UNIVERSITÀ
DEGLI STUDI
DI UDINE

LABORATORIO BASI DI DATI
Relazione Progetto

Casasola Thomas 162372

Dalla Costa Daniele 162752

Durì Gioele 158484

Nordio Carlo 163185

Sepulcri Marco 158662

Anno Accademico 2024/2025

Contents

1 RACCOLTA E ANALISI DEI REQUISITI

1.1 STUDIO DELLA RICHIESTA

1.1.1 Richiesta originale

Si progetti uno schema entità/relazioni di un sistema per la gestione informatica delle partite di un campionato di calcio, a partire dalle seguenti specifiche. Per ogni partita, descrivere il girone (andata e ritorno) e la giornata in cui si è svolta (prima giornata del campionato, seconda giornata, ecc..) il numero progressivo nella giornata (es. prima partita della giornata, seconda partita, ecc..), la data, con giorno, mese e anno, le squadre coinvolte nella partita, con nome, città della squadra e allenatore e, infine, per ciascuna squadra, se ha giocato in casa. Si vogliono conoscere i giocatori che giocano in ogni squadra, con i loro nomi e cognomi, la loro data di nascita e il ruolo principale. Si vuole conoscere, per ogni giornata quanti punti ha ogni squadra. Si vogliono anche conoscere per ogni partita, i giocatori che hanno giocato, i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita) e nome, cognome, città e regione di nascita dell'arbitro della partita. Distinguere le partite giocate regolarmente dalle partite rinviate. Per quelle rinviate, rappresentare la data in cui si sono effettivamente svolte. Distinguere anche le partite giocate in una città diversa da quella della squadra ospitante; per queste si vuole rappresentare la città in cui sono svolte; nonché il motivo della variazione di sede. Dei giocatori interessa anche la data di nascita. Dopo aver riorganizzato in gruppi omogenei le specifiche del sistema, rappresentarle attraverso uno schema entità/relazioni. Lo schema dovrà essere completato con attributi ragionevoli per ciascuna entità, identificando le possibili chiavi e relazioni necessarie per la gestione del sistema in esame.

1.1.2 Analisi dei requisiti

Per analizzare questo sistema delle partite di un campionato di calcio, possiamo suddividere le specifiche fornite in entità omogenee per strutturare la base di quello che sarà successivamente lo schema entità-relazioni. Di seguito vengono presentate:

PARTITA:

- Girone (andata, ritorno);
- Data effettiva della partita (SOLO SE è stata rinviata); nda, ., partita della giornata);

SQUADRA:

- Nome;
- Città;
- Allenatore;
- Punti.

GIOCATORI:

- nome;
- cognome;
- data di nascita;
- ruolo principale.

ARBITRO:

- Nome;
- Cognome;
- Città;
- RegioneNascita.

ALLENATORE:

- Nome;
- Cognome.

1.1.3 Vincoli di integrità

Per proseguire il lavoro con la fase di progettazione concettuale, abbiamo effettuato le seguenti assunzioni:

- non ci può essere una partita con una squadra contro sé stessa;
- due squadre possono avere lo stesso stadio (ad esempio Inter e Milan);
- un giocatore non può aver giocato una partita che non è ancora stata giocata;
- un giocatore non può giocare più partite di quante ne abbia giocate la sua squadra;
- un giocatore non può cambiare squadra durante il campionato;
- la città di una squadra deve essere la stessa del suo stadio;
- se una città è presente nel nostro database vuol dire che ha per forza una squadra ed il suo stadio;
- uno stadio può anche non ospitare nessuna partita per tutto il campionato (ad esempio se uno stadio per tutto l'anno è in ristrutturazione tutte le partite della sua squadra saranno giocate in uno stadio diverso);
- se due o più squadre condividono lo stesso stadio (esempio Inter e Milan) queste non possono giocare in casa nella stessa giornata;
- gli esiti possono essere solo VITTORIA, PAREGGIO o SCONFITTA ed assegnano rispettivamente punti pari a: 3, 1, 0;
- in una giornata se una squadra ha esito PAREGGIO anche l'altra deve avere l'esito PAREGGIO. Se una squadra ha esito VITTORIA, l'altra deve avere esito SCONFITTA e viceversa;

1.2 REQUISITI STRUTTURALI

1.2.1 Glossario

TERMINE	DEFINIZIONE	SINONIMI	COLLEGAMENTI
Girone	Fase del campionato in cui le squadre si affrontano in base a un calendario prestabilito.	Fase	"Partita"
Giornata	Insieme delle partite giocate in una specifica data o intervallo di tempo.	Turno, Round	"Esito", "Partita"
Numero progressivo	Identificatore numerico che indica l'ordine di una partita all'interno di una giornata.	Ordine partita	"Partita", "Giornata"
Ruolo principale	Posizione abituale di un giocatore in campo.	Posizione preferita	"Giocatore"
Ruolo in partita	Posizione che un giocatore assume in una specifica partita, che può differire dal ruolo principale.	Posizione temporanea	"Giocatore", "Partita giocata"
Partita regolare	Partita giocata secondo il calendario ufficiale senza rinvii o variazioni.	Partita standard	"Partita","Partita giocata"
Partita rinviata	Partita giocata secondo il calendario ufficiale senza rinvii o variazioni.	Partita sospesa, posticipata	"Partita","Partita giocata"
Partita in sede diversa	Partita disputata in una città differente da quella della squadra ospitante.	Cambio stadio	"Partita","Partita giocata"
Motivo variazione sede	Ragione per cui una partita è stata giocata in una sede diversa da quella prevista.	Causa spostamento	"Partita giocata"

1.3 DATI E OPERAZIONI

Il campionato di calcio prevede che la base di dati venga progettata in modo che faciliti ed efficienti le seguenti operazioni:

1.3.1 Inserimento

Operazione 1: Inserimento di un giocatore in una squadra.

1.3.2 Aggiornamento

Operazione 2: Aggiornamento di una partita da futura a giocata

1.3.3 Cancellazione

Operazione 3: Cancellazione di un giocatore

1.3.4 Query

2/3 operazioni (Inserimento, cancellazione, update)
8 query (prime 3 fornite dal professore):

- Tutte le città in cui la squadra UDINESE ha giocato almeno 2 ed al massimo 3 volte;
- Il numero di giocatori che hanno giocato almeno 3 partite;

- Il numero di giocatori che appartengono alla squadra UDINESE e che hanno giocato solo partite contro o la squadra INTER o la squadra MILAN;
- Il nome delle squadre che hanno giocato almeno una partita in casa in uno stadio diverso dal proprio;
- il CF degli arbitri dell'Illinois che hanno diretto esattamente 1 o 3 partite;
- le partite rinviate che sono state giocate successivamente nello stadio della squdra in trasferta;
- Le squadre che hanno in comune la città con un altra squadra;
- Tutte le squadre che hanno vinto almeno 15 partite alla fine del campionato.

1.3.5 Tabella delle frequenze

OPERAZIONE	TIPO	FREQUENZA
Operazione 1	Inserimento	22 per giorno
Operazione 2	Aggiornamento	20 per giorno
Operazione 3	Cancellazione	500 per giorno
Operazione 4	Query	1 per giorno
Operazione 5	Query	500 per giorno
Operazione 6	Query	1 per giorno
Operazione 7	Query	20 per giorno
Operazione 8	Query	20 per giorno
Operazione 9	Query	10 per giorno
Operazione 10	Query	1 per giorno
Operazione 11	Query	1 per giorno

1.3.6 Tabella dei volumi

Concetto	Tipo	Volume
Partita	E	380
Stadio	E	18
Squadra	E	20
Città	E	18
Giornata	E	38
Arbitro	E	12
Allenatore	E	20
Giocatore	E	500
PianificataIn	R	380
Ospita	R	380
EIn	R	18
Possiede	R	20
EDi	R	20
Esito	R	760
Casa	R	380
Trasferta	R	380
Dirige	R	380
HaGiocato	R	11400
EAllenataDa	R	20
Appartiene	R	500

2 PROGETTAZIONE CONCETTUALE

2.1 Modello E/R

Definiti i requisiti iniziali e le regole di interazione tra le entità si procede riassumendo tali informazioni grezze mediante un modello grafico detto "E/R", il quale permette di ottenere una rappresentazione chiara e comprensibile della struttura concettuale dei dati del sistema di riferimento e gioca un ruolo fondamentale nel proseguo della progettazione della base di dati.

Sono state individuate le seguenti entità:

- Squadra;
- Stadio;
- Città;
- Giornata;
- Persona:
  - Arbitro;
  - Allenatore;
  - Giocatore;
- Partita:
  - Futura;
  - Giocata:
    - Rinviata;
    - Rgolare;
    - Stadio di Proprietà;
    - Stadio Diverso.

Queste entità si relazionano tra loro nel seguente modo:

- L'entità **Squadra** si relaziona con sei entità:
  - Giocatore → La cardinalità è N:1 (molti a uno): Ogni giocatore deve appartenere ad una squadra, ed ogni squadra può avere più giocatori.
  - Giornata → La cardinalità è N:N (molti a molti): una squadra gioca in molte giornate, ed una giornata ha molte squadre che giocano;

- Città → La cardinalità è 1:N (uno a molti): una squadra è di una città ed una città può avere molte squadre;
- Stadio → La cardinalità è 1:N (uno a molti): Una squadra può possedere uno stadio ed uno stadio può essere posseduto da una squadra;
- Partita → La cardinalità è N:1 (molti a uno): una squadra può essere la squadra di "casa" in molte partite e può essere la squadra "ospite" in molte partite. Da notare che le due relazioni sono separate, quindi una squadra gioca n partite in casa e n partite in trasferta, non n partite totali divise tra casa e trasferta.
- Allenatore → La cardinalità è 1:1 (uno a uno): Una squadra è allenata da un allenatore ed un allenatore allena una squadra.

- L'entità persona viene suddivisa, tramite una specializzazione totale e disgiunta, in tre entità:
  - L'entità **Giocatore**, la quale si relaziona con due entità:
    - Giocata → La cardinalità è N:N (motli a molti)
    - Squadra → La cardinalità è 1:N (uno a molti)
  - L'entità **Arbitro**, la quale si relaziona con due entità:
    - Giocata → La cardinalità è N:1 (motli a uno): un arbitro può dirigere più partite giocate, ed una partita giocata può essere diretta da un solo arbitro.
  - L'entità **Allenatore**, la quale si relazione con un'entità:
    - Squadra.

- L'entità **Partita** si relaziona con tre entità:
  - Giornata → La cardinalità è 1:N (uno a molti): una partita è pianificata in una giornata ed una giornata può avere molte partite.
  - Stadio → La cardinalità è 1:N (uno a molti): una partita si gioca in uno stadio ed uno stadio può ospitare molte partite;
  - Squadra.

    L'entità persona viene suddivisa, tramite una specializzazione totale e disgiunta, in 2 entità:
    - L'entità **Futura**;
    - L'entità **Giocata**, che si relaziona con due entità:
      - Arbitro → La cardinalità è 1:N (uno a molti)
      - Giocatore.

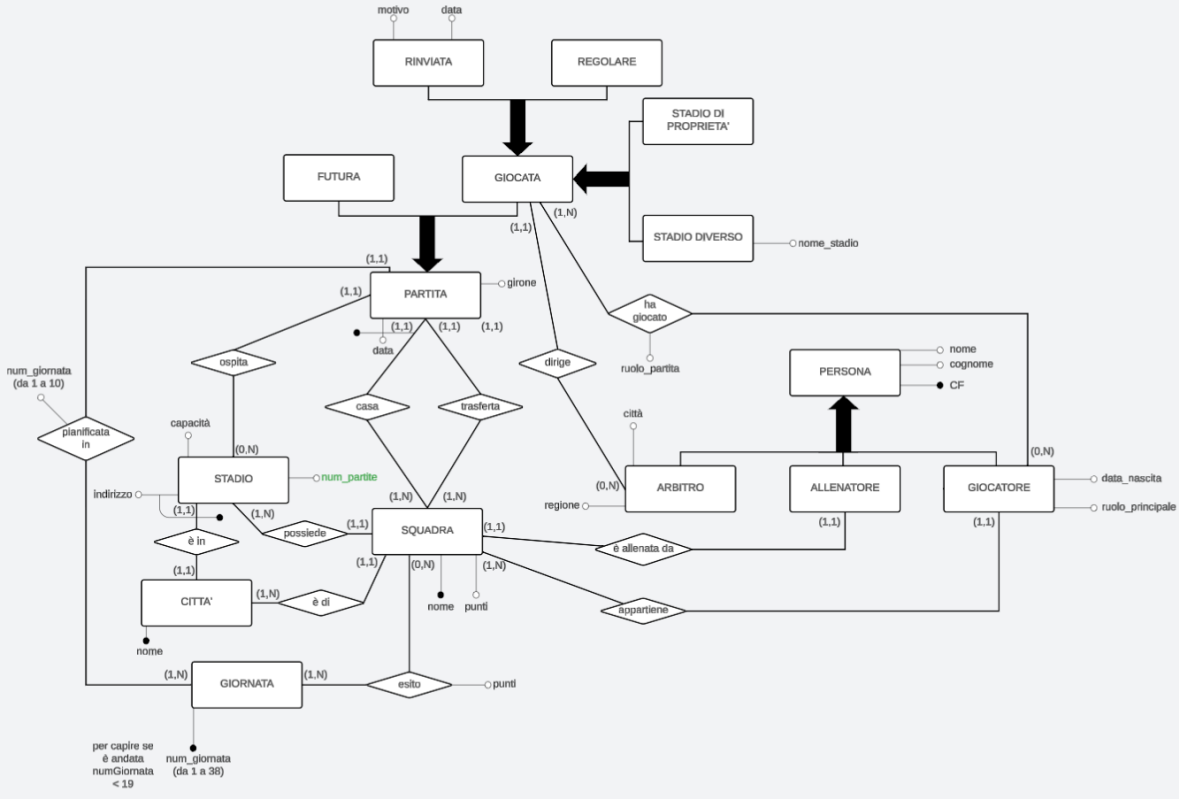
      Quest'ultima si suddivide in due specializzazioni totali e disgiunte: La prima in

- Rinviata;
- Regolare.

La seconda in

- Stadio di proprietà;
- Stadio diverso.

- L'entità **Città**, che si relaziona con due entità:
  - Stadio;
  - Squadra.



3 PROGETTAZIONE LOGICA

3.1 RISTRUTTURAZIONE MODELLO E/R

Dopo aver definito la struttura concettuale dei dati, si procede con la fase di "ristrutturazione" dello schema ER. Questa fase consiste nell'apportare modifiche per ridurre il livello di astrazione e preparare la progettazione logica della base di dati. Nel nostro caso, la progettazione logica sarà realizzata utilizzando il modello relazionale.

3.1.1 Analisi delle ridondanze

La prima fase consiste nell'eseguire un'analisi delle ridondanze per capire quali di queste conviene considerare e a quali si può rinunciare in termini di efficienza.

3.1.2 Analisi dell'attributo derivato `Num_Partite`

Nello schema E/R l'attributo `Num_Partite` è ridondante in quanto si può ricavare tale valore anche attraverso il numero di relazioni "ospita" tra stadio e partita. Si effettua quindi, un'analisi delle ridondanze sulla base delle operazioni 'Inserimento di una nuova partita' e 'Numero di partite per ogni stadio' definite nella tabella delle operazioni.

3.1.3 Tabella degli accessi in presenza di ridondanza

Si considera l'operazione di 'Inserimento di una nuova partita' effettuata 10 volte al giorno.

Concetto	Costrutto	Accessi	Tipo
Partita	E	1	S
Ospita	R	1	S
Stadio	E	1	L
Stadio	E	1	S

In presenza di ridondanza l'operazione "Inserimento di una nuova partita" richiederà:

- 10×3 (accessi in scrittura) e 10×1 (accessi in lettura);
- $(10\times3)\times2$(raddoppio delle scritture)$+(10\times1)=70$ unità di operazioni totali.

Si considera l'operazione "Numero di partite per ogni stadio" effettuata 10 volte al giorno:

Concetto	Costrutto	Accessi	Tipo
Stadio	E	1	L

In presenza di ridondanza l'operazione "Numero di partite per ogni stadio" richiederà: (10×1)×1=10 letture in Stadio In totale, queste due operazioni, con presenza di ridondanza, richiedono (70+10) 80 accessi totali.

3.1.4 Tabella degli accessi in assenza di ridondanza

Si considera l'operazione di "Inserimento di una nuova partita" effettuata 10 volte al giorno.

Concetto	Costrutto	Accessi	Tipo
Partita	E	1	S
Ospita	R	1	S

In assenza di ridondanza l'operazione "Inserimento di una nuova partita" richiederà: (10×2)×2(raddoppio delle scritture)=40 scritture in Partita,Ospita.

Si considera l'operazione 'Numero di partite per ogni stadio' effettuata 10 volte al giorno.

Concetto	Costrutto	Accessi	Tipo
Stadio	E	1	L
Ospita	R	21,11111111	L
Partita	E	1	L

In assenza di ridondanza l'operazione "Numero di partite per ogni stadio" richiederà:

- Dobbiamo accedere a ospita un numero medio di volte pari al numero medio di partite ospitate in uno stadio= (volume di "partita"/volume di "stadio")=380/18=21,1111;
- 10 query al giorno×21,111 partite medie per ogni stadio=211,1111 operazioni in lettura in Stadio, Ospita, Partita;
- Si ottengono quindi 211,1111 unità di operazione al giorno.

In totale, queste due operazioni, con assenza di ridondanza, richiedono (40+211,11111) 251,1111 accessi totali.

A fronte dell'analisi effettuata possiamo vedere come le due operazioni con presenza di ridondanza richiedono 80 accessi totali, mentre le due operazioni in assenza di ridondanza richiedono 251,1111 accessi totali, possiamo stabilire che sia più conveniente mantenere l'attributo `Num_Partite` nonostante la ridondanza.

3.1.5 Eliminazione delle generalizzazioni

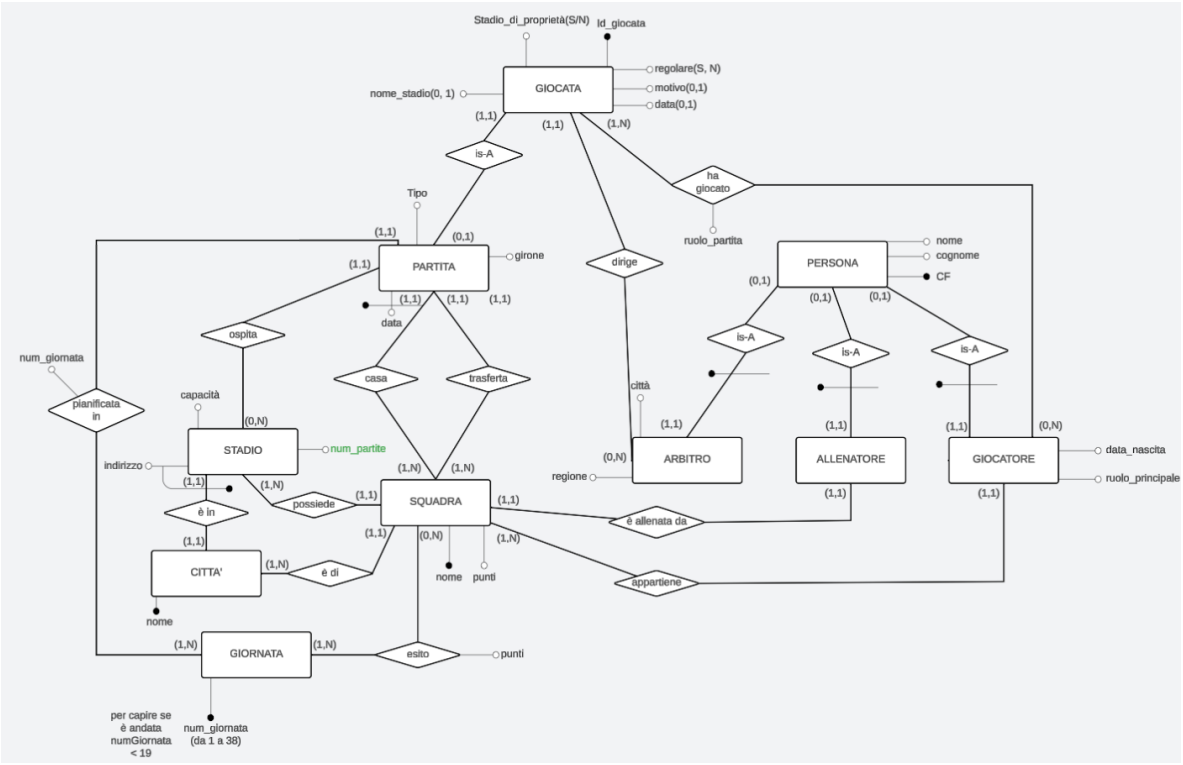
Al fine di ristrutturare lo schema E/R per proseguire con la fase di progettazione logica, è necessario rimuovere le generalizzazioni. Esistono tre pattern di ristrutturazione per fare ciò:

- Accorpamento delle entità figlie nell'entità padre;
- Accorpamento dell'entità padre nelle entità figlie;
- Sostituzione della generalizzazione con associazioni.

Nello schema E/R sono presenti quattro generalizzazioni: una per "Partita", due per "Giocata" ed una per "Persona".

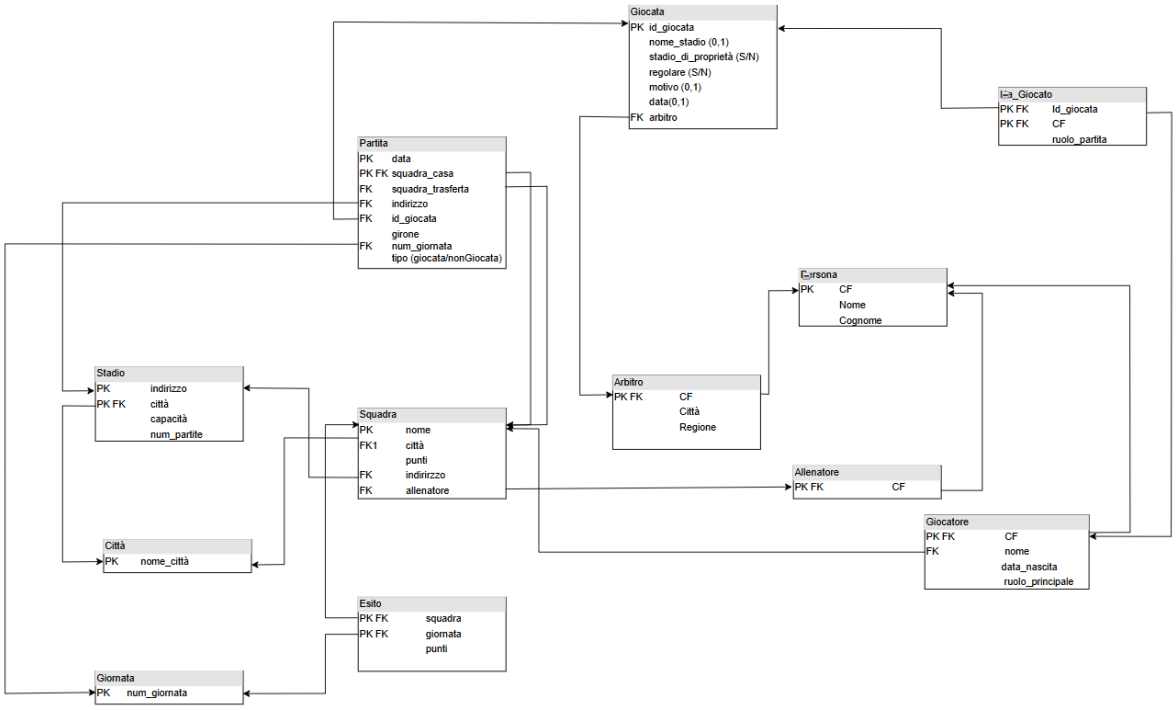
- "Partita": è presente una specializzazione nella quale si specifica se una partita è stata giocata oppure rinviata. In questo caso abbiamo deciso di accorpare l'entità figlia 'Rinviata' all'interno dell'entità padre 'Partita', tramite l'attributo 'tipo' che definisce se la partita è stata giocata o meno. L'entità figlia giocata, invece, diventa una nuova associazione denominata 'is-A'.
- "Giocata": in questo caso sono presenti due generalizzazioni, di cui una ha lo scopo di differenziare le partite giocate nella data prestabilita e le partite rinviate; mentre l'altra generalizzazione divide le partite giocate nello stadio di proprietà e quelle in uno stadio diverso. In entrambi i casi abbiamo deciso di effettuare l'accorpamento delle entità figlie nell'entità padre: quindi resta una sola entità ("Giocata") con tutti gli attributi ed associazioni originali, più quelli delle entità figlie (non obbligatori).
- "Persona": nella fattispecie si è sostituita la generalizzazione con delle nuove associazioni, denominate "is-A", di tipo 1:1.

3.1.6 Schema E/R normalizzato



3.2 SCHEMA RELAZIONALE

Verranno illustrate le tabelle, le chiavi primarie e le chiavi esterne, nonché i vincoli di integrità volti a preservare la consistenza dei dati. Nell'immagine sottostante (Figura 3) è riportato lo schema relazionale derivato dal nostro E/R:



- La relazione “dirige” tra Giocata e Arbitro porta all’aggiunta di un attributo di chiave esterna alla relazione Giocata denominata "arbitro" che “punta alla chiave primaria” dell’entità Arbitro. Ciò vuol dire che, presa una generica tupla della tabella Arbitro, il suo attributo “arbitro” dovrà avere un valore uguale al valore “CF” di una tupla della tabella Arbitro. In modo analogo vengono tradotte le relazioni tra le tabelle Partita-Stadio, Giocatore-Squadra, Allenatore-Squadra, Squadra-Città, Squadra-Stadio e Partita-Giornata. Nella traduzione da schema E/R a schema relazionale è presente un caso particolare, per il quale si è utilizzata la metologia descritta prima, tra le tabelle Partita e Squadra: in questa situazione sono presenti due relazioni differenti tra queste tabelle ("casa" e "trasferta"), di conseguenza Partita avrà due chiavi esterne: una dal nome "casa" che punta alla chiave primaria di Squadra (questa però, insieme all’attributo "data", è anche chiave primaria) e l’altra dal nome "trasferta" che punta alla chiave primaria di Squadra.

- Le tre relazioni "is-A" (tra le tabelle Persona-Giocatore, Persona-Arbitro e Persona-Allenatore) sono state gestite nel seguente modo: Persona è diventata una tabella avente come chiave primaria "CF" e come attributi "nome" e "cognome", mentre Allenatore diventa anch’esso un’entità avente come chiave primaria la chiave esterna "CF" che punta alla chiave primaria di Persona. In modo analogo si sono effettuate le stesse operazioni con Giocatore e Arbitro. Esiste anche un quarto caso di relazione "is-A", che è quello rappresentato tra le tabelle Giocata-Partita, in cui è presente una partecipazione opzionale dell’entità Partita espressa con

un vincolo (0,1), la quale viene tradotta, in termini di modello relazione con la semplice aggiunta della chiave esterna "id_giocata" alla tabella Partita.

- Nello schema E/R sono presenti anche delle relazioni molti a molti (N-N). Questa casistica è rapprentata dalle tabelle Giocatore-Giocata e Partita-Giornata. In entrambi i casi la relazione tra le due tabelle diventa una nuova tabella, avente come chiavi primarie le chiavi esterne che puntato alle due tabelle più gli attributi della relazione.

4 PROGETTAZIONE FISICA

4.1 Domini

Sono stati definiti domini per gli attributi che richiedono una particolare formattazione (come il codice fiscale o un indirizzo) e per condizioni di check che ricorrono all'interno del sistema in questione:

- **cf**: accetta solo stringhe che rispettano il formato di un codice fiscale.
- **tipo**: accetta solo le stringhe 'giocata' o 'non giocata' per definire lo stato di una partita.
- **ruolo**: accetta solo stringhe relative al ruolo di un giocatore che può essere 'portiere', 'difensore', 'centrocampista' o 'attaccante'.
- **girone**: accetta solo le stringhe 'andata' o 'ritorno' per definire il girone in cui viene giocata una partita.
- **indirizzo**: accetta solo stringhe che rispettano il formato di un indirizzo, quindi con il prefisso "Via".
- **SiNo**: accetta solo le stringhe 'S' o 'N' per indicare se una partita è stata giocata regolarmente o no.

4.2 Creazione delle Tabelle

Di seguito viene riportato il codice SQL per la creazione delle tabelle:

```sql
CREATE TABLE "citta" (
    "nome_citta" varchar(30) PRIMARY KEY
);

CREATE TABLE "stadio" (
    "indirizzo" indirizzo UNIQUE,
    "citta" varchar(30),
    "capacita" integer CHECK ("capacita" >= 0),
    "num_partite" integer CHECK ("num_partite" >= 0),
    PRIMARY KEY ("indirizzo", "citta"),
    FOREIGN KEY ("citta") REFERENCES "citta" ("nome_citta") ON DELETE CASCADE
);

CREATE TABLE "giornata" (
    "num_giornata" integer PRIMARY KEY CHECK ("num_giornata" BETWEEN 1 AND 38)
);
```

```
CREATE TABLE "squadra" (
    "nome" varchar(30) PRIMARY KEY,
    "punti" integer,
    "indirizzo" varchar(30),
    "citta" varchar(30),
    FOREIGN KEY ("indirizzo", "citta") REFERENCES "stadio" ("indirizzo", "citta"),
    FOREIGN KEY ("citta") REFERENCES "citta" ("nome_citta")
);

CREATE TABLE "persona" (
    "cf" cf PRIMARY KEY,
    "nome" varchar(30),
    "cognome" varchar(30)
);

CREATE TABLE "giocatore" (
    "cf" cf PRIMARY KEY,
    "data_nascita" date,
    "ruolo_principale" ruolo,
    "squadra" varchar(30) REFERENCES "squadra" ("nome"),
    FOREIGN KEY ("cf") REFERENCES "persona" ("cf") ON DELETE CASCADE
);



CREATE TABLE "allenatore" (
    "cf" cf PRIMARY KEY,
    "squadra" varchar(30) UNIQUE,
    FOREIGN KEY ("squadra") REFERENCES "squadra" ("nome"),
    FOREIGN KEY ("cf") REFERENCES "persona" ("cf") ON DELETE CASCADE
);

CREATE TABLE "arbitro" (
    "cf" cf PRIMARY KEY,
    "citta" varchar(30),
    "regione" varchar(30),
    FOREIGN KEY ("cf") REFERENCES "persona" ("cf") ON DELETE CASCADE
);

CREATE TABLE "giocata" (
    "id_giocata" integer PRIMARY KEY,
    "data" date NULL,
    "regolare" SiNo,
    "motivo" varchar(50) NULL,
    "stadio_proprieta" SiNo,
    "nome_stadio" varchar(30) NULL,
    "arbitro" cf REFERENCES "arbitro" ("cf")
);
```

```sql
CREATE TABLE "partita" (
    "data" date,
    "tipo" tipo,
    "girone" girone,
    "squadra_casa" varchar(30),
    "squadra_trasferta" varchar(30),
    "indirizzo" varchar(30),
    "citta_stadio" varchar(30),
    "id_giocata" integer,
    "num_giornata" integer,
    PRIMARY KEY ("data", "squadra_casa"),
    FOREIGN KEY ("squadra_casa") REFERENCES "squadra" ("nome"),
    FOREIGN KEY ("squadra_trasferta") REFERENCES "squadra" ("nome"),
    FOREIGN KEY ("id_giocata") REFERENCES "giocata" ("id_giocata"),
    FOREIGN KEY ("num_giornata") REFERENCES "giornata" ("num_giornata"),
    FOREIGN KEY ("indirizzo") REFERENCES "stadio" ("indirizzo")
);

CREATE TABLE "esito" (
    "squadra" varchar(30),
    "giornata" integer,
    "punti" integer,
    PRIMARY KEY ("squadra", "giornata"),
    FOREIGN KEY ("squadra") REFERENCES "squadra" ("nome") ON DELETE CASCADE,
    FOREIGN KEY ("giornata") REFERENCES "giornata" ("num_giornata") ON DELETE SET NULL
);
```

```sql
CREATE TABLE "ha_giocato" (
    "id_giocata" integer,
    "cf" cf,
    "ruolo_partita" ruolo,
    PRIMARY KEY ("id_giocata", "cf"),
    FOREIGN KEY ("id_giocata") REFERENCES "giocata" ("id_giocata") ON DELETE CASCADE,
    FOREIGN KEY ("cf") REFERENCES "giocatore" ("cf") ON DELETE CASCADE
);
```

4.3 Indici secondari

Gli indici svolgono un ruolo cruciale nel migliorare le prestazioni delle interrogazioni, in particolare in tabelle con un elevato numero di record.

Nel caso del sistema in esame, sono stati definiti i seguenti indici secondari:

```sql
-- INDICI SECONDARI
CREATE INDEX idx_stadio_citta ON stadio(citta);
CREATE INDEX idx_partita_data ON partita(data);
```

5 IMPLEMENTAZIONE

5.1 TRIGGER

5.1.1 Trigger 1

Modifica del CF dell'albitro –> verificare che si modifichi anche in "Giocata".

```
CREATE OR REPLACE FUNCTION VerArbitroCF()
RETURNS TRIGGER AS $$
BEGIN
  IF OLD.cf IS DISTINCT FROM NEW.cf THEN
    UPDATE Giocata
    SET arbitro = NEW.cf
    WHERE arbitro = OLD.cf;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Verifica_CF_arbitro
BEFORE UPDATE ON Arbitro
FOR EACH ROW EXECUTE FUNCTION VerArbitroCF();
```

5.1.2 Trigger 2

Aggiornamento dell'indirizzo in Squadra (quindi se si modifica "indirizzo" in "stadio" bisogna aggiornare "indirizzo" in "Squadra").

```
CREATE OR REPLACE FUNCTION AggiornaIndirizzoSquadra()
RETURNS TRIGGER AS $$
BEGIN
  IF OLD.indirizzo IS DISTINCT FROM NEW.indirizzo THEN
    UPDATE Squadra
    SET indirizzo = NEW.indirizzo
    WHERE indirizzo = OLD.indirizzo
      AND citta = NEW.citta;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

5.1.3 Trigger 3

Verificare che l'inserimento di un giocatore appartenga ad una squadra

```
CREATE OR REPLACE FUNCTION VerificaGiocatoreSquadra()
RETURNS TRIGGER AS $$
DECLARE
  cnt INTEGER;
BEGIN
  IF NEW.squadra IS NULL THEN
    RAISE EXCEPTION 'Il giocatore deve appartenere ad una squadra.';
  END IF;
  SELECT COUNT(*) INTO cnt
  FROM Squadra
  WHERE nome = NEW.squadra;

  IF cnt = 0 THEN
    RAISE EXCEPTION 'La squadra % non esiste.', NEW.squadra;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TriggerVerificaGiocatoreSquadra
BEFORE INSERT ON Giocatore
FOR EACH ROW
EXECUTE FUNCTION VerificaGiocatoreSquadra();
```

5.2 QUERY

5.2.1 Tutte le città in cui la squadra Udinese ha giocato almeno 2 ed al massimo 3 volte

```
SELECT DISTINCT S.citta
FROM Squadra SCasa, Squadra STrasferta, Partita P, Giocata G, Stadio S
WHERE SCasa.nome = P.squadra_casa AND
      STrasferta.nome = P.squadra_trasferta AND
      P.id_giocata = G.id_giocata AND
      S.indirizzo = P.indirizzo AND
      (SCasa.nome = 'Udinese' OR STrasferta.nome = 'Udinese') AND
      EXISTS(SELECT * --Almeno 2
                FROM Squadra SCasa2, Squadra STrasferta2, Partita P2, Giocata G2, Stadio S2
                WHERE SCasa2.nome = P.squadra_casa AND
                      STrasferta2.nome = P2.squadra_trasferta AND
                      P2.id_giocata = G2.id_giocata AND
                      S2.indirizzo = P2.indirizzo AND
                      (SCasa2.nome = 'Udinese' OR STrasferta2.nome = 'Udinese')  AND
                      P2.data <> P.data AND
                      S2.citta = S.citta
                      )
    AND NOT EXISTS(SELECT * -- Al max 3 (esattamente 3)
                    FROM Squadra SCasa2, Squadra STrasferta2, Partita P2, Giocata G2, Stadio S2,
                         Squadra SCasa3, Squadra STrasferta3, Partita P3, Giocata G3, Stadio S3,
                         Squadra SCasa4, Squadra STrasferta4, Partita P4, Giocata G4, Stadio S4
                    WHERE SCasa2.nome = P2.squadra_casa AND
                          STrasferta2.nome = P2.squadra_trasferta AND
                          P2.id_giocata = G2.id_giocata AND
                          S2.indirizzo = P2.indirizzo AND
                          (SCasa2.nome = 'Udinese' OR STrasferta2.nome = 'Udinese')  AND

                          SCasa3.nome = P3.squadra_casa AND
                          STrasferta3.nome = P3.squadra_trasferta AND
                          P3.id_giocata = G3.id_giocata AND
                          S3.indirizzo = P3.indirizzo AND
                          (SCasa3.nome = 'Udinese' OR STrasferta3.nome = 'Udinese')  AND

                          SCasa4.nome = P4.squadra_casa AND
                          STrasferta4.nome = P4.squadra_trasferta AND
                          P4.id_giocata = G4.id_giocata AND
                          S4.indirizzo = P4.indirizzo AND
                          (SCasa4.nome = 'Udinese' OR STrasferta4.nome = 'Udinese')  AND

                          P2.data <> P.data AND P2.data <> P3.data AND P2.data <> P4.data AND P3.data <> P4.data
                          AND P3.data <> P.data AND P.data <> P4.data AND

                          S2.citta = S.citta AND S3.citta = S.citta AND S4.citta <> S.citta
                    )
```

5.2.2 Il numero di giocatori che hanno giocato almeno 3 partite

```
SELECT COUNT(DISTINCT cf) AS NumGiocatori
FROM ha_giocato g1
WHERE EXISTS (
    SELECT *
    FROM ha_giocato g2
    WHERE g2.cf = g1.cf
      AND g1.id_giocata <> g2.id_giocata
      AND EXISTS (
            SELECT *
            FROM ha_giocato g3
            WHERE g3.cf = g1.cf
              AND g3.id_giocata <> g2.id_giocata
              AND g1.id_giocata <> g3.id_giocata
      )
)
```

5.2.3 Il numero di giocatori che appartengono alla squadra Udinese e che hanno giocato solo partite contro o la squadra Inter o la squadra Milan

```
SELECT COUNT(DISTINCT cf) AS NumGiocatori
FROM giocatore g1
WHERE g1.squadra = 'Udinese' AND
      NOT EXISTS(SELECT *
                 FROM ha_giocato h1, partita p1, giocatore g2
                 WHERE p1.id_giocata= h1.id_giocata AND
                       g2.cf = g1.cf AND
                       h1.cf = g2.cf AND
                       (p1.squadra_casa <> 'Milan' OR p1.squadra_trasferta <> 'Milan') AND
                       (p1.squadra_casa <> 'Inter' OR p1.squadra_trasferta <> 'Inter')
                 )
```

5.2.4 Il nome delle squadre che hanno giocato almeno una partita in casa in uno stadio diverso dal proprio

```
SELECT DISTINCT squadra_casa
FROM giocata g1, partita p1
WHERE g1.id_giocata = p1.id_giocata AND
      g1.stadio_proprieta = 'N'
```

5.2.5 Il CF degli arbitri di Illinois che hanno diretto esattamente 1 o 3 partite

```
SELECT cf
FROM arbitro a1, giocata g1
WHERE a1.citta = 'Illinois' AND
      a1.cf = g1.arbitro AND
      NOT EXISTS(SELECT *
                 FROM arbitro a2, giocata g2
                 WHERE a1.cf = a2.cf AND
                       a2.cf = g2.arbitro AND
                       g1.id_giocata <> g2.id_giocata)
UNION
SELECT a1.cf
FROM arbitro a1, arbitro a2, arbitro a3, giocata g1, giocata g2, giocata g3
WHERE a1.citta = 'Illinois' AND
      a1.cf = g1.arbitro AND
      a2.cf = g2.arbitro AND
      a3.cf = g3.arbitro AND
      a1.cf = a2.cf AND
      a2.cf = a3.cf AND
      g1.id_giocata <> g2.id_giocata AND
      g2.id_giocata <> g3.id_giocata AND
      g1.id_giocata <> g3.id_giocata AND
      NOT EXISTS (SELECT *
                  FROM arbitro a4, giocata g4
                  WHERE a4.cf = g4.arbitro AND
                        a4.cf = a1.cf AND
                        g1.id_giocata <> g4.id_giocata AND
                        g2.id_giocata <> g4.id_giocata AND
                        g3.id_giocata <> g4.id_giocata )
```

5.2.6 Le partite rinviate che sono state giocate successivamente nello stadio della squadra in trasferta

```
SELECT g1.id_giocata
FROM giocata g1, partita p1, squadra qc, squadra qt, stadio sc, stadio st
WHERE g1.id_giocata = p1.id_giocata AND
      p1.squadra_casa = qc.nome AND
      p1.squadra_trasferta = qt.nome AND
      qc.indirizzo = sc.indirizzo AND
      qt.indirizzo = st.indirizzo AND
      p1.tipo = 'nonGiocata' AND
      g1.stadio_proprieta = 'N' AND
      g1.nome_stadio = qt.indirizzo
```

5.2.7 Le squadre che hanno in comune la città con un altra squadra

```
SELECT nome
FROM squadra s1
WHERE EXISTS (SELECT citta
              FROM squadra s2
              WHERE s1.nome <> s2.nome AND
                    s1.citta = s2.citta)
```

5.2.8 Tutte le squadre che hanno vinto almeno 15 partite alla fine del campionato

```
CREATE VIEW ContaVittorie AS
SELECT squadra, COUNT(*) AS conteggio
FROM esito e1
WHERE e1.punti = 3
GROUP BY squadra

SELECT cv1.squadra
FROM ContaVittorie cv1
WHERE cv1.conteggio >= 15
```

5.3 POPOLAMENTO

Il popolamento della base di dati è stato effettuato in modo automatizzato tramite l'esecuzione di uno script in R che, dopo aver stabilito la connessione con il database, inserisce valori pseudo-casuali nelle tabelle rispettando i vincoli di integrità. In particolare, prima di avviare lo script di popolamento, è necessario creare le tabelle, definire i domini, impostare gli indici e dichiarare i trigger. Una volta verificata l'installazione corretta delle librerie R utilizzate nel codice e configurata la connessione al proprio server PostgreSQL, è possibile eseguire lo script completo per popolare la base di dati.

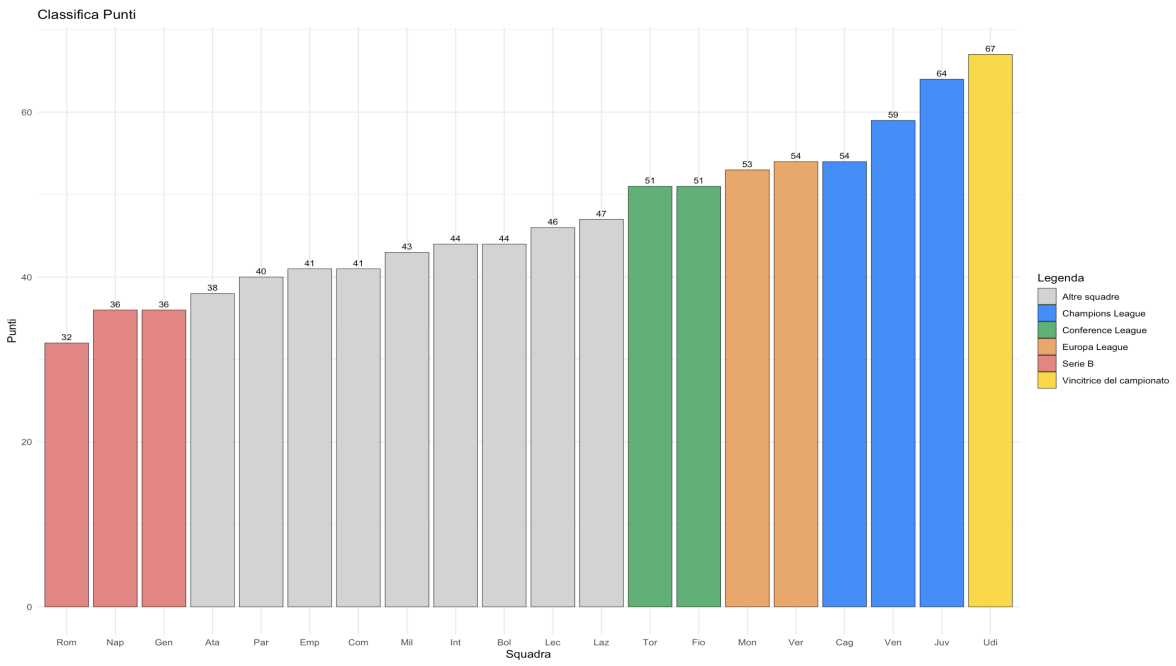
6 ANALISI DATI IN R

Dopo aver popolato in modo automatizzato la base di dati, è stata effettuata un'analisi attraverso il linguaggio R.

Di seguito si propone l'analisi dei dati ricavati con la seguente interrogazione:

```
SELECT nome, punti
FROM squadra
ORDER BY punti DESC;
```

Il grafico ci mostra la classifica finale del nostro campionato con i relativi punteggi.

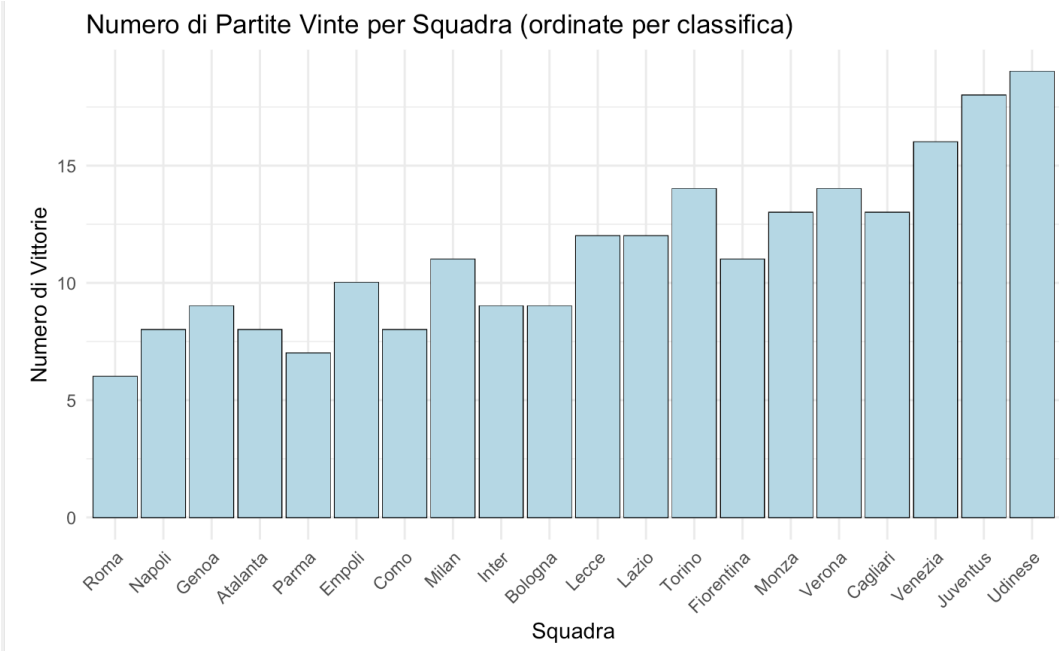


Dai colori delle barre del grafico possiamo distinguere le squadre che hanno ottenuto la partecipazione ad una competizione europea e quelle che retrocederanno nel campionato di Serie B. Tuttavia, è importante notare che, sebbene il grafico fornisca una visione d'insieme efficace, esso non descrive, ad esempio, il percorso e l'evoluzione della classifica durante la stagione, né il contesto delle singole prestazioni nelle varie giornate. Quindi, per rendere l'analisi ancora più ottimale, abbiamo creato dei grafici temporali analizzano l'andamento delle performance di tutte le squadre durante ogni giornata del campionato.

Il prossimo grafico mostra il numero di partite vinte da ciascuna squadra, ordinate in base al punteggio finale in classifica. È evidente una correlazione tra il numero di vittorie ed il posizionamento.

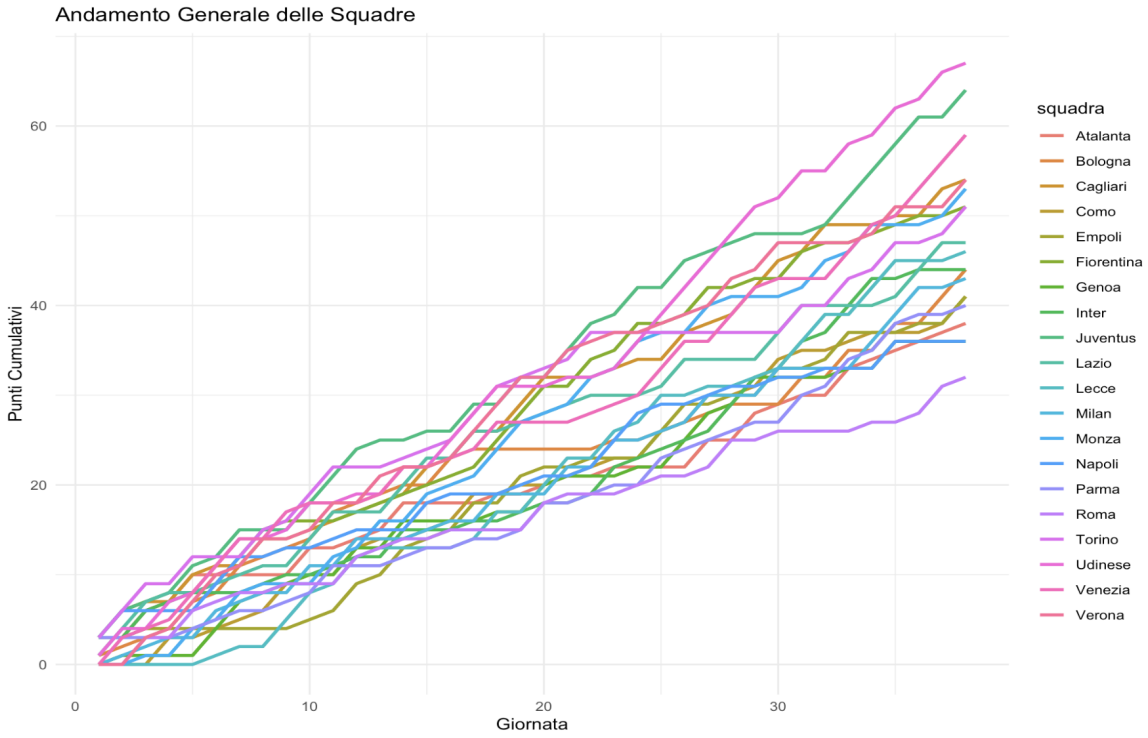
Nonostante questo, non sempre il numero di vittorie è stato deteriminante per ottenere più punti rispetto a squadre che hanno vinto meno.

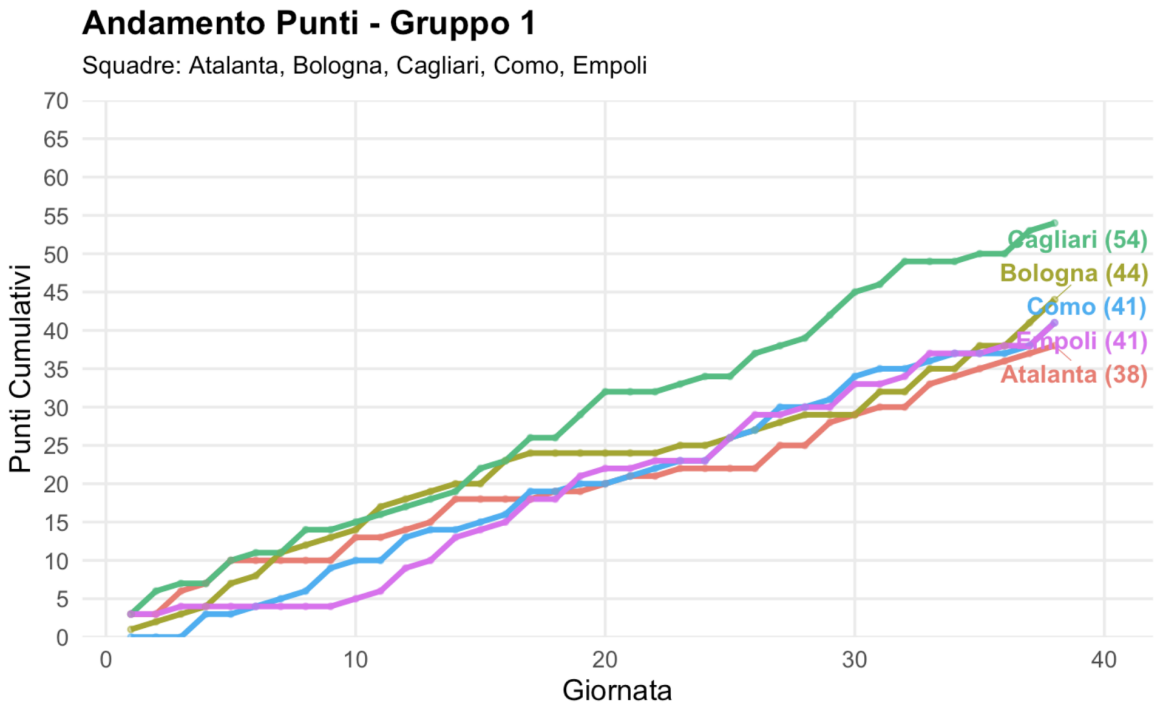
```
SELECT s.nome, COUNT(*) AS vittorie, s.punti
FROM squadra s, esito e
WHERE s.nome = e.squadra AND
      e.punti = 3
GROUP BY s.nome, s.punti
ORDER BY s.punti DESC;
```

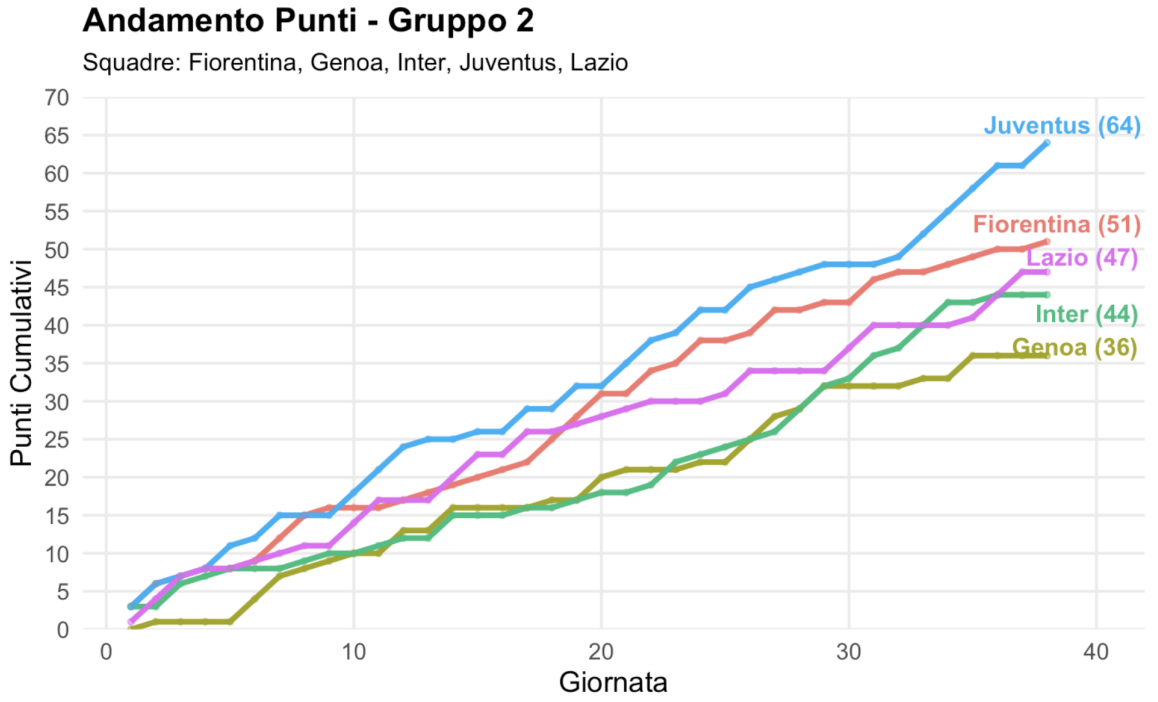


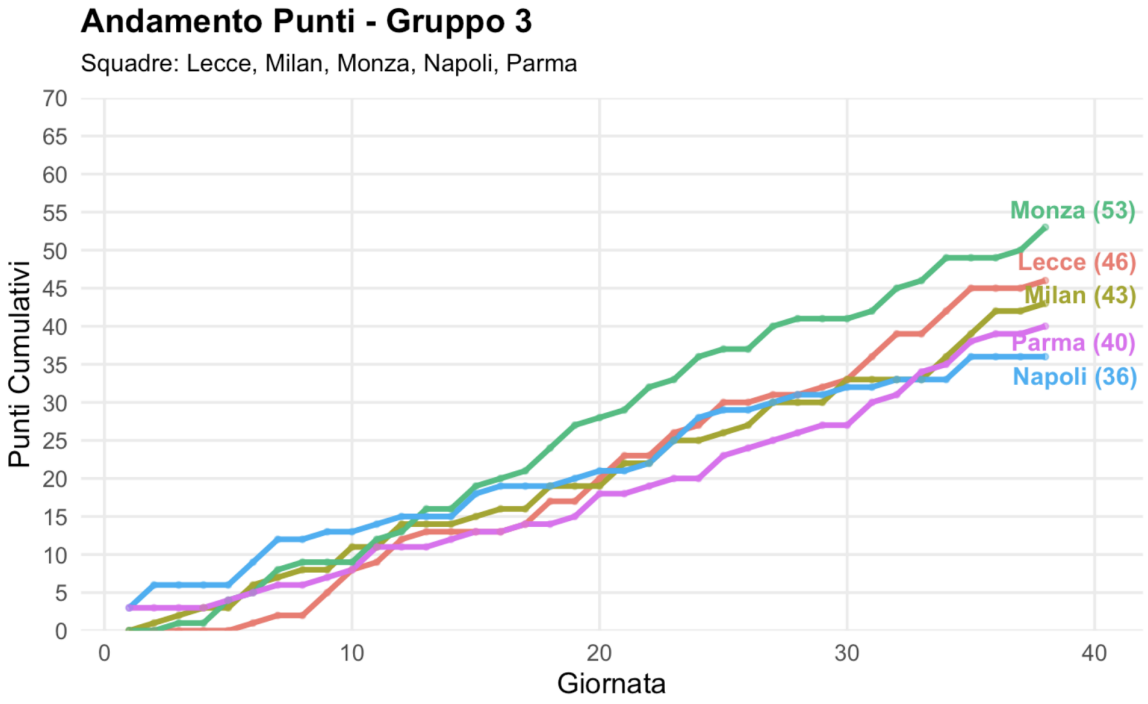
Abbiamo poi approfondito l'analisi esplicitando l'andamento dei punti ottenuti dalle squadre nel corso delle 38 partite.

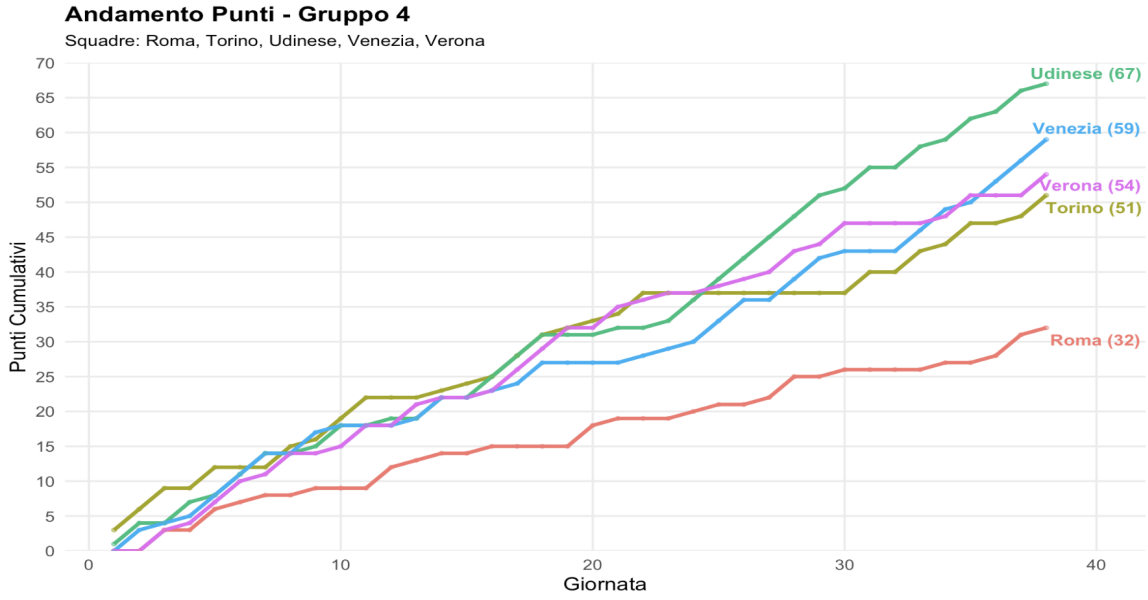
In seguito anche suddividendole in quattro gruppi in base alle loro performance: questo approccio ci ha permesso di evidenziare differenze tra le squadre.




Andamento Punti - Gruppo 1
Squadre: Atalanta, Bologna, Cagliari, Como, Empoli
Punti Cumulativi
Giornata
Cagliari (54)
Bologna (44)
Como (41)
Empoli (41)
Atalanta (38)


Andamento Punti - Gruppo 2
Squadre: Fiorentina, Genoa, Inter, Juventus, Lazio
Punti Cumulativi
Giornata
Juventus (64)
Fiorentina (51)
Lazio (47)
Inter (44)
Genoa (36)


Andamento Punti - Gruppo 3
Squadre: Lecce, Milan, Monza, Napoli, Parma
Punti Cumulativi
Giornata
Monza (53)
Lecce (46)
Milan (43)
Parma (40)
Napoli (36)


Andamento Punti - Gruppo 4
Squadre: Roma, Torino, Udinese, Venezia, Verona
Punti Cumulativi
Giornata
Udinese (67)
Venezia (59)
Verona (54)
Torino (51)
Roma (32)