BASI DI DATI - PROFF. S. CERI, M. MATERA

III RECUPERO - 10 SETTEMBRE 2004

I PROVA

Si consideri il seguente schema, relativo a un sistema di prenotazione e noleggio di biciclette, in cui le indicazioni orarie possono assumersi piene (dalle 8 alle 19):

CLIENTE (Num, Nome, TipoDoc, NumeroDoc)
PRENOTA (Num-Cli, TipoBici, Giorno, OraInizio, OraFine)
USA (Num-Cli, Num-Bici, Giorno, OraInizio, OraFine, Costo)
BICICLETTA (Num-Bici, TipoBici, CostoOra, CostoGiorno)

A. DDL (2 punti)

Scrivere i comandi SQL per creare le tabelle PRENOTA e BICICLETTA, effettuando opportune ragionevoli ipotesi su domini, vincoli e reazioni ai cambiamenti.

```
Create Table BICICLETTA (
    Num-Bici        integer primary key,
    TipoBici        varchar(50) not null,
    CostoOra        decimal(4,2),
    CostoGiorno     decimal(5,2)
)

Create table PRENOTA (
    Num-Cli         integer references CLIENTE(Num) on delete cascade on update cascade,
    TipoBici        varchar(50) default "City Bike" check TipoBici IN ( SELECT TipoBici
                                                                        FROM BICICLETTA )
    Giorno          date,
    OraInizio       integer between 8 and 18,
    OraFine         integer check ( OraFine between 9 and 19 and OraFine - OraInizio > 0) ,
    Primary key (Num-Cli, TipoBici, Giorno)
)
```

N.B. **NON È POSSIBILE** accendere un vincolo di integità referenziale tra il TipoBici della tabella PRENOTA e il TipoBici della tabela BICICLETTA, poiché in BICICLETTA TipoBici ***non è unique*** [si rammenta infatti che, affinché sia possibile definire una chiave esterna, i valori esterni devono (collettivamente) essere unique – *non chiave ma neanche meno di unique* – e che ciò è necessario, ad esempio, per poter applicare in modo non ambiguo le politiche di reazione a modifiche e cancellazioni nella tabella esterna (o "padre")].
Ciononostante, è del tutto sensato imporre il vincolo che non si possa prenotare una bicicletta di un tipo di cui il noleggiatore non possegga alcun esemplare (che sia o che non sia in prestito al momento della prenotazione) . Per esprimere il vincolo si può usare una **check** che ha lo stesso potere vincolante della chiave esterna ma non garantisce alcuna politica di reazione ai cambiamenti.

B. LINGUAGGI FORMALI (7 punti)

1. Esprimere in Algebra Relazionale ottimizzata, Calcolo Relazionale e Datalog la seguente interrogazione **(4.5 punti)**:
   Trovare il nome dei clienti che hanno prenotato bici di tipo "da corsa" senza mai usarle

Algebra relazionale (non ottimizzata):

A) Prima interpretazione: CLIENTI CHE HANNO FATTO QUALCHE PRENOTAZIONE MA MAI NESSUN USO
a1) Una prima idea per una soluzione: join tra chi non ha mai usato una bici da corsa e chi ne ha prenotata una:

$$\Pi_{Nome} \; (\; (CLI - (CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (USA \rhd\!\lhd (\sigma_{Tipo="Corsa"} BIC)))) \rhd\!\lhd_{Num=Num} (CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (\sigma_{Tipo="Corsa"} PRE)) \;)$$

a2) Si può anche pensare di sottrarre i clienti che hanno usato una bici da corsa dai quelli che ne hanno prenotata una:

$$\Pi_{Nome} (\; (CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (\sigma_{Tipo='Corsa'} PRE)) - (CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (USA \rhd\!\lhd (\sigma_{Tipo='Corsa'} BIC))) \;)$$

o, forse ancora più chiaro…:

$$\Pi_{Nome} (CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (\Pi_{Num\text{-}Cli} (\sigma_{Tipo='Corsa'} PRE) - \Pi_{Num\text{-}Cli} (USA \rhd\!\lhd (\sigma_{Tipo='Corsa'} BIC)) \;)$$

B) Seconda interpretazione: CLIENTI PER CUI A NESSUNA DELLE LORO PRENOTAZIONI DI BICI DA CORSA (e una deve esistere di sicuro) CORRISPONDA UN USO DI UNA BICI DA CORSA NELLO STESSO GIORNO (quindi gli eventuali usi di bici da corsa di tali clienti, che pure possono esserci, non devono però avere una corrispondente prenotazione). Diventa: tutti i clienti con almeno una prenotazione di bici da corsa meno tutti i clienti con una prenotazione di bici da corsa a cui corrisponde un uso per lo stesso giorno e ora:

$$\Pi_{Nome}(CLI \rhd\!\lhd_{Num=Num\text{-}Cli} (\Pi_{Num\text{-}Cli}(\sigma_{Tipo='Corsa'} PRE) - \Pi_{Num\text{-}Cli}(\sigma_{Tipo='Corsa'} PRE \rhd\!\lhd_{\substack{Giorno = Giorno \wedge \\ OraInizio = OraInizio \wedge \\ Num\text{-}Cli = Num\text{-}Cli}} USA \rhd\!\lhd_{Num\text{-}Bici = Num\text{-}Bici} \sigma_{Tipo='Corsa'} BIC)))$$

N.B. A) e B) sono in generale diversi, ma se la prenotazione è obbligatoria sono equivalenti. Si noti però che lo schema non permette in alcun modo di arguire se la prenotazione sia obbligatoria.

TRC (secondo l'interpretazione A):

$$\{ t \mid \exists t_P \in PRE, \exists t_C \in CLI \quad \textit{(Esiste una prenotazione)}$$
$$(t[Num, Nome] = t_C[Num, Nome] \wedge$$
$$t_P[Num\text{-}Cli] = t_C[Num] \wedge \quad \textit{("join" tra cliente e prenotazione)}$$
$$t_P[Tipo] = \text{"Corsa"} \wedge \quad \textit{(per una bici da corsa)}$$
$$\neg (\exists t_B \in BIC, \exists t_U \in USA \quad \textit{(però non esiste alcun uso)}$$
$$(t_U[Num\text{-}Cli] = t_P[Num\text{-}Cli] \wedge \quad \textit{(dello } \textbf{\textit{stesso}} \textit{ cliente)}$$
$$t_B[Num\text{-}Bici] = t_U[Num\text{-}Bici] \wedge \quad \textit{(join)}$$
$$t_B[TipoBici] = \text{"Corsa"})) \quad \textit{(per una bici da corsa)}$$
$$) \}$$

Datalog (secondo l'interpretazione A):

```
UsataBiciTipo(C, T)  :-   BIC(N, T, _, _), USA(C, N, _, _, _, _)

Sola(Nom)   :-  CLI(Cliente, Nom, _, _),
                ¬UsataBiciTipo(Cliente, "Corsa"),
                PRE(Cli, "Corsa", _, _, _)

? – Sola(Nom)
```

2. Esprimere in algebra relazionale ottimizzata la seguente interrogazione (**2.5 punti**):
Trovare il tipo dell'ultima bicicletta usata da Mario Rossi

Intendendo da "un" Mario Rossi – il nome non è chiave.
Si noti che la parte in blu è necessaria per restringere l'insieme degli usi da sottrarre ai soli usi di Mario Rossi. Diversamente si considererebbero gli ultimi solo gli ultimi usi IN ASSOLUTO, e non gli ultimi tra i soli usi di qualche Mario Rossi).

Senza ottimizzazione rispetto alle proiezioni:

Uso l'assegnamento per rendere più "leggibile la soluzione" – Si rammenta che l'assegnamento consiste nel definire una "sottoquery" il cui nome rappresenta la relazione calcolata come risultato dell'espressione associata, che si può poi liberamente usare in altre espressioni:

$$\text{UltimoUsoMarioRossi} := (\text{USA} \rhd\!<_{\text{Num-Cli=Num}} (\sigma_{\text{Nome="MarioRossi"}} \text{CLI})) - (\text{(USA} \rhd\!<_{\text{Num-Cli=Num}} (\sigma_{\text{Nome="MarioRossi"}} \text{CLI})) \rhd\!<_{\text{Num-Cli=Num-Cli} \wedge (\textbf{Giorno<Giorno} \vee \textbf{Giorno=Giorno} \wedge \textbf{OraInizio} < \textbf{OraInizio})} (\text{USA} \rhd\!<_{\text{Num-Cli=Num}} (\sigma_{\text{Nome="MarioRossi"}} \text{CLI})))$$

Soluzione → Π_{TipoBici} (BIC $\rhd\!<$ UltimoUsoMarioRossi)

Si noti che la soluzione si basa sul fatto che anche se in uno stesso giorno uno stesso cliente non può usare la stessa bici più di una volta non è detto che non usi due bici diverse nello stesso giorno. Quindi non sarebbe sufficiente un $\rhd\!<_{\text{Num-Cli=Num-Cli} \wedge \text{Giorno<Giorno}}$

C. Interrogazioni in SQL (7 PUNTI)

1. *Elencare il nome dei clienti che hanno usufruito di una bicicletta di tipo diverso da quello da loro prenotato[nota: far corrispondere prenotazioni e relativi usi]* **(2 punti)**

```
Select Nome
From CLI C, PRE P, USA U, BIC B
Where C.Num = P.Num-Cli and
      C.Num = U.Num-Cli and
      U.Num-Bici=B.Num-Bici and
      (P.Giorno, P.OraInizio, P.OraFine) = (U.Giorno, U.OraInizio, U.OraFine) and
       B.TipoBici <> P.TipoBici
```

Non è particolarmente rilevante che l'uso corrisponda alla prenotazione anche per quanto riguarda l'ora di riconsegna della bicicletta.

2. *Elencare il nome dei clienti che hanno prenotato una bicicletta senza usarla per piu' di una volta, e che non hanno mai usato una bicicletta (***2 punti***)*

Intersechiamo chi non ha mai usato alcuna bici con chi ha almeno due prenotazioni (intendo "due generiche prenotazioni", non "due volte la **stessa** bici")

```
Select Nome, Num
From CLI
Where Num not in (select Num-Cli from USA)
      intersect
Select Nome, Num
From CLI join PRE on Num = Num-Cli
Group By Nome, Num
Having count(*) > 1
```

3. *Per ogni cliente, indicare la bicicletta usata per il maggior numero di ore ed il corrispondente numero totale di ore* ***(3 punti)***

In caso di pari monte ore di due bici per lo stesso cliente, estae entrambe le coppie (C.Num, U.Num-Bici)

```
SELECT Num, Nome, NumBici, sum(OraFine-OraInizio) as TotOre
FROM CLI C join USA U on Num=Num-Cli
GROUP BY Num, NumBici, Nome
HAVING sum(OraFine-OraInizio) >= ALL
  ( SELECT sum(OraFine-OraInizio)        Controlla che la coppia sia quella di max uso per quel cliente
    FROM USA U2
    WHERE C.Num=U2.Num-Cli
    GROUP BY  NumBici  )
```

L'aggiunta di Nome alla target list rende più leggibile il risultato. Per poter includere tale attributo, però, occorre che sia anche nella clausola GROUP BY. Aggiungerlvelo, peraltro, non modifica la struttura del raggruppamento (perché se nel gruppo ci sono solo tuple relative allo stesso cliente, allora anche il suo Nome sarà uguale).
Lo stesso potrebbe valere per il TipoBici, se si volesse includerlo nello schema della tabella risultato.

Se invece avessimo voluto trovare **per ogni bici** qual è il cliente più affezionato avremmo usato la stessa struttura per la query esterna ma una (leggermente!!) diversa query annidata:

```
SELECT NumBici, TipoBici, Num, Nome, sum(OraFine-OraInizio) as TotOre
FROM CLI C join USA U on Num=Num-Cli
GROUP BY Num, NumBici, Nome, TipoBici
HAVING sum(OraFine-OraInizio) >= ALL
  ( SELECT sum(OraFine-OraInizio)        Controlla che la coppia sia quella di max uso per quella bici
    FROM USA U2
    WHERE U.Num-Bici=U2.Num-Bici
    GROUP BY Num-Cli )
```