Corso di Basi di dati

Secondo Compitino - Fila B - Soluzione
6 giugno 2008

1. Comune(codC,*nomeC*, *provincia*, regione, numAb)
   Risiede(codF, cognome, nome, dataN, via, nCiv, nInt, codC$^{\text{Comune}}$)
   Fiume(nomeF, lunghezza, sorgente$^{\text{Comune}}$, foce$^{\text{Comune}}$)
   Bagna(nomeF$^{\text{Fiume}}$, codC$^{\text{Comune}}$)

   Nomi dei comuni della provincia di Pavia con meno di 1000 abitanti bagnati da un fiume lungo meno di 200 km in cui abita almeno una persona che ha il nome uguale al cognome (es. Armando Armando)

   $\{t : \{nomeC\} \mid (\exists c)((\exists r)((\exists f)((\exists b)(c \in Comune \wedge r \in Risiede \wedge f \in Fiume \wedge b \in Bagna \wedge c[provincia] =' pv' \wedge c[numAb] < 1000 c[codC] = r[codC] \wedge r[nome] = r[cognome] \wedge c[codC] = b[codC] \wedge b[nomeF] = f[nomeF] \wedge f[lunghezza] < 200))))\}$

2. (a)
```
CREATE TABLE Comune
(codC      NUMERIC(4,0) PRIMARY KEY,
 nomeC     VARCHAR(30),
 provincia VARCHAR(20),
 regione   VARCHAR(20),
 numAb     NUMERIC(7,0),
 UNIQUE (nomeC,provincia));

CREATE TABLE Bagna
(nomeF     VARCHAR(30) REFERENCES Fiume,
 codC      NUMERIC(4,0) REFERENCES Comune,
 PRIMARY KEY (nomeF, codC));
```

   (b) i. un fiume non puó sorgere e sfociare nello stesso comune
   vincolo `CHECK (sorgente <> foce)` nella tabella `Fiume`

   ii. ogni fiume deve bagnare i comuni in cui sorge ed in cui sfocia

```
CREATE ASSERTION bagnaSF
CHECK(NOT EXISTS (SELECT *
                  FROM Fiume F
                  WHERE sorgente NOT IN (SELECT codC FROM Bagna WHERE nomeF = F.nomeF)
                     OR foce NOT IN (SELECT codC FROM Bagna WHERE nomeF = F.nomeF)));
```

   (c) i. determinare i nomi delle regioni in cui non sfocia alcun fiume lungo piú di 200 km

```
SELECT regione
FROM Comune
WHERE regione NOT IN (SELECT regione
                      FROM Comune JOIN Fiume ON codC = foce
                      WHERE lunghezza > 200);
```

   ii. determinare le regioni bagnate da almeno cinque fiumi, insieme alla lunghezza media di tali fiumi

```
SELECT regione, AVG(lunghezza)
FROM Fiume NATURAL JOIN Bagna NATURAL JOIN Comune
GROUP BY regione
HAVING COUNT(DISTINCT nomeF) >= 5;
```

   iii. determinare il fiume più lungo tra quelli che attraversano almeno due province distinte del Veneto

```
SELECT nomeF
FROM Fiume
WHERE nomeF IN (SELECT nomeF
                FROM Bagna NATURAL JOIN Comune
                WHERE regione = 'veneto'
                GROUP BY nomeF
                HAVING COUNT(DISTINCT provincia) >= 2)
      AND lunghezza >= ALL(SELECT lunghezza
```

```
                                FROM Fiume
                                WHERE nomeF IN (SELECT nomeF
                                                FROM Bagna NATURAL JOIN Comune
                                                WHERE regione = 'veneto'
                                                GROUP BY nomeF
                                                HAVING COUNT(DISTINCT provincia) >= 2));
```

iv. determinare per ogni fiume, la provincia bagnata da quel fiume con il maggior numero di abitanti provincia e suo numero di abitanti

```
PROVINCE = SELECT provincia, SUM(numAb) AS AbProv
           FROM Coume
           GROUP BY provincia;

SELECT DISTINCT nomeF, provincia
FROM Bagna B NATURAL JOIN Comune NATURAL JOIN PROVINCE
WHERE AbProv >= ALL (SELECT AbProv
                     FROM Bagna NATURAL JOIN Comune NATURAL JOIN PROVINCE
                     WHERE nomeF = B.nomeF);
```

3. RIPARAZIONE(numS, marca, modello, codFProp, nomeProp, telProp, guasto, dataRip, importoRip, tecnico)

(a) i. ogni apparecchiatura (identificata da numero di serie) ha una certa marca, modello e proprietario
`numS` → `marca modello codFProp`

ii. ogni proprietario (identificato da codice fiscale) ha un nome e numero di telefono
`codFProp` → `nomeProp telProp`

iii. ogni numero di telefono corrisponde ad un unico proprietario
`telProp` → `codFProp`

iv. gli apparecchi di una stessa marca e modello sono riparati sempre dallo stesso tecnico
`marca modello` → `tecnico`

v. apparecchiature di diverse marche e modelli possono essere soggette allo stesso tipo di guasto
non formalizzabile tramite dipendenza funzionale

vi. l'importo della riparazione dipende dal tipo di guasto, dalla marca e modello dell'apparecchio e dalla data della riparazione
`guasto marca modello dataRip` → `importoRip`

vii. un'apparecchiatura puó subire riparazioni per guasti differenti nella stessa data
non formalizzabile tramite dipendenza funzionale

viii. ogni guasto di ogni apparecchiatura viene riparato al piú una volta per ogni data
`guasto numS dataRip` → `importoRip tecnico`

(b) insieme minimale:

`marca modello` → `tecnico`
`guasto marca modello dataRip` → `importoRip`
`numS` → `marca`
`numS` → `modello`
`numS` → `codFProp`
`codFProp` → `nomeProp`
`codFProp` → `telProp`
`telProp` → `codFProp`

(c) chiave: (`numS`, `guasto`, `dataRip`)

(d) lo schema non é in 3NF né in BCNF

(e) scomposizione lossless join dello schema in BCNF:
(`codFProp`, `nomeProp`, `telProp`), (`numS`, `marca`, `modello`, `codFProp`), (`numS`, `guasto`, `dataRip`, `importoRip`, `tecnico`)
non preserva ad es. la dipendenza `marca modello` → `tecnico`

(f) scomposizione lossless join dello schema in 3NF che preserva le dipendenze
(`codFProp`, `nomeProp`, `telProp`), (`numS`, `marca`, `modello`, `codFProp`), (`marca`, `modello`, `tecnico`), (`guasto`, `marca`, `modello`, `dataRip`, `importoRip`), (`numS`, `guasto`, `dataRip`)

4. $R = (A, B, C, D, E, F)$, dipendenze funzionali:

$$A\ B \to E \qquad A\ C \to F \qquad A\ D \to B \qquad B \to C \qquad C \to D$$

(a) no, controesempio:

$R =$

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_1	d_1	e_2	f_2

$\Pi_{ABC}(R) \bowtie \Pi_{CDEF}(R) =$

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_1	d_1	e_2	f_2
a_1	b_1	c_1	d_1	e_2	f_2
a_2	b_2	c_1	d_1	e_1	f_1

(b) l'insieme di dipendenze è in forma minimale

(c) chiavi: AB, AC, AD

(d) lo schema é in 3NF ma non in BCNF

(e) scomposizione lossless join in BCNF: (CD), (BC), $(ABEF)$ non preserva ad es. la dipendenza $AD \to B$

(f) lo schema è già in 3NF