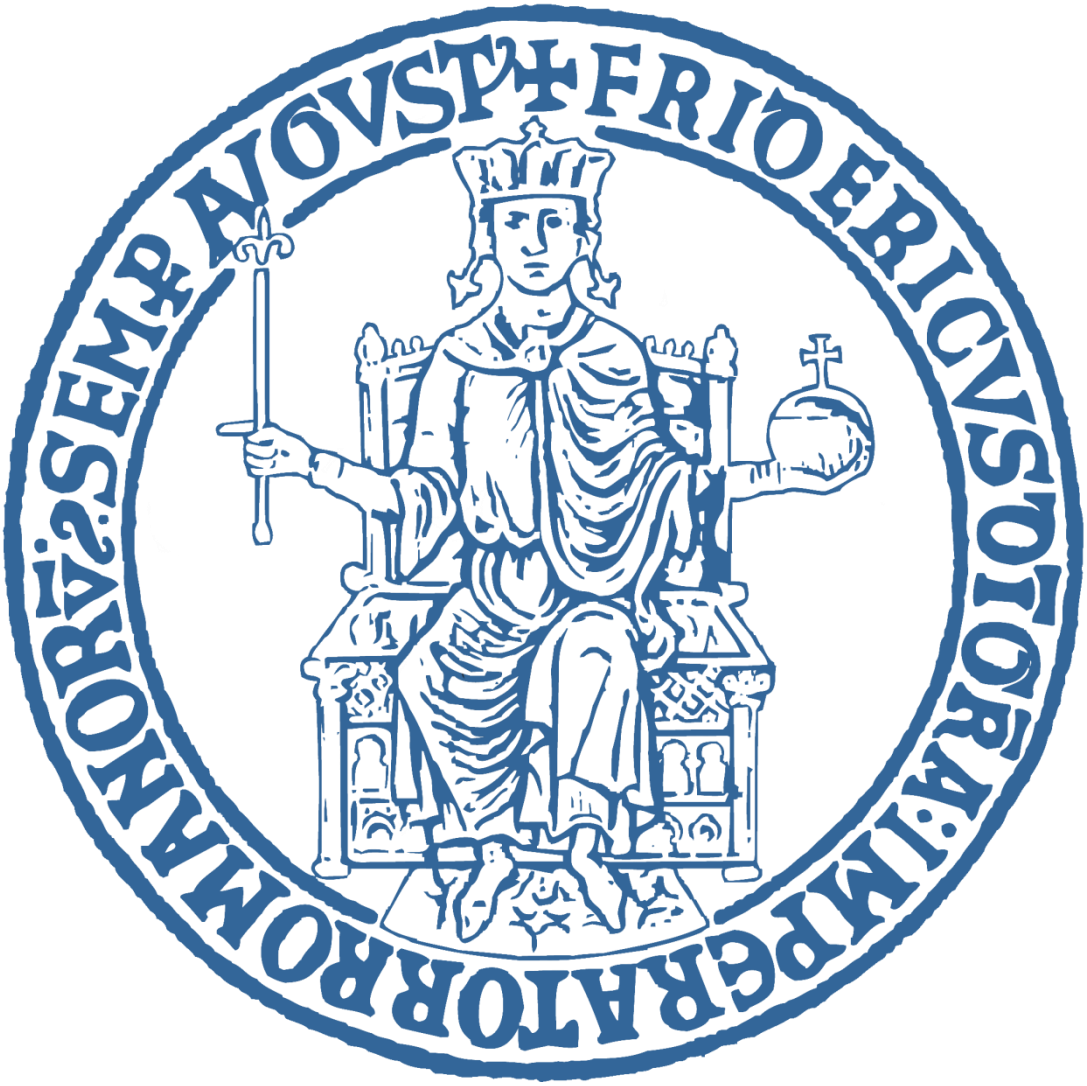
UNIVERSITA' DEGLI STUDI FEDERICO II DI NAPOLI

Facoltà di Scienze Informatiche

BASI DI DATI E
SISTEMI
INFORMATIVI

Prof. A.Peron



STUDENTI:

Grazioso Marco N86/391

mar.grazioso@studenti.unina.it

Michele Fattoruso N86/299

mi.fattoruso@studenti.unina.it

Indice:

1 Descrizione del problema

1.1 Problema

Si scriva un database per la memorizzazione di partite giocate secondo le regole del noto gioco Monopoli. Il database deve contenere tutte le informazioni necessarie per ripercorrere la partita passo dopo passo in tutti i suoi dettagli. Non fa parte del progetto lo sviluppo di un'interfaccia grafica che consenta lo svolgimento o la simulazione del gioco.

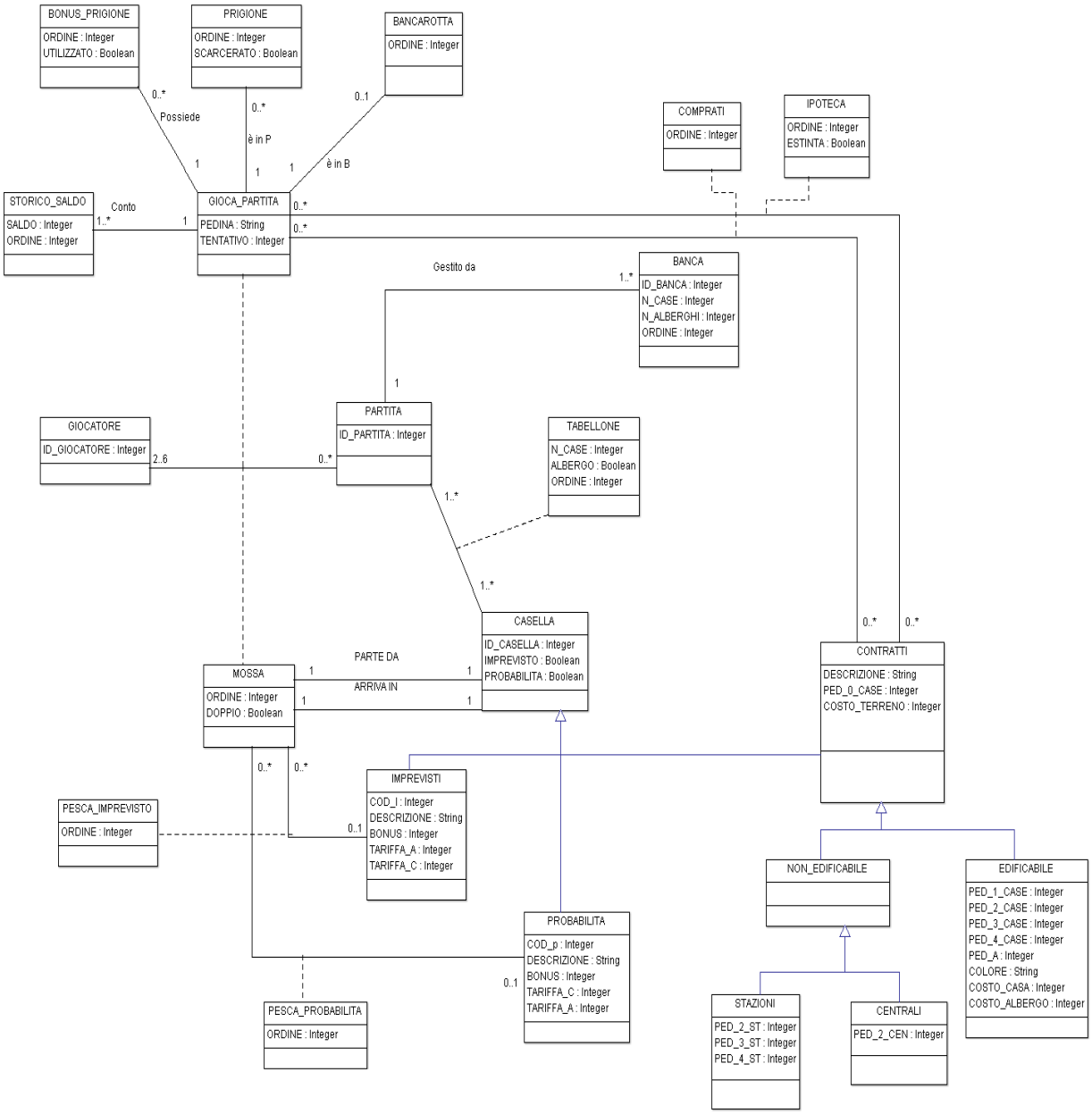
1.2 Analisi del problema

È richiesta, per la risoluzione del problema, la progettazione di una base di dati che sia in grado di gestire e memorizzare partite giocate secondo le regole del monopoli. Il gioco prevede l'avanzamento di un giocatore, rappresentato da una pedina, mediate il lancio di due dadi, che rappresenteranno il numero di passi da compiere lungo il percorso definito dal tabellone. Il tabellone sarà composto da caselle ed avrà una lunghezza fissa pari a quaranta caselle. Ogni casella apparterrà ad una particolare tipologia, che determinerà degli eventi legati alla mossa e le possibili azioni che il giocatore potrà compiere. Le tipologie si suddividono in: imprevisti, probabilità, terreni edificabili e non. Ogni giocatore avrà un saldo iniziale che subirà incrementi o decrementi a seconda dello svolgimento del gioco. La dipartita di un giocatore sarà decretata dall'esaurimento del proprio saldo e delle sue proprietà. Il vincitore sarà decretato quando il penultimo giocatore lascerà la partita per fallimento. Per la realizzazione della base di dati è stato utilizzato un attributo "ordine" che ci permetterà di gestire e recuperare tutti gli eventi legati ad ogni singolo turno (spostamento , modifica del saldo, compravendita di terreni ecc…).

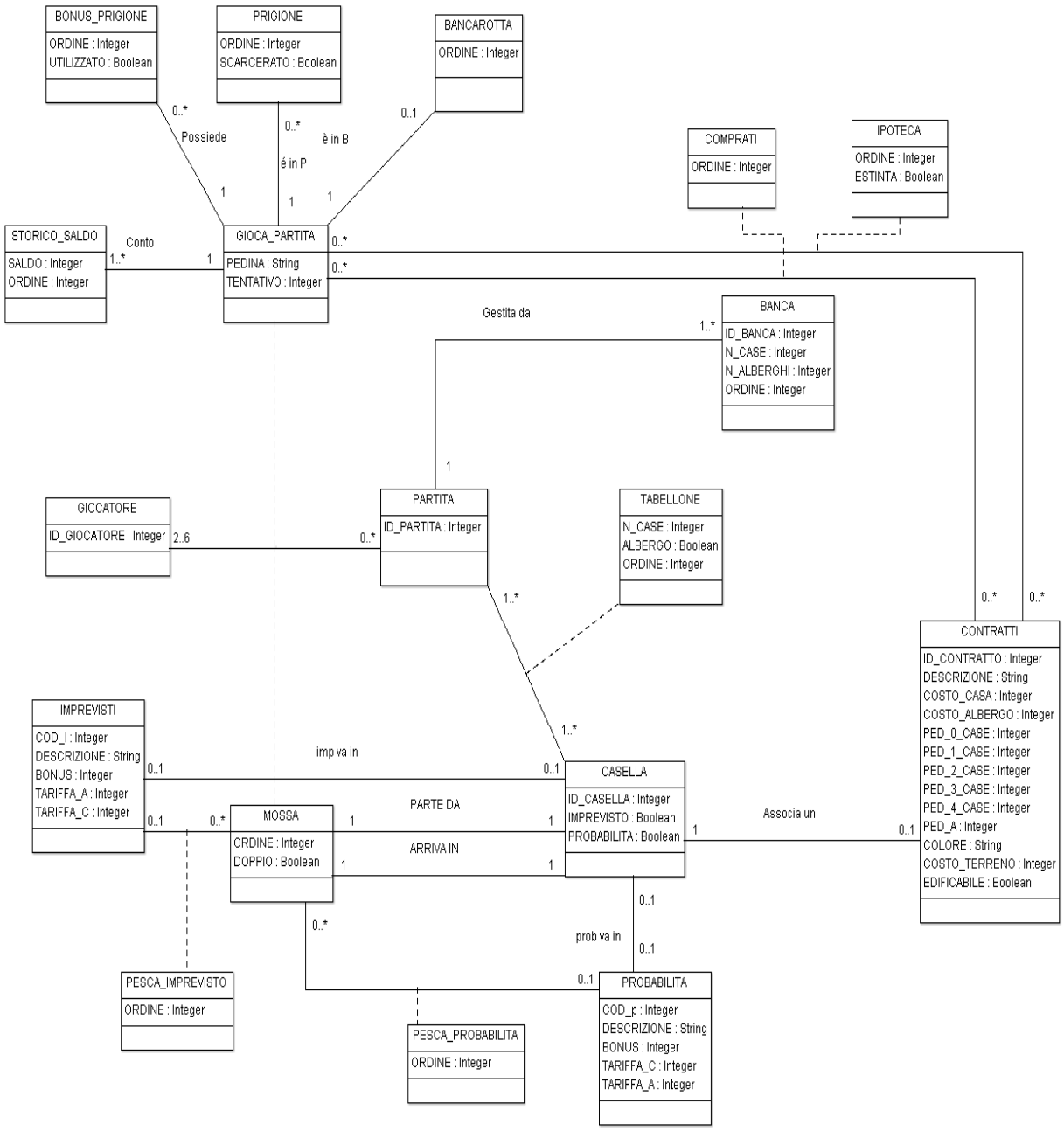
Durante la realizzazione del database si è reso necessario aggiungere degli attributi che permettessero di effettuare determinati controlli atti a preservare la correttezza di alcuni trigger e la coerenza della base di dati.

2 Progettazione Concettuale

2.1 Class Diagram



2.2 Ristrutturazione del Class Diagram



2.3 Dizionario delle Classi

Entità	Descrizione	Attributi	Tipo	Cardinalità	Identificatore
Partita	Identifica la partita giocata	ID_PARTITA ID_BANCA	Integer Integer	1 1	ID_PARTITA
Giocatore	Identifica il giocatore che sta giocando la partita	ID_GIOCATORE	Integer	1	ID_GIOCATORE
Casella	Identifica il tipo di casella ed il contratto ad essa associato	ID_CASELLA IMPREVISTO PROBABILITA CONTRATTO	Integer Boolean Boolean integer	1 1 1 (0..1)	ID_CASELLA
Banca	Identifica la banca ed il numero di case ed alberghi rimanenti	ID_BANCA N_CASE N_ALBERGHI ORDINE	Integer Integer Integer Integer	1 1 1 1	ID_BANCA
Imprevisti	Identifica un cartellino di imprevisto e ne descrive gli effetti causati	COD_I DESCRIZIONE BONUS CAS_A TARIFFA_A TARIFFA_C	Integer String Integer Integer Integer Integer	1 1 (0..1) (0..1) (0..1) (0..1)	COD_I
Probabilita	Identifica un cartellino di probabilità e ne descrive gli effetti causati	COD_P DESCRIZIONE BONUS CAS_A TARIFFA_A TARIFFA_C	Integer String Integer Integer Integer Integer	1 1 (0..1) (0..1) (0..1) (0..1)	COD_P
Contratti	Identifica un contratto associato ad una casella e ne descrive dettagliata detta tutti i particolari	ID_CONTRATTO CASELLA DESCRIZIONE COSTO_CASA COSTO_ALBERGO PED_0_CASE PED_1_CASE PED_2_CASE PED_3_CASE PED_4_CASE PED_A COLORE COSTO_TERRENO EDIFICABILE	Integer Integer String Integer Integer Integer Integer Integer Integer Integer String Integer Boolean Boolean	1 1 1 (0..1) (0..1) 1 1 (0..1) (0..1) (0..1) (0..1) (0..1) 1 1	ID_CONTRATTO
Storico_saldo	Identifica il	GIOCATORE	Integer	1	GIOCATORE

	saldo associato ad un giocatore per una determinata partita	PARTITA ORDINE SALDO	Integer Integer Integer	1 1 1	PARTITA ORDINE
Prigione	Indica se un giocatore,in una determinata mossa è andato in prigione	PARTITA ORDINE SCARCERATO	Integer Integer Boolean	1 1 1	PARTITA ORDINE
Bancarotta	Identifica i giocatori che,dopo aver esaurito il proprio saldo hanno terminato il gioco	PARTITA ORDINE GIOCATORE	Integer Integer Integer	1 1 1	PARTITA ORDINE
Bonus_Prigione	Identifica il possessore di un cartellino di Imprevisto/ Probabilita che permette l'uscita gratuita di prigione	GIOCATORE PARTITA ORDINE UTILIZZATO	Integer Integer Integer Boolean	1 1 1 1	PARTITA ORDINE GIOCATORE

2.4 Dizionario delle Associazioni

Associazione	Descrizione	Classi	Attributi
Mossa	Rappresenta la mossa effettuata dal giocatore	Giocatore Partita	ID_PARTITA ID_GIOCATORE ORDINE CAS_A CAS_P DOPPIO IMPRIGIONATO
Comprati	Identifica i contratti comprati da un giocatere	Gioca_partita Contratti	ID_GIOCATORE ID_PARTITA CONTRATTO ORDINE
Ipoteca	Identifica i contratti ipotecati da un giocatore	Gioca_partita Contratti	GIOCATORE PARTITA CONTRATTO ORDINE ESTINTA
Gioca_Partita	Indica quali giocatori stanno giocando una determinata partita	Giocatore Partita	PARTITA GIOCATORE PEDINA TENTATIVI
Tabellone	Rappresenta la composizione di una plancia di gioco con le relative strutture costruite dai giocatori	Partita Casella	PARTITA CASELLA N_CASE ALBERGO ORDINE GIOCATORE
Pesca_imprevisto	Identifica quale imprevisto il giocatore abbia pescato in seguito ad una mossa	Mossa Imprevisti	PARTITA ORDINE IMPREVISTO
Pesca_probabilita	Identifica quale probabilità il giocatore abbia pescato in seguito ad una mossa	Mossa Probabilita	PARTITA ORDINE PROBABILITA
Parte da	Rappresenta la casella di partenza di una mossa	Mossa Casella	/
Arriva in	Rappresenta la casella di arrivo si una mossa	Mossa Casella	/
Gestita da	Indica quale banca gestisce la situazione economica dei giocatori	Partita Banca	/
Possiede	Indica se un giocatore in una determinata partita possiede un cartellino per uscire gratis di prigione	Gioca_partita Bonus_prigione	/
È in B	Indica se un giocatore	Gioca_partita	/

	in una determinata partita è in bancarotta	Bancarotta	
È in P	Indica se un giocatore in una determinata partita è in Prigione	Gioca_partita Prigione	/
Conto	Indica il saldo di un giocatore in una determinata partita	Gioca_partita Storico_saldo	/
Imp va in	Indica se la casella è una casella imprevisto	Casella Imprevisti	/
Prob va in	Indica se la casella è una casella probabilità	Casella Probabilita	/
Associa un	Indica che quella casella è associata ad un contratto	Casella Contratti	/

2.5 Elenco dei Vincoli

L'operazione successiva all'individuazione delle classi (e dei relativi attributi) e delle associazioni,è l'individuazione dei vincoli.Lo scopo di tale operazione è quello di rendere la base di dati il più robusta possibile.oltre ai vincoli di chiave primaria e di chiave esterna , che saranno imprelentati nella fase di creazione delle tabelle in SQL sarà implementato il seguente insieme di vincoli

1) Una partita non può essere giocata da più di sei giocatori
2) In una casella non possono essere inserite più di quattro case
3) In una casella non può essere inserito più di un albergo
4) Un giocatore non può costruire case su un territorio se non possiede tutti i contratti di quel colore
5) La costruzione delle case deve avvenire in modo proporzionato sui territori dello stesso colore
6) La costruzione delle case deve avvenire soltanto se la banca possiede un numero sufficiente di case
7) La costruzione di un albergo può avvenire soltanto se la banca possiede un numero sufficiente di alberghi
8) Deve essere rispettato un ordine crescente nell'inserimento di "ordine" in mossa
9) In una partita ogni giocatore deve avere pedine differenti
10) Ogni giocatore deve possedere una delle pedine messe a disposizione dal gioco
11) Il colore del contratto deve essere uguale ad uno dei colori messi a disposizione dal gioco
12) Quando un giocatore si ferma sulla casella "in prigione" deve essere immediatamente imprigionato
13) Se un giocatore non è in grado di sostenere una spesa (uscita di prigione,pagamento tariffa ad un giocatore avversario ecc…) con il proprio saldo è obbligato ad ipotecare o vendere qualcosa di sua proprietà.nel caso in cui il giocatore non riesca lo stesso a sostenere la spesa il giocatore sarà considerato in bancarotta

14) se un giocatore ipoteca un contratto,la banca valuterà quel contratto la metà del prezzo del cartellino.nel caso in cui il giocatore vorrà riscattare l'ipoteca ,dovrà pagare alla banca il 10% in più rispetto al costo di cartellino
15) su un territorio è possibile costruire un albergo solo se su di esso vi sono gia costruite quattro case
16) se un giocatore vende un albergo sul territorio dovranno essere nuovamente posizionate quattro case,se la banca conferma la loro disponibilità
17) bisogna controllare che ogni spostamento effettuato da un giocatore sia lecito
18) ogni volta che un giocatore passa dal via deve riscuotere 500 euro dalla banca
19) se un giocatore si ferma su un territorio già posseduto da un giocatore,dovrà versare all'avversario una somma di denaro ,somma che varia in base alla situazione del territorio(numero di case,numero di alberghi ecc..)
20) un giocatore che ha dichiarato bancarotta,non può più partecipare al gioco
21) un giocatore può uscire gratuitamente di prigione usando un cartellino "esci gratis di prigione" oppure effettuando un tiro doppio,se fallisce il tiro doppio per tre turni sarà costretto a pagare una multa di 125 euro
22) un giocatore può uscire in qualsiasi momento di prigione pagando una multa di 125 euro
23) se un giocatore dichiara bancarotta la banca venderà tutte le sue proprietà
24) se un giocatore fa un tiro doppio il giocatore ha diritto ad effettuare un secondo lancio di dadi

3 Progettazione Logica

3.1 Passaggio dal Class Diagram al Modello Relazionale Dei Dati

Il modello relazionale è un modello logico di rappresentazione dei dati implementato su sistemi di gestione di basi di dati. Esso si basa sull'algebra relazionale e sulla teoria degli insiemi ed è strutturato attorno al concetto di relazione, in base al quale, tutte le informazioni devono essere rappresentate da valori inseriti in relazioni (cioè in tabelle).Esso viene costruito partendo dal Class Diagram e valutando tutte le considerazioni fatte nel paragrafo precedente. di ciascuna entità del diagramma, La prima operazione da effettuare, dunque, è la codifica di ciascuna entità del diagramma

PARTITA(ID_PARTITA,ID_BANCA)
GIOCATORE(ID_GIOCATORE)
CASELLA(ID_CASELLA,IMPREVISTO,PROBABILITA,CONTRATTO)
BANCA(ID_BANCA,N_CASE,N_ALBERGHI,ORDINE)
IMPREVISTI(COD_I,DESCRIZIONE,BONUS,CAS_A,TARIFFA_A,TARIFFA_C)
PROBABILITA(COD_P, DESCRIZIONE,BONUS,CAS_A,TARIFFA_A,TARIFFA_C)
CONTRATTI(ID_CONTRATTO,CASELLA,DESCRIZIONE,COSTO_CASA,COSTO_ALBERGO,PED_0_CASE,PED_1_CASE,PED_2_CASE,PED_3_CASE,PED_4_CASE,PED_A,COLORE,COSTO_TERRENO,EDIFICAFILE
STORICO_SALDO(GIOCATORE,PARTITA,ORDINE,SALDO)
PRIGIONE(PARTITA,ORDINE,SCARCERATO)
BANCAROTTA(PARTITA,ORDINE,GIOCATORE)
BONUS_PRIGIONE(GIOCATORE,PARTITA,ORDINE,UTILIZZATO)

Una volta stabilita la codifica delle entità, all' interno del nostro schema relazionale, si dovranno codificare le associazioni.

MOSSA(ID_PARTITA,ID_GIOCATORE,ORDINE,CAS_A,CAS_P,DOPPIO)
COMPRATI(ID_GIOCATORE,CONTRATTO,ID_PARTITA,ORDINE)
IPOTECA(GIOCATORE,PARTITA,CONTRATTO,ORDINE,ESTINTA)
GIOCA_PARTITA(PARTITA,GIOCATORE,PEDINA,TENTATIVI)
TABELLONE(PARTITA,CASELLA,N_CASE,ALBERGO,ORDINE,GIOCATORE)
PESCA_IMPREVISTO(PARTITA,ORDINE,IMPREVISTO,GIOCATORE)
PESCA_PROBABILITA(PARTITA,ORDINE,PROBABILITA,GIOCATORE)

3.2 Struttuta logica del Database

Dopo aver codificato dettagliatamente ogni associazione,la struttura della base di dati sarà la seguente:

PARTITA(ID_PARTITA,ID_BANCA)
GIOCATORE(ID_GIOCATORE)
CASELLA(ID_CASELLA,IMPREVISTO,PROBABILITA,CONTRATTO)
BANCA(ID_BANCA,N_CASE,N_ALBERGHI,ORDINE)
IMPREVISTI(COD_I,DESCRIZIONE,BONUS,CAS_A,TARIFFA_A,TARIFFA_C)
PROBABILITA(COD_P, DESCRIZIONE,BONUS,CAS_A,TARIFFA_A,TARIFFA_C)
CONTRATTI(ID_CONTRATTO,CASELLA,DESCRIZIONE,COSTO_CASA,COSTO_ALBERGO,PED_0_CASE,PED_1_CASE,PED_2_CASE,PED_3_CASE,PED_4_CASE,PED_A,COLORE,COSTO_TERRENO,EDIFICAFILE)
STORICO_SALDO(GIOCATORE,PARTITA,ORDINE,SALDO)
PRIGIONE(PARTITA,ORDINE,SCARCERATO)
BANCAROTTA(PARTITA,ORDINE,GIOCATORE)
BONUS_PRIGIONE(GIOCATORE,PARTITA,ORDINE,UTILIZZATO)
MOSSA(ID_PARTITA,ID_GIOCATORE,ORDINE,CAS_A,CAS_P,DOPPIO,IMPRIGIONATO)
COMPRATI(ID_GIOCATORE,CONTRATTO,ID_PARTITA,ORDINE)
IPOTECA(GIOCATORE,PARTITA,CONTRATTO,ORDINE,ESTINTA)
GIOCA_PARTITA(PARTITA,GIOCATORE,PEDINA,TENTATIVI)
TABELLONE(PARTITA,CASELLA,N_CASE,ALBERGO,ORDINE,GIOCATORE)
PESCA_IMPREVISTO(PARTITA,ORDINE,IMPREVISTO,GIOCATORE)
PESCA_PROBABILITA(PARTITA,ORDINE,PROBABILITA,GIOCATORE)

Il passo successivo per la progettazionedi un Database è la codifica di ciascuno schema relazionale in una tabella definita dallo standard SQL

3.3 Implementazione delle Tabelle

Tabella Partita

```
CREATE TABLE  "PARTITA"
   (   "ID_PARTITA" NUMBER,
       "ID_BANCA" NUMBER NOT NULL ENABLE,
        CONSTRAINT "PARTITA_PK" PRIMARY KEY ("ID_PARTITA") ENABLE,
        CONSTRAINT "PARTITA_UNQ" UNIQUE ("ID_BANCA") ENABLE
   )
/
```

Tabella giocatore

```
CREATE TABLE  "GIOCATORE"
   (   "ID_GIOCATORE" NUMBER,
        CONSTRAINT "GIOCATORE_PK" PRIMARY KEY ("ID_GIOCATORE") ENABLE
   )
/
```

Tabella Casella

```
CREATE TABLE  "CASELLA"
   (   "ID_CASELLA" NUMBER(2,0),
       "IMPREVISTO" CHAR(1) NOT NULL DISABLE,
       "PROBABILITA" CHAR(1) NOT NULL DISABLE,
       "CONTRATTO" NUMBER,
        CONSTRAINT "CASELLA_PK" PRIMARY KEY ("ID_CASELLA") ENABLE,
        CONSTRAINT "IMPREVISTO_CK" CHECK (imprevisto = 'Y' or imprevisto = 'N') ENABLE,
        CONSTRAINT "PROBABILITA_CK" CHECK (probabilita = 'Y' or probabilita = 'N') ENABLE,
        CONSTRAINT "CASELLA_FK" FOREIGN KEY ("CONTRATTO")
         REFERENCES  "CONTRATTI" ("ID_CONTRATTO") ENABLE
   )
/
```

Tabella Banca

```
CREATE TABLE  "BANCA"
   (   "ID_BANCA" NUMBER,
       "N_CASE" NUMBER(2,0) DEFAULT (32) NOT NULL ENABLE,
       "N_ALBERGHI" NUMBER(2,0) DEFAULT (12) NOT NULL ENABLE,
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "BANCA_CK" CHECK ( "ORDINE" >= 0) ENABLE,
        CONSTRAINT "BANCA_PK" PRIMARY KEY ("ID_BANCA", "ORDINE") ENABLE
   )
/
```

Tabella Imprevisti

```
CREATE TABLE  "IMPREVISTI"
   (   "COD_I" NUMBER,
       "DESCRIZIONE" VARCHAR2(4000),
       "BONUS" NUMBER,
       "CAS_A" NUMBER,
       "TARIFFA_A" NUMBER,
       "TARIFFA_C" NUMBER,
        CONSTRAINT "IMPREVISTI_PK" PRIMARY KEY ("COD_I") ENABLE
   )
/
```

Tabella Probabilità

```
CREATE TABLE  "PROBABILITA"
   (   "COD_P" NUMBER,
       "DESCRIZIONE" VARCHAR2(4000),
       "BONUS" NUMBER,
       "CAS_A" NUMBER,
       "TARIFFA_A" NUMBER,
       "TARIFFA_C" NUMBER,
        CONSTRAINT "PROBABILITA_PK" PRIMARY KEY ("COD_P") ENABLE
   )
/
```

Tabella Contratti

```
CREATE TABLE  "CONTRATTI"
   (   "ID_CONTRATTO" NUMBER,
       "CASELLA" NUMBER NOT NULL ENABLE,
       "DESCRIZIONE" VARCHAR2(2000) NOT NULL ENABLE,
       "COSTO_CASA" NUMBER,
       "COSTO_ALBERGO" NUMBER,
       "PED_1_CASE" NUMBER NOT NULL ENABLE,
       "PED_2_CASE" NUMBER,
       "PED_3_CASE" NUMBER,
       "PED_4_CASE" NUMBER,
       "PED_0_CASE" NUMBER NOT NULL ENABLE,
       "PED_ALBERGO" NUMBER,
       "COSTO_TERRENO" NUMBER NOT NULL ENABLE,
       "EDIFICABILE" CHAR(1) NOT NULL ENABLE,
```

```
       "COLORE" VARCHAR2(100),
        CONSTRAINT "CONTRATTI_CK1" CHECK (edificabile = 'Y' or edificabile='N') ENABLE,
        CONSTRAINT "CONTRATTI_PK" PRIMARY KEY ("ID_CONTRATTO") ENABLE,
        CONSTRAINT "CONTRATTI_CK3" CHECK ( "COLORE" = 'ROSSO' OR COLORE = 'GIALLO' OR COLORE =
'VERDE' OR COLORE = 'BLU' OR COLORE = 'MARRONE' OR COLORE = 'MAGENTA' OR COLORE = 'VIOLA' OR
COLORE = 'ARANCIONE'
) ENABLE,
        CONSTRAINT "CONTRATTI_FK" FOREIGN KEY ("CASELLA")
         REFERENCES  "CASELLA" ("ID_CASELLA") ENABLE
   )
/
```

Tabella Storico_Saldo

```
CREATE TABLE  "STORICO_SALDO"
   (   "GIOCATORE" NUMBER,
       "PARTITA" NUMBER,
       "SALDO" NUMBER(10,0) NOT NULL ENABLE,
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "STORICO_SALDO_CK1" CHECK (SALDO >= 0) ENABLE,
        CONSTRAINT "STORICO_SALDO_CK2" CHECK (ORDINE >= 0) ENABLE,
        CONSTRAINT "STORICO_SALDO_PK" PRIMARY KEY ("GIOCATORE", "PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "STORICO_SALDO_FK" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "STORICO_SALDO_FK2" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Prigione

```
CREATE TABLE  "PRIGIONE"
   (   "PARTITA" NUMBER NOT NULL ENABLE,
       "ORDINE" NUMBER(38,0),
       "SCARCERATO" CHAR(1) NOT NULL ENABLE,
       "GIOCATORE" NUMBER NOT NULL ENABLE,
        CONSTRAINT "PRIGIONE_CK1" CHECK (SCARCERATO = 'Y' OR SCARCERATO = 'N') ENABLE,
        CONSTRAINT "PRIGIONE_CK2" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "PRIGIONE_PK" PRIMARY KEY ("PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "PRIGIONE_FK" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Bancarotta

```
CREATE TABLE  "BANCAROTTA"
   (   "PARTITA" NUMBER(38,0) NOT NULL ENABLE,
       "GIOCATORE" NUMBER(38,0) NOT NULL ENABLE,
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "BANCAROTTA_CK1" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "BANCAROTTA_PK" PRIMARY KEY ("PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "BANCAROTTA_FK2" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "BANCAROTTA_FK" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Bonus_Prigione

```
CREATE TABLE  "BONUS_PRIGIONE"
   (   "GIOCATORE" NUMBER,
       "PARTITA" NUMBER,
       "ORDINE" NUMBER,
       "UTILIZZATO" CHAR(1),
        CONSTRAINT "BONUS_PRIGIONE_CK1" CHECK (UTILIZZATO = 'Y' OR UTILIZZATO = 'N') ENABLE,
        CONSTRAINT "BONUS_PRIGIONE_CK2" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "BONUS_PRIGIONE_PK" PRIMARY KEY ("GIOCATORE", "PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "BONUS_PRIGIONE_FK" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
```

```
        CONSTRAINT "BONUS_PRIGIONE_FK2" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Mossa

```
CREATE TABLE  "MOSSA"
   (   "ID_PARTITA" NUMBER,
       "ID_GIOCATORE" NUMBER NOT NULL ENABLE,
       "ORDINE" NUMBER(10,0),
       "CAS_A" NUMBER NOT NULL ENABLE,
       "CAS_P" NUMBER NOT NULL ENABLE,
       "DOPPIO" CHAR(1),
       "IMPRIGIONATO" CHAR(1),
        CONSTRAINT  "CK_IMPRIGIONATO"  CHECK  (  "IMPRIGIONATO"  =  'Y'  OR  IMPRIGIONATO  =  'N')
ENABLE,
        CONSTRAINT "MOSSA_CK1" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "MOSSA_CK2" CHECK ( "DOPPIO" = 'Y' OR DOPPIO = 'N') ENABLE,
        CONSTRAINT "MOSSA_PK" PRIMARY KEY ("ID_PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "MOSSA_FK3" FOREIGN KEY ("CAS_A")
         REFERENCES  "CASELLA" ("ID_CASELLA") ENABLE,
        CONSTRAINT "MOSSA_FK4" FOREIGN KEY ("CAS_P")
         REFERENCES  "CASELLA" ("ID_CASELLA") ENABLE,
        CONSTRAINT "MOSSA_FK2" FOREIGN KEY ("ID_GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "MOSSA_FK" FOREIGN KEY ("ID_PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Comprati

```
CREATE TABLE  "COMPRATI"
   (   "ID_PARTITA" NUMBER,
       "CONTRATTO" NUMBER,
       "ID_GIOCATORE" NUMBER,
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "COMPRATI_CK1" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT  "COMPRATI_PK"  PRIMARY  KEY  ("ID_PARTITA",  "CONTRATTO",  "ID_GIOCATORE",
"ORDINE") ENABLE,
        CONSTRAINT "COMPRATI_FK2" FOREIGN KEY ("CONTRATTO")
         REFERENCES  "CONTRATTI" ("ID_CONTRATTO") ENABLE,
        CONSTRAINT "COMPRATI_FK3" FOREIGN KEY ("ID_GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "COMPRATI_FK" FOREIGN KEY ("ID_PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Ipoteca

```
CREATE TABLE  "IPOTECA"
   (   "GIOCATORE" NUMBER NOT NULL ENABLE,
       "CONTRATTO" NUMBER NOT NULL ENABLE,
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
       "PARTITA" NUMBER NOT NULL ENABLE,
       "ESTINTA" CHAR(1) NOT NULL ENABLE,
        CONSTRAINT "IPOTECA_CK" CHECK ( "ORDINE" > 0) ENABLE,
        CONSTRAINT "IPOTECA_CK2" CHECK ( "ESTINTA" = 'Y' OR ESTINTA = 'N') ENABLE,
        CONSTRAINT  "IPOTECA_PK"  PRIMARY  KEY  ("PARTITA",  "GIOCATORE",  "CONTRATTO",  "ORDINE")
ENABLE,
        CONSTRAINT "IPOTECA_FK" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "IPOTECA_FK3" FOREIGN KEY ("CONTRATTO")
         REFERENCES  "CONTRATTI" ("ID_CONTRATTO") ENABLE,
        CONSTRAINT "IPOTECA_FK4" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Gioca_partita

```
CREATE TABLE  "GIOCA_PARTITA"
   (   "GIOCATORE" NUMBER,
       "PARTITA" NUMBER,
       "PEDINA" VARCHAR2(100) NOT NULL ENABLE,
       "TENTATIVI" NUMBER(1,0),
        CONSTRAINT "GIOCA_PARTITA_CK2" CHECK ( "TENTATIVI" >=0 AND TENTATIVI <=3) ENABLE,
        CONSTRAINT "GIOCA_PARTITA_CK_PEDINA" CHECK ( "PEDINA" = 'FUNGO' OR PEDINA = 'CANDELA'
OR PEDINA = 'BOTTE' OR PEDINA = 'ZUCCA' OR PEDINA = 'FIASCO' OR PEDINA = 'PIANTA') ENABLE,
        CONSTRAINT "GIOCA_PARTITA_PK" PRIMARY KEY ("GIOCATORE", "PARTITA") ENABLE,
        CONSTRAINT "GIOCA_PARTITA_FK" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "GIOCA_PARTITA_CON" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Tabellone

```
CREATE TABLE  "TABELLONE"
   (   "PARTITA" NUMBER,
       "CASELLA" NUMBER,
       "N_CASE" NUMBER(1,0),
       "ALBERGO" CHAR(1),
       "ORDINE" NUMBER(38,0) NOT NULL ENABLE,
       "GIOCATORE" NUMBER(38,0),
        CONSTRAINT "ALBERGO_CK2" CHECK (ALBERGO = 'Y' OR ALBERGO ='N') ENABLE,
        CONSTRAINT "N_CASE_CK1" CHECK (N_CASE >= 0 AND N_CASE < 5) ENABLE,
        CONSTRAINT "TABELLONE_CK" CHECK ( "ORDINE"  >= 0) ENABLE,
        CONSTRAINT "TABELLONE_PK" PRIMARY KEY ("PARTITA", "CASELLA", "ORDINE") ENABLE,
        CONSTRAINT "TABELLONE_FK2" FOREIGN KEY ("CASELLA")
         REFERENCES  "CASELLA" ("ID_CASELLA") ENABLE,
        CONSTRAINT "TABELLONE_FK3" FOREIGN KEY ("GIOCATORE")
         REFERENCES  "GIOCATORE" ("ID_GIOCATORE") ENABLE,
        CONSTRAINT "TABELLONE_FK" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Pesca_Impresvisto

```
CREATE TABLE  "PESCA_IMPREVISTO"
   (   "PARTITA" NUMBER,
       "ORDINE" NUMBER(1,0),
       "IMPREVISTO" NUMBER(38,0) NOT NULL ENABLE,
       "GIOCATORE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "ORDINE_CK" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "PESCA_IMPREVISTO_PK" PRIMARY KEY ("PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "IMPREVISTO_FK" FOREIGN KEY ("IMPREVISTO")
         REFERENCES  "IMPREVISTI" ("COD_I") ENABLE,
        CONSTRAINT "PESCA_IMPREVISTO_FK" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

Tabella Pesca_Probabilita

```
CREATE TABLE  "PESCA_PROBABILITA"
   (   "PARTITA" NUMBER,
       "ORDINE" NUMBER(1,0),
       "PROBABILITA" NUMBER(38,0) NOT NULL ENABLE,
       "GIOCATORE" NUMBER(38,0) NOT NULL ENABLE,
        CONSTRAINT "ORDINE_CK1" CHECK (ORDINE > 0) ENABLE,
        CONSTRAINT "PESCA_PROBABILITA_PK" PRIMARY KEY ("PARTITA", "ORDINE") ENABLE,
        CONSTRAINT "PESCA_PROBABILITA_FK3" FOREIGN KEY ("PROBABILITA")
         REFERENCES  "PROBABILITA" ("COD_P") ENABLE,
        CONSTRAINT "PESCA_PROBABILITA_FK" FOREIGN KEY ("PARTITA")
         REFERENCES  "PARTITA" ("ID_PARTITA") ON DELETE CASCADE ENABLE
   )
/
```

3.4 Implementazione Vincoli

Mostreremo l'implementazione dei vincoli ritenuti più importanti,evitando di inserire trigger il cui funzionamento sarebbe molto simile o addirittura uguale ad altri menzionati.

Trigger CK_CASE

Il seguente vincolo si occupa del monitoraggio degli inserimenti nella tabella tabellone.
I primi inserimenti,quelli con ordine uguale a zero dovrannno avere valori che soddisfano le condizioni di default. Per gli inserimenti successivi andremo a verificare :

1) nel caso dell'acquisto di case che il giocatore possegga tutti i territori dello stesso colore,che possegga abbastanza denaro per effettuare l'operazione ,che la banca possegga un numero sufficiente di case e che la costruzione avvenga in modo proporzionato su tutti i territori
2) nel caso della vendita di case andremo a controllare che la vendita non violi la distribuzione proporzionata delle case

in entrambe i casi verranno aggiornate a dovere le tabelle banca e storico_saldo

```
CREATE OR REPLACE TRIGGER  "CK_CASE" BEFORE INSERT OR UPDATE ON TABELLONE FOR EACH ROW
DECLARE
  N INT := 0;
  K INT := 0;
  CASA INT := 0;
  CONTO STORICO_SALDO%ROWTYPE;
  BANC BANCA%ROWTYPE;
  COL CONTRATTI.COLORE%TYPE;
BEGIN
    IF :NEW.ALBERGO = 'N' THEN

      IF :NEW.N_CASE <> 0 THEN

        SELECT COLORE INTO COL
        FROM CONTRATTI
        WHERE CASELLA = :NEW.CASELLA;

        SELECT COUNT(*) INTO N
        FROM COMPRATI CM JOIN CONTRATTI CN ON CM.CONTRATTO = CN.ID_CONTRATTO
        WHERE CM.ID_PARTITA = :NEW.PARTITA AND CM.ID_GIOCATORE = :NEW.GIOCATORE AND CN.CASELLA
= :NEW.CASELLA AND CM.ORDINE = (SELECT MAX(CM1.ORDINE)
                                FROM COMPRATI CM1 JOIN CONTRATTI CN1 ON CM1.CONTRATTO =
CN1.ID_CONTRATTO
                                WHERE CM1.ID_PARTITA = :NEW.PARTITA AND CN1.CASELLA =
:NEW.CASELLA);

        IF N = 0 THEN
          RAISE_APPLICATION_ERROR(-20014,'IL GIOCATORE INSERITO NON POSSIEDE IL CONTRATTO
SCELTO' );
        ELSE
          SELECT COUNT(*) INTO N
          FROM COMPRATI CM JOIN CONTRATTI CN ON CM.CONTRATTO = CN.ID_CONTRATTO
          WHERE CM.ID_PARTITA = :NEW.PARTITA AND CM.ID_GIOCATORE = :NEW.GIOCATORE AND
CN.COLORE = COL AND NOT EXISTS (SELECT *
                                FROM COMPRATI CM2
                                WHERE CM2.ID_PARTITA = :NEW.PARTITA AND CM2.CONTRATTO =
CM.CONTRATTO AND CM.ORDINE < CM2.ORDINE);

          SELECT COUNT(*) INTO K
          FROM CONTRATTI
          WHERE COLORE = COL;

          IF N <> K THEN
            RAISE_APPLICATION_ERROR(-20007, 'IL GIOCATORE NON POSSIEDE TUTTI I CONTRATTI DI
QUEL COLORE');
          ELSE
            SELECT N_CASE INTO N
            FROM TABELLONE
            WHERE PARTITA = :NEW.PARTITA AND CASELLA = :NEW.CASELLA
                  AND ORDINE = (SELECT MAX (ORDINE)
```

```
                                FROM TABELLONE
                                WHERE PARTITA = :NEW.PARTITA AND CASELLA = :NEW.CASELLA);
            IF :NEW.N_CASE > N THEN
              CASA := :NEW.N_CASE - N;
              SELECT B.N_CASE INTO K
              FROM BANCA B JOIN PARTITA P ON B.ID_BANCA = P.ID_BANCA
              WHERE P.ID_PARTITA = :NEW.PARTITA AND B.ORDINE =
                  (SELECT MAX(B1.ORDINE)
                   FROM BANCA B1 JOIN PARTITA P1 ON B1.ID_BANCA = P1.ID_BANCA
                   WHERE P1.ID_PARTITA = :NEW.PARTITA);
              IF CASA > K THEN
                RAISE_APPLICATION_ERROR(-20008, 'NUMERO DI CASE DISPONIBILI INSUFFICIENTE');
              ELSE
                SELECT MIN(T.N_CASE) INTO N
                FROM TABELLONE T JOIN CONTRATTI C ON T.CASELLA = C.CASELLA
                WHERE C.COLORE = COL AND T.PARTITA = :NEW.PARTITA AND T.ORDINE =
                  (SELECT MAX(T1.ORDINE)
                   FROM TABELLONE T1
                   WHERE T1.PARTITA = :NEW.PARTITA AND T1.CASELLA=T.CASELLA);
                N:=N+1;
                IF :NEW.N_CASE > N THEN
                  RAISE_APPLICATION_ERROR(-20009, 'LA COSTRUZIONE DELLE CASE DEVE AVVENIRE IN
MODO PROPORZIONATO SUI TERRENI DELLO STESSO COLORE');
                ELSE
                  SELECT COSTO_CASA * CASA INTO N
                  FROM CONTRATTI
                  WHERE CASELLA = :NEW.CASELLA;

                  SELECT * INTO CONTO
                  FROM STORICO SALDO
                  WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
                           AND ORDINE = (SELECT MAX(ORDINE)
                                         FROM STORICO_SALDO
                                         WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA =
:NEW.PARTITA);
                  IF CONTO.SALDO < N THEN
                    RAISE_APPLICATION_ERROR(-20010, 'SALDO INSUFFICIENTE');
                  ELSE
                    IF CONTO.ORDINE = :NEW.ORDINE THEN
                      UPDATE STORICO_SALDO
                        SET SALDO = CONTO.SALDO - N
                      WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA AND ORDINE =
:NEW.ORDINE;
                    ELSE
                      INSERT INTO STORICO_SALDO (GIOCATORE, PARTITA, ORDINE, SALDO)
                        VALUES(:NEW.GIOCATORE, :NEW.PARTITA, :NEW.ORDINE, CONTO.SALDO - N);
                    END IF;

                    SELECT B.ID_BANCA, B.N_CASE, B.N_ALBERGHI, B.ORDINE INTO BANC
                    FROM BANCA B JOIN PARTITA P ON B.ID_BANCA = P.ID_BANCA
                    WHERE P.ID_PARTITA = :NEW.PARTITA AND ORDINE =
                           (SELECT MAX(B1.ORDINE)
                            FROM BANCA B1 JOIN PARTITA P1 ON B1.ID_BANCA = P1.ID_BANCA
                            WHERE P1.ID_PARTITA = :NEW.PARTITA);
                    IF BANC.ORDINE = :NEW.ORDINE THEN
                      UPDATE BANCA
                        SET N_CASE = BANC.N_CASE - CASA
                      WHERE ID_BANCA = BANC.ID_BANCA AND ORDINE = :NEW.ORDINE;
                    ELSE
                      INSERT INTO BANCA (ID_BANCA , N_CASE , N_ALBERGHI, ORDINE )
                        VALUES (BANC.ID_BANCA , (BANC.N_CASE - CASA), BANC.N_ALBERGHI ,
:NEW.ORDINE);
                    END IF;
                  END IF;
                END IF;
              END IF;
            ELSE
              CASA := (N - :NEW.N_CASE);

              SELECT MAX(T.N_CASE) INTO N
              FROM TABELLONE T JOIN CONTRATTI C ON T.CASELLA = C.CASELLA
              WHERE C.COLORE = COL AND T.PARTITA = :NEW.PARTITA AND T.ORDINE =
                  (SELECT MAX(T1.ORDINE)
                   FROM TABELLONE T1
                   WHERE T1.PARTITA = :NEW.PARTITA AND T1.CASELLA=T.CASELLA);
```

```
              IF :NEW.N CASE < (N - 1) THEN
                RAISE_APPLICATION_ERROR(-20011, 'LA COSTRUZIONE DELLE CASE DEVE AVVENIRE IN
MODO PROPORZIONATO SUI TERRENI DELLO STESSO COLORE');
              ELSE
                SELECT (COSTO_CASA * CASA)/2 INTO N
                FROM CONTRATTI
                WHERE CASELLA = :NEW.CASELLA;

                SELECT * INTO CONTO
                FROM STORICO_SALDO
                WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
                  AND ORDINE = (SELECT MAX(ORDINE)
                                FROM STORICO_SALDO
                                WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA);

                IF CONTO.ORDINE = :NEW.ORDINE THEN
                  UPDATE STORICO_SALDO
                    SET SALDO = CONTO.SALDO + N
                  WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA AND ORDINE =
:NEW.ORDINE;
                ELSE
                  INSERT INTO STORICO_SALDO (GIOCATORE, PARTITA, ORDINE, SALDO)
                    VALUES(:NEW.GIOCATORE, :NEW.PARTITA, :NEW.ORDINE, CONTO.SALDO + N);
                END IF;
                SELECT B.ID_BANCA, B.N_CASE, B.N_ALBERGHI, B.ORDINE INTO BANC
                FROM BANCA B JOIN PARTITA P ON B.ID_BANCA = P.ID_BANCA
                WHERE P.ID_PARTITA = :NEW.PARTITA AND B.ORDINE =
                  (SELECT MAX(B1.ORDINE)
                   FROM BANCA B1 JOIN PARTITA P1 ON B1.ID_BANCA = P1.ID_BANCA
                   WHERE P1.ID PARTITA = :NEW.PARTITA);
                IF BANC.ORDINE = :NEW.ORDINE THEN
                  UPDATE BANCA
                    SET N_CASE = BANC.N_CASE + CASA
                  WHERE ID_BANCA = BANC.ID_BANCA AND ORDINE = :NEW.ORDINE;
                ELSE
                  INSERT INTO BANCA (ID_BANCA , N_CASE , N_ALBERGHI, ORDINE )
                    VALUES (BANC.ID_BANCA , (BANC.N_CASE + CASA), BANC.N_ALBERGHI ,
:NEW.ORDINE);
                END IF;
              END IF;
            END IF;
          END IF;
        END IF;
      END IF;
    END IF;

END;
/
ALTER TRIGGER  "CK_CASE" ENABLE
/
```

Trigger CK_FUORI_PRIGIONE

Il seguente trigger si occuperà di gestire l'uscita di prigione di un giocatore.in base al tipo di inserimento sapremo se il giocatore sta tentando di uscire utilizzando un cartellino "esci gratis di prigione " a sua disposizione o se tentato un tiro doppio o pagando la cauzione.nel caso in cui il giocatore abbia tentato senza successo per tre volte il tiro doppio,questo sarà obbligato a pagare la cauzione

```
CREATE OR REPLACE TRIGGER  "CK_FUORI_PRIGIONE" BEFORE INSERT ON MOSSA FOR EACH ROW
DECLARE
  X INT:=0;
  N INT:=0;
  CONTO STORICO_SALDO%ROWTYPE;
  TENT GIOCA PARTITA.TENTATIVI%TYPE;
BEGIN

SELECT TENTATIVI INTO TENT
  FROM GIOCA_PARTITA
  WHERE PARTITA = :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE ;
```

```
  IF TENT IS NULL THEN
    TENT := 0;
  END IF;

  SELECT COUNT(*) INTO N
  FROM PRIGIONE
  WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND SCARCERATO = 'N';

  SELECT COUNT(*) INTO X
  FROM BONUS_PRIGIONE
  WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND UTILIZZATO = 'N';

  IF N=0 AND :NEW.IMPRIGIONATO='Y' THEN
    RAISE_APPLICATION_ERROR(-20023,'IL GIOCATORE NON E IMPRIGIONATO');
  END IF;
  IF N > 0 THEN
    IF :NEW.IMPRIGIONATO = 'N' THEN
      RAISE_APPLICATION_ERROR(-20021,'IL GIOCATORE E ATTUALMENTE IMPRIGIONATO,CONTROLLA IL TUO
INSERIMENTO');
    ELSE
      IF :NEW.DOPPIO IS NULL THEN
        IF X > 0 THEN
          IF :NEW.IMPRIGIONATO = 'Y' THEN
            RAISE_APPLICATION_ERROR(-20022,'IL GIOCATORE POSSIEDE UN CARTELLINO PER USCIRE DI
PRIGIONE,IN IMPRIGIONATO NON DOVRA ESSERE INSERITO NESSUN VALORE');
          ELSE
            UPDATE BONUS_PRIGIONE
              SET UTILIZZATO = 'Y'
            WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND UTILIZZATO =
'N' AND ORDINE =(SELECT MIN(ORDINE)
                 FROM BONUS_PRIGIONE
                 WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND
UTILIZZATO = 'N');
            UPDATE PRIGIONE
              SET SCARCERATO = 'Y'
            WHERE PARTITA = :NEW.ID_PARTITA AND ORDINE =
                  (SELECT ORDINE
                   FROM PRIGIONE
                   WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND
SCARCERATO = 'N');
            UPDATE GIOCA_PARTITA
               SET TENTATIVI = 0
             WHERE PARTITA = :NEW.ID PARTITA AND GIOCATORE = :NEW.ID GIOCATORE;
          END IF;
        ELSE
          SELECT * INTO CONTO
          FROM STORICO_SALDO
          WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA
                AND ORDINE = (SELECT MAX(ORDINE)
                              FROM STORICO_SALDO
                              WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA =
:NEW.ID_PARTITA);
          SELECT COUNT(*) INTO N
          FROM COMPRATI C
          WHERE C.ID_GIOCATORE = :NEW.ID_GIOCATORE AND C.ID_PARTITA = :NEW.ID_PARTITA
                AND C.CONTRATTO NOT IN (SELECT I.CONTRATTO
                                        FROM IPOTECA I
                                        WHERE I.GIOCATORE = :NEW.ID_GIOCATORE AND I.PARTITA =
:NEW.ID_PARTITA AND I.ORDINE > C.ORDINE AND I.CONTRATTO = C.CONTRATTO) AND NOT EXISTS
         (SELECT *
          FROM COMPRATI C1
          WHERE C.CONTRATTO = C1.CONTRATTO AND C1.ID_PARTITA = :NEW.ID_PARTITA AND C.ORDINE >
C1.ORDINE);
          IF CONTO.SALDO < 125 AND N=0 THEN
            IF :NEW.IMPRIGIONATO IS NULL OR :NEW.IMPRIGIONATO = 'N' THEN
              RAISE_APPLICATION_ERROR(-20026,'IL GIOCATORE ANDRA IN BANCAROTTA,IMPRIGIONATO
DEVE ESSERE UGUALE A "Y"');
            ELSE
              INSERT INTO BANCAROTTA(PARTITA,GIOCATORE,ORDINE) VALUES
(CONTO.PARTITA,CONTO.GIOCATORE,:NEW.ORDINE);
            END IF;
          ELSIF CONTO.SALDO < 125 AND N>0 THEN
            RAISE_APPLICATION_ERROR(-20019,'SALDO INSUFFICIENTE,E NECESSARIO VENDERE QUALCHE
CASA O IPOTECARE QUALCHE CONTRATTO');
          ELSE
            IF :NEW.IMPRIGIONATO='Y' THEN
```

```
              RAISE APPLICATION ERROR(-20022,'IL GIOCATORE HA DECISO DI PAGARE LA CAUZIONE,IN
IMPRIGIONATO NON DOVRA ESSERE INSERITO NESSUN VALORE');
            ELSE
              IF CONTO.ORDINE = :NEW.ORDINE THEN
                UPDATE STORICO_SALDO
                  SET SALDO = SALDO - 125
                WHERE ORDINE = :NEW.ORDINE AND PARTITA= :NEW.ID_PARTITA AND GIOCATORE =
:NEW.ID_GIOCATORE;
              ELSE
                INSERT INTO STORICO_SALDO(GIOCATORE, PARTITA, SALDO, ORDINE) VALUES
(CONTO.GIOCATORE,CONTO.PARTITA,CONTO.SALDO - 125,:NEW.ORDINE);
              END IF;
              UPDATE PRIGIONE
                SET SCARCERATO = 'Y'
              WHERE PARTITA = :NEW.ID_PARTITA AND ORDINE =
                  (SELECT ORDINE
                   FROM PRIGIONE
                   WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA AND
SCARCERATO = 'N');
              UPDATE GIOCA_PARTITA
               SET TENTATIVI = 0
              WHERE PARTITA = :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE;
            END IF;
          END IF;
        END IF;
      ELSIF :NEW.DOPPIO = 'N' THEN
          IF TENT = 3 AND :NEW.IMPRIGIONATO = 'Y' THEN
            RAISE_APPLICATION_ERROR(-20022,'IL GIOCATORE HA EFFETTUATO IL NUMERO MASSIMO DI
TENTATIVI,IN IMPRIGIONATO NON DOVRA ESSERE INSERITO NESSUN VALORE');
          ELSIF TENT = 3 THEN
            SELECT * INTO CONTO
            FROM STORICO_SALDO
            WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA
                  AND ORDINE = (SELECT MAX(ORDINE)
                                FROM STORICO_SALDO
                                WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA =
:NEW.ID_PARTITA);
            SELECT COUNT(*) INTO N
            FROM COMPRATI C
            WHERE C.ID_GIOCATORE = :NEW.ID_GIOCATORE AND C.ID_PARTITA = :NEW.ID_PARTITA
                  AND C.CONTRATTO NOT IN (SELECT I.CONTRATTO
                                          FROM IPOTECA I
                                          WHERE I.GIOCATORE = :NEW.ID GIOCATORE AND I.PARTITA =
:NEW.ID_PARTITA AND I.ORDINE > C.ORDINE AND I.CONTRATTO = C.CONTRATTO) AND NOT EXISTS
           (SELECT *
            FROM COMPRATI C1
            WHERE C.CONTRATTO = C1.CONTRATTO AND C1.ID_PARTITA = :NEW.ID_PARTITA AND C.ORDINE >
C1.ORDINE);

            IF (CONTO.SALDO < 125 AND N=0) THEN
              IF :NEW.IMPRIGIONATO IS NULL OR :NEW.IMPRIGIONATO = 'N' THEN
                RAISE_APPLICATION_ERROR(-20026,'IL GIOCATORE ANDRA IN BANCAROTTA,IMPRIGIONATO
DEVE ESSERE UGUALE A "Y"');
              ELSE
                INSERT INTO BANCAROTTA(PARTITA,GIOCATORE,ORDINE) VALUES
(CONTO.PARTITA,CONTO.GIOCATORE,:NEW.ORDINE);
              END IF;
            ELSIF CONTO.SALDO < 125 AND N>0 THEN
              RAISE_APPLICATION_ERROR(-20021,'SALDO INSUFFICIENTE,E NECESSARIO VENDERE QUALCHE
CASA O IPOTECARE QUALCHE CONTRATTO');
            ELSE
              IF CONTO.ORDINE = :NEW.ORDINE THEN
                UPDATE STORICO_SALDO
                  SET SALDO = SALDO - 125
                WHERE ORDINE = :NEW.ORDINE AND PARTITA= :NEW.ID_PARTITA AND GIOCATORE =
:NEW.ID_GIOCATORE;
              ELSE
                INSERT INTO STORICO_SALDO(GIOCATORE, PARTITA, SALDO, ORDINE) VALUES
(CONTO.GIOCATORE,CONTO.PARTITA,CONTO.SALDO - 125,:NEW.ORDINE);
              END IF;
              UPDATE PRIGIONE
                SET SCARCERATO = 'Y'
              WHERE PARTITA = :NEW.ID_PARTITA AND ORDINE = (SELECT ORDINE
                                                            FROM PRIGIONE
                                                            WHERE GIOCATORE = :NEW.ID_GIOCATORE
AND PARTITA = :NEW.ID_PARTITA AND SCARCERATO = 'N');
```

```
            UPDATE GIOCA_PARTITA
              SET TENTATIVI = 0
            WHERE PARTITA = :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE;
          END IF;
        ELSIF :NEW.IMPRIGIONATO IS NULL THEN
            RAISE_APPLICATION_ERROR(-20026,'IL GIOCATORE NON HA EFFETTUATO UN DOPPIO TIRO E
DEVE RIMANERE IN PRIGIONE,IMPRIGIONATO DEVE ESSERE UGUALE A "Y"');
          ELSE
            UPDATE GIOCA_PARTITA
              SET TENTATIVI = (TENT + 1)
            WHERE PARTITA = :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE;
        END IF;

      ELSE
        IF :NEW.IMPRIGIONATO='Y'  THEN
          RAISE_APPLICATION_ERROR(-20025,'IL GIOCATORE DEVE USCIRE GRATUITAMENTE DI
PRIGIONE,NON DEVE ESSERE INSERITO NESSUN VALORE IN IMPRIGIONATO');
        ELSE
          UPDATE GIOCA_PARTITA
            SET TENTATIVI = 0
          WHERE PARTITA = :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE;
        END IF;
      END IF;
    END IF;
  END IF;
END;
/
ALTER TRIGGER  "CK_FUORI_PRIGIONE" ENABLE
/
```

Trigger BONUS_MALUS_IMPREVISTO

Il seguente trigger si occupa della gestione dei cartellini di imprevisto,in particolare di quelli che agiscono sul saldo dell'utente.

```
CREATE OR REPLACE TRIGGER  "BONUS_MALUS_IMPREVISTO" AFTER INSERT ON PESCA_IMPREVISTO FOR EACH
ROW
DECLARE
  N INT :=0;
  IMP IMPREVISTI%ROWTYPE;
  GIOC STORICO_SALDO%ROWTYPE;
BEGIN
  SELECT * INTO IMP
  FROM IMPREVISTI
  WHERE COD_I = :NEW.IMPREVISTO;

  IF IMP.BONUS IS NOT NULL AND IMP.BONUS < 0 THEN
    SELECT * INTO GIOC
    FROM STORICO_SALDO
    WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM STORICO_SALDO
                        WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA);
    IF GIOC.SALDO + IMP.BONUS > 0 THEN
      INSERIMENTO_ST_SALDO(:NEW.GIOCATORE,:NEW.PARTITA,:NEW.ORDINE,IMP.BONUS);
    ELSE
      SELECT COUNT(*) INTO N
      FROM COMPRATI C
      WHERE C.ID_GIOCATORE = :NEW.GIOCATORE AND C.ID_PARTITA = :NEW.PARTITA
            AND C.CONTRATTO NOT IN (SELECT I.CONTRATTO
                                    FROM IPOTECA I
                                    WHERE I.GIOCATORE = :NEW.GIOCATORE AND I.PARTITA =
:NEW.PARTITA AND I.ORDINE > C.ORDINE AND I.CONTRATTO = C.CONTRATTO) AND NOT EXISTS
       (SELECT *
        FROM COMPRATI C1
        WHERE C.CONTRATTO = C1.CONTRATTO AND C1.ID_PARTITA = :NEW.PARTITA AND C.ORDINE >
C1.ORDINE);
      IF N > 0 THEN
        RAISE_APPLICATION_ERROR(20051,'SALDO INSUFFICIENTE,BISOGNA IPOTECARE QUALCHE
CONTRATTO');
      ELSE
```

```
        INSERT INTO BANCAROTTA(PARTITA,GIOCATORE,ORDINE) VALUES
(:NEW.PARTITA,:NEW.GIOCATORE,:NEW.ORDINE);
      END IF;
    END IF;
  ELSIF IMP.BONUS IS NOT NULL AND IMP.BONUS > 0 THEN
    INSERIMENTO_ST_SALDO(:NEW.GIOCATORE,:NEW.PARTITA,:NEW.ORDINE,IMP.BONUS);
  END IF;
END;
/
ALTER TRIGGER  "BONUS_MALUS_IMPREVISTO" ENABLE
/
```

nel precedente trigger abbiamo fatto uso della funzione INSERIMENTO_ST_SALDO di cui adesso andremo a mostrare l'implementazione

```
CREATE OR REPLACE PROCEDURE INSERIMENTO_ST_SALDO (GIOC STORICO_SALDO.GIOCATORE%TYPE,PART
STORICO_SALDO.PARTITA%TYPE,ORD STORICO_SALDO.ORDINE%TYPE,X INT) IS

    CONTO STORICO_SALDO%ROWTYPE;

BEGIN
    SELECT * INTO CONTO
    FROM STORICO_SALDO
    WHERE GIOCATORE = GIOC AND PARTITA = PART
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM STORICO_SALDO
                        WHERE GIOCATORE = GIOC AND PARTITA = PART);
    IF CONTO.ORDINE = ORD THEN
         UPDATE STORICO_SALDO
             SET SALDO = SALDO + X
         WHERE ORDINE = ORD AND PARTITA = PART AND GIOCATORE = GIOC;
    ELSE
        INSERT INTO STORICO_SALDO(GIOCATORE, PARTITA, SALDO, ORDINE)
             VALUES (GIOC,PART,CONTO.SALDO + X,ORD);

    END IF;
END;
```

La precedente funzione gestirà opportunamente la tabella storico_saldo non violando nessun vincolo precedentemente definite su di essa.

Trigger CK_IPOTECA

Il seguente vincolo si occupa della corretta gestione delle ipoteche,il trigger controllerà che un contratto già ipotecato non venga re-ipotecato e controllerà anche che un ipoteca già estinta non venga ri-estinta

```
CREATE OR REPLACE TRIGGER  "CK_IPOTECA" BEFORE INSERT ON IPOTECA FOR EACH ROW
DECLARE
  N INT := 0;
BEGIN
  IF :NEW.ESTINTA = 'N' THEN
    SELECT COUNT(*) INTO N
    FROM IPOTECA
    WHERE PARTITA = :NEW.PARTITA AND ESTINTA = 'N' AND :NEW.CONTRATTO = CONTRATTO
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM IPOTECA
                        WHERE PARTITA = :NEW.PARTITA AND CONTRATTO = :NEW.CONTRATTO);
    IF(N > 0) THEN
      RAISE_APPLICATION_ERROR (-20005, 'CONTRATTO GIA IPOTECATO');
    END IF;
  ELSE
    SELECT COUNT(*) INTO N
    FROM IPOTECA
    WHERE PARTITA = :NEW.PARTITA AND ESTINTA = 'Y' AND :NEW.CONTRATTO = CONTRATTO
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM IPOTECA
                        WHERE PARTITA = :NEW.PARTITA AND CONTRATTO = :NEW.CONTRATTO);
    IF(N > 0) THEN
      RAISE_APPLICATION_ERROR (-20040, 'IPOTECA GIA ESTINTA');
    END IF;
  END IF;
```

```
END;
/
ALTER TRIGGER  "CK_IPOTECA" ENABLE
/
```

Trigger CK_ORDINE_MOSSA

Il seguente vincolo andrò a gestire la tabella mossa controllando venga rispettato l'ordine crescente per l'attributo "ordine"

```
CREATE OR REPLACE TRIGGER  "CK_ORDINE_MOSSA" BEFORE INSERT ON MOSSA FOR EACH ROW
DECLARE
  N INT := 0;
BEGIN
  SELECT COUNT(*) INTO N
      FROM MOSSA
      WHERE ID_PARTITA = :NEW.ID_PARTITA;
  IF :NEW.ORDINE <> (N + 1) THEN
      RAISE_APPLICATION_ERROR (-20002, 'ORDINE MOSSA ERRATO');
  END IF;
END;
/
ALTER TRIGGER  "CK_ORDINE_MOSSA" DISABLE
/
```

Il seguente trigger è stato successivamente disabilitato poiché si è preferito gestire l'ordine di ogni mossa direttamente da client e non più da server.

Trigger CK_PAGA_PEDAGGIO

Il seguente trigger si occupa della corretta gestione dei pedaggi,ogni qual volta un giocatore si ferma su un territorio già posseduto il trigger elabora il costo del pedaggio in base alla quantità di strutture costruite su di esso,e sulla quantità di contratti dello stesso colore posseduti dall'utente

```
CREATE OR REPLACE TRIGGER  "CK_PAGA_PEDAGGIO" AFTER INSERT ON MOSSA FOR EACH ROW
DECLARE
  N INT := 0;
  X INT :=0;
  K INT :=0;
  W INT :=0;
  ALBERGO INT :=0;
  COSTO INT :=0;
  SALDO_PROP STORICO_SALDO%ROWTYPE;
  SALDO_GIOC STORICO_SALDO%ROWTYPE;
  CASA TABELLONE.N_CASE%TYPE;
  PROP COMPRATI.ID_GIOCATORE%TYPE;
  COL CONTRATTI.COLORE%TYPE;
BEGIN

  SELECT COUNT(*) INTO N
  FROM COMPRATI CM JOIN CASELLA CA ON CA.CONTRATTO = CM.CONTRATTO
  WHERE CM.ID_PARTITA = :NEW.ID_PARTITA AND CA.ID_CASELLA = :NEW.CAS_A
        AND CM.ORDINE = (SELECT MAX(CM2.ORDINE)
                         FROM COMPRATI CM2
                         WHERE CM2.ID_PARTITA = :NEW.ID_PARTITA AND CM2.CONTRATTO =
CM.CONTRATTO)
        AND NOT EXISTS(SELECT *
                       FROM IPOTECA I
                       WHERE I.PARTITA = :NEW.ID_PARTITA AND I.CONTRATTO = CA. CONTRATTO
                             AND ORDINE > CM.ORDINE);

  IF N>0 THEN

    SELECT COLORE INTO COL
    FROM CONTRATTI
    WHERE CASELLA = :NEW.CAS_A;

    SELECT CM.ID_GIOCATORE INTO PROP
    FROM COMPRATI CM JOIN CASELLA CA ON CA.CONTRATTO = CM.CONTRATTO
    WHERE CM.ID_PARTITA = :NEW.ID_PARTITA AND CA.ID_CASELLA = :NEW.CAS_A
          AND CM.ORDINE = (SELECT MAX(CM2.ORDINE)
```

```
                          FROM COMPRATI CM2
                          WHERE CM2.ID_PARTITA = :NEW.ID_PARTITA AND CM2.CONTRATTO =
CM.CONTRATTO);

    SELECT COUNT(*) INTO X
    FROM COMPRATI CM JOIN CONTRATTI CN ON CM.CONTRATTO = CN.ID_CONTRATTO
    WHERE CM.ID_PARTITA = :NEW.ID_PARTITA AND CM.ID_GIOCATORE = PROP AND CN.COLORE = COL
          AND NOT EXISTS (SELECT *
                          FROM COMPRATI CM2
                          WHERE CM2.ID_PARTITA = :NEW.ID_PARTITA AND CM2.CONTRATTO =
CM.CONTRATTO AND CM.ORDINE < CM2.ORDINE);

    SELECT COUNT(*) INTO K
    FROM CONTRATTI
    WHERE COLORE = COL;



    SELECT N_CASE INTO CASA
    FROM TABELLONE
    WHERE CASELLA = :NEW.CAS_A AND PARTITA = :NEW.ID_PARTITA
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM TABELLONE
                        WHERE CASELLA = :NEW.CAS_A AND PARTITA = :NEW.ID_PARTITA);

    SELECT COUNT(*) INTO ALBERGO
    FROM TABELLONE
    WHERE CASELLA = :NEW.CAS_A AND PARTITA = :NEW.ID_PARTITA AND ALBERGO = 'Y'
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM TABELLONE
                        WHERE CASELLA = :NEW.CAS_A AND PARTITA = :NEW.ID_PARTITA);

    IF CASA IS NULL THEN
      CASA := 0;
    ELSIF ALBERGO > 0 THEN
      CASA := 5;
    END IF;

    SELECT * INTO SALDO_PROP
    FROM STORICO_SALDO
    WHERE GIOCATORE = PROP AND PARTITA = :NEW.ID_PARTITA
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM STORICO_SALDO
                        WHERE GIOCATORE = PROP AND PARTITA = :NEW.ID_PARTITA);

    SELECT * INTO SALDO_GIOC
    FROM STORICO_SALDO
    WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA
          AND ORDINE = (SELECT MAX(ORDINE)
                        FROM STORICO_SALDO
                        WHERE GIOCATORE = :NEW.ID_GIOCATORE AND PARTITA = :NEW.ID_PARTITA);

    SELECT COUNT(*) INTO W
    FROM COMPRATI C
    WHERE C.ID_GIOCATORE = :NEW.ID_GIOCATORE AND C.ID_PARTITA = :NEW.ID_PARTITA
          AND C.CONTRATTO NOT IN (SELECT I.CONTRATTO
                                  FROM IPOTECA I
                                  WHERE I.GIOCATORE = :NEW.ID_GIOCATORE AND I.PARTITA =
:NEW.ID_PARTITA AND I.ORDINE > C.ORDINE AND I.CONTRATTO = C.CONTRATTO)
          AND NOT EXISTS (SELECT *
                          FROM COMPRATI C1
                          WHERE C.CONTRATTO = C1.CONTRATTO AND C1.ID_PARTITA = :NEW.ID_PARTITA
AND C.ORDINE > C1.ORDINE);

    IF X <> K THEN
      INSERIMENTO_ST_SALDO(:NEW.ID_GIOCATORE,:NEW.ID_PARTITA,:NEW.ORDINE,-
COSTO_PEDAGGIO(:NEW.CAS_A,0));

INSERIMENTO_ST_SALDO(SALDO_PROP.GIOCATORE,SALDO_PROP.PARTITA,:NEW.ORDINE,COSTO_PEDAGGIO(:NEW.C
AS_A,0));
    ELSIF X = K THEN

      IF CASA = 0 THEN
        COSTO := 2 * COSTO_PEDAGGIO(:NEW.CAS_A,CASA);
        IF SALDO_GIOC.SALDO - COSTO >=0 THEN
```

```
          INSERIMENTO ST SALDO(:NEW.ID GIOCATORE,:NEW.ID PARTITA,:NEW.ORDINE,-COSTO);
          INSERIMENTO_ST_SALDO(SALDO_PROP.GIOCATORE,SALDO_PROP.PARTITA,:NEW.ORDINE,COSTO);
        ELSE
          IF W > 0 THEN
            RAISE_APPLICATION_ERROR(-20050,'SALDO INSUFFICIENTE,BISOGNA IPOTECARE QUALCHE
CONTRATTO');
          ELSE
            INSERT INTO BANCAROTTA(PARTITA,GIOCATORE,ORDINE) VALUES
(SALDO_GIOC.PARTITA,SALDO_GIOC.GIOCATORE,:NEW.ORDINE);
          END IF;
        END IF;
      ELSE
        COSTO :=COSTO_PEDAGGIO(:NEW.CAS_A,CASA);
        IF (SALDO_GIOC.SALDO - COSTO) >=0 THEN
          INSERIMENTO_ST_SALDO(:NEW.ID_GIOCATORE,:NEW.ID_PARTITA,:NEW.ORDINE,- COSTO);
          INSERIMENTO_ST_SALDO(SALDO_PROP.GIOCATORE,SALDO_PROP.PARTITA,:NEW.ORDINE,COSTO);
        ELSE
          IF W > 0 THEN
            RAISE_APPLICATION_ERROR(-20050,'SALDO INSUFFICIENTE,BISOGNA IPOTECARE QUALCHE
CONTRATTO');
          ELSE
            INSERT INTO BANCAROTTA(PARTITA,GIOCATORE,ORDINE) VALUES
(SALDO_GIOC.PARTITA,SALDO_GIOC.GIOCATORE,:NEW.ORDINE);
          END IF;
        END IF;
      END IF;
    END IF;
  END IF;
END;
/
ALTER TRIGGER  "CK_PAGA_PEDAGGIO" ENABLE
/
```

Nel precedente trigger abbiamo fatto usa della funzione INSERIMENTO_ST_SALDO precedentemente implementata e della funzione COSTO_PEDAGGIO che andremo ad implementare

```
CREATE OR REPLACE FUNCTION COSTO_PEDAGGIO (CAS MOSSA.CAS_A%TYPE,CASA TABELLONE.N_CASE%TYPE)
RETURN INT IS
    CONTR CONTRATTI%ROWTYPE;
BEGIN
    SELECT * INTO CONTR
    FROM CONTRATTI
    WHERE CASELLA = CAS;
    IF CASA = 0 THEN
        RETURN CONTR.PED_0_CASE;
    ELSIF CASA = 1 THEN
        RETURN CONTR.PED_1_CASE;
    ELSIF CASA = 2 THEN
        RETURN CONTR.PED_2_CASE;
    ELSIF CASA = 3 THEN
        RETURN CONTR.PED_3_CASE;
    ELSIF CASA = 4 THEN
        RETURN CONTR.PED_4_CASE;
    ELSIF CASA = 5 THEN
        RETURN CONTR.PED_ALBERGO;
    END IF;
END;
```

Trigger CK_PASSA_DAL_VIA

Il seguente trigger verifica che ogni utente,dopo esser passato dal via riceva,come da regolamento,un bonus di 500 euro

```
CREATE OR REPLACE TRIGGER  "CK_PASSA_DAL_VIA" BEFORE INSERT ON MOSSA FOR EACH ROW
DECLARE
  CONTO STORICO_SALDO%ROWTYPE;
BEGIN
  SELECT * INTO CONTO
```

```
  FROM STORICO SALDO
  WHERE PARTITA= :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE
    AND ORDINE = (SELECT MAX(ORDINE)
                  FROM STORICO_SALDO
                  WHERE PARTITA= :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE );
  IF :NEW.CAS_P >= 28 AND :NEW.CAS_A >=0 THEN
    IF CONTO.ORDINE = :NEW.ORDINE THEN
      UPDATE STORICO_SALDO
        SET SALDO = SALDO + 500
      WHERE PARTITA= :NEW.ID_PARTITA AND GIOCATORE = :NEW.ID_GIOCATORE AND ORDINE =
:NEW.ORDINE;
    ELSE
      INSERT INTO STORICO_SALDO (GIOCATORE,PARTITA,ORDINE,SALDO) VALUES
(:NEW.ID_GIOCATORE,:NEW.ID_PARTITA,:NEW.ORDINE,CONTO.SALDO + 500);
    END IF;
  END IF;
END;
/
ALTER TRIGGER  "CK_PASSA_DAL_VIA" ENABLE
/
```

Trigger CK_SPOSTAMENTO

Il seguente trigger si occupa del corretto inserimento nella casella mossa,considerando tutti i possibili spostamenti permessi dal gioco

```
CREATE OR REPLACE TRIGGER  "CK_SPOSTAMENTO" BEFORE INSERT ON MOSSA FOR EACH ROW
BEGIN
  IF :NEW.CAS_A > :NEW.CAS_P AND :NEW.CAS_A - :NEW.CAS_P > 12 THEN
      RAISE_APPLICATION_ERROR(-20033, 'SPOSTAMENTO NON CONSENTITO');
  ELSIF :NEW.CAS_A < :NEW.CAS_P AND (:NEW.CAS_P < 28 OR :NEW.CAS_A > MOD(:NEW.CAS_P + 12, 40))
THEN
    RAISE_APPLICATION_ERROR(-20034, 'SPOSTAMENTO NON CONSENTITO');
  ELSIF :NEW.CAS_A = :NEW.CAS_P THEN
    RAISE_APPLICATION_ERROR(-20035, 'SPOSTAMENTO NON CONSENTITO');
  END IF;
END;
/
ALTER TRIGGER  "CK_SPOSTAMENTO" ENABLE
/
```

Trigger RISCATTO_IPOTECA

Il seguente trigger si occupa della gestione delle ipoteche facendo in modo che al giocatore che riscatta un ipoteca venga fatta pagare la giusta cifra

```
CREATE OR REPLACE TRIGGER  "RISCATTO_IPOTECA" AFTER INSERT ON IPOTECA FOR EACH ROW
DECLARE
  N INT :=0;
  CONTO STORICO_SALDO%ROWTYPE;
BEGIN
  SELECT * INTO CONTO
  FROM STORICO_SALDO
  WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
    AND ORDINE = ( SELECT MAX(ORDINE)
                   FROM STORICO_SALDO
                   WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA);
  SELECT COSTO TERRENO INTO N
  FROM CONTRATTI
  WHERE ID_CONTRATTO = :NEW.CONTRATTO;
  IF :NEW.ESTINTA = 'N' THEN
    IF CONTO.ORDINE = :NEW.ORDINE THEN
      UPDATE STORICO_SALDO
        SET SALDO = CONTO.SALDO + (N/2)
      WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
        AND ORDINE = ( SELECT MAX(ORDINE)
                       FROM STORICO_SALDO
                       WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA);
    ELSE
      INSERT INTO STORICO_SALDO (GIOCATORE,PARTITA,ORDINE,SALDO)
        VALUES  (:NEW.GIOCATORE,:NEW.PARTITA,:NEW.ORDINE,CONTO.SALDO + (N/2));
    END IF;
```

```
  ELSE
    IF CONTO.ORDINE = :NEW.ORDINE THEN
      UPDATE STORICO_SALDO
        SET SALDO = CONTO.SALDO - ((N/2)*1.10)
      WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA
        AND ORDINE = ( SELECT MAX(ORDINE)
                       FROM STORICO_SALDO
                       WHERE GIOCATORE = :NEW.GIOCATORE AND PARTITA = :NEW.PARTITA);
    ELSE
      INSERT INTO STORICO_SALDO (GIOCATORE,PARTITA,ORDINE,SALDO)
        VALUES  (:NEW.GIOCATORE,:NEW.PARTITA,:NEW.ORDINE,CONTO.SALDO - ((N/2)*1.10));
    END IF;
  END IF;
END;
/
ALTER TRIGGER  "RISCATTO_IPOTECA" ENABLE
/
```

Trigger SPOSTAMENTO_IMPREVISTO

Il seguente trigger si occupa della gestione dei cartellini di imprevisto,il cui effetto sposta la pedina del giocatore dalla sua posizione originaria

```
CREATE OR REPLACE TRIGGER  "SPOSTAMENTO_IMPREVISTO" AFTER INSERT ON PESCA_IMPREVISTO FOR EACH
ROW
DECLARE
  IMP IMPREVISTI%ROWTYPE;
BEGIN
  SELECT * INTO IMP
  FROM IMPREVISTI
  WHERE COD_I = :NEW.IMPREVISTO;

  IF IMP.CAS_A IS NOT NULL THEN
    UPDATE MOSSA
      SET CAS_A = IMP.CAS_A
    WHERE ID_PARTITA = :NEW.PARTITA AND ID_GIOCATORE = :NEW.GIOCATORE AND ORDINE =
:NEW.ORDINE;
  END IF;
END;
/
ALTER TRIGGER  "SPOSTAMENTO_IMPREVISTO" ENABLE
/
```

4 Esempi D'uso

Tabela PARTITA

ID_PARTITA	ID_BANCA
22	33
55	99

Tabella GIOCATORE

ID_GIOCATORE
123
456

Tabella CASELLA

ID_CASELLA	IMPREVISTO	PROBABILITA	CONTRATTO
7	**Y**	**N**	NULL
30	**N**	**N**	NULL
2	**N**	**Y**	NULL
13	**N**	**N**	13

Tabella BANCA

ID_BANCA	N_CASE	N_ALBERGHI	ORDINE
33	32	18	0
99	32	18	0
33	29	18	19

Tabella IMPREVISTI

COD_I	DESCRIZIONE	BONUS	CAS_A	TARIFFA_A	TARIFFA_C
10	PAGA 50€	-50	NULL	NULL	NULL
6	ARRIVA A LARGO COLOMBO	NULL	13	NULL	NULL

Tabella PROBABILITA

COD_P	DESCRIZIONE	BONUS	CAS_A	TARIFFA_A	TARIFFA_C
10	PAGA 10€ PER OGNI CASA E 20€ PER OGNI ALBERGO	NULL	NULL	20	10
6	HAI VINTO 20€ ALLA LOTTERIA	20	NULL	NULL	NULL

Tabella CONTRATTI

ID_CONTRATTO	CASELLA	DESCRIZIONE	COSTO_CASA	COSTO_ALBERGO	PED_0_CASE	PED_1_CASE
111	13	LARGO COLOMBO	20	30	10	20

PED_2_CASE	PED_3_CASE	PED_4_CASE	PED_A	COLORE	COSTO_TERRENO	EDIFICABILE
30	40	50	60	BLU	50	Y

Tabella STORICO_SALDO

GIOCATORE	PARTITA	ORDINE	SALDO
123	22	0	1000
123	22	1	984

Tabella PRIGIONE

PARTITA	ORDINE	SCARCERATO
22	5	N

Tabella BANCAROTTA

PARTITA	ORDINE	GIOCATORE
55	1	456

Tabella BONUS_PRIGIONE

GIOCATORE	PARTITA	ORDINE	UTILIZZATO
123	22	1	N

Tabella MOSSA

ID_PARTITA	ID_GIOCATORE	ORDINE	CAS_A	CAS_P	DOPPIO	IMPRIGIONATO
22	123	1	11	0	N	NULL
22	456	2	10	0	N	NULL
22	123	3	22	11	N	NULL
22	456	4	13	10	N	NULL
22	123	5	10	22	N	NULL

Tabella COMPRATI

ID_GIOCATORE	CONTRATTO	ID_PARTITA	ORDINE
123	11	22	1
456	13	22	4

Tabella IPOTECA

GIOCATORE	PARTITA	CONTRATTO	ORDINE	ESTINTA
456	22	13	6	N

Tabella GIOCA_PARTITA

GIOCATORE	PARTITA	PEDINA	TENTATIVI
123	22	ZUCCA	0
456	22	FUNGO	0

Tabella TABELLONE

PARTITA	CASELLA	N_CASE	ALBERGO	ORDINE	GIOCATORE
22	13	0	N	0	NULL
22	13	1	N	2	456

Tabella PESCA_PROBABILITA

PARTITA	ORDINE	PROBABILITA	GIOCATORE
22	8	6	123

Tabella PESCA_IMPREVISTO

PARTITA	ORDINE	IMPREVISTO	GIOCATORE
22	9	10	456