Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

SISTEMA DI GESTIONE DELLE SCOMMESSE PER UN CAMPIONATO DI CALCIO

0266833

Adrian Petru Baba

Indice

1. Descrizione del Minimondo

Si vuole realizzare un sistema informativo per la gestione di scommesse sportive per le partite di un mondiale di calcio, tenendo conto delle seguenti informazioni.

Le squadre che partecipano al campionato di calcio sono identificate univocamente dal nome della nazione di appartenenza. Per ogni squadra è noto il nome dell'attuale allenatore e l'elenco delle precedenti edizioni dei campionati vinti dalla squadra stessa.

Il campionato è organizzato in turni di gioco. Ciascun turno è identificato univocamente dal nome del turno stesso, (esempio: ``qualificazione", ``quarti", ``semifinale" e ``finale"). Il sistema contiene l'elenco delle squadre che prendono parte a ciascun turno di gioco e, al termine di ciascun turno, calcola automaticamente la rosa delle squadre che prenderà parte al turno successivo, in funzione dei risultati ottenuti alle partite precedenti.

Le partite di calcio sono identificate attraverso un numero d'ordine univoco all'interno di ciascun turno di gioco. Per ogni partita sono noti i nomi delle due squadre coinvolte, dove si gioca l'incontro (Amsterdam, Bruxelles, Rotterdam o Bruges) ed a che ora.

I giocatori sono identificati univocamente dal nome. Per ciascuno è inoltre nota la squadra in cui gioca negli europei, e con quale numero di maglia. Per ogni giocatore è ancora noto un recapito. Per ciascuno dei giocatori che ha segnato, si vuole memorizzare nella base di dati, per ogni partita giocata, il minuto di gioco in cui tale giocatore ha segnato un goal, e se questo è avvenuto su rigore.

Infine la base di dati contiene l'informazione relativa a quale arbitro è stato assegnato a ciascuna partita. Per ogni arbitro è noto il nome, che lo identifica univocamente, un recapito, ed il numero complessivo di presenze agli europei.

Gli addetti allo sportello possono raccogliere le scommesse degli utenti del sistema. Ciascuna scommessa è associata ad un cliente, identificato dal suo codice fiscale e di cui la base di dati mantiene tutta l’anagrafica. La scommessa di un giocatore riguarda una specifica partita ed è associata al vincitore della partita o ad un pareggio.

Ciascuna scommessa è associata ad un importo versato e ad un moltiplicatore di vincita, associato alla partita su cui si sta scommettendo. All'atto della scommessa, al cliente viene fornita una ricevuta riportante le informazioni sulla partita su cui hanno scommesso ed un codice alfanumerico univoco che identifica la loro giocata.

I gestori del servizio di scommesse hanno la possibilità di configurare, tra quelle ancora in gara, le squadre che si affronteranno ad ogni turno. Per ciascuna partita devono individuare un fattore moltiplicativo che associa le scommesse alle vincite.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
3	Campionato	Mondiale	Sono usati come sinonimi, “mondiale” è il termine più adatto ed è già presente alla linea 2.
13	Incontro	Partita	Sono usati come sinonimi, “partita” è il termine che rende più chiaro il significato ed è già presente alla linea 2.
15	Europei	Mondiale	Sono usati come sinonimi in modo errato, “mondiale” è il termine più adatto ed è già presente alla linea 2
16	Recapito	Indirizzo	Termine poco chiaro, un recapito può essere anche telefonico, assumiamo il significato di “indirizzo”
21	Recapito	Indirizzo	Stesso della riga precedente
21	Europei	Mondiale	Sono usati come sinonimi in modo errato, “mondiale” è il termine più adatto ed è già presente alla linea 2
22	Utenti	Clienti	Sono usati come sinonimi, “clienti” è il termine che rende più chiaro il significato ed è già presente alla linea 23
24	Giocatore	Cliente	Sono usati come sinonimi, “cliente” è più adatto ed è già presente alla linea 23, inoltre giocatore è già utilizzato in riferimento ai componenti delle squadre, quindi sono omonimi.
29	Giocata	Scommessa	Sono usati come sinonimi, “scommessa” è il termine più adatto ed è già presente alla linea 27

Specifica disambiguata

Si vuole realizzare un sistema informativo per la gestione di scommesse sportive per le partite di un mondiale di calcio, tenendo conto delle seguenti informazioni.

Le squadre che partecipano al mondiale sono identificate univocamente dal nome della nazione di appartenenza. Per ogni squadra è noto il nome dell'attuale allenatore e l'elenco delle precedenti edizioni dei campionati vinti dalla squadra stessa.

Il mondiale è organizzato in turni di gioco. Ciascun turno è identificato univocamente dal nome del turno stesso, (esempio: ``qualificazione", ``quarti", ``semifinale" e ``finale"). Il sistema contiene l'elenco delle squadre che prendono parte a ciascun turno di gioco e, al termine di ciascun turno, calcola automaticamente la rosa delle squadre che prenderà parte al turno successivo, in funzione dei risultati ottenuti alle partite precedenti.

Le partite di calcio sono identificate attraverso un numero d'ordine univoco all'interno di ciascun turno di gioco. Per ogni partita sono noti i nomi delle due squadre coinvolte, dove si gioca la partita (Amsterdam, Bruxelles, Rotterdam o Bruges) ed a che ora.

I giocatori sono identificati univocamente dal nome. Per ciascuno è inoltre nota la squadra in cui gioca negli europei, e con quale numero di maglia. Per ogni giocatore è ancora noto un indirizzo. Per ciascuno dei giocatori che ha segnato, si vuole memorizzare nella base di dati, per ogni partita giocata, il minuto di gioco in cui tale giocatore ha segnato un goal, e se questo è avvenuto su rigore.

Infine la base di dati contiene l'informazione relativa a quale arbitro è stato assegnato a ciascuna partita. Per ogni arbitro è noto il nome, che lo identifica univocamente, un indirizzo, ed il numero complessivo di presenze agli europei.

Gli addetti allo sportello possono raccogliere le scommesse dei clienti del sistema. Ciascuna scommessa è associata ad un cliente, identificato dal suo codice fiscale e di cui la base di dati mantiene tutta l’anagrafica. La scommessa di un cliente riguarda una specifica partita ed è associata al vincitore della partita o ad un pareggio.
Ciascuna scommessa è associata ad un importo versato e ad un moltiplicatore di vincita, associato alla partita su cui si sta scommettendo. All’atto della scommessa, al cliente viene fornita una ricevuta riportante le informazioni sulla partita su cui hanno scommesso ed un codice alfanumerico univoco che identifica la loro giocata.

I gestori del servizio di scommesse hanno la possibilità di configurare, tra quelle ancora in gara, le squadre che si affronteranno ad ogni turno. Per ciascuna partita devono individuare un fattore moltiplicativo che associa le scommesse alle vincite.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Squadra	L’insieme dei giocatori di una nazione che partecipa al mondiale.	\\	Turno, Giocatore, Partita
Turno	Fase in cui si colloca una determinate partita.	\\	Squadra, Partita
Giocatore	Singolo componente di una squadra.	\\	Squadra, Partita
Partita	Evento in cui due squadre giocano tra di loro.	Incontro	Squadra, Turno, Giocatore
Arbitro	Persona che si assicura che durante una partita le regole vengano rispettate.	\\	Partita
Scommessa	La scommessa che una persona può puntare in riferimento ad una specifica partita.	Scommessa	Cliente, Partita
Cliente	Persona che piazza una scommessa.	Utente, Giocatore	Scommessa

Raggruppamento dei requisiti in insiemi omogenei

Frasi di carattere generale
Si vuole realizzare un sistema informativo per la gestione di scommesse sportive per le partite di un mondiale di calcio, tenendo conto delle seguenti informazioni.

Frasi relative alle squadre
Le squadre che partecipano al mondiale sono identificate univocamente dal nome della nazione di appartenenza. Per ogni squadra è noto il nome dell'attuale allenatore e l'elenco delle precedenti edizioni dei campionati vinti dalla squadra stessa.

Frasi relative ai turni
Ciascun turno è identificato univocamente dal suo nome, (esempio: ``qualificazione", ``quarti", ``semifinale" e ``finale"). Il sistema contiene l'elenco delle squadre che prendono parte a ciascun turno di gioco e, al termine di ciascun turno, calcola automaticamente quali sono le squadre che parteciperanno al turno successivo, in funzione dei risultati ottenuti alle partite precedenti.

Frasi relative ai giocatori
I giocatori sono identificati univocamente dal nome. Per ciascuno è inoltre nota la squadra in cui gioca nei mondiali, e con quale numero di maglia. Per ogni giocatore è ancora noto un indirizzo. Per ciascuno dei giocatori che ha segnato, si vuole memorizzare nella base di dati, per ogni partita giocata, il minuto di gioco in cui tale giocatore ha segnato un goal, e se questo è avvenuto su rigore.

Frasi relative alle partite
Le partite di calcio sono identificate attraverso un numero d'ordine univoco all'interno di ciascun turno di gioco. Per ogni partita sono noti i nomi delle due squadre coinvolte, la città in cui si gioca (per esempio Amsterdam, Bruxelles, Rotterdam o Bruges) ed a che ora.

Frasi relative agli arbitri
La base di dati contiene l'informazione relativa a quale arbitro è stato assegnato a ciascuna partita. Per ogni arbitro è noto il nome, che lo identifica univocamente, un indirizzo, ed il numero complessivo di presenze ai mondiali.

Frasi relative alle scommesse
La scommessa di un cliente riguarda una specifica partita ed è associata al vincitore della partita o ad un pareggio. Ciascuna scommessa è associata ad un importo versato e ad un moltiplicatore di vincita, associato alla partita su cui si sta scommettendo. All'atto della scommessa, al cliente viene fornita una ricevuta riportante le informazioni sulla partita su cui hanno scommesso ed un codice alfanumerico univoco che identifica la loro giocata.

Frasi relative ai clienti
Ciascuna scommessa è associata ad un cliente, identificato dal suo codice fiscale e di cui la base di dati mantiene tutta l'anagrafica.

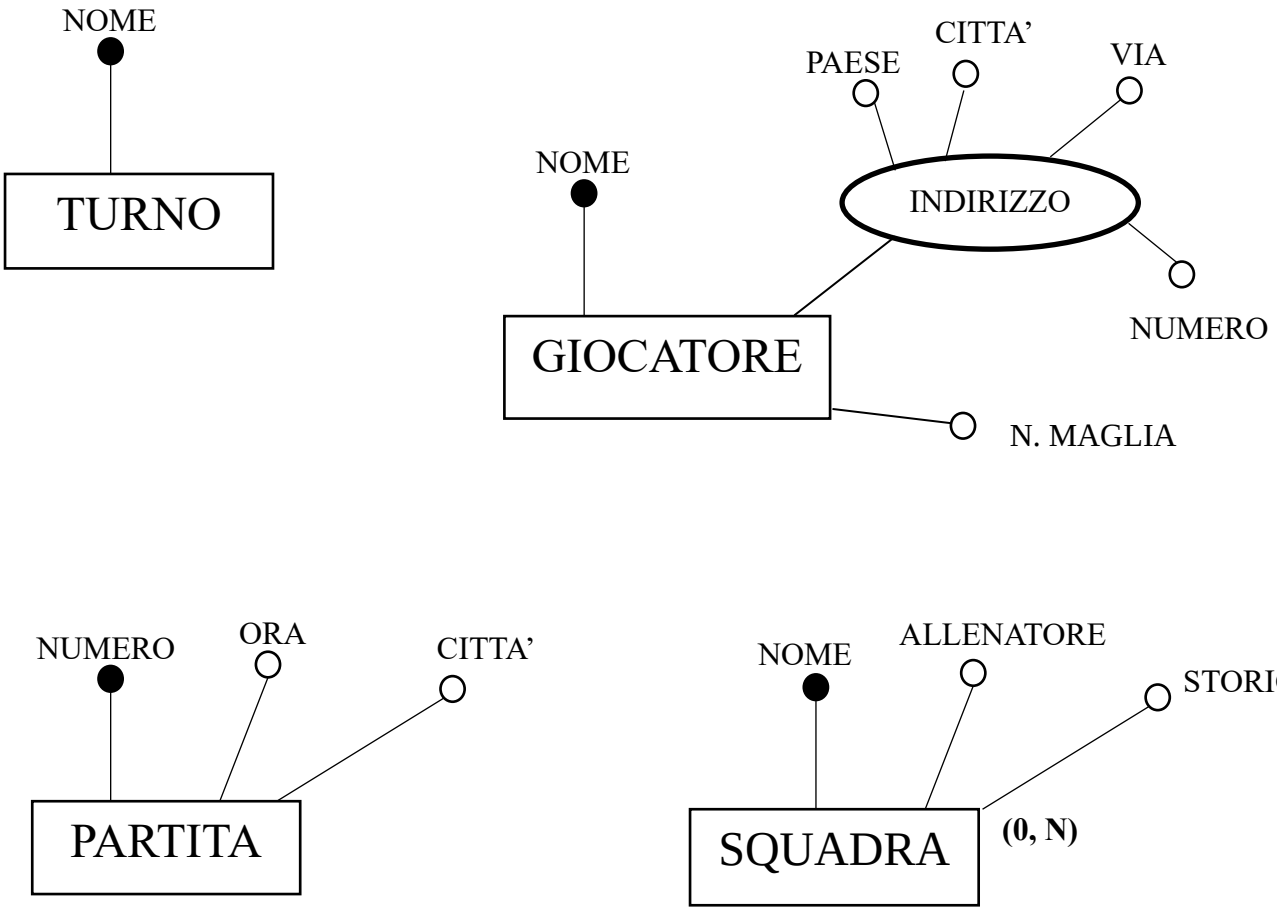
3. Progettazione concettuale

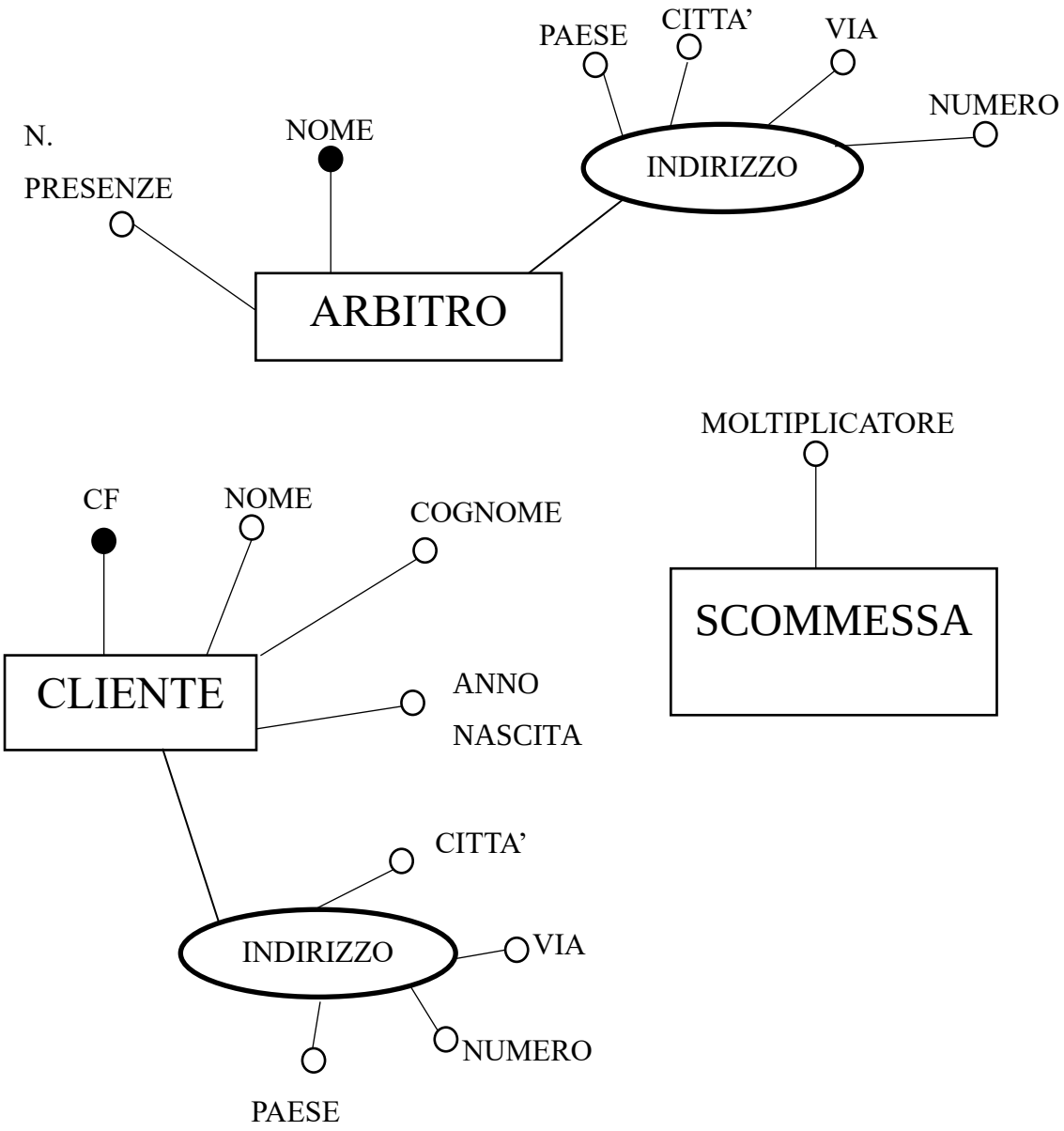
Costruzione dello schema E-R

Il modo in cui il testo del progetto commissionato è impostato, ovvero suddiviso in gruppi di frasi riferite a uno specifico aspetto del sistema, senza particolari incomprensioni e miscugli di concetti, nonché la limitata dimensione del progetto, mi fanno propendere verso una strategia "inside-out" o a macchia d'olio.

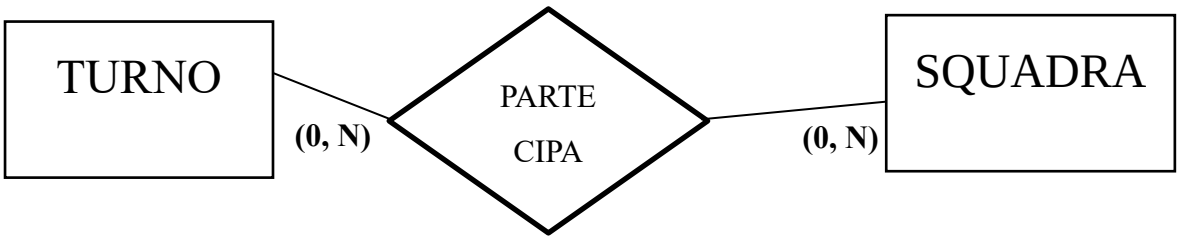
Iniziamo dai singoli elementi fondamentali ricavati da una prima analisi del testo: qui di seguito sono riportati questi elementi che con grande probabilità rimarranno uguali nello schema completato.

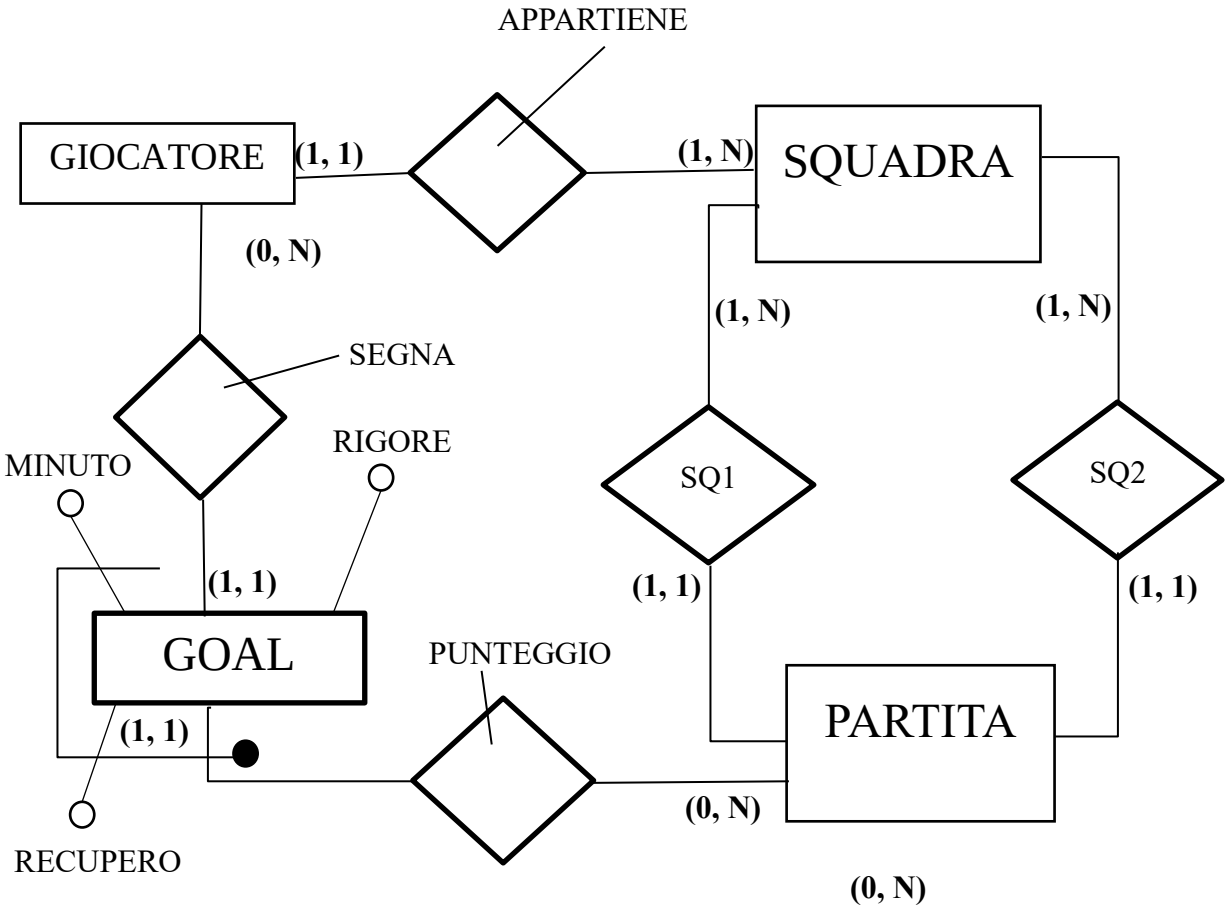
In particolare, i seguenti elementi e i relativi attributi derivano direttamente dagli elementi "fondamentali" identificati e descritti nel paragrafo precedente.

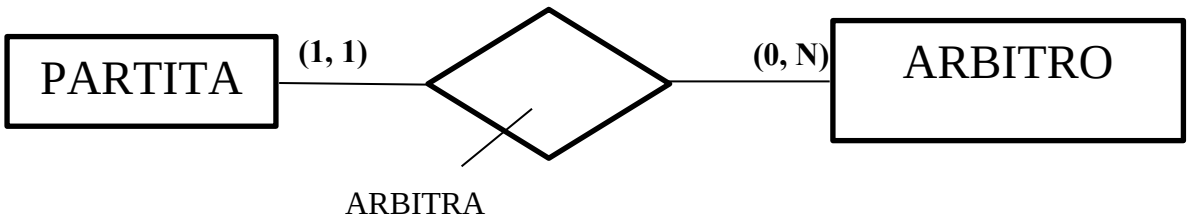


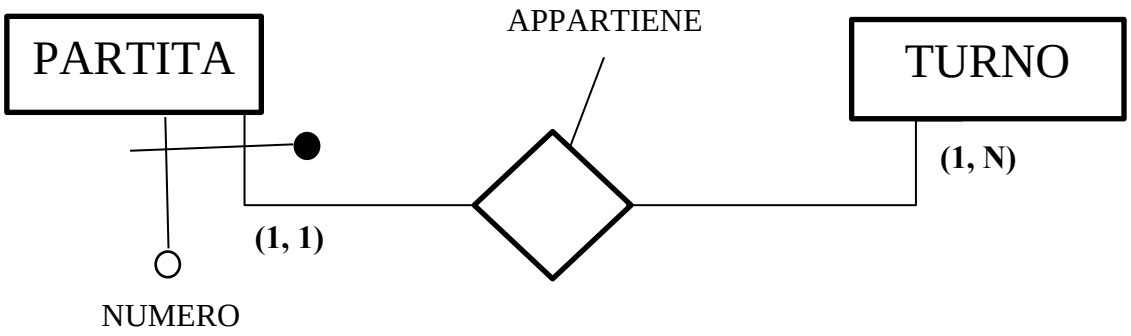


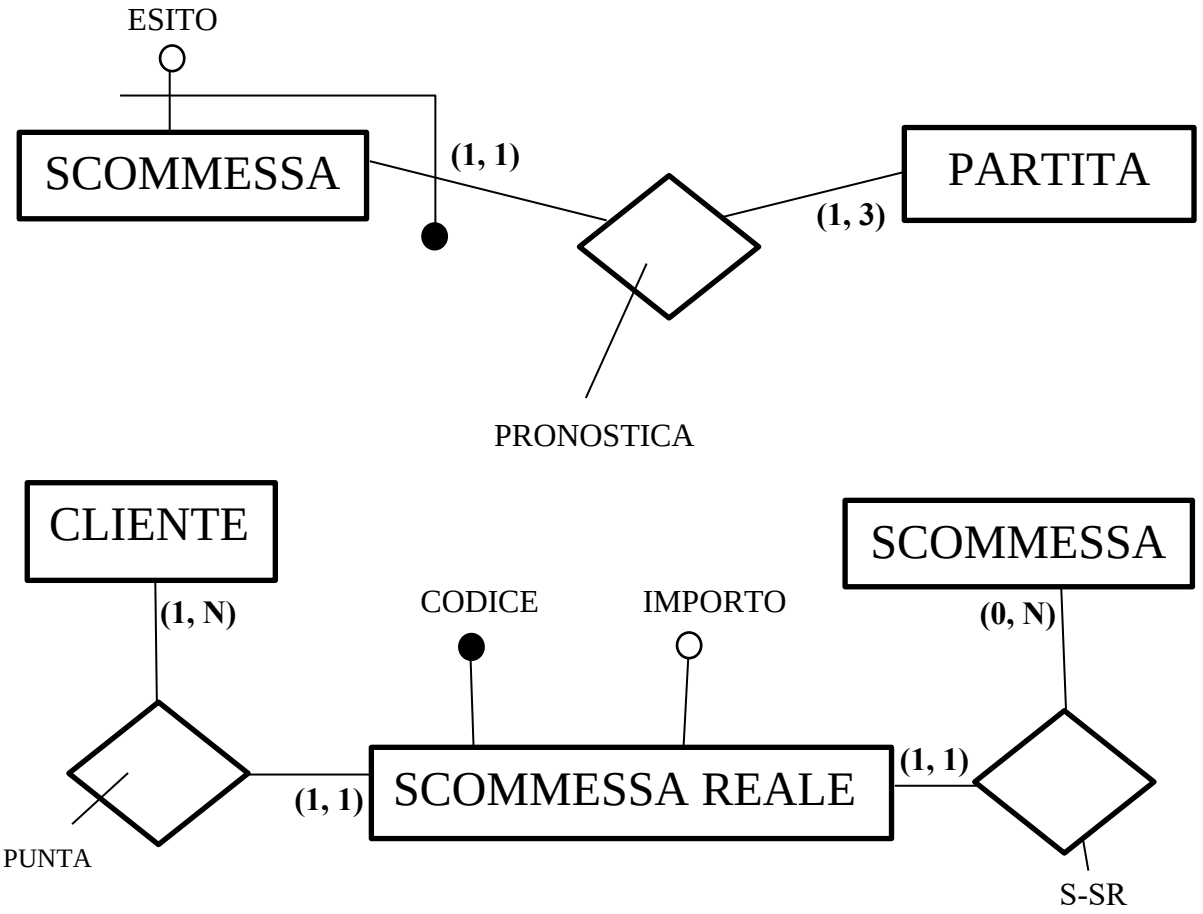
Procediamo raffinando i vari pezzi attenendoci alle indicazioni del committente. In questa fase iniziamo a delineare quali sono le relazioni fra le entità, omettendo per semplicità gli attributi (tranne quando fanno parte del processo di raffinamento).




APPARTIENE
GIOCATORE
(1, 1)
(1, N)
SQUADRA
(0, N)
(1, N)
(1, N)
SEGNA
MINUTO
RIGORE
SQ1
SQ2
(1, 1)
(1, 1)
(1, 1)
GOAL
PUNTEGGIO
PARTITA
(1, 1)
(0, N)
RECUPERO
(0, N)


PARTITA
(1, 1)
(0, N)
ARBITRO
ARBITRA

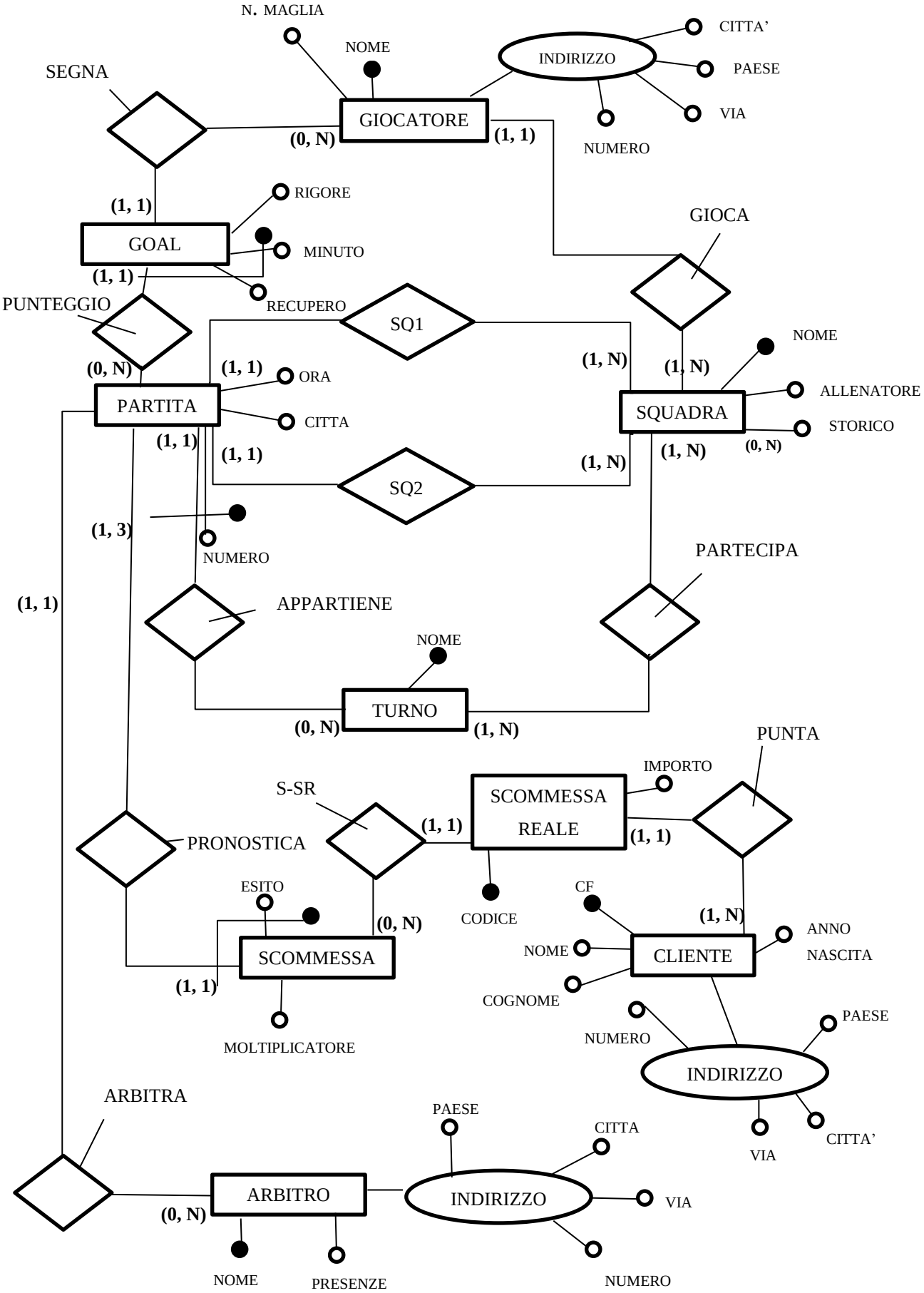

PARTITA
APPARTIENE
TURNO
(1, N)
(1, 1)
NUMERO



Integrazione finale

Integrando il diagramma e/r, l'unico conflitto venuto fuori riguarda il nome di due associazioni: tra giocatore e squadra e poi tra partita e turno, il nome delle associazioni risultava lo stesso. Per evitare problemi legati all'omonimia, l' associazione tra giocatore e squadra è stata rinominata "gioca".

Non si sono verificati conflitti strutturali di alcun tipo.


N. MAGLIA
NOME
INDIRIZZO
CITTA'
PAESE
VIA
NUMERO
SEGNA
GIOCATORE
(0, N)
(1, 1)
(1, 1)
RIGORE
GOAL
MINUTO
RECUPERO
(1, 1)
(1, 1)
GIOCA
PUNTEGGIO
SQ1
(0, N)
(1, 1)
ORA
PARTITA
CITTA
(1, N)
(1, N)
NOME
SQUADRA
ALLENATORE
STORICO
(0, N)
(1, 1)
(1, 1)
SQ2
(1, N)
(1, N)
(1, 3)
NUMERO
PARTECIPA
(1, 1)
APPARTIENE
NOME
TURNO
(0, N)
(1, N)
PUNTA
IMPORTO
S-SR
SCOMMESSA
REALE
(1, 1)
(1, 1)
PRONOSTICA
ESITO
CODICE
CF
(1, N)
ANNO
NASCITA
(0, N)
SCOMMESSA
NOME
CLIENTE
(1, 1)
COGNOME
MOLTIPLICATORE
NUMERO
PAESE
INDIRIZZO
VIA
CITTA'
ARBITRA
PAESE
CITTA
ARBITRO
INDIRIZZO
VIA
(0, N)
NOME
PRESENZE
NUMERO

Regole aziendali

1) Un giocatore può figurare tra i marcatori di una partita solo se appartiene ad una delle due squadre che si confrontano
2) La base di dati in analisi è struturata seguendo il funzionamento che ci si aspetta nelle fasi ad eliminazione diretta di ogni torneo, dunque: al turno "finale" devono partecipare solo ed esclusivamente 2 squadre, al turno "semifinale" solo ed esclusivamente 4 squadre e così via;
3) Il minuto in cui un goal è stato segnato deve essere compreso tra "1" e "130"
4) Il numero di maglia di un giocatore deve essere tra 0 e 99
5) Due giocatori della stessa squadra non possono avere lo stesso numero di maglia
6) Superata la fase di qualificazione, l'esito "pareggio" non è più possibile
7) Superato il turno "qualificazione" una squadra non può partecipare a più partite diverse
8) L'attributo "Esito" della scommessa può valere 0,1,2 dove 0 è il pareggio, 1 la vittoria della squadra 1 e 2 la vittoria della squadra 2
9) L'importo di una scommessa non può essere un valore negativo
10) Il numero civico degli indirizzi non può assumere un valore negativo
11) L'anno di nascita di un cliente deve essere compreso tra 1900 e 2003
12) Una scommessa di un cliente può essere puntata solo se la partita non è stata ancora giocata
13) Il valore degli anni delle edizioni nello storico delle squadre deve essere un numero maggiore di 1930 e con un salto di valore 4 tra un valore e l'altro (1930=OK, 1934=OK, 1932=NO)
14) Il valore degli anni delle edizioni nello storico delle squadre non può valere 1942 o 1946
15) Il valore del nome del Turno può essere solo uno tra "qualificazione", "ottavi", "quarti", "semifinale" oppure "finale"
16) Ad una partita di un determinato turno possono affrontarsi solo squadre che effettivamente partecipano a quel turno
17) L'attributo min_recupero di partita può essere solo =0 oppure >0 quando il minuto di partita vale 45, 90, 105 o 120
18) I minuti di partita >120 sono riservati ai goal segnati nella fase dei rigori
19) Il numero di presenze di un arbitro non può essere < 0
20) Il valore del moltiplicatore di una scommessa non può essere < 1
21) Non si può inserire un goal per una partita non ancora giocata
22) Ad una partita non può partecipare la stessa squadra sia come prima che come seconda squadra

23) Le partite devono essere correttamente ordinate (per esempio la finale di una squadra non si può giocare prima della semifinale)
24) Duarante il turno “qualificazione” non ci sono I tempi supplementari: il minuto del goal non può essere >90

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Turno	Fase specifica del mondiale	Nome	Nome
Giocatore	Componente di una squadra	Nome, N. Maglia, Paese, Città. Via, Numero	Nome
Partita	Sfida tra due squadre	Numero, Ora, Città	Numero, Turno
Squadra	Squadra che partecipa al mondiale	Nome, Allenatore, Storico	Nome
Arbitro	Chi fa rispettare le regole durante una partita	Nome, N. Presenze, Paese, Città, Via, Numero	Nome
Scommessa	Possibile scommessa e i suoi dati	Esito, Moltiplicatore	Esito, Partita
Scommessa Reale	Scommessa effettiva da parte di un cliente	Codice, Importo	Codice
Cliente	Colui che scommette su una partita	CF, Nome, Cognome, Paese, Città, Via, Numero, Anno Nascita	CF
Goal	Rete segnata durante una partita	Minuto, Rigore, Recupero	Minuto, Partita, Recupero

Associazione	Descrizione	Componenti	Attributi
Gioca	Appartenenza ad una squadra	Squadra, Giocatore	\\
Segna	Goal segnato da un giocatore	Goal, Giocatore	\\
Punteggio	Goal segnati durante una partita	Partita, Goal	\\
Sq1	Squadra 1 di una partita	Partita, Squadra	\\
Sq2	Squadra 2 di una partita	Partita, Squadra	\\
Partecipa	Squadra ammessa ad un determinato turno	Squadra, Turno	\\
Appartiene	Partita che fa parte di un determinato turno	Partita, Turno	\\
Pronostica	Previsioni su un determinato esito di una partita	Partita, Scommessa	\\
S-SR	Scommessa effettiva	Scommessa, Scommessa reale	\\
Punta	Puntata effettuata da un cliente	Cliente, Scommessa reale	\\
Arbitra	Arbitraggio di una specifica partita	Arbitro, Partita	\\

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
Turno	E	5
Giocatore	E	580
Partita	E	64
Squadra	E	32
Arbitro	E	30
Scommessa	E	176
Scommessa Reale	E	50000
Cliente	E	37500
Goal	E	200
Gioca	R	580
Segna	R	75
Punteggio	R	175
Sq1	R	64
Sq2	R	64
Partecipa	R	62
Appartiene	R	64
Pronostica	R	176
S-SR	R	50000
Punta	R	50000
Arbitra	R	64

[1] Indicare con E le entità, con R le relazioni

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
001	Inserisci un nuovo cliente	1250/giorno
002	Inserisci la scommessa di un cliente	1666/giorno
003	Inserisci una partita e imposta i dati delle possibili scommesse che la riguardano	2/giorno
004	Inserisci i goal segnati durante le partite	5/giorno
005	Modifica dati cliente	50/giorno
006	Trova numero di presenze di un certo arbitro	5/giorno
007	Aggiungi giocatore	3/giorno
008	Trova i dati del capocannoniere del mondiale	5/giorno
009	Trova i dati di una partita e del cliente che ha scommesso più soldi su uno qualsiasi degli esiti possibili	10/giorno
010	Trova le partite che non si sono ancora disputate	48/giorno
011	Trova la scommessa con moltiplicatore maggiore	150/giorno
012	Trova la partita su cui sono stati effettuati il maggior numero di scommesse	50/giorno
013	Trova i dati della partita in cui sono stati segnati più goal	5/giorno
014	Trova i dati di tutte le squadre che partecipano ad un determinato turno	5/giorno

Costo delle operazioni

001: Scrittura in Cliente, 1*2*1250=2500 accessi/giorno

002: Scrittura in Scommessa Reale: 1*2*1666=3333 accessi/giorno

003: Scrittura in Partita, 3 scritture in Pronostica, 3 scritture in Scommessa:
(1*2*2) + (3*2*2)*2 =28 accessi/giorno

004: Scrittura in Goal: 1*2*5=10 accessi/giorno

005: Lettura in Cliente: Scrittura in Cliente: (1+2)*50=100 accessi/giorno

006: Lettura in Arbitro: 1*5=5 accessi/giorno

007: Scrittura in Giocatore: 1*2*3=6 accessi/giorno

008: Letture in Goal, lettura in Segna e Giocatore: (200+1+1)*5=1010 accessi/giorno

009: Lettura in Partita, 3 Letture in Pronostica e Scommessa, diverse letture in S-SR e Scommessa Reale (852, ovvero 50000/176*3), lettura in Punta e Cliente:
(1+3+3+(852*2)+1+1)*10=17130 accessi/giorno

010: Letture in Partita: 1*64*48=3072 accessi/giorno

011: Letture in Scommessa: 1*176*150= 26400 accessi/giorno

012: Letture in Scommessa Reale, S-SR, Scommessa, Pronostica e Partita: (50000+5000+176+1+1

013: Letture in Goal, lettura in punteggio e partita: (200+1+1)*1*5=1010 accessi/giorno

014: Lettura in Turno, letture in Partecipa e Squadra (6, ovvero media aritmetica di squadre per turno, 32/6): (1+6+6)*1*5=65 accessi/giorno

Ristrutturazione dello schema E-R

Nel nostro schema E/R sono presenti due ridondanze: l'attributo "numero presenze" dell'entità arbitri, che è calcolabile contando il numero di partite in cui gli arbitri hanno lavorato, e l'associazione "partecipa" tra le squadre ed I turni del torneo.

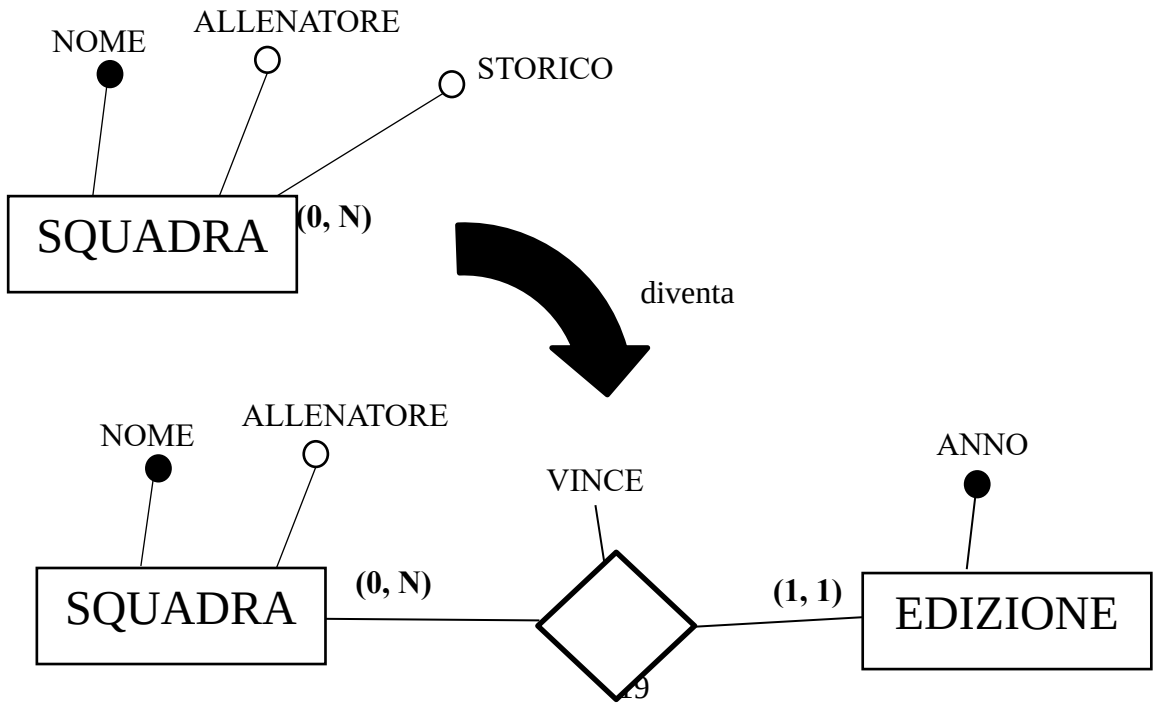
Per l'attributo e in riferimento all'operazione 006, possiamo osservare che mantenendo l'attributo l'operazione avrebbe un costo pari a 5, rimuovendolo il costo slitterebbe a (1 + 64) * 5 = 325. Supponendo il peso in memoria di 4 byte per questo dato, pagando un prezzo irrisorio di 4*30 = 120byte possiamo tenerci questo attributo. In questo caso specifico, siccome ambedue le scelte prevedono numeri piuttosto bassi, preferiamo mantenere il dato in memoria come proposto fino ad ora.

Ragionamento analogo si può fare per l'associazione "partecipa": in riferimento all'operazione 014, sapendo che il numero di turni e squadre (nonché la frequenza dell'operazione stessa) sono tutti valori molto contenuti, preferiamo mantenere l'associazione pur pagando un piccolo prezzo in memoria.

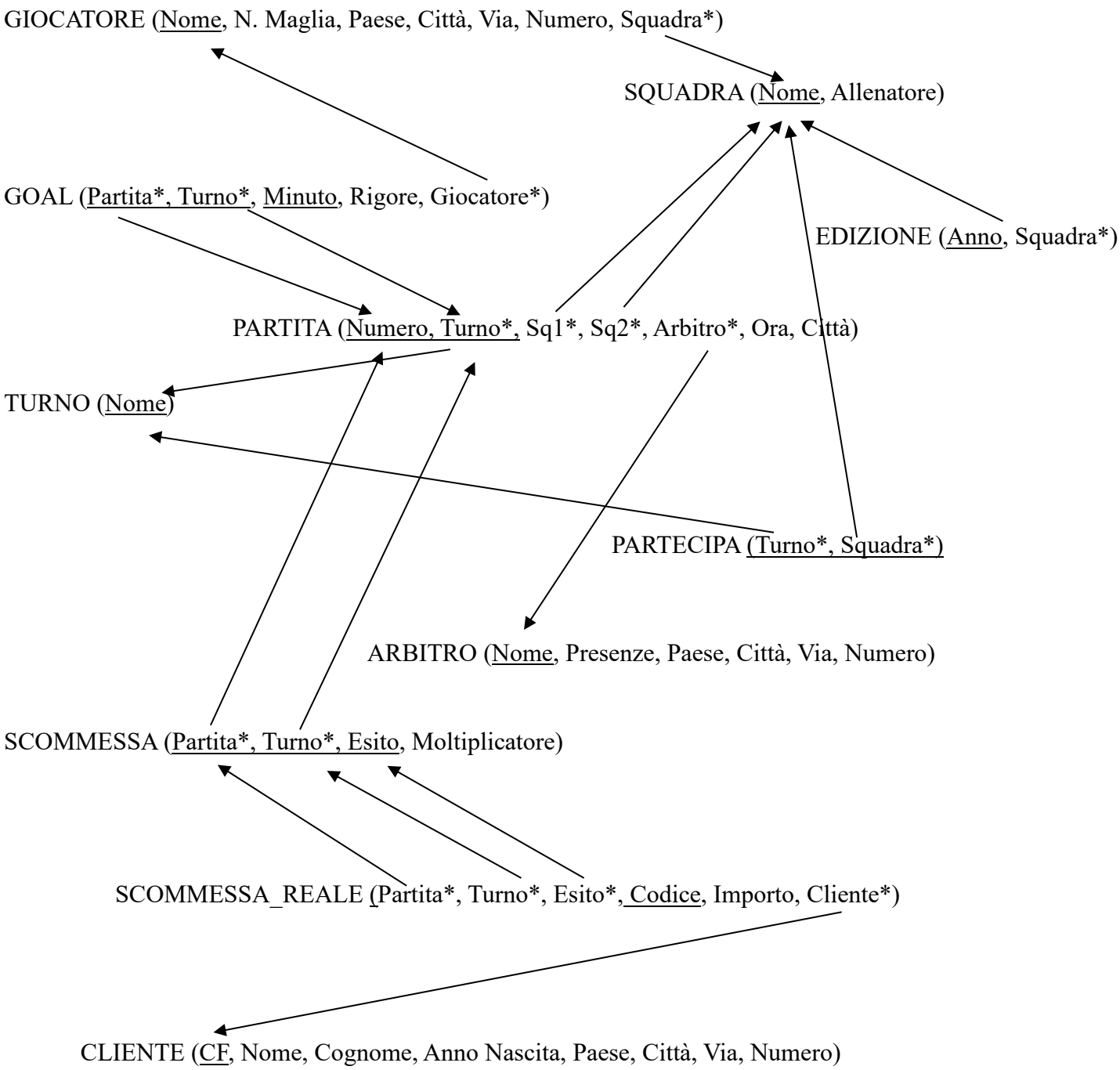
Generalizzazioni non sono presenti nello schema in esame.

Trasformazione di attributi e identificatori

Unica trasformazione da evidenziare è quella che riguarda l'attributo multivalore "Storico" della Squadra: siccome nel modello relazionale non esistono attributi multivalore, trasformiamo le edizioni dei mondiali in un' entità separata associata alla squadra come segue:



Traduzione di entità e associazioni



Normalizzazione del modello relazionale

Il nostro modello realzionale è già in 1NF: ogni attributo contiene un unico valore tra i possibili valori di un dominio con valori atomici.

Il modello è anche già in 2NF: tutti gli attributi non chiave dipendono funzionalmente dall' intera chiave e non solo da una parte di essa.

Il modello è anche in 3NF: non c'è alcun attributo non chiave che dipende funzionalmente da altri attributi che non siano quelli che compongono la chiave primaria.

5. Progettazione fisica

Utenti e privilegi

Nell'applicazioni sono stati previsti 3 utenti distinti: amministratore, addetto_sportelli, login.
A ciascun utente è associato un ruolo (omonimo all'utente stesso) e ciascun ruolo non ha accesso a nulla al di fuori delle procedure (stored procedures, vedere paragrafi successivi), e su queste procedure ciascun utente e quindi ruolo ha solo il privilegio di esecuzione.
L' utente login ha come unico scopo quello di creare una bolla di sicurezza intorno al db da accessi indesiderati potenzialmente causati da bug a livello più alto, per esempio nel client in esecuzione.
Gli altri due utenti sono quelli rilevanti per il minimondo; di seguito l'elenco:

a) login : login_utente

b) addetto_sportelli :

- aggiungi_cliente
- modifica_cliente
- aggiungi_scommessa_cliente
- trova_moltiplicatore_maggiore
- trova_partite_future
- trova_partita_piu_scommessa
- trova_sq_per_turno
- trova_capocannoniere
- trova_cliente_ricco
- trova_piu_goal
- trova_presenze_arbitro

c) amministratore:

- aggiorna_tornero
- aggiungi_arbitro
- aggiungi_giocatore
- aggiungi_goal
- aggiungi_partita_e_scommesse
- aggiungi_squadra
- aggiungi_partecipazione
- aggiungi_utente

Strutture di memorizzazione

Tabella giocatore		
Attributo	**Tipo di dato**	**Attributi**[2]
nome	Varchar(45)	PK, NN
n_maglia	Int	\\
paese	Varchar(30)	\\
citta	Varchar(45)	\\
via	Varchar(45)	\\
numero	Int	\\
squadra	Varchar(45)	FK

Tabella goal		
Attributo	**Tipo di dato**	**Attributi**
partita	Int	PK, NN
minuto	Int	PK, NN
turno	Varchar(45)	PK, NN
rigore	Tinyint (=boolean)	\\
giocatore	Varchar(45)	FK
min_recupero	int	PK, NN

Tabella squadra		
Attributo	**Tipo di dato**	**Attributi**
nome	Varchar(45)	PK, NN
allenatore	Varchar(45)	\\

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella edizione		
Attributo	**Tipo di dato**	**Attributi**
anno	Int	PK, NN
squadra	Varchar(45)	FK

Tabella partita		
Attributo	**Tipo di dato**	**Attributi**
numero	Int	PK, NN
turno	Varchar(15)	PK, NN
sq1	Varchar(45)	FK
sq2	Varchar(45)	FK
arbitro	Varchar(45)	FK
citta	Varchar(45)	\\
ora	Timestamp	\\

Tabella arbitro		
Attributo	**Tipo di dato**	**Attributi**
nome	Varchar(45)	PK, NN
presenze	Int	\\
paese	Varchar(30)	\\
citta	Varchar(45)	\\
via	Varchar(45)	\\
numero	Int	\\

Tabella turno		
Attributo	**Tipo di dato**	**Attributi**
nome	Varchar(15)	PK, NN

Tabella partecipa		
Attributo	**Tipo di dato**	**Attributi**
turno	Varchar(15)	PK, NN, FK
squadra	Varchar(45)	PK, NN, FK

Tabella scommessa		
Attributo	**Tipo di dato**	**Attributi**
partita	Int	PK, NN, FK
turno	Varchar(15)	PK, NN, FK
esito	Int	PK, NN
moltiplicatore	Double	NN

Tabella scommessa_reale		
Attributo	**Tipo di dato**	**Attributi**
codice	Varchar(15)	PK, NN
partita	Int	FK, NN
turno	Varchar(15)	FK, NN
esito	Int	FK, NN
importo	Double	NN
cliente	Varchar(16)	FK, NN

Tabella cliente		
Attributo	**Tipo di dato**	**Attributi**
cf	Varchar(16)	PK, NN
nome	Varchar(45)	\\
cognome	Varchar(45)	\\
anno_nascita	Int	\\
paese	Varchar(30)	\\
citta	Varchar(45)	\\
via	Varchar(45)	\\
numero	Int	\\

Tabella utenti		
Attributo	**Tipo di dato**	**Attributi**
username	Varchar(45)	PK, NN
nome	Varchar(45)	\\
cognome	Varchar(45)	\\
password	Varchar(45)	NN
ruolo	Enum(‘amministratore’, ‘addetto_sportelli’)	NN

Indici

Gli unici indici creati sono quelli standard sulle chiavi primarie, sulle chiavi esterne ed eventualmente su singoli attributi di chiavi primarie composte da più di un attributo che erano riferite da altri atribbuti di altre tabelle dello schema.

E' stata valutata la possibilità di inserire ulteriori indici, ma la conclusione è stata che non sarebbe stato necessario, anzi controproducente: una parte del sistema ha dimensioni limitate, in base alle previsioni o per propria natura (per esempio le partite ed i turni) ed inserire indici avrebbe pesato sulle prestazioni di inserimenti e aggiornamenti senza creare un reale vantaggio.

Un'indice su eventuali campi riguardanti le scommesse degli utenti, che rappresenta anche la tabella per cui è previsto il volume di dati maggiori, avrebbe significato comunque un degrado delle prestazioni, essendo una tabella con inserimenti e aggiornamenti frequenti e costanti.

Tabella arbitro	
Indice <nome>	**Tipo[3]:**
PRIMARY	PR

Tabella cliente	
Indice <nome>	**Tipo:**
PRIMARY	PR

Tabella edizione	
Indice <nome>	**Tipo:**
PRIMARY	PR
edizione.squadra_idx	IDX

[3] IDX = index, UQ = unique, FT = full text, PR = primary.

Tabella giocatore	
Indice <nome>	**Tipo:**
PRIMARY	PR
giocatore.squadra_idx	IDX

Tabella goal	
Indice <nome>	**Tipo:**
PRIMARY	PR
goal.giocatore_idx	IDX
goal.partita_idx	IDX
goal.turno_idx	IDX

Tabella partecipa	
Indice <nome>	**Tipo:**
PRIMARY	PR
partecipa.squadra_idx	IDX

Tabella partita	
Indice <nome>	**Tipo:**
PRIMARY	PR
partita.arbitro_idx	IDX
partita.numero_idx	IDX
partita.sq1_idx	IDX
partita.sq2_idx	IDX

Tabella scommessa	
Indice <nome>	**Tipo:**
PRIMARY	PR
scommessa.turno_idx	IDX
scommessa.esito_idx	IDX

Tabella scommessa_reale	
Indice <nome>	**Tipo:**
PRIMARY	PR
scommessa_reale.turno_idx	IDX
scommessa_reale.esito_idx	IDX
scommessa_reale.cliente_idx	IDX
scommessa_reale.partita_idx	IDX

Tabella squadra	
Indice <nome>	**Tipo:**
PRIMARY	PR

Tabella turno	
Indice <nome>	**Tipo:**
PRIMARY	PR

Tabella scommessa	
Indice <nome>	**Tipo:**
PRIMARY	PR
scommessa.turno_idx	IDX
scommessa.esito_idx	IDX

Tabella utente	
Indice <nome>	**Tipo:**
PRIMARY	PR

Trigger

Gli unici trigger utilizzati in questo progetto riguardano l'implementazione dei check necessari per rispettare le regole aziendali proposte. Di seguito il codice SQL per generarli.

```
USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`arbitro_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`arbitro_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`arbitro`
FOR EACH ROW
BEGIN
	if NEW.numero < 0 or NEW.presenze < 0 then
			signal sqlstate '45000';
	end if;
END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`cliente_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`cliente_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`cliente`
FOR EACH ROW
BEGIN
	if NEW.numero < 0 then
			signal sqlstate '45000';
	end if;
   if NEW.anno_nascita < 1900 or NEW.anno_nascita > 2003 then
			signal sqlstate '45000';
	end if;
END$$
```

```
USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`cliente_BEFORE_UPDATE` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`cliente_BEFORE_UPDATE`
BEFORE UPDATE ON `calcioscommesse`.`cliente`
FOR EACH ROW
BEGIN
        if NEW.numero < 0 then
                        signal sqlstate '45000';
        end if;
   if NEW.anno_nascita < 1900 or NEW.anno_nascita > 2003 then
                        signal sqlstate '45000';
        end if;
END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`edizione_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`edizione_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`edizione`
FOR EACH ROW
BEGIN
        if NEW.anno = 1942 or NEW.anno = 1946 or NEW.anno < 1930 then
                        signal sqlstate '45000';
        end if;
   if ((NEW.anno - 1930)%4) != 0 then
                        signal sqlstate '45000';
        end if;
END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`giocatore_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`giocatore_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`giocatore`
FOR EACH ROW
BEGIN
        declare counter INT;
        if NEW.n_maglia < 0 or NEW.n_maglia > 99 then
                        signal sqlstate '45000';
        end if;
   if NEW.numero < 0 then
```

```
                        signal sqlstate '45000';
        end if;
        select count(*) from giocatore where squadra=NEW.squadra and n_maglia=NEW.n_maglia
into counter;
        if counter>0 then
                signal sqlstate '45000';
        end if;

END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`turno_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`turno_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`turno`
FOR EACH ROW
BEGIN
        declare c1,c2,c3,c4,c5 INT;
   select strcmp(NEW.nome, "qualificazione") into c1;
   select strcmp(NEW.nome, "ottavi") into c2;
   select strcmp(NEW.nome, "quarti") into c3;
   select strcmp(NEW.nome, "semifinale") into c4;
   select strcmp(NEW.nome, "finale") into c5;
        if c1!=0 and c2!=0 and c3!=0 and c4!=0 and c5!=0 then
                signal sqlstate '45000';
        end if;
END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`partita_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`partita_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`partita`
FOR EACH ROW
BEGIN
        declare counter1, counter2 INT;
   declare turnoprec varchar(15);
   declare matcht1, matcht2 timestamp;
   if NEW.turno != "qualificazione" then
                select count(*) from partita where turno = NEW.turno and (sq1 = NEW.sq1 or sq2 =
NEW.sq1) into counter1;
                select count(*) from partita where turno = NEW.turno and (sq1 = NEW.sq2 or sq2 =
NEW.sq2) into counter2;
                if counter1 != 0 or counter2 != 0 then
```

```
                                signal sqlstate '45000' SET MESSAGE_TEXT = 'Una delle squadre
partecipa gia a una partita!';
                end if;
        end if;
   select count(*) from partecipa where squadra=NEW.sq1 and turno=NEW.turno into counter1;
   select count(*) from partecipa where squadra=NEW.sq2 and turno=NEW.turno into counter2;

   if counter1 = 0 or counter2 = 0 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Le squadre non partecipano a
questo turno!';
        end if;

   if NEW.sq1=NEW.sq2 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Non puo partecipare solo una
squadra alla partita!';
        end if;

        if(NEW.turno="quarti") then
                set turnoprec="ottavi";
   elseif(NEW.turno="semifinale") then
                set turnoprec="quarti";
   elseif(NEW.turno="finale") then
                set turnoprec="semifinale";
        end if;

   select ora
   from partita
   where sq1=NEW.sq1 or sq2=NEW.sq1 and turno=turnoprec
   into matcht1;

   select ora
   from partita
   where sq1=NEW.sq2 or sq2=NEW.sq2 and turno=turnoprec
   into matcht2;

   if((NEW.ora-matcht1)<0 or (NEW.ora-matcht2)<0) then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Partite non ordinate correttamente!';
        end if;

   update arbitro set presenze=presenze+1 where nome=NEW.arbitro;
END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`goal_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`goal_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`goal`
```

```
FOR EACH ROW
BEGIN
        declare squadra1, squadra2, sqgiocatore varchar(45);
   declare matcht timestamp;
        if NEW.minuto<1 or NEW.min_recupero<0  then
                        signal sqlstate '45000' SET MESSAGE_TEXT = 'Valore minuti non valido!';
        end if;

   select sq1,sq2 from partita where numero=NEW.partita and turno=NEW.turno into
squadra1,squadra2;
   select squadra from giocatore where nome=NEW.giocatore into sqgiocatore;
   select ora from partita where numero=NEW.partita and turno=NEW.turno into matcht;

   if sqgiocatore != squadra1 and sqgiocatore != squadra2 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Il giocatore non appartiene ad una
delle squadre che si sono affrontate!';
        end if;

   if (NEW.minuto!=45 and NEW.minuto!=90 and NEW.minuto!=105 and NEW.minuto!=120) and
NEW.min_recupero !=0 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Minuto di recupero non valido!';
        end if;

   if NEW.minuto>120 and NEW.rigore=0 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Durante la fase a rigori, solo i rigori
sono ammessi!';
        end if;

   if NEW.rigore!=0 and NEW.rigore!=1 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'Valore attributo rigore sbagliato!';
        end if;

   if (matcht-NOW())>0 then
                signal sqlstate '45000' SET MESSAGE_TEXT = 'La partita non e stata ancora
giocata!';
        end if;

        if(NEW.turno="qualificazione" and NEW.minuto>90) then
                signal sqlstate '45000' SET MESSAGE_TEXT = "In questa fase non ci sono i tempi
supplementari!";
        end if;


END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`partecipa_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
```

```
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`partecipa_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`partecipa`
FOR EACH ROW
BEGIN
        declare counter INT;
   if NEW.turno = "qualificazione" then
                select count(*) from partecipa where turno="qualificazione" into counter;
      if counter > 31 then
                        signal sqlstate '45000' SET MESSAGE_TEXT = 'Numero squadre non
congruo alle regole del calcio!';
                end if;
   elseif NEW.turno = "ottavi" then
                select count(*) from partecipa where turno="ottavi" into counter;
      if counter > 15 then
                        signal sqlstate '45000'SET MESSAGE_TEXT = 'Numero squadre per questo
turno non congruo alle regole del calcio!';
                end if;
   elseif NEW.turno = "quarti" then
                select count(*) from partecipa where turno="quarti" into counter;
      if counter > 7 then
                        signal sqlstate '45000'SET MESSAGE_TEXT = 'Numero squadre per questo
turno non congruo alle regole del calcio!';
                end if;
         elseif NEW.turno = "semifinale" then
                select count(*) from partecipa where turno="semifinale" into counter;
      if counter > 3 then
                        signal sqlstate '45000'SET MESSAGE_TEXT = 'Numero squadre per questo
turno non congruo alle regole del calcio!';
                end if;
         elseif NEW.turno = "finale" then
                select count(*) from partecipa where turno="finale" into counter;
      if counter > 1 then
                        signal sqlstate '45000'SET MESSAGE_TEXT = 'Numero squadre per questo
turno non congruo alle regole del calcio!';
                end if;
         end if;

END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`scommessa_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`scommessa_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`scommessa`
FOR EACH ROW
BEGIN
```

```
        if NEW.esito < 0 or NEW.esito > 2 then
                        signal sqlstate '45000'SET MESSAGE_TEXT = 'Esito non valido!';
        end if;
   if NEW.esito = 0 and NEW.turno != "qualificazione" then
                        signal sqlstate '45000' SET MESSAGE_TEXT = 'Non si puo pareggiare dopo
le qualificazioni!';
        end if;
   if NEW.moltiplicatore<1 then
                        signal sqlstate '45000' SET MESSAGE_TEXT = 'Non si puo avere un
moltiplicatore <1!';
        end if;

END$$


USE `calcioscommesse`$$
DROP TRIGGER IF EXISTS `calcioscommesse`.`scommessa_reale_BEFORE_INSERT` $$
USE `calcioscommesse`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `calcioscommesse`.`scommessa_reale_BEFORE_INSERT`
BEFORE INSERT ON `calcioscommesse`.`scommessa_reale`
FOR EACH ROW
BEGIN
        declare orariop TIMESTAMP;
        if NEW.importo < 0 then
                        signal sqlstate '45000' set message_text = "Non si puo scommettere un
importo negativo!";
        end if;
   select ora from partita where numero=NEW.partita and turno=NEW.turno into orariop;
   if orariop - NOW() < 0 then
                        signal sqlstate '45000' set message_text = "Non si puo scommettere su una
partita gia giocata!";
        end if;

END$$
```

Eventi

Per la realizzazione di questo progetto nessun evento è stato implementato.

Viste

La dimensione ridotta della base di dati e delle operazioni ha fatto si che nessuna vista risultasse necessaria.

Stored Procedures e transazioni

```
-- -----------------------------------------------------
-- procedure aggiorna_tornero
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiorna_tornero`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiorna_tornero`(in turnosucc
varchar(15))
BEGIN
        declare counter1, counter2, counterp int;
   declare ultima_partita timestamp;
   declare squadra1, squadra2 varchar(45);
   declare turnoprec varchar(15);

   declare exit handler for sqlexception
   begin
      rollback;  -- rollback any changes made in the transaction
      resignal;  -- raise again the sql exception to the caller
   end;

   if turnosucc="quarti" then
                set turnoprec = "ottavi";
      set counterp = 8;
   end if;

   if turnosucc="semifinale" then
                set turnoprec = "quarti";
      set counterp = 4;
   end if;

   if turnosucc="finale" then
                set turnoprec = "semifinale";
      set counterp = 2;
   end if;



   set transaction isolation level serializable;
   start transaction;

   select count(*) from partecipa where turno=turnosucc into counter1;
   if counter1!=0 then
                signal sqlstate '45001' set message_text = "Squadre del turno successivo gia
calcolate!";
        end if;
```

```
   select max(ora) from partita where turno = turnoprec into ultima_partita;
   if ultima_partita > (NOW() - interval 3 hour) then
                signal sqlstate '45002' set message_text = "Non sono ancora state giocate tutte le
partite!";
         end if;

   select count(*) from partita where turno = turnoprec into counter1;
   if counter1!=counterp then
                signal sqlstate '45003' set message_text = "Non c'e un numero di partite sufficiente
per determinare il prossimo turno!";
   end if;

   begin

                                    #squadre per ciascuna partita
                                    declare done int default false;
                                    declare cur cursor for select sq1, sq2
                                    from `partita`
                                    where turno = turnoprec;

                                    declare continue handler for not found set done = true;

                           open cur;
         contagoal: loop
                                    fetch cur into squadra1, squadra2;

            if done then
                                             leave contagoal;
                                    end if;

            #goal squadra1
            select count(*)
            from partita join goal on goal.partita=partita.numero and goal.turno=partita.turno
            join giocatore on goal.giocatore=giocatore.nome
            where giocatore.squadra=squadra1  into counter1;

            #goal squadra2
            select count(*)
            from partita join goal on goal.partita=partita.numero and goal.turno=partita.turno
            join giocatore on goal.giocatore=giocatore.nome
            where giocatore.squadra=squadra2  into counter2;

            if counter1=counter2 then
                                             signal sqlstate '45000' set message_text = "La partita non puÃ²
essere finita in parita! Aggiornamento goal richiesto!";
                                    end if;

            #inserisci squadra vincitrice nel turno successivo
            if counter1 > counter2 then
```

```
                                        INSERT INTO partecipa(turno, squadra) VALUES (turnosucc,
squadra1);
                                else
                                        INSERT INTO partecipa(turno, squadra) VALUES (turnosucc,
squadra2);
                                end if;


        end loop;
        close cur;
        end;
        commit;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_arbitro
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_arbitro`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_arbitro`(in v_nome varchar(45),
in v_pres int, in v_paese varchar(30), in v_citta varchar(45), in v_via varchar(45), in v_num int)
BEGIN
        insert into arbitro values (v_nome, v_pres, v_paese, v_citta, v_via, v_num);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_cliente
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_cliente`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_cliente`(IN v_cf varchar(16), IN
v_nome varchar(45), IN v_cognome varchar(45), IN v_anno INT, IN v_paese varchar(30), IN
v_citta varchar(45), IN v_via varchar(45), IN v_num INT)
BEGIN
        insert into cliente values(v_cf, v_nome, v_cognome, v_anno, v_paese, v_citta, v_num,
v_via);
END$$
```

```
DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_giocatore
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_giocatore`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_giocatore`(in v_nome
varchar(45), in v_maglia int, in v_paese varchar(30), in v_citta varchar(45), in v_via varchar(45), in
v_numero int, in v_squadra varchar(45) )
BEGIN

   insert into giocatore values ( v_nome, v_maglia, v_paese, v_citta, v_via, v_numero, v_squadra);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_goal
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_goal`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_goal`(in v_partita int, in v_turno
varchar(15), in v_minuto int, in  v_rigore tinyint, in v_giocatore varchar(45), in v_min_recupero int)
BEGIN

        insert into goal values(v_partita, v_turno, v_minuto, v_rigore, v_giocatore, v_min_recupero);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_partecipazione
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_partecipazione`;

DELIMITER $$
USE `calcioscommesse`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_partecipazione`(in v_turno
varchar(15), in v_squadra varchar(45))
BEGIN
	insert into partecipa values(v_turno, v_squadra);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_partita_e_scommesse
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_partita_e_scommesse`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_partita_e_scommesse`(IN v_num
int, IN v_turno varchar(15), in v_sq1 varchar(45), in v_sq2 varchar(45), IN v_arbitro varchar(45), IN
v_orario timestamp, IN v_citta varchar(45), IN v_molt1 double, IN v_molt2 double, IN v_moltpar
double)
BEGIN

	declare exit handler for sqlexception
  begin
    rollback;  -- rollback any changes made in the transaction
    resignal;  -- raise again the sql exception to the caller
  end;

  set transaction isolation level read committed;
  start transaction;

		insert into partita values(v_num, v_turno, v_sq1, v_sq2, v_arbitro, v_orario, v_citta);
		if v_turno="qualificazione" then
			insert into scommessa values (v_num, v_turno, 0, v_moltpar);
		end if;
		insert into scommessa values (v_num, v_turno, 1, v_molt1);
		insert into scommessa values (v_num, v_turno, 2, v_molt2);
	commit;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_scommessa_cliente
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_scommessa_cliente`;
```

```
DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_scommessa_cliente`(IN v_partita
int, IN v_turno varchar(15), IN v_esito int, IN v_codice varchar(15), IN v_importo double, IN
v_cliente varchar(16))
BEGIN
	insert into scommessa_reale values (v_partita, v_turno, v_esito, v_codice, v_importo,
v_cliente);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_squadra
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_squadra`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_squadra`(in v_nome varchar(45),
in v_allenatore varchar(45))
BEGIN
	insert into squadra values (v_nome, v_allenatore);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure aggiungi_utente
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`aggiungi_utente`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `aggiungi_utente`(in v_nome varchar(45),
in v_cognome varchar(45), in v_username varchar(45), in v_pass varchar(45), in v_ruolo
varchar(45))
BEGIN
	insert into utente values( v_nome, v_cognome, v_username, MD5(v_pass), v_ruolo);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure login_utente
-- -----------------------------------------------------
```

```
USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`login_utente`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `login_utente`(in v_username varchar(45),
in v_pass varchar(45), out v_ruolo INT)
BEGIN
	declare var_user_role ENUM('amministratore', 'addetto_sportelli');

	select ruolo from utente
		where username = v_username
      and password = md5(v_pass)
      into var_user_role;

      -- See the corresponding enum in the client
		if var_user_role = 'amministratore' then
			set v_ruolo = 1;
		elseif var_user_role = 'addetto_sportelli' then
			set v_ruolo = 2;
		else
			set v_ruolo = 3;
		end if;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure modifica_cliente
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`modifica_cliente`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `modifica_cliente`(IN v_cf varchar(16), IN
v_nome varchar(45), IN v_cognome varchar(45), IN v_anno INT, IN v_paese varchar(30), IN
v_citta varchar(45), IN v_via varchar(45), IN v_num INT)
BEGIN
	update cliente
   set nome=v_nome,
   cognome=v_cognome,
   anno_nascita=v_anno,
   paese=v_paese,
   citta=v_citta,
   via=v_via,
   numero=v_num
   where cf=v_cf;
```

```
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_capocannoniere
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_capocannoniere`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_capocannoniere`()
BEGIN

        declare countg int;
    declare cgiocatore varchar(45);
        declare exit handler for sqlexception
    begin
      rollback;  -- rollback any changes made in the transaction
      resignal;  -- raise again the sql exception to the caller
    end;

    set transaction isolation level read committed;
    start transaction;

                    select count(*), giocatore
        from goal as g
        group by g.giocatore
        having count(*) = (
                                                        select count(*) as conta
                        from goal
                        where giocatore = g.giocatore
                        group by giocatore
                        order by conta desc)
                    limit 1 into countg, cgiocatore;

        select countg as numero_goal;
        select nome, n_maglia, squadra from giocatore where nome = cgiocatore;

          commit;


END$$

DELIMITER ;

-- -----------------------------------------------------
```

```
-- procedure trova_cliente_ricco
-- -----------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_cliente_ricco`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_cliente_ricco`()
BEGIN
        declare n_cliente varchar(16);
   declare importoc double;
        declare exit handler for sqlexception
   begin
      rollback;  -- rollback any changes made in the transaction
      resignal;  -- raise again the sql exception to the caller
   end;

   set transaction isolation level serializable;
   start transaction;
                select sc.cliente as cliente, sc.importo as importo
      from scommessa_reale as sc
      join partita as p on sc.partita=p.numero and sc.turno=p.turno
      join cliente as c on sc.cliente=c.cf
      having sc.importo = (
                                                select max(importo)
                  from scommessa_reale
                  ) into n_cliente, importoc;


                select c.nome as nome, c.cognome as cognome, c.paese as paese, sr.importo as
importo_scommessa,
      p.turno as turno, p.sq1 as squadra_1, p.sq2 as squadra_2, sr.esito as esito_scommesso
      from scommessa_reale as sr
      join cliente as c on c.cf=sr.cliente
      join partita as p on p.numero=sr.partita and p.turno=sr.turno
      where c.cf=n_cliente;


END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_moltiplicatore_maggiore
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_moltiplicatore_maggiore`;
```

```
DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_moltiplicatore_maggiore`()
BEGIN
	select p.sq1 as squadra1, p.sq2 as squadra2, p.turno as turno, s.esito as esito, s.moltiplicatore
as molitiplicatore
   from scommessa as s join partita as p on s.partita=p.numero and s.turno=p.turno
   where s.moltiplicatore = (
                                                                select max(moltiplicatore)
                        from scommessa);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_partita_piu_scommessa
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_partita_piu_scommessa`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_partita_piu_scommessa`()
BEGIN
	declare v_numero, v_partita int;
   declare v_turno varchar(15);

   declare exit handler for sqlexception
   begin
      rollback;  -- rollback any changes made in the transaction
      resignal;  -- raise again the sql exception to the caller
   end;

   set transaction isolation level read committed;
   start transaction;

                select count(*), partita, turno
                from scommessa_reale
                group by partita, turno
                order by count(*) desc
                limit 1  into v_numero, v_partita, v_turno;

      select v_numero as bet_number;

                select * from partita where numero=v_partita and turno=v_turno;

         commit;

END$$
```

```
DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_partite_future
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_partite_future`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_partite_future`()
BEGIN
	select *
   from partita
   where ((ora + interval 3 hour) - NOW() )>0;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_piu_goal
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_piu_goal`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_piu_goal`()
BEGIN


	declare v_goal, v_partita int;
   declare v_turno varchar(15);

   declare exit handler for sqlexception
	begin
			rollback;  -- rollback any changes made in the transaction
			resignal;  -- raise again the sql exception to the caller
	end;

   set transaction isolation level read committed;
   start transaction;

	select count(*),partita, turno
   from goal
   group by partita, turno
   order by count(*) desc
```

```
    limit 1 into v_goal, v_partita, v_turno;

    select v_goal as goal_number;

    select * from partita where numero=v_partita and turno=v_turno;

    commit;

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_presenze_arbitro
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_presenze_arbitro`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_presenze_arbitro`(in v_nome
varchar(45))
BEGIN
        select presenze from arbitro where nome=v_nome;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure trova_sq_per_turno
-- -----------------------------------------------------

USE `calcioscommesse`;
DROP procedure IF EXISTS `calcioscommesse`.`trova_sq_per_turno`;

DELIMITER $$
USE `calcioscommesse`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trova_sq_per_turno`(in v_turno
varchar(15))
BEGIN
                select nome as squadre
    from partecipa as p join squadra as s on p.squadra = s.nome
    where p.turno = v_turno;
END$$

DELIMITER ;
USE `calcioscommesse`;

DELIMITER $$
```

Una parte importante di queste procedure servono per inserimenti singoli o altre funzioni immediate che non richiedono transizioni di alcun tipo.
Le procedure che meritano di essere menzionate sono:

- aggiorna_torneo: la procedura più complessa implementata, ha richiesto una serie di operazioni in una transazione ad un livello di isolazione serializable: per aggiornare automaticamente la rosa delle squadra partecipanti ad un determinato turno del torneo in base ai risultati del turno successivo, sono necessari più di un paio di accessi in diverse tabelle del database solo per i controlli preliminari per verificare l'effettiva possibilità di realizzare questa funzione automaticamente, si è ritenuto necessario che nessuna delle tabelle accedute possa essere modificata durante l'esecuzione di questa transazione

- trova_cliente_ricco: pur essendo abbastanza semplice come procedura, come per la precedente ci sono diversi accessi a diverse tabelle: affinché il risultato sia consistente, non vogliamo interferenze ed usiamo il livello serializable

- aggiungi_partita_e_scommessa: questa transizione è stata adoperata solo per sfruttare la proprietà di atomicità delle transazioni: in questa procedura abbiamo diversi inserimenti in serie e la base di dati risulterebbe inconsistente nel caso in cui, per esempio, solo una parte di essi andasse a buon fine. Un livello read committed è sufficiente.

- trova_piu_goal: in caso di inserimenti non andati ancora in commit, potremmo ottenere un risultato non consistente nella ricerca della partita con più goal segnati: è stato ritenuto necessario l'uso di una transazione di livello read committed per evitare ciò.

Appendice: Implementazione

Codice SQL per instanziare il database

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema calcioscommesse
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `calcioscommesse` ;

-- -----------------------------------------------------
-- Schema calcioscommesse
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `calcioscommesse` DEFAULT CHARACTER SET latin1 ;
USE `calcioscommesse` ;

-- -----------------------------------------------------
-- Table `calcioscommesse`.`arbitro`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`arbitro` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`arbitro` (
  `nome` VARCHAR(45) NOT NULL,
  `presenze` INT(11) NULL DEFAULT NULL,
  `paese` VARCHAR(30) NULL DEFAULT NULL,
  `citta` VARCHAR(45) NULL DEFAULT NULL,
  `via` VARCHAR(45) NULL DEFAULT NULL,
  `numero` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`nome`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`cliente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`cliente` ;
```

```
CREATE TABLE IF NOT EXISTS `calcioscommesse`.`cliente` (
  `cf` VARCHAR(16) NOT NULL,
  `nome` VARCHAR(45) NULL DEFAULT NULL,
  `cognome` VARCHAR(45) NULL DEFAULT NULL,
  `anno_nascita` INT(11) NULL DEFAULT NULL,
  `paese` VARCHAR(30) NULL DEFAULT NULL,
  `citta` VARCHAR(45) NULL DEFAULT NULL,
  `numero` INT(11) NULL DEFAULT NULL,
  `via` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`cf`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`squadra`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`squadra` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`squadra` (
  `nome` VARCHAR(45) NOT NULL,
  `allenatore` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`nome`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`edizione`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`edizione` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`edizione` (
  `anno` INT(11) NOT NULL,
  `squadra` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`anno`),
  CONSTRAINT `edizione.squadra`
    FOREIGN KEY (`squadra`)
    REFERENCES `calcioscommesse`.`squadra` (`nome`)
    ON DELETE SET NULL
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `edizione.squadra_idx` ON `calcioscommesse`.`edizione` (`squadra` ASC)
VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`giocatore`
```

```
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`giocatore` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`giocatore` (
  `nome` VARCHAR(45) NOT NULL,
  `n_maglia` INT(11) NULL DEFAULT NULL,
  `paese` VARCHAR(30) NULL DEFAULT NULL,
  `citta` VARCHAR(45) NULL DEFAULT NULL,
  `via` VARCHAR(45) NULL DEFAULT NULL,
  `numero` INT(11) NULL DEFAULT NULL,
  `squadra` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`nome`),
  CONSTRAINT `giocatore.squadra`
    FOREIGN KEY (`squadra`)
    REFERENCES `calcioscommesse`.`squadra` (`nome`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `giocatore.squadra_idx` ON `calcioscommesse`.`giocatore` (`squadra` ASC)
VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`turno`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`turno` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`turno` (
  `nome` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`nome`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`partita`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`partita` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`partita` (
  `numero` INT(11) NOT NULL,
  `turno` VARCHAR(15) NOT NULL,
  `sq1` VARCHAR(45) NULL DEFAULT NULL,
  `sq2` VARCHAR(45) NULL DEFAULT NULL,
  `arbitro` VARCHAR(45) NULL DEFAULT NULL,
  `ora` TIMESTAMP NULL DEFAULT NULL,
  `citta` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`turno`, `numero`),
```

```
  CONSTRAINT `partita.arbitro`
    FOREIGN KEY (`arbitro`)
    REFERENCES `calcioscommesse`.`arbitro` (`nome`)
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  CONSTRAINT `partita.sq1`
    FOREIGN KEY (`sq1`)
    REFERENCES `calcioscommesse`.`squadra` (`nome`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `partita.sq2`
    FOREIGN KEY (`sq2`)
    REFERENCES `calcioscommesse`.`squadra` (`nome`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `partita.turno`
    FOREIGN KEY (`turno`)
    REFERENCES `calcioscommesse`.`turno` (`nome`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `partita.sq1_idx` ON `calcioscommesse`.`partita` (`sq1` ASC) VISIBLE;

CREATE INDEX `partita.sq2_idx` ON `calcioscommesse`.`partita` (`sq2` ASC) VISIBLE;

CREATE INDEX `partita.arbitro_idx` ON `calcioscommesse`.`partita` (`arbitro` ASC) VISIBLE;

CREATE INDEX `partita.numero_idx` ON `calcioscommesse`.`partita` (`numero` ASC) VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`goal`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`goal` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`goal` (
  `partita` INT(11) NOT NULL,
  `turno` VARCHAR(15) NOT NULL,
  `minuto` INT(11) NOT NULL,
  `rigore` TINYINT(4) NULL DEFAULT NULL,
  `giocatore` VARCHAR(45) NULL DEFAULT NULL,
  `min_recupero` INT(11) NOT NULL,
  PRIMARY KEY (`partita`, `turno`, `minuto`, `min_recupero`),
  CONSTRAINT `goal.giocatore`
    FOREIGN KEY (`giocatore`)
    REFERENCES `calcioscommesse`.`giocatore` (`nome`)
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  CONSTRAINT `goal.partita`
```

```
    FOREIGN KEY (`partita` , `turno`)
    REFERENCES `calcioscommesse`.`partita` (`numero` , `turno`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `goal.giocatore_idx` ON `calcioscommesse`.`goal` (`giocatore` ASC) VISIBLE;

CREATE INDEX `goal.turno_idx` ON `calcioscommesse`.`goal` (`turno` ASC) VISIBLE;

CREATE INDEX `goal.partita_idx` ON `calcioscommesse`.`goal` (`partita` ASC) VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`partecipa`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`partecipa` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`partecipa` (
  `turno` VARCHAR(15) NOT NULL,
  `squadra` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`turno`, `squadra`),
  CONSTRAINT `partecipa.squadra`
    FOREIGN KEY (`squadra`)
    REFERENCES `calcioscommesse`.`squadra` (`nome`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `partecipa.turno`
    FOREIGN KEY (`turno`)
    REFERENCES `calcioscommesse`.`turno` (`nome`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `partecipa.squadra_idx` ON `calcioscommesse`.`partecipa` (`squadra` ASC)
VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`scommessa`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`scommessa` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`scommessa` (
  `partita` INT(11) NOT NULL,
  `turno` VARCHAR(15) NOT NULL,
  `esito` INT(11) NOT NULL,
  `moltiplicatore` DOUBLE NOT NULL,
  PRIMARY KEY (`partita`, `turno`, `esito`),
  CONSTRAINT `scommessa.partita`
```

```
    FOREIGN KEY (`partita` , `turno`)
    REFERENCES `calcioscommesse`.`partita` (`numero` , `turno`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `scommessa.esito_idx` ON `calcioscommesse`.`scommessa` (`esito` ASC)
VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`scommessa_reale`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`scommessa_reale` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`scommessa_reale` (
  `partita` INT(11) NOT NULL,
  `turno` VARCHAR(15) NOT NULL,
  `esito` INT(11) NOT NULL,
  `codice` VARCHAR(15) NOT NULL,
  `importo` DOUBLE NOT NULL,
  `cliente` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`codice`),
  CONSTRAINT `scommessa_reale.cliente`
    FOREIGN KEY (`cliente`)
    REFERENCES `calcioscommesse`.`cliente` (`cf`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `scommessa_reale.partita`
    FOREIGN KEY (`esito` , `partita` , `turno`)
    REFERENCES `calcioscommesse`.`scommessa` (`esito` , `partita` , `turno`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

CREATE INDEX `scommessa_reale.cliente_idx` ON `calcioscommesse`.`scommessa_reale`
(`cliente` ASC) VISIBLE;

CREATE INDEX `scommessa_reale.esito_idx` ON `calcioscommesse`.`scommessa_reale` (`esito`
ASC, `partita` ASC, `turno` ASC) VISIBLE;


-- -----------------------------------------------------
-- Table `calcioscommesse`.`utente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calcioscommesse`.`utente` ;

CREATE TABLE IF NOT EXISTS `calcioscommesse`.`utente` (
  `nome` VARCHAR(45) NULL DEFAULT NULL,
  `cognome` VARCHAR(45) NULL DEFAULT NULL,
  `username` VARCHAR(45) NOT NULL,
```

```
  `password` VARCHAR(45) NOT NULL,
  `ruolo` ENUM('amministratore', 'addetto_sportelli') NOT NULL,
  PRIMARY KEY (`username`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;

USE `calcioscommesse` ;




DELIMITER ;
SET SQL_MODE = '';
DROP USER IF EXISTS login;
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
CREATE USER 'login' IDENTIFIED BY 'login';

GRANT EXECUTE ON procedure `calcioscommesse`.`login_utente` TO 'login';
SET SQL_MODE = '';
DROP USER IF EXISTS amministratore;
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
CREATE USER 'amministratore' IDENTIFIED BY 'amministratore';

GRANT EXECUTE ON procedure `calcioscommesse`.`aggiorna_tornero` TO 'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_arbitro` TO 'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_giocatore` TO 'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_goal` TO 'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_partita_e_scommesse` TO
'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_squadra` TO 'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_partecipazione` TO
'amministratore';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_utente` TO 'amministratore';
SET SQL_MODE = '';
DROP USER IF EXISTS addetto_sportelli;
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
CREATE USER 'addetto_sportelli' IDENTIFIED BY 'addetto_sportelli';

GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_cliente` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`modifica_cliente` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`aggiungi_scommessa_cliente` TO
'addetto_sportelli';
```

```
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_moltiplicatore_maggiore` TO
'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_partite_future` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_partita_piu_scommessa` TO
'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_sq_per_turno` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_capocannoniere` TO
'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_cliente_ricco` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_piu_goal` TO 'addetto_sportelli';
GRANT EXECUTE ON procedure `calcioscommesse`.`trova_presenze_arbitro` TO
'addetto_sportelli';

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

INSERT INTO `calcioscommesse`.`turno` (`nome`) VALUES ('qualificazione');
INSERT INTO `calcioscommesse`.`turno` (`nome`) VALUES ('ottavi');
INSERT INTO `calcioscommesse`.`turno` (`nome`) VALUES ('quarti');
INSERT INTO `calcioscommesse`.`turno` (`nome`) VALUES ('semifinale');
INSERT INTO `calcioscommesse`.`turno` (`nome`) VALUES ('finale');
```

Codice del Front-End

main.c

```
#pragma warning(disable : 4996)
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>

#include "defines.h"

typedef enum {
	ADMINISTRATOR = 1,
	COUNTER_CLERK,
	FAILED_LOGIN
} role_t;

#define DIM 46


struct configuration conf;

static MYSQL* conn;

static role_t attempt_login(MYSQL* conn, char* username, char* password) {
	MYSQL_STMT* login_procedure;
	MYSQL_BIND param[3]; // Used both for input and output
	int role = 0;

	if (!setup_prepared_stmt(&login_procedure, "call login_utente(?, ?, ?)", conn)) {
		print_stmt_error(login_procedure, "Unable to initialize login statement\n");
		goto err2;
	}
	// Prepare parameters
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
	param[0].buffer = username;
	param[0].buffer_length = strlen(username);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
	param[1].buffer = password;
	param[1].buffer_length = strlen(password);

	param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
	param[2].buffer = &role;
	param[2].buffer_length = sizeof(role);
	if (mysql_stmt_bind_param(login_procedure, param) != 0) { // Note _param
		print_stmt_error(login_procedure, "Could not bind parameters for login");
		goto err;
	}

	// Run procedure
	if (mysql_stmt_execute(login_procedure) != 0) {
		print_stmt_error(login_procedure, "Could not execute login procedure");
		goto err;
	}
	// Prepare output parameters
	memset(param, 0, sizeof(param));
```

```
	param[0].buffer_type = MYSQL_TYPE_LONG; // OUT
	param[0].buffer = &role;
	param[0].buffer_length = sizeof(role);

	if (mysql_stmt_bind_result(login_procedure, param)) {
		print_stmt_error(login_procedure, "Could not retrieve output parameter");
		goto err;
	}

	// Retrieve output parameter
	if (mysql_stmt_fetch(login_procedure)) {
		print_stmt_error(login_procedure, "Could not buffer results");
		goto err;
	}
	mysql_stmt_close(login_procedure);
	return role;

err:
	mysql_stmt_close(login_procedure);
err2:
	return FAILED_LOGIN;
}



int main() {

	role_t role;


	if (!parse_config("users/login.json", &conf)) {
		fprintf(stderr, "Unable to load login configuration\n");
		system("PAUSE");
		exit(EXIT_FAILURE);
	}

	conn = mysql_init(NULL);
	if (conn == NULL) {
		fprintf(stderr, "mysql_init() failed (probably out of memory)\n");
		system("PAUSE");
		exit(EXIT_FAILURE);
	}

	if (mysql_real_connect(conn, conf.host, conf.db_username, conf.db_password, conf.database, conf.port, NULL,
CLIENT_MULTI_STATEMENTS | CLIENT_MULTI_RESULTS) == NULL) {
		fprintf(stderr, "mysql_real_connect() failed\n");
		mysql_close(conn);
		system("PAUSE");
		exit(EXIT_FAILURE);
	}

	printf("Insert Username: ");
	fflush(stdout);
	fgets(conf.username, DIM, stdin);
	conf.username[strlen(conf.username) - 1] = '\0';
	printf("\nInsert Password: ");
	fflush(stdout);
	insertPassword(conf.password);


	role = attempt_login(conn, conf.username, conf.password);

	switch (role) {
```

```
        case COUNTER_CLERK:
                run_as_counter_clerk(conn);
                break;

        case ADMINISTRATOR:
                run_as_administrator(conn);
                break;

        case FAILED_LOGIN:
                fprintf(stderr, "Invalid credentials\n");
                exit(EXIT_FAILURE);
                break;

        default:
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                abort();
        }

        printf("\nBye!\n");

        mysql_close(conn);


}
```

counter_clerk.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

#pragma warning(disable : 4996)

void add_customer(MYSQL* conn)
{
	MYSQL_STMT* add_customer_stmt;
	MYSQL_BIND param[8];


	char fiscal_code[20];
	char name[48], surname[48];
	char s_byear[16], s_number[16];
	char country[36], city[48], street[48];


	printf("\nFiscal code: ");
	fflush(stdout);
	fgets(fiscal_code, 20, stdin);
	fiscal_code[strlen(fiscal_code) - 1] = '\0';

	printf("\nName: ");
	fflush(stdout);
	fgets(name, 48, stdin);
	name[strlen(name) - 1] = '\0';

	printf("\nSurname: ");
	fflush(stdout);
	fgets(surname, 48, stdin);
	surname[strlen(surname) - 1] = '\0';

	printf("\nYear of birth[1900-2003]: ");
	fflush(stdout);
	fgets(s_byear, 16, stdin);

	printf("\nCountry of residence: ");
	fflush(stdout);
	fgets(country, 36, stdin);
	country[strlen(country) - 1] = '\0';

	printf("\nCity of residence: ");
	fflush(stdout);
	fgets(city, 48, stdin);
	city[strlen(city) - 1] = '\0';

	printf("\nStreet of residence: ");
	fflush(stdout);
	fgets(street, 48, stdin);
	street[strlen(street) - 1] = '\0';

	printf("\nHouse number[>=0]: ");
	fflush(stdout);
	fgets(s_number, 16, stdin);


	int byear = atoi(s_byear);
	int number = atoi(s_number);
```

```
	if (!setup_prepared_stmt(&add_customer_stmt, "call aggiungi_cliente(?, ?, ?, ?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, add_customer_stmt, "Unable to initialize customer insertion statement\n",
false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = fiscal_code;
	param[0].buffer_length = strlen(fiscal_code);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = name;
	param[1].buffer_length = strlen(name);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[2].buffer = surname;
	param[2].buffer_length = strlen(surname);

	param[3].buffer_type = MYSQL_TYPE_LONG;
	param[3].buffer = &byear;
	param[3].buffer_length = sizeof(byear);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = country;
	param[4].buffer_length = strlen(country);

	param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[5].buffer = city;
	param[5].buffer_length = strlen(city);

	param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[6].buffer = street;
	param[6].buffer_length = strlen(street);

	param[7].buffer_type = MYSQL_TYPE_LONG;
	param[7].buffer = &number;
	param[7].buffer_length = sizeof(number);


	if (mysql_stmt_bind_param(add_customer_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_customer_stmt, "Could not bind parameters for customer insertion\n",
true);
	}
	if (mysql_stmt_execute(add_customer_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_customer_stmt, "An error occurred while adding the customer, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_customer(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
```

```
	else {
		printf("\nCustomer correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(add_customer_stmt);
	}
}

void update_customer(MYSQL* conn)
{
	MYSQL_STMT* update_customer_stmt;
	MYSQL_BIND param[8];


	char fiscal_code[20];
	char name[48], surname[48];
	char s_byear[16], s_number[16];
	char country[36], city[48], street[48];


	printf("\nFiscal code of the customer you are trying to update: ");
	fflush(stdout);
	fgets(fiscal_code, 20, stdin);
	fiscal_code[strlen(fiscal_code) - 1] = '\0';

	printf("\nNew Name: ");
	fflush(stdout);
	fgets(name, 48, stdin);
	name[strlen(name) - 1] = '\0';

	printf("\nNew Surname: ");
	fflush(stdout);
	fgets(surname, 48, stdin);
	surname[strlen(surname) - 1] = '\0';

	printf("\nNew Year of birth[1900-2003]: ");
	fflush(stdout);
	fgets(s_byear, 16, stdin);

	printf("\nNew Country of residence: ");
	fflush(stdout);
	fgets(country, 36, stdin);
	country[strlen(country) - 1] = '\0';

	printf("\nNew City of residence: ");
	fflush(stdout);
	fgets(city, 48, stdin);
	city[strlen(city) - 1] = '\0';

	printf("\nNew Street of residence: ");
	fflush(stdout);
	fgets(street, 48, stdin);
	street[strlen(street) - 1] = '\0';

	printf("\nNew House number: ");
	fflush(stdout);
	fgets(s_number, 16, stdin);


	int byear = atoi(s_byear);
	int number = atoi(s_number);

	if (!setup_prepared_stmt(&update_customer_stmt, "call modifica_cliente(?, ?, ?, ?, ?, ?, ?, ?)", conn)) {
```

```
                finish_with_stmt_error(conn, update_customer_stmt, "Unable to initialize customer update statement\n",
false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = fiscal_code;
        param[0].buffer_length = strlen(fiscal_code);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = name;
        param[1].buffer_length = strlen(name);

        param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[2].buffer = surname;
        param[2].buffer_length = strlen(surname);

        param[3].buffer_type = MYSQL_TYPE_LONG;
        param[3].buffer = &byear;
        param[3].buffer_length = sizeof(byear);

        param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[4].buffer = country;
        param[4].buffer_length = strlen(country);

        param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[5].buffer = city;
        param[5].buffer_length = strlen(city);

        param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[6].buffer = street;
        param[6].buffer_length = strlen(street);

        param[7].buffer_type = MYSQL_TYPE_LONG;
        param[7].buffer = &number;
        param[7].buffer_length = sizeof(number);


        if (mysql_stmt_bind_param(update_customer_stmt, param) != 0) {
                finish_with_stmt_error(conn, update_customer_stmt, "Could not bind parameters for customer update\n",
true);
        }
        if (mysql_stmt_execute(update_customer_stmt) != 0) {
                char optionserr[2] = { '1','2' };
                char er;
                print_stmt_error(update_customer_stmt, "An error occurred while updating the customer, what do you want
to do?\n\t1)Try again;\n\t2)Quit\n");
                er = multiChoice("Select your answer: ", optionserr, 2);
                switch (er)
                {
                case '1':
                        update_customer(conn);
                        break;
                case '2':
                        return;
                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
                }
        }
        else {
                printf("\nCustomer correctly updated\n");
```

```
                fflush(stdout);
                system("PAUSE");
                mysql_stmt_close(update_customer_stmt);
        }
}

void add_bet(MYSQL* conn)
{
        MYSQL_STMT* add_bet_stmt;
        MYSQL_BIND param[6];

        char options[5] = { '1','2','3','4','5' };
        char r;


        char customer[20];
        char tournament[20], random_code[16];
        char s_match[16], s_amount[16];
        int result;

        static char charset[] = "abcdefghijklmnopqrstuvwxyz0123456789";

        for (int n = 0;n < 15;n++) {
                        int key = rand() % (int)(sizeof(charset) - 1);
                        random_code[n] = charset[key];
        }

        random_code[15] = '\0';

        printf("\nGenerated code: %s", random_code);



        printf("\nFiscal code of the customer who is betting: ");
        fflush(stdout);
        fgets(customer, 20, stdin);
        customer[strlen(customer) - 1] = '\0';

        printf("\nMatch number: ");
        fflush(stdout);
        fgets(s_match, 16, stdin);

        printf("\nWich tournament stage this match is?");
        printf("\n1)Qualifications");
        printf("\n2)Last 16");
        printf("\n3)Last 8");
        printf("\n4)Last 4");
        printf("\n5)Finals\n");
        r = multiChoice("Select the answer", options, 5);

        switch (r)
        {
        case '1':
                strcpy(tournament, "qualificazione");
                break;
        case '2':
                strcpy(tournament, "ottavi");
                break;
        case '3':
                strcpy(tournament, "quarti");
                break;
        case '4':
                strcpy(tournament, "semifinale");
```

```
            break;
case '5':
            strcpy(tournament, "finale");
            break;
default:
            fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
            abort();
}

printf("\nWho will win?");
printf("\n1)Noone");
printf("\n2)Team 1");
printf("\n3)Team 2\n");
r = multiChoice("Select the answer", options, 2);

switch (r)
{
case '1':
            result = 0;
            break;
case '2':
            result = 1;
            break;
case '3':
            result = 2;
            break;
default:
            fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
            abort();
}

printf("\nBet amount ( >0): ");
fflush(stdout);
fgets(s_amount, 16, stdin);

int match = atoi(s_match);
double amount = atof(s_amount);

if (!setup_prepared_stmt(&add_bet_stmt, "call aggiungi_scommessa_cliente(?, ?, ?, ?, ?, ?)", conn)) {
            finish_with_stmt_error(conn, add_bet_stmt, "Unable to initialize bet insertion statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &match;
param[0].buffer_length = sizeof(match);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = tournament;
param[1].buffer_length = strlen(tournament);

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &result;
param[2].buffer_length = sizeof(result);

param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
param[3].buffer = random_code;
param[3].buffer_length = strlen(random_code);

param[4].buffer_type = MYSQL_TYPE_DOUBLE;
param[4].buffer = &amount;
param[4].buffer_length = sizeof(amount);
```

```
	param[5].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[5].buffer = customer;
	param[5].buffer_length = strlen(customer);



	if (mysql_stmt_bind_param(add_bet_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_bet_stmt, "Could not bind parameters for bet insertion\n", true);
	}
	if (mysql_stmt_execute(add_bet_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_bet_stmt, "An error occurred while adding the bet, what do you want to do?\n\t1)Try
again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_bet(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {
		printf("\nBet correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(add_bet_stmt);
	}
}

void find_highest_multiplier(MYSQL* conn)
{
	MYSQL_STMT* find_highest_mult_stmt;


	if (!setup_prepared_stmt(&find_highest_mult_stmt, "call trova_moltiplicatore_maggiore()", conn)) {
		finish_with_stmt_error(conn, find_highest_mult_stmt, "Unable to initialize bet multiplier search query
statement\n", false);
	}


	if (mysql_stmt_execute(find_highest_mult_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_highest_mult_stmt, "An error occurred while trying to find the match and bet
multiplier, what do you want to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_highest_multiplier(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
```

```
		}
	}
	else {



		dump_result_set(conn, find_highest_mult_stmt, "Bet Infos:\n");
		system("PAUSE");
		if (mysql_stmt_next_result(find_highest_mult_stmt) > 0) {
			finish_with_stmt_error(conn, find_highest_mult_stmt, "Unexpected contidion\n", true);
			system("PAUSE");
		}
	}

	mysql_stmt_close(find_highest_mult_stmt);

}

void find_matches(MYSQL* conn)
{
	MYSQL_STMT* find_matches_stmt;


	if (!setup_prepared_stmt(&find_matches_stmt, "call trova_partite_future()", conn)) {
		finish_with_stmt_error(conn, find_matches_stmt, "Unable to initialize match search query statement\n",
false);
	}


	if (mysql_stmt_execute(find_matches_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_matches_stmt, "An error occurred while trying to find the match, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_matches(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {



		dump_result_set(conn, find_matches_stmt, "Match Infos:\n");
		system("PAUSE");
		if (mysql_stmt_next_result(find_matches_stmt) > 0) {
			finish_with_stmt_error(conn, find_matches_stmt, "Unexpected contidion\n", true);
			system("PAUSE");
		}
	}

	mysql_stmt_close(find_matches_stmt);

}
```

```
void find_more_bets(MYSQL* conn)
{
	MYSQL_STMT* find_bets_stmt;
	int status;


	if (!setup_prepared_stmt(&find_bets_stmt, "call trova_partita_piu_scommessa()", conn)) {
		finish_with_stmt_error(conn, find_bets_stmt, "Unable to initialize match search query statement\n", false);
	}


	if (mysql_stmt_execute(find_bets_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_bets_stmt, "An error occurred while trying to find the player, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_more_bets(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {


		do {
			dump_result_set(conn, find_bets_stmt, "");

			// more results? -1 = no, >0 = error, 0 = yes (keep looking)
			status = mysql_stmt_next_result(find_bets_stmt);
			if (status > 0)
				finish_with_stmt_error(conn, find_bets_stmt, "Unexpected condition", true);
		} while (status == 0);

	}
	system("pause");
	mysql_stmt_close(find_bets_stmt);
}

void find_turn_teams(MYSQL* conn)
{
	MYSQL_STMT* find_turn_matches_stmt;
	MYSQL_BIND param[1];

	char options[5] = { '1','2','3','4','5' };
	char r;

	char tournament[20];

	printf("\nWich tournament stage are you looking for?");
	printf("\n1)Qualifications");
	printf("\n2)Last 16");
	printf("\n3)Last 8");
	printf("\n4)Last 4");
	printf("\n5)Finals\n");
	r = multiChoice("Select the answer", options, 5);
```

```
	switch (r)
	{
	case '1':
		strcpy(tournament, "qualificazione");
		break;
	case '2':
		strcpy(tournament, "ottavi");
		break;
	case '3':
		strcpy(tournament, "quarti");
		break;
	case '4':
		strcpy(tournament, "semifinale");
		break;
	case '5':
		strcpy(tournament, "finale");
		break;
	default:
		fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
		abort();
	}

	if (!setup_prepared_stmt(&find_turn_matches_stmt, "call trova_sq_per_turno(?)", conn)) {
		finish_with_stmt_error(conn, find_turn_matches_stmt, "Unable to initialize matches search query
statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = tournament;
	param[0].buffer_length = strlen(tournament);

	if (mysql_stmt_bind_param(find_turn_matches_stmt, param) != 0) {
		finish_with_stmt_error(conn, find_turn_matches_stmt, "Could not bind parameters for matches search
query\n", true);
	}

	if (mysql_stmt_execute(find_turn_matches_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_turn_matches_stmt, "An error occurred while trying to find the player, what do you
want to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_turn_teams(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {



		dump_result_set(conn, find_turn_matches_stmt, "Matches Infos:\n");
		system("PAUSE");
```

```
		if (mysql_stmt_next_result(find_turn_matches_stmt) > 0) {
			finish_with_stmt_error(conn, find_turn_matches_stmt, "Unexpected contidion\n", true);
			system("PAUSE");
		}
	}

	mysql_stmt_close(find_turn_matches_stmt);
}

void find_goal_beast(MYSQL* conn)
{
	MYSQL_STMT* find_goal_beast_stmt;
	int status;


	if (!setup_prepared_stmt(&find_goal_beast_stmt, "call trova_capocannoniere()", conn)) {
		finish_with_stmt_error(conn, find_goal_beast, "Unable to initialize player search query statement\n", false);
	}




	if (mysql_stmt_execute(find_goal_beast_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_goal_beast_stmt, "An error occurred while trying to find the player, what do you want
to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_goal_beast(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {

		do {
				dump_result_set(conn, find_goal_beast_stmt, "");

			// more results? -1 = no, >0 = error, 0 = yes (keep looking)
			status = mysql_stmt_next_result(find_goal_beast_stmt);
			if (status > 0)
				finish_with_stmt_error(conn, find_goal_beast_stmt, "Unexpected condition", true);
		} while (status == 0);

	}
	system("pause");
	mysql_stmt_close(find_goal_beast_stmt);
}

void find_rich_customer(MYSQL* conn)
{
	MYSQL_STMT* find_rich_customer_stmt;


	if (!setup_prepared_stmt(&find_rich_customer_stmt, "call trova_cliente_ricco()", conn)) {
```

```
		finish_with_stmt_error(conn, find_goal_beast, "Unable to initialize customer search query statement\n",
false);
	}


	if (mysql_stmt_execute(find_rich_customer_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_rich_customer_stmt, "An error occurred while trying to find the customer, what do
you want to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_rich_customer(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {



		dump_result_set(conn, find_rich_customer_stmt, "Customer and bet infos:\n");
		system("PAUSE");
		if (mysql_stmt_next_result(find_rich_customer_stmt) > 0) {
			finish_with_stmt_error(conn, find_rich_customer_stmt, "Unexpected contidion\n", true);
			system("PAUSE");
		}
	}

	mysql_stmt_close(find_rich_customer_stmt);
}

void find_highest_goaln_match(MYSQL* conn)
{
	MYSQL_STMT* find_moregoals_stmt;
	int status;

	if (!setup_prepared_stmt(&find_moregoals_stmt, "call trova_piu_goal()", conn)) {
		finish_with_stmt_error(conn, find_goal_beast, "Unable to initialize match search query statement\n", false);
	}


	if (mysql_stmt_execute(find_moregoals_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(find_moregoals_stmt, "An error occurred while trying to find the match, what do you want
to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			find_highest_goaln_match(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
```

```
                        abort();
                }
        }
        else {

                do {
                        dump_result_set(conn, find_moregoals_stmt, "");

                        // more results? -1 = no, >0 = error, 0 = yes (keep looking)
                        status = mysql_stmt_next_result(find_moregoals_stmt);
                        if (status > 0)
                                finish_with_stmt_error(conn, find_moregoals_stmt, "Unexpected condition", true);
                } while (status == 0);

        }
        system("pause");
        mysql_stmt_close(find_moregoals_stmt);
}

void find_referee_presences(MYSQL* conn)
{
        MYSQL_STMT* find_referee_presences_stmt;
        MYSQL_BIND param[1];

        char referee[48];

        printf("\nReferee name: ");
        fflush(stdout);
        fgets(referee, 48, stdin);
        referee[strlen(referee) - 1] = '\0';

        if (!setup_prepared_stmt(&find_referee_presences_stmt, "call trova_presenze_arbitro(?)", conn)) {
                finish_with_stmt_error(conn, find_referee_presences_stmt, "Unable to initialize referee search query
statement\n", false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = referee;
        param[0].buffer_length = strlen(referee);

        if (mysql_stmt_bind_param(find_referee_presences_stmt, param) != 0) {
                finish_with_stmt_error(conn, find_referee_presences_stmt, "Could not bind parameters for referee search
query\n", true);
        }

        if (mysql_stmt_execute(find_referee_presences_stmt) != 0) {
                char optionserr[2] = { '1','2' };
                char er;
                print_stmt_error(find_referee_presences_stmt, "An error occurred while trying to find the referee, what do
you want to do?\n\t1)Try again;\n\t2)Quit\n");
                er = multiChoice("Select your answer: ", optionserr, 2);
                switch (er)
                {
                case '1':
                        find_referee_presences(conn);
                        break;
                case '2':
                        return;
                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
```

```
                }
        }
        else {


                printf("%s presences: \n", referee);
                dump_result_set(conn, find_referee_presences_stmt, "");
                system("PAUSE");
                if (mysql_stmt_next_result(find_referee_presences_stmt) > 0) {
                        finish_with_stmt_error(conn, find_referee_presences_stmt, "Unexpected contidion\n", true);
                        system("PAUSE");
                }
        }

        mysql_stmt_close(find_referee_presences_stmt);

}

void less_important_functions(MYSQL* conn)
{
        char options[5] = { '1','2','3','4','5' };
        char op;
        printf("\033[2J\033[H");
        printf("1) Find the player who scored more goals\n");
        printf("2) Find the customer with the highest bet for a match\n");
        printf("3) Find the match with the highest number of goals\n");
        printf("4) Find the number of presences of a referee\n");
        printf("5) Go back\n");

        op = multiChoice("Select an option", options, 5);

        switch (op)
        {
        case '1':
                find_goal_beast(conn);
                break;
        case '2':
                find_rich_customer(conn);
                break;
        case '3':
                find_highest_goaln_match(conn);
                break;
        case '4':
                find_referee_presences(conn);
                break;
        case '5':
                return;

        default:
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                abort();

        }

}

void important_funtions(MYSQL* conn)
{
        char options[9] = { '1','2','3','4','5','6','7','8','9' };
        char op;
        while (true) {
                printf("\033[2J\033[H");
                printf("*** What should I do for you? ***\n\n");
```

```
                printf("1) Add a customer\n");
                printf("2) Modify a customer data\n");
                printf("3) Add someones bet\n");
                printf("4) Find the match with the highest multiplier for a bet\n");
                printf("5) Find the matches not already played that someone can bet on\n");
                printf("6) Find the match with the highest numer of bets on it\n");
                printf("7) Find all the teams playing a game turn\n");
                printf("8) Other, less important stuff\n");
                printf("9) Quit\n");

                op = multiChoice("Select an option", options, 9);

                switch (op) {
                case '1':
                        add_customer(conn);
                        break;

                case '2':
                        update_customer(conn);
                        break;

                case '3':
                        add_bet(conn);
                        break;

                case '4':
                        find_highest_multiplier(conn);
                        break;

                case '5':
                        find_matches(conn);
                        break;

                case '6':
                        find_more_bets(conn);
                        break;

                case '7':
                        find_turn_teams(conn);
                        break;

                case '8':
                        less_important_functions(conn);
                        break;

                case '9':
                        return;

                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
                }
        }


}



void run_as_counter_clerk(MYSQL* conn)
{

        printf("Switching to counter clerk role...\n");
```

```
	if (!parse_config("users/addetto_sportelli.json", &conf)) {
		fprintf(stderr, "Unable to load counter clerk configuration\n");
		exit(EXIT_FAILURE);
	}

	if (mysql_change_user(conn, conf.db_username, conf.db_password, conf.database)) {
		fprintf(stderr, "mysql_change_user() failed\n");
		exit(EXIT_FAILURE);
	}

	important_funtions(conn);
}
```

administrator.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


#include "defines.h"

#pragma warning(disable : 4996)


void update_tournament(MYSQL* conn)
{
	MYSQL_STMT* update_tournament_stmt;
	MYSQL_BIND param[1];

	char options[3] = { '1','2','3'};
	char r;
	char tournament[20];


	printf("The system can calculate the teams of the next tournament stage\n but only if all the matches were already played.\n ");
	printf("Choose the tournament stage to calculate: \n\t1)Last 8\n\t2)Last 4\n\t3)Finals\n");
	r = multiChoice("Select the answer: ", options, 3);

	switch (r)
	{
	case '1':
		strcpy(tournament, "quarti");
		break;
	case '2':
		strcpy(tournament, "semifinale");
		break;
	case '3':
		strcpy(tournament, "finale");
		break;
	default:
		fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
		abort();

	}


	if (!setup_prepared_stmt(&update_tournament_stmt, "call aggiorna_tornero(?)", conn)) {
		finish_with_stmt_error(conn, update_tournament_stmt, "Unable to initialize the tournament update statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = tournament;
	param[0].buffer_length = strlen(tournament);


	if (mysql_stmt_bind_param(update_tournament_stmt, param) != 0) {
		finish_with_stmt_error(conn, update_tournament_stmt, "Could not bind parameters for tournament update procedure\n", true);
	}
	if (mysql_stmt_execute(update_tournament_stmt) != 0) {
		char optionserr[2] = { '1','2' };
```

```
                        char er;
                        print_stmt_error(update_tournament_stmt, "An error occurred while updating the tournament, what do you
want to do?\n\t1)Try again;\n\t2)Quit\n");
                        er = multiChoice("Select your answer: ", optionserr, 2);
                        switch (er)
                        {
                        case '1':
                                update_tournament(conn);
                                break;
                        case '2':
                                return;
                        default:
                                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                                abort();
                        }
                }
                else {
                        printf("\nTournament correctly updated\n");
                        fflush(stdout);
                        system("PAUSE");
                        mysql_stmt_close(update_tournament_stmt);
                }

}

void add_referee(MYSQL* conn)
{
        MYSQL_STMT* add_referee_stmt;
        MYSQL_BIND param[6];


        char name[48];
        char s_presences[16], s_number[16];
        char country[36];
        char city[48];
        char street[48];

        printf("\nName: ");
        fflush(stdout);
        fgets(name, 48, stdin);
        name[strlen(name) - 1] = '\0';

        printf("\nNumber of presences in past matches: ");
        fflush(stdout);
        fgets(s_presences, 16, stdin);

        printf("\nCountry: ");
        fflush(stdout);
        fgets(country, 36, stdin);
        country[strlen(country) - 1] = '\0';

        printf("\nCity: ");
        fflush(stdout);
        fgets(city, 48, stdin);
        city[strlen(city) - 1] = '\0';

        printf("\nStreet: ");
        fflush(stdout);
        fgets(street, 48, stdin);
        street[strlen(street) - 1] = '\0';

        printf("\nHouse number [>=0]: ");
        fflush(stdout);
```

```
	fgets(s_number, 16, stdin);

	int presences = atoi(s_presences);
	int number = atoi(s_number);


	if (!setup_prepared_stmt(&add_referee_stmt, "call aggiungi_arbitro(?, ?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, add_referee_stmt, "Unable to initialize referee insertion statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = name;
	param[0].buffer_length = strlen(name);

	param[1].buffer_type = MYSQL_TYPE_LONG;
	param[1].buffer = &presences;
	param[1].buffer_length = sizeof(presences);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[2].buffer = country;
	param[2].buffer_length = strlen(country);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[3].buffer = city;
	param[3].buffer_length = strlen(city);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = street;
	param[4].buffer_length = strlen(street);

	param[5].buffer_type = MYSQL_TYPE_LONG;
	param[5].buffer = &number;
	param[5].buffer_length = sizeof(number);

	if (mysql_stmt_bind_param(add_referee_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_referee_stmt, "Could not bind parameters for referee insertion\n", true);
	}
	if (mysql_stmt_execute(add_referee_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_referee_stmt, "An error occurred trying to add the referee, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_referee(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {
		printf("Referee correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(add_referee_stmt);
	}
```

```
}

void add_player(MYSQL* conn)
{
	MYSQL_STMT* add_player_stmt;
	MYSQL_BIND param[7];


	char name[48];
	char s_jersey[16], s_number[16];
	char country[36];
	char city[48];
	char street[48];
	char team[48];

	printf("\nName: ");
	fflush(stdout);
	fgets(name, 48, stdin);
	name[strlen(name) - 1] = '\0';

	printf("\nJersey number: ");
	fflush(stdout);
	fgets(s_jersey, 16, stdin);

	printf("\nCountry: ");
	fflush(stdout);
	fgets(country, 36, stdin);
	country[strlen(country) - 1] = '\0';

	printf("\nCity: ");
	fflush(stdout);
	fgets(city, 48, stdin);
	city[strlen(city) - 1] = '\0';

	printf("\nStreet: ");
	fflush(stdout);
	fgets(street, 48, stdin);
	street[strlen(street) - 1] = '\0';

	printf("\nHouse number [>=0]: ");
	fflush(stdout);
	fgets(s_number, 16, stdin);

	printf("\nTeam: ");
	fflush(stdout);
	fgets(team, 48, stdin);
	team[strlen(team) - 1] = '\0';

	int jersey = atoi(s_jersey);
	int number = atoi(s_number);


	if (!setup_prepared_stmt(&add_player_stmt, "call aggiungi_giocatore(?, ?, ?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, add_referee, "Unable to initialize player insertion statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = name;
	param[0].buffer_length = strlen(name);
```

```
	param[1].buffer_type = MYSQL_TYPE_LONG;
	param[1].buffer = &jersey;
	param[1].buffer_length = sizeof(jersey);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[2].buffer = country;
	param[2].buffer_length = strlen(country);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[3].buffer = city;
	param[3].buffer_length = strlen(city);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = street;
	param[4].buffer_length = strlen(street);

	param[5].buffer_type = MYSQL_TYPE_LONG;
	param[5].buffer = &number;
	param[5].buffer_length = sizeof(number);

	param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[6].buffer = team;
	param[6].buffer_length = strlen(team);

	if (mysql_stmt_bind_param(add_player_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_player_stmt, "Could not bind parameters for player insertion\n", true);
	}
	if (mysql_stmt_execute(add_player_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_player_stmt, "An error occurred while trying to add the player, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_player(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {
		printf("\nPlayer correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(add_player_stmt);
	}
}

void add_goal(MYSQL* conn)
{
	MYSQL_STMT* add_goal_stmt;
	MYSQL_BIND param[6];

	char options[5] = { '1','2','3','4','5'};
	char r;
```

```
char s_match[16], s_minute[16], s_r_minute[16];
char tournament[20];
signed char penalty;
char player[48];
int r_minute = 0;


printf("\nMatch number: ");
fflush(stdout);
fgets(s_match, 16, stdin);

printf("\nWich tournament stage this goal was scored?");
printf("\n1)Qualifications");
printf("\n2)Last 16");
printf("\n3)Last 8");
printf("\n4)Last 4");
printf("\n5)Finals\n");
r = multiChoice("Select the answer", options, 5);

switch (r)
{
case '1':
        strcpy(tournament, "qualificazione");
        break;
case '2':
        strcpy(tournament, "ottavi");
        break;
case '3':
        strcpy(tournament, "quarti");
        break;
case '4':
        strcpy(tournament, "semifinale");
        break;
case '5':
        strcpy(tournament, "finale");
        break;
default:
        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
        abort();
}

printf("\nPlayer name: ");
fflush(stdout);
fgets(player, 48, stdin);
player[strlen(player) - 1] = '\0';

printf("\nMinute of score, exluding injury time: ");
fflush(stdout);
fgets(s_minute, 16, stdin);

int match = atoi(s_match);
int minute = atoi(s_minute);
if (minute == 45 || minute == 90 || minute == 105 || minute == 120)
{
        printf("\nMinute of injury time when the goal was scored (starting from 0): ");
        fflush(stdout);
        fgets(s_r_minute, 16, stdin);
        r_minute = atoi(s_r_minute);
}

printf("The goal was scored on a penalty?\n\t1)Yes\n\t2)No\n ");
r = multiChoice("Select the answer: ", options, 2);
```

```
	switch (r)
	{
	case '1':
		penalty = 1;
		break;
	case '2':
		penalty = 0;
		break;
	default:
		fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
		abort();

	}

	if (!setup_prepared_stmt(&add_goal_stmt, "call aggiungi_goal(?, ?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, add_goal_stmt, "Unable to initialize goal insertion statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_LONG;
	param[0].buffer = &match;
	param[0].buffer_length = sizeof(match);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = tournament;
	param[1].buffer_length = strlen(tournament);

	param[2].buffer_type = MYSQL_TYPE_LONG;
	param[2].buffer = &minute;
	param[2].buffer_length = sizeof(minute);

	param[3].buffer_type = MYSQL_TYPE_TINY;
	param[3].buffer = &penalty;
	param[3].buffer_length = sizeof(penalty);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = player;
	param[4].buffer_length = strlen(player);

	param[5].buffer_type = MYSQL_TYPE_LONG;
	param[5].buffer = &r_minute;
	param[5].buffer_length = sizeof(r_minute);


	if (mysql_stmt_bind_param(add_goal_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_goal_stmt, "Could not bind parameters for goal insertion\n", true);
	}
	if (mysql_stmt_execute(add_goal_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_goal_stmt, "An error occurred while adding the goal, what do you want to do?\n\t1)Try
again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_goal(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
```

```
                        abort();
                }
        }
        else {
                printf("\nGoal correctly added\n");
                fflush(stdout);
                system("PAUSE");
                mysql_stmt_close(add_goal_stmt);
        }
}

void add_match_and_bets(MYSQL* conn)
{
        MYSQL_STMT* add_match_stmt;
        MYSQL_BIND param[10];


        char options[5] = { '1','2','3','4','5'};
        char r;



        char s_match[16];
        char tournament[20];
        char team1[48], team2[48];
        char referee[48];
        char city[48];
        char year[16], month[16], day[16], hour[16], minute[16];
        char v_mult1[16], v_mult2[16], v_multp[16];




        printf("\nMatch number: ");
        fflush(stdout);
        fgets(s_match, 16, stdin);

        printf("\nWich tournament stage this match is?");
        printf("\n1)Qualifications");
        printf("\n2)Last 16");
        printf("\n3)Last 8");
        printf("\n4)Last 4");
        printf("\n5)Finals\n");
        r = multiChoice("Select the answer", options, 5);

        switch (r)
        {
        case '1':
                strcpy(tournament, "qualificazione");
                break;
        case '2':
                strcpy(tournament, "ottavi");
                break;
        case '3':
                strcpy(tournament, "quarti");
                break;
        case '4':
                strcpy(tournament, "semifinale");
                break;
        case '5':
                strcpy(tournament, "finale");
                break;
        default:
```

```
            fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
            abort();
}


printf("\nTeam 1 name: ");
fflush(stdout);
fgets(team1, 48, stdin);
team1[strlen(team1) - 1] = '\0';

printf("\nTeam 2 name: ");
fflush(stdout);
fgets(team2, 48, stdin);
team2[strlen(team2) - 1] = '\0';

printf("\nReferee name: ");
fflush(stdout);
fgets(referee, 48, stdin);
referee[strlen(referee) - 1] = '\0';

printf("\nCity where the match is played: ");
fflush(stdout);
fgets(city, 48, stdin);
city[strlen(city) - 1] = '\0';


MYSQL_TIME date;
memset(&date, 0, sizeof(date));

printf("\nYear[Ex. 2022] of the match: ");
fflush(stdout);
fgets(year, 16, stdin);

date.year = atoi(year);

printf("\nMonth[1-12] of the match: ");
fflush(stdout);
fgets(month, 16, stdin);

date.month = atoi(month);
while (date.month < 1 || date.month > 12)
{
            printf("\nWrong data! Month[1-12] of the match:");
            fflush(stdout);
            fgets(month, 16, stdin);
            date.month = atoi(month);

}

printf("\nDay[1-31] of the match: ");
fflush(stdout);
fgets(day, 16, stdin);

date.day = atoi(day);
while (date.day < 1 || date.day > 31)
{
            printf("\nWrong data! Day[1-31] of the match: ");
            fflush(stdout);
            fgets(day, 16, stdin);
            date.day = atoi(day);

}
```

```
	printf("\nHour[0-23] of the match: ");
	fflush(stdout);
	fgets(hour, 16, stdin);

	date.hour = atoi(hour);
	while (date.hour < 0 || date.hour > 23)
	{
		printf("\nWrong data! Hour[0-23] of the match: ");
		fflush(stdout);
		fgets(hour, 16, stdin);
		date.hour = atoi(hour);

	}


	printf("\nMinute[0-59] of the match: ");
	fflush(stdout);
	fgets(minute, 16, stdin);
	date.minute = atoi(minute);
	while (date.minute < 0 || date.minute > 59)
	{
		printf("\nWrong data! Minute[0-59] of the match: ");
		fflush(stdout);
		fgets(minute, 16, stdin);
		date.minute = atoi(minute);

	}


	printf("\n In team 1 (%s) wins, what should be the bet multiplier?[>1]: ", team1);
	fflush(stdout);
	fgets(v_mult1, 16, stdin);

	printf("\n In team 2 (%s) wins, what should be the bet multiplier?[>1]: ", team2);
	fflush(stdout);
	fgets(v_mult2, 16, stdin);

	double multp;
	if (strcmp(tournament, "qualificazione") == 0)
	{
		printf("\n If noone wins, what should be the bet multiplier[>1]: ");
		fflush(stdout);
		fgets(v_multp, 16, stdin);
		multp = atof(v_multp);
	}

	double mult1 = atof(v_mult1);
	double mult2 = atof(v_mult2);

	int match = atoi(s_match);




	if (!setup_prepared_stmt(&add_match_stmt, "call aggiungi_partita_e_scommesse(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, add_match_stmt, "Unable to initialize match and bets insertion statement\n",
false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_LONG;
	param[0].buffer = &match;
```

```
	param[0].buffer_length = sizeof(match);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = tournament;
	param[1].buffer_length = strlen(tournament);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[2].buffer = team1;
	param[2].buffer_length = strlen(team1);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[3].buffer = team2;
	param[3].buffer_length = strlen(team2);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = referee;
	param[4].buffer_length = strlen(referee);

	param[5].buffer_type = MYSQL_TYPE_TIMESTAMP;
	param[5].buffer = &date;
	param[5].buffer_length = sizeof(date);

	param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[6].buffer = city;
	param[6].buffer_length = strlen(city);

	param[7].buffer_type = MYSQL_TYPE_DOUBLE;
	param[7].buffer = &mult1;
	param[7].buffer_length = sizeof(mult1);

	param[8].buffer_type = MYSQL_TYPE_DOUBLE;
	param[8].buffer = &mult2;
	param[8].buffer_length = sizeof(mult2);

	param[9].buffer_type = MYSQL_TYPE_DOUBLE;
	param[9].buffer = &multp;
	param[9].buffer_length = sizeof(multp);


	if (mysql_stmt_bind_param(add_match_stmt, param) != 0) {
		finish_with_stmt_error(conn, add_match_stmt, "Could not bind parameters for match and bets insertion\n",
true);
	}
	if (mysql_stmt_execute(add_match_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_match_stmt, "An error occurred while adding the match and/or the bets, what do you
want to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_match_and_bets(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {
		printf("\nMatch and bets correctly added\n");
```

```
                fflush(stdout);
                system("PAUSE");
                mysql_stmt_close(add_match_stmt);
        }


}

void add_team(MYSQL* conn)
{
        MYSQL_STMT* add_team_stmt;
        MYSQL_BIND param[2];


        char name[48];
        char coach[48];

        printf("\nName: ");
        fflush(stdout);
        fgets(name, 48, stdin);
        name[strlen(name) - 1] = '\0';

        printf("\nCoach name: ");
        fflush(stdout);
        fgets(coach, 48, stdin);
        coach[strlen(coach) - 1] = '\0';


        if (!setup_prepared_stmt(&add_team_stmt, "call aggiungi_squadra(?, ?)", conn)) {
                finish_with_stmt_error(conn, add_team_stmt, "Unable to initialize team insertion statement\n", false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = name;
        param[0].buffer_length = strlen(name);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = coach;
        param[1].buffer_length = strlen(coach);



        if (mysql_stmt_bind_param(add_team_stmt, param) != 0) {
                finish_with_stmt_error(conn, add_team_stmt, "Could not bind parameters for team insertion\n", true);
        }
        if (mysql_stmt_execute(add_team_stmt) != 0) {
                char optionserr[2] = { '1','2' };
                char er;
                print_stmt_error(add_team_stmt, "An error occurred while adding the team, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
                er = multiChoice("Select your answer: ", optionserr, 2);
                switch (er)
                {
                case '1':
                        add_team(conn);
                        break;
                case '2':
                        return;
                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
```

```
                }
        }
        else {
                printf("\nTeam correctly added\n");
                fflush(stdout);
                system("PAUSE");
                mysql_stmt_close(add_team_stmt);
        }
}

void add_to_qualifications(MYSQL* conn)
{
        MYSQL_STMT* add_team_qor16_stmt;
        MYSQL_BIND param[2];

        char options[2] = { '1','2' };
        char r;

        char tournament[20];
        char team[48];



        printf("\nTeam name: ");
        fflush(stdout);
        fgets(team, 48, stdin);
        team[strlen(team) - 1] = '\0';

        printf("You want to insert the team in qualifications or in last 16?\n\t1)Qualifications\n\t2)Last 16\n ");
        r = multiChoice("Select the answer: ", options, 2);

        switch (r)
        {
        case '1':
                strcpy(tournament, "qualificazione");
                break;
        case '2':
                strcpy(tournament, "ottavi");
                break;
        default:
                fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                abort();

        }

        if (!setup_prepared_stmt(&add_team_qor16_stmt, "call aggiungi_partecipazione(?, ?)", conn)) {
                finish_with_stmt_error(conn, add_team_qor16_stmt, "Unable to initialize the team insertion statement\n",
false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = tournament;
        param[0].buffer_length = strlen(tournament);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = team;
        param[1].buffer_length = strlen(team);



        if (mysql_stmt_bind_param(add_team_qor16_stmt, param) != 0) {
```

```
		finish_with_stmt_error(conn, add_team_qor16_stmt, "Could not bind parameters for team insertion\n", true);
	}
	if (mysql_stmt_execute(add_team_qor16_stmt) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(add_team_qor16_stmt, "An error occurred while trying to add the team, what do you want
to do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_to_qualifications(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
	}
	else {
		printf("\nTeam correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(add_team_qor16_stmt);
	}
}

void add_user(MYSQL* conn)
{
	MYSQL_STMT* create_user;
	MYSQL_BIND param[5];

	char options[2] = { '1','2' };
	char r;

	char name[48];
	char surname[48];
	char username[48];
	char password[48];
	char role[20];

	printf("\nName: ");
	fflush(stdout);
	fgets(name, 48, stdin);
	name[strlen(name) - 1] = '\0';

	printf("\nSurname: ");
	fflush(stdout);
	fgets(surname, 48, stdin);
	surname[strlen(surname) - 1] = '\0';

	printf("\nUsername: ");
	fflush(stdout);
	fgets(username, 48, stdin);
	username[strlen(username) - 1] = '\0';

	printf("\nPassword: ");
	fflush(stdout);
	insertPassword(password);

	printf("\nAssign the role:\n\t1)Amministratore\n\t2)Addetto sportelli");
	fflush(stdout);
```

```
	r = multiChoice("Select role: ", options, 2);
	switch (r)
	{
	case '1':
		strcpy(role, "amministratore");
		break;
	case '2':
		strcpy(role, "addetto_sportelli");
		break;
	default:
		fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
		abort();
	}

	if (!setup_prepared_stmt(&create_user, "call aggiungi_utente(?, ?, ?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, create_user, "Unable to initialize user insertion statement\n", false);
	}


	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = name;
	param[0].buffer_length = strlen(name);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = surname;
	param[1].buffer_length = strlen(surname);

	param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[2].buffer = username;
	param[2].buffer_length = strlen(username);

	param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[3].buffer = password;
	param[3].buffer_length = strlen(password);

	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = role;
	param[4].buffer_length = strlen(role);


	if (mysql_stmt_bind_param(create_user, param) != 0) {
		finish_with_stmt_error(conn, create_user, "Could not bind parameters for user insertion\n", true);
	}
	if (mysql_stmt_execute(create_user) != 0) {
		char optionserr[2] = { '1','2' };
		char er;
		print_stmt_error(create_user, "An error occurred while trying to add the user, what do you want to
do?\n\t1)Try again;\n\t2)Quit\n");
		er = multiChoice("Select your answer: ", optionserr, 2);
		switch (er)
		{
		case '1':
			add_user(conn);
			break;
		case '2':
			return;
		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
		}
```

```
	}
	else {
		printf("User correctly added\n");
		fflush(stdout);
		system("PAUSE");
		mysql_stmt_close(create_user);
	}

}

void run_as_administrator(MYSQL* conn)
{

	char options[9] = { '1','2','3','4','5','6','7','8','9'};
	char op;

	printf("Switching to counter administrator...\n");

	if (!parse_config("users/amministratore.json", &conf)) {
		fprintf(stderr, "Unable to load administrator configuration\n");
		exit(EXIT_FAILURE);
	}

	if (mysql_change_user(conn, conf.db_username, conf.db_password, conf.database)) {
		fprintf(stderr, "mysql_change_user() failed\n");
		exit(EXIT_FAILURE);
	}
	while (true) {
		printf("\033[2J\033[H");
		printf("1) Update the tournament\n");
		printf("2) Add a referee\n");
		printf("3) Add a player\n");
		printf("4) Add a goal\n");
		printf("5) Add a new match and its bets\n");
		printf("6) Add a team\n");
		printf("7) Add a team to the qualifications or last 16\n");
		printf("8) Add a new user\n");
		printf("9) Quit\n");

		op = multiChoice("Select an option", options, 9);

		switch (op)
		{
		case '1':
			update_tournament(conn);
			break;
		case '2':
			add_referee(conn);
			break;
		case '3':
			add_player(conn);
			break;
		case '4':
			add_goal(conn);
			break;
		case '5':
			add_match_and_bets(conn);
			break;
		case '6':
			add_team(conn);
			break;
		case '7':
			add_to_qualifications(conn);
```

```
                        break;
                case '8':
                        add_user(conn);
                        break;
                case '9':
                        return;

                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();

                }
        }

}
```

defines.h

```
#pragma once

#include <stdbool.h>
#include <mysql.h>

struct configuration {
	char *host;
	char *db_username;
	char *db_password;
	unsigned int port;
	char *database;

	char username[128];
	char password[128];
};

struct param_type {
	char* varchar[7];
	MYSQL_TIME* date[2];
	MYSQL_TIME* time[2];
	int* integer[3];
	float* f;
	short int* b;
	int numVarchar;
	int numDate;
	int numTime;
	int numInt;
	int numFloat;
	int numBool;
};

extern struct configuration conf;
extern struct param_type paramStruct;

extern int parse_config(char *path, struct configuration *conf);
extern bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive);
extern char multiChoice(char *domanda, char choices[], int num);
extern void insertPassword(char password[]);
extern void print_error (MYSQL *conn, char *message);
extern void print_stmt_error (MYSQL_STMT *stmt, char *message);
extern void finish_with_error(MYSQL *conn, char *message);
extern void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message, bool close_stmt);
extern bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn);
extern void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title);
extern char* randstring(size_t length);
extern void run_as_counter_clerk(MYSQL *conn);
extern void run_as_administrator(MYSQL *conn);
```

inout.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <stdbool.h>

#include "defines.h"

// Per la gestione dei segnali
static volatile sig_atomic_t signo;
typedef struct sigaction sigaction_t;
static void handler(int s);



bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive)
{

	// I caratteri 'yes' e 'no' devono essere minuscoli
	yes = tolower(yes);
	no = tolower(no);

	// Decide quale delle due lettere mostrare come predefinite
	char s, n;
	if(predef) {
		s = toupper(yes);
		n = no;
	} else {
		s = yes;
		n = toupper(no);
	}

	// Richiesta della risposta
	while(true) {
		// Mostra la domanda
		printf("%s [%c/%c]: ", domanda, s, n);

		char c;
		fgets(&c, 2, stdin);

		// Controlla quale risposta è stata data
		if(c == '\0') { // getInput() non può restituire '\n'!
			return predef;
		} else if(c == yes) {
			return true;
		} else if(c == no) {
			return false;
		} else if(c == toupper(yes)) {
			if(predef || insensitive) return true;
		} else if(c == toupper(yes)) {
			if(!predef || insensitive) return false;
		}
	}
}

char multiChoice(char *domanda, char choices[], int num)
{
	char inputStr[3];
```

```
	// Genera la stringa delle possibilità
	char *possib = malloc(2 * num * sizeof(char));
	int i, j = 0;
	for(i = 0; i < num; i++) {
		possib[j++] = choices[i];
		possib[j++] = '/';
	}
	possib[j-1] = '\0'; // Per eliminare l'ultima '/'

	// Chiede la risposta
	while(true) {
		// Mostra la domanda
		printf("%s [%s]: ", domanda, possib);

		fgets(inputStr, 3, stdin);
		char c = inputStr[0];

		// Controlla se è un carattere valido
		for(i = 0; i < num; i++) {
			if(c == choices[i])
				return c;
		}
	}
}

void insertPassword(char password[]) {
	int ch;
	int i = 0;
	while ((ch = _getch()) != EOF && ch != '\n' && ch != '\r' && i < sizeof(password) - 1) {
		if (ch == '\b' && i > 0) {
			printf("\b \b");		// Destructive backspace
			fflush(stdout);
			i--;
			password[i] = '\0';
		}
		else if (ch != '\b') {
			putchar('*');
			password[i++] = (char)ch;
		}
	}
	password[i] = '\0';

	printf("\n");
	fflush(stdout);
}
```

parse.c

```
#pragma warning(disable : 4996)
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

#define BUFF_SIZE 4096

// The final config struct will point into this
static char config[BUFF_SIZE];

/**
 * JSON type identifier. Basic types are:
 *	o Object
 *	o Array
 *	o String
 *	o Other primitive: number, boolean (true/false) or null
 */
typedef enum {
	JSMN_UNDEFINED = 0,
	JSMN_OBJECT = 1,
	JSMN_ARRAY = 2,
	JSMN_STRING = 3,
	JSMN_PRIMITIVE = 4
} jsmntype_t;

enum jsmnerr {
	/* Not enough tokens were provided */
	JSMN_ERROR_NOMEM = -1,
	/* Invalid character inside JSON string */
	JSMN_ERROR_INVAL = -2,
	/* The string is not a full JSON packet, more bytes expected */
	JSMN_ERROR_PART = -3
};

/**
 * JSON token description.
 * type		type (object, array, string etc.)
 * start	start position in JSON data string
 * end		end position in JSON data string
 */
typedef struct {
	jsmntype_t type;
	int start;
	int end;
	int size;
#ifdef JSMN_PARENT_LINKS
	int parent;
#endif
} jsmntok_t;

/**
 * JSON parser. Contains an array of token blocks available. Also stores
 * the string being parsed now and current position in that string
 */
typedef struct {
	unsigned int pos; /* offset in the JSON string */
	unsigned int toknext; /* next token to allocate */
	int toksuper; /* superior token node, e.g parent object or array */
```

```
} jsmn_parser;

/**
 * Allocates a fresh unused token from the token pool.
 */
static jsmntok_t *jsmn_alloc_token(jsmn_parser *parser, jsmntok_t *tokens, size_t num_tokens) {
	jsmntok_t *tok;
	if (parser->toknext >= num_tokens) {
		return NULL;
	}
	tok = &tokens[parser->toknext++];
	tok->start = tok->end = -1;
	tok->size = 0;
#ifdef JSMN_PARENT_LINKS
	tok->parent = -1;
#endif
	return tok;
}

/**
 * Fills token type and boundaries.
 */
static void jsmn_fill_token(jsmntok_t *token, jsmntype_t type,
		int start, int end) {
	token->type = type;
	token->start = start;
	token->end = end;
	token->size = 0;
}

/**
 * Fills next available token with JSON primitive.
 */
static int jsmn_parse_primitive(jsmn_parser *parser, const char *js,
		size_t len, jsmntok_t *tokens, size_t num_tokens) {
	jsmntok_t *token;
	int start;

	start = parser->pos;

	for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
		switch (js[parser->pos]) {
#ifndef JSMN_STRICT
			/* In strict mode primitive must be followed by "," or "}" or "]" */
			case ':':
#endif
			case '\t' : case '\r' : case '\n' : case ' ' :
			case ','  : case ']'  : case '}' :
				goto found;
		}
		if (js[parser->pos] < 32 || js[parser->pos] >= 127) {
			parser->pos = start;
			return JSMN_ERROR_INVAL;
		}
	}
#ifdef JSMN_STRICT
	/* In strict mode primitive must be followed by a comma/object/array */
	parser->pos = start;
	return JSMN_ERROR_PART;
#endif

found:
	if (tokens == NULL) {
```

```
                parser->pos--;
                return 0;
        }
        token = jsmn_alloc_token(parser, tokens, num_tokens);
        if (token == NULL) {
                parser->pos = start;
                return JSMN_ERROR_NOMEM;
        }
        jsmn_fill_token(token, JSMN_PRIMITIVE, start, parser->pos);
#ifdef JSMN_PARENT_LINKS
        token->parent = parser->toksuper;
#endif
        parser->pos--;
        return 0;
}

/**
 * Fills next token with JSON string.
 */
static int jsmn_parse_string(jsmn_parser *parser, const char *js,
                size_t len, jsmntok_t *tokens, size_t num_tokens) {
        jsmntok_t *token;

        int start = parser->pos;

        parser->pos++;

        /* Skip starting quote */
        for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
                char c = js[parser->pos];

                /* Quote: end of string */
                if (c == '\"') {
                        if (tokens == NULL) {
                                return 0;
                        }
                        token = jsmn_alloc_token(parser, tokens, num_tokens);
                        if (token == NULL) {
                                parser->pos = start;
                                return JSMN_ERROR_NOMEM;
                        }
                        jsmn_fill_token(token, JSMN_STRING, start+1, parser->pos);
#ifdef JSMN_PARENT_LINKS
                        token->parent = parser->toksuper;
#endif
                        return 0;
                }

                /* Backslash: Quoted symbol expected */
                if (c == '\\' && parser->pos + 1 < len) {
                        int i;
                        parser->pos++;
                        switch (js[parser->pos]) {
                                /* Allowed escaped symbols */
                                case '\"': case '/' : case '\\' : case 'b' :
                                case 'f' : case 'r' : case 'n'  : case 't' :
                                        break;
                                /* Allows escaped symbol \uXXXX */
                                case 'u':
                                        parser->pos++;
                                        for(i = 0; i < 4 && parser->pos < len && js[parser->pos] != '\0'; i++) {
                                                /* If it isn't a hex character we have an error */
                                                if(!((js[parser->pos] >= 48 && js[parser->pos] <= 57) || /* 0-9 */
```

```
                                                                (js[parser->pos] >= 65 && js[parser->pos]
<= 70) || /* A-F */
                                                                (js[parser->pos] >= 97 && js[parser->pos]
<= 102))) { /* a-f */
                                                                        parser->pos = start;
                                                                        return JSMN_ERROR_INVAL;
                                                                }
                                                                parser->pos++;
                                                        }
                                                        parser->pos--;
                                                        break;
                                                /* Unexpected symbol */
                                                default:
                                                        parser->pos = start;
                                                        return JSMN_ERROR_INVAL;
                                        }
                                }
                        }
                        parser->pos = start;
                        return JSMN_ERROR_PART;
}

/**
 * Parse JSON string and fill tokens.
 */
static int jsmn_parse(jsmn_parser *parser, const char *js, size_t len, jsmntok_t *tokens, unsigned int num_tokens) {
                        int r;
                        int i;
                        jsmntok_t *token;
                        int count = parser->toknext;

                        for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
                                        char c;
                                        jsmntype_t type;

                                        c = js[parser->pos];
                                        switch (c) {
                                                        case '{': case '[':
                                                                        count++;
                                                                        if (tokens == NULL) {
                                                                                        break;
                                                                        }
                                                                        token = jsmn_alloc_token(parser, tokens, num_tokens);
                                                                        if (token == NULL)
                                                                                        return JSMN_ERROR_NOMEM;
                                                                        if (parser->toksuper != -1) {
                                                                                        tokens[parser->toksuper].size++;
#ifdef JSMN_PARENT_LINKS
                                                                                        token->parent = parser->toksuper;
#endif
                                                                        }
                                                                        token->type = (c == '{' ? JSMN_OBJECT : JSMN_ARRAY);
                                                                        token->start = parser->pos;
                                                                        parser->toksuper = parser->toknext - 1;
                                                                        break;
                                                        case '}': case ']':
                                                                        if (tokens == NULL)
                                                                                        break;
                                                                        type = (c == '}' ? JSMN_OBJECT : JSMN_ARRAY);
#ifdef JSMN_PARENT_LINKS
                                                                        if (parser->toknext < 1) {
                                                                                        return JSMN_ERROR_INVAL;
                                                                        }
```

```
                                token = &tokens[parser->toknext - 1];
                                for (;;) {
                                        if (token->start != -1 && token->end == -1) {
                                                if (token->type != type) {
                                                        return JSMN_ERROR_INVAL;
                                                }
                                                token->end = parser->pos + 1;
                                                parser->toksuper = token->parent;
                                                break;
                                        }
                                        if (token->parent == -1) {
                                                if(token->type != type || parser->toksuper == -1) {
                                                        return JSMN_ERROR_INVAL;
                                                }
                                                break;
                                        }
                                        token = &tokens[token->parent];
                                }
#else
                                for (i = parser->toknext - 1; i >= 0; i--) {
                                        token = &tokens[i];
                                        if (token->start != -1 && token->end == -1) {
                                                if (token->type != type) {
                                                        return JSMN_ERROR_INVAL;
                                                }
                                                parser->toksuper = -1;
                                                token->end = parser->pos + 1;
                                                break;
                                        }
                                }
                                /* Error if unmatched closing bracket */
                                if (i == -1) return JSMN_ERROR_INVAL;
                                for (; i >= 0; i--) {
                                        token = &tokens[i];
                                        if (token->start != -1 && token->end == -1) {
                                                parser->toksuper = i;
                                                break;
                                        }
                                }
#endif
                                break;
                        case '\"':
                                r = jsmn_parse_string(parser, js, len, tokens, num_tokens);
                                if (r < 0) return r;
                                count++;
                                if (parser->toksuper != -1 && tokens != NULL)
                                        tokens[parser->toksuper].size++;
                                break;
                        case '\t' : case '\r' : case '\n' : case ' ':
                                break;
                        case ':':
                                parser->toksuper = parser->toknext - 1;
                                break;
                        case ',':
                                if (tokens != NULL && parser->toksuper != -1 &&
                                                tokens[parser->toksuper].type != JSMN_ARRAY &&
                                                tokens[parser->toksuper].type != JSMN_OBJECT) {
#ifdef JSMN_PARENT_LINKS
                                        parser->toksuper = tokens[parser->toksuper].parent;
#else
                                        for (i = parser->toknext - 1; i >= 0; i--) {
                                                if (tokens[i].type == JSMN_ARRAY || tokens[i].type ==
JSMN_OBJECT) {
```

```
                                                                if (tokens[i].start != -1 && tokens[i].end == -1) {
                                                                        parser->toksuper = i;
                                                                        break;
                                                                }
                                                        }
                                                }
#endif
                                        }
                                        break;
#ifdef JSMN_STRICT
                                /* In strict mode primitives are: numbers and booleans */
                                case '-': case '0': case '1' : case '2': case '3' : case '4':
                                case '5': case '6': case '7' : case '8': case '9':
                                case 't': case 'f': case 'n' :
                                        /* And they must not be keys of the object */
                                        if (tokens != NULL && parser->toksuper != -1) {
                                                jsmntok_t *t = &tokens[parser->toksuper];
                                                if (t->type == JSMN_OBJECT ||
                                                                (t->type == JSMN_STRING && t->size != 0)) {
                                                        return JSMN_ERROR_INVAL;
                                                }
                                        }
#else
                                /* In non-strict mode every unquoted value is a primitive */
                                default:
#endif
                                        r = jsmn_parse_primitive(parser, js, len, tokens, num_tokens);
                                        if (r < 0) return r;
                                        count++;
                                        if (parser->toksuper != -1 && tokens != NULL)
                                                tokens[parser->toksuper].size++;
                                        break;

#ifdef JSMN_STRICT
                                /* Unexpected char in strict mode */
                                default:
                                        return JSMN_ERROR_INVAL;
#endif
                        }
                }

                if (tokens != NULL) {
                        for (i = parser->toknext - 1; i >= 0; i--) {
                                /* Unmatched opened object or array */
                                if (tokens[i].start != -1 && tokens[i].end == -1) {
                                        return JSMN_ERROR_PART;
                                }
                        }
                }

                return count;
}

/**
 * Creates a new parser based over a given  buffer with an array of tokens
 * available.
 */
static void jsmn_init(jsmn_parser *parser) {
                parser->pos = 0;
                parser->toknext = 0;
                parser->toksuper = -1;
}
```

```
static int jsoneq(const char *json, jsmntok_t *tok, const char *s)
{
	if (tok->type == JSMN_STRING
	   && (int) strlen(s) == tok->end - tok->start
	   && strncmp(json + tok->start, s, tok->end - tok->start) == 0) {
		return 0;
	}
	return -1;
}

static size_t load_file(char *filename)
{
	FILE *f = fopen(filename, "rb");
	if(f == NULL) {
		fprintf(stderr, "Unable to open file %s\n", filename);
		exit(1);
	}

	fseek(f, 0, SEEK_END);
	size_t fsize = ftell(f);
	fseek(f, 0, SEEK_SET);  //same as rewind(f);

	if(fsize >= BUFF_SIZE) {
		fprintf(stderr, "Configuration file too large\n");
		abort();
	}

	fread(config, fsize, 1, f);
	fclose(f);

	config[fsize] = 0;
	return fsize;
}

char* strndup(char* str, int chars)
{
	char* buffer;
	int n;

	buffer = (char*)malloc(chars + 1);
	if (buffer)
	{
		for (n = 0; ((n < chars) && (str[n] != 0)); n++) buffer[n] = str[n];
		buffer[n] = 0;
	}

	return buffer;
}

int parse_config(char *path, struct configuration *conf)
{
	int i;
	int r;
	jsmn_parser p;
	jsmntok_t t[128]; /* We expect no more than 128 tokens */

	load_file(path);

	jsmn_init(&p);
	r = jsmn_parse(&p, config, strlen(config), t, sizeof(t)/sizeof(t[0]));
	if (r < 0) {
		printf("Failed to parse JSON: %d\n", r);
		return 0;
```

```
	}

	/* Assume the top-level element is an object */
	if (r < 1 || t[0].type != JSMN_OBJECT) {
		printf("Object expected\n");
		return 0;
	}

	/* Loop over all keys of the root object */
	for (i = 1; i < r; i++) {
		if (jsoneq(config, &t[i], "host") == 0) {
			/* We may use strndup() to fetch string value */
			conf->host = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "username") == 0) {
			conf->db_username = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "password") == 0) {
			conf->db_password = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "port") == 0) {
			conf->port = strtol(config + t[i+1].start, NULL, 10);
			i++;
		} else if (jsoneq(config, &t[i], "database") == 0) {
			conf->database = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else {
			printf("Unexpected key: %.*s\n", t[i].end-t[i].start, config + t[i].start);
		}
	}
	return 1;
}
```

utils.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

void print_stmt_error (MYSQL_STMT *stmt, char *message)
{

	if (stmt != NULL) {
		fprintf (stderr, "Error %u (%s): %s\n",
			mysql_stmt_errno (stmt),
			mysql_stmt_sqlstate(stmt),
			mysql_stmt_error (stmt));
	}
	fprintf(stderr, "%s\n", message);
	fflush(stdout);
}


void print_error(MYSQL *conn, char *message)
{
	fprintf (stderr, "%s\n", message);
	if (conn != NULL) {
		#if MYSQL_VERSION_ID >= 40101
		fprintf (stderr, "Error %u (%s): %s\n",
		mysql_errno (conn), mysql_sqlstate(conn), mysql_error (conn));
		#else
		fprintf (stderr, "Error %u: %s\n",
		mysql_errno (conn), mysql_error (conn));
		#endif
	}
}

bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn)
{
	my_bool update_length = true;

	*stmt = mysql_stmt_init(conn);
	if (*stmt == NULL)
	{
		print_error(conn, "Could not initialize statement handler");
		return false;
	}

	if (mysql_stmt_prepare (*stmt, statement, strlen(statement)) != 0) {
		print_stmt_error(*stmt, "Could not prepare statement");
		return false;
	}

	mysql_stmt_attr_set(*stmt, STMT_ATTR_UPDATE_MAX_LENGTH, &update_length);

	return true;
}

void finish_with_error(MYSQL *conn, char *message)
{
	print_error(conn, message);
	mysql_close(conn);
	exit(EXIT_FAILURE);
```

```
}

void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message, bool close_stmt)
{
	print_stmt_error(stmt, message);
	if(close_stmt)	mysql_stmt_close(stmt);
	mysql_close(conn);
	exit(EXIT_FAILURE);
}

static void print_dashes(MYSQL_RES *res_set)
{
	MYSQL_FIELD *field;
	unsigned int i, j;

	mysql_field_seek(res_set, 0);
	putchar('+');
	for (i = 0; i < mysql_num_fields(res_set); i++) {
		field = mysql_fetch_field(res_set);
		if (field->type == MYSQL_TYPE_DOUBLE)
		{
			for (j = 0; j < 20; j++)
				putchar('-');
		}
		else
		{
			for (j = 0; j < field->max_length + 2; j++)
				putchar('-');
		}
		putchar('+');
	}
	putchar('\n');
}

static void dump_result_set_header(MYSQL_RES *res_set)
{
	MYSQL_FIELD *field;
	unsigned long col_len;
	unsigned int i;

	/* determine column display widths -- requires result set to be */
	/* generated with mysql_store_result(), not mysql_use_result() */

	mysql_field_seek (res_set, 0);

		for (i = 0; i < mysql_num_fields(res_set); i++) {
			field = mysql_fetch_field(res_set);
			if (field->type == MYSQL_TYPE_DOUBLE)
			{
				col_len = 28;

			}
			else
			{
				col_len = strlen(field->name);

				if (col_len < field->max_length)
					col_len = field->max_length;
				if (col_len < 4 && !IS_NOT_NULL(field->flags))
					col_len = 4; /* 4 = length of the word "NULL" */
				field->max_length = col_len; /* reset column info */
			}
		}
```

```
	print_dashes(res_set);
	putchar('|');
	mysql_field_seek (res_set, 0);
	for (i = 0; i < mysql_num_fields(res_set); i++) {
		field = mysql_fetch_field(res_set);
		if (field->type == MYSQL_TYPE_DOUBLE)
		{
			printf(" %-*s  |", 28-strlen(field->name), field->name);
		}
		else
		{
			printf(" %-*s |", (int)field->max_length, field->name);
		}
	}
	putchar('\n');

	print_dashes(res_set);
}

void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title)
{
	int i;
	int status;
	int num_fields;       /* number of columns in result */
	MYSQL_FIELD *fields;  /* for result set metadata */
	MYSQL_BIND *rs_bind;  /* for output buffers */
	MYSQL_RES *rs_metadata;
	MYSQL_TIME *date;
	size_t attr_size;

	/* Prefetch the whole result set. This in conjunction with
	 * STMT_ATTR_UPDATE_MAX_LENGTH set in `setup_prepared_stmt`
	 * updates the result set metadata which are fetched in this
	 * function, to allow to compute the actual max length of
	 * the columns.
	 */
	if (mysql_stmt_store_result(stmt)) {
		fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
		fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
		exit(0);
	}

	/* the column count is > 0 if there is a result set */
	/* 0 if the result is only the final status packet */
	num_fields = mysql_stmt_field_count(stmt);

	if (num_fields > 0) {
		/* there is a result set to fetch */
		printf("%s\n", title);

		if((rs_metadata = mysql_stmt_result_metadata(stmt)) == NULL) {
			finish_with_stmt_error(conn, stmt, "Unable to retrieve result metadata\n", true);
		}

		dump_result_set_header(rs_metadata);

		fields = mysql_fetch_fields(rs_metadata);

		rs_bind = (MYSQL_BIND *)malloc(sizeof (MYSQL_BIND) * num_fields);
```

```
if (!rs_bind) {
	finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n", true);
}
memset(rs_bind, 0, sizeof (MYSQL_BIND) * num_fields);
/* set up and bind result set output buffers */
for (i = 0; i < num_fields; ++i) {
	// Properly size the parameter buffer
	switch(fields[i].type) {
		case MYSQL_TYPE_DATE:
		case MYSQL_TYPE_TIMESTAMP:
		case MYSQL_TYPE_DATETIME:
		case MYSQL_TYPE_TIME:
			attr_size = sizeof(MYSQL_TIME);
			break;
		case MYSQL_TYPE_FLOAT:
			attr_size = sizeof(float);
			break;
		case MYSQL_TYPE_DOUBLE:
			attr_size = sizeof(double);
			break;
		case MYSQL_TYPE_TINY:
			attr_size = sizeof(signed char);
			break;
		case MYSQL_TYPE_SHORT:
		case MYSQL_TYPE_YEAR:
			attr_size = sizeof(short int);
			break;
		case MYSQL_TYPE_LONG:
		case MYSQL_TYPE_INT24:
			attr_size = sizeof(int);
			break;
		case MYSQL_TYPE_LONGLONG:
			attr_size = sizeof(int);
			break;
		default:
			attr_size = fields[i].max_length;
			break;
	}

	// Setup the binding for the current parameter
	rs_bind[i].buffer_type = fields[i].type;
	rs_bind[i].buffer = malloc(attr_size + 1);
	rs_bind[i].buffer_length = attr_size + 1;

	if(rs_bind[i].buffer == NULL) {
		finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n", true);
	}
}

if(mysql_stmt_bind_result(stmt, rs_bind)) {
	finish_with_stmt_error(conn, stmt, "Unable to bind output parameters\n", true);
}

/* fetch and display result set rows */
while (true) {
	status = mysql_stmt_fetch(stmt);

	if (status == 1 || status == MYSQL_NO_DATA)
		break;

	putchar('|');

	for (i = 0; i < num_fields; i++) {
```

```
                                if (rs_bind[i].is_null_value) {
                                        printf (" %-*s |", (int)fields[i].max_length, "NULL");
                                        continue;
                                }

                                switch (rs_bind[i].buffer_type) {

                                        case MYSQL_TYPE_VAR_STRING:
                                        case MYSQL_TYPE_DATETIME:
                                                printf(" %-*s |", (int)fields[i].max_length, (char*)rs_bind[i].buffer);
                                                break;

                                        case MYSQL_TYPE_DATE:
                                        case MYSQL_TYPE_TIMESTAMP:
                                                date = (MYSQL_TIME *)rs_bind[i].buffer;
                                                printf(" %d-%02d-%02d %02d:%02d                |", date->year, date-
>month, date->day, date->hour, date->minute);
                                                break;

                                        case MYSQL_TYPE_STRING:
                                                printf(" %-*s |", (int)fields[i].max_length, (char *)rs_bind[i].buffer);
                                                break;

                                        case MYSQL_TYPE_FLOAT:
                                        case MYSQL_TYPE_DOUBLE:
                                                printf(" %.02f            |", *(double *)rs_bind[i].buffer);
                                                break;

                                        case MYSQL_TYPE_LONG:
                                        case MYSQL_TYPE_SHORT:
                                        case MYSQL_TYPE_TINY:
                                                printf(" %-*d |", (int)fields[i].max_length, *(int *)rs_bind[i].buffer);
                                                break;

                                        case MYSQL_TYPE_NEWDECIMAL:
                                                printf(" %-*.02lf |", (int)fields[i].max_length, *(float*)
rs_bind[i].buffer);
                                                break;

                                        default:
                                           printf("ERROR: Unhandled type (%d)\n", rs_bind[i].buffer_type);
                                           abort();
                                }
                        }
                        putchar('\n');
                        print_dashes(rs_metadata);
                }

                mysql_free_result(rs_metadata); /* free metadata */

                /* free output buffers */
                for (i = 0; i < num_fields; i++) {
                        free(rs_bind[i].buffer);
                }
                free(rs_bind);
        }
}


char* randstring(size_t length) {

        static char charset[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,.-#'?!";
```

```
	char* randomString = NULL;

	if (length) {
		randomString = malloc(sizeof(char) * (length + 1));

		if (randomString) {
			for (int n = 0;n < length;n++) {
				int key = rand() % (int)(sizeof(charset) - 1);
				randomString[n] = charset[key];
			}

			randomString[length] = '\0';
		}
	}

	return randomString;
}
```