Progetto Basi di Dati

Francesco Amato
Antonio Di Noia
Anna Sconocchia

11 dicembre 2006

Indice

1 Introduzione

Il progetto realizzato si propone di modellare la nozione di campionato di calcio, con particolare riferimento ai seguenti aspetti:

- catturare i dati ritenuti salienti, relativi sia allo svolgimento di ogni singola partita (es. risultati, reti segnate e loro marcatori, dati tecnici e disciplinari) sia all'andamento globale del campionato (es. classifica, classifica marcatori e dati tecnici globali);
- realizzare due interfacce grafiche separate, una che permetta al cronista l'inserimento in tempo reale dei dati di ciascun incontro, l'altra che consenta agli utenti la consultazione di tali dati;
- realizzare una applicazione web che consenta la consultazione dell'intero sistema.

2 Progettazione concettuale

Le entità partecipanti alla realtà "campionato di calcio" sono state identificate in:

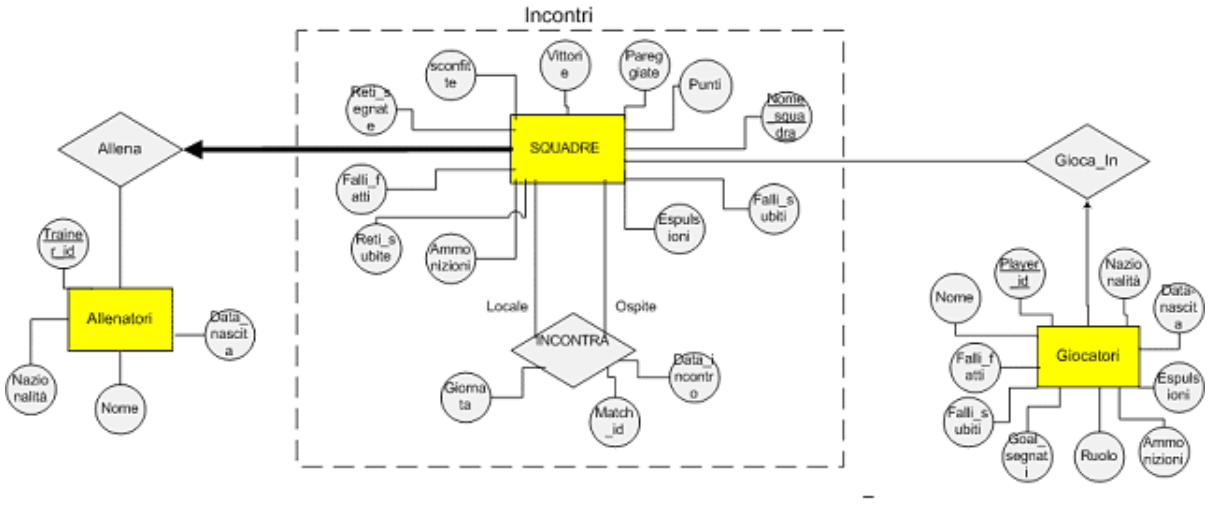
- giocatori, squadre, allenatori e arbitri, che ne sono i protagonisti;
- gli incontri disputati, modellati come entità aggregate;

- le azioni compiute dai giocatori e le regioni del campo in cui queste sono state compiute;
- l'entità-tempo, che consente di contestualizzare tali azioni nel corso di ciascun incontro.

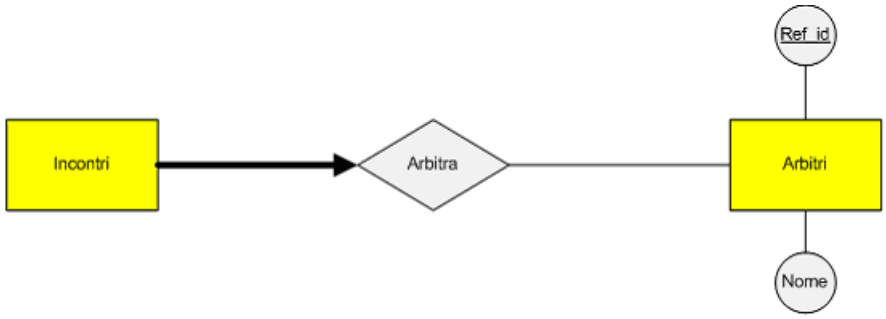
Nel modellare la realtà sono state fatte le seguenti scelte ed assunzioni:

- ciascuna squadra ha uno ed un solo allenatore;
- ciascun giocatore può giocare in una sola squadra;
- è lasciato al cronista l'onere di verificare che agli incontri non stiano prendendo parte squadre prive del numero di giocatori necessario a disputare l'incontro;
- ciascuna partita è arbitrata da uno ed un solo arbitro;
- le percentuali di gioco nelle zone del campo non vengono calcolate dinamicamente a partire da dati del database, ma inserite dal cronista al termine di ciascuna frazione di gioco, supponendo che ad esso siano note in altri modi;
- non accade mai che un giocatore compia più volte una stessa azione nello stesso minuto di una stessa partita;
- se, a campionato in corso, un giocatore abbandona una squadra per passare in una di un altro campionato, esso viene cancellato dalla squadra, ma non dall'insieme di entità "giocatori", in modo che non si perdano informazioni sui dati statistici che lo riguardano (es. reti segnate nel campionato...);
- i giocatori NON sono considerati entità deboli: se una squadra viene cancellata (ad esempio perchè alla fine del campionato è retrocessa) i giocatori rimangono nel database, e verranno cancellati manualmente soltanto quelli che non troveranno impiego in una nuova squadra.

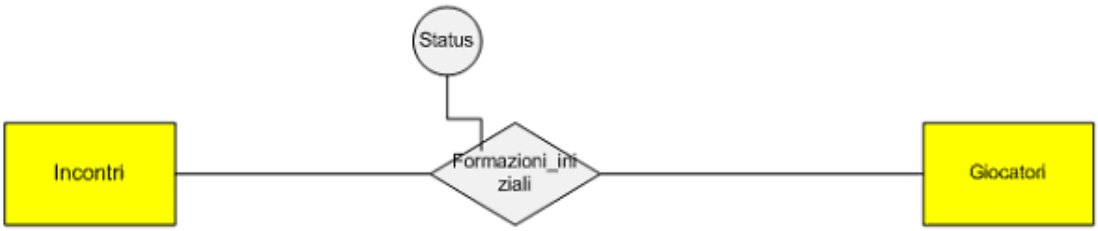
Per motivi di leggibilità, il diagramma Entità-Relazione verrà presentato in maniera non unitaria, ma suddiviso in parti.
Innanzitutto, nella figura seguente si mostra la parte di diagramma relativa a squadre, incontri, ed allenatori.



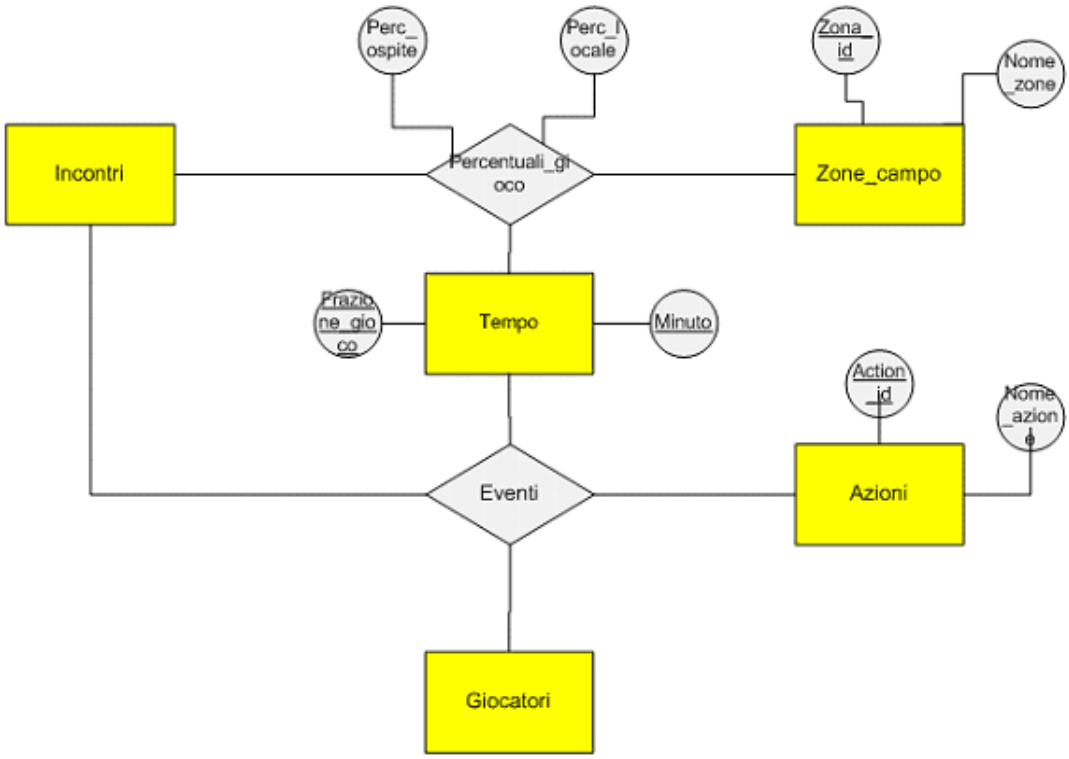
Per semplicità, in ciascuna delle immagini successive si farà riferimento all'aggregazione come ad una unica entità chiamata "incontri", e si ometterà di riportare attributi relativi ad entità già definite in precedenza.
La parte successiva del diagramma mostra l'insieme di relazioni tra gli insiemi di entità "arbitri" ed "incontri", con i relativi vincoli di partecipazione e chiave.



Il fatto che un giocatore possa partecipare ad un incontro partendo da titolare oppure dalla panchina, o non parteciparvi affatto, viene catturato nella seguente parte del diagramma:



L'attributo "status" evidenzia la modalità di partecipazione di ciascun giocatore all'incontro: ovviamente un giocatore non può contemporaneamente essere titolare e partire dalla panchina, per cui la cattura dello status in tale attributo non è limitativa. Si segnala, inoltre, che non si è riusciti ad implementare il vincolo di avere undici giocatori per squadra titolari in una gara, vincolo che sarà poi imposto applicativamente.
Ultima parte del diagramma ER riguarda la gestione degli eventi, che è implementata nella maniera seguente:



L'insieme di entità "azioni" comprende una serie di azioni che il cronista può voler contabilizzare (rete, autorete, tiro, corner...), mentre l'insieme di relazioni "eventi" istanzia tali azioni, ovvero ne specifica, man mano che vengono compiute, la partita in cui avvengono, nonchè l'autore ed il minuto di occorrenza.

3 Progettazione logica

Lo schema relazionale non è esattamente in corrispondenza uno a uno con lo schema ER: in particolare, l'indicatore che codifica ciascun incontro compare nello schema ER come attributo della relazione "incontra", e come tale non può essere assunto come chiave primaria, mentre nello schema relazionale

viene proprio utilizzato come tale, in modo da identificare in maniera sintetica ciascun incontro, senza dover specificare i nomi delle squadre partecipanti quando la tabella "incontri" viene referenziata dall'esterno. Questa discordanza, comunque, non sembra avere controindicazioni nell'implementazione della base di dati. Si riportano di seguito le istruzioni per la creazione delle tabelle del database in MySQL.

```
Create table tempo
(frazione_gioco char(20),
minuto integer,
PRIMARY KEY (frazione_gioco, minuto));

Create table Arbitri
(ref_id integer,
nome char(20),
PRIMARY KEY (ref_id));

Create table Allenatori
(trainer_id integer,
nome char(20),
data_nascita date,
nazionalita char(20),
PRIMARY KEY(trainer_id));

Create table Azioni
(action_id integer,
nome_azione char(20),
PRIMARY KEY (action_id));

Create table Zone_campo
(zona_id integer,
nome_zona char(20),
PRIMARY KEY(zona_id));

Create table Squadre
(nome_squadra char (20),
trainer_id integer NOT NULL,
punti integer,
```

```
vittorie integer,
pareggi integer,
sconfitte integer,
reti_segnate integer,
reti_subite integer,
ammonizioni integer,
espulsioni integer,
falli_fatti integer,
falli_subiti integer,
PRIMARY KEY (nome_squadra),
FOREIGN KEY(trainer_id) references Allenatori(trainer_id) on delete no action);

Create table Giocatori
(player_id integer,
nome char(20),
data_nascita date,
nazionalita char(20),
squadra char(20),
ruolo char(20),
goal_segnati integer,
ammonizioni integer,
esplusioni integer,
falli_fatti integer,
falli_subiti integer,
PRIMARY KEY (player_id),
FOREIGN KEY(squadra) references Squadre(nome_squadra));

Create table incontri
(match_id integer,
squadra_locale char(20),
squadra_ospite char(20),
giornata integer,
data_incontro date,
id_arbitro integer NOT NULL,
PRIMARY KEY (match_id),
FOREIGN KEY (id_arbitro) references Arbitri(ref_id));

Create table formazioni_iniziali
```

```
(id_giocatore integer,
id_partita integer,
status char(20),
PRIMARY KEY(id_giocatore, id_partita),
FOREIGN KEY(id_giocatore) references Giocatori(player_id),
FOREIGN KEY(id_partita) references Incontri(match_id));

Create table percentuali_gioco
(id_partita integer,
id_zona integer,
half char(20),
perc_locale integer,
perc_ospite integer,
PRIMARY KEY(id_partita, id_zona, half),
FOREIGN KEY(id_partita) references Incontri(match_id),
FOREIGN KEY(id_zona) references Zone_Campo(zona_id),
FOREIGN KEY(half) references Tempo(frazione_gioco));

Create table eventi
(id_gara integer,
id_autore integer,
id_azione integer,
min integer,
half char(20),
PRIMARY KEY (id_gara, id_autore, id_azione, min, half),
FOREIGN KEY(id_gara) references Incontri(match_id),
FOREIGN KEY(id_autore) references Giocatori(player_id),
FOREIGN KEY(id_azione) references Azioni(action_id),
FOREIGN KEY(half, min) references Tempo(frazione_gioco, minuto));
```

Si è scelto di fare in modo che il verificarsi di un evento (rete, fallo, provvedimento disciplinare...) causi l'aggiornamento automatico delle relative statistiche dei giocatori e di squadra, con le seguenti limitazioni:

- non si è riusciti a realizzare l'inserimento automatico di un evento "espulsione" nel caso in cui un giocatore venga ammonito due volte in una partita, perchè MySQL non consente triggers annidati di questo

tipo, per cui tale inserimento dovrà necessariamente essere a cura del cronista ;

- l'aggiornamento automatico della classifica deve necessariamente essere delegato al programma applicativo, per il motivo seguente: non è stato trovato il modo per aggiornare la classifica in tempo reale, nè per indicare al database la fine di un incontro, e sarà quindi il cronista, al termine di ogni partita, ad invocare un opportuno comando che causi tale aggiornamento.

Anche in questo caso si riporta di seguito il listato di creazione del trigger di aggiornamento delle statistiche.

```
CREATE TRIGGER statistiche AFTER INSERT ON eventi
FOR EACH ROW
BEGIN
IF new.id_azione=1
THEN
 UPDATE giocatori g
 SET g.goal_segnati=g.goal_segnati+1
 WHERE g.player_id=new.id_autore;
 UPDATE squadre s
 SET s.reti_segnate=s.reti_segnate+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
 IF (SELECT g.squadra FROM giocatori g
     WHERE g.player_id=new.id_autore)=(SELECT i.squadra_locale
                                       FROM incontri i
                                       WHERE i.match_id=new.id_gara)
  THEN
  UPDATE squadre s
   SET s.reti_subite=s.reti_subite+1
   WHERE s.nome_squadra=(SELECT i.squadra_ospite
                         FROM incontri i
                         WHERE i.match_id=new.id_gara);
  ELSE
  UPDATE squadre s
   SET s.reti_subite=s.reti_subite+1
   WHERE s.nome_squadra=(SELECT i.squadra_locale
```

```
                        FROM incontri i
                        WHERE i.match_id=new.id_gara);
  END IF;
ELSE IF new.id_azione=2
THEN
UPDATE squadre s
 SET s.reti_subite=s.reti_subite+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
IF (SELECT g.squadra FROM giocatori g
     WHERE g.player_id=new.id_autore)=(SELECT i.squadra_locale
                                      FROM incontri i
                                      WHERE i.match_id=new.id_gara)
  THEN
  UPDATE squadre s
   SET s.reti_segnate=s.reti_segnate+1
   WHERE s.nome_squadra=(SELECT i.squadra_ospite
                        FROM incontri i
                        WHERE i.match_id=new.id_gara);
  ELSE
  UPDATE squadre s
   SET s.reti_segnate=s.reti_segnate+1
   WHERE s.nome_squadra=(SELECT i.squadra_locale
                        FROM incontri i
                        WHERE i.match_id=new.id_gara);
  END IF;
ELSE IF new.id_azione=11
THEN
 UPDATE giocatori g
 SET g.falli_fatti=g.falli_fatti+1
 WHERE g.player_id=new.id_autore;
 UPDATE squadre s
 SET s.falli_fatti=s.falli_fatti+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
ELSE IF new.id_azione=12
THEN
 UPDATE giocatori g
```

```
 SET g.falli_subiti=g.falli_subiti+1
 WHERE g.player_id=new.id_autore;
 UPDATE squadre s
 SET s.falli_subiti=s.falli_subiti+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
ELSE IF new.id_azione=13
THEN
 UPDATE giocatori g
 SET g.ammonizioni=g.ammonizioni+1
 WHERE g.player_id=new.id_autore;
 UPDATE squadre s
 SET s.ammonizioni=s.ammonizioni+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
ELSE IF new.id_azione=14
THEN
 UPDATE giocatori g
 SET g.espulsioni=g.espulsioni+1
 WHERE g.player_id=new.id_autore;
 UPDATE squadre s
 SET s.espulsioni=s.espulsioni+1
 WHERE s.nome_squadra=(SELECT squadra from giocatori g
          WHERE g.player_id=new.id_autore);
END IF;
END IF;
END IF;
END IF;
END IF;
END IF;
END;
```

4 Progettazione ed implementazione della componente procedurale

La componente procedurale è stata realizzata in modo da implementare due interfacce differenti:

- una che consenta al cronista l'inserimento dei dati puntuali;
- una che permetta all'utente di consultare i dati.

Più precisamente, il database è strutturato in modo che le tabelle "squadre" e "giocatori" siano precompilate, mentre la tabella incontri abbia precompilati gli incontri del campionato, suddivisi per giornate (questo perchè il calendario del campionato è noto all'inizio della stagione). All'inizio di ogni partita, il cronista effettuerà le seguenti azioni:

- inserirà la data ed il nome dell'arbitro (scegliendolo da un elenco), che quindi verranno inseriti nella tabella "incontri", aggiornando valori di default preesistenti;
- inserirà le formazioni iniziali, ricevendo un messaggio di errore (implementato applicativamente) se i titolari inseriti non sono undici per squadra;
- procederà all'inserimento degli eventi, annotandone l'autore ed il minuto di occorrenza;
- inserirà, a fine primo tempo ed a fine partita, i dati sulle percentuali di gioco nelle varie zone del campo.

Il semplice utente sarà in grado di consultare dati quali:

- la classifica del campionato;
- la classifica dei cannonieri;
- dati tecnici relativi ad ogni partita, che può essere scelta da un elenco.

La possibilità di inserire dati nel database è garantita solo previo login con password, in modo che solo i cronisti, e non gli utenti semplici, possano modificare il contenuto della base di dati. A titolo di esempio, come nome

utente si è utilizzato il solito "root" (la password è quella di root), ma una gestione più raffinata della sicurezza dovrebbe prevedere l'impiego di nomi utenti diversi. In altre parole, il meccanismo qui proposto per garantire la sicurezza del database è piuttosto rudimentale: sia che si acceda al database come utente semplice, sia che vi si acceda come cronista, l'accesso viene fatto sempre come root, e con la password di root, ma

- per accedere come utente semplice, ossia per consultare i dati, la password di root viene passata al database in maniera automatica, all'insaputa dell'utente stesso;
- per accedere all'area inserimento dati bisognerà qualificarsi esplicitamente come root e fornire la password.

Tutto si basa su due assunzioni, che sarebbero eccessivamente semplificative se applicate a casi reali:

- un semplice utente accede al database come root, ma non sa di farlo, nè conosce la password, per cui non ha accesso all'area riservata;
- un cronista, invece, conosce nome utente e password e può accedere.

5 Istruzioni per l'uso

Alla relazione sul progetto si allega un CD recante il database di origine, estratto con mysqldump, e tutte le pagine html e php relative alla parte applicativa (non pubblicata su internet in quanto www.db4free.net sembra non comprendere il comando "delimiter" presente nel dump file). Dal momento che le interrogazioni al database fatte dall'applicazione si servono di funzioni, e visto il bug di mysqldump che impedisce la loro esportazione, nel CD è presente una cartella contenente tali funzioni, che devono essere caricate a mano nel database prima di avviare qualsiasi operazione, pena il mancato riconoscimento delle istruzioni. Il nome della cartella è, appunto, "Funzioni".