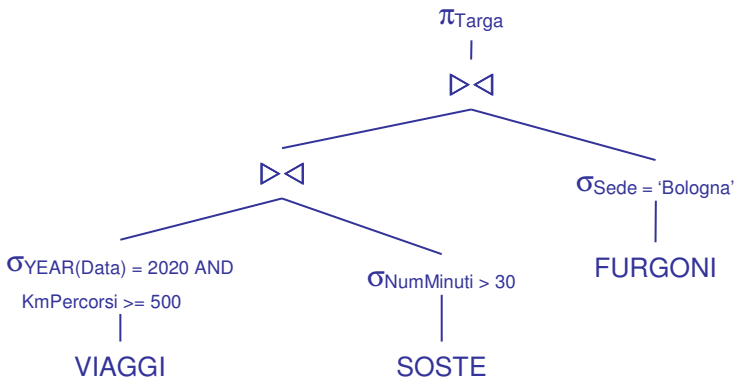
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

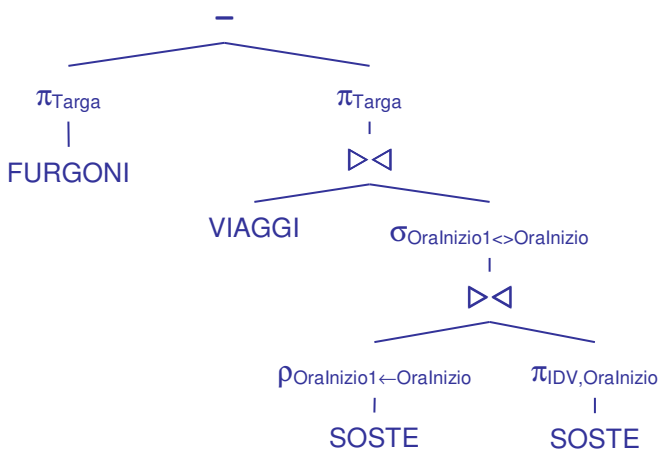
```
FURGONI(Targa,Sede);
VIAGGI(IDV,Targa,Data,OraPartenza,OraArrivo,KmPercorsi),
      Targa REFERENCES FURGONI,
      UNIQUE(Targa,Data,OraPartenza);
SOSTE(IDV,OraInizio,NumMinuti),
      IDV REFERENCES VIAGGI;

-- OraPartenza, OraArrivo e OraInizio sono di tipo TIME (es. '12:32').
-- KmPercorsi e NumMinuti sono di tipo INT.
-- L'ora di inizio di una sosta è sempre compresa tra l'ora di
-- partenza e l'ora di arrivo.
-- Per semplicità si suppone che ogni viaggio inizi e termini in uno
-- stesso giorno.
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Le targhe dei furgoni della sede di Bologna che nel 2020 hanno fatto un viaggio di almeno 500 km facendo almeno una sosta di più di 30 minuti



1.2) [2 p.] Le targhe dei furgoni che non hanno mai fatto 2 o più soste in uno stesso viaggio



Sistemi Informativi T
31 gennaio 2020
Risoluzione

SQL (5 punti totali)

Con riferimento al DB dell'esercizio 1, si scrivano in SQL le seguenti interrogazioni:

2.1) **[2 p.]** Per ogni sede, il tempo medio delle soste dei propri furgoni, considerando solo i viaggi in cui c'è stata almeno una sosta di almeno 15 minuti

```
SELECT   F.Sede, DEC(AVG(S.NumMinuti*1.0),6,2) AS DURATA_MEDIA_SOSTA
FROM     FURGONI F, VIAGGI V, SOSTE S
WHERE    F.Targa = V.Targa
AND      V.IDV = S.IDV
AND      EXISTS (    SELECT *
                     FROM   SOSTE S1
                     WHERE  S1.IDV = V.IDV
                     AND    S1.NumMinuti >= 15 )
GROUP BY F.Sede;
```

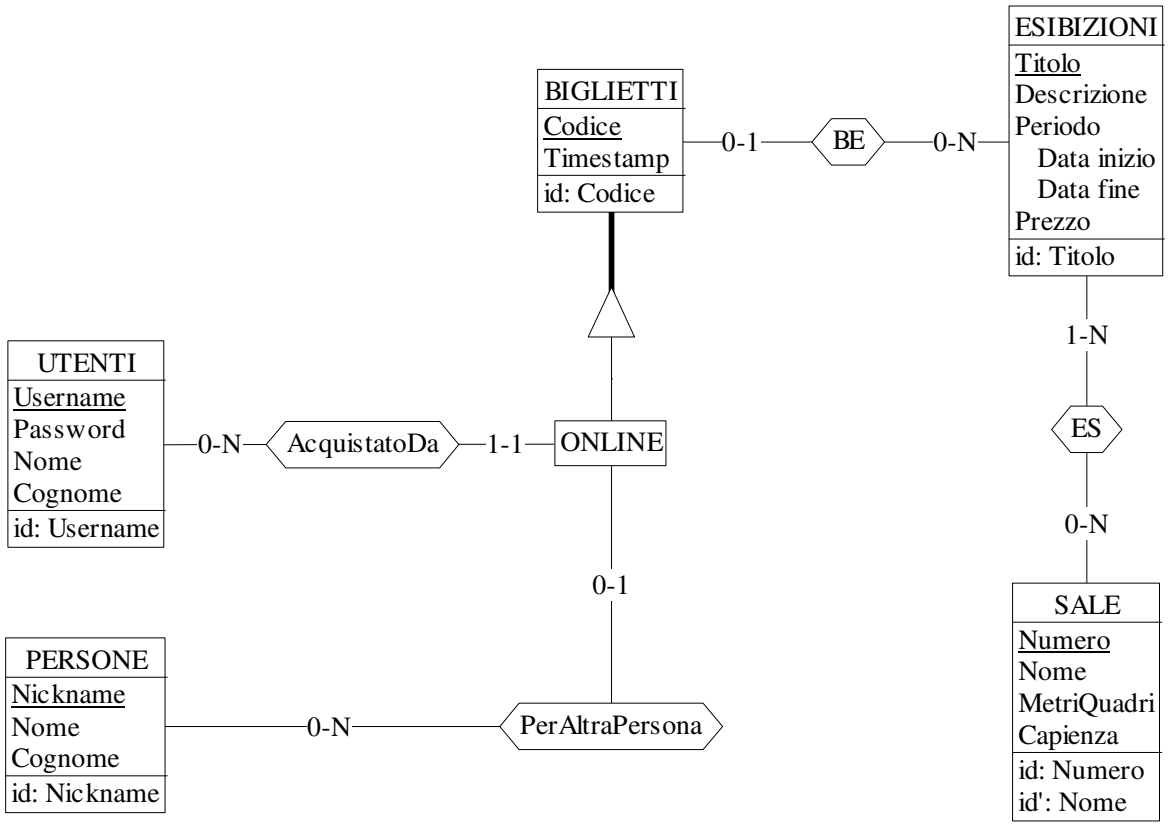
2.2) **[3 p.]** Per ogni anno, si determini la targa del furgone e l'identificativo del relativo viaggio in cui è stata massima la durata totale delle soste rispetto alla durata del viaggio

```
WITH
TOT_DURATA_SOSTE (Anno,Targa,IDV,DurataViaggio,TotMinuti) AS (
   SELECT YEAR(V.Data), V.Targa, S.IDV,
          HOUR(V.OraArrivo-V.OraPartenza)*60+
                                MINUTE(V.OraArrivo-V.OraPartenza),
          SUM(S.NumMinuti)
   FROM   VIAGGI V, SOSTE S
   WHERE  V.IDV = S.IDV
   GROUP BY YEAR(V.Data), V.Targa, S.IDV, V.OraArrivo-V.OraPartenza
   )
SELECT   T.Anno, T.Targa, T.IDV
FROM     TOT_DURATA_SOSTE T
WHERE    T.TotMinuti*1.0/T.DurataViaggio =
   (     SELECT      MAX(T1.TotMinuti*1.0/T1.DurataViaggio)
         FROM  TOT_DURATA_SOSTE T1
         WHERE T1.Anno = T.Anno        );

-- La c.t.e. calcola per ogni viaggio la durata del viaggio in minuti
-- e la durata complessiva delle relative soste
```

Sistemi Informativi T
31 gennaio 2020
Risoluzione

3) Progettazione concettuale (6 punti)

Il Museo di Arte Moderna e Ipermoderna (MAMI) ha istituito la vendita di biglietti online per la visita dei propri spazi e delle esposizioni che si tengono in questi. Ogni utente, registrato con username, password, nome e cognome, può comprare biglietti per sé o anche per altre persone, in quest'ultimo caso fornendo un nominativo (nome e cognome) ed un nickname univoco. I biglietti si possono anche acquistare all'ingresso del museo, ma in questo caso non hanno un nominativo. Tutti i biglietti hanno un codice numerico, che viene generato automaticamente all'atto della vendita, e un timestamp di emissione (quando il biglietto è stato generato e venduto).
Assieme a un biglietto ordinario si può acquistare un biglietto aggiuntivo per un'esposizione. Ogni esposizione ha un titolo, una descrizione, un prezzo, un periodo di svolgimento (data inizio e data fine) e l'elenco delle sale del MAMI presso cui si svolge. Di ogni sala, identificata da un numero o, alternativamente, da un nome (es.: la sala 15 si chiama "Duchamp") il sistema del MAMI mantiene informazioni sui metri quadri e sulla capienza (numero massimo di persone).



Commenti:

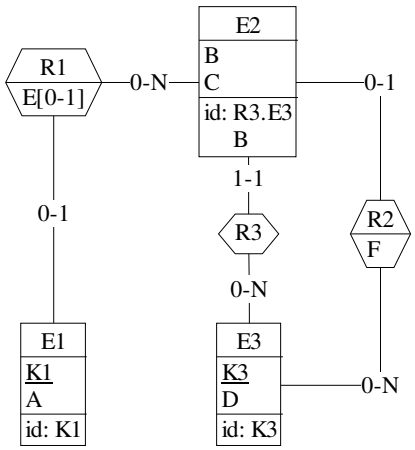
- L'associazione BE è sufficiente per modellare la presenza di biglietti aggiuntivi, data l'assenza di specifiche più precise. In particolare, in questa soluzione un biglietto aggiuntivo non ha un identificativo proprio (in quanto non è rappresentato mediante un'entità), cosa non richiesta dalle specifiche.
- La definizione del subset ONLINE pemette di modellare agevolmente le specifiche riguardanti l'acquisto di biglietti online. Senza tale entità per un biglietto si avrebbero due proprietà opzionali: essere stato acquistato (online) da un utente registrato ed essere intestato a un'altra persona, diversa dall'utente stesso. Tuttavia non si esprimerebbe il vincolo che questa seconda proprietà si può avere solo se si ha anche la prima, cosa che risulta invece evidente nella soluzione proposta.

Sistemi Informativi T
31 gennaio 2020
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) tutti gli attributi sono di tipo INT;
b) nessuna associazione viene tradotta separatamente;
c) un'istanza di E1 può essere associata, tramite R1 e R2, solo a un'istanza di E3 con D>A;
d) la somma dei valori di B delle istanze di E2 identificate da una stessa istanza di E3 è minore di 50;



4.1) [3 p.] Si progettino gli opportuni schemi relazionali e si definiscano tali schemi in DB2 (sul database SIT_STUD) mediante un file di script denominato SCHEMI.txt (o SCHEMI.sql)

```
CREATE TABLE E3 (
K3      INT NOT NULL PRIMARY KEY,
D       INT NOT NULL                        );

CREATE TABLE E2 (
K3R3    INT NOT NULL REFERENCES E3,
B       INT NOT NULL,
C       INT NOT NULL,
K3R2    INT REFERENCES E3,
F       INT,
CONSTRAINT R2 CHECK ((K3R2 IS NULL AND F IS NULL) OR
                     (K3R2 IS NOT NULL AND F IS NOT NULL) ),
PRIMARY KEY (K3R3,B)            );

CREATE TABLE E1 (
K1      INT NOT NULL PRIMARY KEY,
A       INT NOT NULL,
K3R3    INT,
B       INT,
E       INT,
CONSTRAINT R1 CHECK ((K3R3 IS NULL AND B IS NULL AND E IS NULL) OR
                     (K3R3 IS NOT NULL AND B IS NOT NULL) ),
CONSTRAINT FKE2 FOREIGN KEY (K3R3,B) REFERENCES E2              );
```

4.2) [3 p.] Per i vincoli non esprimibili a livello di schema si predispongano opportuni **trigger che evitino inserimenti di singole tuple non corrette**, definiti in un file TRIGGER.txt (o TRIGGER.sql) e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( N.A >= (    SELECT  E3.D
                   FROM    E2, E3
                   WHERE  (N.K3R3,N.B) = (E2.K3R3,E2.B)
                   AND     E2.K3R2 = E3.K3    ) )
SIGNAL SQLSTATE '70001' ('L''istanza di E1 è associata a un''istanza di E3 con D<=A!');

CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( 50 <= N.B + ( SELECT  SUM(E2.B)
                     FROM    E2
                     WHERE  E2.K3R3 = N.K3R3 ) )
SIGNAL SQLSTATE '70002' ('Il valore di B è troppo alto!');
```