Corso di Basi di Dati 1

Secondo Compitino

13 Dicembre 2002

Si consideri lo schema relazionale

SQUADRA(<u>NomeS</u>, Città, Sponsor, ColoriSociali, Allenatore)
GIOCATORE(<u>NTessera</u>, NomeS, Numero, Nome, Cognome, AnnoN, Ruolo)
PARTITA(<u>IdPartita</u>, Giornata, NomeSCasa, NomeSTrasf, GoalCasa, GoalTrasf)
GOAL(<u>IdPartita</u>, <u>Minuto</u>, NTessera, AutoGoal)

relativo alla base di dati di un campionato di calcio. Nella relazione PARTITA gli attributi NomeSCasa e NomeSTrasf memorizzano i nomi delle squadre che disputano la partita in casa e in trasferta, rispettivamente, e gli attributi GoalCasa e GoalTrasf rappresentano il numero di goal segnati da tali squadre. Nella relazione GOAL l'attributo NTessera memorizza il numero di tessera del giocatore che ha segnato il goal, mentre l'attributo AutoGoal é un attributo booleano che vale TRUE se il goal é stato un autogoal.

1. (Indicativamente, punti 2) Considerando la relazione PARTITA, come posso definire IdPartita come chiave primaria e (Giornata,NomeSCasa), (Giornata, NomeSTrasf), (NomeSCasa, NomeSTrasf) come coppie di attributi per cui non possono esistere valori duplicati in PARTITA? Si specifichi il comando SQL che deve essere utilizzato.

2. (Indicativamente, punti 2) Come posso specificare che gli attributi NomeSCasa e NomeSTrasf della relazione PARTITA sono chiavi esterne sulla relazione SQUADRA? Si specifichi il comando SQL che deve essere utilizzato, commentando i comportamenti scelti in caso di violazioni dell'integrità referenziale.

3. Formulare in SQL le seguenti interrogazioni:

   (a) (indicativamente, punti 4) determinare i portieri delle squadre che hanno disputato la partita in cui sono stati segnati più goal;

   (b) (indicativamente, punti 5) determinare le squadre che non hanno mai perso nei derby (derby = partita contro squadra della stessa città);

   (c) (indicativamente, punti 6) per ogni squadra determinare il capocannoniere (giocatore che ha segnato più goal, senza contare gli autogoal);

   (d) (indicativamente, punti 6) determinare la giornata in cui sono stati segnati più goal (anche autogoal) tra quelle in cui c'è stata almeno una vittoria fuori casa ed un pareggio.

4. (Indicativamente, punti 6) Data la relazione

   RISULTATI(<u>NomeS</u>, NumVCasa, NumVTrasf, NumPCasa, NumPTrasf, NumSCasa, NumSTrasf)

   definire una sequenza di comandi di aggiornamento opportuni per inserire in tale relazione le informazioni relative a numero di vittorie, pareggi, sconfitte, in casa e fuori casa, di ogni squadra facendo attenzione alle squadre che non hanno mai vinto/perso/pareggiato in casa/fuori casa.

5. (Indicativamente, punti 3) Utilizzando la relazione RISULTATI definire una query SQL che per ogni squadra mostri: nome squadra, punteggio, media inglese, numero partite vinte, numero partite pareggiate, numero partite perse. Le squadre devono essere restituite in ordine di punteggio decrescente.

   Nota:

   - Regola per calcolo punteggio: 3 punti per vittoria, 1 per pareggio, 0 per sconfitta.
   - Regola per calcolo media inglese: +2 vittoria fuori casa, +1 vittoria in casa o pareggio fuori casa, 0 pareggio in casa o sconfitta fuori casa, -1 sconfitta in casa.

- CREATE TABLE R
  (spec_col$_1$, ..., spec_col$_n$)
  [, PRIMARY KEY (*Lista Nomi Colonne*)]
  [, UNIQUE (*Lista Nomi Colonne*)]
  [, FOREIGN KEY (*Lista Nomi Colonne*)
  REFERENCES R'
  [ON DELETE {NO ACTION | RESTRICT | CASCADE |
  SET NULL | SET DEFAULT}]
  [ON UPDATE {NO ACTION | RESTRICT | CASCADE |
  SET NULL | SET DEFAULT}]]
  [, FOREIGN KEY ...]
  [CHECK *Condizione*];
  ...
  [CHECK *Condizione*];
  );

  spec_col$_i$ = *Nome Colonna$_i$ Dominio$_i$* [DEFAULT *Valore Default*] [NOT NULL] [PRIMARY KEY] [UNIQUE]
  [CHECK *Condizione*], $1 \leq i \leq n$

  CREATE ASSERTION *Nome Vincolo*
  CHECK *Condizione*

- DROP TABLE R

- ALTER TABLE R ADD COLUMN spec_col;

  ALTER TABLE R ALTER COLUMN spec_col;

  ALTER TABLE R DROP COLUMN nome_colonna;

- SELECT $R_{i_1}.C_1, R_{i_2}.C_2, \ldots, R_{i_n}.C_n$
  FROM $R_1, R_2, \ldots, R_k$
  WHERE F;

  - C BETWEEN v_1 AND v_2
  - C IN $(v_1, \ldots, v_n)$ e C IN sq
  - C LIKE pattern, con `pattern` stringa di caratteri che può contenere i caratteri speciali % e _
  - EXISTS sq, NOT EXISTS sq
  - ANY sq, ALL sq
  - UNION, INTERSECT, EXCEPT tra query
  - funzioni di gruppo: MAX, MIN, SUM, AVG, COUNT - per le prime quattro è f(*nome_colonna*), per COUNT ci sono le due varianti COUNT(*) e COUNT(DISTINCT nome_colonna)
  - l'ordine delle clausole (se presenti) è SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY

- INSERT INTO R [$(C_1, \ldots, C_n)$]
  { VALUES $(e_1, \ldots, e_n)$ | sq}

- DELETE R [alias] [WHERE F];

- UPDATE R [alias]
  SET C_1 = {e_1 | NULL }, ..., C_n = {e_n | NULL }
  [WHERE F];

- CREATE VIEW *Nome* [(*Lista Nomi Colonne*)] AS Q
  WITH [[LOCAL | CASCADED] CHECK OPTION];