Sistemi Informativi T
17 febbraio 2016
Risoluzione

Tempo a disposizione: 2:30 ore

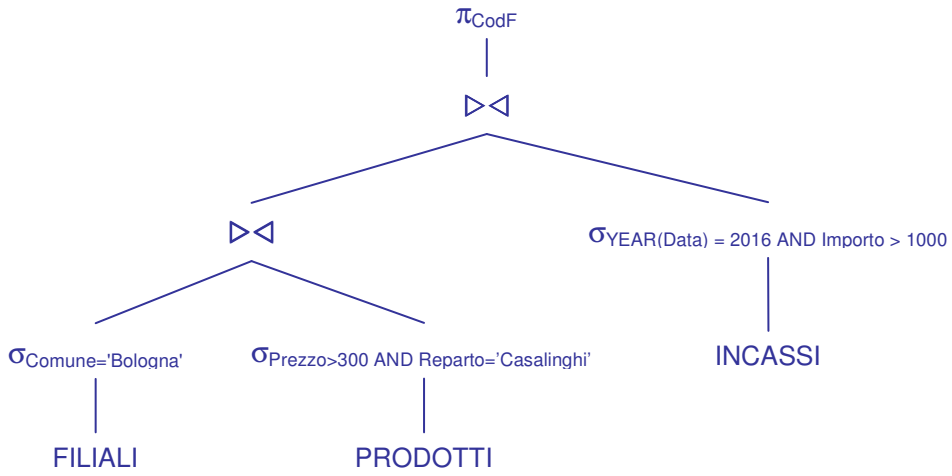
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

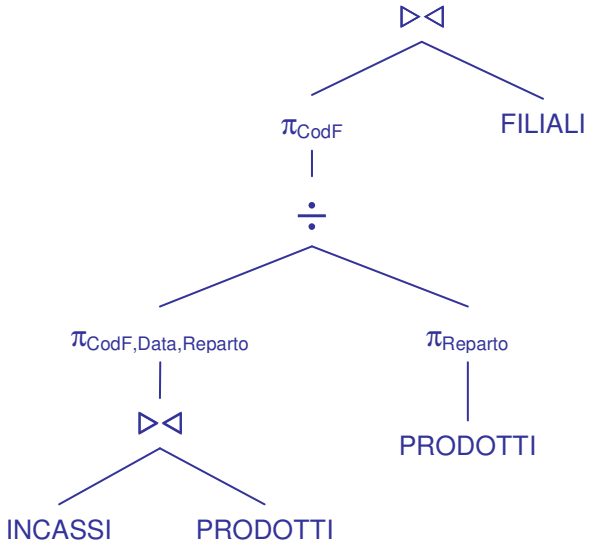
```
FILIALI(CodF,Indirizzo,Comune);
PRODOTTI(CodP,Reparto,Prezzo);
INCASSI(CodF,CodP,Data,Importo),
      CodF REFERENCES FILIALI, CodP REFERENCES PRODOTTI;
-- Importo e Prezzo sono in formato DEC(6,2)
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I codici delle filiali di Bologna che hanno venduto un prodotto del reparto Casalinghi con prezzo maggiore di 300 € e incassando per tale prodotto più di 1000 € in un giorno del 2016



1.2) **[2 p.]** I dati completi delle filiali che, in uno stesso giorno, hanno venduto almeno un prodotto di ogni reparto



La condizione "in uno stesso giorno" è automaticamente verificata dalla divisione, che restituisce le coppie (CodF,Data) associate a tutti i reparti

Sistemi Informativi T
17 febbraio 2016
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] Per ogni giorno, la media degli incassi considerando solo i prodotti venduti, in quel giorno, in più di una filiale

```
SELECT   I.Data, CAST(AVG(I.Importo) AS DEC(6,2)) AS MediaIncassi
FROM     INCASSI I
WHERE    EXISTS (    SELECT *
                     FROM  INCASSI I1
                     WHERE I1.Data = I.Data
                     AND   I1.CodP = I.CodP
                     AND   I1.CodF <> I.CodF )
GROUP BY I.Data

-- La subquery verifica che, in una stessa data, più di una filiale
-- abbia venduto un dato prodotto
```

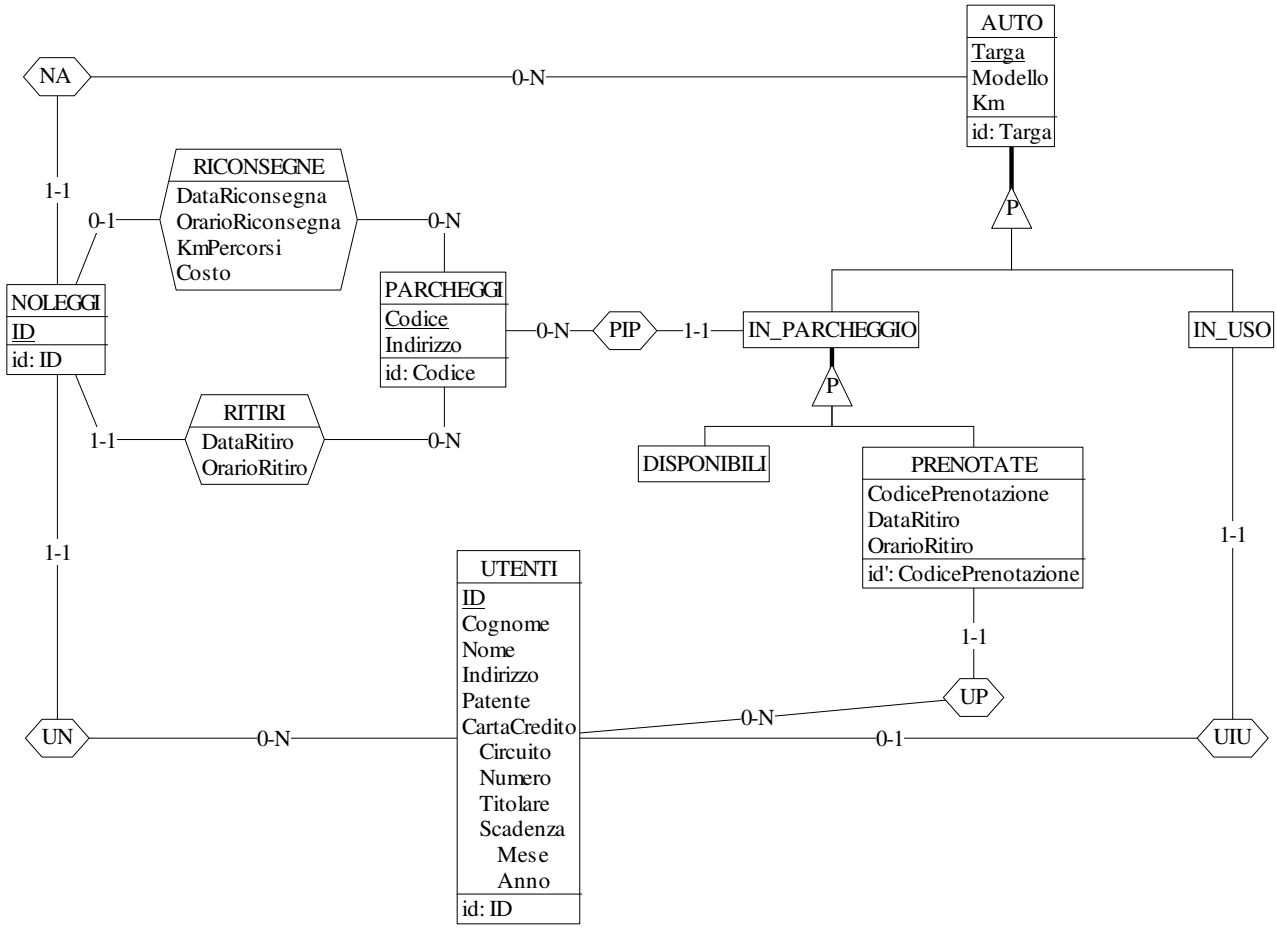
2.2) [3 p.] Per ogni reparto, le filiali che hanno complessivamente incassato più della media degli incassi complessivi, per lo stesso reparto, delle altre

```
WITH INCASSI_REP(Reparto,CodF,TotImporto) AS (
   SELECT P.Reparto, I.CodF, SUM(I.Importo)
   FROM   PRODOTTI P, INCASSI I
   WHERE  P.CodP = I.CodP
   GROUP BY  I.CodF, P.Reparto )
SELECT I.*
FROM   INCASSI_REP I
WHERE  I.TotImporto > ( SELECT   AVG(I1.TotImporto)
                        FROM     INCASSI_REP I1
                        WHERE    I1.Reparto = I.Reparto
                        AND      I1.CodF <> I.CodF )

-- La C.T.E. calcola, per ogni filiale e ogni reparto, l'importo totale
-- delle relative vendite.
-- Un'intepretazione alternativa della query aggiungerebbe al risultato
-- anche i reparti per cui ci sono stati incassi in una sola filiale.
-- In questo caso si può ad esempio completare la clausola WHERE con:

OR     NOT EXISTS (  SELECT *
                     FROM   INCASSI_REP I1
                     WHERE  I1.Reparto = I.Reparto
                     AND    I1.CodF <> I.CodF )
```

Sistemi Informativi T
17 febbraio 2016
Risoluzione

3) Progettazione concettuale (6 punti)

PickACar (PAC) è un servizio che permette di noleggiare auto, prelevandole da un parcheggio e rilasciandole presso lo stesso o un altro parcheggio PAC. Per la prenotazione occorre preventivamente registrarsi, fornendo i dati anagrafici (nome, cognome, indirizzo), un numero di patente e una carta di credito (per quest'ultima servono: circuito, numero, titolare e scadenza). La ricerca di un'auto viene effettuata specificando il parcheggio da cui si vuole ritirare l'auto e la data e orario previsto di ritiro. Per evitare spiacevoli inconvenienti, PAC permette di prenotare solo le auto disponibili, ovvero né in uso né già prenotate.
Effettuata la scelta, all'utente viene fornito un codice di prenotazione e il numero di targa dell'auto. Alla riconsegna del veicolo viene calcolato l'effettivo costo (che tiene conto della data e orario di riconsegna e dei chilometri percorsi) e quindi addebitato sulla carta di credito.
Si progetti il database per la PAC, tenendo conto che non è richiesto mantenere informazioni sulle prenotazioni non più attive (ovvero che han già dato luogo a un noleggio oppure che sono state disdette).
.

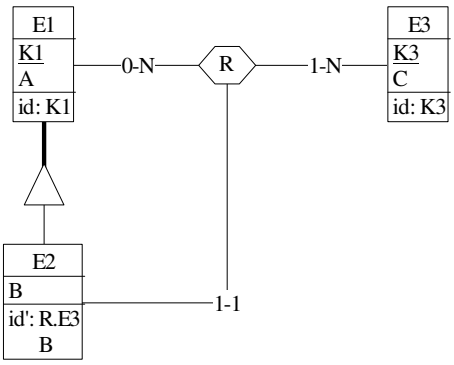


Commenti:

- La distinzione tra i diversi stati in cui può trovarsi un'auto e l'introduzione dell'entità IN_PARCHEGGIO permettono di rappresentare in maniera chiara le specifiche che riguardano lo stato attuale del sistema.
- La parte storica riguarda, per esclusione, i soli NOLEGGI, ed è supportata dalle associazioni RITIRI, RICONSEGNE, UN e NA. Quest'ultima, proprio perché relativa a tutti i noleggi (sia presenti che passati), associa NOLEGGI con AUTO. In alternativa si poteva distinguere tra noleggi passati (conclusi) e presenti (in corso), e associare solo i presenti a IN_USO.
- All'atto del ritiro di un'auto prenotata viene creata un'istanza di NOLEGGI e una di RITIRI. DataRitiro e OrarioRitiro vengono inseriti come attributi di RITIRI in quanto, poiché quando l'auto viene ritirata diventa IN_USO e rimossa da PRENOTATE, non così facendo i dati di inizio noleggio si perderebbero.
- Molti vincoli non sono esprimibili mediante il modello E/R, ad esempio quello per cui un'auto è IN_USO solo se l'utente relativo ha effettivamente in corso un noleggio per la stessa auto.

Sistemi Informativi T
17 febbraio 2016
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) le entità E1 ed E2 vengono tradotte assieme;
c) l'associazione R non viene tradotta separatamente;
d) un'istanza di E2 non referenzia mai, tramite R, se stessa;

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

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

CREATE TABLE E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
TIPO2 SMALLINT NOT NULL CHECK (TIPO2 IN (0,1)),          -- 1: istanza anche di E2
K3R INT REFERENCES E3,
B INT,
K1R INT REFERENCES E1,
CONSTRAINT E2 CHECK ((TIPO2 = 0 AND K3R IS NULL AND B IS NULL AND K1R IS NULL)
     OR (TIPO2 = 1 AND K3R IS NOT NULL AND B IS NOT NULL AND K1R IS NOT NULL)),
CONSTRAINT PUNTO_D CHECK (K1R <> K1)                                                    );
```

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 e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce l'unicità dei valori della coppia (K3R,B), chiave che ammette valori nulli

CREATE TRIGGER UNIQUE_K3R_B
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS ( SELECT * FROM E1
                    WHERE (N.K3R,N.B) = (E1.K3R,E1.B)  )  )
SIGNAL SQLSTATE '70001' ('La coppia (K3R,B) non ammette valori duplicati!');

-- A causa del vincolo di cardinalità minima, ogni inserimento in E3 deve avvenire contestualmente all'inserimento
-- di almeno due tuple in E1 (di cui almeno una deve essere anche istanza di E2), ovvero nella stessa transazione
```