Progetto Basi Di Dati

Gabriele Pasqualini IN0500861

1 Presentazione del progetto

Si desidera realizzare un database per un sito di scacchi, il cui scopo principale è l'archiviazione e la gestione degli utenti e delle loro partite. L'utilizzo del database è incentrato sulle partite di scacchi, che saranno caratterizzate da vari dettagli tra cui il numero di mosse effettuate, il risultato finale, i giocatori coinvolti e il colore dei pezzi utilizzati da ciascun giocatore.

Ogni partita sarà documentata con diversi dettagli, consentendo ai giocatori di rivedere ogni singola mossa effettuata. Sarà possibile identificare chi ha effettuato ciascuna mossa e monitorare il tempo residuo per ogni giocatore al momento di ciascuna mossa.

Oltre alle mosse, i giocatori potranno visualizzare eventuali commenti relativi a una partita.

Ogni partita sarà associata a una specifica modalità di gioco. Le modalità di gioco saranno definite da parametri come limiti di tempo e eventuali incrementi di tempo per mossa. Il sistema permetterà di visualizzare il punteggio di un giocatore in ciascuna modalità, facilitando la valutazione delle performance individuali in contesti diversi.

Per incentivare la competizione e il miglioramento continuo, il sito offrirà la possibilità di stilare classifiche dei migliori giocatori per ogni modalità di gioco. Le classifiche potranno essere visualizzate sia a livello globale che all'interno delle cerchie di amici.

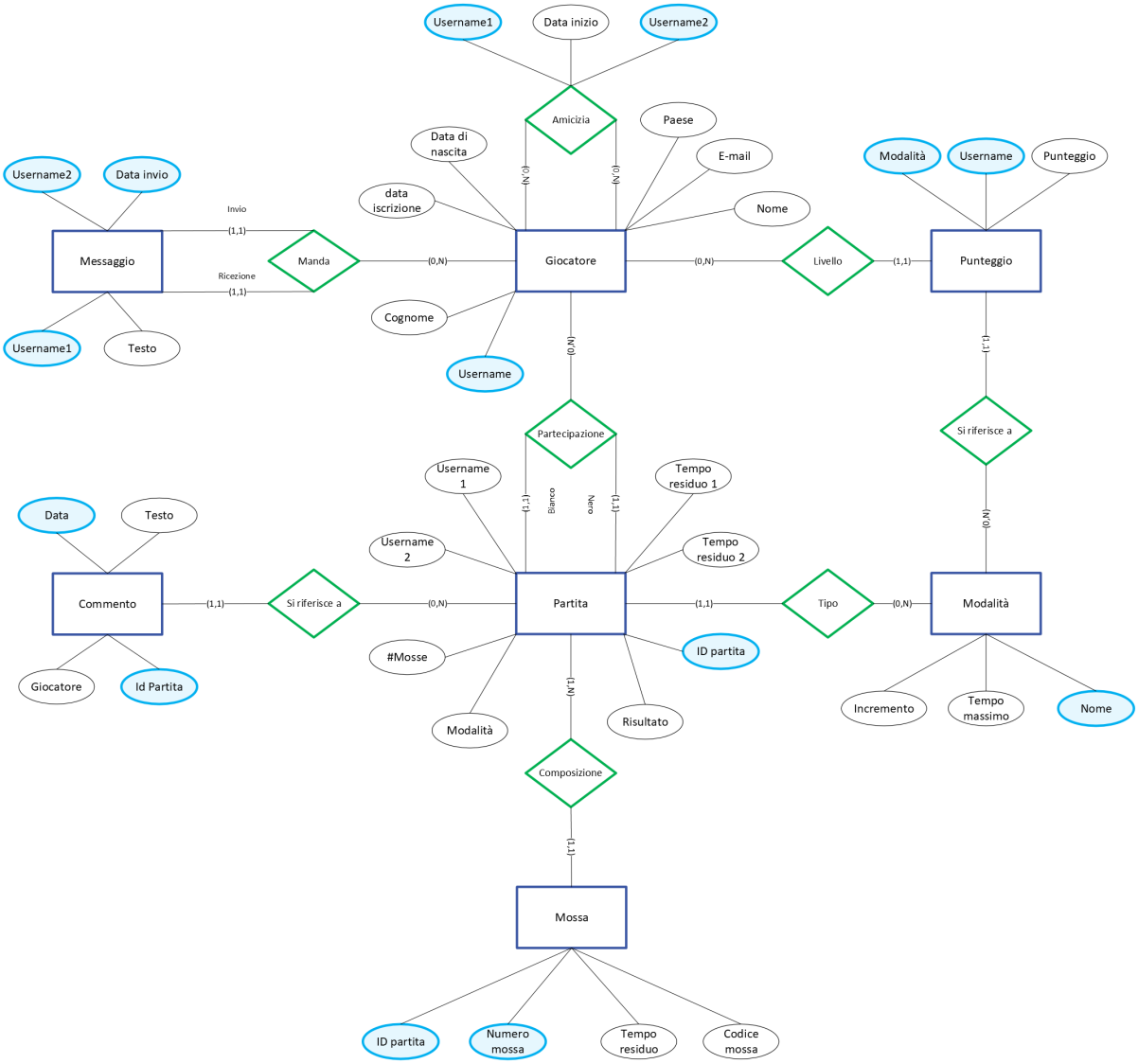
Il database conterrà anche una tabella delle amicizie tra i giocatori, permettendo loro di connettersi e confrontarsi con amici e altri utenti. Ogni giocatore avrà un proprio account che includerà dati anagrafici (nome, cognome, data di nascita, paese di origine), username, email, data di iscrizione ed eventuale numero di telefono.

Essendo il sito anche una piattaforma di scambio di messaggi tra i giocatori, il database dovrà memorizzare le conversazioni tra gli iscritti.

2 Operazioni sul database

- Rivedere le mosse di una determinata partita, ordinandole per indice e visualizzando i tempi residui dopo ogni mossa.
- Visualizzare i commenti di una determinata partita e l'autore degli eventuali commenti.
- Contare quante partite un giocatore ha giocato per ogni modalità, e calcolare il numero medio di partite giornaliere.
- Visualizzare le prime mosse preferite da un giocatore.
- Stilare una classifica globale di N giocatori in base alla modalità di gioco.
- Stilare una classifica degli amici in base alla modalità di gioco.

3 Schema concettuale Entity-Relationship



4 Dizionario delle entità

Entità	Attributo	Descrizione	Chiave Primaria
Giocatore	UserName	Nome utente univoco	UserName
	Nome	Nome del giocatore	
	Cognome	Cognome del giocatore	
	Email	Indirizzo email del giocatore	
	Data di nascita	Data di nascita del giocatore	
	Paese	Paese di residenza del giocatore	
	Data iscrizione	Data di iscrizione al sito	
Partita	ID partita	Identificativo univoco della partita	ID partita
	Risultato	Risultato della partita (1-0, 0-1, $\frac{1}{2}$-$\frac{1}{2}$)	
	Modalità	Modalità di gioco (es. Blitz, Rapid, Classic)	
	UserName1	Nome utente del primo giocatore	
	UserName2	Nome utente del secondo giocatore	
Mosse	ID partita	Identificativo della partita associata	ID partita, Numero della mossa
	Numero della mossa	Ordine della mossa nella partita	
	Codice della mossa	Notazione della mossa effettuata	
	Tempo residuo	Tempo residuo sull'orologio del giocatore	
	Giocatore	Giocatore che ha fatto la mossa	
Punteggio	UserName	Nome utente del giocatore	UserName, Modalità
	Punteggio	Punteggio del giocatore	
	Modalità	Modalità di gioco associata al punteggio	
Modalità	Nome della modalità	Tipo di modalità di gioco	Nome della modalità
	Tempo massimo	Tempo massimo assegnato per la modalità	
	Incremento	Incremento di tempo per mossa	
Messaggi	UserName1	Nome utente del mittente	Data invio, UserName1, UserName2
	UserName2	Nome utente del destinatario	
	Testo	Contenuto del messaggio	
	Data invio	Data di invio del messaggio	
Commento	Data_ora	Data e ora del commento	Data_ora, ID_partita
	ID_partita	Identificativo della partita associata	
	Giocatore	Username del giocatore autore del commento	
	Testo	Contenuto del commento	

5 Dizionario delle relazioni

Nome	Descrizione	Componenti	Attributi
Invio	Messaggio dal giocatore mittente al giocatore destinatario	Giocatore, Messaggio	
Amicizia	Relazione tra un giocatore ed un altro giocatore	Giocatore, Giocatore	Data inizio
Livello	Punteggio di un giocatore in una determinata modalità	Punteggio, Giocatore	
Riferimento	Modalità alla quale si riferisce un punteggio	Punteggio, Modalità	
Tipo	Modalità della partita	Partita, Modalità	
Partecipazione	Giocatori che hanno disputato una partita	Partita, Giocatore	
Appartenenza	Appartenenza di un commento ad una partita	Partita, Commento	
Composizione	Relazione tra le mosse che compongono la partita e la partita	Partita, Mosse	

6 Tabella dei Volumi

Concetto	Tipo	Volume
Giocatore	E	1000
Partita	E	500/giorno
Messaggio	E	100/giorno
Punteggio	E	3000
Modalità	E	10
Mossa	E	8000/giorno
Commento	E	50/giorno

7 Analisi delle Ridondanze

7.1 Attributi derivabili

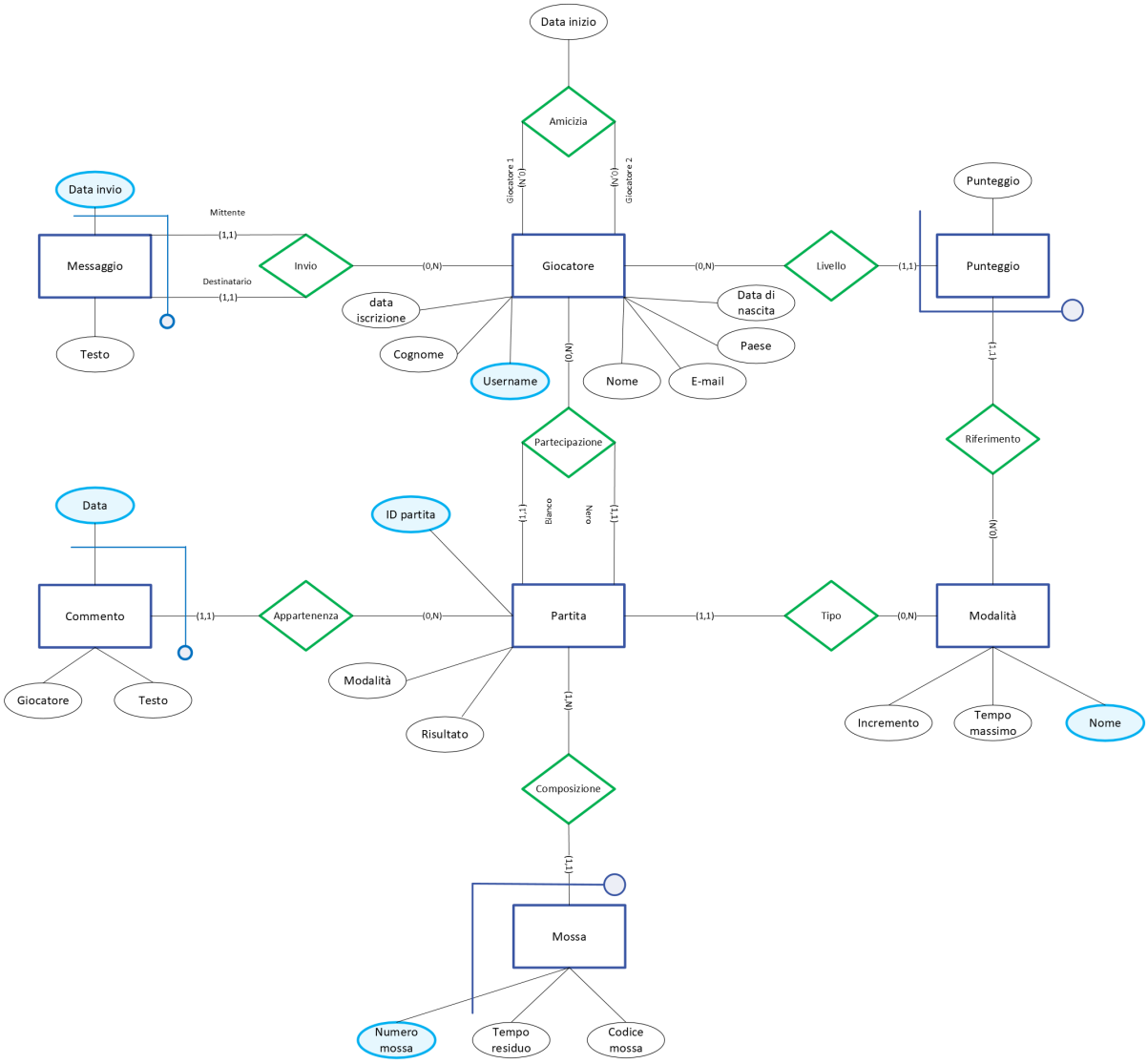
Sono presenti 2 attributi dell'entità Partita che sono derivabili dalla tabella delle mosse:

- L'attributo numero di mosse dell'entità partita è derivabile, basta contare quante sono le mosse nella tabella delle mosse che fanno riferimento a quella determinata partita.
- Anche gli attributi Tempo residuo giocatore 1 e Tempo residuo giocatore 2 sono derivabili sempre dalla tabella delle mosse, andando a controllare l'ultima mossa svolta dal giocatore in quella partita e prendendo il campo del tempo rimanente.
- Inoltre anche l'attributo Giocatore nella tabella delle mosse è derivabile, poiché la mossa 1 sarà del bianco, quindi basterà prendere il giocatore bianco dalla partita in cui è stata giocata la mossa, lo stesso vale per il giocatore nero. Se il numero della mossa è dispari il giocatore sarà il bianco, se è pari il giocatore sarà il nero.

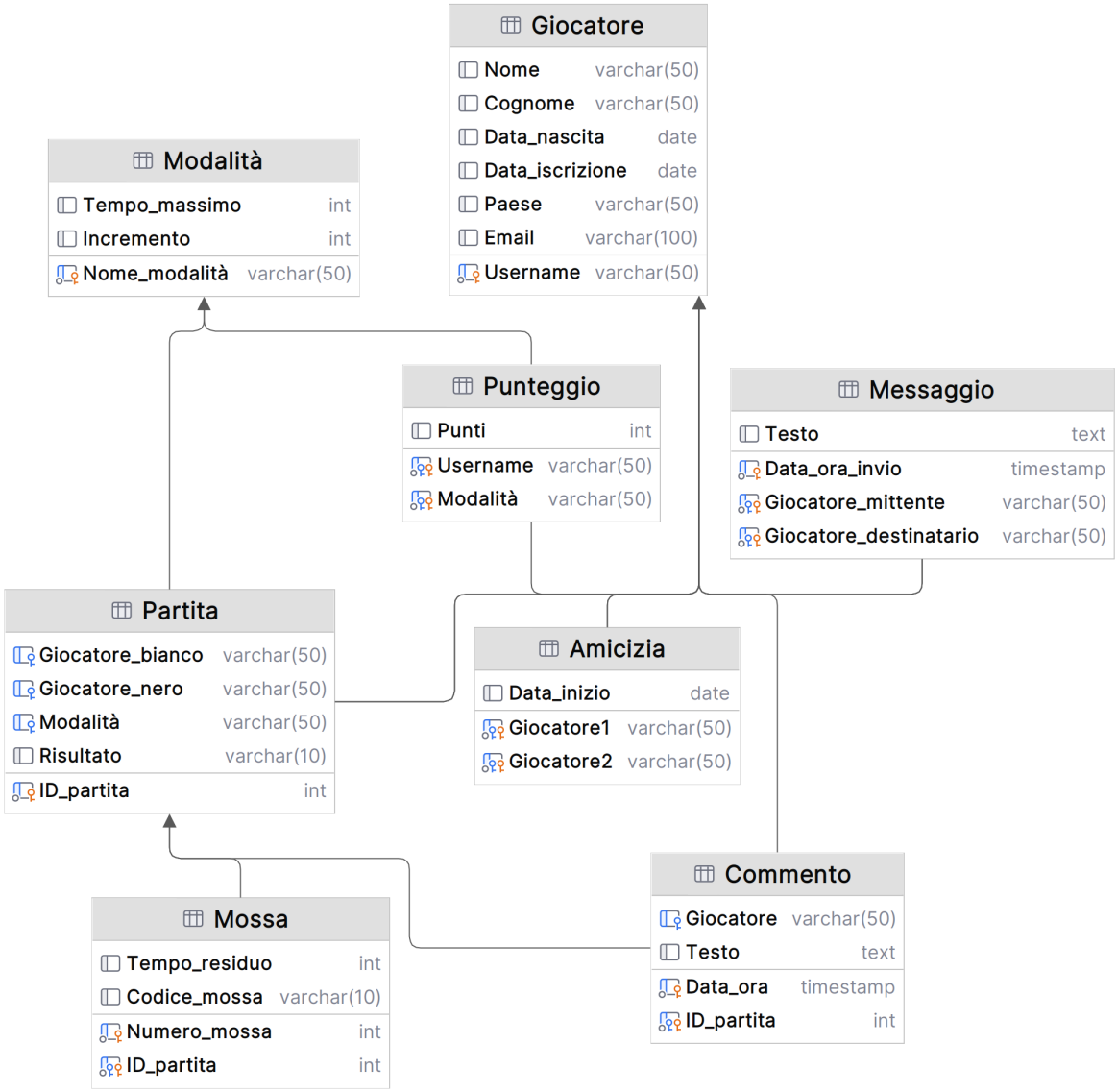
7.2 Analisi dei cicli

Nel database è presente un ciclo che coinvolge le entità Giocatore, Partita, Modalità e Punteggio. Analizzando le varie entità e le relazioni fra esse, essendo presenti solamente relazioni 1 a molti, non è possibile ricavare un entità dalle altre.

8 Schema concettuale Entity-Relationship ristrutturato



9 Schema logico



9.1 Normalizzazione

Il database rispetta le 3 forme normali:

- è in prima forma normale poiché c'è una sola colonna per ogni valore e non ci sono unità ripetute
- è in seconda forma normale poiché ogni attributo non chiave dipende completamente dalla chiave primaria. Non ci sono attributi parzialmente dipendenti dalla chiave primaria.
- è in terza forma normale poiché non sono presenti campi calcolati, e ogni attributo dipende esclusivamente dalla chiave primaria e non da altri attributi

10 Stored Procedures e Triggers

1. Fare una lista dei migliori N giocatori in una modalità:

```
DELIMITER $$
CREATE PROCEDURE TopPlayers(
    IN modalita varchar(50),
    IN topN int
)
BEGIN
    SELECT Nome, Cognome, p.Modalità, p.Punti
        FROM Giocatore g
        INNER JOIN Punteggio p
        USING (Username)
        WHERE p.Modalità = modalita
        ORDER by Punti DESC
        LIMIT topN;

END$$
```

2. Elencare i punteggi di un giocatore:

```
DELIMITER $$
CREATE PROCEDURE ShowPoints(
    IN player varchar(50)
)
BEGIN
    SELECT Modalità, Punti
    FROM Punteggio
    WHERE Username = player;
END$$
DELIMITER ;
```

3. Visualizzare in ordine le mosse di una partita:

```
DELIMITER $$
CREATE PROCEDURE ShowMoves(
    IN game varchar(50)
)
BEGIN
    SELECT Codice_mossa
    FROM Mossa
    WHERE ID_partita = game
    order by Numero_mossa;
END$$
DELIMITER ;
```

4. Visualizzare gli N giocatori con la media dei punteggi più alta:

```
DELIMITER $$
CREATE PROCEDURE BestNPlayerAVG(IN N INT)
  BEGIN
      SELECT Username, Nome, Cognome, AVG(Punti) a
      FROM Giocatore
      INNER JOIN Punteggio
      USING (Username)
      GROUP by Username
      ORDER by a DESC
      LIMIT N;
  END$$
DELIMITER ;
```

5. Se A è amico di B, allora è sarebbe ridondante aggiungere che B è amico di A:

```
CREATE TRIGGER trg_before_insert_amicizia
BEFORE INSERT ON Amicizia
FOR EACH ROW
BEGIN
    IF EXISTS
    (SELECT 1 FROM Amicizia WHERE Giocatore1 = NEW.Giocatore2 AND Giocatore2 = NEW.Giocatore1)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Amicizia già esistente';
    END IF;
END;
```

6. Aggiornamento punteggio giocatori in caso di vittoria o sconfitta

```
DELIMITER $$

CREATE TRIGGER trg_update_punteggio
AFTER INSERT ON Partita
FOR EACH ROW
BEGIN
    IF NEW.Risultato = '1-0' THEN
        -- Giocatore bianco vince
        UPDATE Punteggio
        SET Punti = Punti + 5
        WHERE Username = NEW.Giocatore_bianco AND Modalità = NEW.Modalità;

        UPDATE Punteggio
        SET Punti = Punti - 5
        WHERE Username = NEW.Giocatore_nero AND Modalità = NEW.Modalità;

    ELSEIF NEW.Risultato = '0-1' THEN
        -- Giocatore nero vince
        UPDATE Punteggio
        SET Punti = Punti + 5
        WHERE Username = NEW.Giocatore_nero AND Modalità = NEW.Modalità;

        UPDATE Punteggio
        SET Punti = Punti - 5
        WHERE Username = NEW.Giocatore_bianco AND Modalità = NEW.Modalità;

    END IF;
END $$
DELIMITER ;
```